Intelligent Vehicle Systems: A 4D/RCS Approach

Editors
R. Madhavan, E.R. Messina, and **J.S. Albus**

Nova Science Publishers, Inc.

Table of Contents

Preface

The autonomous driving capabilities of unmanned ground vehicles are advancing more rapidly than most people – including many experts in the field – are aware. A fundamental understanding of how to integrate perception, world modeling, knowledge representation, task decomposition, planning, and control for autonomous ground vehicles is emerging. LADAR technology has developed to the point where it is possible to construct geometric and dynamic models of the world that can support competent real-time driving behavior. The computing power required to support high performance mobility is becoming available. The U.S. military services are investing large sums of money in autonomous mobility research for autonomous ground, air, undersea and surface vehicle systems. Commercial automotive companies throughout the world are also making large investments in advanced cruise control, lane departure warning, and collision mitigation systems. It seems likely that these investments will pay off in terms of significant advancements in the next two decades. In the military domain, useful on-road and off-road autonomous driving capabilities may be achieved by 2010. Human level performance in autonomous driving may be feasible by 2020. In the commercial sector, intelligent driver assistance systems may produce major reductions in automotive traffic injuries and fatalities within next 15 years.

Research in autonomous mobility began in the late 1960s. The first serious attempts at building intelligent mobile robots took place at the Stanford Research Institute (SRI) with the "Shakey" robot project [1] and at the Stanford University AI Lab with a robot cart [2]. The SRI project developed the theoretical foundations of task planning and execution, and the Stanford cart demonstrated some primitive elements of computer vision for navigation. These projects provided early insight into the type of algorithms that are required and the surprising amount of computational power needed to achieve even the most rudimentary forms of autonomous mobility.

The Autonomous Land Vehicle (ALV) program sponsored by DARPA during the 1980's was the first major research program to address the problem of autonomous driving in an outdoor environment. The ALV program assembled a team of the nation's most capable researchers, and provided them with state-of-the-art sensing and computing resources. Yet, the ALV was only partially successful in following paved roads at disappointingly slow speeds. The ALV program showed clearly that autonomous driving would be neither easy nor cheap. In fact, many experts concluded that autonomous driving was an unrealistic goal, and the Army should rely on teleoperation for unmanned systems. The rationale was that this approach had proven successful for control of the Unmanned Air Vehicle systems for many years.

Thus, between 1989 and 1991, when the Army Research Lab sponsored Demo I (Shoemaker Chapter 10), the first of three major Department of Defense programs designed to advance the state-of-the-art of robotic driving, the focus was on teleoperation, with some minor enhancements such as "Retro-Traverse," and driving paths selected by an operator indicating waypoints on a TV screen. Demo I results revealed several serious problems with the teleoperation strategy. The first and most important was the communication bandwidth problem. Teleoperation requires communicating real-time TV images from the remote vehicle to a human operator. A variety of compression algorithms were tried, but the basic problem is that communications on the battlefield between ground vehicles are unreliable, and often non-existent. Furthermore, teleoperation places a heavy workload on the operator, and often induces nausea when the operator is in a moving vehicle.

Thus, during the mid 1990's, DARPA and the Army cosponsored the Demo II program that once again focused on autonomy (Shoemaker Chapter 10). The Demo II vehicles were Army HMMWVs heavily loaded with computers and sensors. Monocular video was used for road following, and stereo vision was used for detecting and avoiding obstacles. The Demo II program demonstrated the ability of autonomous vehicles to navigate off-road at a few kilometers per hour, in environments consisting of relatively flat desert terrain sparsely populated with large obstacles. However, stereo has proven problematic in producing reliable high-resolution range information needed for detecting small obstacles, seeing through sparse vegetation, or segmenting distant objects from the background.

Beginning in 1998, the Army Research Lab sponsored the Demo III eXperimental Unmanned Vehicle (XUV) program (Shoemaker Chapter 10). The Demo III XUVs were built specifically for autonomous mobility research and development. They demonstrated the ability to reliably drive at average speeds between 5 and 7 km/h on trails through the woods and across rolling fields of tall weeds and underbrush. They were able to follow roads and to navigate through urban streets cluttered with rubble and obstacles such as parked cars and phone poles. A desired path could be provided to the XUVs in the form of widely spaced (more than 50 m apart) geo-referenced waypoints specified by an operator. LADAR was used to build an internal 3D model of the nearby terrain for planning safe paths through complex environments in real time. The XUV mobility subsystem recomputed its local map and local path plans 10 times per second. Multi-resolutional maps were used to support multi-resolutional planning. The Demo III program also demonstrated the ability of a single operator at a remote computer terminal to simultaneously supervise up to four XUVs driving autonomously.

Recent experiments with XUVs, Stryker vehicles, autonomous helicopters, and teleoperated hand-launched aircraft have shown the ability of a single operator to supervise collaborative operations between multiple unmanned ground vehicles and unmanned air vehicles. XUVs have successfully traveled distances of more than 40 km without human assistance over dirt roads and trails, through woods and fields, with a variety of vegetation.

DARPA has also continued to fund autonomous driving research. Recent programs related to autonomous vehicles include PerceptOR, Unmanned Ground Combat Vehicle (UGCV), Software for Distributed Robots (SDR), Mobile Autonomous Robotic Software (MARS), Learning Applied to Ground Robots (LAGR), and the DARPA Grand Challenge. The PerceptOR program subjected a variety of sensors and perception algorithms to a rigorous program of testing and evaluation. The UGCV program developed a vehicle with articulated suspension designed for high mobility in difficult terrain. The MARS program funded studies of the technical requirements for autonomous on-road driving in normal traffic. The LAGR program is investigating how autonomous driving behaviors can be improved by learning from experience. The DARPA Grand Challenge offered a $2 million prize to the team that builds an autonomous vehicle that achieves the fastest time over a 135 mile course through the Nevada desert. The Grand Challenge was won in October 2005 by a Stanford University team headed by Sebastian Thrun using a vehicle specially modified for autonomous driving by the Volkswagen Corporation[1].

Current programs sponsored by the Army Research Laboratory include the Robotic Collaborative Technology Alliance (RCTA) [**Error! Reference source not found.**]. The RCTA funds a number of universities and companies to work with General Dynamics Robotic Systems as a system integrator to develop and demonstrate the latest autonomous driving technology. ARL also supports autonomous mobility and tactical behaviors research at the National Institute of Standards and Technology. The work at NIST is focused on how to enable autonomous vehicles to perform tactical behaviors that require driving safely at tactical speeds on-roads in highway and city traffic in the presence of pedestrians.

Technology developed under the ARL research program is being transitioned to military vehicles under Tank Automotive Research, Development, and Engineering Center (TARDEC) programs in Vetronics Technology Integration, Crew Automation Testbed, and Road Following. Autonomous driving technology developed by ARL and TARDEC is also being transitioned through the Mounted Maneuver Battle Lab at Fort Knox into the Army Future Combat System. General Dynamics Robotic Systems is building an Autonomous Navigation System that will enable Future Combat Systems vehicles to drive autonomously during tactical maneuvers that involve collaboration between manned and unmanned vehicles, both on the ground and in the air.

In addition, there are a number of well funded autonomous air and undersea vehicle programs that are addressing issues of autonomous mobility in other environments. These include the Joint Unmanned Combat Air System, the RotorCraft Pilot's Associate, the Global Hawk, and several others. These programs are sponsored by DARPA, Office of Naval Research, and the Air Force Research Laboratories.

In the civilian sector, during the 1990s, the U.S. Department of Transportation (DOT) sponsored a series of experiments and demonstrations of autonomous highway driving. As part of this effort, a team from

Carnegie Mellon University demonstrated the ability to drive autonomously over 90% of the distance from Pittsburgh to Los Angeles [4]. Around 2000, the National Traffic Safety Administration of DOT sponsored studies on using LADAR to detect pedestrians and prevent accidents involving buses and pedestrians [5]. Current programs funded by DOT at NIST and elsewhere are focused on evaluating the effectiveness and reliability of driver warning systems for lane departure and collision prediction [6]. Overseas, during the early 1990s, a team headed by Prof. Dickmanns at Universitat der Bundeswehr, in Munich, Germany demonstrated the ability to drive autonomously on the autobahn in traffic at speeds of 100 km/h [7]. More recently, autonomous vehicles using similar technology have achieved speeds of 180 km/h on the autobahn [8]. Over the last decade, the German Ministry of Defense sponsored a series of demonstrations of off-road driving similar to the U.S. Army Demo III program, and the European Union has funded auto manufacturers to aggressively pursue technologies for semi-autonomous driving systems. Similar government and industry programs in Japan have demonstrated a variety of semi-autonomous driving skills, such as collision warning, collision mitigation, lane departure warning, adaptive cruise control, stop-and-go automatic driving in congested traffic, and automatic reverse parallel parking [9].

As a result of these programs, practical applications in both civilian and military sectors have begun to emerge. In the United States, military interest in unmanned vehicles (air, ground, undersea, and on the surface) has grown rapidly as increasingly sophisticated autonomous vehicle capabilities have been demonstrated [10]. In Japan, Europe, and the U.S., automotive companies are actively pursuing commercial applications of intelligent driver assistance technology [9].

The current rate of technological progress suggests that on-road vehicles will soon have the capability to autonomously follow commercial trip planner type directions (e.g. MapQuest®) to distant locations, while obeying rules of the road, negotiating intersections, and avoiding collisions with other vehicles and pedestrians along the way. Within a decade or two, it may be possible to purchase automobiles and trucks equipped with intelligent automotive autopilots –robot chauffeurs.

Overview and Organization of the Book

The purpose of this book is twofold:

1. To put into perspective what autonomous mobility capabilities are available now, and what advances can be anticipated in the coming two decades.
2. To describe the theoretical foundations and engineering approaches that enable these capabilities.

Chapter 1 provides a brief introduction to the 4D/RCS reference model architecture and design methodology that has proven successful in guiding the development of autonomous mobility systems. Chapters 2 through 7 provide more detailed descriptions of research that has been conducted and algorithms that have been developed to implement the various aspects of the 4D/RCS reference model architecture and design methodology. Chapters 8 and 9 discuss applications, performance measures, and standards. Chapter 10 provides a history of Army and DARPA research in autonomous ground mobility. Chapter 11 provides a perspective on the potential future developments in autonomous mobility.

Intended Audience

To be filled in

Acknowledgments

We would like to thank the reviewers of the book Prof. Kevin Passino of the Ohio State University and Mrs. Mylene Ouimette of NIST. Their comments were invaluable in getting it to its current form. We would also like to thank our sponsors, ARL, DARPA, DOT, DHS, and NIJ for their funding provided on the research reported in the chapters of this book. *More to be filled in …*

Commercial Product Disclaimer

Commercial equipment and materials are identified in this paper in order to adequately specify certain procedures. Such identification does not imply recommendation or endorsement by the National Institute of Standards and Technology (NIST), nor does it imply that the materials or equipment identified are necessarily the best available for the purpose.

References

1. Fikes, R., Hart, P., Nilsson, N. "Learning and Executing Generalized Robot Plans", Artificial Intelligence, Vol. 3, No. 1-4, 1972.
2. Moravec, H. "Towards Automatic Visual Obstacle Avoidance", Proceedings of the Fifth International Conference on Artificial Intelligence, Cambridge, MA, pp. 584, August, 1977.
3. (RCTA Annual Report) Proceedings of Collaborative Technology Alliances Conference – Robotics, April 29-May 1, 2003, College Park, MD, Requests for this document to U.S. Army Research Laboratory, ATTN: AMSRL-WM-RP (Mr. Charles Shoemaker), 2300 Powder Mill Road, Adelphi, MD 20783.
4. No Hands Across America, http://www.cs.cmu.edu/afs/cs/usr/tjochem/www/nhaa/nhaa_home_page.html
5. Thorpe, C., Duggins, D., McNeil, S., Mertz, C. "Side Collision Warning System (SCWS) Performance Specifications for a Transit Bus", Final Report, Federal Transit Administration under PennDOT Agreement Number 62N111, Project TA-34, May 2002.
6. Szabo, S., Murphy, K.N., Juberts, M. "The AUTONAV/DOT Project: Baseline Measurement System for Evaluation of Roadway Departure Warning System", NISTIR 6300, National Institute of Standards and Technology, Gaithersburg, MD, March 1999.
7. Dickmanns, E. D. *et al.* "The Seeing Passenger Car 'VaMoRs-P'", Proceedings of the International Symposium on Intelligent Vehicles, pp. 24-26, 1994.
8. Dickmanns, E. "An Expectation-based, Multi-focal, Saccadic (EMS) Vision System for Vehicle Guidance", Proceedings of the International Symposium on Robotics Research (ISRR), Salt Lake City, October 1999.
9. Bishop, R. "Presentation to a NIST/DARPA Workshop on Autonomous Driving", DARPA Mobile Autonomous Robot Software Workshop, 2003.
10. Shoemaker, C., Bornstein, J., Myers, S., and Brendle, B. "Demo III: Department of Defense Testbed for Unmanned Ground Mobility", SPIE Conference on Unmanned Ground Vehicle Technology, SPIE Vol. 3693, April 1999.

Intelligent Vehicle Systems: A 4D/RCS Approach
Abbreviations/Acronyms

AI	Artificial Intelligence
AM	Autonomous Mobility (RCS node)
ARL	Army Research Laboratory
BG	Behavior Generation
C3	Command, Control, and Communications
C4ISR	Command, Control, Communications, Computers, Intelligence, Surveillance, and Reconnaissance
CCD	Charge Coupled Device
DHS	Department of Homeland Security
EX	Executor
FCS	Future Combat Systems
FLIR	Forward Looking Infrared Imaging
FSM	Finite State Machine
GPS	Global Positioning System
HMMWV	High Mobility Multipurpose Wheeled Vehicle
h	hour
INS	Inertial Navigation System
ISD	NIST Intelligent Systems Division
JA	Job Assignor
KD	Knowledge Database
LADAR	LAser Distance And Ranging
m	meter
ms	millisecond
min	minute
NIST	National Institute of Standards and Technology
NN	Neural Network
O&O	Operational and Organizational (Plan Document)
OSD	Office of the Secretary of Defense
PL	Planner
PRIM	Primitive (RCS node)
RCS	Real-time Control System
RSTA	Reconnaissance, Surveillance, and Target Acquisition
s	second
SC	Scheduler
SME	Subject Matter Expert
SP	Sensory Processing
TACOM	U.S. Army Tank-Automotive and Armaments Command
TARDEC	Tank Automotive Research, Development and Engineering Center
UAV	Unmanned Aerial Vehicle
UGV	Unmanned Ground Vehicle
VJ	Value Judgement
WM	World Modeling
WSTAWG	Weapon System Technical Architecture Working Group
XUV	eXperimental Unmanned Vehicle
DARPA	Defense Advanced Research Projects Agency

Chapter 1

4D/RCS Reference Model Architecture for Unmanned Ground Vehicles

James Albus and **Anthony Barbera**
National Institute of Standards and Technology (NIST)
{james.albus,tony.barbera}@nist.gov

1. Introduction

Rapid advances in autonomous driving capabilities over the past two decades have been enabled by five major technological developments:

1. advances in microelectronics that have produced three orders of magnitude increase in computing power
2. advances in knowledge of how the human brain functions in perceiving the environment, modeling the world, imagining the future, and generating behavior
3. development of imaging sensors that provide unambiguous information about the 3D geometry and dynamics of the world
4. development of a reference model architecture that enables the integration of perception, knowledge representation, value judgment, reasoning, planning, and control into a unified intelligent real-time control system
5. development of a software engineering methodology that enables the embedding of tactical behaviors in the reference model architecture

Each of these developments is significant in and of itself. However, in combination, these five developments have enabled a major scientific breakthrough in autonomous vehicle systems. The next few paragraphs elaborate on these enabling factors in more detail, and examine what might reasonably be expected over the course of the next two decades.

1.1 Computational Power

Massive computational power is a necessary (but not sufficient) enabler of intelligent systems. Over the past two decades, the computational power of computers of size, weight, and cost that can be easily embedded in an average sized automobile has grown by more than 1000 times. In 1985, a computer costing a $1000 was capable of about ten million operations per second. In 2005, a computer of similar size, weight, and cost is capable of nearly ten billion operations per second. An increase of three orders of magnitude is more than a quantitative improvement. A factor of a thousand produces a qualitative transformation of revolutionary proportions. And, there is no indication that this exponential rate of growth in computing power is declining, or about to decline. Rather, it is increasing. Today, the rate of growth in computational power is about an order of magnitude every 5 years. Thus, over the next two decades, it seems likely that the available computational power will increase by four orders of magnitude (i.e., by 10,000 times.) This means that algorithms that a few years ago were completely impractical for real-time systems are now quite feasible. And algorithms that are not feasible today will become practical in 10 to 15 years.

By the year 2025, the computational power of a computer the size and cost of today's laptop will be on the order of 100 trillion operations per second (10^{14} ops.) This is a number that exceeds some estimates of the computational power of the human brain. Today's largest supercomputer already exceeds this capability [3].

1.2 Knowledge of Cognitive Processes

Much has been learned over the past 20 years in the neurosciences and brain modeling community regarding how the brain functions. Much is now known about how neural computational centers in the brain and spinal cord control muscles in planning and executing coordinated movements. Much has been learned about how the visual system processes images to extract edges, surfaces, objects, and groups. Regions of the brain have been identified that recognize faces and hands; that represent space and time; that interpret spoken words and phrases; and combine representations of visual objects with linguistic tokens. Much is known about how acoustic signals are filtered and processed to compute direction, and recognize frequencies, sounds, phonemes, words, and sentences. Knowledge is emerging of how the brain represents spatial information in images and maps; how it represents procedural information in loops between neuronal clusters that act like finite-state automata; how it represents symbolic information in the neural equivalent of abstract symbolic data structures; how it links these data structures with the neural equivalent of pointers that define relationships and class membership; and how it links symbolic data structures back to images and maps with neural pointers that provide symbol grounding. Neuroscientists are learning how the brain encodes state-variables, attributes, vectors, strings, and arrays with populations of neurons; how cortico-thalamic loops provide filtering and recursive estimation; how top-down goals and priorities act to focus attention, steer segmentation, and generate gestalt grouping hypotheses. And it is becoming clear how the brain generates plans and executes strings of goal-driven sensory-interactive actions that form characteristic behaviors [12,15,16,17].

Much also has been learned by computer scientists regarding how to build intelligent controllers, and how to embed knowledge in computer systems. Cognitive architectures, expert systems, image processing algorithms, neural nets, fuzzy systems, and blackboard architectures have been developed that model various aspects of neuronal computation. Signal processing, information theory, game theory, decision making, spatial reasoning, and search-based planning are mature disciplines. Hierarchical task decomposition, process scheduling, representation of objects and processes, and control of complex machines and vehicle systems are well understood, and software tools exist that enable engineers to build large reliable systems. A reference model architecture and software development methodology provides a unified framework for engineering systems of any desired level of intelligence.

1.3 Range Imaging

Safe driving at operational speeds in traffic, on roads, highways, and city streets requires the ability to perceive and model the 3D environment with sufficient speed and precision to support real-time tactical reasoning, path planning, obstacle avoidance, and dynamic control of vehicle motion. Until recently these perceptual capabilities were beyond the capability of machine vision systems.

The problem is that projecting the 3D world onto a 2D image plane (of a television camera) produce only 2D images in which every pixel is completely ambiguous with regard to range. Biological vision systems suffer from the same problem. The retina of the vertebrate eye produces only a 2D image, as does the compound eye of the invertebrates. To recover the 3D information necessary for navigation in the 3D world, the brain employs a number of techniques. These include stereo processing of images from two eyes, and depth cues from image flow, motion parallax, occlusion, shading, and texture. The human brain also applies cognitive knowledge about the expected size of objects, and a variety of assumptions to infer a "best guess" as to the physical structure of the environment. Unfortunately, these processes are not fully understood, and attempts by machine vision researchers to generate robust 3D models of the world from 2D images have been less than successful.

Of course, there are sensing modalities that directly measure the 3D shape of the world. These include sonar, radar, structured light, and LADAR (Laser Detection And Ranging). Of these, sonar and radar are too low in angular resolution to reliably distinguish between objects that are both far away and close together. Both sonar and radar are also subject to multi-path problems that generate unreliable results. In addition, sonar is slow and structured light is effective only at short ranges.

Of the 3D sensing modalities, only LADAR is currently able to provide both the range and resolution needed to build reliable, detailed, 3D dynamic models of the external world at distances required for autonomous driving. LADAR works by measuring the time of flight of a laser beam projected from a source and reflected back to a detector. There are two basic types of LADAR cameras. One operates by mechanically scanning one or more laser beams over the target in a raster pattern to form an image. The other operates by illuminating the target with a single pulse of light, and measuring the time of flight to each pixel in the image with a focal plane array of detectors. Both types of LADAR cameras provide a range value for each pixel. The result is a range image where there is a range value (or no-return) at each pixel. No-return pixels occur where the target does not reflect enough energy to detect.

Current LADAR cameras provide range images with sufficient speed, reliability, and precision that image processing algorithms can build a geometric and dynamic model of the environment within about 50 m of the sensor. Input from an inertial navigation system enables the perception system to build a 3D map of the world in the vicinity of the robot. Update rates of greater than 10 frames per second, enable off-road driving speeds that are typically limited more by the roughness of the terrain than by the capabilities of the LADAR. At night, autonomous vehicles have demonstrated off-road driving speeds higher than can be safely achieved by human drivers using night-vision goggles.

LADAR cameras designed specifically for highway driving are being developed by commercial automobile companies, especially in Europe and Japan. These cameras currently have the ability to detect and track cars up to 80 m away and pedestrians at 40 m. The next generation of these systems will be capable of detecting and tracking cars up to 150 m away. Currently, these cameras are quite expensive, e.g., in the $50K to $100K price range. However, once they are manufactured in large quantities, it is anticipated that prices will drop to a few thousand, and eventually to a few hundred dollars.

Hybrid CCD/LADAR cameras are being developed that overlay color images on range images. Some of these have angular resolution approaching, and even exceeding, the resolution of the human fovea. These cameras typically have range accuracy of a few centimeters at more than 100 m. This is much more precise than human depth perception.

Segmentation and grouping are the computational processes that enable perception to distinguish objects from background in the image domain. Segmentation separates pixels that belong to an object from those that do not, and grouping clusters those pixels into a unified object. Segmentation and grouping are prerequisites for transforming from the iconic domain to the symbolic domain. Segmentation enables the computation of entity attributes, such as size, shape, position, orientation, velocity, color, and texture. Grouping enables the representation of entity attributes in an abstract data structure such as a Lisp list, a C struct, or a C++ object or class. Computation of entity attributes is a prerequisite for classification. When it is determined that entity attributes match class prototype attributes, entities can be assigned to classes. The significance of LADAR imaging resides in the fact that the operations of segmentation and grouping can be performed much more easily and robustly in the 3D space of a LADAR point cloud than in the 2D space of a color or intensity image array.

Examples of entity classes required for autonomous mobility are roads, trails, lane markings, cars, trucks, motorcycles, pedestrians, intersections, traffic signals, and road signs. Once segmented, these objects can be classified and recognized by geometric shape, motion, color, or texture. Relationships between entities, events, and places can be represented by pointers, and pointers can define lists, strings, semantic networks, grammars, and ontologies.

Types of relationships include spatial, temporal, causal, logical, and class relationships. These forms of knowledge enable an autonomous vehicle to identify situations that focus attention and trigger action. Knowledge of entity attributes, classes, and relationships are required by behavior generation processes to generate appropriate behavior for situations such as on-coming traffic, intersections, cross traffic, pedestrians, and traffic signals. Knowledge of entity attributes and state is required for recursive estimation of situations, and prediction of future trajectories of moving objects. This knowledge is also required for

planning obstacle-free paths, and computing safe clearance for driving behaviors such as merging, passing, and parking.

Next generation LADAR cameras and sensory processing algorithms are approaching the capabilities of human perception in building geometric models of the environment. These have not yet been engineered as real-time systems capable of the dynamic performance required for autonomous driving in normal traffic at operational speeds, but the sensors, processing algorithms, and computational power to do this is clearly within reach. What remains is largely an engineering effort to improve reliability and reduce cost.

1.4 System Integration

Engineering intelligent systems requires more than the existence of computational speed, understanding of brain function, techniques for knowledge representation, and sensors and algorithms for perception. It requires a reference model architecture for integrating the many different aspects of perception, world modeling, knowledge representation, reasoning, decision-making, planning, and control into a unified system capable of generating appropriate behavior under operational conditions. And it requires a systematic methodology for engineering software to populate that architecture.

An architecture is a framework consisting of functional modules, interfaces, and data structures. A reference model architecture defines how the functional modules and data structures are integrated into subsystems and systems. The architecture represents a framework wherein issues such as functional modularity, network connectivity, latency, bandwidth, reliability, semantics of communications between modules, and overall system performance can be addressed. A reference model architecture for intelligent systems provides a framework for representing knowledge about the environment (e.g., attributes of the terrain, objects, groups, classes, places, and situations); the mission (e.g., goals, tasks, plans, schedules, intentions, and priorities); and rules of behavior (e.g., beliefs, values, cost, risk, and worth). A reference architecture provides infrastructure for perception, attention, and cognition, including methods for reasoning, modeling, planning, guessing, and learning. And it defines human-machine interfaces (e.g., displays and controls for operators, simulation and training environments for users, and programming and debugging tools for software developers.)

There are a number of architectures that have been developed for building intelligent systems. Some of these such as Soar [18], ACT-R [19], Pilot's Associate, and various Black-Board and Expert Systems architectures are designed to model high-level cognitive elements of human reasoning. Others such as Subsumption [2] are designed to model low-level reactive behaviors. Still others such as AuRA [5] 3-T [6], Claraty [7], and 4D/RCS [4] are hybrid architectures designed to combine high-level planning with low-level behaviors.

1.5 Engineering Methodology

The development of an engineering methodology methodology that enables the embedding of tactical behaviors in the reference model architecture will be the subject of Chapter 2, and thus will not be elaborated here.

2. The 4D/RCS Reference Model Architecture

4D/RCS (4Dimension/Real-time Control System) is the latest version of the RCS (Real-time Control System) reference model architecture for intelligent control system design that has evolved over the past 30 years at NIST and elsewhere.

4D/ RCS addresses the full range of complexity inherent in embodied cognitive systems, from sensing through perception to cognition, decision making, planning and control of appropriate behavior in real-world environments. It incorporates and integrates many different and diverse concepts and approaches into a harmonious whole. It is hierarchical but distributed, deliberative yet reactive. It spans the space

between the cognitive and reflexive, between planning and feedback control. It shows how high level goals can be decomposed and merged with sensory feedback to generate action that is both goal directed and sensory interactive. RCS bridges the gap between spatial distances ranging from kilometres to millimetres, and between time intervals ranging from months to milliseconds. And it does so in small regular steps, each of which can be easily understood and readily accomplished through well known computational processes.

RCS has been used by many different engineers for building controllers for a wide variety of robots and intelligent systems. These include laboratory robots, machine tools, inspection machines, intelligent manufacturing systems, industrial robots, automated general mail facilities, automated stamp distribution systems, automated mining equipment, unmanned underwater vehicles, autonomous operations for nuclear submarines, and unmanned ground vehicles [8]. All implementations of RCS have resulted in real-time intelligent control systems for industrial or military machines operating on realistic objects in real world applications. Simulators have been built for software development and testing, but the emphasis has always been on control of embodied systems.

RCS was originally inspired by a study of the neurological structures in the spinal cord and cerebellum that provide coordinated dynamic control of muscles in the performance of skilled movements. The basic design principle was the combining of commands from higher motor centers with feedback from sensors in the muscles, joints, and tendons, and the transforming of these inputs into control signals to drive the muscles in producing appropriate behaviors. This principle was formalized as a theory of cerebellar function, and implemented in a neural model of the cerebellum that demonstrated the ability to learn simple coordinated motions [22,23]. A neural network based on this model (the Cerebellar Model Articulation Controller (CMAC)) was developed [24,25] that has subsequently been used by a number of researchers for building controllers with the ability to learn simple behavioral skills [20,21]. The principle of top-down control merging with bottom-up feedback remains a central feature of RCS today. Sensory processing is influenced both by top-down expectations and bottom-up observations. World modeling is controlled both by behavioral priorities and sensory feedback. Behavior generation is driven both by task goals and world model situation assessments.

The first version of RCS was developed as a real-time sensory-interactive goal-directed control system for a laboratory robot. Over the next 3 decades, the RCS reference model evolved from a neural model of coordinated dynamic control into a cognitive architecture that bridges the gap between the fields of artificial intelligence, image understanding, and modern control theory.

4D/RCS was developed for the Army Research Lab Demo III Experimental Unmanned Vehicle (XUV) program [9]. 4D/RCS combines the RCS design concept with the 4D approach to machine vision developed by Dickmanns et al. for autonomous highway driving [1,26].

4D/RCS provides an organizational framework of intelligent control nodes, each of which has a well defined role, responsibilities, span of control, range of interest, and resolution of detail in space and time. These can be configured to conform to any type of task management structure, including a military style hierarchical command and control structure. They can be reconfigured at any time in response to requirements imposed by changing operational orders, battlefield losses, or reinforcements.

An example of a 4D/RCS reference model architecture for a single vehicle in a scout platoon attached to a battalion is shown in Figure 1. The architecture consists of a multi-layered multi-resolutional hierarchy of computational nodes, each containing elements of sensory processing (SP), world modeling (WM), value judgment (VJ), behavior generation (BG), and a knowledge database (KD) (included in the WM in Figure 1). Behavior generation plans and executes actions. Sensory processing transforms sensor data into perceived and classified objects, events, and situations. WM processes maintain the KD that is the node's current best estimate of the external world at the scale and resolution that is appropriate for BG planner and executor processes. The WM also generates predictions – both for BG planning and SP recursive estimation. Value judgment evaluates the costs and benefits of predicted results of simulated plans for BG. VJ also computes the level of confidence assigned to information stored in the KD, and assigns worth to

perceived objects, events, and situations. Each node in the architecture represents an operational unit in an organizational hierarchy.

Processing nodes are organized such that the BG processes form a chain of command. There are, in addition, horizontal communication pathways within nodes, and information in the knowledge database is shared between WM processes in nodes above, below, and at the same level within the same subtree. On the right in Figure 1, are examples of the functional characteristics of the BG processes at each level. On the left, are examples of the scale of maps generated by the SP processes and populated by the WM in the KD at each level. Sensory data paths flowing up the SP hierarchy typically form a graph, not a tree. VJ processes are hidden behind WM processes in the diagram. A control loop may be closed at every node. An operator interface may provide input to, and obtain output from, processes in every node (Numerical values are representative examples only. Actual numbers depend on parameters of specific vehicle dynamics).

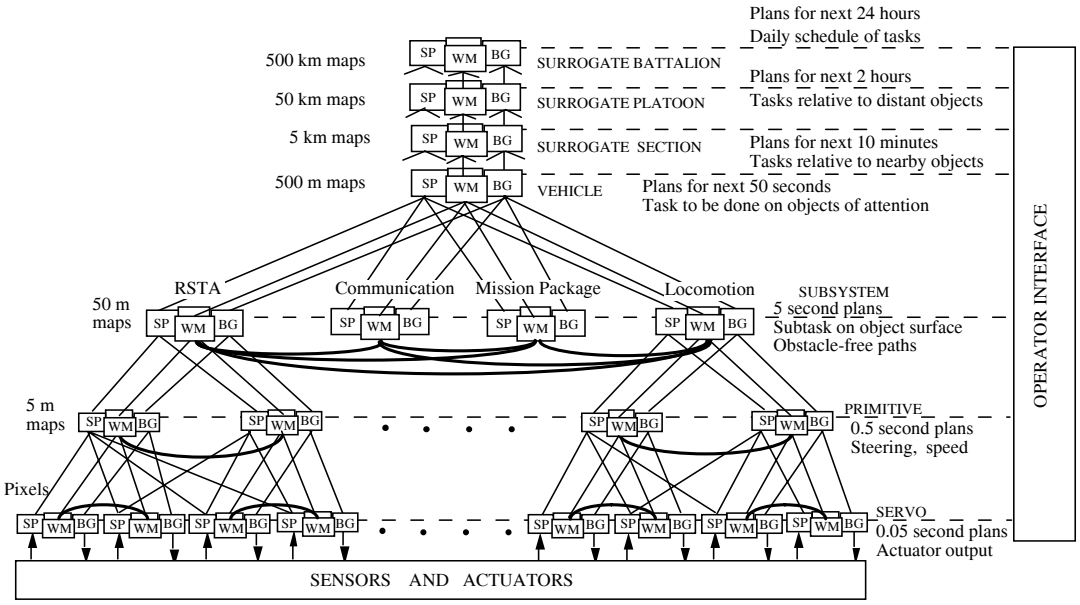


Figure 1. A 4D/RCS reference model architecture for an autonomous vehicle in a scout platoon.

Throughout the 4D/RCS hierarchy, interaction between SP, WM, VJ, BG, and KD give rise to perception, cognition, and reasoning. At lower echelons, the nodes generate goal-seeking reactive behavior where representations of space and time are short-range and high-resolution. At higher echelons, nodes generate goals and deliberative behavior where representations of space and time are long-range and low-resolution. This enables high-precision fast-action response at low echelons, while long-range plans and abstract concepts are being simultaneously formulated over broad regions of time and space at high echelons. Typically, planning horizons expand by an order of magnitude in time and space at each higher echelon.

This hierarchical decomposition of roles and responsibilities provides a way to manage computational complexity. Each node at each level has a well defined and limited set of task skills. Each node relies on the echelon above to define goals and priorities, and provide long-range plans. Each node relies on the echelon below to carry out the details of assigned tasks. Within each node, the KD provides a model of the external world at a range and resolution that is appropriate for the behavioral decisions that are the responsibility of that node.

Each 4D/RCS node is designed to carry out specific duties and responsibilities. Each node is assigned a specified span of control, both in terms of supervision of subordinates, and in terms of range and resolution

in space and time. Typically, control loops (or OODA loops[1]) are closed through each 4D/RCS control node. Each node in the 4D/RCS hierarchy has a characteristic loop latency and update rate. Loop bandwidth tends to decrease at higher echelons in the hierarchy, while span of control increases at each higher echelon. One reason for the decline in loop bandwidth at the higher echelons is that patterns that span greater range in time and space take longer to recognize. Another reason is that it takes longer to generate plans in a larger search space.

As shown in Figure 1, there are surrogate nodes within each vehicle for the Section, Platoon, and Battalion echelons. These enable any vehicle to assume the role of a command vehicle for a section, platoon, or battalion commander. They also provide each vehicle with higher echelon plans, models, goals, and priorities during those periods of time when the vehicle is not in direct communications with its supervisor. For Army vehicles on the ground, this frequently occurs. Surrogate nodes enable single vehicles or groups of autonomous vehicles to cooperate effectively and act appropriately, even when contact with human operators is interrupted.

When 4D/RCS is applied to a system-of-systems, there is node for each organizational unit in the chain of command. At the Vehicle echelon, there is a 4D/RCS node for each vehicle. At the Section echelon, there is a node for each section, at the Platoon echelon a node for each platoon, and at the Battalion echelon a node for each battalion. Duties and responsibilities at these upper echelons involve tactics, techniques, and procedures that require a significant degree of interaction between manned and unmanned vehicles, and a great deal of sophistication in perception, situation understanding, and tactical reasoning. Plans are developed for tasks lasting minutes to hours. 4D/RCS nodes at the Section, Platoon, and Battalion echelons are currently implemented by human commanders and their staff, using computer-assisted data fusion, situation assessment, modeling, simulation, and planning systems. Tactics are developed that make best use of unmanned ground and air vehicles by assisting manned vehicles to maximize effectiveness and minimize causalities.

Figure 2 shows a first level of detail in a typical 4D/RCS node. In each node, a behavior generation process accepts task commands with goals and parameters from a behavior generation process at the next higher echelon, and issues commanded actions with subgoals and parameters to one or more behavior generation process at the next lower echelon.

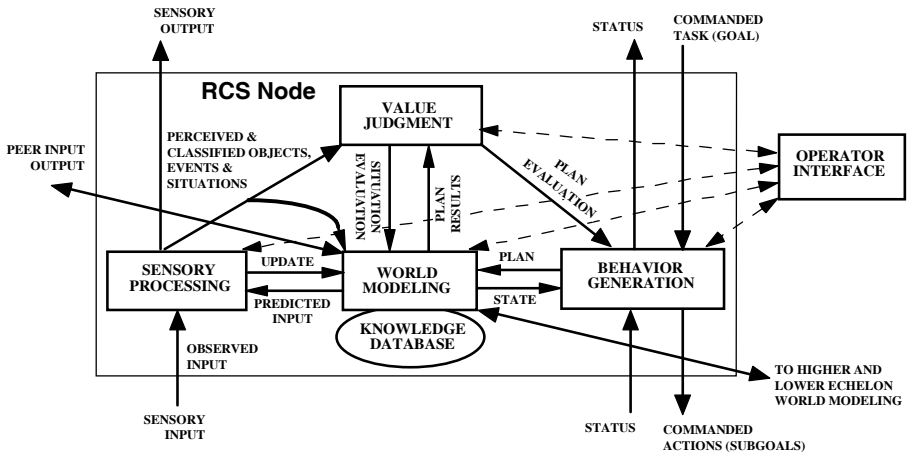


Figure 2. Internal structure of a 4D/RCS node.

[*] OODA refers to Observe, Orient, Decide, and Act [10]. The OODA loop was developed by Col John Boyd, USAF (Ret).

Each node reports on the status of its current task to its supervisor, and receives status reports from each of its subordinates. Within the node, tentative plans are submitted by BG to WM where expected results are simulated. These are sent to VJ for evaluations that are returned to BG for decision making. This is a planning loop.

Each node also receives sensory input from lower levels of the SP hierarchy. This is compared with predicted input from WM. Variance between predictions and observations are used by WM to update KD. This is a recursive estimation loop. SP algorithms in each node also perform attention, segmentation, and classification operations. Perceived and classified entities, events, and situations are evaluated by VJ and entered into the KD by WM. Output from SP is forwarded up the SP hierarchy.

In each node, WM processes maintain a rich and dynamic database of knowledge about the world in the form of images, maps, entities, events, and relationships, all of which are represented at a scale and resolution optimal for performing the perceptual and behavioral responsibilities of that node. Other WM processes use the knowledge database to generate estimates and predictions that support perception, reasoning, and planning appropriate to that node. VJ processes in each node assign worth and importance to objects and events, compute confidence levels for variables in the knowledge database, and evaluate the anticipated results of hypothesized plans.

Communications between WM and SP processes in each node provide the information required for focusing attention, segmentation, grouping, and recursive estimation of attributes and state, and classification of objects, events, and relationships. Interactions between WM and BG processes provide the simulation and modeling capabilities required for decision-making, planning, feedforward and feedback control.

The 4D/RCS architecture also provides well-defined human-machine interfaces that allow human operators to interact with the intelligent control modules, to monitor what is going on in any node, and/or to provide supervision, or exert control in any node, at any time at any echelon of the hierarchy. In Figure 2, solid lines indicate normal data pathways. Dotted lines indicate channels by which an operator can peek and poke at data, and insert or modify control commands.

Figure 3 shows a second level of detail in a typical 4D/RCS node. The BG process accepts tasks and plans and executes behavior designed to accomplish those tasks. The internal structure of the BG process consists of a planner and a set of executors (EX). At the upper right of Figure 3, task commands from a supervisor BG process are input. A planner module decomposes each task into a set of coordinated plans for subordinate BG processes. For each subordinate there is an Executor that issues commands, monitors progress, and compensates for errors between desired plans and observed results. The Executors use feedback to react quickly to emergency conditions with reflexive actions. Predictive capabilities provided by the WM may enable the Executors to eliminate delays in the feedback loop, and even generate pre-emptive behavior.

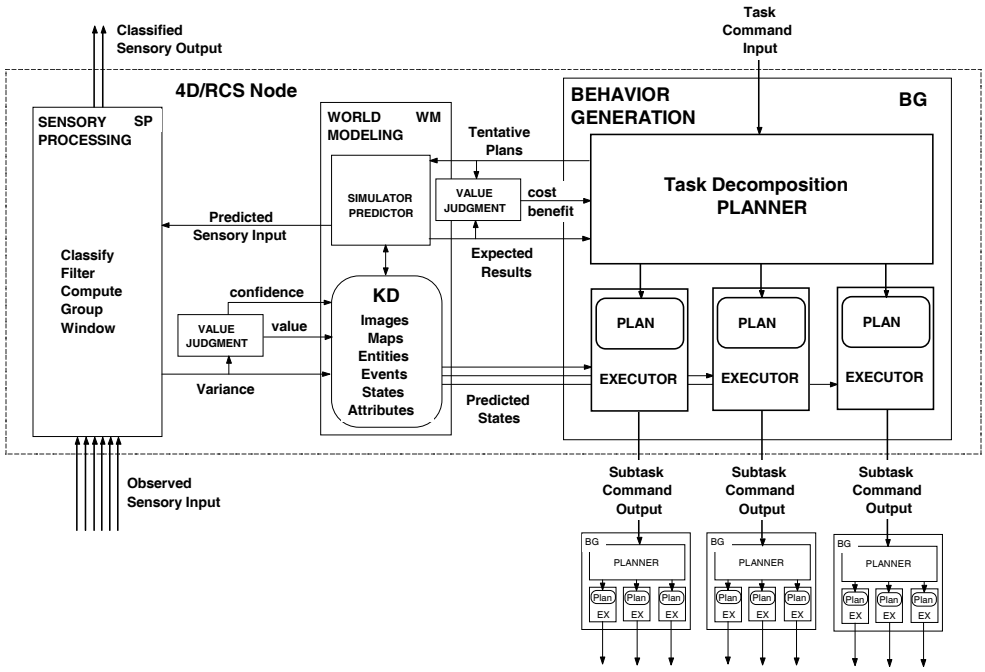


Figure 3. A second level of detail of processes within a typical 4D/RCS computational node.

SP processes operate on input from lower echelons and sensors by windowing (i.e., focusing attention), grouping (i.e., segmentation), computing attributes, filtering (i.e., recursive estimation), and classifying (e.g., recognizing) entities, events, and situations.

A VJ process evaluates expected results of tentative plans. VJ process also assigns confidence and worth to entities, events, and situations entered into the KD.

Figure 3 also illustrates that each node contains both a deliberative and a reactive component. Top-down, each node generates and executes plans designed to satisfy task goals, priorities, and constraints conveyed by commands from above. Bottom-up, each node closes a reactive control loop driven by feedback from sensors. Within each node, deliberative plans are merged with reactive behaviors. Whenever the planner develops a new plan, it is substituted for the old plan in the plan buffers of the Executors. Each Executor treats its current plan as a reference trajectory. It uses planned actions as feedforward control signals, and uses the difference between planned states and estimated states as feedback control signals.

2.1 Planners

In 4D/RCS, plans may be generated by any of a variety of planning algorithms, e.g., case-based reasoning, search-based optimization, or schema-based scripting. Planning is distributed throughout the hierarchy. Each RCS node has its own autonomous planner that generates plans for the agents in its operational unit. Any node can accept plans from an outside source such as a human operator. Typically, planning at the lower echelons is performed autonomously on-board the unmanned vehicle, and planning at the upper echelons is done interactively by a human operator using a computer assisted planner in a human-robot interface.

There are two methods for generating plans that are currently being implemented in 4D/RCS.

2.1.1 Case-Based Planning

Case-based planning uses situation-action logic as the primary method of task decomposition. Case-based planning uses a library of plans represented as state-graphs (or state-tables) with at least one plan for each task command. There are typically two types of case-based planning: one that decomposes a task into job assignments for multiple agents, and a second that decomposes each agent's job assignment into a schedule that may be coordinated with peer agents. Slots for parameters such as priorities, constraints, tolerances, modes, and speeds may be filled in at planning time. This approach is discussed further in Chapter 2.

2.1.2 Search-Based Planning

Search-based planning performs a search over the space of possible states and actions to find the best course of action, or the best sequence of subgoal states, to achieve the goal. Search-based planning may use a map or spatial graph with cost overlays to evaluate various possible paths through the map or graph. Typically, this requires a model that predicts how each planned action will affect the system state at the appropriate level of resolution. Alternatively, search-based planning may use an inverse model that predicts what actions are required to generate a sequence of desired states. In either case, the planner generates a series of planned actions and resulting states that predict how the system is expected to behave in the real world environment. These simulated actions and predicted states can then be evaluated by a cost-function that takes into account constraints and priorities that are passed down from higher echelons, as well as uncertainties and knowledge about the environment from sensors. A typical cost-function for an autonomous ground vehicle may take into account the cost of fuel or time, the traversibility of terrain, the estimated cost of collision with various types of objects, the risk of detection or attack by an enemy, and the benefit or payoff of achieving or maintaining a goal.

Search-based planning is a widely researched field. A large variety of planning techniques have been reported in the literature. One algorithm that has proven particularly efficient is the incremental creation and evaluation of the planning graph. This incremental approach reduces the number of planning-graph nodes that must be created and evaluated to find the cost-optimal path through the planning graph. This approach is discussed further in Chapter 3.

3. 4D/RCS for Demo III

Figure 4 is a view of the first five echelons in a chain of command passing through the locomotion subsystem of the 4D/RCS architecture developed for Demo III. On the right of Figure 4, Behavior Generation (consisting of Planner and Executor) decompose high level mission commands into low level actions. The text inside the Planner at each echelon indicates the planning horizon at that echelon.

At all echelons, 4D/RCS planners are designed to generate new plans well before current plans become obsolete. Thus, action always takes place in the context of a recent plan, and feedback through the executors closes reactive control loops using recently selected control parameters. To meet the demands of dynamic battlefield environments, the 4D/RCS architecture specifies that replanning should occur within about one-tenth of the planning horizon at each echelon.

The Executor at each echelon executes the plan generated by the Planner. Meanwhile, the Planner is replanning based on an updated world state. Each planner has a World Model simulator that is appropriate for the problems encountered within the node at its echelon. The Planners and Executors operate asynchronously. At each echelon, the Planner generates a new plan and the Executor outputs new commands to subordinates on the order of ten times within each planning horizon. At each lower echelon, the planning horizons shrink by a factor of about ten. The relative timing relationships between echelons are designed to facilitate control stability and smooth transitions among hierarchical control echelons. The timing numbers in Figure 4 are illustrative only. The actual rates may differ in different applications.

At all echelons, Executors are designed to react to sensory feedback more quickly than the replanning interval. The Executors monitor feedback from the lower echelons on every control cycle. Whenever an Executor senses an error between its output CommandGoal and the predicted state (status

from the subordinate BG Planner) at the GoalTime, it may react by modifying the commanded action so as to cope with that error. This closes a feedback loop through the Executor at that echelon within a specified reaction latency.

At the bottom of Figure 4 are actuators that act on the world and sensors that measure phenomena in the world. 4D/RCS is designed to accommodate a variety of sensors that include a LADAR, stereo CCD cameras, stereo FLIRs, a color CCD, vegetation penetrating radar, GPS (Global Positioning System), an inertial navigation package, actuator feedback sensors, and a variety of internal sensors for measuring parameters such as engine temperature, speed, vibration, oil pressure, and fuel level.

To the left of the Planners and Executors are maps used for path planning. At each echelon, the planning maps have range and resolution that are appropriate for path planning at its echelon. The bottom (Servo) echelon has no map representation. The Servo echelon deals with actuator dynamics and reacts to sensory feedback from actuator sensors. The Primitive echelon map has range of 5 m with resolution of 4 cm. This enables the vehicle to perform precise maneuvers such as parking or moving through narrow passages, and to make small path corrections to avoid bumps and ruts. The Primitive echelon uses accelerometer data to control vehicle dynamics and prevent roll-over during high speed driving. The Subsystem echelon map has range of 50 m with resolution of 40 cm. This map is used to plan about 5 s into the future to find a path that avoids obstacles and provides a smooth and efficient ride. The Vehicle echelon map has a range of 500 m with resolution of 4 m. This map is used to plan paths about 1 min into the future taking into account terrain features such as roads, bushes, gullies, or tree lines. The Section echelon map has a range of 5 km with resolution of about 40 m. This map is used to plan about 10 min into the future to accomplish tactical behaviors. Higher echelon maps (not shown in Figure 4) can be used to plan platoon, company, and battalion missions lasting about 1, 5, and 24 h respectively. These are derived from military maps and intelligence provided by the digital battlefield database.

On the far-left of Figure 4 are five levels of sensory processing. At each level there are segmented images with labeled regions (or entities), symbolic frames representing entities, and links between the image regions and frames. In the center are coordinate transforms that use range information to assign pixels and regions in images on the left to regions in maps on the right. The maps have range and resolution defined by the planning horizon of the planners at each echelon. The images have resolution and field of view defined by the imaging optics and the attention window.

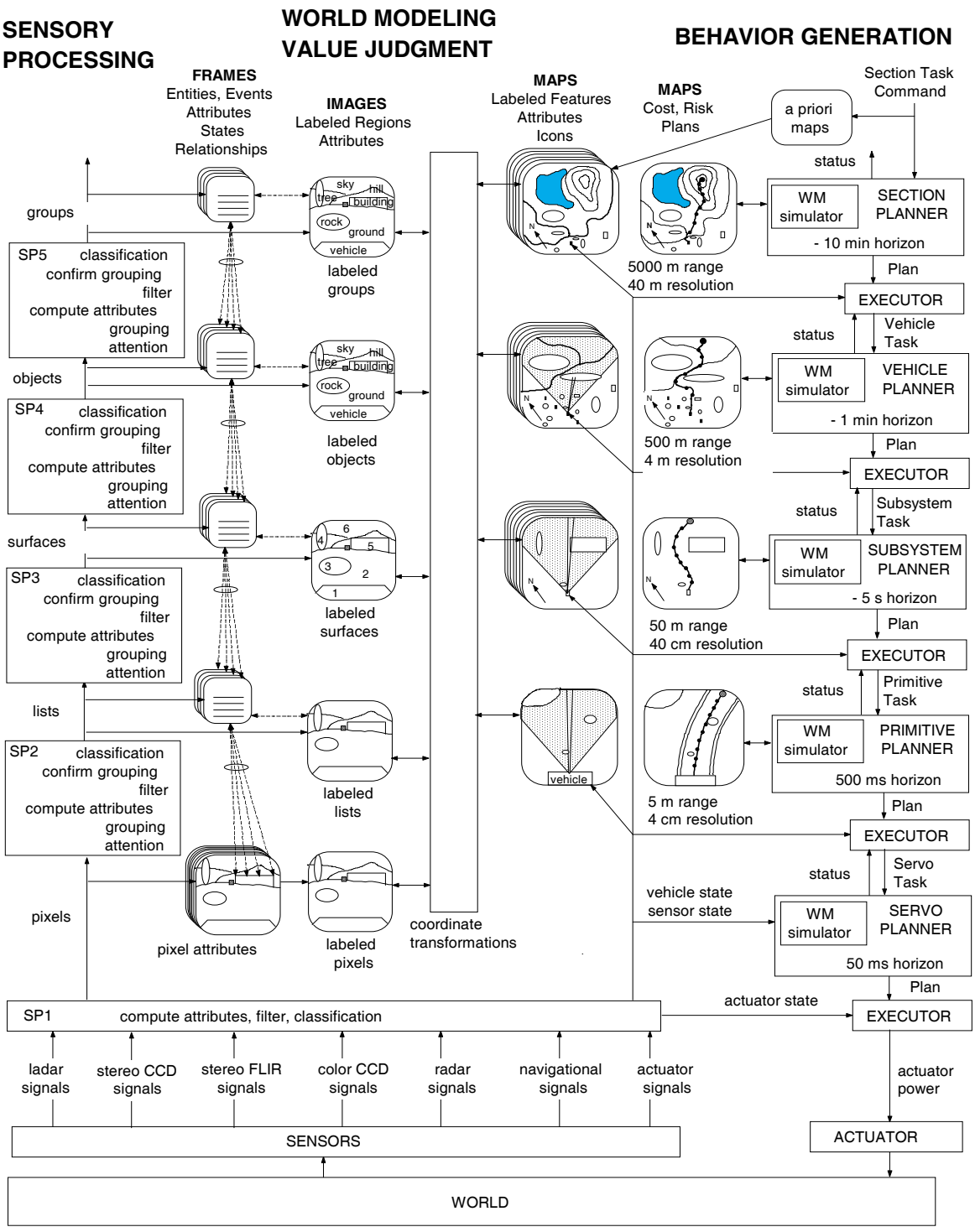


Figure 4. Five layers in a chain of command through the 4D/RCS architecture.

During the Demo III program, only the first three echelons in the locomotion subsystem of the 4D/RCS reference model architecture were implemented. Most of the emphasis was focused on the Subsystem echelon where terrain maps were overlaid with traversability costs, and planning algorithms were used to

search for the lowest cost path out to the visual horizon provided by the LADAR camera. Higher echelon path planning using Digital Terrain Databases was performed in the Human-Robot Interface (HRI). Plans generated in the HRI were sent to the XUV in the form of Vehicle echelon plans. The point where a Vehicle echelon plan intersected the Subsystem echelon planning horizon was chosen as a goal for the Subsystem echelon path planner. The Primitive echelon BG process used a "Pure Pursuit" algorithm to generate commands to the Servo echelon, where a position servo algorithm was implemented to control hydraulic steering actuators and hydrostatic wheel motors. Additional details are available from Chapter 10.

The most recent implementation of the 4D/RCS architecture resides on the ARL/NIST HMMWV mobility testbed. On this vehicle, both feedforward and feedback algorithms have been implemented at the Servo echelons to provide precise steering and speed control. Algorithms at the Primitive echelon generate dynamically safe paths defined by straight lines and constant curvature arcs, with blending to minimize jerk. Path planning at the Subsystem echelon provides obstacle avoidance and gaze control for mobility sensors. The most recent application of 4D/RCS also addresses issues at the Vehicle, Section, and Platoon echelons where tactical behaviors for missions such as route reconnaissance are being developed. The Vehicle echelon coordinates all activities on the vehicle, including gaze control for distant vision, communications with the C4ISR network, and mission packages such as weapons and countermine sensors. More details are available from Chapter 3.

4. Representing Knowledge in 4D/RCS

The 4D/RCS architecture is designed to accommodate multiple types of knowledge representation formalisms and provide an elegant way to integrate these formalisms into a common unifying framework. This section will briefly describe the types of knowledge representations that have been researched and/or implemented within the 4D/RCS architecture for autonomous driving and the mechanisms that have been deployed to integrate them. More detail is provided in Chapter 4.

The 4D/RCS architecture represents knowledge in both declarative and procedural forms. Declarative knowledge resides in the Knowledge Database (KD) in the form of images, maps, entities, events, attributes, states, episodes, and relationships. Procedural knowledge resides in BG processes in the form of rules, formulae, programs, procedures, recipes, scripts, and algorithms for decision-making, learning, planning, and control. Procedural knowledge also resides in SP and WM processes for perception and world modeling.

4.1 Declarative Knowledge

Declarative knowledge is about the condition of the world. Declarative knowledge may describe the size, shape, state (i.e., position, orientation, velocity), and class of entities. It may describe the start-time, duration, frequency, or temporal pattern of events. Declarative knowledge may also describe the spatial or temporal relationships that exist between and among entities and events in places and situations. Declarative knowledge is represented in a format that may be manipulated, decomposed, and analyzed by reasoning engines.

Declarative knowledge enables a system to know the current state of the environment and its own situation relative to other entities in the environment. Declarative knowledge enables a system to reason logically or mathematically to predict what will result from possible future actions and events. Two types of declarative knowledge that are captured within 4D/RCS are symbolic and iconic.

4.1.1 Symbolic Knowledge

Symbolic knowledge representations use abstract data structures (such as LISP frames, C structs, or C++ objects and classes) to represent things (e.g., actions, entities, or events) in the world that can be referenced by name. 4D/RCS uses entity and event frames to represent symbolic knowledge. An example of an entity frame is shown in Figure 5.

```
NAME = entity_id (uncertainty)     // this is the frame address in the KD

// attributes – these are characteristics that address the question  What?
color              = red, green, blue intensities
size               = length, height, width dimensions
shape              = curvature, moments, axes of symmetry, etc.

//  state – these are dynamic properties that address the question  Where?
position           = azimuth, elevation, range (uncertainty)
orientation        = roll, pitch, yaw (uncertainty)
velocity           =v-azimuth, v-elevation, v-range, v-roll, v-pitch, v-yaw (uncertainty)

// class – pointers to the entity image and classes to which the entity belongs
entity image       = pointer to the entity image in which the entity appears
generic class1     = pointer to the generic class1 prototype
generic class2     = pointer to the generic class2 prototype
generic class3     = pointer to the generic class3 prototype

// value or worth of the entity
worth to preserve = value of preserving
worth to acquire  = value of acquiring
worth to defend   = value of defending
worth to defeat   = value of defeating

// pointers that define parent-child relationships
belongs to         = pointer to parent entity
has part1                  = pointer to subentity1
has part2                  = pointer to subentity2
has part3                  = pointer to subentity3

// pointer that define situational relationships
on top of                  = pointer to entity below
beside-right       =pointer to entity on right

// queues that store short-term state history and expected future states
short term memory                  = pointer to STM queue
short term expectations    = pointer to STE queue

// functions that define behavior
behavior1                  = responds-to
behavior2                  = acts-like
```

Figure 5. The structure of a typical entity frame.

The entity name is an address or index by which the entity frame can be accessed in a database or library of entities, and to which other entity frames can be linked. An uncertainty parameter can be associated with the name to indicate how certain the system is that the entity has been properly identified. The entity frame contains entity attributes that describe attributes such as color, size, and shape. Entity attributes are characteristics and properties of the region occupied by the entity. Observed entity attributes can be computed by integrating attributes of the pixels belonging to the entity in the observed image. Predicted entity attributes can be computed by applying a model-based predictor to recently estimated entity attributes stored in the world model. This is a recursive filter approach to perception.

Entity state-variables describe dynamic properties such as position, orientation, and velocity of the entity in a particular coordinate system. These can be used as parameters in world modeling processes for

prediction and simulation, and by sensory processing functions for classification, detection, and recognition. Uncertainty parameters can be associated with state-variables to indicate the dependability of the estimates of these values. Observed state-variables are typically defined in sensor egosphere coordinate system (i.e., a polar coordinate system with the sensor at the origin.) Estimated and predicted entity state-variables may also be in egosphere coordinates, or in some other more convenient coordinate system.

In general, entity state-variables are time dependent. Thus, for each state-variable, there exists a string of states that define its trajectory through state space. This can be represented in the entity frame by a queue of states that store a short-term memory trace for each state-variable. Predicted future states of each entity state-variable can also be represented in an entity frame by a queue of states that represent short-term expectations for each entity state-variable.

An entity frame may contain a pointer to the image or map in which it appears. This can enable a graphic engine to overlay entity names or attributes on the image or map. An entity frame may also contain pointers to the class or classes to which the entity belongs. This enables an observed entity to inherit the attributes of the class to which it belongs. An entity may have a hierarchy of class pointers. For example an object entity frame may have a generic class pointer that identifies it as a member of the class of trees. The object entity frame may also have a second generic class pointer that identifies it as a member of the class of pine trees, as well as a specific class pointer that identifies it as a particular pine tree.

The entity frame may contain value attributes that define how valuable an object is as a target (if a foe), or how worth defending (if friend). Value attributes may also define how much a particular entity or situation should be feared or avoided, what its worth is as a source of shelter or food, or as a vantage point. This enables a decision process to calculate whether the expected benefit of achieving a situation would be worth the anticipated cost or risk of pursuing it.

An entity frame may contain pointers that define inheritance relationships with other entities. Each entity frame has a pointer to the frame of a parent entity of which it is a part, and a set of pointers to frames of child entities (or sub-entities) that are its parts. For example, an object entity frame typically will have a parent pointer to the entity frame of the group to which the object belongs. It will have a number of sub-entity pointers to the surface entity frames of the surfaces and boundaries that are its parts. Inheritance pointers are established and maintained by grouping and classification operations performed by sensory processing functions at various levels in the hierarchy.

An entity frame may also contain pointers that define spatial, temporal, causal, or other types of relationships that pertain to that particular entity. An entity frame may include a set of functions that describe the behavior of the entity under certain conditions or in response to certain stimuli. Simple functions may define the behavior of objects under the influence of gravity or friction. Complex functions may define how the entity might be expected to respond to an attack or a gesture of friendship. Behavioral functions may include parameters such as speed, endurance, strength, or range of weapons. Behavioral functions and parameters may be inherited from generic or specific class prototypes. A block diagram of an entity frame is shown in Figure 6.

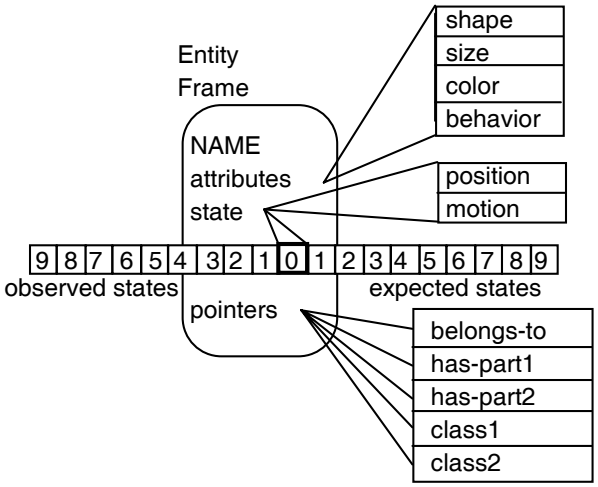


Figure 6. An entity frame consisting of a name, an attribute vector, a state vector, a queue of state vectors consisting of observed and expected states, and a set of pointers that describe relationships with other patterns.

Entity, event, and task frames can be interconnected by pointers to form networks that represent causal, semantic, and situational relationships. Situational networks may represent situations or geometric relationships such as *"on-top-of", "beneath", "to-the-right-of", "in-front-of", and "inside-of"*. Situational networks may also have pointers to maps or images that pictorially display spatial relationships. Causal networks represent the cause-and-effect relationships between entities, events, situations, and actions that occur in the world. Semantic networks represent the relationships between entities, attributes, situations, actions, and events that define meaning and enable reasoning, logic, and language. Forward pointers from images to nodes in a semantic network, and backward pointers from semantic net nodes to images on the egosphere provide symbol grounding between entities, events, and relationships in the external real-world, and their corresponding representations in the world model.

4.1.2 Iconic Knowledge

Iconic knowledge provides information about objects and situations in space and time in a manner that directly represents spatial and temporal relationships (e.g., images, maps, and state trajectories.) Iconic representations typically use scalars, vectors, or arrays to represent things that can be measured (e.g., attributes) about the world. Iconic representations typically subdivide the world and are referenced by location. The location of each element in an iconic representation often corresponds to (or projects onto) a dimension or location in physical space. For example, the location of a pixel in an image corresponds to a geometrical projection of the world onto the image, and vice versa. The location of an element in a vector or array may also correspond to the location of a tactile sensor on the skin. Iconic depictions are a form of metrical representations.

Each pixel of an iconic array may be represented by a Boolean or real number representing the value of a physical attribute such as light intensity, color, altitude, range, or density at that point in the array. Each pixel may also be represented by a vector of attributes representing the values of spatial or temporal gradients of intensity, color, and range; or of image flow direction and magnitude. A pixel vector may also contain a pointer to a symbolic data structure representing an entity (e.g., an edge, vertex, surface, object, or group) to which the pixel belongs. An array of pixel vectors generates a 3D matrix of attribute, entity, class, and worth images (as shown in Figure 7).

Iconic representations have scale, and are limited in range and resolution. Both images and maps have a finite number of pixel elements. Images have limited fields of view and maps have boundaries. Similarly, temporal events have beginning and end, and can only be sampled at a finite rate.

Examples of iconic knowledge currently used within 4D/RCS include maps and images. Maps may be expressed in a variety of formats including survey and aerial maps, or Digital Terrain Elevation Databases (DTED) containing information about hydrology, ground cover, roads, bridges, streams, woods, and buildings. Images include video or LADAR images from cameras mounted on the ground vehicle. To be useful for path planning beyond line of sight, the information gathered by sensors on the ground vehicle must be registered with a priori maps provided by external sources.

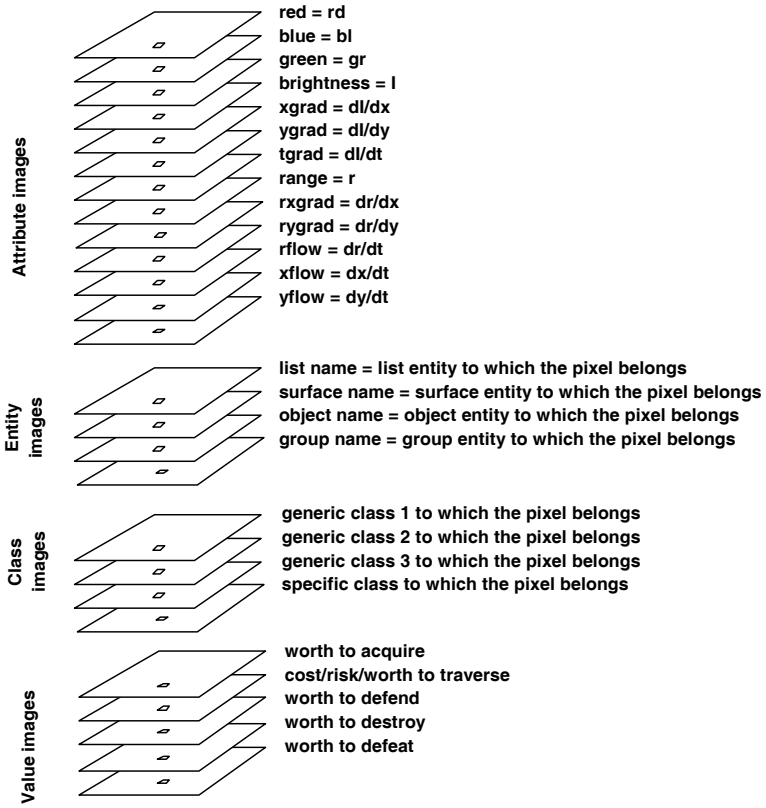


Figure 7. Attribute, entity, class, and value images. These images are registered to form a 3D matrix such that each pixel has an attribute vector, a pointer to an entity frame, one or more pointers to the class(es) to which the pixel belongs, and values assigned to the region where the pixel is located.

A hybrid iterative algorithm has been developed for registering 3D LADAR range images obtained from unmanned aerial vehicles with LADAR images obtained from unmanned ground vehicles [13]. Registration of the UGV LADAR to the aerial survey map minimizes the dependency on GPS for position estimation.

This is important when GPS estimates are unreliable or unavailable. More details are available from Chapter 6.

4.2 Procedural Knowledge

Procedural knowledge is the knowledge of how to perform tasks. Procedural knowledge can be captured in task frames. A task frame is a data structure specifying all the knowledge necessary for accomplishing a task. A task frame is essentially a recipe consisting of a task name, a goal, a set of parameters, a list of materials, tools, and procedures, and set of instructions of how to accomplish a task. For each task that a RCS node is able to perform, there exists a task frame. For example, a task frame may include:

1. the task name (index into the library of tasks the RCS node can perform) The task name is a pointer or an address in a database where the task frame can be found.
2. a task identifier (unique id for each task command) The task identifier provides a means for keeping track of tasks in a queue.
3. a task goal (a desired state to be achieved or maintained by the task) The task goal is the desired result of executing the task.
4. a task goal time (time at which the task goal should be achieved, or until which the goal state should be maintained)
5. one or more task objects (on which the task is to be performed) Examples of task objects include objects to be observed, sectors to be reconnoitered, vehicles to be driven, targets to be attacked or defended, weapons or cameras to be aimed.
6. a set of task parameters (that specify, or modulate, how the task should be performed) Examples of task parameters are speed, force, priority, constraints, tolerance on goal position, tolerance on goal time, tolerance on path, coordination requirements, and level of aggressiveness.
7. agents (that are responsible for executing the task) Agents are the subsystems and actuators that carry out the task.
8. task requirements (tools needed, resources required, conditions that must obtain, information needed) Tools may include instruments, sensors, and actuators. Resources may include fuel and materials. Conditions may include temperature, pressure, weather, visibility, soil conditions, daylight or darkness. Information needed may include the state of a task, or a description of an event or situation in the world.
9. task constraints (upon the performance of the task) Task constraints may include speed limits, force limits, position limits, timing requirements, visibility requirements, tolerance, geographical boundaries, or requirements for cooperation with others.
10. task procedures (plans or schema for accomplishing the task, or procedures for generating plans) Plans may be prepared in advance and stored in a library, or they may be computed on-line in real-time. Task procedures may be simple strings of things to do, or may specify contingencies for what to do under various kinds of circumstances.
11. control laws and error correction procedures (defining what action should be taken for various combinations of commands and feedback conditions) These typically are developed during system design, but may be refined through learning from experience.

Some of the slots in the task frame are filled by information from commands. Others are properties of the task itself and what is known about how to perform it. Still others are parameters that are supplied by the WM.

The task procedures (slot #10 in the task frame) consist either of plans, or planning procedures for generating plans. In general, plans can be represented as state-tables (or state-graphs.) State-tables and state-graphs are duals. The advantage of the state-graph representation is that behavior can easily be visualized. The advantage of the state-table representation is that state-tables can be directly executed by an extended finite-state automata. A state-table (or corresponding state-graph) may contain as many state-dependent branching conditions as necessary to cover the space of things that the system is capable of doing in response to situations represented in the system's world model.

Both state-graph and state-table representations can be changed by adding or modifying rules at any node in the state graph, or by adding nodes to the graph. This enables the 4D/RCS node to learn new rules, and

formulate new behaviors to optimize its probability of success. What is required for state-graph learning is for an expert critic to point out where in the state-graph the system should have performed differently, what piece of information in the system's world model should have been used to trigger the different behavior, and what the different behavior should have been. This is the type of information typically supplied to a human student by a human instructor or teacher.

State-table representations of task decomposition often call functions to perform mathematical calculations such as feedback and feedforward control equations. Parameters in these equations can be adjusted by a variety of machine learning techniques.

5. Perception

Perception is the intelligent system's window onto the world. Perception begins with sensing and ends with a World Model containing information that is relevant to the task at hand and adequate for decision-making and planning.

In biological creatures, perception is a hierarchical process that begins with arrays of tactile sensors in the skin, arrays of photoreceptors in the eyes, arrays of acoustic sensors in the ears, arrays of inertial sensors in the vestibular apparatus, arrays of proprioceptive sensors (that measure position, velocity, and force) in the muscles and joints, and a variety of internal sensors that measure chemical composition of the blood, pressure in the circulatory system, and several other sensory modalities. Biological perception results in an awareness of the situation in the world and of the self in relation to the world.

In 4D/RCS, visual perception is a hierarchical process that begins with arrays of pixels in cameras, signals from inertial sensors and GPS receivers, and signals from actuator encoders. It ends with a world model consisting of data structures that include a registered set of images and maps with labeled regions, or entities, that are linked to each other and to entity frames that contain entity attributes (e.g., size, shape, color, texture, temperature), state (e.g., position, orientation, velocity), class membership (e.g., road, lane marker, tree, vehicle, pedestrian, building), plus a set of pointers that define relationships among and between entities and events (e.g., situations.) These provide the autonomous vehicle with awareness of the world and of itself in relation to objects in the world.

It should be noted that, contrary to popular opinion, perception does not function by reducing a large amount of sensory data to a few symbolic variables that are then used to trigger appropriate behaviors. Instead, perception increases and enriches the sensory data by computing attributes and combining it with *a priori* information so that the World Model contains much more information (not less) than what is contained in the sensory input. For example, of the multiple image representations in Figure 7, only the intensity, color, and range images come directly from sensory input. Enriching the sensory input expands the dimensionality of the decision space and enables the perception processes to segment the world into meaningful entities, events, and relationships, and to detect patterns and recognize situations that can then be bound to symbolic variables that trigger behavior.

To cope with complexity, perception does not treat all regions of the visual world equally. Attention focuses sensory processing resources on those parts of the world that are important to goal-driven task decomposition processes, or that are novel or unexpected. Attention masks out (or assigns to the background) those parts of the sensory input that are irrelevant to task goals, or those aspects of sensory input that are predictable and therefore not noteworthy. The role of attention is to focus perceptual resources on what is important for achieving current and near future task goals.

In 4D/RCS, sensory processing generates a hierarchy of image entities and entity frames. These are linked to a hierarchy of maps with differing range and resolution. It should be noted, however, that the hierarchy of range and resolution for maps is not parallel with the hierarchy of image entities and entity frames. The hierarchy of entities is generated by grouping and segmentation processes at each level of the SP hierarchy. The hierarchy of range and resolution of maps is specified by the planning horizon of the behavior generation processes in the BG hierarchy that use the maps. This can be seen in Figure 4, where distant

objects in the image may appear only in the section echelon map, whereas close objects in the image appear magnified in the primitive echelon map.

There are five sensory processing steps that may be implemented at each level in the SP hierarchy in Figure 4.

5.1 Focus Attention

At the lowest level in the SP hierarchy, focusing attention means pointing the high resolution part of the visual field toward those regions of the world that contain information important to the task. At higher levels, focusing attention means that SP computing resources are committed to regions in the image that are important to the task, while remaining regions are largely ignored.

5.2 Grouping and Segmentation

Portions of the visual field that belong together must be grouped into entities and segmented from the rest of the image. At the lowest level, grouping consists of integrating all the energy imaged on each single pixel in the camera. At higher levels, pixels and entities are grouped according to gestalt heuristics such as proximity, similarity, contiguity, continuity, and symmetry. Grouping also establishes pointers from segmented regions in the image to entity frames that contain knowledge about the entity attributes, state, and relationships. Each grouping operation is a gestalt hypothesis that represents the system's best guess of how to interpret the sensory input.

5.3 Computation of Entity Attributes

Attributes and state of each entity must be computed and stored in an entity frame. Attributes may include size, shape, color, texture, and temperature. State includes position, orientation, and velocity.

5.4 Recursive Estimation

Recursive estimation on entity attributes filters noise and enables the perception system to confirm or deny the gestalt hypothesis that created the entity in step two. Recursive estimation uses entity state and state-prediction algorithms to track entities from one image to the next. When predictions correlate with observations, confidence in the gestalt hypothesis is strengthened. When variance occurs between predictions and observations, confidence in the gestalt hypothesis is reduced. When confidence rises above a credibility threshold, the gestalt hypothesis that established the entity is confirmed.

5.5 Classification

Confirmed entity attributes can be compared with attributes of class prototypes. When a match occurs, the entity can be assigned to the class. Once an entity has been classified, it inherits attributes of the class. There is a hierarchy of classes to which an entity may belong. For example, an entity may be classified as a geometrical object, as a tree, as an evergreen tree, as a spruce tree, and as a particular spruce tree. More computing resources are required to achieve more specific classifications. Thus, an intelligent system typically performs only the least specific classifications required to achieve the task.

The 4D/RCS architecture suggests that these five steps should be performed at each echelon of the sensory processing hierarchy.

A simple example of these processes is illustrated in Figure 8. Figure 8(a) is a range image from a high resolution LADAR camera. The range image in Figure 8(a) is segmented (using a connected components algorithm based on proximity in 3D space) into the object entity image shown in Figure 8(b). In Figure 8(b), each object is labeled with a different color. For each labeled object entity in Figure 8(b), attributes are computed and stored in an object entity frame of the form shown on the left in Figure 8(c). The attributes in the object entity frame are then compared with stored class prototype attributes, as shown on the right in Figure 8(c). When a match is detected between object attributes and class prototype attributes,

the entities in the object entity image are assigned to the matching class, and a class image can be created as shown in Figure 8(d). In this simple example, only height and width attributes were needed to classify two objects in Figure 8 as adult humans.

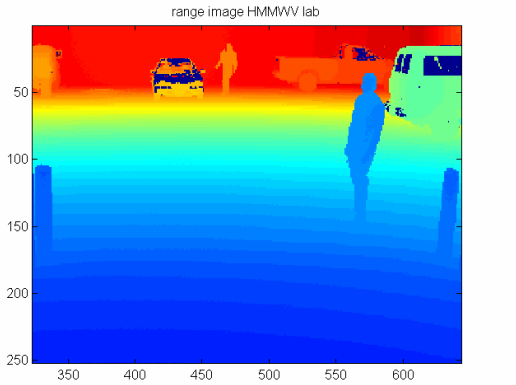


Figure 8(a). A range image of two human mannequins, vehicles, two posts, and a wall in the background.

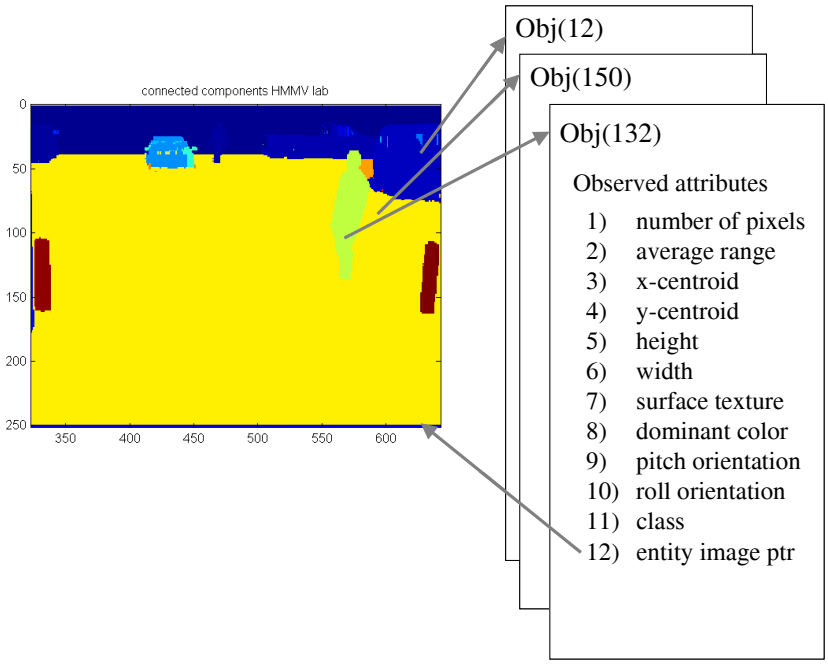


Figure 8(b). A segmented entity image. The different colors refer to different objects. Each pixel in the entity image points to a entity frame with a set of attributes and a class pointer.

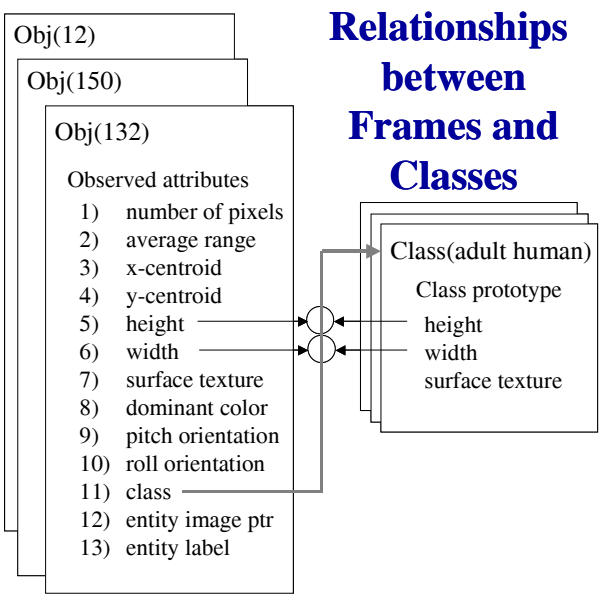


Figure 8(c). When object attributes match class prototype attributes, the objects can be can be assigned to the matching class.

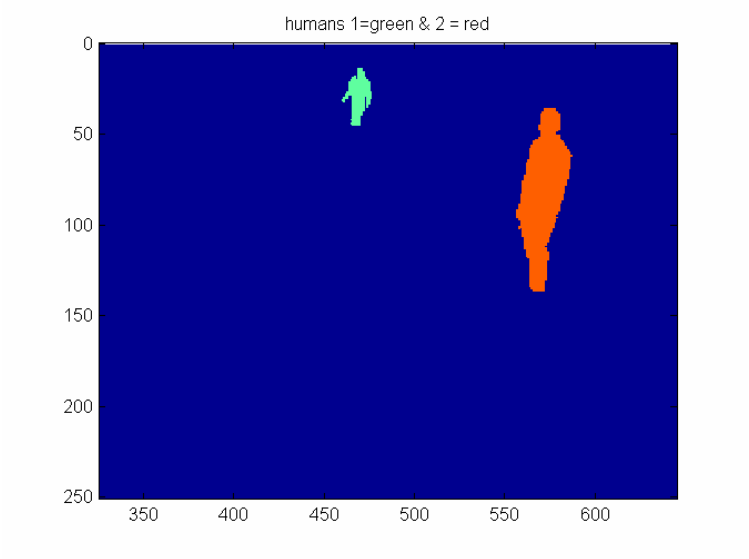


Figure 8(d). A class image. Each pixel points to the class to which it belongs.

6. Experimental Results

Experimental validation of the 4D/RCS architecture has been provided by the performance of the Army Research Lab Demo III eXperimental Unmanned Vehicle (XUV) shown in Figure 9.

Figure 9. The Army Demo III Experimental Unmanned Vehicle. On the left-top is the LADAR. In the center-top is a Reconnaissance Camera ball. On the right-top is a pan/tilt unit with a color stereo pair, a FLIR stereo pair, and a color high resolution monocular camera. The white panel in the center front is for a radar. The front bumper is instrumented to detect obstacles hidden in the weeds (Photo courtesy of General Dynamics Robotic Systems).

Four of these XUVs were put through an extended series of demonstrations and field tests during the fall and winter of 2002-2003. The vehicles were built and operated by General Dynamics Robotic Systems. The tests were conducted by the National Institute of Standards and Technology. The purpose of the tests was to show that autonomous mobility technology had reached a technology readiness level where a system prototype could be demonstrated in a relevant environment.

The XUVs were equipped with an inertial reference system, a commercial grade GPS receiver (accurate to about +/- 20 m), a LADAR camera with a frame rate of 10 frames per second, and a variety of internal sensors. The LADAR had a field of view 90 degrees wide and 20 degrees high with resolution of about ½ degree per pixel. It was mounted on a pan/tilt head that enabled it to look in the direction that it planned to drive. The LADAR was able to detect the ground out to a range of about 20 m, and detect vertical surfaces (such as trees) out to a range of about 60 m. Routes for XUV missions were laid out on a terrain map by trained Army scouts, and given to the XUVs in terms of GPS waypoints spaced more than 50 m apart.

During the technology readiness tests, the XUVs operated completely autonomously until they got into trouble and called for help. Typical reasons for calling for help were the XUV was unable to proceed because of some terrain condition or obstacle (such as soft sand on a steep slope, or dense woods), and was unable to find an acceptable path plan after several attempts at backing up and heading a different direction. At such a point, an operator was called upon to teleoperate the vehicle out of difficulty. During these operations, data was collected on the cause of the difficulty, the type of operator intervention required to extract the XUV, the time required before the XUV could be returned to autonomous mode, and the work load on the operator.

During three major experiments designed to determine the technology readiness of autonomous driving, the Demo III experimental unmanned vehicles (XUVs) were driven a total of 550 km, over rough terrain: 1) in the desert, 2) in the woods, 3) through rolling fields of weeds and tall grass, 4) on dirt roads and trails, and 5) through an urban environment with narrow streets cluttered with parked cars, dumpsters, culverts,

telephone poles, and mannequins. Tests were conducted under various conditions including night, day, clear weather, rain, and falling snow (See Figure 10). The unmanned vehicles operated without any operator assistance over 90% of both time and distance. A detailed report of these experiments has been published [11]. High resolution LADAR ground truth data describing the terrain where the XUVs experienced difficulties was also gathered and analyzed [14]. Chapter 9 describes the technology readiness level experiments in more detail.

Figure 10. An Army Demo III eXperimental Unmanned Vehicle driving autonomously through the woods during a snow storm at Ft. Indiantown Gap, Pennsylvania in January, 2003.

It should be noted that the Demo III technology readiness tests were performed in environments devoid of moving objects such as on-coming traffic, pedestrians, or other vehicles. The inclusion of moving objects in the environment, and the development of perception, world modeling, and planning algorithms for operating in the presence of moving objects is a topic of current research.

7. Summary

The 4D/RCS reference model architecture has enabled the integration of perception, knowledge representation, planning, adaptation, learning, and control into a system of systems that enables intelligent behavior in single autonomous vehicles, as well as groups of vehicles that include semi-autonomous manned vehicles, and unmanned ground and air vehicles.

The 4D/RCS architecture provides a framework for integrating high-level cognitive functions with low-level reactive behaviors in a unified real-time control system design.
4D/RCS is hierarchical but distributed, deliberative yet reactive. It spans the space between the cognitive and reflexive, between planning and feedback control. It bridges the gap between spatial distances ranging from kilometers to millimeters, and between time intervals ranging from months to milliseconds. And it does so in small regular steps, each of which can be easily understood and readily accomplished through well known computational processes.

Each organizational unit in 4D/RCS refines tasks with about an order of magnitude increase in detail, and an order of magnitude decrease in scale, both in time and space. At the upper levels, most of the computational power is spent on cognitive tasks, such as analyzing the past, understanding the present, and planning for the future. At the lower levels, most of the computational power is spent in motor control, and the early stages of perception.

Yet at every level, the computational infrastructure is fundamentally the same (except for scale in time and space). The architecture consists of generic computational nodes and modules (that theoretically could be

implemented as neural nets, or finite state automata, or production rules). Each node accepts inputs and produces outputs with specified range and resolution in time and space. Knowledge is represented in arrays (images and maps), strings, pointers, frames (entities and events), and rules. At various levels and in many different ways, computational nodes (groups of agents) process sensory data, model the world, and decompose high-level intentions into low-level actions. Within each node, this process is both limited in complexity and finite in scope.

4D/RCS makes the processes of intelligent behavior understandable in terms of computational modules that can be engineered into practical machines. The 4D/RCS architecture can be implemented using a software development methodology described in Chapter 2. The 4D/RCS methodology provides software development guidelines and tools for the capture of knowledge, skills, and abilities from textbooks, subject matter experts, and experience in real and virtual training environments.

Many current DOD projects have chosen 4D/RCS as a reference model architecture. These include, the Army Research Lab Robotics Collaborative Technology Alliance, the Army Future Combat System Autonomous Navigation System, and the Army Tank Automotive Research, Development, and Engineering Center Vetronics Integration program.

It should be noted, however, that many features of the 4D/RCS reference model architecture have yet to be fully implemented in any application. Although the fundamental concept has been demonstrated as valid, and the more advanced features have been shown to be feasible, much more work remains to be done before the 4D/RCS architecture is fully populated with operational software.

The problem of autonomous driving is far from solved. There is still much left to do. Many new algorithms need to be developed for attention, segmentation, grouping, recursive estimation, predictive filtering, world modeling, knowledge representation, decision-making, reasoning, planning, reacting, and controlling actuators. Many new approaches to perception, cognition, and behavior need to be developed and tested.

References

1. Dickmanns, E.D., Graefe, V. "Dynamic Monocular Machine Vision", and "Application of Dynamic Monocular Machine Vision", Journal of Machine Vision and Applications, pp. 223-261, November 1988.
2. Brooks, R.A. "A Robust Layered Control System for a Mobile Robot", IEEE Journal of Robotics and Automation, RA-2, pp. 14-23, 1986.
3. An Overview of the BlueGene/L Super Computer, The BlueGene Team, http://www.llnl.gov/asci/platforms/bluegenel/sc2002-pap207.pdf, 2002.
4. Albus, J., *et al.* "4D/RCS: A Reference Model Architecture for Unmanned Vehicle Systems", Version 2.0, NISTIR 6910, National Institute of Standards and Technology, Gaithersburg, MD 20899, 2002.
5. Arkin, R.C. "Behavior-Based Robotics", MIT Press, Cambridge, MA, 1998.
6. Gat, E. "Three-Layer Architectures. In Mobile Robots and Artificial Intelligence", D. Kortenkamp, R. P. Bonasso, and R. Murphy (eds.), pp. 195-210, Menlo Park, CA, AAAI Press, 1998.
7. Volpe, R., Nesnas, I.A.D., Estlin, T., Mutz, D., Petras, R., Das, H. "CLARAty: Coupled Layer Architecture for Robotic Autonomy" JPL Technical Report D-19975, December 2000.
8. Albus, J. "The NIST Real-time Control System (RCS): An Approach to Intelligent Systems Research", Journal of Experimental and Theoretical Artificial Intelligence, Vol. 9, pp. 157-174, 1997.
9. Shoemaker, C., Bornstein, J., Myers, S., and Brendle, B. "Demo III: Department of Defense Testbed for Unmanned Ground Mobility", SPIE Conference on Unmanned Ground Vehicle Technology, SPIE Vol. 3693, Orlando, FL, April 1999.
10. http://www.12manage.com/methods_boyd_ooda_loop.html
11. Camden, R., Bodt, B., Schipani, S., Bornstein, J., Runyon, T., French, F., Shoemaker, C., Jacoff, A., and Lytle, A., "Autonomous Mobility Technology Assessment Final Report", Army Research Laboratory Technical Report – 3471, Aberdeen Proving Ground, MD, April 2005.

12. Albus, J.S. and Meystel, A.M., "Engineering of Mind: An Introduction to the Science of Intelligent Systems", John Wiley & Sons, Inc., New York, NY, 2001.
13. Madhavan, R., Hong, T., Messina, E. "Temporal Range Registration for Unmanned Ground and Aerial Vehicles", Journal of Intelligent and Robotic Systems, Springer Science & Business Media B.V., Vol. 4, No. 1, pp. 47-69, September 2005.
14. Witzgall, C., Cheok, G.S., Gilsinn, D.E., "Terrain Characterization from Ground-Based LADAR", Proceedings of the Performance Metrics for Intelligent Systems (PerMIS) Workshop, National Institute of Standards and Technology, Gaithersburg, MD, 2003.
15. Koch, C. "The Quest for Consciousness: A Neurobiological Approach", Roberts & Company Publishers, March 2004.
16. Kandel, E., Schwartz, J., Jessell, T., "Essentials of Neural Science and Behavior", McGraw-Hill/Appleton & Lange, September 1996.
17. Fuster, J. "Memory in the Cerebral Cortex: An Empirical Approach to Neural Networks in the Human and Nonhuman Primate", MIT Press, June 1999.
18. Laird, J., Newell, A., and Rosenbloom, P., "SOAR: An Architecture for General Intelligence", Artificial Intelligence, Vol. 33, pp. 1-64, 1987.
19. Anderson, J. "The Architecture of Cognition", Lawrence Erlbaum Associates, Mahwah, N.J., 1983.
20. Miller, W.T., Glanz, F.H., Kraft, L.G. "CMAC: An Associative Neural Network Alternative to Backpropagation", Proceedings of the IEEE, Special Issue on Neural Networks, vol. 78, pp. 1561-1567, October 1990.
21. van der Smagt, P., Hirzinger, G. "The Cerebellum as Computed Torque Model", Proceedings of the Fourth International Conference on Knowledge-Based Intelligent Engineering Systems & Applied Technologies, R. J. Howlett and L. C. Jain (eds.), pp. 760-763, 2000.
22. Albus, J.S., "A Theory of Cerebellar Function", Mathematical Biosciences, Vol. 10, pp. 25-61, 1971.
23. Albus, J.S., "Theoretical and Experimental Aspects of a Cerebellar Model", Ph. D. Thesis, University of Maryland, College Park, MD, 1972.
24. Albus, J.S., "A New Approach to Manipulator Control: The Cerebellar Model Articulation Controller (CMAC)", Transactions of the ASME Journal of Dynamic Systems, Measurement, and Control, September pp. 220-227, 1975.
25. Albus, J.S., "Data Storage in the Cerebellar Model Articulation Controller (CMAC)", Transactions of the ASME Journal of Dynamic Systems, Measurement, and Control, pp. 228-233, September 1975.
26. Dickmanns, E., "A General Dynamic Vision Architecture for UGV and UAV", Journal of Applied Intelligence, Vol. 2, pp. 251-270, 1992.

Chapter 2

A Task Analysis Methodology for the Derivation and Organization of Knowledge for Real-time Control Systems

Anthony Barbera and **James Albus**
National Institute of Standards and Technology (NIST)
{tony.barbera,james.albus}@nist.gov

2.1 Introduction: The 4D/RCS Task Analysis Methodology for Organizing and Representing Task Knowledge

The RCS methodology has evolved over a number of years as a technique to capture task knowledge and organize it in a framework conducive to implementation in computer control systems. A fundamental premise of this methodology is that the present state of the task activities sets the context that identifies the requirements for all of the support processing. In particular, the task context at any time determines what is to be sensed in the world, what world model states need to be evaluated, which situations need to be analyzed, what plans should be invoked, and what behavior generation knowledge needs to be accessed. This results in a design methodology that concentrates first and foremost on a clear understanding of the task and all of the possible subtask activities.

This methodology starts with the definition of the task knowledge in the form of a task decomposition tree that represents the branching of tasks into layers of simpler and simpler subtask activities. This task decomposition framework is used to provide an organized manner of partitioning the knowledge into smaller and smaller pieces so at the end one is left with detailed descriptions of the knowledge required for each subtask activity in terms of the component actions that make up that specific activity along with the identification of the relevant objects and events that affect its execution.

This chapter explores the RCS design methodology in some detail, showing how it is used to define and represent knowledge in a task context-sensitive framework that can be reliably and consistently updated as new knowledge is discovered. An important attribute of this representation is that knowledge be accessible to Subject Matter Experts (SMEs) for analysis and critique. The RCS methodology is a scenario-based engineering process. In this chapter, the analysis of an autonomous on-road driving task will be used as a real-world example to detail the steps of this process.

The Generic 4D/RCS module

The 4D/RCS methodology models real-time, goal oriented task control as containing three major processing components (see Chapter 1) within each control module [1]:

1) sensory processing to measure, recognize, and classify entities and events of task interest in the environment;
2) internal world model processing that represents and derives world states, situations, and evaluations in a task context manner and provides value judgment processing that computes the expected cost, benefit, and risk of alternative courses of action; and
3) behavior generation processing that reasons from this world model, selects plans, and executes appropriate output actions that best accomplishes the goal task.

A 4D/RCS control system is formed from a number of these control modules organized in a hierarchical relationship where each module performs a finer task decomposition of the goal it receives from its supervising module. In each of these control modules, these three processing components work together, receiving a goal task, breaking it down into a set of simpler subtasks, determining what has to be known in the internal world model to decide on the next course of action, and alerting the sensory processing as to what internal world objects have to have their states updated by new sensory readings/measurements. The design methodology for understanding the

control problem for a 4D/RCS control system, therefore, tries to discover and represent the task knowledge in a manner consistent with this view of the three major component activities.

Task Decomposition Steps

The 4D/RCS methodology concentrates on the task decomposition as the primary means of understanding the knowledge required for intelligent control. This approach begins with the knowledge "mining" activities to retrieve knowledge from SMEs. The gathering and formatting of this knowledge can be summarized in six steps, each of which follows from the knowledge uncovered by the previous step. Figure 1 presents a high level summary view of the overall approach that will be detailed step-by-step in the following sections.

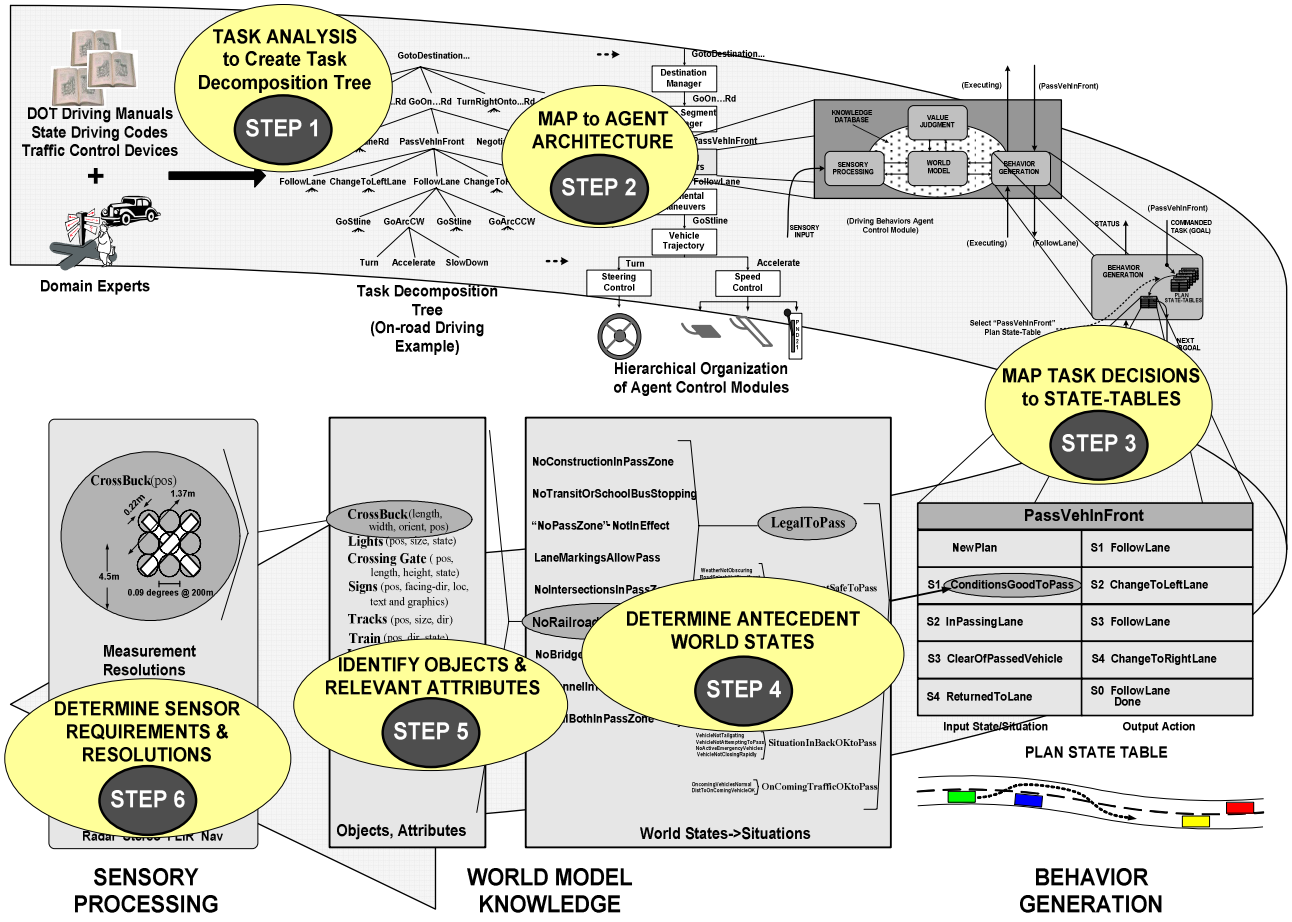


Figure 1. The six steps of the RCS methodology approach to knowledge acquisition and representation.

This methodology has been formalized into an engineering process that provides a well-structured approach to identifying the task knowledge. The purpose of this engineering process is to provide a standard approach to begin analyzing any problem for which one is trying to develop an intelligent control system. Without this type of process, one is left with ad hoc approaches that model every system differently, result in development of components of very uneven complexity with ill-defined task responsibilities, unclear interfaces, and tangled execution interactions which result in unreliable systems. A standard engineering process on the other hand, identifies a starting point to attack large complex systems that seem overwhelming when first approached. It provides formulaic steps that result in a systematic discovery of the knowledge set in a manner consistent with its use in the implementation of the control system. And it results in a representation that remains understandable by the SMEs. Thus, the task knowledge can be evaluated, critiqued, and added to in a manner that allows continued robust growth and improvement of the capabilities of large complex intelligent control system implementations.

The 4D/RCS engineering process for the discovery of and structuring of the task knowledge follows a six-step process:

1) **Determine Task Decomposition:** The first step involves an intensive analysis of domain knowledge from manuals and SMEs, especially through the use of scenarios. The output of this effort is a structuring of this knowledge into a task tree form of finer and finer commands (actions/verbs) at finer and finer levels of task description. The use of scenarios in a task analysis has been discussed in [2].
2) **Define Agent Architecture:** Next, the hierarchical organization of agent control modules that will execute these layers of commands is defined. This is the same as coming up with a business or military organizational structure of agent control modules (people, soldiers) to accomplish the desired tasks. This step forces a more formal structuring of all of the subtask activities in the sense of defining which knowledge will reside with which agent control module.
3) **Define and Group Task Decision Rules:** This step clarifies how each agent's input command will decompose through the use of rules that identify all of the task branching conditions with their corresponding output commands (Input Condition - Output Action rules). Each of these command decompositions at each agent control module will be represented in the form of a state table of ordered production rules (which is an implementation of an extended Finite State Machine (FSM)). The ordered list of simpler output commands required to accomplish the input command and the named input condition-branching situations that transition the state table to the next output command are the primary knowledge represented in this step.
4) **Determine Dependent World States:** The condition side of each rule is a named situation that identifies the transitioning condition to the next output action. In this step, these situations are defined in greater detail in terms of their dependencies on world states and task states. This step attempts to define the detailed antecedent states that cause a particular situation to be true. As an example, the situation of ConditionsGoodToPass used to transition a rule to the output action of changing into the passing (left – oncoming) lane within a PassVehInFront state table might include an antecedent world state such as "NoConstructionInPassZone" (Figure 1). Throughout this chapter, concatenated phrases such as PassVehInFront will be used to signify actions and their selecting situations.
5) **Identify Objects to be Classified:** Here, all of the objects and entities together with their particular features and attributes that are relevant to defining the above world states identified in step 4 are identified and named.
6) **Specify Sensing Requirements:** The last step is to use the context of each particular subtask to establish the required minimum sensing distances and, therefore, the resolutions at which the above objects and entities must be measured and recognized by the sensory processing component. This step establishes a set of requirements and/or specifications for the sensor system at the level of each separate subtask activity. Also relevant for each subtask activity will be whether the sensor system or the objects or both will be moving and at what speed since this will dramatically affect sensing requirements. Environmental conditions such as temperature, lighting conditions, weather, etc. and their effects on sensing are also considered here.

2.2 The use of scenarios as the primary tool to discover task knowledge

The 4D/RCS engineering process approach is to analyze the driving tasks through an evaluation of a large number of particular individual scenarios of previously experienced on-road driving events and from these descriptions to derive a task decomposition representation of possible task activities at various levels of abstraction and detail. Additionally, one has to also think of what the situation might be that causes one to choose to execute one subtask activity rather than other possible alternate subtask activities. For example, if one wishes to pass the vehicle in front of own vehicle on a two lane undivided road, the conditions/situations that would cause own vehicle to begin the passing maneuver have to be determined i.e., start to move into the oncoming left lane to go around the vehicle.

These conditions/situations become the input conditions of the if-then rules of the knowledge set that define how the task is execute in response to input state changes. Detailed analysis of each of these conditions/situations is used to identify their antecedent world states. These world states, in turn, are further analyzed to identify all of the entities, objects, and attributes that have to be sensed to determine if any of these world states exist. The major problem is how to uncover all of this knowledge in the first place. To do this, a technique of having the subject matter expert recount past events through scenarios will be used.

The use of scenarios to recount particular events that the SME has experienced has the benefit of aiding the associative memory recall process of humans. Humans are particularly good at storing and retrieving information in a story format that follows a sequence – this sequential, associative retrieval is symptomatic of human associative

memory. It appears human memories are stored in the sequence that they occurred along with everything else that was sensed at the time. This is suggestive of why humans use stories to record and recall oral histories and traditions and why "war-stories" of past notable events can be recalled with such great detail. It is this ability to recall all of the associated detail that goes along with the sequence of activities concerned with a particular event or activity that leads to the use of scenarios as the primary technique to get at the SME's embedded knowledge.

Discussion of various scenarios allows one to determine which tasks seem to be subsets of other tasks. This allows one to first put structure to the task decomposition tree, to establish what the higher-level tasks are and what tasks become component subtasks to them. From this, one can establish that FollowLane is a subtask to PassVehInFront which is a subtask to a commercial trip planner-like (e.g. MapQuest®) tasks of GoOn-*roadname*, TurnRightOnto-*roadname*, etc.

During this process, each of the various scenarios will be discussed in great detail, noting sequences of simpler activities and the situations that define when a particular activity was selected in a particular instance. For example, the PassVehInFront task requires that the FollowLane behind the vehicle to be passed until the ConditionsGoodToPass situation occurs, followed by pulling out into the oncoming lane, which is a ChangeToLeftLane maneuver. Then proceed down this lane (FollowLane) until the vehicle has been cleared with sufficient distance (ClearOfPassedVehicle) and there is an opening in front of that vehicle that can be pulled back into. Now ChangeToRightLane and continue normal FollowLane activities. From this scenario vignette, a sequence of finer resolution subtasks can be defined that make up the pass vehicle task. These are FollowLane, ChangeToLeftLane, FollowLane, ChangeToRightLane, and FollowLane. Details exposed during the recounting of specific instances of passing maneuvers result in uncovering various world states that combine to create situations that affect the decisions of when to execute each of these simpler subtask activities.

A specific remembrance of a passing maneuver might identify the need to watch for a bicyclist ahead in the road because of the significant increase in risk that results with two cars going side-by-side around the bicyclist thereby considerably reducing the allowed road space offset that the vehicle being passed can provide around the bicyclist. Another recounting might identify a time when a passing maneuver was begun only to have the expert surprised by a second vehicle from behind already trying to pass own vehicle. These types of recounting result in the identification of all sorts of other pertinent world states relevant to the passing task.

As a result, requiring the SME to recount various experiences as scenarios/stories leads to a very detailed discussion of a number of relevant subtasks; their correct sequencing to accomplish larger tasks; alternate contingent activities; and the detailed situational parameters (world states) that have to be looked for in order to choose the best action out of the possible set of available actions. This then is the expert's knowledge.

Once an initial task tree has been identified, the activities can be organized into a more rigorous layering by creating an organizational structure of agent control modules that are responsible for executing the different levels of the task decisions. This use of separate executing agents organized into an execution hierarchy provides a mechanism to formalize the task decision tree by assigning certain decisions to particular agent control modules. This creates well-defined sets of subtask commands from each supervisor agent control module to its subordinate agent control module, thus forcing to group and label various sets of related activities of the driving task with a subtask context identifier such as PassVehInFront, TurnLeftAt_StopSign, PullOffOnto_LeftShoulder, etc. Each of these identifiers is really a subtask goal command at some level in the execution hierarchy. The task decision rules that identify the task decomposition of subtask goal command within an agent control module can be encoded within an FSM. These FSMs can be represented either in the form of a state graph or as a state table. In each of these state tables, the rules that are relevant to a particular subtask activity at a particular level of abstraction are grouped. These rules identify the specific situation that will trigger the state table to transition to the next state and the output actions that will occur as a result. The use of these grouped sets of rules into separate state tables applies a well-structured formalism to the task description. It also has the advantage of keeping the task description easily understandable to the user since each state table only encodes the small number of rules associated with one particular subtask activity at one level in the task decomposition decision tree.

Scenarios will be continually used during this development to see if the state tables can be stepped through detailed sequences of activities to correctly respond to various situations. In other words, this organizational representation of state tables should correctly execute the sequence of subtasks required to carryout high-level commands. This procedure tests the basic sequencing of the finer level tasks at each layer to validate that an entire operational sequence can be represented and also to ensure correct representation of any coordination between different control modules that might be required during the task execution.

Scenarios also are used to test alternate execution of the possible subtasks by introducing different world states/situations and see if the representation of the state table grouping of rules will correctly adapt. During this process, many additional situations and world states are usually identified along with concomitant finer delineated output actions. These become new rules that are added to the appropriate state tables.

The six steps of the 4D/RCS engineering methodology will be examined now in more detail:

2.3 Step 1: The development of the task decomposition tree and identification of task coordination requirements

The first step is to gather as much task related knowledge as possible with the goal of defining a set of commands that incorporate all of the activities at all levels of detail. For on-road driving, this knowledge source would include driving manuals [3], state and federal driving codes, manuals on traffic control devices and detailed scenario narratives from SMEs of large numbers of different driving experiences.

Many scenarios are explored in great detail in an attempt to come up with the names of commands that describe the activities at finer and finer resolutions of detail. Figure 2 provides an example. The high level goal of GotoDestination-*name* (such as GoToDestination-*post office*) is decomposed into a set of simpler commands such as, GoOnRoad-*roadname*, TurnLeftOnto-*roadname*. At the next level down, these commands are decomposed into simpler commands such as DriveOnTwoLaneRoad, PassVehicleInFront and these are decomposed to yet simpler commands such as FollowLane, ChangeToLeftLane, etc.

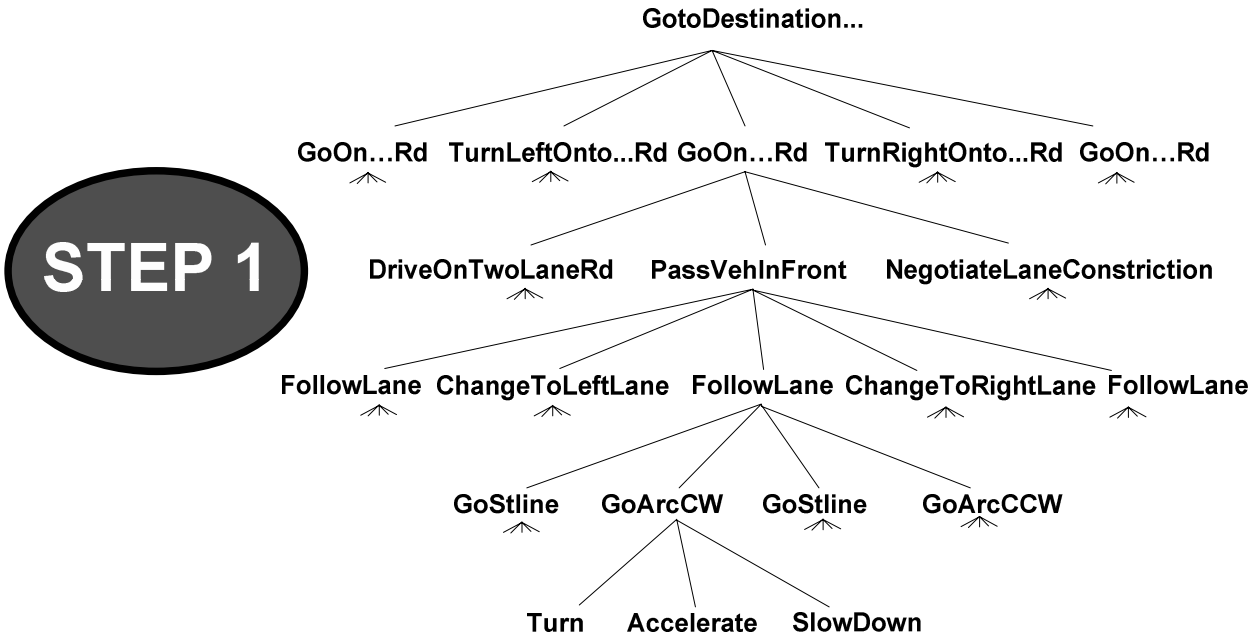


Figure 2. Task decomposition decision tree for on-road driving example. Shows the simpler commands that are used at each lower layer to represent the finer and finer resolutions of detail activities.

Four very important things are being done with the knowledge in this step.

1) The first is the discovery and naming of simpler component subtasks that go into making up the more complex tasks.
2) The second is that for each of these component subtasks, a subtask command name is defined.
3) The third is the understanding of the coordination of subtask activities that the task involves.
4) The fourth is the careful grouping of these commands by layer and decomposition to ensure that all of the examples of on-road driving tasks can be completely described, from the start to finish of a scenario, by the proper sequencing of these commands at the appropriate levels.

2.4 Step 2: The design of a multi-resolutional organizational architecture to structure task responsibilities and extent of authority

Once a set of commands is defined, an organization of agents is needed to execute them. This step is identical to laying out an organizational structure of people in a business or the military. What needs to be done at various

levels of detail is known and now an organization of intelligent agents is needed to do it. This structure is built from the bottom up. The above detailed task decomposition will provide the levels of agents to have in the organization but not how many agents at each level or how they are grouped and coordinated. This step starts

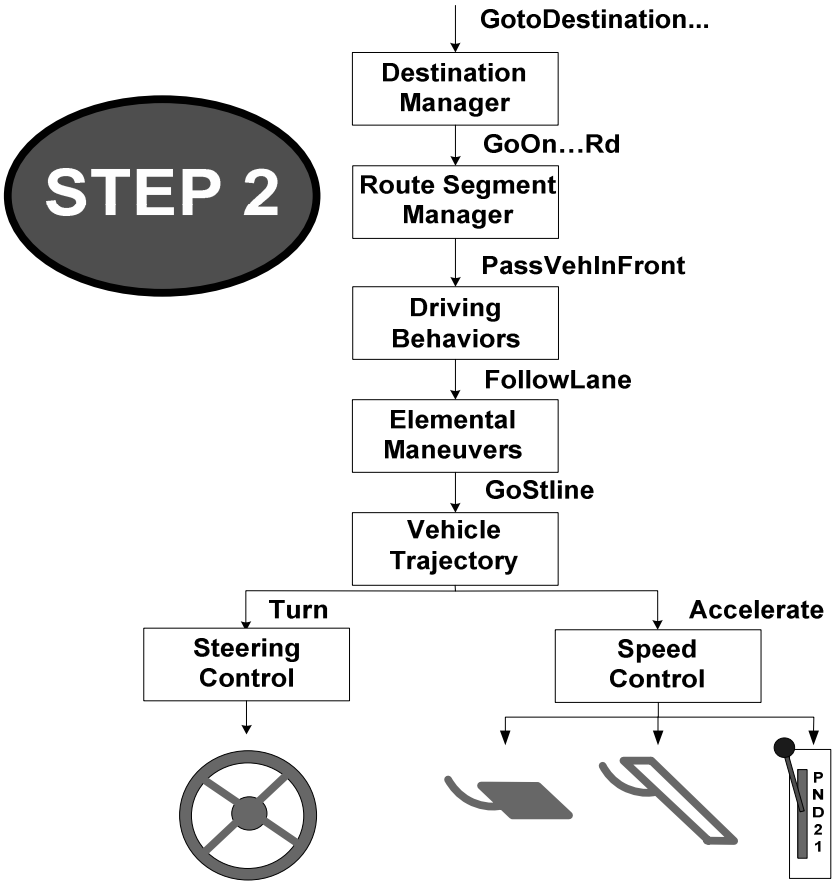


Figure 3. The hierarchical organization of agent control modules that are to execute the task command decomposition.

at the bottom with an agent control module assigned to each actuator in the system and then uses the knowledge of the task activities to understand how agents should be grouped under supervisors to best coordinate the task commands from step 1.

Figure 3 illustrates how a grouping of agent control modules is assembled to accomplish the commands defined in step 1. In this example, the lowest level servo agent control modules are represented by icons of the actuators being controlled. The steering wheel servo control module is represented by a steering wheel icon, the brake servo by a brake pedal icon, etc. For this simple example, only four actuator agent control module icons are shown.

The brake, throttle, and transmission servo agent control modules are grouped under a single supervisor agent control module, which will be termed the Speed Control Agent. This supervisor agent control module will receive commands such as Accelerate at some value. This module then has to coordinate its output commands to the brake, the throttle, and the transmission to accomplish this command. By a similar analysis, the Steering Wheel Servo Agent is placed under a supervisor agent will be termed the Steering Control Agent Module. This supervisor agent control module will receive commands such as TurnToAbsHeading and will compute the required steering wheel commands to output to the Steering Wheel Servo Agent.

The Vehicle Trajectory Control Agent receives commands in the form of constant curvature paths for the vehicle to be following. It continuously calculates the real-time vehicle dynamic motion vector and coordinates steering commands to the Steering Control Agent and speed commands to the Speed Control Agent to accomplish dynamically feasible moves.

The Elemental Maneuvers Control Agent receives commands of basic movement operations such as FollowLane, ChangeToRightLane, StopAtPoint, etc. along with a table of Objects-Of-Interest. These objects are

listed along with their positions, motion vectors, passing speeds, passing offset distances, cost-to-violate passing speed or offset, and other parameters that allow the Elemental Maneuvers Control Agent to calculate modifications to the commanded basic movement path to generate the output constant curvature paths that should successfully negotiate collision-free movement around these local objects. These calculated constant curvature paths are output to the Vehicle Trajectory Control Agent.

The Driving Behaviors Control Agent Module receives commands of basic drive behavior operations such as FollowRoad, CrossThruIntersection, TurnRightAtIntersectionTo-*roadname,* etc. This module detects and recognizes relevant vehicles and objects, reasons about the driving codes and the effect of the present state of the other vehicles and objects (such as pedestrians or traffic control devices – signal lights), chooses appropriate behaviors (such as PassVehInFront), builds the Objects-of-Interest table and outputs this table along with the next basic movement command such as FollowLane to the Elemental Maneuvers Control Agent.

The Route Segment Control Agent Module receives commercial trip planner-like (e.g. MapQuest®) commands of major route segment moves i.e. a single command that identifies a road to be traveled until an intersection with a designated crossing road where a right or left turn is to be made. This route segment may encompass a number of intersections that have to be crossed before the goal intersection is reached. These commands are of the form GoOn-*roadname*TurnRightOnto-*roadnam*e. This module has access to *a priori* maps, detects and reads road name signs, detects intersections and navigates at the level of commanding the Driving Behavior Control Agent to FollowRoad, CrossThruIntersection, TurnRightAtIntersectionTo-*roadname*, etc. This module also notes upcoming travel constraints such as lane restrictions (NoTrucks, ExitOnly, etc.) and upcoming turns and exits in order to command preferred goal lanes (for multilane roads) to the Driving Behaviors Control Agent Module.

The Destination Manager Control Agent Module receives commands in the form of end point destinations such as school, work place, gas station, post office, grocery store, etc. This module has access to *a priori* maps, driving/traffic history such as traffic conditions at different times of the day or under different weather conditions. This module also has access to real-time remote sensing through radio traffic, construction, accident, and weather reports. All of this information is continuously evaluated to develop a sequence of commercial trip planner-like (e.g. MapQuest®) route segments that map a path to the destination. If any remote sensing or local sensing by the vehicle (a backup is detected past the next intersection) indicates that a change in this route may be desirable, this module recalculates another set of route segments from its present location to the destination and immediately outputs the first route segment command of this new list to the Route Segment Control Agent.

As seen in Figure 3, the execution of the commands at the levels above the Vehicle Trajectory Control Agent are being executed by a single agent at each level since there are no multiple subordinate agent control modules to be coordinated at these levels in this simple example.

In a more detailed example implementation, there would be additional agent control modules for engine control, lights and turn signal control, windshield wiper/washer control, multiple pan/tilt turrets that carry different sensor sets, etc. This step 2 would be the point at which these additional organizational elements would be defined and laid out to properly coordinate their activities in accordance with the task decomposition descriptions from step 1.

2.5 Step 3: Defining and grouping the rules that define the procedural knowledge of the task decomposition

At this stage of the knowledge definition process, the vocabulary and syntax of commands have been defined. Also available are a preliminary representation of how each command decomposes into subcommands, and where in the agent control hierarchy these decompositions take place. Step 3 defines the process to establish the rules that govern each command's decomposition into its appropriate sequence of simpler output commands. These rules are discovered by first listing the approximate sequence set of simpler output commands for a particular input command and then by identifying the conditions that select when to execute these output commands. In this manner, each state table that governs the execution of each command at each level is defined.

Figure 4 illustrates this step with a state table of the PassVehInFront input command at the Driving Behaviors Agent Control Module. This PassVehInFront command is decomposed into five simpler output commands: FollowLane, ChangeToLeftLane, FollowLane, ChangeToRightLane, and FollowLane. These output commands came from the task decomposition tree for PassVehInFront identified in step 1 and their sequence is illustrated with the simple scenario drawing of the green car passing the blue car before the oncoming red car gets too close. This collection of input and output commands are assigned, in step 2, to the particular agent control module where this state table resides. These output commands are placed in the right hand side or Output Action column of the PassVehInFront state table. The knowledge that is being added by this step 3 is to identify and name each situation (Input State/Situation - the left hand column of the state table) that, when true, will transition the line to the

corresponding Output Action/Command. These named situations are the conditions that cause the branching in the task tree that results in the execution of this particular subtask (output command).

Each newly named state transition situation with its corresponding output action command represent a single production rule that is shown as a single line in the state table. The sequence that these lines (rules) are executed is ordered by the addition of a state variable ("S1", "S2", etc). In the example in Figure 4, the first rule shown in the state table says that if this is a "New Plan" (input condition), then the output action side of the rule (the right hand side of the state table) sets the state to "S1" and outputs the command to Follow Lane. As a result of executing this rule, this agent module is now in state "S1" and can only execute rules that include the state value of "S1" in their input condition.

The only rules that will be searched by this module are those in the particular state table that groups the rules relating to this particular input command (PassVehInFront). In this state table as shown, there is only one line (rule) that contains the state value "S1" as one of its input conditions. Thus, only that line can match and it will not match until the situation ConditionsGoodToPass is also true. When this situation occurs, this line will match (this rule will fire) and the module will go to state "S2" and will issue the output command to ChangeToLeftLane. This output command is sent to the subordinate agent control module (Elemental Maneuvers Agent Control Module) where it becomes that module's input command invoking a corresponding state table evaluation of that command at this subordinate level.

By this process, the large set of rules governing the task decision tree execution is grouped both by the agent control module in the hierarchy and by the task context of the particular command at each agent. This results in only a very small number of rules (i.e., only those that are relevant to the particular active command at each agent control module) being searched at any given time. Note that this knowledge discovery representation and organization have been completely driven by looking at the problem from the detailed task decomposition point of view.

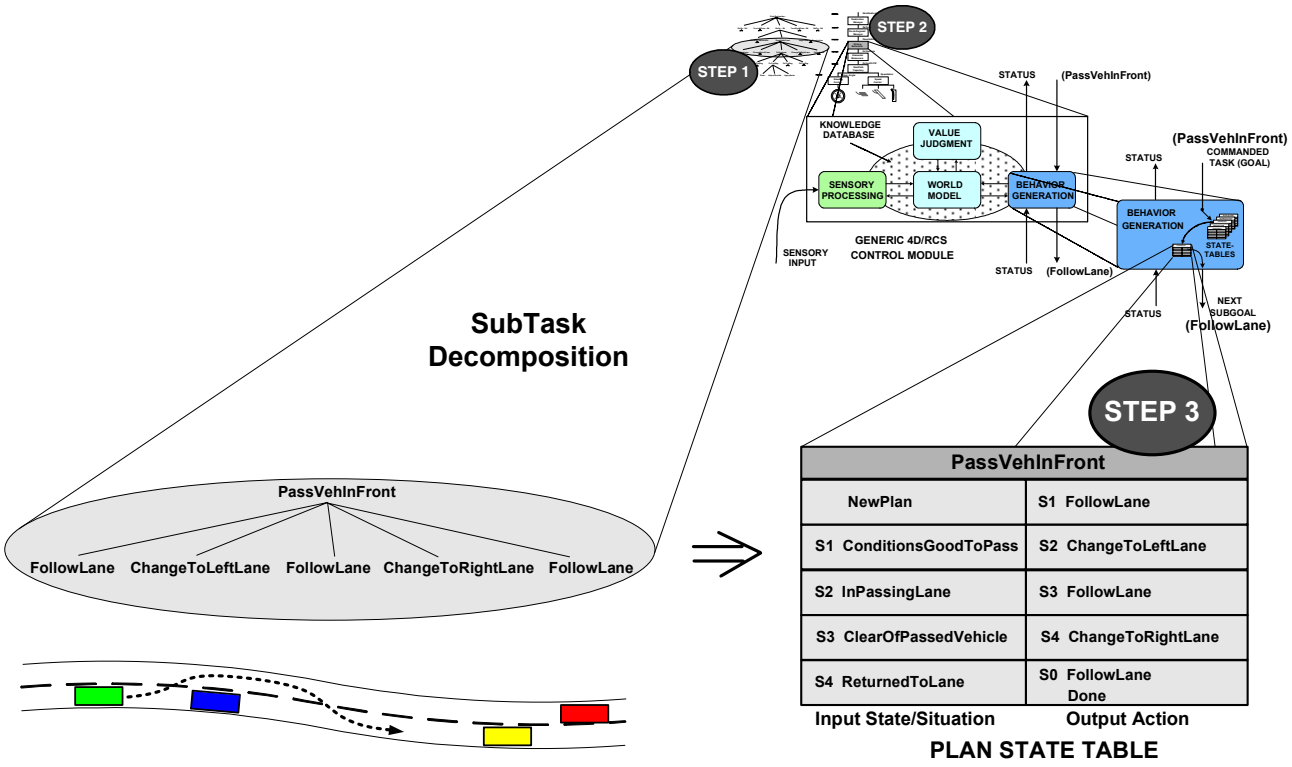


Figure 4. State table for the Pass Vehicle In Front command. Each line is a rule relevant to this task. The left column contains the branching conditions of the task decision tree. The right column contains the output commands or branches.

The branching shown in these state tables is of the transition type, i.e., the identification of a situation in the world that will cause a branching to a new activity – here when the situation ConditionsGoodToPass is true, transitionining to the action to ChangeToLeftLane from the action of FollowLane. Until this situation becomes true, the last action that had been transitioned to, namely, FollowLane will be the output. As part of this process, one should continuously ask what happens if the transitioning situation becomes untrue and none of the other listed situations are true. This often leads to discovering additional branches not uncovered previously. For example, if ConditionsGoodToPass becomes true and if transition is made to the ChangeToLeftLane action, what should be the response activity if ConditionsGoodToPass becomes false while ChangeToLeftLane is being executed?

Using this as an incitement to further explore this particular activity, experts are asked to come up with a scenario that illustrates this situation. In this new scenario, the passing maneuver (changing to the left lane) will have started, when the vehicle that is being attempted to pass suddenly starts to also move into the left lane to pass the vehicle in front of it (for example). The situation of ConditionsGoodToPass has just become false. In this scenario, the passing maneuver will have to be aborted to return back to the current lane. When the scenario is examined in detail, one could either be in the process of changing to the left lane or already be in the left lane and executing a FollowLane output action. From this, two additional situation-action pairs are identified – one for the case where own vehicle is still in the process of changing to the left lane and one for the case where own vehicle is in the left lane and trying to overtake the vehicle in front.

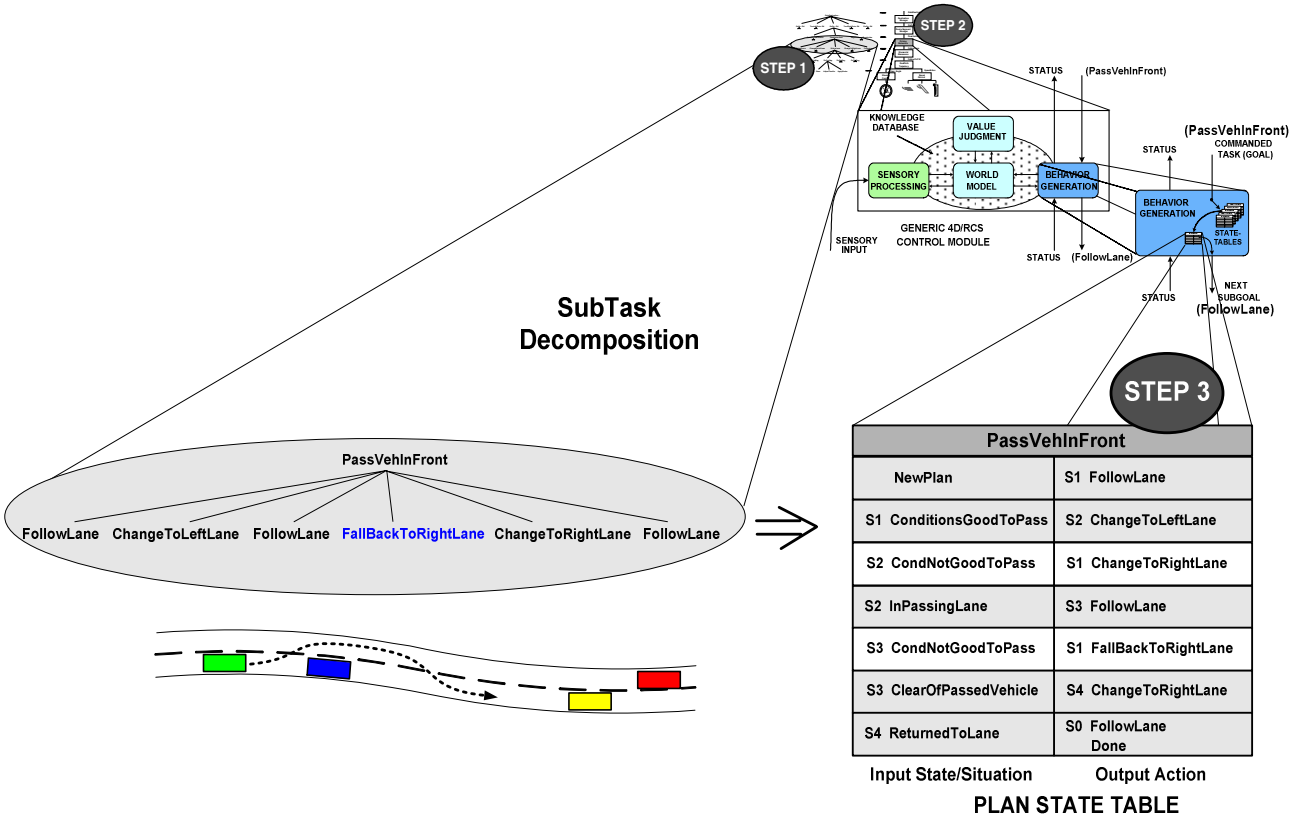


Figure 5. State table for the Pass Vehicle In Front command. A new subtask branch (FallBackToRightLane) has been added and the state table has had two additional lines added to represent the two cases where the pass has to be aborted due to CondNotGoodToPass.

A slightly different output action has also been identified, namely, to FallBackToRightLane which requires own vehicle to slow until it has sufficient re-entry room behind the vehicle that is being tried to pass before changing to the right lane. This action will be executed once own vehicle has already moved entirely into the left lane and has transitioned to the output action of FollowLane to speed pass the vehicle in front.

Figure 5 illustrates the resulting changes. An output action command (FallBackToRightLane) has been added in addition to two additional lines to the state table to deal with the two cases that have been identified where the

situation ConditionsGoodToPass becomes false after this situation has been already used to transition to a ChangeToLeftLane output action.

It can be seen that in both added lines, the return is to the state "S1" so that after the output action of either of these two lines has been accomplished, one will be in the correct logic state to once again start looking for the opportunity to pass.

Summary of first three steps

An important summary will now be made about the structuring of the knowledge base in these first three steps. These three steps were concerned with task knowledge expressed as the finer and finer branching of the decision process where each branch represents a different subtask activity relevant to the branch above it. This task branching was divided up and assigned to particular agents organized in a hierarchical structure consistent with how the various subtasks should be coordinated. Finally, the conditions or situations that must be true for a particular branch to be selected are identified and represented with the branch activity as condition-action pairs or rules. All of the rules that identify the sub-branches of one part of the tree are organized into a state table.

These first three steps provide a complete listing of the task decomposition rules (i.e. these rules that determine when the system has to do something different in order maintain progress towards the goal.) These rules have been grouped into layers of resolution (where they are contained within specific agent control modules), and within each layer, grouped into tables of rules relevant to a single input command. Within each table they are ordered in their execution sequence by additional state values.

These first three steps can be thought of as identifying the procedural knowledge involved in the task decision process, i.e. naming all of the task branching conditions and their selected output actions. The next three steps will identify all of the knowledge that is used to evaluate whether or not the branching conditions are true.

2.6 Step 4: Identifying relevant world states whose analysis and evaluation will provide the basis for the real-time situational assessment

In this step, the set of antecedent world states on which the task branching situations (in the input condition side of the state tables) depend in order to be evaluated are to be indentified. This is best illustrated with an example. Figure 6 shows the PassVehInFront state table. The output command to ChangeToLeftLane is issued when the ConditionsGoodToPass situation occurs as described above.

At this point in the process, the expert knowledge sources will be asked: "What does one have to know about the state of the world to say that ConditionsGoodToPass?" Again, a number of detailed scenarios are used to drill down to the parameters that go into this situation assessment. There is a very large set of states of the world that affect this situation. These are listed as they come up in different scenario discussions as well as the information that is discovered in driving manuals and driving codes. Examples of some of these are: "there cannot be an on-coming car within the passing distance", "there must be a broken yellow lane marker on own vehicle side of center in the lane", "there cannot be a railroad crossing within the passing distance", "own vehicle is not being passed by another vehicle", etc. These are referred to as world states since they seem to describe certain attributes about the present state of the world that are relevant to the present task.

Once the world states relevant to ConditionsGoodToPass are listed, they are checked to see if groupings can be made. In this case, some world states are grouped into a category termed LegalToPass, and others into other categories termed EnvironmentSafeToPass, SituationInFrontOkToPass, SituationInBackOkToPass, OncomingTrafficOkToPass, etc. These groupings are aggregate world states leading to the evaluation of the situation ConditionsGoodToPass. For this example, for this situation to be true, all of the aggregate world states have to be true. For each of the aggregate world states to be true, all of their component world states have to be true. This provides a classification of world states that aids in the understanding and discovery of additional relevant world states.

The purpose of this step is to provide a listing of all of the dependent world states that affect whether the task branching condition/situation identified in the first three steps is true or not.

The sensitivity of this situation to neither these world states nor what functions are used to weight and evaluate their individual or combined truth have not yet been identified. The identification of these dependent world states does not

rely on whatever technique is used to evaluate their contribution to the situation (such as ConditionsGoodToPass). Different implementation paradigms will determine this sensitivity, weighting, costing, and other evaluation functions as they see fit.

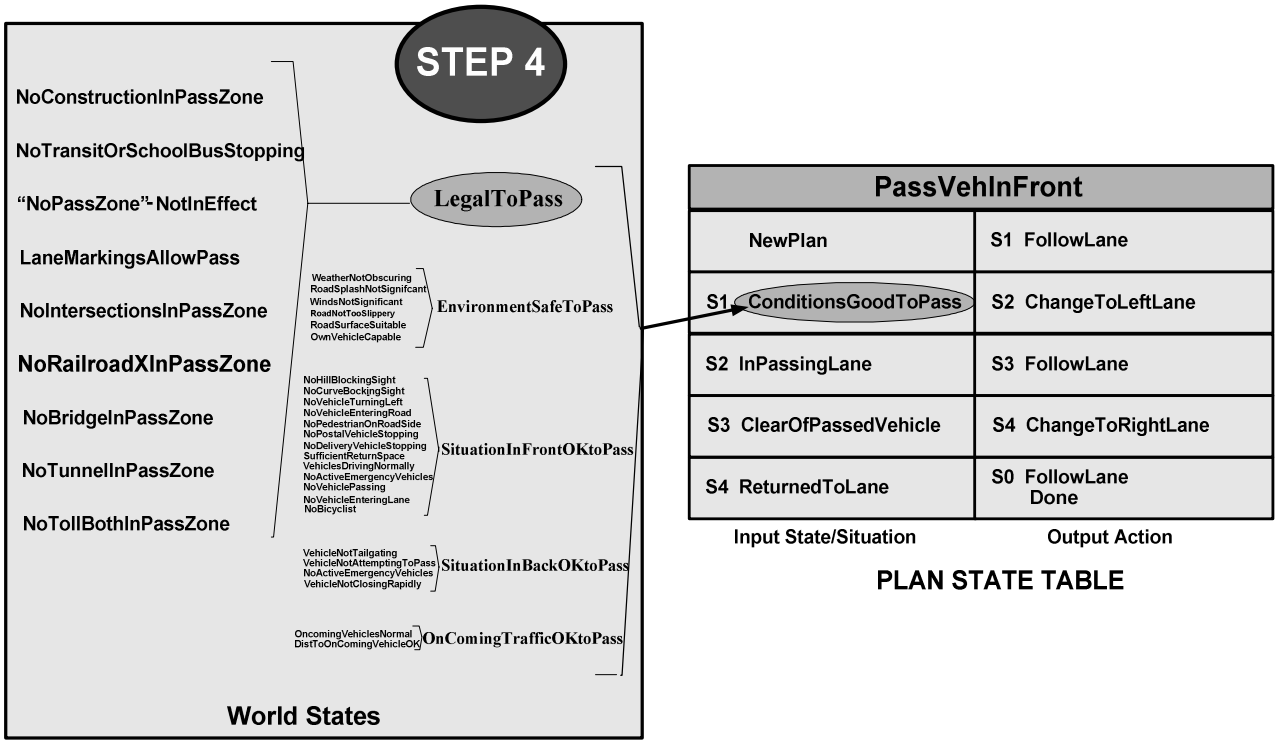


Figure 6. The identification of all of the antecedent world states used to evaluate whether the situation ConditionsGoodToPass is true or not.

2.7 Step 5: Identifying the objects, attributes, and events in the real world that create the relevant world states

This step identifies all of the objects, their features and attributes that need to be measured by the sensing system to create the world states described above. Figure 7 continues with the passing example. As described above, one of the aggregate world model states was LegalToPass which was a grouping of a number of related world states all of which deal with various legal restrictions on the passing operation. One of these component world states that identify a legal restriction is NoRailroadCrossingInPassZone. In this step, all of the objects, their features, and attributes that are relevant to creating each world state are to be identified. For the world state named NoRailroadCrossingInPassZone, these relevant objects would include the railroad crossbuck emblem, crossing lights, crossing gate, crossing signs either alongside the road or painted on the road surface, the railroad tracks, and the train itself. Further, each object is defined in terms of its characteristic features or attributes that will be used for recognition of the object (e.g. the width and length and height above ground of the crossbuck planks) and/or its state (e.g. flashing lights or a lowered gate as indicator of active warning state).

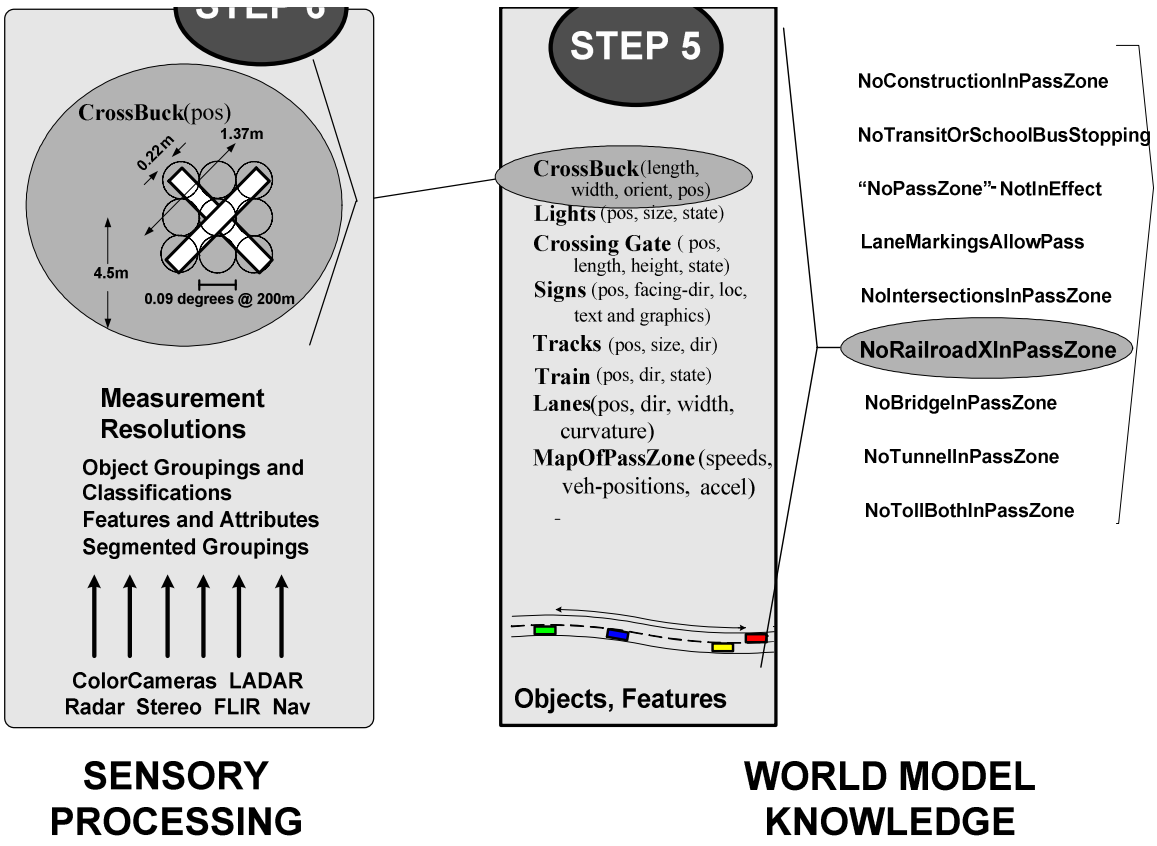


Figure 7. Example of the objects that are used to establish the NoRailroadXinPassZone world state and how the sensor measurement resolutions are determined.

2.8 Step 6: Deriving sensing capabilities required to measure and recognize the objects, attributes, and events

In this last step, the resolution requirements for the measurement of the objects by the sensors are to be defined. This is done by determining the expected recognition distances to these objects during particular subtask activities. In the case of the subtask activity of PassVehInFront, the ability to see objects such as the railroad crossing crossbuck sign at the far limit of the expected passing zone is required. For a vehicle attempting to pass another vehicle on a 75 km/h two lane undivided road, the passing zone could easily be 200 m or more. This means that the crossbuck, which is found at the railroad crossing itself, would have to be sensed and recognized by the sensory processing system at this distance of 200 m (Figure 7). Using this distance together with the knowledge of the size of the crossbuck plank elements, an estimate of the absolute minimum sensory processing capability required to recognize it at this distance can be made. If a minimum of a 3 pixel square to detect the crossbuck is assumed, this results in the requirement that the image sensors having a minimum resolution capability of 0.09°. This is an absolute minimum estimate and for recognition it is desirable to have considerable more pixels-on-target requiring even higher sensor resolution. But this process has provided a technique to establish these ranges of sensor resolutions for all of the various subtask activities.

This process also helps the identification of all of the other relevant objects and their relationships as they apply to each subtask activity. In this example, it is important to establish that the railroad crossbuck that is being detected is on the road of travel as opposed to a side road up ahead. So it is not sufficient to detect objects, it is also very important to establish the relationship of certain of those objects with one another (in this example, the crossbuck is located on the road of travel, not an adjacent road) for each particular subtask activity.

2.9 Discussion of example implementation

Application of the 4D/RCS engineering methodology to the problem of on-road driving has produced a control architecture of seven layers of agent control modules (including the lowest level steer, throttle, brake, and transmission agent control modules) with a total of over 170 command/plans, each of which is described by a state

table of Situation/Action rules. This is illustrated in Figure 8. For the RouteSegment and DriveBehavior agent control modules, commands are underlined and the plans that can be selected are listed under the appropriate underlined command. For example, at the DriveBehavior agent, the FollowRoad command can select a plan to PassVehInFront, or DriveOnTwoLaneRd, or DriveOnMultiLaneRd, or PullOntoRoad, etc. depending on the present world state. Each of these plans is a separate state table describing this task behavior with an appropriate set of rules. Each of these rules would have a branching condition/situation from which are derived detail world states, objects, and measurement resolutions. At the Subsystem, Trajectory, and Servo levels only the commands are listed.

A large number of commands and plans have been detailed out in their respective state tables, identifying the named branching condition/situations required. A number of these situations have been examined to specify their antecedent world states and objects. As a result, an initial estimate has been made of the number of knowledge items involved for autonomous on-road driving. These are approximately:

a) 1000 named condition/branching situations in the input condition side of the state tables.
b) 10000 world states that are antecedents to the situation evaluations.
c) 1000 to 2000 objects to be detected in the world to arrive at the world states.

Thus, on-road autonomous driving is characterized by a very large (but finite) knowledge set that would be extremely difficult to deal with if not for a systematic way to organize it.

The task decomposition approach has provided a single consistent process of well-defined sequential steps to both discover the relevant knowledge and to organize it. The knowledge has been partitioned into two large elements:

1) task procedural knowledge (i.e. how to do things) concerned with the description and representation of the task decomposition through layers of agent control modules performing control decision branching decisions, encoded as rules in state tables. This is basically the set of procedures to follow for every situation that occurs in on-road driving; this is an explicit set of knowledge represented in rules.
2) world knowledge (i.e. what is the present situation) concerned with the description and representation of all of the states of the world and all of the objects to be sensed that are used to generate each of the condition branching situations in each state table. This is basically the expert pattern recognition and reasoning knowledge that looks at the world and identifies/assesses the present situation so that a task decomposition procedure can be followed. World knowledge is a more intuitive set of expert knowledge and harder to get at. It is represented in the situation dependencies on world states and objects in the environment.

The representational format for all of the knowledge has been driven by a requirement to identify everything according to its relevance to task activities. This has a very important impact on the implementation. This organization of the task knowledge is in a form that can be directly implemented. It has threaded access to all of the knowledge from the task through world model states to objects and their attributes to be measured. This is exactly the form the control system needs so it can access all of the information relevant to the present task activity as rapidly as possible.

The definition of the knowledge base in this task-oriented format also supports a number of other processes, in particular, this knowledge base:

1) provides the basis for developing specifications and conformance tests for the procuring of complex autonomous systems,
2) provides the essential task detail to support writing Operational Requirements Documents (ORDs) and Operational and Organizational (O&O) plan documents,
3) provides the consistent detailed requirements definitions necessary for system developers to build robust, reliable control systems,
4) identifies where the project goals are limited by the present state of sensor and sensory processing capabilities, and clearly identifies those research and development areas where significant impacts can be made, and

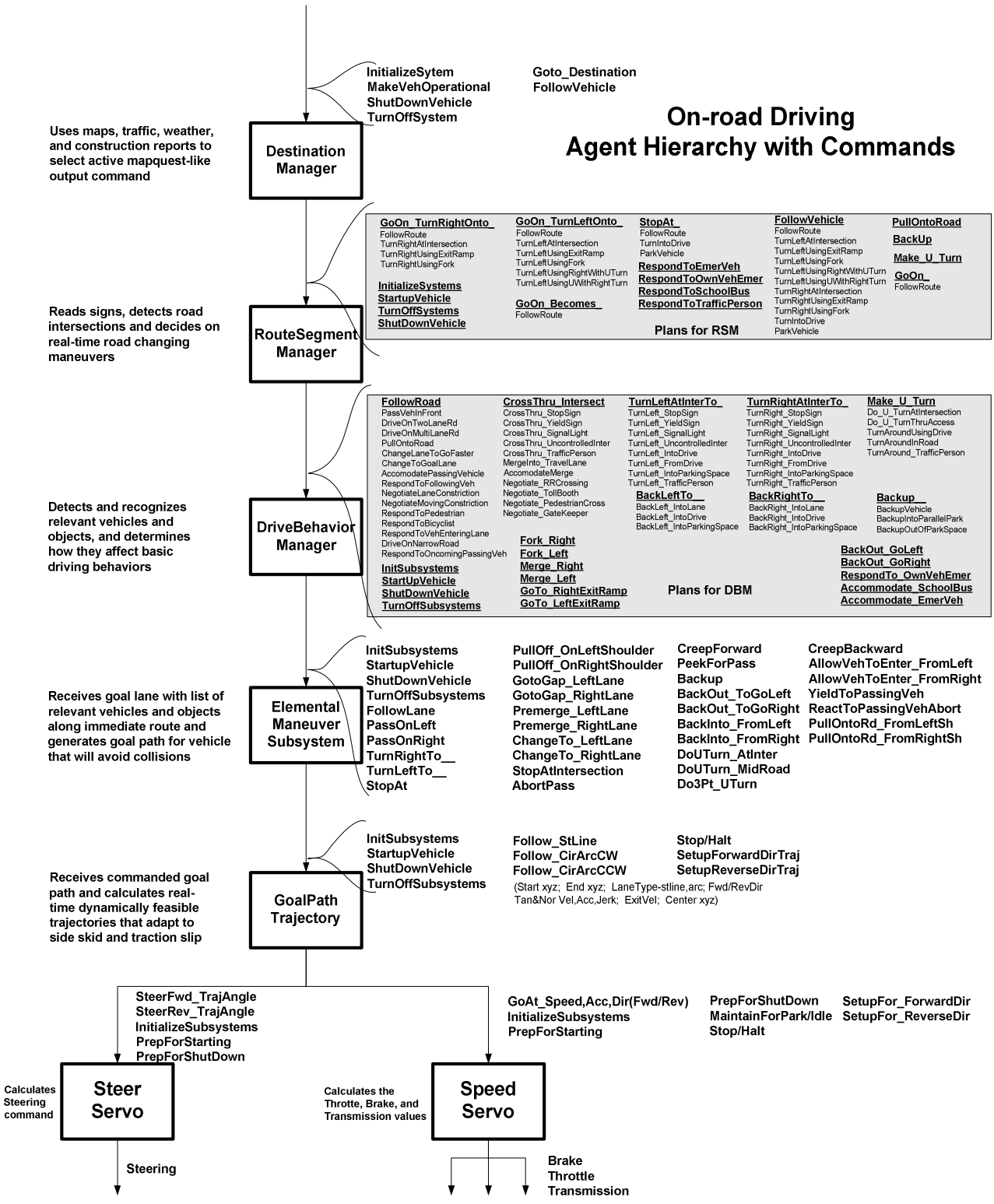


Figure 8. Illustration of the agent control architecture for on-road driving with the command/plan names listed and a brief description of the task responsibility at each level.

5) greatly aids in the creation of performance metrics [4] that can be used to accurately assess component capabilities at very fine levels rather than treating entire autonomous systems as black boxes.

2.10 Analysis of multiple vehicle tasks

This chapter has detailed the analysis of task activities for a single vehicle control system, namely, that of on-road driving autonomous vehicle. However, the hierarchical organizational structure of 4D/RCS nodes that will execute the various layers of subtask activities has been branched into separate subtrees (see the SteerServo and SpeedServo subtrees detailed above and in the Chapter 3 description of the NIST High Mobility Multpurpose Wheeled Vehicle (HMMWV) mobility implementation) where it is necessary to represent the control of actions of separate physical subsystems. The control of these separate subsystems can and does occur in parallel and has varying degrees of interlocked coordination depending on the particular subtasks.

Similarly, groups of separate physical subsystems can also be groups of vehicles. Two detailed sets of analyses of tasks involving group coordination and control of multiple vehicles have been analyzed in exactly the same manner as described above. These tasks have been:

1) Conduct Port Security (provide security for an in-foreign-port navy ship) by a group (eight) of Unmanned Aerial Vehicles (UAVs), and
2) Several Army missions (Conduct a Route Reconnaissance, Establish an Observation Post (OP), Conduct a Bridge Assessment) for a Cavalry Scout Platoon of either six (Bradleys) or 10 (HMMWVs) vehicles.

In both of these projects, the high level mission was analyzed and the appropriate commands to each and all of the subsystems were determined that included the coordination activities at each supervisory level where coordination of multiple subsystems occurred. The task decomposition of the Army missions starts with commands/operational orders to the top level node, equivalent to the scout platoon leader. This might be a mission/order to Conduct a Route Reconnaissance. Using the 4D/RCS task analysis approach, this mission/order is decomposed down through finer and finer task resolutions, branching to identify the commands to equivalent section leaders, and below that, to equivalent vehicle commanders (whose node is very closely associated with the RCS node that was defined as the Route Segment Manager in the on-road driving example above). However, for these systems the equivalent vehicle commander node would also be coordinating a Surveillance subsystem (equivalent to a soldier mission observer), a Lethality subsystem (equivalent to a soldier gunner), and a Communication subsystem as well as the Drive Behavior subsystem (which is equivalent to the soldier driver) that was described in the example in this chapter.

This analysis identifies the coordinating knowledge required for the supervisory nodes to determine the next appropriate commands to send to their subordinate nodes (platoon leader to section leaders, section leader to vehicle commanders, and vehicle commander to vehicle subsystems such as Surveillance, Lethality, Drive Behavior, and Communications). It details the indicators to be sensed in the environment to be able to evaluate the present relevant situations that are used to select the next actions, contingencies, commands, and reports for the proper coordination of the subordinates in accomplishing the supervisor's command.

Thus, the analysis makes no distinction for multiple vehicles. The analysis is an approach to discover the knowledge necessary to accomplish the goal by controlling and coordinating the available subsystem components whether these subsystem components are groups of vehicles, individual vehicles, or separate subsystems on a single vehicle.

2.11 Conclusions and Future Work

Having thus created this knowledge base that offers all of the above listed advantages, one is still faced with the problem that the knowledge base is extremely large and no single computer storage technique captures it. The 4D/RCS task analysis process has resulted in the knowledge being defined and documented in drawings, spread sheets, documents, data bases etc., but it is still stored as data in a computer, which means that it will require significant effort by the humans involved to retrieve and reassemble it into relevant knowledge views for different applications or even for accessing for future enhancements.

Current research is exploring the use of ontologies [5] to store this knowledge in a semantic model format in a computer-interpretable form. The hope is that ontology tools and techniques will provide a more consistent, single representational model through their descriptions of agents, processes, classes, properties, and inter-relationship definitions that will eventually be able to capture this knowledge and, most importantly, all of the implied

relationships in a single consistent computer-interpretable knowledge base. This super-ontology would allow the ability:

1) to add additional knowledge into the knowledge base and have it automatically check consistency with the previous knowledge and with the deep knowledge that expresses more basic concepts that the new knowledge descriptions have to obey;
2) to have it automatically retrieve responses to semantic based inquires;
3) to have it automatically construct executable models of the knowledge that can be used to drive simulations for operational testing and specification evaluation as well as the automatic development of training systems;
4) to have it automatically populate cost-based planning and evaluation algorithms; and
5) to have it automatically build out and update intelligent control systems.

References

1. J. Albus, et.al. 2002, "4D/RCS Version 2.0: A Reference Model Architecture for Unmanned Vehicle Systems," NISTIR 6910, National Institute of Standards and Technology, Gaithersburg, MD.
2. E. Mettala, D. J. Cook and K. Harbison, Application of the Scenario-Based Engineering Process to the Unmanned Ground Vehicle Project, in *Reconnaissance, Surveillance, and Target Acquisition for the Unmanned Ground Vehicle* , O. Firschein and T. Strat (eds.), 1997.
3. J. McKnight, and B. Adams, *Driver Education Task Analysis. Volume 1. Task Descriptions*, Human Resource Research Organization, Department of Transportation, National Highway Safety Bureau 1970.
4. A. Barbera, J. Horst, C. Schlenoff, E. Wallace, D. Aha, 2003, Developing World Model Data Specifications as Metrics for Sensory Processing for On-Road Driving Tasks. In *Proceedings of the 2003 PerMIS Workshop.* Gaithersburg, MD.: NIST Special Publication 990.
5. C. Schlenoff, R. Washington, A. Barbera, 2004, Experiences in Developing An Intelligent Ground Vehicle (IGV) Ontology In Protégé, presented at the 2004 Protégé Conference in Washington, D.C.

Chapter 3

Behavior Generation

Stephen Balakirsky, Tom Kramer, Fred Proctor, and **Tony Barbera**
National Institute of Standards and Technology (NIST)
{stephen.balakirsky,thomas.kramer,frederick.proctor,tony.barbera}@nist.gov

3.1: Introduction

Previous chapters in this book have outlined a comprehensive reference model architecture for the control of an autonomous system and illustrated how a complex problem may be decomposed to fit into this task hierarchy. This chapter will attempt to provide insight into how autonomous behaviors may be elicited from such a system through the use of the behavior generation (BG) portion of the RCS echelon. Each RCS echelon contains a BG process which, as shown in Figure 3.1, contains a Job Assigner (JA) and one or more agents. Each of these agents may be further decomposed to contain one or more planners and an executor.

The JA performs four functions within BG. It accepts input task commands from an executor in a higher level BG process, decomposes the input task into job assignments for each planning agent within the BG process, transforms each job assignment's coordinate frame to the agent's preferred frame of reference, and allocates resources to the agents to allow them to accomplish their assigned jobs.

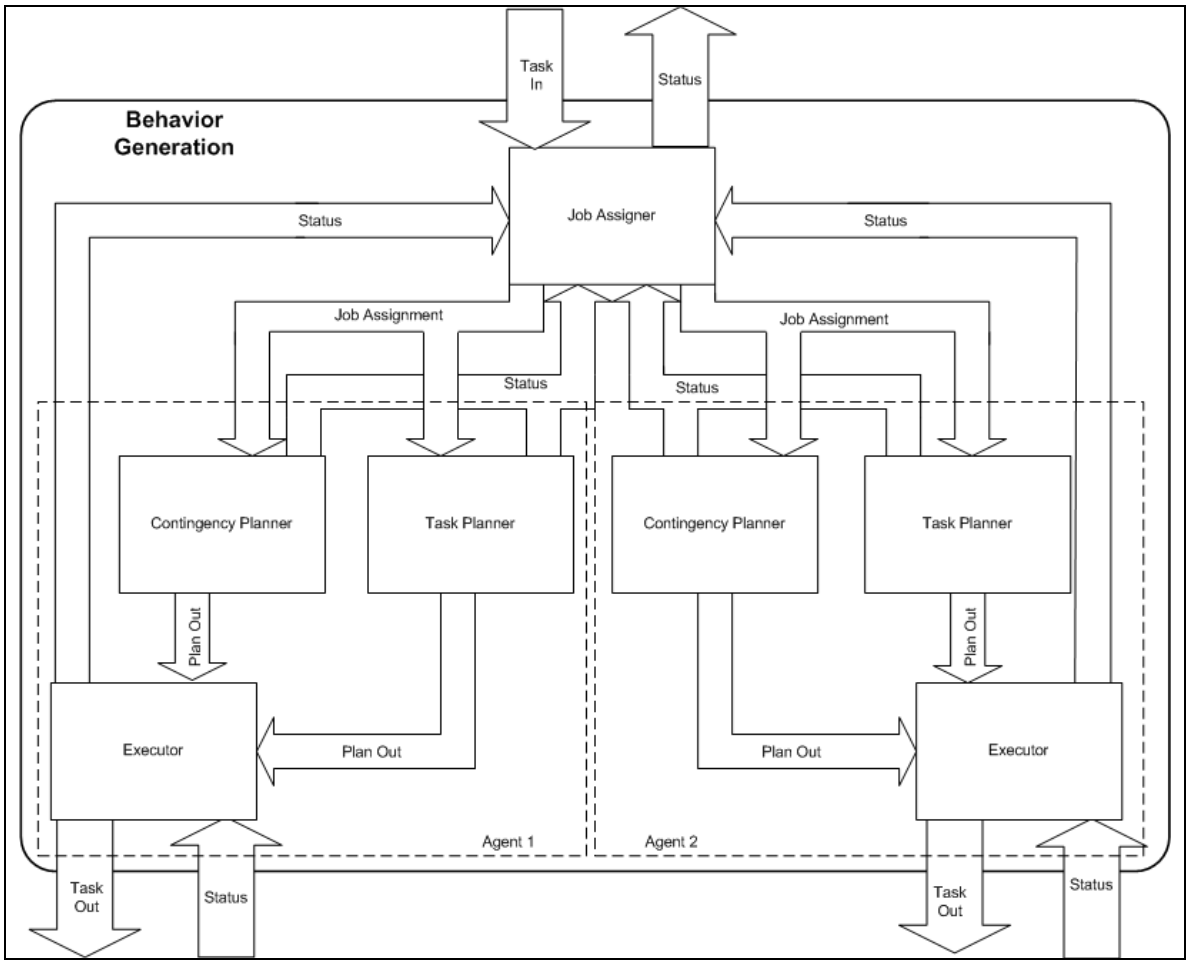


Figure 3.1: The BG portion of an RCS_NODE.

The task planner accepts a job assignment from the JA and computes one or more sequences of activities that accomplish the assigned job. In parallel to this operation, the contingency planner computes a set of contingency plans that may be executed if abnormal or unexpected readings are obtained from the world model. For example, the autonomous mobility echelon BG of an on-road driving system may compute a sequence of constant curvature arcs to traverse in order to meet its superior's goals in the task planner and an additional set of constant curvature arcs that would bring the vehicle to a safe and controlled stop on the road's shoulder in case of an emergency. The best sequence of activities from each planning system is passed to the executor (EX) which selects the appropriate plan to execute and then executes all or a portion of that plan.

4D/RCS supports several modes of behavior generation including reactive behavior, off-line planning, and real-time planning. Reactive behaviors strive to embed the control strategy into a collection of pre-programmed reactions (sense-action mappings) that are very similar to human reflexes [5]. This approach provides a direct, constant-time response to the sensed environment, which requires an expert to isolate each possible combination of sensor output and map them to actions. Under the 4D/RCS approach, reactive behaviors are most often used to correct small control errors and in response to emergency situations. The reactive behaviors are controlled by the EX which is constantly evaluating the predicted state of the world against the sensed state of the world. If the error between these two becomes too large, the EX is able to swap out the current plan for one of a number of pre-computed reactive plans or actions that are stored in a plan library, or a deliberative plan from the contingency planner.

4D/RCS and its predecessor RCS have long supported off-line planning. In fact, Versions I and II of RCS used off-line planning exclusively [2]. Through the 6-step process illustrated in Chapter 2, off-line plans may be devised to accomplish many different classes of tasks. These off-line plans consist of a carefully designed state graph or state table with branching conditions that represent alternative actions that are triggered by environmental stimuli and events. The resulting plans define one or more paths through the state space and consist of a string of actions and a string of resulting states. In general, the actions and resulting states may be viewed as subtasks that will be passed through the hierarchy. At each hierarchical level, this planner "computes plans that extend from the anticipated starting state out to a planning horizon characteristic of that level. On average, planning horizons shrink by about an order of magnitude at each lower level" [2].

Real-time planning consists of the process of performing a real-time search through the system's state space in order to devise an optimal or near-optimal sequence of activities to perform in order to accomplish a set of goals. As 4D/RCS matured, this class of planning was incorporated for such tasks as vehicle route planning, and the resulting state sequences have been viewed as subtasks for lower levels of the hierarchy. The remainder of this chapter will focus on new techniques for integrating real-time planning into the 4D/RCS architecture and with off-line planning systems. Three real-time planning techniques will be discussed along with a novel cooperative planning mechanism that will operate in the place of task decomposition between 4D/RCS hierarchical levels. Finally, a discussion of a behavior generation system that has been implemented on NIST's High Mobility Multi-purpose Wheeled Vehicle (HMMWV) will be presented.

3.2 BG Interfaces

In order for any hierarchical system to function properly, there must be well defined interfaces between the various levels of the hierarchy. Under a traditional off-line planning task decomposition scheme, a large amount of prior work is devoted to the creation of a vocabulary that is transmitted between levels. Under this scheme, a received command is decomposed through the application of a pre-computed plan into a series of activities that will accomplish the given command. Each of these activities is then sent to a lower-level off-line planner that repeats this process. This scheme bottoms out at a planning level whose output controls the actual hardware. This form of planning may be viewed as plan refinement; the highest level planning system creates the framework of the plan to be executed, and this framework is then refined by adding details and complexity throughout the hierarchy.

3.2.1 Task Decomposition Planning

A detailed example of a task decomposition for on-road driving was presented in Chapter 2. The depicted vocabulary attempts to define a set of commands that incorporates all of the activities at all levels of detail. With this planning technique, the high-level planner determines a course of action for the system to follow from the system's current state through its goal state. For example, the vehicle will "GoOnRoad-*name1*", then "TurnLeftOntoRoad-*name2*", etc. The lower level planning system is then left to compute how the vehicle will travel on the road, but is no longer free to decide which road to travel on. For this technique to function properly, the high-level planner must have information over a sufficient extent to compute the initial complete plan.

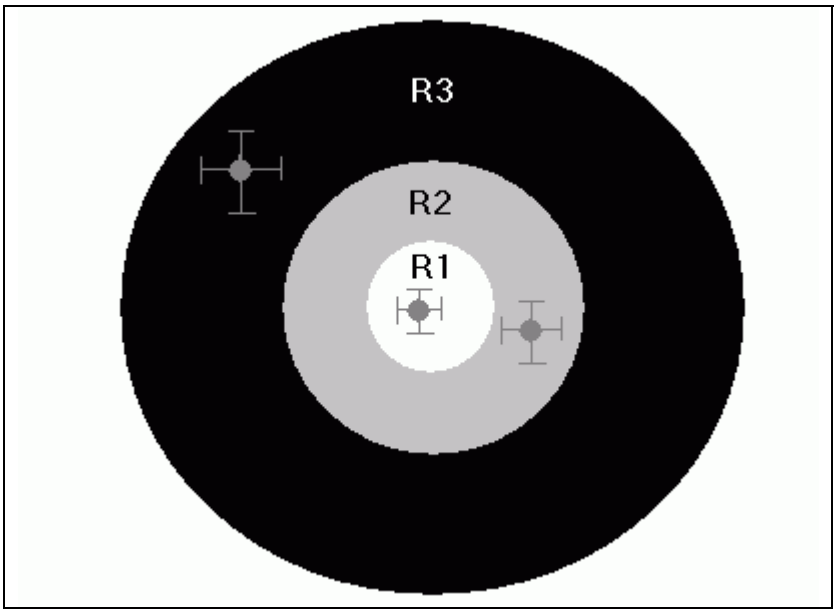


Figure 3.2: Example of feature precision for 2D mobile robot application. R1 is coverage region of high-precision sensor, R2 is coverage region of lower precision sensor, R3 is region of *a priori* data. The small filled circles represent a detected feature, and the lines are the location error bars.

Real-time task decomposition

For many real-time off-road planning systems, the world model is realized as an occupancy grid whose resolution is formed by examining real-world phenomena. In these systems, areas near the robot (region R1 in Figure 3.2) are observed by sensors that are able to provide high precision of feature locations. Further from the robot (region R2 in Figure 3.2), wide area sensors detect large objects and the corresponding location precision decreases. Finally (region R3 in Figure 3.2), once the limits of the sensors have been exceeded, the represented features correspond to the features and precision represented in an *a priori* dataset. As the data precision varies with range from the vehicle, so do the storage requirements on the occupancy grid. Since the entire cell of an occupancy grid is labeled with any particular feature, the cell should be sized large enough to contain the location error bars, but small enough to preserve the sensor precision. Therefore, the occupancy grid may be viewed as requiring high precision near the robot with decreasing precision as the distance to the robot increases. Forming regions of equal precision as shown in Figure 3.2 can be used to create the basis of a hierarchy.

If this hierarchy is treated as a task decomposition hierarchy, a BG system operating in region R3 will compute a plan from the vehicle location to a planning horizon that lies in R3. This plan will be based on low-precision information from region R3 combined with summary data from regions R2 and R1. The R1 and R2 data is summarized to the precision of region R3 in order to reduce the computational burden on the BG system. A subset of this plan (extending from the vehicle location to a planning horizon in region R2) will then be refined by the echelon R2 BG which will compute a higher-precision plan that roughly follows the R3 plan and is based on the higher precision information from region R2 combined with summary data from region R1. Similarly, the region R1 BG will compute a plan that roughly corresponds to the R2 plan and extends from the vehicle location to its planning horizon.

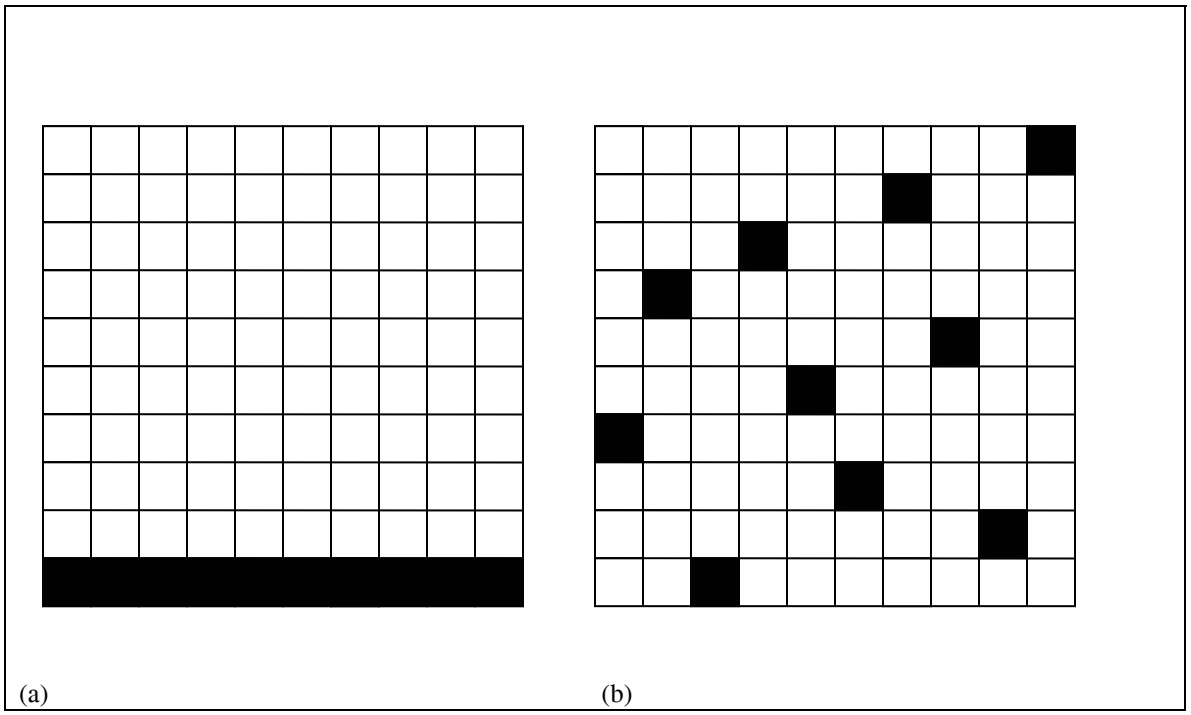


Figure 3.3: Sample high-resolution grid squares that compose a single low-resolution grid.

Of major concern with the task decomposition technique is that data must be summarized. Careful attention must be paid as to how to accurately represent the data summary. For example, in Figure 3.3 both of the low-resolution cells depicted contain an identical number of occupied high-resolution cells (shown in black), however for a robot that that is able to traverse through a single unoccupied high-resolution cell, the high-level binary summary should be "occupied" for (a) and "free" for (b), or from another viewpoint, cell (a) will be highly traversable from left-to-right (a straight line path exists) and will be untraversable from top to bottom while cell (b) is moderately traversable (a path with turns exists) in any direction. If the robot is two or more cells in diameter, then cell-to-cell boundary conditions also become an issue.

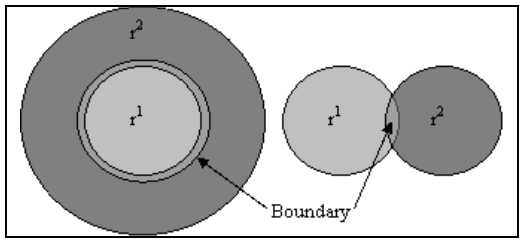


Figure 3.4: Planning region boundaries.

3.2.2 Cooperative Planning

By shifting the focus from a task-decomposition style of interface to a cooperative planning interface, the need for cell summarization can be totally eliminated. Rather than having each planning system plan from the vehicle location to its planning horizon (thus requiring summarization of high-precision data), a cooperative planning system allows each hierarchical echelon to plan in the region that matches its native precision. This final plan may be seen as a cooperative plan that is stitched together by a series of boundaries. This boundary is defined as an interface region between two or more planning systems.

Boundaries may be hierarchical as depicted in the left-hand side of Figure 3.4, or between two planning systems functioning at the same echelon as shown in the right-hand side of the figure. In a hierarchical system, each planning system will typically have two boundaries; one with its supervisor and one with its subordinate. In this case, each of the planning systems is responsible for computing the section of the plan that is between its boundaries.

Graph-based planning

Cooperative planning as defined here relies on the use of a graph-based planning technique such as uniform cost search (described below). Graph definitions are depicted in Figure 3.5. Graph-based planning relies on the existence of path segments that are directed from one location (or state) to another in the system's state space. These path segments may be thought of as the *arcs* of a *directed planning graph*. If it is possible to have identical bi-directional connections between pairs of states, then the two arcs may be replaced by a single *edge*. A graph that contains only edges is referred to as an *undirected graph*. The end points of these edges then become the *nodes* of the graph, and the start point may be referred to as the *root node*. Any node that has no successors is called a *tip node* or *leaf node*. The act of exploring an edge that connects two nodes is referred to as *expanding* that edge. Additionally, the nodes connected to the current node under consideration are referred to as that node's *spanning set* or *children* and each of these children possess a *back-pointer* that names its parent. It is often desirable to assign a positive *cost* to an arc that represents the cost of the corresponding action that causes the node transition from parent to child.

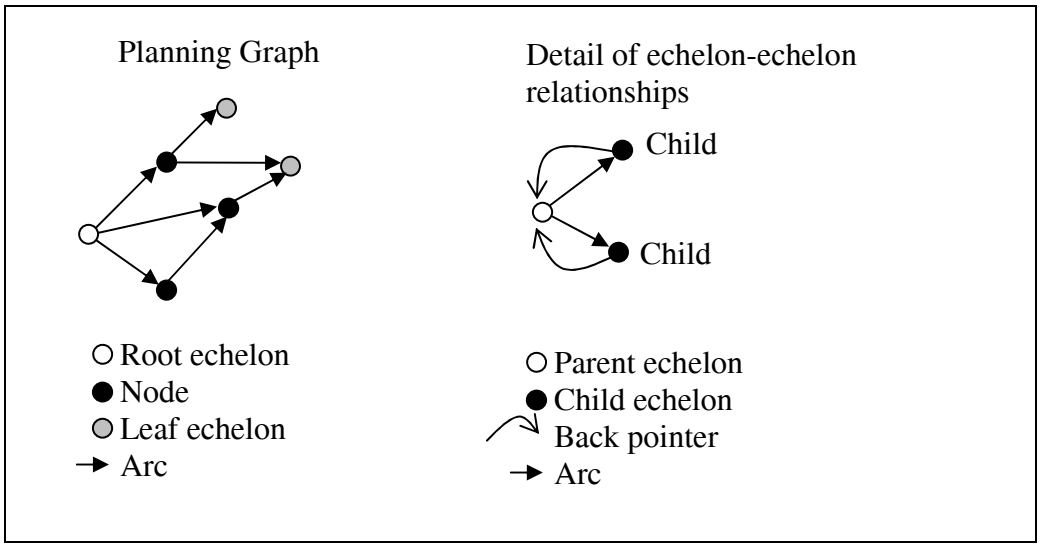


Figure 3.5: Graph space definitions.

Once the possible path segments through the space have been defined and the plan graph has been constructed, a plan is formed by finding a combination of these segments that connects the system's starting state to the system's goal state. This may be accomplished by simply following the back-pointers from the goal to the plan origin. The *path cost* is then defined as the sum of the individual arc costs that constitute the path. The path that has the minimum cost between two nodes is referred to as the *optimal cost path* [12].

Graph search provides a set of techniques for finding a path (or optimal path) from the start node to the goal node. Uniform cost search is an uninformed search strategy developed by Dijkstra in 1959 [9] that finds an optimal path through a graph with respect to an arbitrary cost function. This algorithm begins by fully expanding the start node. During the expansion, the cost of the edge connecting the start node to each of its children is computed and assigned to the child node. Each child is then placed on a list of partially evaluated nodes known as the *open list*. The search now moves to the least expensive child from the open list (denote this node n_2), which is removed from the open list and is fully expanded. The costs of each edge connecting n_2 to its children are then computed. For each child, if the cost of the connecting edge plus the

cost of n_2 is less then the child's current cost (which is initialized to infinity), the child is assigned this cost, its back-pointer is set to n_2, and it is added to the open list. This procedure is now repeated with the expansion moving to the new lowest cost node. The search terminates when the goal node is removed from the open list. This search technique will produce a path that is both complete and optimal provided that the path cost function is non-decreasing.

Implementation

Cooperative planning may be implemented in a two-step approach in a hierarchical real-time graph-based planning system. In step 1, a "prepare to *x*" (where *x* is some activity) command originates at the highest level of the hierarchy and propagates down to lower levels. In response to this command, the lowest level planning system uses a graph search technique such as uniform cost search to compute a partial plan graph that originates at the system's current state, and has computed cost values for all of the level's boundary nodes. These cost values are sent in a status message to the next higher level and placed in this level's open list. Placing these nodes in the superior's open list has the effect of "seeding" the superior's planning graph with costs. Instead of starting from a single point (the system's current state), the graph search now proceeds from the set of states that were determined by the subordinate. This level then continues the development of the planning graph until its entire set of boundary nodes have costs associated with them. This procedure continues until the planning terminates with a cost being assigned to the system's overall goal at the highest planning level.

Figure 3.6: Example of multiple planning regions (one inside the black box and the other outside).

Since the boundary region is shared by two or more planning systems, it is possible that a planning level may find a cheaper way of achieving its subordinate's shared boundary node than was reported by the subordinate. If this occurs and optimal planning is desired, the subordinate must incorporate this knowledge into its partial planning graph which may result in new, lower values being found for some set of boundary nodes. This may be accomplished by having the superior send an "alter graph" command to the subordinate along with a list of nodes whose values have been affected.

An example of this is shown in Figure 3.6 which shows an actual exit ramp from a highway. When planning in isolation, the subordinate located on the north-south running highway will not be able to find any way to achieve the east-west road that contains the goal **X**. The only path that lies entirely in its planning space would require the system to jump off of the bridge or drive off-road. However, the superior will find a less expensive way of achieving the boundary node 'b' that permits access to the goal road (taking the off-ramp from the boundary node 'a' to the boundary node 'b') thus updating the subordinate's search space and allowing the subordinate to plan a viable path to the goal road. A complete study of the handshaking necessary for optimal planning with this technique may be found in [6].

Once a path to this goal has been determined, the graph back-pointers from the goal may be followed to find the boundary node that is part of the plan. A "perform *x*" command is then sent down the hierarchy (step 2 of the procedure) where the command contains the boundary node that the subordinate system should strive to achieve. Through this technique, each planning system is uniquely responsible for planning in its area of expertise, and a jointly optimal path may be found.

Communication Channels

Each of the BG nodes described in the remainder of this chapter functions as an independent finite state machine (FSM). Communication between superior and subordinate is supported over two sets of matched command and status communication channels. The first set of channels is used to communicate commands to the subordinate and status back to the superior. At a minimum, each node supports commands that allow for:

- Initialization – places the system in a known initialized state.
- Halt – commands the system to perform an orderly cessation of activity.
- Abort – commands the system to cease activity immediately.
- Shutdown – commands the system to power-down.

In addition to these commands, each node accepts node specific commands that are germane to the node's planning level (e.g. drive to goal command for the high-level BG). When a new command is received over the command channel, execution of the previous command's FSM is immediately terminated, and execution of the new command is begun.

At times, it is desirable to alter system parameters without interrupting the currently executing FSM (e.g. it may be necessary to change the planning cycle time to aid in CPU load balancing). An additional set of command/status channels known as the configuration/settings channel is provided for this use. These channels run their own independent FSM that interacts with the primary FSM without destroying its internal state. Interlocks may be utilized to protect areas of shared data from corruption during sensitive areas of execution.

3.3 Rule Based High-level BG

The highest level in the control system is called the vehicle level. This level provides control (in some cases optimal) for the achievement of future goals while factoring in hard and soft constraints on the control strategy and real-time constraints on performance. In addition, it must operate under challenges such as dynamic environments, user objectives, and system goals. In this sample system, it operates in an area about 100 m across with a cycle time of about one second.

Dynamic, or seemingly dynamic environments occur in all but the simplest planning cases. These dynamics may be caused by newly sensed environmental features that are inconsistent with prior knowledge (e.g. seemingly dynamic events such as a mobile robot which has sensed information missing from its a priori map), or by actual events that alter the environment (e.g. moving objects). In either case, these changes may prove catastrophic for a deliberative system if its pre-computed plan is not altered.

Dynamic user objectives occur when a user changes the way in which a particular behavior is to be performed after the system has already begun operation. This change may take a form that the system designer has previously anticipated, or may place new, unexpected requirements on the system. Finally, dynamic goals occur when the system must not only change the way in which a behavior is performed, but

must also change the final goals of the behavior, and may even need to alter which behavior or set of behaviors is currently active.

For the vehicle level, the plans must do more than successfully find a path that will move the robot from point 'a' to point 'b'. The robot must exhibit intelligence in the manner in which it performs this movement. For example, a robot that is only capable of on-road travel must be able to plan a route that follows roads, while an all terrain terrestrial rover may need to follow the contour of a particular interesting rock formation.

To compute these plans, the vehicle level utilizes a graph-based planning algorithm, and the planning process may be decomposed into the potentially concurrent phases of graph creation and graph search. The determination of the node spanning sets that are used for graph construction and the evaluation of the arc cost function that is used for graph search is performed in cooperation with the vehicle level's world model process.

3.3.1 World Model

The World Model (WM) is responsible for maintaining a model of the features of the environment and vehicle self and for converting this information into a form that is usable by BG for the creation and evaluation of the planning graph. Information from the WM is used in the determination of the spanning sets of graph nodes and in the determination of the cost of arcs.

The WM is made up of multiple feature layers. Each of these feature layers may be viewed as an expert system that is capable of providing advice to BG in the areas of node spanning set generation and arc cost evaluation. The number of actual feature layers is a design decision that is driven by a combination of feature independence and complexity. The layer design must balance having the layer contain the proper information to provide valid feature based planning information, while being able to provide this information in a time frame that is suitable for real-time operation. For example, a layer that is in charge of determining conformance to traffic laws will also need to contain information on road markings to provide correct results. However, including obstacle information in the same layer produces additional complexity without adding benefit to its traffic law conformance decisions. The WM may be further decomposed into a Knowledge Base (KB) and Plan Simulator (PS) for each layer and a Value Judgment (VJ) module.

3.3.2 Knowledge Base

Each of the layers in the WM contains its own KB that maintains information relevant to the decisions made by the layer. The KB is responsible for providing the PS with the information necessary to perform valid simulations of states and state transitions for use in the construction of the arc cost function, and for providing VJ with the information necessary to construct node spanning sets.

In order to perform these duties, the KB contains an implementation specific knowledge representation whose specifics are driven by the class of data being represented. Example representations include rule bases, cell-based scrolling maps, or skeleton-based scrolling maps.

Rule Base

Some forms of knowledge may be compactly represented by a set of rules or equations. This may be seen in boundary equation representation techniques for Cspace [11,14] and rule bases that govern domain independent planning [4]. Rules may be used as soft constraints for the evaluation of the validity of state-to-state transitions (in the arc cost function) and for hard constraints in the incremental expansion of a planning space (the selection of a node's spanning set).

Layers that contain rules based on predicate logic have been constructed to work in several different planning environments including domain independent planning and the solution of the towers of Hanoi

[5][1]. In these layers, rules are used in both graph expansion (what states are members of the spanning set of state *x*) and graph edge traversal (what cost factors are incurred as a result of this transition).

Hybrid rule-based and cell-based layers have also been designed for both on- and off-road driving environments [7,13]. Under these environments, a rule-base is used in conjunction with a cell-based scrolling map that is updated by the sensor processing system. Graph expansion is controlled through the application of a rule base that is used to determine node spanning sets. The selected nodes are then mapped into the cell-based scrolling map where graph edge traversal is simulated for the determination of arc cost. The information available in the cell-based map is also made available for use during rule evaluation. This provides a hybrid structure that allows for the seamless integration of cost-based and rule-based information sources.

Cell-based Representation of a Layer

An example of a composite view of a knowledge layer based on cell decomposition is shown in Figure 3.7 for a mobile robot application. This figure shows an *a priori* layer that contains static knowledge about the environment.

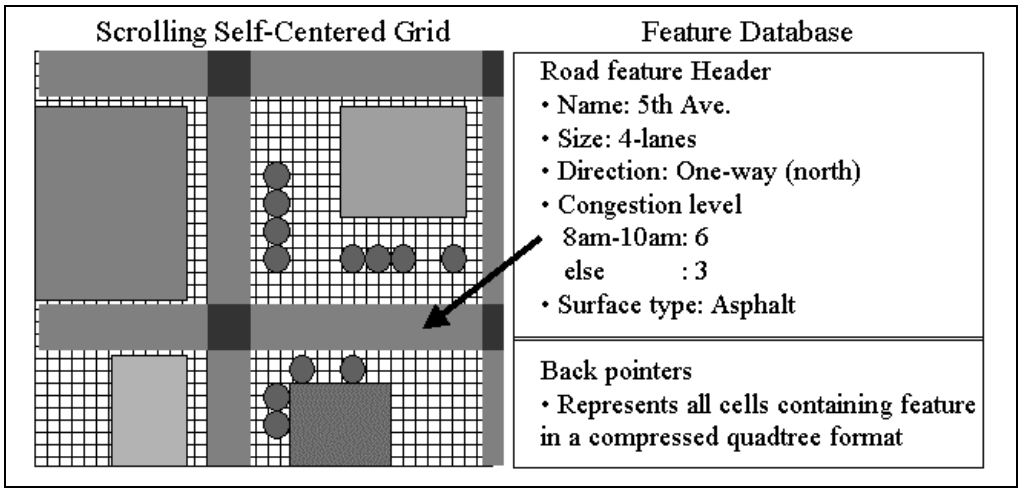


Figure 3.7: Structure of composite cell-based component of Knowledge Base.

The basic form of the layer is a combination of a regular, *n*-dimensional grid of cells that represents the system's discrete state space with regard to the layer's feature classes and a database of specific feature instantiations. Each cell of the grid contains a representation of the layer's features that are contained in the cell, and a link to a relational database containing specific details of each contained feature. The 4D/RCS architecture strictly governs the resolution and extents of the feature information required by the vehicle level, thus allowing a limited size grid to be implemented.

In order to operate in this limited size grid, the system's current state is always maintained at the center of the grid. As the system's state changes, the grid is scrolled to maintain its centered position. This scrolling has the possibly undesirable effect of causing data to be lost off of the edges as the layer scrolls. In order to prevent this loss of data, KB layers may implement a function that summarizes this data and passes it up to WM at the next level of the hierarchy (which has a greater spatial extent at lower resolution).

[1] The Tower of Hanoi puzzle was invented by the French mathematician Edouard Lucas in 1883. The starting configuration of the puzzle consists of a number of uniquely sized disks stacked in increasing size on one of three pegs. The objective is to transfer all of the pegs to another tower while moving only one disk at a time and never placing a larger disk onto a smaller one.

Skeleton-based Representation of a Layer

Not all information can be easily expressed as cell-based data. In some instances, for example vector data such as road networks, a skeleton representation may be more appropriate. For this approach, the vector data is maintained as a sorted list of skeleton vertices. As in the cell-based layer, skeleton-based layers consist of a representation of the layer's features that are contained in the vertex, and a link to a relational database containing specific details of each contained feature. Unlike the cell-based layer, where connectivity may be assumed (each cell is connected to its neighbors), the skeleton-based layer must also contain explicit connection information.

Since memory usage is also a concern for skeleton-based layers, the list of vertices is maintained so that the system's current state is at the center of the cloud of vertices as judged by some distance criteria. For the example of a road network, this distance criterion may be a simple Euclidean distance of the geographic location of the vertex. As the robot moves, vertices that are deemed to be too far from the robot are disconnected and removed.

Relational Database

Each cell of a cell based layer or vertex of a skeleton based layer contains a link into an object oriented or relational database for each of its contained features. For example, a cell or vertex that contains the feature "road" would have a link to a specific road object that contains information about the road (speed limit, number of lanes, etc.). All cells or vertices that contain a piece of that specific road share this specific road object. In addition, as shown in Figure 3.7, the road object contains a link to each cell or vertex that contains the road. This representation allows for the identification of all cells/vertices that contain a specific feature.

3.3.3 Plan Simulator

A PS resides with each KB. The PS is responsible for determining the validity of potential children in a node's spanning set and for determining the consequences of traversing a graph arc. All simulations are performed with respect to the information contained in an individual KB. The results from all of the PSs are then combined by the VJ module.

Validity checking of potential children provides a mechanism for reducing the size of a node's spanning set and thus, the size of the planning graph. For example, it may be determined that a road driving robot is capable of occupying a location in an opposing lane of traffic. The potential child would be evaluated by the KB in charge of rules-of-the-road as not valid. Assuming that this is undesirable, a hard planning constraint could be implemented by eliminating any children that are judged not valid by this KB. This would make it impossible for the planning system to plan to go to such a location.

However, in some cases it may be necessary to cross over into opposing traffic. This action could be allowed, but strongly discouraged, through the arc evaluation process. In this case, VJ would allow the node to be added as a child, but when the PS evaluates the connecting arc and designates it in violation of rules-of-the-road, the cost could be made prohibitive. This arc would only be selected for execution only if no better alternative is available. In addition to simulating node validity, the PS simulates the consequences of the traversal of each arc. These results are passed onto VJ which translates consequences into cost.

3.3.4 Value Judgment

VJ is responsible for combining the results from the multiple plan simulators into a single response. These results consist of the children of each node in the planning graph and the cost of each arc traversal. The results are obtained by applying behavior and planning objective specific rules to the results of the PS simulations. To date, very simple if-then-else rules have proven sufficient to elicit road driving behavior from simulated robotic vehicles. It is envisioned that more complex rule engines will be necessary to fully comply with complex rules-of-the-road and mission objectives.

3.3.5 Planning Engine

The interactions of all of the components of the vehicle level BG are coordinated by a graph search engine. The current search engine used is a Dijkstra searcher as outlined in the previous section. This search engine operates in the following loop:

1) The current cheapest node from its open list is sent to all of the PS modules for evaluation. Each PS examines the state information associated with the node in order to determine the node's children. Grid-based layers may simply suggest the nearest neighbors for inclusion, while skeleton-based layers may perform a visibility graph analysis in order to determine reachable nodes.
2) A validity check is then performed on all potential children by all of the PS modules. These results are sent to VJ which collates the results and creates a final spanning set for use by the search engine.
3) The arc cost of reaching each child must now be computed. Individual PS modules once again compute simulation results that are passed back to VJ for final analysis. The rules used for simulating the arc transition may be changed on a planning-cycle-by-planning-cycle basis. In fact, they may even be tied to individual attributes of a node. For example, the traffic-law conformance layer may examine the time that an arc will be traversed and the number of vehicle occupants in order to determine conformance with High Occupancy Vehicle (HOV) rules.
4) VJ combines all of the simulation results and determines a single final cost of each arc. The rules that govern the cost determination may also be varied on a cycle-by-cycle or arc-by-arc basis.
5) The cost of each child is determined and the children are added to the open list.

The loop then repeats itself until (1) an optimal solution has been found, or (2) a non-optimal solution has been found and the planning time limit has expired, or (3) no solution has been found and an error must be returned. If a valid plan is found, then this plan is sent to the executor for execution.

3.4 Cost-based Mid-level BG

The middle level in the control system is called the Autonomous Mobility (AM) level. It performs cooperative hierarchical planning, and in this example, executes moves within a square ten meters on a side centered on the vehicle, and has a cycle time of about 0.2 s. At this level, plans are made to move from the current location of the vehicle to points on the perimeter of the square, commands are received to move from the current location to some point on the perimeter, and commands are issued to move on an arc or line from one point of the square to another.

At the AM level, consideration is given to vehicle dynamics, cost-based real-time planning is performed, and a relative (or local) coordinate system is used.

3.4.1 AM Level World Model

The world is modeled using two grid-based maps, one for obstacles, the other for elevation. The resolution of the maps has been set at 0.2 m during development of the system, but it is a parameter that may be changed. RCS theory [1] also provides for symbolic objects such as roads edges and ditches at this level, but symbolic objects are not yet included. The obstacle map indicates the difficulty of driving over a patch of ground if it were level. It represents the effects of trees, rocks, lakes, cliffs, fences, holes, land mines, local bumpiness, other vehicles, etc. The elevation map records the average elevation of each grid square. The elevation map is used for computing the slope of the ground for use in considering vehicle dynamics.

The maps are expressed in a "local" coordinate system. The distinguishing features of the local coordinate system are: (1) that any object is located by finding its offset from the vehicle location and adding the offset to the location of the vehicle, and (2) that the vehicle location is tracked from moment to moment by adding the offset occurring between moments. In particular, vehicle position is never adjusted by reference to any device such as the Global Positioning System that provides absolute location. Because location by adding offsets always drifts over time, the location in absolute coordinates of the origin of the local system also drifts over time, and the location of objects that were perceived a long time ago is likely to have a large

error. Only those things that were perceived recently are located accurately. The local coordinate system works well for AM level BG because the vehicle sensory processing system can sense everything inside the ten meter square in a short time, and things outside the square are not remembered.

3.4.2 Using Vehicle Dynamics

The world model in the AM planner does not provide a level of detail sufficient to support calculation during planning of dynamic vehicle behavior such as bouncing, sliding, and tipping. Devising and implementing algorithms capable of accurately calculating vehicle behavior given sufficient input data is difficult if not impossible. If software were in place to perform the calculations given sufficient data, it is certain that sensory processing and world modeling would not provide sufficient data at the level of detail needed by the algorithms. For example, the compactability of every patch of dirt over which the vehicle might drive would be required. If sufficient data were also available, sufficient time would not be available since many alternative courses of action are considered during planning, and each would require dynamic analysis. The AM planner, therefore, considers dynamics qualitatively. The paths it produces are nearly physically realizable, and the rules it uses to produce the paths are elementary.

In some control systems, the interface between levels provides commands that may be physically impossible to carry out. For example, an upper level language for controlling a vehicle might include a command to set the speed to a constant value and a command to travel in a straight line from one point to another. When successive straight line commands are given going in different directions, the path, in principle, should have a sharp corner where the two lines intersect. It is physically impossible, however, to make a sharp corner at constant speed; the acceleration would be infinite. The lower level controller that receives the commands deals with this by having an implicit path following tolerance, and slowing down at the corner if necessary to achieve that tolerance.

The AM controller deals with the sharp corner problem by never including one in the paths it generates. All paths consist of sequences of constant curvature segments (arcs of circles and straight line segments) that join smoothly end to end.

Paths that are smooth as just described look good, but are still impossible to follow exactly. At each juncture where constant curvature segments meet, to follow the path exactly, it would be necessary to turn the steering mechanism instantaneously from one direction to another (or change drive wheel speeds instantaneously), and that is impossible. The error introduced by this impossibility, however, is much less than that caused by a sharp corner. The AM planner helps to minimize the error by generating paths in which only a modest change in the position of the steering mechanism (or in drive wheel speeds) is needed to go from one path segment to the next.

The content of the command to move along a constant curvature segment was determined with dynamics in mind. The speed with which to move along each segment is part of the data about the segment provided by the command, as is the tolerance with which the segment must be followed. The AM planner commands only modest changes in speed between segments since instantaneous changes are impossible.

Perhaps the most obvious qualitative dynamics requirements of driving are to go slower on tight curves than on wide curves and to adjust speed on a curve according to how much the curve is banked. The AM planner observes both of these.

Where the vehicle must go through a gap almost as wide as the vehicle, the vehicle is slowed down. This is partly because of dynamics (a small error in heading could lead to a large path following error at high speed, causing a crash), partly because of control (control error is usually less at low speed), and partly because of safety (a low-speed crash causes less damage).

3.4.3 Path Planning Commands and Methods

The AM mobility BG process may be commanded (1) to plan, or (2) to move the vehicle from where it is through a series of waypoints (usually only one), or (3) to do both at the same time. A basic principle of the architecture used in the system described in this chapter is that each controller executes only one command at a time. It is necessary to have a command to plan and move simultaneously (and not feasible to use two

separate commands) because if a command to plan were given while a plan to move was in progress, starting work on the plan command would wipe out knowledge of the status of the move, and if the planning command were not given until the move was completed, it would be impossible to have continuous motion.

The command to plan means: find the cheapest path from the vehicle's current position to each of 40 points on the periphery of the world model square and report the cost of each. Using 40 points allows one point for each meter of the periphery of the square, which gives one point on the edge of each vehicle level cell that is adjacent to the square. The periphery of the square is the boundary between levels shown in Figure 3.4. The command to move requires that a plan for the move be made and executed. If only a short time passes between a plan command and a move command, it might be feasible to use one of the 40 plans already made, but that requires remembering the 40 plans and keeping track of how much time has elapsed since the plans were made. Hence, when a move command is received, a new plan is made and its execution is started.

Except when a goal has been reached, it is anticipated that a plan, which might take several seconds to execute, will not run to completion. Rather, on the next planning cycle (a fraction of a second after plan execution starts), the plan will be replaced by a new plan.

The AM planner does not perform its work instantaneously. Therefore, a mechanism needed to be developed that would allow the vehicle level to know when a plan had been developed to reach a particular periphery point. Since the cost of reaching a point can never be negative, when given a command to plan, the AM planner initializes the cost of reaching each of the 40 periphery points to a known, negative value. This value then serves as a flag indicating that no plan has yet been made. The vehicle level can check the progress of planning by looking at the flags and, if planning is not complete, decide whether to allow planning to continue or to issue a move command.

As already described, the AM planner is a grid-based planner. A plan from one point to another is produced initially as a series of lines from the center of one cell to the center of another cell.

Two quite different approaches to planning a path from one point to another are used: rubber banding and A*. Rubber banding (described below) is tried first. If rubber banding fails to make a good plan, or if time permits and more confidence in having an optimal solution is desired, A* is used.

In rubber banding, a path straight from the current location to the goal is constructed and examined to see if it passes over unacceptably steep slopes or through high-cost areas such as those containing obstacles. If so, the path is stretched away from them (like a rubber band) into less costly or less steep areas and the cost is recomputed. The stretch and check cycle is repeated either a fixed number of times or until the cost divided by the distance from the current location to the goal falls below some threshold. Rubber banding will provide a good solution quickly if the surface is uniform over the planning area and obstacles are sparse. If those conditions do not hold, rubber banding is not expected to provide a good solution. Rubber banding never provides a guarantee of optimality. Rubber banding is illustrated in Figure 3.8.

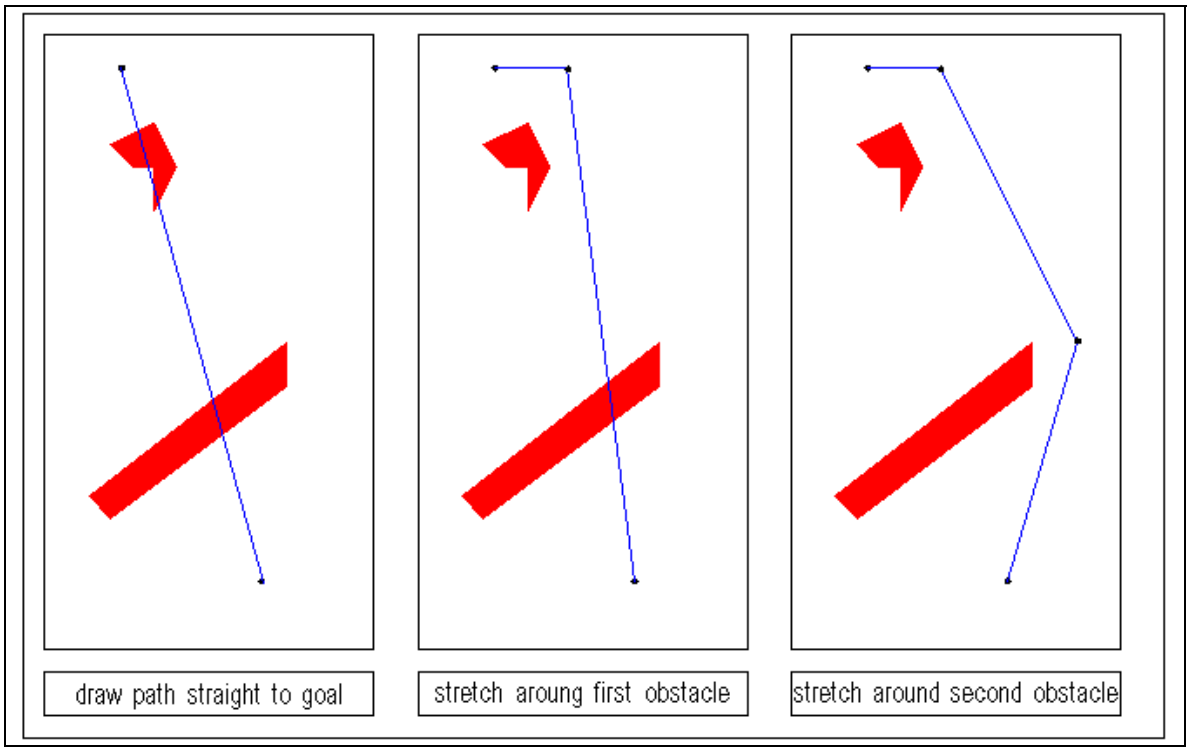


Figure 3.8: Rubber banding.

A* search [10,17] is a graph-based planning method similar to the Dijkstra method described in Section 3.2. A* extends the Dijkstra method by arranging the list of open nodes according to estimated total cost, where the estimate for a node is the sum of the cost to get to the node plus an estimate (called the heuristic) of the cost to get from the node to the goal. As long as the heuristic always provides an underestimate, the search is guaranteed to find a lowest-cost path to the goal if there is any path. The heuristic used in the AM planner is the Euclidean distance in meters from the current location to the goal multiplied by the least possible cost per meter traveled.

To enhance the performance of the A* searcher, the list of open nodes is maintained as a heap [15,16], and the heuristic is calculated during the search only for those nodes for which it is needed. The graph used by the AM planner in A* search has one node for each cell. Arcs of the graph go only from each cell to its eight immediate neighbors. As a result, plans made by A* search consist entirely of short straight line segments through the center points of cells.

After the path has been generated as a sequence of straight line segments, it is remade as a smooth curve through those points, having a continuous second derivative at every point (i.e., the curve is C2 continuous), so that it could be followed without instantaneous changes in steering. It is checked that the cost of the smooth curve is not much different from the cost of the lines. Finally, the smooth curve is approximated by a smooth path consisting of constant curvature segments. To make it physically possible to follow the path closely, the constant curvature segments are selected so that only small changes in speed and curvature occur from one segment to the next. This last approximation may be done using relatively few segments and with sufficient accuracy that it is not necessary to recheck the cost. The three stages of path planning are shown in Figure 3.9.

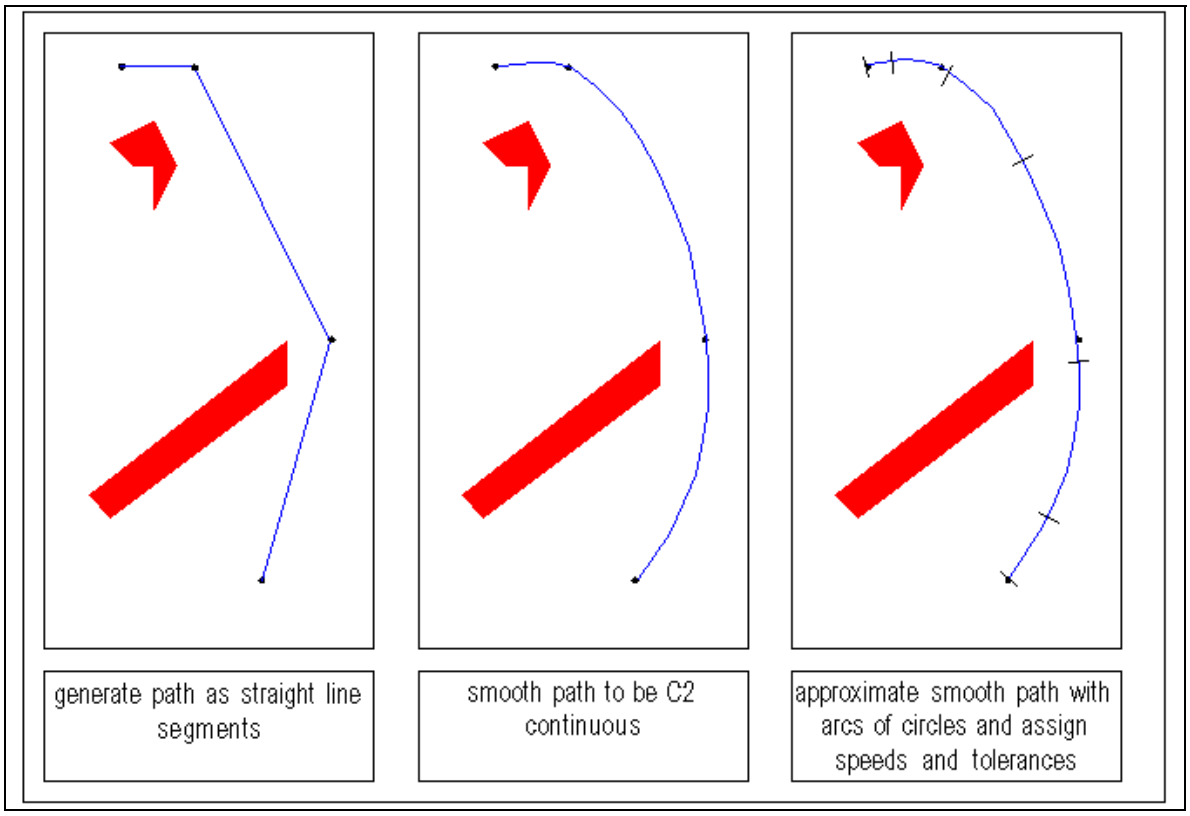


Figure 3.9: Three stages of AM path planning.

To execute a plan, commands to follow the path segments given in the plan are sent to the Primitive (Prim) level. Each Prim command describes a circular arc or a straight line segment with a width tolerance and a speed. The Prim controller is expected to keep the vehicle within the given tolerance of the desired path and to drive the vehicle at the given speed. The Prim controller may adjust the speed to keep the vehicle within the given tolerance.

3.4.4 Cooperative Planning Issues

Two interesting issues arise from the use of cooperative planning: using the same cost measures at different levels of planning, and dealing with paths that pass out of the AM planner's square and then come back in again. The latter was discussed in Section 3.2.

The vehicle level planner and the AM level planner must assign costs in the same way or cooperative planning cannot be said to be optimal, and strange effects may occur. For example, if the vehicle level planner assigned a high cost per distance traveled to driving in streambeds, while the AM planner assigned a low cost, the vehicle level planner would tend to keep the vehicle away of streambeds outside the AM planner's square, but whenever there were a streambed inside the square, the AM planner would tend to drive into it.

The two planners must use the same costs for the same expense items. Even this is not sufficient, however. The AM planner assigns costs to some items the vehicle planner does not think about. For example, the AM planner may assign a cost to moving through a narrow gap between small obstacles that may not appear at the vehicle planner's level of resolution. Also, the AM planner may be able to make a more exact cost calculation than the vehicle level planner since it knows the exact path. The AM planner sees small-radius turns and can assign costs to them while the vehicle planner may not see small-radius turns. It may be useful for the vehicle planner to add in an "expected small-radius turn cost" as a sort of overhead charge. There may be additional items for which the overhead charge method is useful.

3.5 Adaptive Low-Level Behavior Generation

3.5.1 The Primitive (Prim) Level

The Primitive or Prim Level is responsible for planning a series of velocity states given a list of constant-curvature arc moves with tolerances on width and desired tangential speeds. Prim derives its name from the robotics notion of primitive moves, e.g., short move segments that make up a pick-and-place operation by an industrial robot arm. The arc list is represented in the vehicle's relative coordinate system. These velocity states are sent to the subordinate Servo Level for tracking. Prim models the vehicle dynamics so that only feasible velocity states are planned. Dynamics include forces due to accelerations, friction, and tipping induced by the ground gradient. In general the problem is over constrained. For example, the desired speed for a small-radius constant-curvature arc move may result in centripetal forces that tip the vehicle. The velocity constraint is always relaxed in favor of geometric constraints (e.g., the arc radius) since the path geometry is presumed to be correct in the presence of obstacles. That is, a tight move is not made wider in order to maintain speed; speed is reduced to maintain tight turning.

Planning is complicated by the unpredictable performance of the vehicle. For example, the vehicle may have slipped off the nominal circular arc path due to wet conditions. Prim continually monitors the navigation state from sensors such as a GPS receiver or an Inertial Navigation System (INS) to determine if the vehicle is within the width tolerance for the arc. If not, Prim plans a move directly toward the arc at some safe speed in order to prevent collisions with obstacles presumed to lie outside.

Within Prim, arcs are converted to a series of closely-spaced waypoints based on the width tolerance. The tolerance sets a neighborhood around each point within which the vehicle is free to move toward the next waypoint. For the next target waypoint, Prim computes the deviation in heading, and sets the vehicle velocity and angular velocity according to a tunable cutoff angle θ_c. Translational speed is reduced linearly from its desired tangential speed v_{max} at zero heading deviation to some minimum speed v_{min} at the cutoff angle, and clamped to the minimum beyond that. Angular speed is set to zero when there is no heading deviation, and increases linearly to its maximum at and beyond the cutoff angle. In particular

$$\omega = \max(-\omega_{\max}\theta/\theta_c, -\omega_{\max}) \quad \theta \geq 0$$
$$\omega = \min(-\omega_{\max}\theta/\theta_c, \omega_{\max}) \quad \theta < 0$$
$$v = \max(v_{\max}(1 - |\theta|/\theta_c, v_{\min})$$

where θ is the angular difference, θ_c is the cutoff angle, v_{max} is the desired tangential maximum speed, and v_{min} is the minimum achievable speed from the constraint $v_{\min} = \omega / r_{\min}$ for vehicles with a minimum turning radius r_{min}.

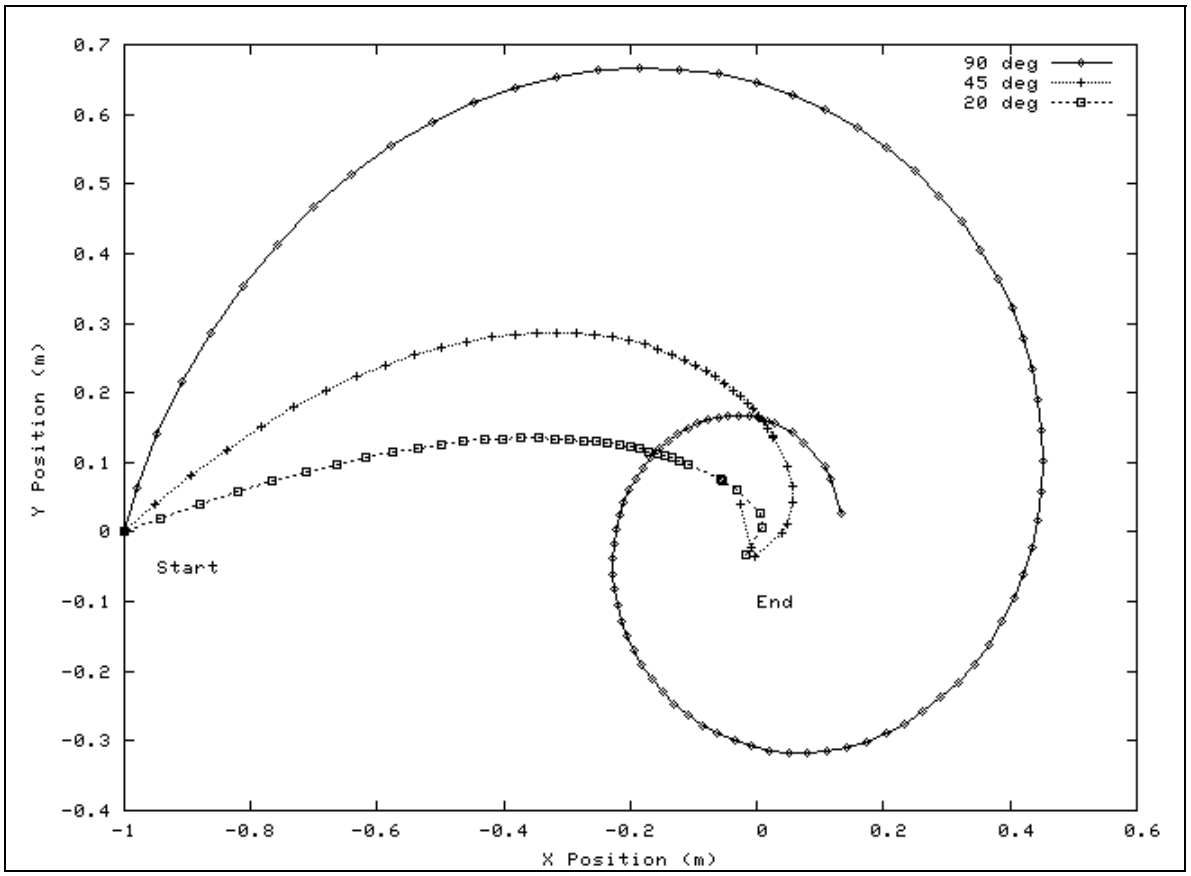


Figure 3.10: Trajectories for various values of the cutoff angle θ_c. Initially the vehicle is pointed perpendicular to the goal direction, and rotates in place until its heading lies within the cutoff angle. Note that for large cutoff angles, the vehicle takes a broad sweeping path toward the goal, which is undesirable. For small cutoff angles, the path is more closely aligned with the straight-line path to the goal.

For wheelchair-like vehicles that have a zero minimum turning radius, the minimum translational speed can be zero. Figure 3.10 shows the behavior of such a vehicle when given a single goal point far away in both distance and heading. The three paths show the track of the vehicle for various values of the cutoff angle. For heading deviations outside the cutoff angle, the translational speed is zero and the vehicle rotates until it is pointed toward the goal, at which point the translational speed picks up and moves the vehicle. For larger cutoff angles, the vehicle is free to move when it is pointed far from the goal, and wide deviations are seen. As the cutoff angle is made smaller, the vehicle never deviates far from a direct line to the goal, although the vehicle speeds are smaller. This is shown in Figure 3.11.
s

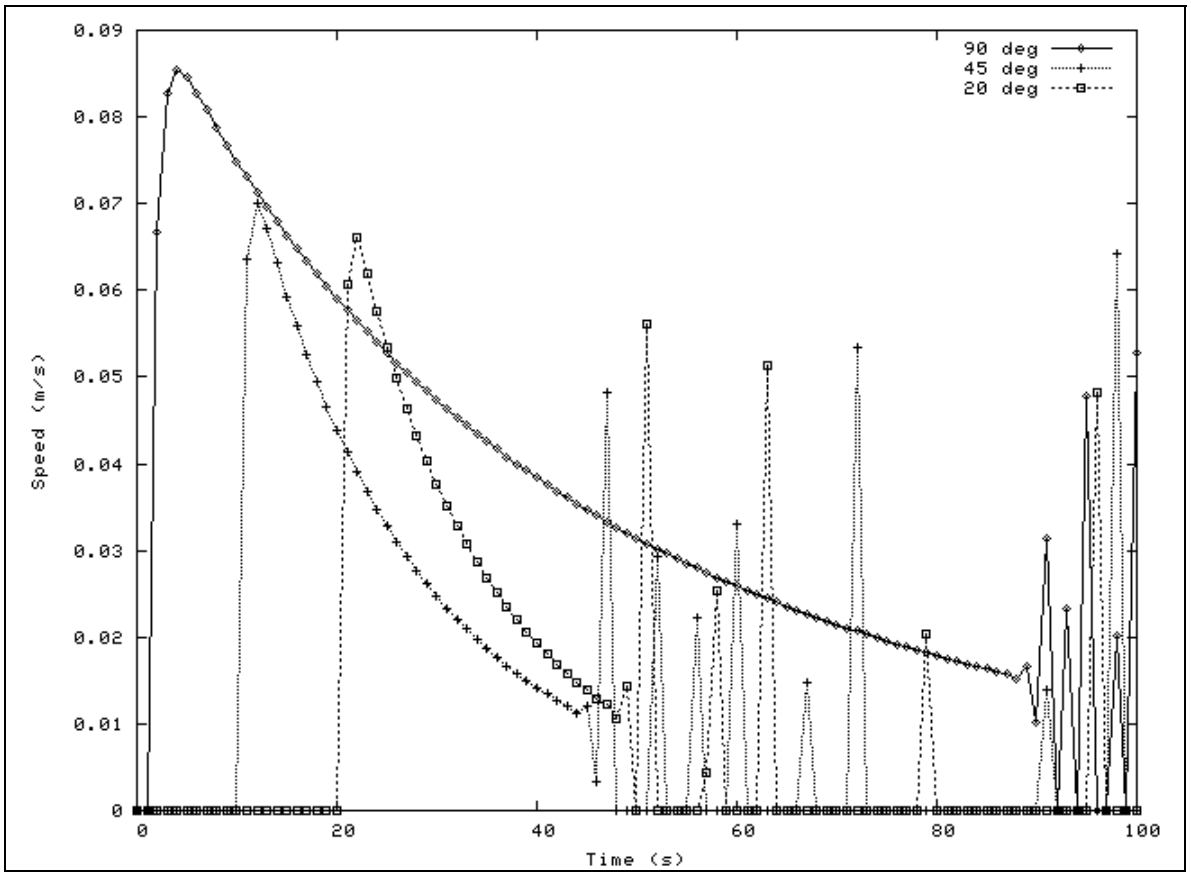


Figure 3.11: Speed versus time for various values of the cutoff angle. Note that the larger the cutoff angle, the faster the vehicle can move.

Notice that near the goal, there is erratic motion characterized by limit cycling due to the large change in heading deviations for small motions. The neighborhood of a target point should be made large enough so that it encloses any region of limit cycling, or the system should switch to a simpler open-loop following method near the target.

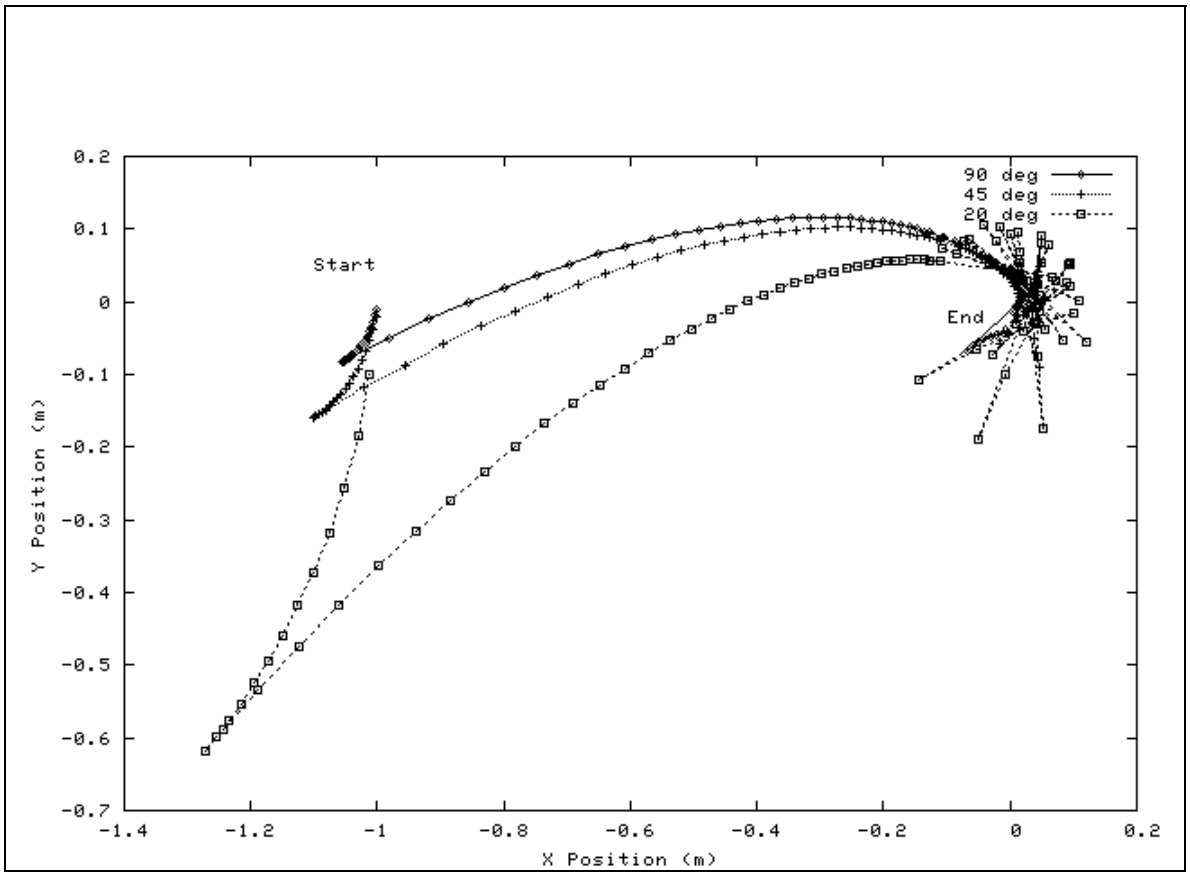


Figure 3.12: Trajectories for various values of minimum radius, at a small cutoff angle. Note the initial backing up of the vehicle, and the numerous "parallel parking" maneuvers near the goal, with backups (negative speeds) shown in Figure 3.13.

For vehicles with non-zero minimum turning radius, the nonholonomic constraint $v = \omega r$ may result in endless maneuvering about the target point. Figure 3.12 shows this behavior for various values of the minimum turning radius, at a constant (small) cutoff angle. Notice the "parallel parking" maneuvering near the goal, with backing up evidenced as negative speeds in Figure 3.13.

The exaggerated behavior depicted in the previous figures arises when the Prim Level is given a far-off target point. In a hierarchical system comprised of the Vehicle, AM and Prim Levels, the target points are planned to lie within a dynamically feasible envelope around the vehicle's current speed and heading, and the deviation angle should rarely lie outside the cutoff angle. The slowing and turning evident in the previous figures will arise only when the vehicle is significantly perturbed from its planned trajectory, for example due to sliding. In those cases the closed-loop behavior of the Prim Level will ensure that the vehicle will eventually return to its nominal path.

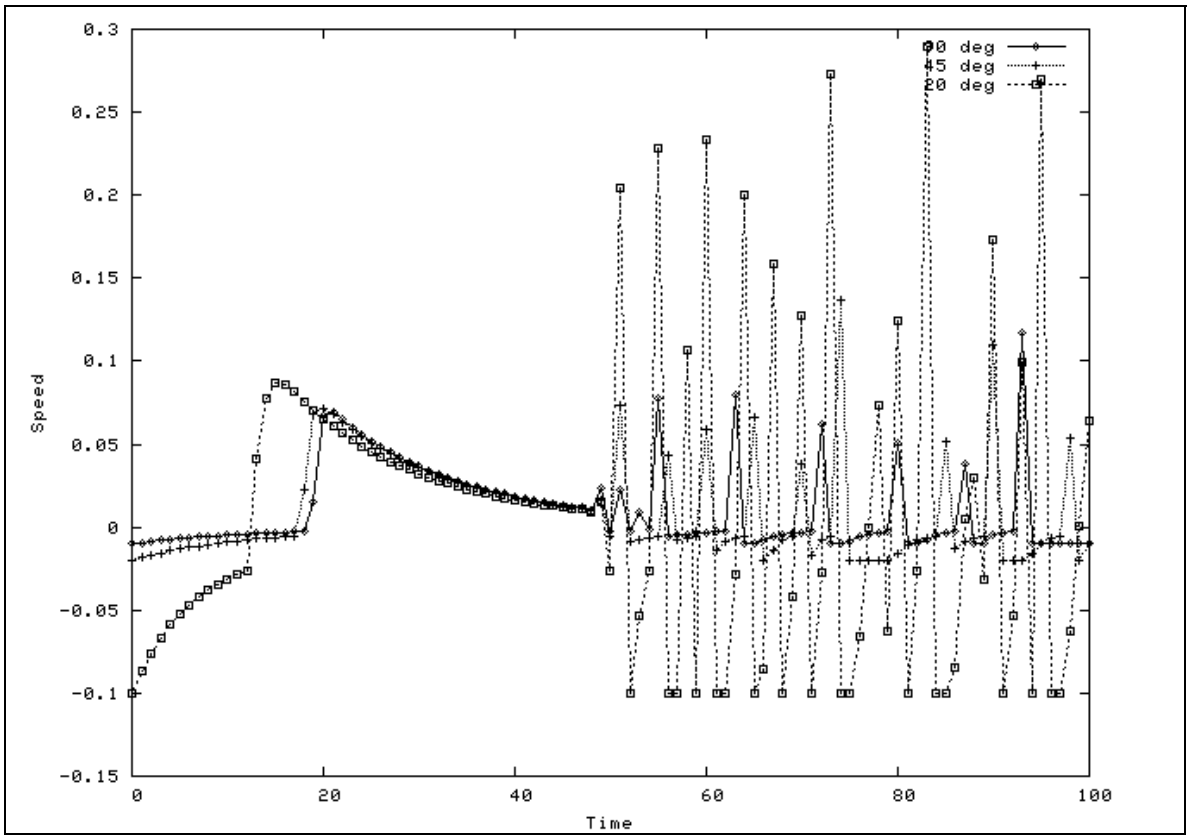


Figure 3.13: Velocities for various values of the minimum radius for a small cutoff angle. The backing up is evident as negative speeds.

3.5.2 The Servo Level

The Servo Level is the lowest level of the BG hierarchy, and interfaces directly with the sensors and actuators that make up the mobility platform. Sensors include those that measure discrete components of the vehicle itself, such as brake pressure, throttle position and wheel encoders, and those that measure navigation values such as GPS and INS for absolute and relative vehicle position. Actuators include those that affect the mobility of the vehicle, such as the steering, brake, throttle, transmission and transfer case settings.

The Servo Level is responsible for transforming periodic commands that the vehicle achieve some desired mobility state into actuator outputs that drive the vehicle toward the desired state. For mobility, the desired state is the velocity of the vehicle at some time in the short-term future. In planning, velocity states are transformed into actuator states using the inverse Jacobian function J^{-1} [8]. For example, with Ackerman (car-like) steering, the speed v and angular speed ω are transformed into a steering angle γ as $\gamma = \tan^{-1}(\omega L / v)$ where L is the wheelbase length. With dual independent drive wheels, the speeds v and ω are transformed into individual left and right wheel speeds ω_l and ω_r as:

$$\omega_l = \frac{1}{R}(v - \omega L/2)$$

$$\omega_r = \frac{1}{R}(v + \omega L/2)$$

In execution, the desired actuator states are compared with measured estimates and error signals are generated to drive the actuators to track their target values, using for example a proportional-integral-derivative (PID) control loop. In PID control, actuator driving output is computed from the difference error $e(t)$ between the commanded setpoint and the actual measurement of the actuator. The output $u(t)$is the sum of components due to the error, its time integral and its derivative, as:

$$u(t) = Pe(t) + I\int_o^t e(\tau)d\tau + D\frac{de(t)}{dt}$$

where P, I, and D are the gain coefficients that determine the amount of influence each term has on the output. The proportional P gain determines the speed of the response; the integral I gain reduces steady-state error that may accumulate; and the derivative D gain provides damping. The theory of tuning these gains for a particular system, and the discretization of the continuous PID equation for use in a digital computer implementation, are described in detail in [3].

Due to contraints on a vehicle's minimum turning radius, J^{-1} may compute infeasible steering angles. For example, a mobility command with zero translational velocity and some non-zero angular velocity can only be executed by a vehicle with a zero turning radius, i.e., one that can spin in place. Depending upon system design, the Servo Level may choose to execute the tightest turn possible, or stop with a failure status. In either case the superior Prim Level must be prepared to handle deviations between its requested velocity states and those achieved by the Servo Level.

Even if the Prim Level assures that the Servo Level can always compute feasible actuator states, the vehicle may drift from its nominal expected path due to slipping. Because the Prim Level is commanding velocities in order to achieve a position goal, these drifts can accumulate and take the vehicle far from the Prim Level's target goals. For this reason, the Prim Level must close its own control loop. This is unlike trajectory planning for robot manipulators, where open-loop paths can be executed with confidence that the joint servos can make up for any errors.

3.6 Low-Level Autonomous Mobility Implementation

3.6.1 The NIST Autonomous High Mobility Multipurpose Wheeled Vehicle (HMMWV) Implementation

NIST has carried out work in autonomous vehicle control for a number of years. As part of this effort, NIST has instrumented an Army Research Lab (ARL) HMMWV as a test-bed for research and development in sensors, sensory processing, world modeling, and behavior generation for autonomous driving. This section will describe the present implementation of the low-level vehicle control component. Figure 3.14 illustrates the present 4D/RCS control hierarchy structure of RCS control nodes that are involved in the lower level mobility control of the vehicle. These lower level echelons have the responsibility for generating the vehicle goal trajectories and for accurately controlling the vehicle to follow these goal trajectories. A number of considerations such as maximum allowed velocities, accelerations, and jerks both tangential and lateral to the trajectory paths, as well as response characteristics of the different actuator systems will affect these calculations. This section will explore how the responsibilities for various levels of these control functions are distributed through the different RCS echelons; how real-time measurements are fed back into the calculations; and how different auto-tune capabilities are used.

Only those lower level nodes that directly control the trajectory of the vehicle based on an internal model representation of the goal paths to be followed will be examined. The 4D/RCS reference architecture contains a set of RCS echelons to carry out these lower level tasks. These are the PRIM and SERVO echelons. Due to the complexity of this implementation for autonomous on-road driving, the PRIM echelon has been expanded to two levels (Elemental Movement and Primitive/Trajectory) and the SERVO echelon has been expanded to three levels for those control threads that involve servo motor actuators (see Figure 3.14).

The overall on-road driving control is partitioned by the RCS methodology into a hierarchically executed task decomposition structure. For autonomous on-road driving, the echelons above Elemental Movement have decided on the route (turn-by-turn directions), have recognized entities relevant to the driving task, have dealt with driving behaviors such as intersection right-of-way, passing, etc. and have identified the lane the vehicle is to be driving in. This lane specification is an input to Elemental Movement along with an Objects-of-Interest Table, which lists nearby objects that have been determined to affect the local vehicle motion. This table carries a number of parameters developed by the higher echelons that provide offset distances, speeds, and costs that Elemental Movement uses to adjust the input goal path of the vehicle. The last set of the operations within Elemental Movement is to process the list of lane segments that identify the path the vehicle is to follow in order to specify each segment's speed and acceleration parameters. Elemental Movement then passes these lane segments and parameters, one lane segment at a time, to the Primitive/Trajectory echelon. This description will be started with this lane segment processing by Elemental Movement.

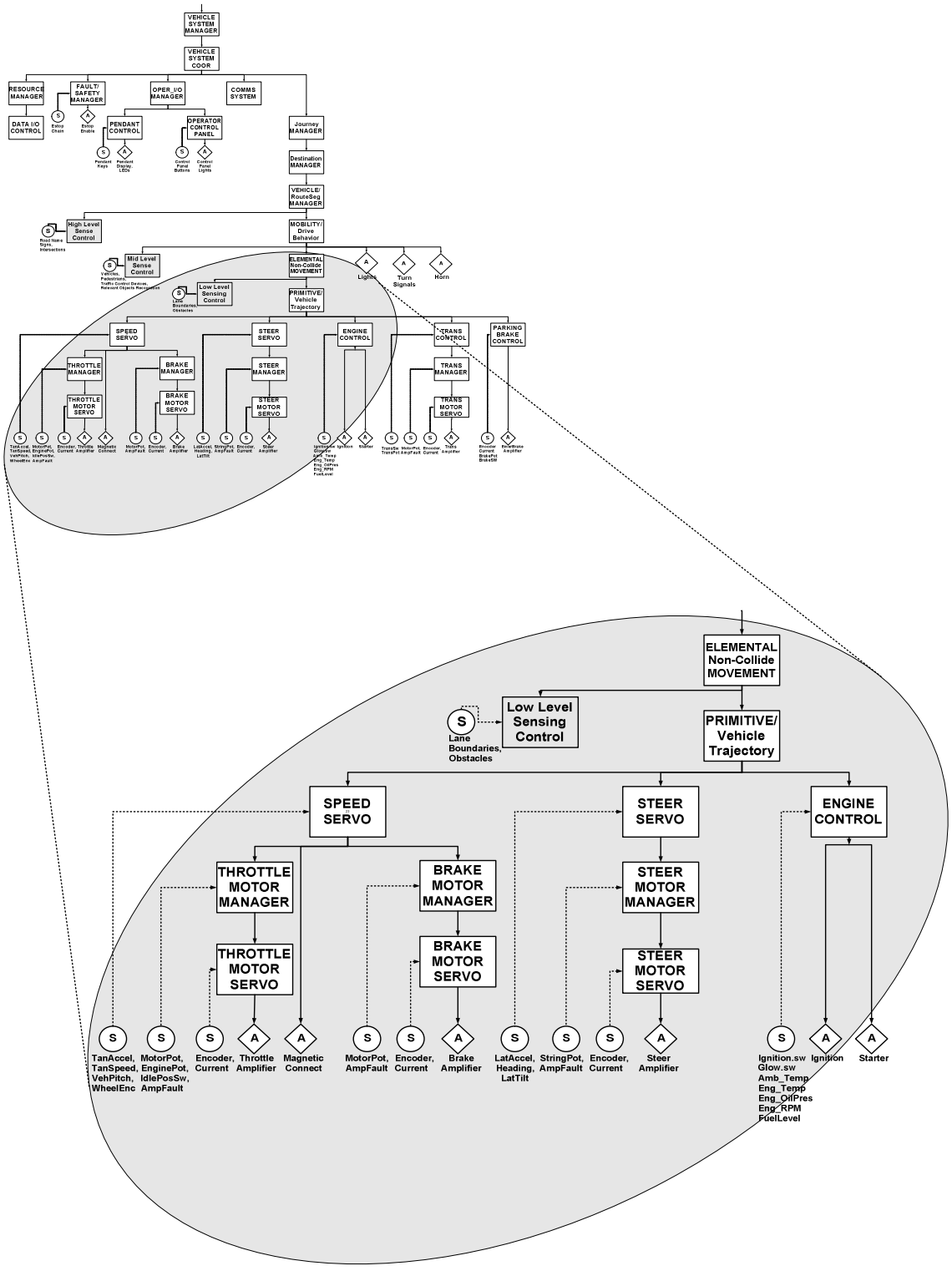


Figure 3.14: The complete HMMWV 4D/RCS controller is shown in the upper left with the lower echelon nodes enlarged. The reference PRIM RCS echelon is expanded to the Elemental Movement and Primitive/Trajectory echelons. The reference SERVO RCS echelon is expanded to the three nodes of servo control shown. The circles with an "S" represent sensor input, the diamonds with an "A" represent actuators being controlled, and the rectangular boxes represent RCS nodes.

3.6.2 Elemental Movement RCS Echelon

A representational data structure called a lane segment is used to identify the desired vehicle path. Figure 3.15 illustrates the evolution of this concept of lane segments, which are connected constant curvature arcs that specify the path for the center of the vehicle to follow. Figure 3.15a shows a vehicle on a two lane undivided road that curves up to a stop sign controlled intersection. Figure 3.15b shows the same road with all of the lane segments identified. These lane segments represent the centerline vehicle travel path as defined by the road and possible turning maneuvers at the intersection. This should be the set of all possible paths for normal driving behavior. Figure 3.15c has reduced this set to only the lane segments that the example vehicle will follow in driving up to the stopping point at the intersection. The vehicle is presently on LaneSegment_10, which is a straight-line path. This lane segment data structure is specified by its start point and end point coordinates in a relative Universal Transverse Mercator (UTM) coordinate system of Northing and Easting values. If the lane segment is an arc, then the UTM coordinates of the center point of the arc are included as well as a specification as whether the direction of travel along the arc is clockwise or counterclockwise.

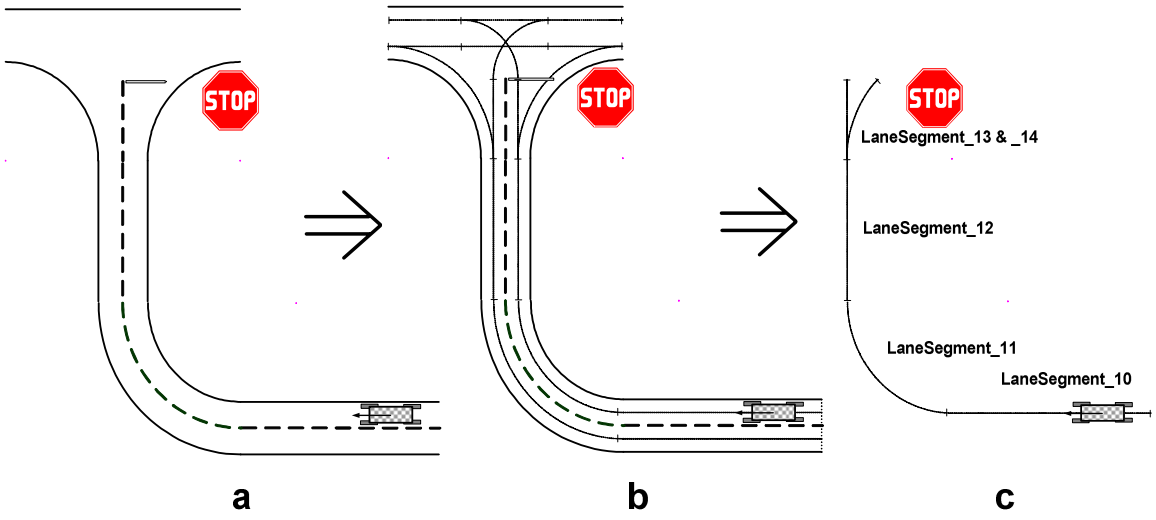


Figure 3.15: The normal road layout with a vehicle approaching a stop sign is shown in 3.15a. The set of possible lane segments for this road is shown in 3.15b. Only the lane segments that the vehicle can follow to the stop are shown in 3.15c. LaneSegment_13 is the straight section and LaneSegment_14 is the right turn section.

Elemental Movement, after its correction of the position of these lane segments to accommodate the Objects-of-Interest list, processes these lane segments to calculate the appropriate trajectory control parameters of speed and acceleration for each. These calculations treat the lane segments as if they lay on a 2D plane. The initial desired speed of travel and allowed vehicle accelerations have been set somewhere else in the control system and are read into Elemental Movement from the world model. In this example, the maximum travel speed is set at 15.0 m/s, the maximum allowed lateral acceleration is set at 3 m/s^2, the normal vehicle along-path acceleration is set at 0.7 m/s^2 and the acceleration value for the vehicle to use to get back on path is 0.5 m/s^2. These parameters are used to process the lane segments as follows: first, any lane segments that have stopping requirements (stop sign) are determined, and the maximum speeds are calculated for all curved lane segments based on the maximum allowed lateral acceleration. Figure 3.16 shows an example of this first step. This step includes determining the exit speed, i.e. the speed the vehicle

should be traveling at the end of each lane segment. This is usually set by some speed requirement of the next segment. Here it may be seen that the curved arc LaneSegment_11 (with a radius of 12.6 m), when evaluated using a maximum lateral acceleration of 3.0 m/s^2 results in a maximum arc speed of 6.2 m/s, down from its initial 15.0 m/s value. Since the vehicle should not be going any faster than 6.2 m/s on this segment, the exit speed of the previous segment (LaneSegment_10) is specified to be 6.2 m/s. This will tell the trajectory generator that the vehicle has to slow to 6.2 m/s by the point where it is exiting LaneSegment_10 so it is prepared for LaneSegment_11.

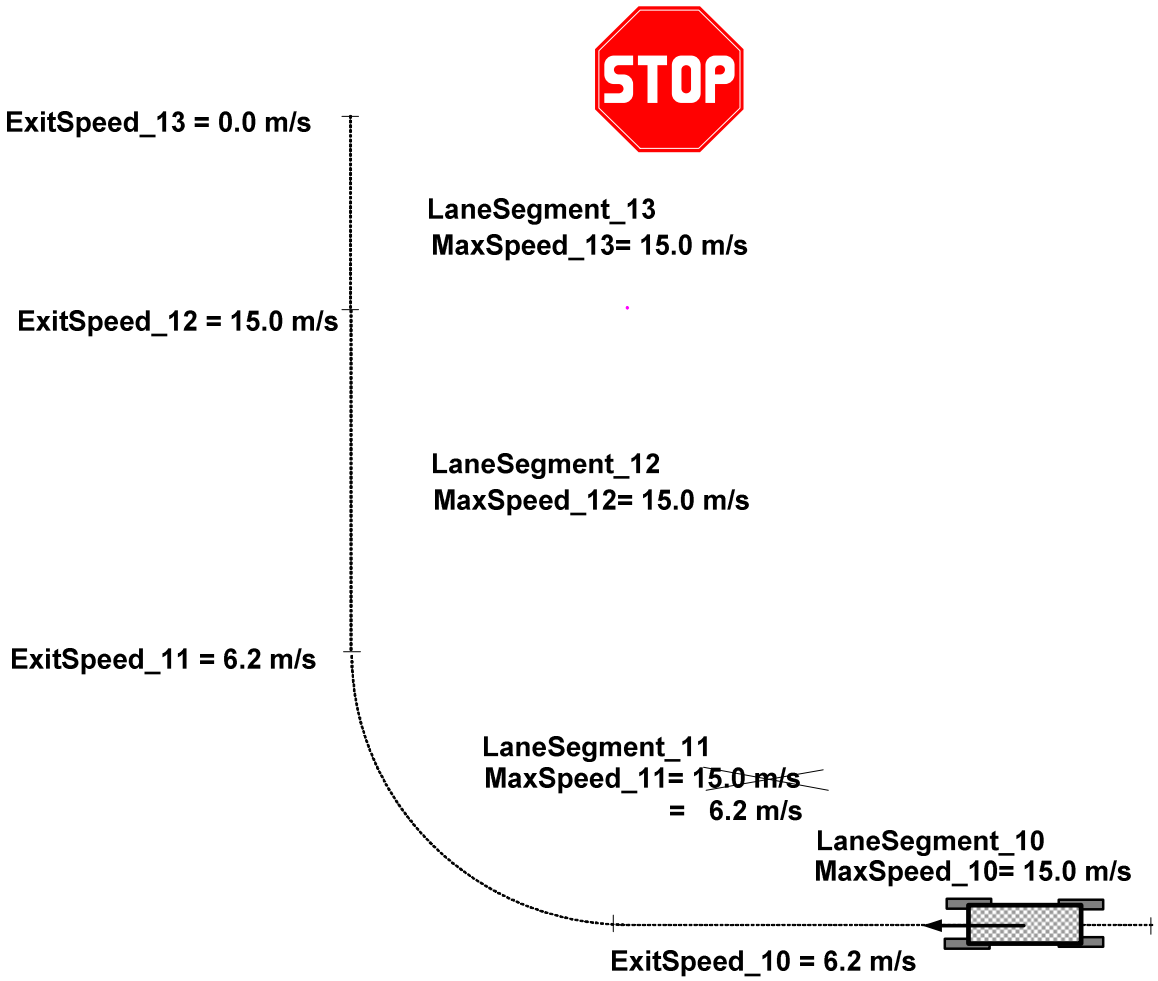


Figure 3.16: After goal lane segments are generated, Elemental Movement first parses them to determine any stopping conditions, i.e. ExitSpeed goes to 0.0 m/s, and calculates speeds on curves from maximum allowed lateral acceleration value, e.g. the arc LaneSegment_11 maximum speed is reduced from 15.0 to 6.2 m/s.

After this first pass through the candidate lane segments, the next pass is done from the farthest to the closest lane segment in order to back out any constraints that will cause a reduction in speed up ahead and determine what the impact is to closer lane segment parameters. This will, for example, keep the vehicle from entering a curve too fast so that the lateral acceleration value that is necessary to maintain traction is exceeded and, as a result, slippage out of the current lane occurs. In this present example, there are two constraints that back up parameter changes into closer lane segments. The first is the stop at the end of LaneSegment_13. Elemental Movement starts with this requirement for an exit speed of 0.0 m/s at the end of LaneSegment_13, determines the path length of the segment (9.5 m), and determines what speed the vehicle must be going to reach an exit speed of 0.0 m/s over the path distance with the specified

deceleration value (0.7 m/s^2). If this initial enter speed (here calculated to be 3.6 m/s) is less than the default maximum speed (15.0 m/s) then it becomes both the enter speed and the maximum speed specified for this lane segment. The enter speed (3.6 m/s) for LaneSegment_13 obviously becomes the exit speed (3.6 m/s) for LaneSegment_12. Now the process is repeated for the LaneSegment_12 and this determines that the vehicle cannot be going any faster than 6.0 m/s to be able to slow to 3.6 m/s over the path length distance of 16.7 m making 6.0 m/s the enter speed for LaneSegment_12 and the exit speed for LaneSegment_11.

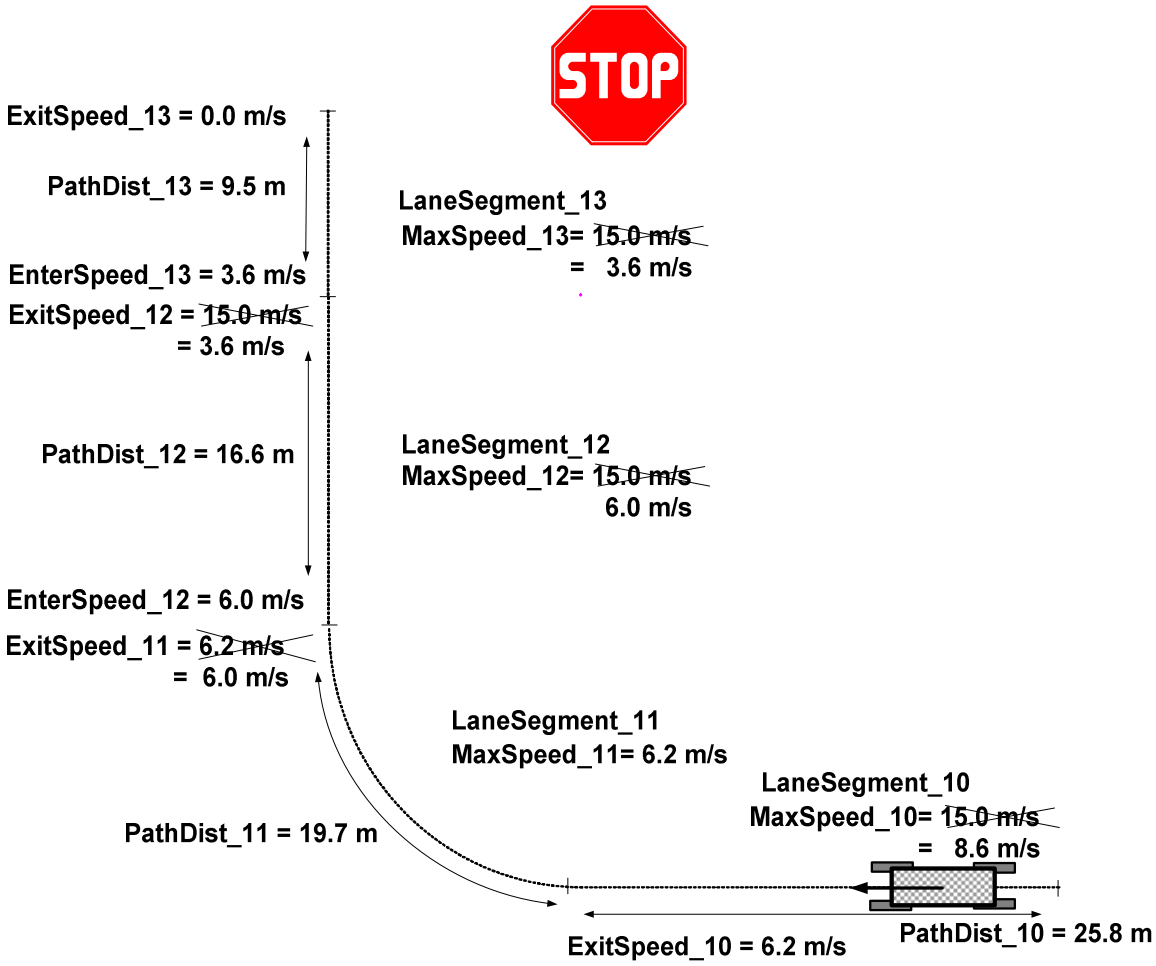


Figure 3.17: The lane segments are processed from the farthest to the closest to back out constraints and their effect on nearby lane segments speed. For example, the requirement to stop at the end of LaneSegment_13 along with a deceleration value of 0.7 m/s^2 causes the maximum speed of LaneSegment_12 to drop from 15.0 to 6.0 m/s.

In a similar manner, the curved LaneSegment_11 forces a decrease in speed to 6.2 m/s which reflects back to the exit speed for LaneSegment_10. With a path distance of only 25.8 m for LaneSegment_10, it is determined that the vehicle cannot be going any faster than 8.6 m/s when it enters this segment to be able to slow in time for the curve (see Figure 3.17). If this were an actual driving situation and the vehicle was already on LaneSegment_10 with a speed higher than the 8.6 m/s, Elemental Movement would increase the

specified deceleration value for this lane segment to accommodate the path distance and exit speed requirement. Indeed this is the same behavior people display in a similar situation where they will suddenly brake harder if they are coming up to a curve faster than appropriate.

The initial preparation of the lane segments by Elemental Movement results in the specified speeds and accelerations to properly traverse the lane segments. Elemental Movement has carried out all of these calculations in one control cycle, which on the NIST HMMWV is 2 ms long. Within the same control cycle it passes the output of the first lane segment along with that lane segment's derived control parameters (Figure 3.18) to the Primitive/Trajectory echelon to begin calculating the component vector moves. Each succeeding cycle, Primitive/Trajectory will calculate the next incremental vector to move the vehicle while trying to follow the commanded lane segment. When Primitive/Trajectory nears the end of the lane segment, it issues an advanced status to Elemental Movement so the next lane segment can be sent coincident with the present executing lane segment just finishing. During the time that Primitive/Trajectory is calculating these vectors, Elemental Movement is continuously reassessing the lane segments based on sensor input to see if the position estimate of lane segment or the status of an object-of-interest has resulted in a change in any of the lane segments. If this is the case, Elemental Movement recalculates the list and immediately sends the newly corrected lane segment to Primitive/Trajectory. This can occur within any control cycle, i.e. any change will be reflected in a new trajectory being started within 2 ms.

3.6.3 Primitive/Trajectory RCS Echelon

Primitive/Trajectory has the responsibility of calculating the vehicle trajectory. This is a fairly involved calculation that breaks the trajectory into a tangential along-path component and a normal back-to-path component, each controlled by its own parameters of maximum speed and acceleration, which have been sent by Elemental Movement. The tangential component will be discussed first. The trajectory algorithm tries to reach the maximum lane segment speed along a tangent-to-path vector using the value of the specified maximum tangential acceleration. Each control cycle, it also checks to see what the remaining segment distance is and determines when, if necessary, to start decelerating so that it can reach the specified exit speed at the segment end point. When it reaches some tolerance value in terms of both distance to the end point and number of control cycles to the end point, it notifies Elemental Movement of this fact so that a new lane segment will be commanded coincident with the end of the present one.

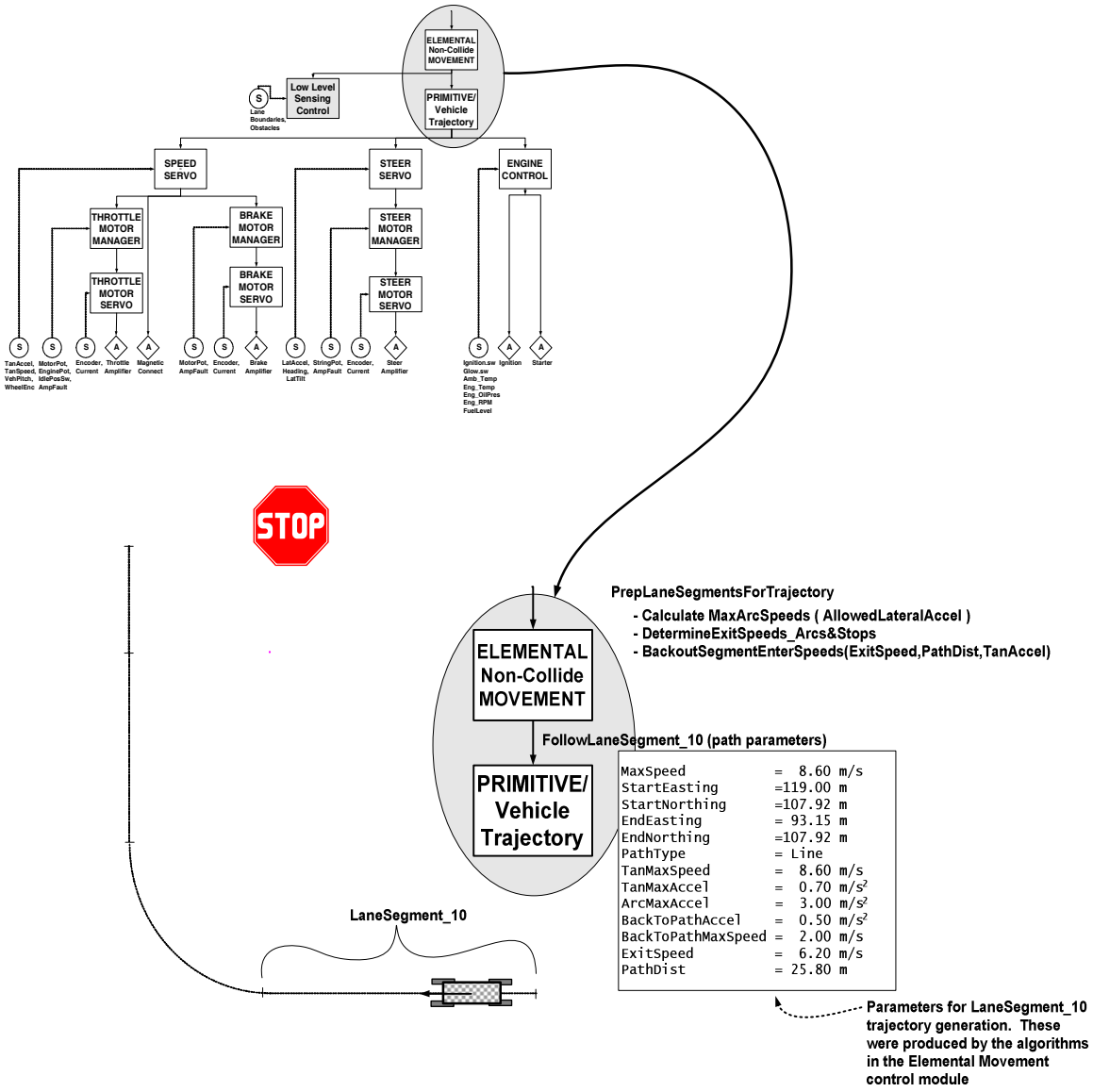


Figure 3.18: Elemental Movement commands Primitive/Trajectory to execute a single lane segment, LaneSegment_10. It also sends all of the path parameters necessary to control the speeds and accelerations during the trajectory generation. These parameters were calculated by Elemental Movement at the end of its processing cycle.

Simultaneously with these tangential calculations, a back-to-path component is also being calculated. If the trajectory's value for the present position of the vehicle is not on the commanded lane segment path, a second trajectory process calculates an independent trajectory motion in a direction normal to the path. This is known as a back-to-path trajectory. Elemental Movement has commanded a separate set of parameters of maximum back-to-path speed and acceleration to control this motion. This trajectory calculation is attempting to move back to the commanded path, ending with a normal speed of 0.0 m/s at the point when the path is reached. Each control cycle, this back-to-path component is calculated and is vector summed with the tangential component to produce a resultant increment motion vector for the vehicle. Figure 3.19 gives a pictorial representation of this trajectory calculation process.

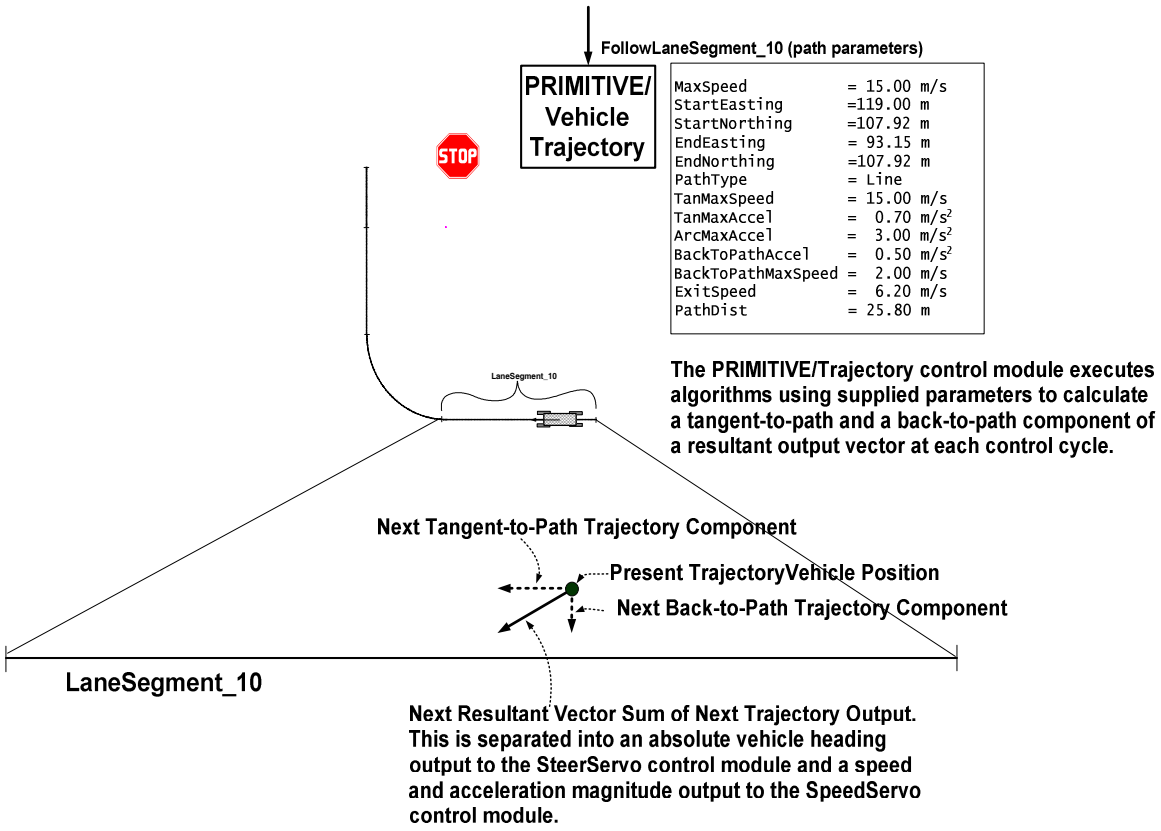


Figure 3.19: Primitive/Trajectory calculates two orthogonal trajectories, one tangential to the path and the other normal to the path (back-to-path). These two components are then vector summed to produce the next motion vector for the vehicle.

Primitive/Trajectory calculates a next incremental motion vector each control cycle. At the end of Primitive/Trajectory's processing each cycle, it decomposes this vector into an absolute heading direction, which it commands to the SteerServo node and a magnitude component of speed and acceleration, which it commands to the SpeedServo node.

3.6.4 Real-time Trajectory Feedback Control

If these lower level nodes (SpeedServo and SteerServo) executed perfectly, then the vehicle would exactly follow the trajectory path. Since this is not the case, a feedback loop is set up that provides the present position, speed, acceleration, and heading of the vehicle to use as an error correction to the trajectory generator. These inputs are read into Primitive/Trajectory at a rate of 20 times each second. If the vector output command of Primitive/Trajectory was realized by the vehicle instantaneously each cycle, then this real-time feedback could just be directly input into the trajectory algorithms as the present state of the vehicle for the next calculation. In reality, it is about 0.3 s before the vehicle reaches the commanded state of a vector that has been output from Primitive/Trajectory. This means that the trajectory calculations are about 0.3 s ahead in time of the real state of the vehicle. At 15.0 m/s, the trajectory is calculating positions, speeds, accelerations, and headings about 5 m in front of the vehicle, therefore the present vehicle's values cannot be directly input as error corrections to the trajectory calculations.

In order to properly correct the trajectory calculations, the present vehicle position, speed, and acceleration must be used to calculate how far the vehicle will travel in 0.3 s. The present heading and wheel steer angle are then used to define the present arc the vehicle is moving on and lay out the projected distance along this arc to estimate the projected values to feed into the trajectory calculations to provide this feedback control.

Figure 3.20 illustrates these calculations. This feedback of present vehicle values projected forward in time to match the time space of the trajectory calculations occurs every 50 ms. For the control cycles between these updates, the trajectory generator simply assumes the next present starting state is the last vector calculated by the trajectory algorithms, i.e. it runs open loop.

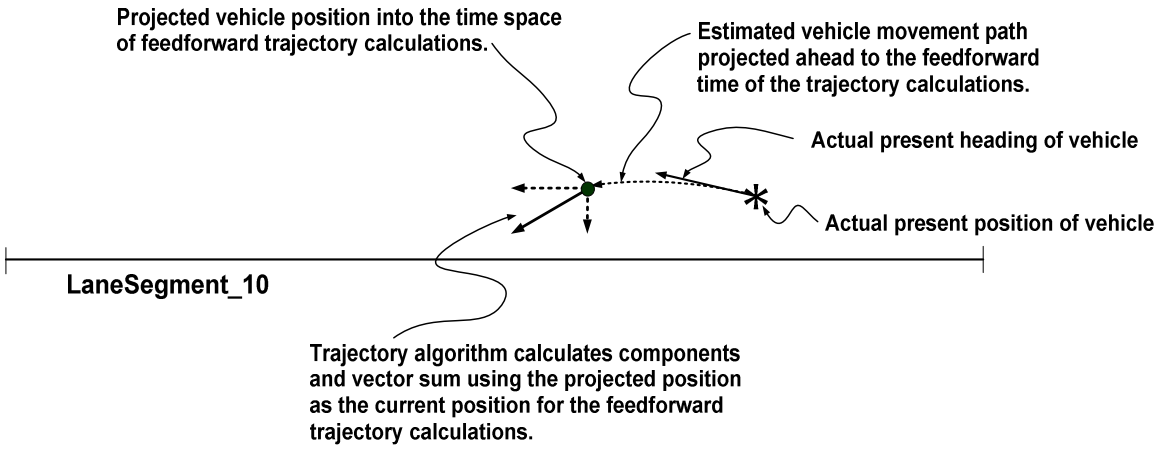


Figure 3.20: To provide real-time feedback corrections to the trajectory generator, every 50 ms the present state values of the vehicle are projected forward in time to calculate an estimated present vehicle state in the time space of the trajectory calculations.

SteerServo RCS Node

SteerServo sits at the top of a chain of three RCS echelons (Figure 3.14) that are involved in converting the commanded vehicle heading into the control voltage outputs to the steering motor amplifier. The commands to SteerServo are absolute vehicle heading values that are the pointing angle of the resultant vector from the trajectory calculations each cycle. SteerServo processes this angle, which is known as the Cmd_VehHeading_Abs through a number of steps to produce a value for the steer motor actuator that is connected to the steering wheel in the vehicle. Figure 3.21 illustrates a number of the relevant geometries used in this operation. Each control cycle, the value for the derived steer motor position is passed down to Steer(Motor)Manager along with acceleration and maximum speed values for the motor motion control. These will set the rate of change of the steer angle and overlay a clothoid-type motion on top of the entry and exits of the constant curvature arcs. For example, when entering an arc from a straight line, the heading starts changing at the constant rate to follow the arc. The constant change of heading will result in an approximate fixed steering angle. The movement of the steering wheel to this setting is done at a linear rate controlled by a trajectory of the steer motor output values calculated by the SteerManager based on the motor maximum speed and acceleration values sent by SpeedServo. This constant rate of change of the steering wheel at the entry and exit of arcs imposes a clothoid-like blend at the cost of some positional error but smoother vehicle motion since the faster the vehicle steering tries to change the larger the jerk felt as the lateral acceleration would be increasing from zero to the full arc value in a very short time increment.

SteerManager calculates the motor trajectory and, each cycle, sends the next motor position, velocity, and acceleration to SteerMotorServo. SteerMotorServo executes a velocity feedforward calculation based on the commanded motor velocity and calculates a position error between the commanded position and a feedback position from encoder feedback to use in an error correction calculation. These values are summed to generate the value to send to the steering wheel motor amplifier. This amplifier develops a velocity loop solution based on a tachometer feedback signal and controls the voltage to a brush dc servomotor.

This represents the basic data flow from the commanded vehicle heading down through three steering related RCS echelons to the voltage outputs to the steering wheel motor amplifier. A closer look reveals that SteerServo is trying to calculate a steering wheel position that will cause the vehicle heading (measured at center of the vehicle and lying parallel to the long axis of the vehicle) to point in the direction of the commanded heading. From looking at Figure 3.21 it may be seen that subtracting the present vehicle heading (VehHeading_Abs) from the commanded heading (Cmd_VehHeading_Abs), produces an angle (Cmd_VehCenterAngle_VehRel) that represents the required angular change of direction as measured at the center of the vehicle. Since this vehicle turns by Ackerman steering of the front wheels, a geometric relationship can be calculated, as seen in the illustration in Figure 3.21, that converts this Cmd_VehCenterAngle_VehRel into a SteerWheelsAngle_VehRel. This relationship is non-linear and is accurately approximated by a table lookup. The values for this table have been experimentally obtained by setting the steering wheel motor to the set of whole integer revolution values and for each setting, experimentally measuring the resulting traveled arcs of the vehicle and converting these into the corresponding angles known as the SteerWheelsAngle_VehRel. This table is then accessed with the calculated value of the SteerWheelsAngle_VehRel and the corresponding steering wheel motor value retrieved. A new motor position value along with the maximum motor speed and acceleration are sent to the SteerManager each control cycle.

Again, the relative timing of these calculations is important. As previously noted, the commanded vehicle heading sent down from Primitive/Trajectory is the value for the vehicle's heading 0.3 s into the future whereas VehHeading_Abs is the vehicle's heading at the present instant. Similarly, as was done in Primitive/Trajectory, the vehicle's present speed, acceleration, heading, and steer wheels angle are used to project the vehicle's position 0.3 s into the future and determine an estimated heading. From this, a modified Cmd_VehCenterAngle_VehRel is calculated that should put the vehicle at the correct heading 0.3 s from now.

Once a candidate Cmd_VehCenterAngle_VehRel is determined, several limit checks are performed on it. First, it is verified that this angle will not exceed the maximum turn angle of the steering wheel lock limit. Next, using the relationship that the arc radius is equal to the speed along the arc divided by the lateral acceleration, the vehicle's present speed and the resulting radius from this Cmd_VehCenterAngle_VehRel are used to determine if it will cause the vehicle to exceed the maximum allowed lateral acceleration value specified in the world model. Then, the Cmd_VehCenterAngle_VehRel is transformed to a corresponding SteerWheelsAngle_VehRel. Next a table look-up is performed to transform to the steer motor position value and, finally, SteerServo sends this command to the SteerManager to build the steer motor trajectory.

All of the above described steer calculations occur in the same control cycle (the same 2 ms) that Elemental Movement and Primitive/Trajectory carried out their described operations. Each control cycle, these calculations repeat from the top to the bottom RCS echelon.

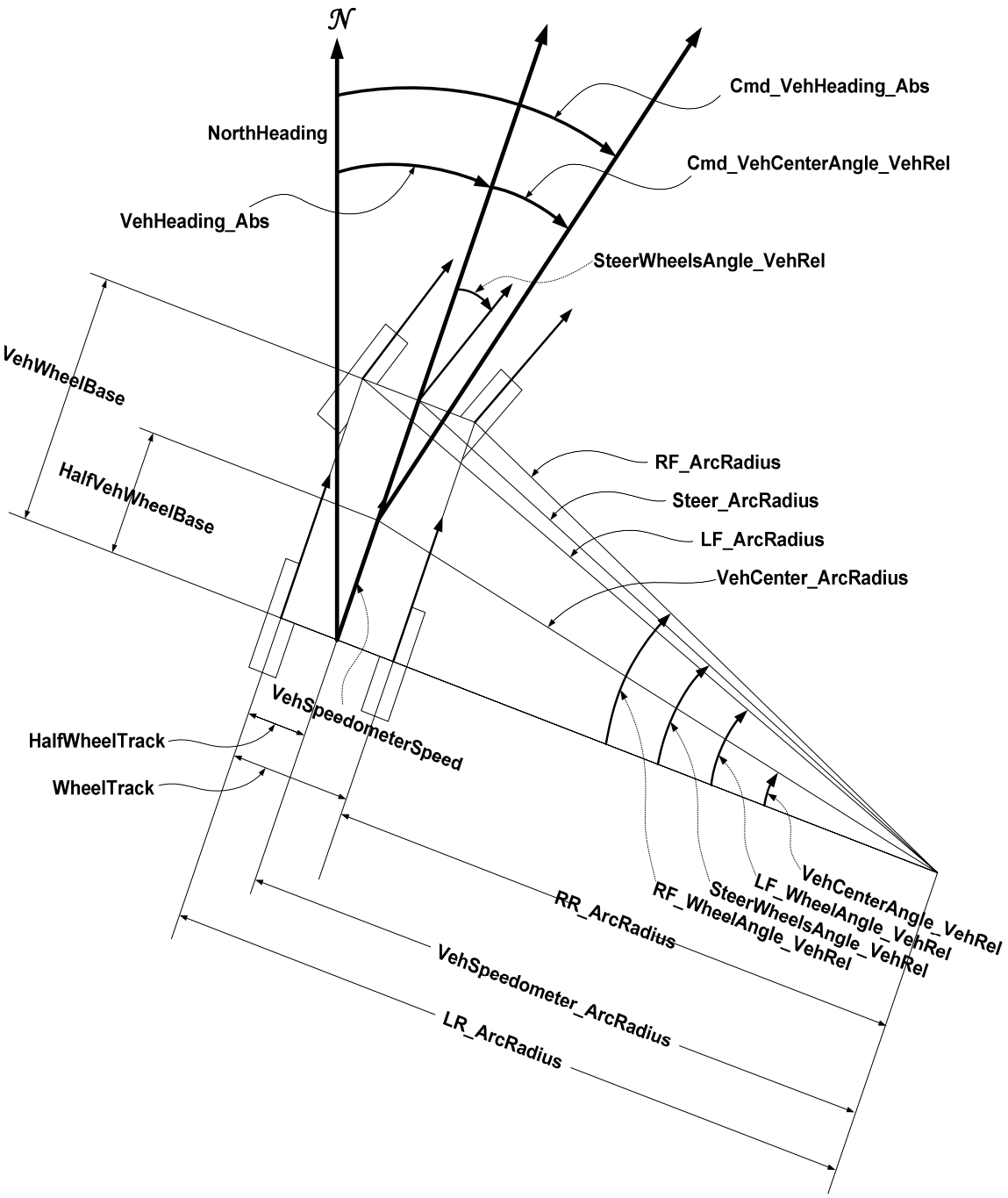


Figure 3.21: Illustration of relevant angles for steering control of the vehicle. SteerServo is commanded with the Cmd_VehHeading_Abs as the goal angle from the present control cycle calculations.

SpeedServo RCS Node

In parallel with this activity, and in the same control cycle, the SpeedServo subtree of RCS echelons (Figure 3.14) is processing the commanded speed and acceleration values. The SpeedServo RCS node determines the necessary engine throttle (motive force) and the brake line pressure (braking force) to command the ThrottleManager and BrakeManager respectively. ThrottleManager does a feedforward

calculation to estimate the throttle motor position and BrakeManager does a feedforward calculation to estimate the brake motor position in order to attain these commanded values. Each of these nodes then calculates the respective motor trajectory and sends the motor position, velocity, and acceleration commands to the ThrottleMotorServo or the BrakeMotorServo node. Just like the SteerMotorServo, the ThrottleMotorServo and the BrakeMotorServo execute velocity feedforward loops with a position error correction function.

In order to compute a desired speed, SpeedServo receives commands of instantaneous speed and acceleration as well as the maximum speed and the exit speed of the present lane segment each control cycle. The response of the vehicle to any changes to throttle or brake takes a minimum of 0.3 to 0.4 s to begin and depending on a number of additional parameters the desired value may not even be realizable. For example, going up an incline of more than one to two degrees, the vehicle can only attain an acceleration of 2.0 m/s^2 up to a speed of 1.7 m/s after which the acceleration quickly drops to a value less than 0.5 m/s^2, even at full throttle. In general, the acceleration that the vehicle can attain drops as the speed increases. For the vehicle deceleration situation, allowing the throttle to return to the idle position without any brake application can cause a deceleration of 0.4 m/s^2 or greater at most speeds and on level ground.

The basic approach, as has been seen in the discussion of all of the RCS echelons here, is to develop a model of the system at the resolution of a particular echelon and to use it to estimate the correct output to the next subordinate echelon in order to carry out the desired action, namely, a feedforward servo is done with real-time feedback of the relevant measurement parameters being used to do a small error correction to achieve the desired goal.

This is an essential component of the RCS approach. Each echelon is performing a servo calculation at different levels of resolution. There are two basic ways to calculate a servo function that will adjust the control signal to accomplish the desired goal:

1) the system can wait until it observes an error between its goal action and what it sees/measures in the real world and use these measurements to calculate a change in the control signal to correct this error; or
2) the system can have a model of how the controlled system behaves in the world and estimate from this model what the correct control signal should be for the desired action.

The first method requires an error to calculate a control signal while the second attempts to apply a model-based correct control signal so no error occurs. The fundamental RCS approach as discussed in Chapter 1 is to develop behavior generation that is based on an internal world model that is updated by sensor measurements of the environment. Therefore, the designed system leans heavily in the direction of model-based or feedforward servo control at all levels.

Consistent with this feedforward approach, SpeedServo works in cooperation with Primitive/Trajectory. Primitive/Trajectory attempts (because of the model-based feedforward approach) to only send a command to SpeedServo (or to SteerServo) that it can accomplish within a reasonably short time period, usually within one to a few control cycles. In a similar manner, SpeedServo attempts to send commands to ThrottleManager and BrakeManager that are also feasible within a short time span. This requires each node to base its output calculations on a fairly accurate model with real-time corrections to this model being made as the environmental situations warrant. A simplified model is currently being used for the SpeedServo calculations. Detailed, physics-based dynamic models have been built off-line and are being studied to see how they may be considered for enhancement of this node. Most real-time implementations of feedforward models have some aspect of table lookup based on key parameters in order to meet the very fast update requirements of these systems.

A combination of table lookup together with a reasoning function based on evaluations of present parameters using a knowledge set of relevant rules is used in SpeedServo. This problem has been partitioned in the following manner. The state of the system is evaluated by a number of rules applied to the present values of a number of parameters (both command and sensed) to identify the present situation as one of several particular states, each of which has a corresponding particular feedforward algorithm that provides the best solution estimate to the correct output action. The system basically determines if it should be accelerating, maintaining a constant speed, decelerating to a lower speed, decelerating to a stop, or

maintaining a stopped position. These states have been further delineated to identify if there is a transition into a new situation or this is a continuation of the previously selected situation. This is important because it affects how the error correction is applied. The error correction is a function of how long the system has been in this situation along with the size and rate of change of the error. Figure 3.22 shows the overall process flow for the SpeedServo RCS node and illustrates the basic operations of the sensory processing and world modeling components in setting up the parameters and carrying out these evaluations. Each of the RCS nodes described here has an identical process flow.

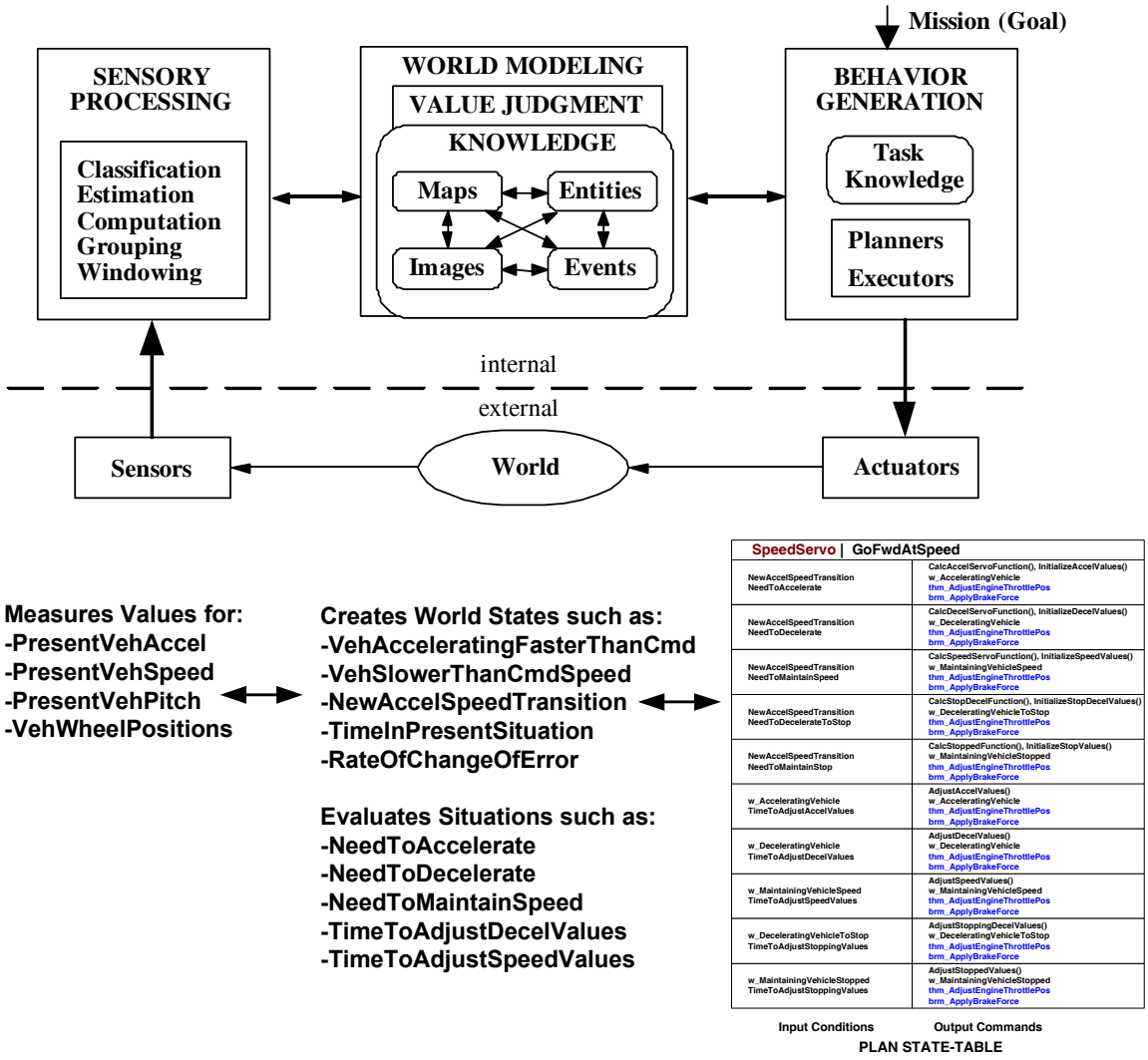


Figure 3.22: Illustration of process flow in the SpeedServo RCS node. Sensory Processing brings in relevant acceleration, speed, pitch, and wheel position values to the world model. The world model processes these along with state and command input parameters to evaluate the present situation and behavior generation selects the most appropriate solution algorithms using rules in a state table format that transition on the evaluated situations.

State Table Process

Figure 3.23 details a simplified version of the state table used at the SpeedServoNode to select the appropriate servo function. The values listed in the left hand column of the table represent the conditions or situations that, if true, result in the execution of the corresponding entry in the right hand column. For instance, in line 1 the left hand entry requires both the condition called NewAccelSpeedTransition and NeedToAccelerate to be true for the line to match. The condition NeedToAccelerate results from the world model evaluation of the input command values from Primitive/Trajectory and the present state of the vehicle acceleration and speed. If the input command is calling for acceleration and the present state of the vehicle has not yet reached this value, then the world model processing would set the condition NeedToAccelerate. If in the previous cycle, this condition state value was evaluated to NeedToMaintainSpeed and then in this cycle is evaluated to NeedToAccelerate, this would cause the additional condition state of NewAccelSpeedTransition to also be set indicating that this is the first cycle that has transitioned to a different accel or speed state, i.e. this is the first cycle that the condition NeedToAccelerate is true. It is important to detect this first transition to this state so that certain parameters relative to the integral error correction terms can be initialized. When both of these conditions are evaluated true, the first line of this state table matches and the entries in the right hand side are executed.

The right hand table entries of line 1 in this example are:

- CalcAccelServoFunction() – an algorithm which uses the commanded acceleration, present vehicle speed, and vehicle pitch (indicating road incline) together with a lookup table to estimate the engine throttle position and the brake line pressure to reach the commanded acceleration.
- InitializeAccelValues() – an algorithm which initializes a number of parameters that begin tracking the length of time the acceleration algorithm has been executing and that measure the relative success in reaching the commanded acceleration. These will be input parameters to the integrating error correction algorithm AdjustAccelValues that will be executed in another line when a different condition set matches.
- w_AcceleratingVehicle - this sets the value of a world state variable that tracks the active servo algorithm being calculated by SpeedServo. In this case, this world state value is set to w_AcceleratingVehicle to indicate that SpeedServo is calculating the algorithms to accelerate the vehicle.
- thm_AdjustEngineThrottlePos – this sets the output command to the ThrottleManager to AdjustEngineThrottlePos. The parameters that accompany this command have been calculated in the CalcAccelServoFunction(). The prefix thm_ is an abbreviation that indicates that the command is to be sent to the ThrottleManager. This command will be sent to the ThrottleManager node at the end of the SpeedServo's processing this cycle. ThrottleManager will execute this command later in the same 2 millisecond control cycle.
- brm_ApplyBrakeForce – in a similar manner as above, this sets the output command to the BrakeManager to ApplyBrakeForce. The prefix brm_ is an abbreviation that indicates that the command is to be sent to the BrakeManager.

SpeedServo | GoFwdAtSpeed	
NewAccelSpeedTransition NeedToAccelerate	CalcAccelServoFunction(), InitializeAccelValues() w_AcceleratingVehicle thm_AdjustEngineThrottlePos brm_ApplyBrakeForce
NewAccelSpeedTransition NeedToDecelerate	CalcDecelServoFunction(), InitializeDecelValues() w_DeceleratingVehicle thm_AdjustEngineThrottlePos brm_ApplyBrakeForce
NewAccelSpeedTransition NeedToMaintainSpeed	CalcSpeedServoFunction(), InitializeSpeedValues() w_MaintainingVehicleSpeed thm_AdjustEngineThrottlePos brm_ApplyBrakeForce
NewAccelSpeedTransition NeedToDecelerateToStop	CalcStopDecelFunction(), InitializeStopDecelValues() w_DeceleratingVehicleToStop thm_AdjustEngineThrottlePos brm_ApplyBrakeForce
NewAccelSpeedTransition NeedToMaintainStop	CalcStoppedFunction(), InitializeStopValues() w_MaintainingVehicleStopped thm_AdjustEngineThrottlePos brm_ApplyBrakeForce
w_AcceleratingVehicle TimeToAdjustAccelValues	AdjustAccelValues() w_AcceleratingVehicle thm_AdjustEngineThrottlePos brm_ApplyBrakeForce
w_DeceleratingVehicle TimeToAdjustDecelValues	AdjustDecelValues() w_DeceleratingVehicle thm_AdjustEngineThrottlePos brm_ApplyBrakeForce
w_MaintainingVehicleSpeed TimeToAdjustSpeedValues	AdjustSpeedValues() w_MaintainingVehicleSpeed thm_AdjustEngineThrottlePos brm_ApplyBrakeForce
w_DeceleratingVehicleToStop TimeToAdjustStoppingValues	AdjustStoppingDecelValues() w_DeceleratingVehicleToStop thm_AdjustEngineThrottlePos brm_ApplyBrakeForce
w_MaintainingVehicleStopped TimeToAdjustStoppingValues	AdjustStoppedValues() w_MaintainingVehicleStopped thm_AdjustEngineThrottlePos brm_ApplyBrakeForce

Input Conditions **Output Commands**

PLAN STATE TABLE

Figure 3.23: Illustration of the Plan state table in SpeedServo for the input command GoFwdAtSpeed. This RCS node receives a number of other commands such as PrepForEngineStart, GoBackwdAtSpeed, MaintainPark, EstopVehicleMotion, etc. Each of these input commands will have its own plan state table and provide task context to the sensory processing and world modeling processing to evaluate the situations relevant to that command.

Continuing the previous example with the speedServo node in the w_AcceleratingVehicle state, the state table checks each control cycle for the TimeToAdjustAccelValues state to also be true which causes a match with the left hand input conditions of line 6 and the resultant execution of this line's right hand entries. The state TimeToAdjustAccelValues is set by the world model processing after a sufficient delay since the last servo correction. This causes the integrator within the algorithm AdjustAccelValues() to wait

a sufficient length of time for the system to respond to the last output change before calculating the magnitude of the next error correction term. After this line matches, it will be a number of cycles before the world model processing evaluates TimeToAdjustAccelValues as true and causes the next execution of this algorithm.

ThrottleManager RCS Node

The SpeedServo state table specifies the thm_AdjustEngineThrottlePos as the output command to the ThrottleManager. The servo algorithm has generated an engine throttle position value for this command. ThrottleManger receives this command and invokes a Plan state table, AdjustEngineThrottlePos, to determine the appropriate motor position value in order to set the engine throttle to this commanded position. Figure 3.24 illustrates the linkage setup for the throttle control. This linkage set is sloppy and results in considerable backlash, such that the throttle motor has to back off four revolutions to begin moving the engine throttle (diesel injector pump lever) in the opposite direction.

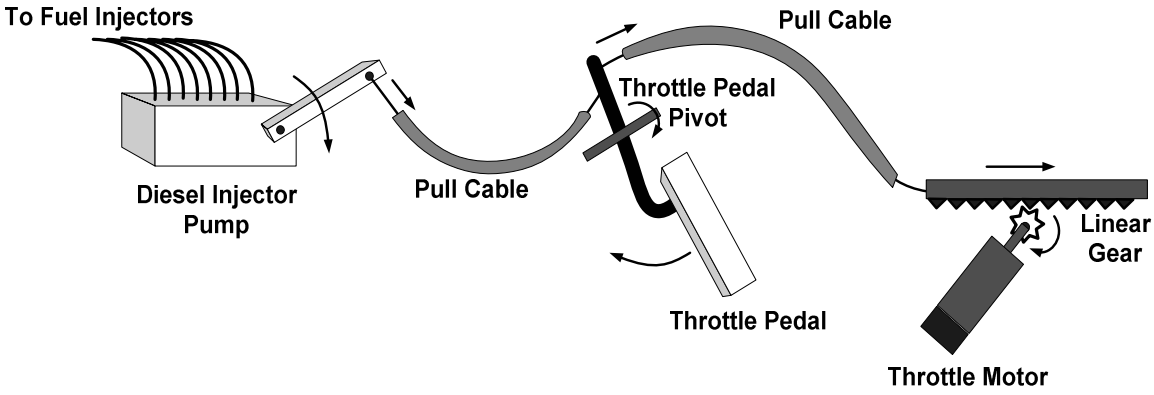


Figure 3.24: Throttle linkage on HMMWV. Throttle motor drives a linear actuator that pulls on cable connected to the throttle pedal mechanism which then pulls on a cable connected to the control lever of the diesel injector pump that controls the amount of fuel to the injectors. There is a potentiometer on the diesel injector pump lever to provide an absolute position measurement.

Part of the responsibilities of the ThrottleManager is to transform the input commanded engine throttle position (which is commanded as a percentage of full throttle) to the correct throttle motor position value, adjusting for nonlinearities and backlash, as well as calculating any required motor trajectory to control the motion of the throttle motor during any changes. This highlights another set of capabilities that are designed into these nodes. In many cases, a self-tuning process can be written for the creation of the values for the lookup tables for the different feedforward algorithms. At this node, the servo that calculates the correct throttle motor output at each control cycle uses a lookup table that encompasses the travel limits, non-linearities, and backlash between the throttle motor position and the diesel injector pump lever position. This table has been automatically generated by the execution of a plan state table (CalibrateEngineThrottle) that first drives the throttle motor in one direction until the motor amplifier indicates an increase in current over some threshold (which indicates the limit of travel in that direction) and records the value of a potentiometer on the injector pump lever, then reverses direction and performs a similar operation in the opposite direction. This sets the limits of travel of the injector pump lever and correlates them to motor position values and to percentages of full throttle. Next, the plan state table steps the throttle motor by fixed small increments (0.1 revolution) from the idle position to the full open throttle position, recording the values corresponding to calculated percentage values of the injector pump lever. The plan state table then reverses direction and performs the same process now going from full throttle to idle. At the end of execution of this plan state table, which takes less than two minutes to execute, the feedforward lookup table for the engine throttle servo function for the ThrottleManager RCS node has been

created. This is done at the startup of the vehicle each day and accommodates changes in the linkage and potentiometers automatically as well as being generic across any other vehicle this system is put on.

Similar plan state tables to create self-generated lookup tables are created at the BrakeManager node to correlate the commanded brake line pressure to the brake motor position value that has a similar linkage arrangement; and at the ThrottleMotorServo, BrakeMotorServo, and SteerMotorServo for the velocity feedforward servos at these nodes.

Accurate Vehicle Trajectory Control

The goal of the described lower level RCS echelons is to provide very accurate vehicle trajectory control. To competently perform autonomous on-road driving, it is necessary to very accurately control the vehicle to follow the path that the higher echelons in the system have decided to be correct to carry out our goal for the present state of the world. On-road driving involves holding in-lane maneuvers to 0.2 m accuracies, especially when negotiating in-lane obstacles such as potholes, debris, bicyclists, jersey barriers etc. The negotiation of turns at intersections, likewise, sets requirements for very accurate path control. An RCS solution of multiple layers of RCS echelons, which decompose the task from the high level goal path, down through layers of feedforward servo echelons to the actuator motor voltage levels that are calculated 500 times each second has been described. The result of this effort is depicted in Figure 3.26 which shows a performance metric data graph made from a real vehicle driving test. The solid line indicates a set of constant curvature arcs that were derived from the actual road geometry from a ground truth survey done on the NIST campus. These were provided to Elemental Movement as the goal set of lane segments along with a specification of speeds and accelerations. The described lower level mobility RCS echelons then executed this set of lane segments while a high accuracy differential GPS system (accurate to within 2 cm of ground truth) recorded the real-time position of the vehicle's center point. This real-time vehicle path is shown as the dotted line. The vehicle traveled up to 15 m/s along the straight stretches and slowed to approximately 6 m/s for the curves at intersections (which had radii of approximately 14 m). Throughout the entire test run, which was approximately 1.2 km and included four intersection turns, the vehicle followed the path with less than 0.2 m error (the grid in the Figure is 1.0 m squares).

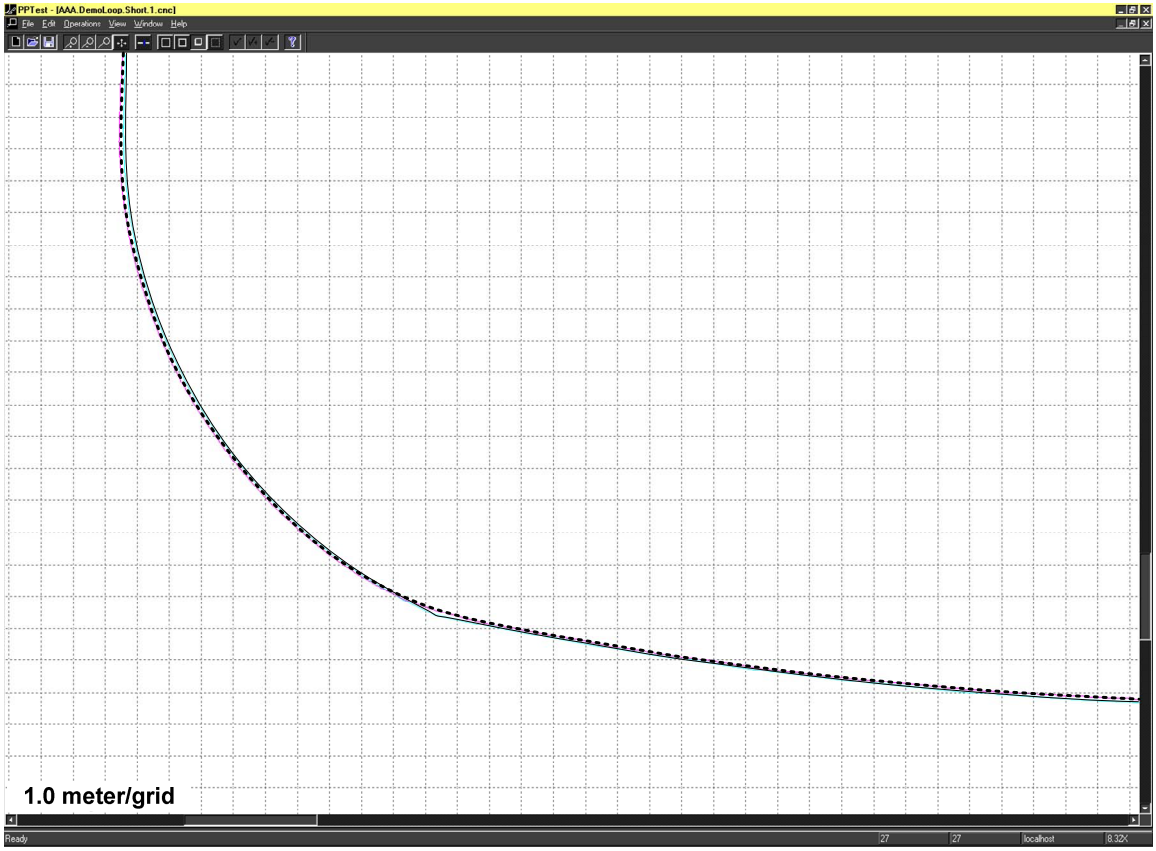


Figure 3.25: Data run with real vehicle. Solid line is the set of goal lane segments expressed as constant curvature arcs. Dotted line is the set of differential GPS points identifying the real path of the center of the vehicle within 2 cm absolute accuracy. Vehicle slowed from 15 m/s to 6 m/s to traverse intersection curve within the allowed maximum lateral acceleration.

This example shows how the 4D/RCS approach has been used in the development of a number of levels of task decomposition behavior generation where each echelon has performed the next lower level servo calculation starting with a real-time trajectory function generating the vehicle's next speed and steer values down through servos of speed and steer, down through servos of engine throttle position and braking force to the servoing of motors on the actuators. All of the echelons have used a feedforward type of model to estimate correct output values for the present state. This has led to very accurate low level vehicle mobility control that causes the vehicle to follow the goal paths generated by the higher level echelons. This capability is essential to competent autonomous on-road driving.

3.6 Conclusion

This chapter has described behavior generation for mobility within the RCS architecture. The first half of the chapter describes the overall approach through the description of three hierarchical levels. The three levels share a common set of commands for getting started and stopping, but each level has task commands specific to that level. A cooperative method of hierarchical planning using search in a graph linked to geometric space has been presented, along with problems circumvented by that method and issues it raises.

The vehicle level (topmost of the three) acts in a region about 100 m across with a cycle time of about one second. It uses rule-based planning working with a world model containing multiple feature layers. Plans are constructed by graph search and evaluated with the help of a plan simulator.

The autonomous mobility level considers a ten-meter square region at about 0.2 s cycle time. It collaborates with the vehicle level to perform cooperative planning, generating smooth paths consisting of arcs of circles and straight line segments with speeds and width tolerances. Planning is done both by rubber-banding and A* graph search.

The prim level plans and executes velocity states for constant curvature path segments of the order of 1 m long having specified tolerances and speeds. Cycle time is about 0.05 s. The prim level sends mobility state commands to the servo level, which transforms them into the actuator outputs for the vehicle's motors, brakes, etc. that actually generate motion.

The second half of the chapter has presented the details of a working controller for the NIST HMMWV. The complexities of this system have required additional control echelons to be included in the hierarchy and an extensive command vocabulary to be defined.

Research is still being conducted in all aspects of the control systems described in this chapter. The controllers developed in the first half of this chapter have been implemented as an open source 4D/RCS repository and are available on the web at http://sourceforge.net/projects/moast. This repository is designed to interface to another open source repository (http://sourceforge.net/projects/usarsim) that contains a high-fidelity agent simulator. These two repositories form the basis of the annual RoboCup Urban Search and Rescue Virtual League, with winning teams being required to contribute their control code back to the community. This is fostering the development of an open community of developers that is enhancing and elaborating on the vital work that has been presented in this chapter.

References

1. Albus, J., Huang, H., Messina, E., Murphy, K., Juberts, M., Lacaze, A., Balakirsky, S., Shneier, M., Hong, T., Scott, H., Proctor, F., Shackleford, W. P., Michaloski, J. L., Wavering, A., Kramer, T., Dagalakis, N., Rippey, W., Stouffer, K., Legowik, S., Bostleman, R., Norcross, R., Jacoff, A., Szabo, S., Falco, J., Bunch, B., Gilsinn, J., Chang, T., Meystel, A., Barbera, A., Fitzgerald, M., DelGiorno, M., and Finkelstein, R., "4D/RCS Version 2.0: A Reference Model Architecture for Unmanned Vehicle Systems," NISTIR 6910, Gaithersburg, MD, 2002.

2. Albus, J. and Meystel, A., *Engineering of Mind: An Introduction to the Science of Intelligent Systems*, John Wiley & Sons, Inc. 2001.

3. Astrom, K. and Haggund, T., *PID Controllers: Theory, Design, and Tuning*, Second ed., Instrument Society of America 1995.

4. Bacchus, F., "The AIPS '00 Planning Competition," *AI Magazine*, Vol. 22, No. 3, 2001, pp. 47-56.

5. Balakirsky, S., *A Framework for Planning with Incrementally Created Graphs in Attributed Problem Spaces*, IOS Press, Berlin, Germany, 2003.

6. Balakirsky, S. and Herzog, O., "Parallel Planning In Partitioned Problem Spaces," *5th IFAC Symposium on Intelligent Autonomous Vehicles, IAV 2004*, 2004.

7. Balakirsky, S. and Scrapper, C., "Knowledge Representation and Planning For On-road Driving," *Robotics and Autonomous Systems*, Vol. 49, No. 1-2, 2004, pp. 57-66.

8. Craig, J. J., *Introduction to Robotics*, Addison-Wesley 1986.

9. Dijkstra, E. W., "A note on two problems in connexion with graphs," *Numerische Mathematik*, Vol. 1, 1959, pp. 269-271.

10. Hart, P. E., Nilsson, N. J., and Raphael, B., "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *IEEE Transactions on Systems Science and Cybernetics*, Vol. 4, No. 2, 1968, pp. 100-107.

11. Hwang, Y. K. and Ahuja, N., "Gross Motion Planning - A Survey," *ACM Computing Surveys*, Vol. 24, No. 3, 1992, pp. 219-291.

12. Nilsson, N. J., *Artificial Intelligence: A New Synthesis*, Morgan Kaufmann Publishers, Inc., San Francisco, 1998.

13. Schlenoff, C., Madhavan, R., and Balakirsky, S., "Representing Dynamic Environments for Autonomous Navigation," *Proceedings IEEE International Conference on Intelligent Robots and Systems*, 2003.

14. Schwartz, J. T. and Sharir, M., "A Survey of Motion Planning and Related Geometric Algorithms," *Artificial Intelligence*, Vol. 37, 1988, pp. 157-169.

15. Sedgewick, R., *Algorithms*, Addison-Wesley 1983, pp. 130-137.

16. Williams, J. W. J., "Algorithm 232: Heapsort," *Communication of the ACM*, Vol. 7, 1964, pp. 347-348

17. Winston, P. H., *Artificial Intelligence*, Second ed., Addison-Wesley Publishing Company, Reading, Massachusetts, 2001.

Chapter 4

World Modeling and Knowledge Representation

Craig Schlenoff and **Elena Messina**
National Institute of Standards and Technology (NIST)
{craig.schlenoff,elena.messina}@nist.gov

1. Introduction

A core component of any deliberative control architecture is the knowledge that is represented within the system and the mechanisms that are present to make the maximum use of that information. Within 4D/RCS, this is referred to as knowledge representation and world modeling, respectively. This chapter describes some of the knowledge databases and world modeling processes that exist within 4D/RCS, and describe how they have been used in various implementations.

1.1. World Modeling and the Knowledge Database

As described in Chapter 1, the world modeling (WM)/knowledge database (KD) is one of the four main components within 4D/RCS. World modeling is a functional process that constructs, maintains, and uses a world model knowledge database in support of behavior generation and sensory processing processes.

The Knowledge Database consists of data structures and the static and dynamic information that collectively form a model of the world. The KD is the information needed by the WM to support the BG, SP, and VJ processes in each node. Knowledge in the knowledge database includes the system's best estimate of the current state of the world plus parameters that define how the world state can be expected to evolve in the future under a variety of circumstances.

4D/RCS is a hierarchical architecture, and as such, supports knowledge representation at different levels of abstraction. Traditionally, the lowest levels of the architecture primarily contain state variables such as actuator positions, velocities, and forces, pressure sensor readings, position of switches, gearshift settings, and inertial sensors for detecting gravitational and locomotion acceleration and rotary motion. The next higher level of the hierarchy (and above) contains map-based information, with decreasing resolution and increasing spatial extent as one proceeds higher up the hierarchy. The further up the hierarchy, a combination of map-based representations and object knowledge bases are used, which contain names and attributes of environmental features such as road edges, holes, obstacles, ditches, and targets. These maps represent the shape and location of terrain features and obstacle boundaries. Still higher up the hierarchy is symbolic information referring to the location of vehicles, targets, landmarks, and local terrain features such as buildings, roads, woods, fields, streams, fences, ponds, etc. The top levels of the hierarchy primarily deal with groups of objects, such as groups of people, buildings, or vehicles. These groups are treated as a single entity, with average characteristics (e.g., speed, location, color) used to describe them.

1.2. Chapter Organization

The 4D/RCS architecture is designed in such a way as to accommodate multiple types of representation formalisms and provide an elegant way to integrate these formalisms into a common, unifying framework. Some of these types of knowledge representation are shown in Figure 1. These formalisms range from iconic to symbolic and from procedural to declarative. Knowledge is captured in formalisms and at levels of abstraction that are suitable for the way that it is expected to be used. Different knowledge representation techniques offer different advantages, and 4D/RCS is designed in such a way as to combine the strengths of all of these techniques into a common unifying architecture in order to exploit the advantages of each. This

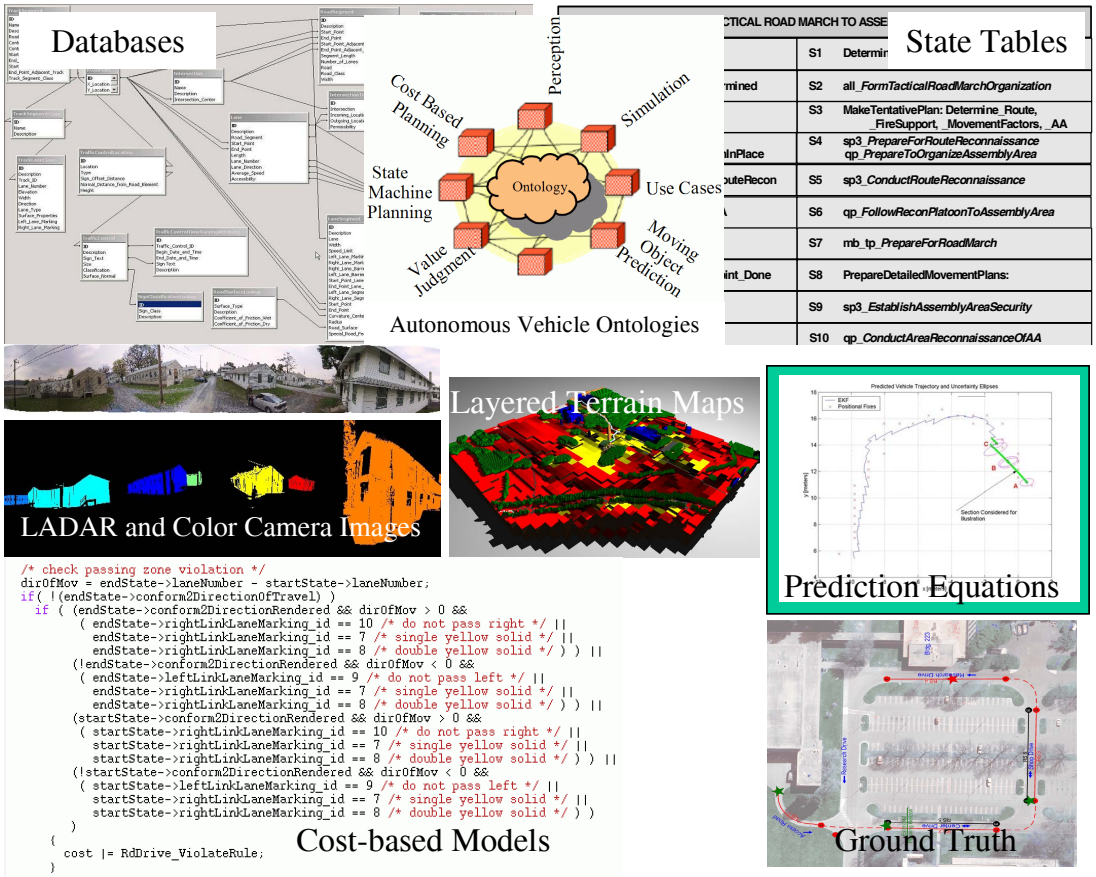


Figure 1: Knowledge Representations in 4D/RCS.

chapter elaborates on ongoing efforts in developing and implementing some of these representations. Section 2 provides an overview of other architectures and describes how they represent and integrate knowledge. Section 3 describes efforts in developing declarative knowledge bases in 4D/RCS, highlighting work in the creation of a Road Network Database (RNDB). Section 4 describes efforts in developing procedural knowledge bases, highlighting work in the creation of an Intelligent Systems Ontology. Section 5 describes efforts in developing world modeling processes, highlighting work in developing processes for moving object prediction. Section 6 discusses issues pertaining to principles for the choice of representations, complexity vs. performance, redundancy, how representations affect situation assessment, and concludes the chapter.

2. Related Work

Several architectural frameworks exist that rely on varying types of underlying knowledge representations. What is lacking is a widely accepted theoretical framework that can integrate multiple representation formalisms into a unified whole. One of the earliest frameworks was the ACT architecture [3]. ACT grew out of research on human memory. Over the years, ACT has evolved into ACT* and more recently, ACT-R. ACT-R is being used in several research projects in an Advanced Decision Architectures Collaborative Technology Alliance for the U.S. Army [18]. ACT-R is also being used by thousands of schools across the country as an algebra tutor – an instructional system that supports learning-by-doing.

Another well-known and widely used architecture is Soar [24,36]. Soar grew out of research on human problem solving, and has been used for many academic and military research projects in problem solving, language understanding, computational linguistics, theorem proving, and cognitive modeling. All knowledge in Soar is represented as productions. Each production represents a retrieval of a piece of

knowledge from long-term memory. The right side of the production represents knowledge of the actions; the left side represents the conditions under which it is appropriate to retrieve that knowledge into working memory.

Like Soar, Prodigy uses search through a problem space to achieve goals cast as first-order expressions [33]. The *Prodigy Description Language* is a declarative representation, based on first-order predicate logic. Thus, in Prodigy, all rules (and more generally knowledge) may be inspected by other rules, allowing the architecture to reflect on its own knowledge. Utilizing a declarative representation is a key component of Prodigy's glass box (transparent) design principle.

ICARUS encodes knowledge as reactive skills [47]. ICARUS uses a single type of representation – the hierarchy of probabilistic concepts – to represent objects, places, plans, and movements. This permits simple addition and modification of knowledge.

IMPRINT (IMproved Performance Research INtegration Tool) is a task description language designed for the Army to capture the procedural specification of tactical behavior scenarios [5]. It contains a dynamic, stochastic, discrete-event network modeling tool designed to help assess the interaction of soldier and system performance throughout the system lifecycle – from concept and design through field testing and system upgrades. IMPRINT has been integrated with ACT-R to model military behaviors [4]. IMPRINT uses an embedded discrete event task network modeling language as its engine. Task-level information is used to construct networks representing the flow and the performance time and accuracy for operational and maintenance missions.

EPIC (Executive-Process Interactive Control) is an architecture that models the detailed timing of human perceptual, cognitive, and motor activity, including the input/output characteristics of the nervous system connecting the higher level cognitive functions to the external world [22]. Human performance in a task is simulated by programming the cognitive processor with production rules organized as methods for accomplishing task goals.

The Polybot architecture [11] is designed to enable various modes of reasoning based on multiple types of data representations. Polybot is built upon a series of specialist modules that use any algorithm or data structure in order to perform inferences or actions. Since specialists may need to share knowledge, which they internally represent in different manners, a common propositional language for communicating information is part of the Polybot system. Examples of specialists implemented in Polybot include perception, a reactive motion planner, spatial location (using a cognitive map), causation (which uses production rules), and object identifier (using neural networks).

4D/RCS is a control system architecture inspired by a theory of cerebellar function [1]. 4D/RCS models the brain as a hierarchy of goal-directed sensory-interactive intelligent control processes that theoretically could be implemented by neural nets, finite state automata, cost-guided search, or production rules [2]. 4D/RCS is similar to other cognitive architectures in that it represents procedural knowledge in terms of production rules, and represents declarative knowledge in abstract data structures such as frames, classes, and semantic nets. 4D/RCS differs from other cognitive architectures in that it also includes signals, images, and maps in its knowledge database, and maintains a tight real-time coupling between iconic and symbolic data structures in its world model. 4D/RCS is also different in: a) its focus on task decomposition as the fundamental organizing principle; b) its level of specificity in the assignment of duties and responsibilities to agents and units in the behavior generating hierarchy; and c) its emphasis on controlling real machines in real-world environments.

3. Declarative Knowledge

Declarative knowledge refers to representations of objects and events and how these knowledge and events are related to other objects and events. It is sometimes described as providing the "what" versus the "how," which procedural knowledge contains. It allows one to think and talk about the world. For example,

declarative knowledge may represent a fact such as that Paris is the capital of France. It is often represented in a format that may be manipulated, decomposed, and analyzed by reasoning engines independent of its content. Unlike procedural knowledge, it does not describe how to perform a given task (e.g., how to bake a cake). Instead, it provides the ability to use knowledge in ways that the system designer did not foresee. Two classes of declarative knowledge captured within 4D/RCS are symbolic knowledge and metrical knowledge. Both of these are described below.

3.1. Symbolic Knowledge

3.1.1. Background

Symbolic representations provide ways of expressing knowledge and relationships, and of manipulating knowledge, including the ability to address objects by property. Much early work in robotics was carried out in the context of Artificial Intelligence (AI) research using symbolic representations [24,37,41]. This had the result of uncoupling robotics from the geometry and dynamics of the real world, and focusing on purely symbolic approaches to perception, planning, and reasoning [16]. Probably the best-known symbolic representation developed in classical AI is frame-based [32]. A frame defines a stereotypical situation, which is instantiated when appropriate. There are slots to be filled out for the particular instantiation. For example, there would be a series of frames related to a building, essentially defining what the robot may be expected to encounter as it travels inside the building. A frame for a room may have concepts for "floor," "ceiling," "right wall," "left wall," "far wall," and so on. The robot would try to find entities using its vision system to fill in the slots for these concepts.

Tying symbolic knowledge back into the spatial representation provides symbol grounding, thereby solving the problem inherent to purely symbolic knowledge representations. It also provides the valuable ability to identify objects from partial observations and then extrapolate facts or future behaviors from the symbolic knowledge.

3.1.2. Symbolic Knowledge in 4D/RCS

Within 4D/RCS, mainly two types of symbolic representations have been implemented thus far: ontologies and relational databases. An ontology for driving determines if objects in the environment are potential obstacles to the autonomous vehicle [42,44]. The system is composed of an ontology of objects representing "things" that may be encountered in the current environment, in conjunction with rules for estimating the damage that would be incurred by collisions with the different objects, as a function of the characteristics of the autonomous vehicle, including the type of vehicle, speed, etc. Automated reasoning is used to qualitatively estimate collision damage, and this information is fed to the route planner to help it decide whether to avoid the object.

An example of the second type of symbolic representation is the NIST Road Network Database, which is described below.

3.1.3. Symbolic Knowledge Highlighted Example: The NIST Road Network Data Base

For an autonomous vehicle to be able to navigate a road network, it must be aware of and must respond appropriately to any object it encounters. This includes other vehicles, pedestrians, debris, construction, accidents, emergency vehicles etc. and it also includes the roadway itself. The road network must be described in such a way that an autonomous vehicle knows where the road lies, rules dictating the traversal of intersections, lane markings, road barriers, road surface characteristics, and other relevant information.

This section provides detailed information about the Road Network Database being developed at NIST as part of the Defense Advanced Research Projects Agency (DARPA) Mobile Autonomous Robotics Systems (MARS) Program. The purpose of the Road Network Database is to provide the data structures necessary to capture all information necessary about road networks so that a planner or control system on an autonomous vehicle can plan routes along the roadway at any level of abstraction. At one extreme, the

database should provide structures to represent information so that a low-level planner can develop detailed trajectories to navigate a vehicle over the span of a few meters. At the other extreme, the database should provide structures to represent information so that a high-level planner can plan a course across a country. Each level of planning requires data at different levels of abstraction, and as such, the Road Network Database must accommodate these requirements.

The fundamental components of the Road Network Database are described below, and shown in Figures 2(a-g):

- **Junctions –** A junction is a generic term referring to two or more paths of transportation that come together or diverge, or a controlled point in a roadway. Paths of transportation could be roadway or non-roadway paths. Examples of junctions are lane splits, forks in the road, merges, intersections, pedestrian crossings, ferry crossings, railroad crossings. Examples of controlled points in the roadway are drawbridges, toll plazas, and guard gates. Junctions are an abstract supertype in the sense that a junction must be one of the types listed above.
- **Intersections** - Intersections are a type of junction in which two or more separate roads come together.
- **Lane Junctions** - A lane junction is a location in a junction in which two or more lanes of traffic overlap. A lane merge contains a lane junction starting at the point in which the two lanes begin to come together and end at the point in which the two lanes are completely together as one. A lane fork contains a lane junction at the point where the lanes begin to fork and ends at the point where the two lanes are completely separated. An intersection contains a lane junction at all points in which the lanes from the two or more intersecting roads overlap.
- **Road –** A road is a stretch of travel lanes in which the name of the travel lanes does not change. An example is "Main Street" or "Route 95."
- **Road Segment** - A road segment is a uni-directional stretch of roadway bounded by intersections. A road segment is roughly analogous to a "block". So the uni-directional piece of road bounded by 1st Street and 2nd Street would be a road segment.
- **Road Element** - A road element is a uni-directional stretch of roadway bounded by any type of junction. Unlike road segments, road elements can be bounded by merging lanes, forks in the road, Junctions include two or more lanes merging together, a fork in the road, a pedestrian crossing, a toll booth, a draw bridge, an intersection, etc.
- **Lane Cluster** - A lane cluster is a set of uni-directional lanes (with respect to flow of traffic) in which no physical attribute of those lanes change over the span of the lane segment. Unlike a road element, lane clusters are not required to be bounded by junctions. Characteristics of the road that cannot change include the addition or subtraction of shoulders, the width of the lane, the separation of lanes to form a median, change in paint striping, and change in lane barriers.
- **Lane** - A lane is a single pathway of travel that is bounded by explicit or implicit lane marking. Lanes span the length of a lane cluster in which they are a part of.
- **Lane Segment** - A lane segment is the most elemental portion of a road network captured by the database structure. Lane segments can be either straight line or constant curvature arcs. In the case of a straight line, the location of the lane segment if fully defined by the beginning and end point of the lane segment. For a constant curvature arc, the lane segment is defined by the beginning and end of the lane segment and the curvature center point. One or more lane segments compose a lane.
- **Junction Lane Segments -** A junction lane segment is a constant curvature path through a portion of a lane junction. Apart from some subtle differences pertaining to connectivity of these junction lane segments, they are extremely similar to lane segments as described above.
- **Time Varying Attribute Tables –** There are a number of tables in the database that address attributes of the above structures that may vary as a function of time. These attributes include speed limits on roadways, the average speed on a roadway, the direction of travel on lanes, the accessibility of a lane (e.g., HOV), and the legal traversibility through intersection (e.g., no right turn between 3 PM and 6 PM on weekdays). In these tables, the pertinent values for these attributes are associated with time intervals.

- **Lookup Tables –** There are a number of lookup tables that include a complete list of all possible values that certain attributes in the certain data structures may have. Lookup tables are used when possible values for a given attribute are finite, and there is value in enumerating them in a table. At the time this chapter was written, there are seven lookup tables in the database:

1. accessibility restrictions on lanes (e.g., HOV-2, HOV-3, cabs only, police only),
2. possible lane barriers on the side of lanes (e.g., jersey barrier, curb, guard rail),
3. lane markings on the side of lanes (e.g., solid yellow line, double solid yellow line, dashed white line, solid white line),
4. lane types (e.g., traversable, shoulder),
5. road class (e.g., interstate highway, beltway, country road, residential, road),
6. road surface (e.g., asphalt, dirt, pebbles), and
7. special road features (e.g., bridge, tunnel).

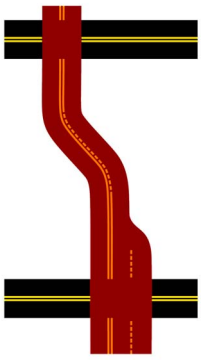


Figure 2a: Road

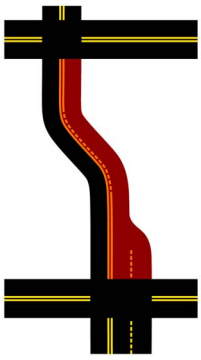


Figure 2b: Road Segment

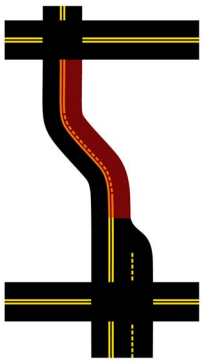


Figure 2c: Road Element

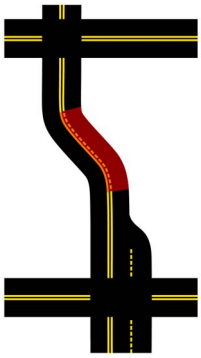


Figure 2d: Lane Cluster

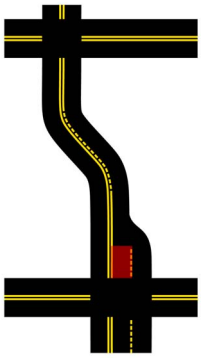


Figure 2e: Lane

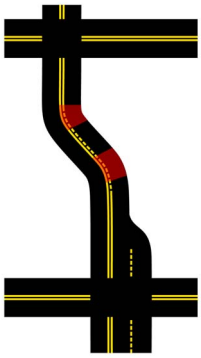


Figure 2f: Lane Segment

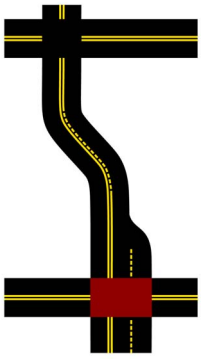


Figure 2g: Intersection

As stated earlier, this data structure is designed to accommodate a control system that may contain planners with various levels of abstraction. This section will provide insight into data structures that will be most appropriate for planners at different levels. The planners, their descriptions, and the data structures which best correspond to their level of responsibility are shown in Table 1. More detail about this table can be found in Chapter 2, Figure 9.

Planner Name	**Planner Description**	**Appropriate Data Structures**
Destination Planner	Plans the sequence of route segments to get to commanded destination goal. Outputs commercial trip planner-like (e.g. MapQuest) directions Plans on the order of 1 to 2 hrs into the future Plans > 10 km distances	Roads Road Segments Intersections Forks (not yet defined) Merges (not yet defined)
Route Segment Planner	Decides on real-time goal lanes for road segments and for negotiating intersections. Deals with intersections, forks, merges, etc. Plans on the order of 10 min into the future Plans up to 10 km distances	Road Segments Road Elements Intersections Forks (not yet defined) Merges (not yet defined)
Drive Behavior Planner	Develops low-level behaviors for negotiating intersections and deciding when to change lanes. Plans on the order of 100 s into the future. Plans up to 500 m distances	Lane Clusters Lanes Intersection Forks (not yet defined) Merges (not yet defined)
Elemental Maneuver Planner	Carries out real-time maneuvers to slow down, stop, speed up, and change lateral position. Plans on the order of 10 s into the future Plans up to 50 m distances	Lanes Lane Segments
Goal Path Trajectory Generator	Calculates the lane segment path dynamic trajectory as a goal path to carry out commanded move while controlling for skid and immediate obstacle response. Plans on the order of 1 s into the future Plans up to 5 m distances	Lane Segments

Table 1: Planner to Data Structure Mapping.

At the time this chapter was written, the database was at Version 1.0, and had been implemented as part of two planners being developed within NIST (a cost-based and a finite state machine-based planner) within a simulated environment. In both cases, this database was part of the underlying knowledge representation that allowed the planners to better understand the environment to enable more appropriate plan generation. The RNDB has also been implemented in a series of algorithms performing moving object prediction for on-road driving, which will be further discussed in Section 5.2.

Though a considerable amount of time and effort has been put into the database, there is still quite a bit of work that has yet to be accomplished. Additional types of junctions must be included, including merges, forks, pedestrian crossings, and railroad crossings. More information about roads also needs to be included, such as the overall width of the road, so that obstacles can be better placed on the roads. The database also needs to continue to be "stress tested" in simulated and real environments to ensure its consistency and completeness.

3.2. Metrical (Spatial) Knowledge

3.2.1. Background

Metrical knowledge is often spatial in nature and is either in 2D or 3D grids and higher-level geometric constructs, such as edges and surfaces. The knowledge is tied to a coordinate system and allows for distance measurements to be computed between the locations within that system. Additional knowledge is associated with a location. The value of each grid cell may be Boolean data (e.g. indicating whether the cell is occupied or not) or real number data representing a physical property such as light intensity, color, altitude, range, or density. Each cell may also contain spatial or temporal gradients of intensity, color, range, or rate of motion. Cells may also point to specific geometric entities (such as an edge, vertex, surface, or object) to which its contents belong.

Digital maps are a natural way of modeling the environment for path planning and obstacle avoidance. Digital terrain maps are referenced to some coordinate frame tied to the ground or earth and hence also facilitate data fusion be it from multiple sensors or from *a priori* data. Although commercial digital terrain maps often have a grid-based implementation (especially for the elevation layer), features are typically represented as vectors. The underlying database implementation facilitates spatial queries even for features that are represented by polygons or polylines. In many mobile robots, a grid-based approach is easier to implement and maintain in real-time. In this case, a map may have multiple layers that represent different "themes" or attributes at each grid element. For instance, there may be an elevation layer, a road layer, a hydrology layer, and an obstacle layer as shown in Figure 3. The software can query if there is a road at grid location (*x, y*) and similarly query for other attributes at the same (*x, y*) coordinates. If the system being implemented is truly three-dimensional, then the queries can be made according to (*x, y, z*). This feature is important for accurately capturing features such as road overpasses and subterranean tunnels.

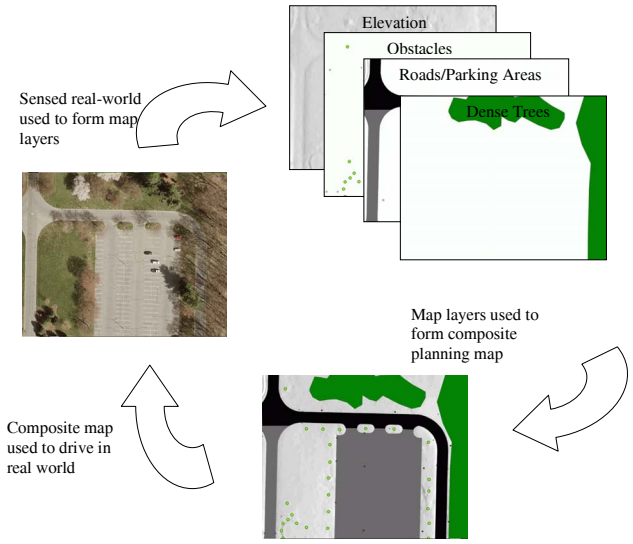


Figure 3: Multiple Layers.

A large number of the implemented mobile robot systems have relied on spatial representations. Decomposing the space that the robot has to travel into uniform or non-uniform regions is one approach. Grid-based structures [10,35] are a convenient means of capturing input from the robot's sensors, especially if multiple readings from one or more sensors are to be fused. They have the advantage of being easy to implement and maintain, due to their uniform, array-like structure. A probability or certainty measure can be assigned to each grid cell indicating the degree of confidence that the cell is really occupied as opposed to purely open space, resulting in each location being marked as probably occupied, probably empty, and unknown. This type of representation is also referred to as an evidence grid [30]. Figure 4 shows an example of a grid representation. The robot location is indicated by cells marked with an "R." The numbers in the other cells are an indicative of the number of times that an obstacle has been detected by sensors within that space. The higher the number, the more probable it is that the cell is occupied.

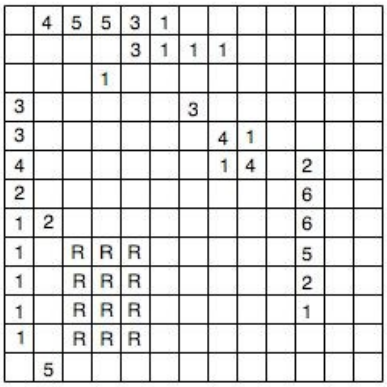


Figure 4: Grid Representation.

Furthermore, it is fairly straightforward to implement path planning and obstacle avoidance algorithms that use a regular structure, which can be readily translated into a graph that is searched (for instance using Dijkstra [14] or A* [38]) to find the lowest-cost path – typically based on shortest distance. A node is placed at the center of each grid location and it can be connected to adjacent cells via arcs that are assigned costs. The costs may be based on distance traveled between cells and may also incur a penalty based on the actions that the robot performs and the state the robot is in once it reaches a grid location (e.g., occupying a grid location that is already occupied). In the most simplistic approach, each cell can be connected to its nearest 4 or 8 neighbors. More efficient approaches build the graph connecting only empty cells or by using other techniques such as visibility graphs [39] or Voronoi borders [15]. The grid itself can be represented more compactly by using adaptive tesselation approaches. These include quadtrees which are efficient if the environment is not uniformly cluttered or when additional spatial information such as depth must be captured. [40,51] describes the use of quadtrees as a two and a half dimensional (2.5 D) approach to capturing the geometry of a lake bed for underwater autonomous vehicles. Multi-resolutional approaches, such as in [9,40] also improve efficiency by giving the cells closer to the robot higher resolution than those further away.

Approaches that tessellate space may need to represent more than two (or two and a half) dimensions. For instance, in cases where a two-dimensional spatial representation is inadequate, the evidence grid approach has been extended to three dimensions [34]. In many applications, it is insufficient to have the robot plan a path that only avoids obstacles. Additional constraints, such as non-planar terrain and the robot's own kinematics and dynamics often need to be taken into consideration. Considering velocity and acceleration while generating the robot's path significantly increases the state space for planning, so it tends to be done in two stages. Generally, the path planning process produces a coarse set of waypoints, which are then smoothed by another process that takes into account the robot's dynamic constraints. However, for systems with complex dynamics (e.g., legged robots, two-wheeled vehicles [23], soccer playing robots, or hovercraft [25]), it may be inadequate to ignore dynamics during the obstacle-avoidance planning process, therefore explicitly modeling the systems' dynamics may be necessary to guarantee collision-free

trajectories.

Some researchers have successfully demonstrated mobile robot systems that use only the sensor image ("windshield view"), also known as the iconic representation, to plan within. From [20]: "According to the model being proposed here, our ability to discriminate inputs depends on our forming 'iconic representations' of them. These are internal analog transforms of the projections of distal objects on our sensory surfaces." This may be two-dimensional spatially, as is the case for CCD (Charged Coupled Device) cameras, or three-dimensional, in the case of range sensors, such as LADARs. Some mobile robots successfully accomplish their goals by planning based on purely the sensor image view. This is particularly true for road-following systems, such as those by Dickmanns [13], where road edges are extracted by sensor processing algorithms and used to plan the vehicle's steering command in the image frame.

Grid-based and other spatial representations vary in choice of coordinate systems and in the relationship to the robot itself. Some implementations use polar coordinates because the sensor data is returned in the form of distance (to object) and angle, reducing the number of calculations in constructing the map and in planning motion. The robot is always at the origin of the coordinate system in this case. However, it is more difficult to maintain a global map as the robot traverses the environment. The majority of implementations use a Cartesian coordinate system. In some approaches, the map is centered on the robot's current location and oriented with respect to the robot. Sensor information is easily placed within the map, but the entire map must be transformed when the robot changes location or orientation (assuming that previous information is kept). Some systems maintain the maps in an absolute global reference frame (for example, based on magnetic north). This facilitates localization with respect to global positioning systems, registration with *a priori* maps, and landmark-based navigation, but requires the transformation from the local sensor frames to the global one.

Other spatial representations are based on the geometric boundaries within the environment [46], such as planar surfaces [26]. These representations may augment the iconic or grid-based ones and often provide efficiencies by providing more compact descriptions of an environment, especially for indoor applications or highly structured environments. Describing a wall as a plane or a line is more efficient storage-wise versus a set of grid cells. However, additional computations by grouping algorithms that process adjacent occupied cells or convert individual pixels into higher-level geometric entities are required to achieve this reduction in memory or disk requirements.

3.2.2. Metrical Knowledge in 4D/RCS

Spatial (metrical) knowledge is an important component of the world model in 4D/RCS and has been implemented at various levels of the hierarchy for control of mobile vehicles. This section will describe examples of implementations of such knowledge. In all implementations, there is an emphasis on supporting

(a) fusion of sensor information from multiple sources though the use of a coordinate-based representation,
(b) multiple levels of resolution within the hierarchy, and
(c) planning algorithms, which primarily use graph-based approaches.

A diagram showing the canonical definition of knowledge at the various levels of a 4D/RCS control system for a mobile robot was shown in Figure 4 in Chapter 1. Metrical knowledge is captured in the "windshield" or sensor views as well as overhead map-based representations. Metrical information travels up and down the hierarchy. The semantic knowledge, the geometric representation, and the resolution of the data vary at the different levels.

This section present an example based on an early implementation of the Demo III XUV Program's world model [21]. In this system, at the Autonomous Mobility (AM) level, the world model fuses information from multiple sensors, including navigation sensors, LADAR, and stereo vision within a uniformly-decomposed occupancy grid. The navigation system provides information about the vehicle's current position, orientation, speed, and velocity. This enables the system to localize sensed information within the coordinate frames. Data from the LADAR sensor includes a range image processed to provide an array in

which each element contains range value (distance in meters from the sensor), position elevation, obstacle label (whether it meets the criteria to be considered an obstacle or not), and terrain class label (tall grass, ground, or cover). In some configurations, the Demo III vehicle is equipped with two pairs of stereo cameras. One provides color imagery, while the other provides infrared (thermal) data. The output from the stereo processing contains information for each cell, including the cell's position, elevation (m) and confidence, terrain roughness and confidence, whether the cell includes an obstacle, and if so, if it is negative or positive, and the terrain class (tall grass, bush, tree, rut, soil, rock). The inputs from the various sensors are fused into a grid-based map. This is facilitated by the use of a common, uniform spatial decomposition. For each planning cycle, a copy of the obstacle map is rotated from a north-oriented into a vehicle-oriented map and sent to the planner. The resulting world model at the AM level includes a header that defines, among other things, the configuration of the occupancy map (e.g., the number of grid cells, and the size of each cell). A typical AM level has an occupancy map of 40 cm (301 by 301) square cells.

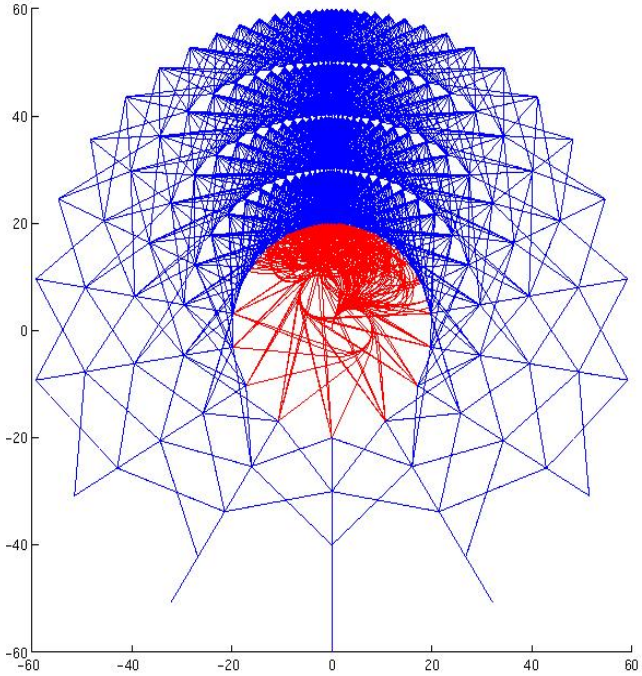


Figure 5: Possible Paths From A Hard Right Wheel.

The world model is enriched through the addition of feasible trajectories computed *a priori* for the Demo III vehicle. A web of potential path segments that extend out to 50 m is used by the path planner to select trajectories that can be driven by the vehicle. If there are no obstacles, the vehicle can drive on any combination of these path segments. There are two types of path segments, straight and curved. Curved segments extend out to 20 m from the vehicle. Each is a series of clothoid segments that are kinematically feasible based on the turn rate of the steering wheel. These paths are generated through offline simulations for different initial speeds and steering wheel positions. Initial steering position is a major factor influencing the paths the vehicle can travel. Figure 5 shows allowable paths with initial steering wheel position to the right and at two different velocities. Straight path segments are used from 20 m to 50 m.

The Vehicle level, immediately above Autonomous Mobility, also relies on metrical maps. Instead of searching within a uniform grid, the Vehicle level creates a graph where each node corresponds to a state that the vehicle may visit. These nodes are a combination of randomly thrown points and interesting locations (states) like roads and bridges. An example of such graph can be seen in Figure 6. In this figure, the vehicle is located at the center of the map. The size of the map is about 500 m on a side, with an

underlying grid composed of 4 m cells. The grid-like background shows the internal representation of the Vehicle Level WM, and the straight segments are possible actions that move the vehicle from one end of the segment to the other. The curvy segments represent segments of very high cost as evaluated by the Value Judgment function at the Vehicle level. The reason for the high cost in this case is that they cross through walls, fences, or forest. The underlying grid at this level was composed of 15,625 cells while the graph to be searched for the shown example was only composed of around 2000 nodes, so the computational advantages are evident. In this case, the path found by the vehicle level planner is shown as a set of marbles driving NW. The distance between the nodes in the found trajectory is under 50 m, therefore the subordinate level can find an accurate (and dynamically correct) trajectory within its map, and avoid obstacles that cannot be seen in the coarse representation of this level. The lighter colored trajectory represents the path sent by the supervisor level [7].

In the example shown in Figure 6, the Vehicle Level WM receives information from lower levels through the use of sensor processing and information from higher levels through filters. The actual control and low-level processing of raw, high-resolution, sensor data is performed at the level subordinate to the vehicle level (the Autonomous Mobility level). The vehicle level receives the subsystem level's representation, and performs processing to convert this data into a resolution and form appropriate for the vehicle level. For example, high-resolution obstacle data will be converted into mobility corridor estimates. Information received from higher levels includes such items as *a priori* map information, which may contain terrain and constructed features such as roads, and mission-specific designated constraints on individual vehicles. This information is filtered, and the relevant information for this particular vehicle is stored in the Vehicle Level WM.

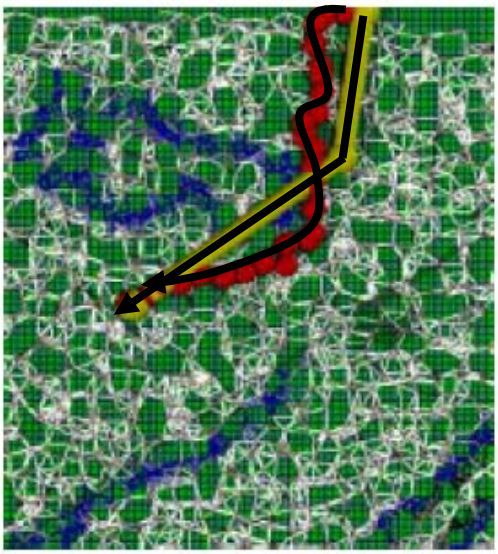


Figure 6: Vehicle Level Graph and Planned Paths.

4. Procedural Knowledge

4.1. Background

Procedural knowledge, or know-how, is the knowledge of how to perform tasks. Procedural knowledge is different from other kinds of knowledge, such as declarative knowledge, in that it can be directly applied to a task. One limitation of procedural knowledge is its job-dependence. The primary advantages of procedural knowledge is that 1) heuristic or domain-specific knowledge can be represented, 2) extended logical inferences, such as default reasoning, is facilitated, 3) side effects of actions may be modeled, and 4) the representation often provides faster usage in a performance system. The primary disadvantages are 1)

representations are often not complete – not all cases may be represented, 2) the representations are often not consistent – not all deductions may be correct, and 3) modularity is sacrificed – changes in the knowledge base migh have far-reaching effects. Productions are a common means of representing procedural knowledge.

4.2. Procedural Knowledge in 4D/RCS

Within 4D/RCS, procedural knowledge is primarily used for planning and control purposes. Two primary planning approaches are implemented, each representing procedural knowledge differently: Finite State Machines (FSM) and cost-based paradigms. In both cases, the application and domain-specific tasks and commands are first defined through a rigorous domain analysis process. The control hierarchy is designed by detailing the responsibilities of each control node, including inputs from the higher-level supervisor and outputs (as commands) to its subordinate nodes.

In the FSM approach, as described in Chapter 2, each of these command decompositions at each node will be represented in the form of a state-table of ordered production rules. The sequence of simpler output commands required to accomplish the input command and the named situations (branching conditions) that transition the state-table to the next output command are the primary knowledge represented in this approach. Each node therefore contains labeled representations of the states and transitions, which is beneficial in terms of making the reasoning of the system explicit [8]. FSM's have the advantage of making the decision criteria and logic obvious to a human reading the code. However, they require the programmers to consider and handle all possible situations ahead of time, which is often not realistic for robots operating in complex situations and environments.

The cost-based approach combines a graph-based search technique with a set of knowledge modules that simulate the effects of alternative actions and provide input to a unified cost model [6]. Different feature layers are discretized. Examples of feature layers are elevation, road networks, and vegetation. The planner at a given level sends candidate trajectories to simulators that compute the cost of state transitions for each of the relevant feature layers. For instance, a proposed path may take the vehicle from an on-road location to off-road. The cost associated with this is dependent on the context of the situation – if going off-road avoids a pedestrian on the road (which would be noted by another feature layer, possibly the obstacle one) this is an acceptable cost. Similarly, the cost/benefit of running a red light would be substantially different for a casual driver than it would be for a police vehicle responding to an emergency. Ontologies and other knowledge bases support the generation of cost models during execution. Whereas this cost-based approach is more general than the FSM, it also more challenging in terms of defining the appropriate costs for each action, especially since they will be combined. This is a good candidate for the application of learning to develop the cost models. In general, graph-based representations can result in an explosion of data (nodes and arcs connecting the nodes) and hence can have very poor performance characteristics when the graph is being searched. This is a concern especially for real-time systems, such as mobile robots. When a robot plans its motions, it must be able to react within an appropriate amount of time to obstacles or events. However, there are several techniques to mitigate these concerns, some of which were noted earlier in this chapter. Other means of mitigating performance issues include reusing parts of the already-processed graph (e.g., Dynamic A*) [49] and using sparse representations that include only relevant features, such as the extrema of an obstacle instead of a uniform grid of the environment [6].

Building off of the knowledge gathered in the FSM approach, an ontology was developed. This ontology, named the Intelligent Systems Ontology, is the highlighted example of procedural knowledge and is discussed below.

4.3. Procedural Knowledge Highlighted Example: The Intelligent Systems (IS) Ontology

The level of automation in ground combat vehicles being developed for the Army's objective force is greatly increasing over the Army's legacy force. This automation is taking many forms in emerging ground vehicles; varying from operator decision aides to fully autonomous unmanned systems. The development of these intelligent systems requires a thorough understanding of all of the intelligent behavior

that needs to be exhibited by the system so that designers can allocate functionality to humans and/or machines. Traditional system specification techniques focus heavily on the functional description of the major systems of a vehicle and implicitly assume that a well-trained crew would operate these systems in a manner to accomplish the tactical mission assigned to the vehicle. In order to allocate some or all of these intelligent behaviors to machines in future ground vehicles, it is necessary to be able to identify and describe these intelligent behaviors.

The U.S. Army Tank Automotive Research, Development and Engineering Center (TARDEC) has funded DCS Corporation and NIST to explore approaches to model the ground vehicle domain with explicit representation of intelligent behavior. This exploration has included the analysis of modeling languages (i.e., UML, DAML, OWL) as well as reference architectures. A major component of this effort has been the development of an IS Ontology.

NIST and DCS Corporation have taken the view that an IS can be viewed as a multi-agent system, where agents can represent components within the vehicle (e.g., a propulsion system, a lethality system, etc). In addition, an Intelligent Ground Vehicle (IGV), as a whole, can serve as a single agent within a troop, platoon, or section, where multiple IGVs are present. In order for a group of agents to work together to accomplish a common goal, they must be able to clearly and unambiguously communicate with each other without the fear of loss of information or misinterpretation. The IGV Ontology has been used to specify a common lexicon and semantics to address this challenge.

4.3.1. The Scenario

This effort uses knowledge derived from the task analysis of scenarios of a light cavalry troop's execution of a Conduct Tactical Road March to Assembly Area mission. In particular, the part of this mission that focuses on the route reconnaissance component by the Scout Platoon has been analyzed. This is done through scenarios that are examined at more and more detailed levels starting at the Troop Commander Level, which will perform a number of planning activities to better identify the priority information items of the route, to define the march column organization, and to specify the formation and movement technique. The troop commander will then dispatch a scout platoon to conduct a route reconnaissance. The scout platoon leader will do finer level planning, organizing the platoon's sections of vehicles and assigning commands to each section leader to do reconnaissance of different areas along the route while maintaining security. Each section leader will evaluate the environment to provide detailed tactical goal paths for each of his vehicles, coordinating their movement by the use of detailed motion commands to control points along with security overwatch commands. Each vehicle, in turn, performs detailed sensory processing to carry out careful analysis of the terrain in context of the mission, security, stealth, and traversability. Each vehicle then decides its optimal real-time path. If some aspect, such as a water obstacle, constrains the vehicle from following the general goal path laid out by the section leader, the vehicle does reconnaissance, moves to a secure point, and reports to the section leader. If the constraint affects the operation of the entire section (e.g. the water obstacle stretches across the entire area that the section is assigned), then the section leader coordinates his vehicles to do reconnaissance and to take up secure positions. The section leader then reports to the platoon leader.

These scenarios provide a rich set of knowledge of organizational structure, activities, commands, rules, status, sensory processing, objects and world states to be recognized, adaptation to events, and procedures required for successful execution.

4.3.2. The IS Ontology

4.3.2.1. Ontology Language

The IS Ontology uses that OWL-S upper ontology [50] as the underlying representation to document 4D/RCS in a more open XML (eXtensible Markup Language) format. OWL-S is a service ontology, which supplies a core set of markup language constructs for describing the properties and capabilities of services in an unambiguous, computer-interpretable format. OWL-S, which is being developed by the Semantic

Web Services arm of the DARPA Agent Markup Language (DAML) program, is based on the OWL [19]. OWL is an extension to XML and RDF (Resource Description Framework) schema that defines terms commonly used in creating a model of an object or process. OWL is a World Wide Wide Consortium (W3C) recommendation, which is analogous to an international standard in other standards bodies.

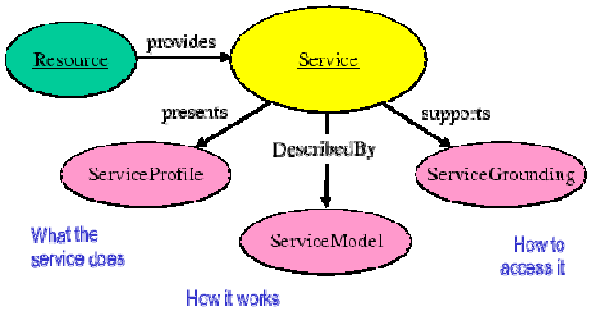


Figure 7: OWL-S Ontology Structure.

OWL-S is structured to provide three types of knowledge about a service (Figure 7), each characterized by the question it answers:

- What does the service require of the user(s), or other agents, and provide for them? The answer to this question is given in the "profile". Thus, the class SERVICE presents a SERVICEPROFILE.
- How does it work? The answer to this question is given in the "model". Thus, the class SERVICE is describedBy a SERVICEMODEL.
- How is it used? The answer to this question is given in the "grounding". Thus, the class SERVICE supports a SERVICEGROUNDING.

Later in this chapter, it is shown how the OWL-S concepts have been used to model a tactical behavior for an intelligent ground vehicle.

4.3.2.2. Tools

Before the ontology can be built, a decision has to be made as to which tool (or set of tools) should be used to enter, capture, and visualize the ontology. This work uses Protégé [45], which is an ontology editor, a knowledge-base editor, as well as an open-source, Java tool that provides an extensible architecture for the creation of customized knowledge-based applications. Protégé was chosen due to its strong user community, its ability to support the OWL language, its ease of use (as determined by previous experience), and its ability to be extended with plug-ins such as visualization tools (discussed below).

4.3.2.3. Using OWL-S to Model the Scenario

Both the 4D/RCS methodology and the OWL-S upper ontology are based on the concept of agents, service that the agents can perform, and procedures that the agents follow to perform the services. As such, there is a very clean mapping between the information that comes out of the 4D/RCS methodology and the OWL-S upper ontology. This section describes that mapping.

The first step involved setting up the agent hierarchy. Figure 8 shows an agent hierarchy for a light cavalry troop. A detailed description of this hierarchy is outside the scope of this chapter. All of these agents are modeled in OWL-S as subclasses of the IGV Agent class, which is a subclass of the ServiceResource class defined in the OWL-S upper ontology. Also specified in the constraints for each class is whom each agent can send external service requests to and who they can receive them from.

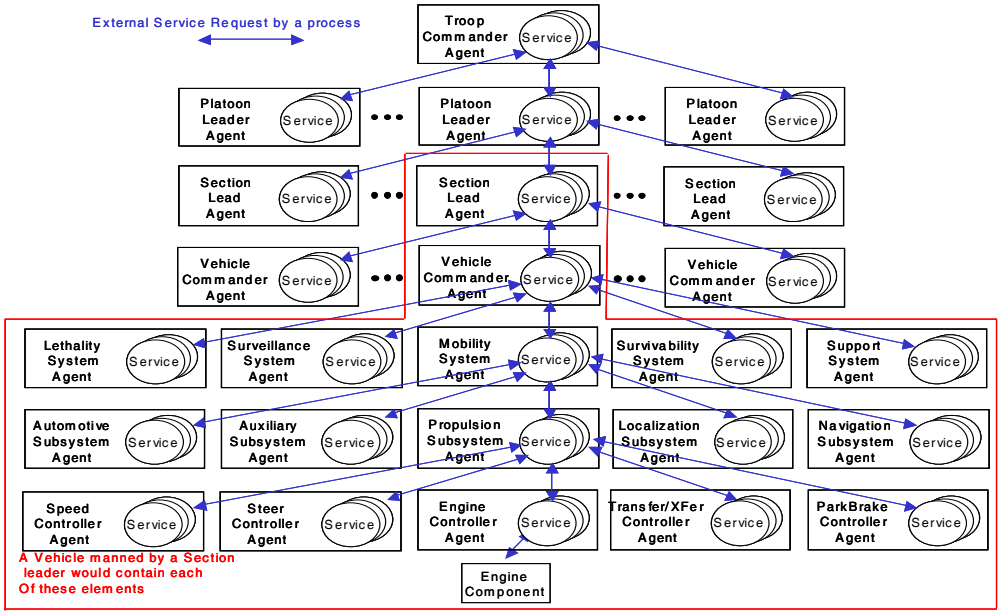


Figure 8: Agent Hierarchy.

The next step involved setting up the services and processes. Any activity that can be called by another agent is considered a service in OWL-S. Any activity that the agent performs internally that cannot be externally requested is called a process. As such, "Conduct A Tactical Road March to an Assembly Area" is modeled as a service that is provided by a Troop agent (and can be called by a Squadron agent). The Troop agent can call services provided by other agents. In this example, a service called "Conduct Route Reconnaissance" is defined and associated with the Scout Platoon agent.

Process models in OWL-S are used to capture the steps that must be accomplished to carry out the service, and the ordering constraints on those steps. Each step can be performed internally by the agent or could

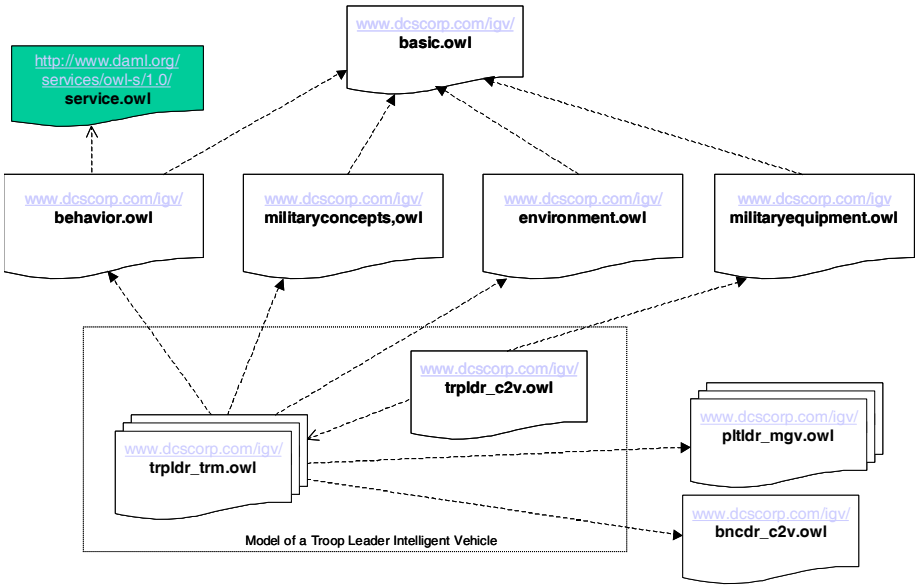


Figure 9: Namespaces.

involve making an external service request (a service call) to another agent. OWL-S provides a number of control constructs that allow one to model just about any type of process flow imaginable. Control constructs provided in OWL-S have been sufficient to model all the behaviors explored to date.

Environmental entities and their attributes are a primary output of the 4D/RCS methodology. These include other vehicles, bridges, vegetation, roads, water bodies; anything that is important to perceive in the environment relative to task that is being performed. An environment ontology in OWL-S has been built from the bottom up (i.e., including only entities that prove to be important based on the output of the RCS methodology). Environmental ontologies have started to be explored to see what could be leveraged.

4.3.2.4. Organizing the Knowledge

Due to the sheer size of the ontology, namespaces have been used to organize the knowledge in the ontology. A namespace is a tag prefixed to the name of the class or instance that separates the knowledge in the ontology into "pieces," where each piece represents a group of like concepts. Numerous namespaces can be imported into a single ontology and a single namespace can be reused in multiple ontologies. The contents of namespaces are often stored in separate files.

For this effort, five high-level namespaces have been identified that build off of the concepts presented in OWL-S (shown in Figure 9), namely:

- Basic – data structures to capture abstract, highly reusable concepts (e.g., location, spatial relations, time)
- Behavior – data structures which help to describe services, agents, and conditions (e.g., and-conditions, or-conditions, external service requests)
- Military Concepts – data structures to capture common concepts within military procedures (e.g., assembly area, control points, troops)
- Environment – data structures to capture environmental concepts (e.g, water bodies, shrubs, weather conditions)
- Military Equipment – data structures to capture information about the equipment that the military uses (e.g., communication devices, weapons, measuring devices)

In addition, separate namespaces have been defined for every agents' services and processes in every tactical behavior, as shown at the bottom of Figure 9. For example, in the "Conduct a Tactical Road March to an Assembly Area" tactical behavior, services and corresponding process models have been defined for most of the agents shown in Figure 8. As such, namespaces such as Troop ConductTacticalRoadMarchToAA, Platoon-ConductTacticalRoadMarchToAA, and so on have been defined.

4.3.2.5. Visualization of the Ontology

As mentioned earlier, one of the reasons Protégé was chosen was due to its ability to be extended using "plug-ins". A plug-in is a piece of code that performs a given functionality that can be incorporated into Protégé. A visualization tool was developed based upon the open source GraphViz program [17] that allows us to graph OWL-S models. A snapshot of the visualization tool, shown in Figure 10, shows process flow and data flow.

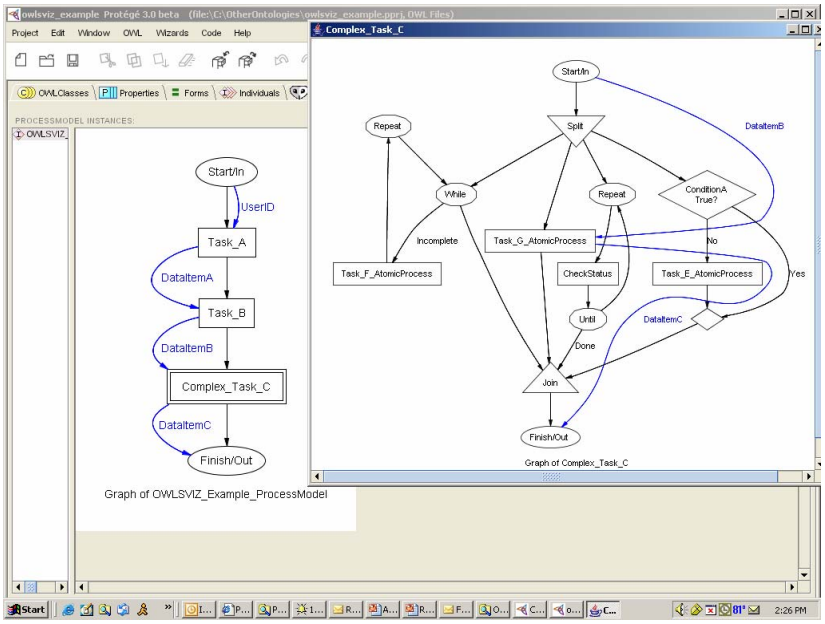


Figure 10: OWL-S Visualization Tool.

4.3.2.6. Status and Future Work

At this time this chapter was written, service models were developed for the following agents: troop, platoon, section, vehicles, mobility, propulsion, engine controller, engine, surveillance, and sensor subsystem (as shown in Figure 8). All of the service models were focused solely on the scenario described in Section 4.3.1. Only one strand of this scenario has been implemented (e.g., one service in each level of the hierarchy was elaborated, although there were almost always many services that each agent may have to perform to accomplish the overall goal of "Conducting a Tactical Road March to An Assembly Area"). Also, this one scenario represents one of hundreds, if not thousands, of tactical behaviors that an army soldier is expected to be able to perform. It is clear that, as the details of this scenario are further modeled and more scenarios are explored, the size of the ontology will grow to be very large. At the time this document was written, 489 classes, 213 properties (attributes), and 2674 instances were modeled.

Through the development of this ontology, it has become apparent that the applications of a neutral and well-defined representation and tactical behaviors are far-reaching above and beyond that of controlling autonomous vehicles. Some of the potential communities that have been identified include:

- Material Acquisition Community: Documenting behavioral requirements for manned, aided, and unmanned systems; and analyzing the completeness/consistency of requirements and predicting the performance of systems developed to meet those requirements.
- Vendor Community: Unambiguously interpreting behavioral requirements; and rapid development of simulation/prototypes to evaluate design alternatives.
- Training Community: Availability of machine-readable sources of tactical knowledge for automatic generation of training materials and training scenarios

5. World Modeling Processes

5.1. Background

The WM process performs four basic functions:

1) **Maintenance and updating of information** in the Knowledge Database (KD),

2) **Prediction** (The WM process generates short term predictions of expected sensory observations that enable SP processes to perform correlation and predictive filtering),
3) **Response to queries** for information required by other processes, and
4) **Simulation** (The WM process performs simulations in response to "What If?" queries in order to support the planning functions of the BG processes) .

5.2. World Modeling Processes in 4D/RCS

As mentioned in Section 1, world modeling processes construct, maintain, and use a world model knowledge database in support of behavior generation and sensory processing. Within 4D/RCS, there are three primary efforts that are developing world modeling processes, the first two of which focuses on world model maintenance and updating while the third focuses on making predictions. The first is exploring sensor registration, dealing with registration of 3D LADAR data for unmanned ground and aerial vehicles [28,29]. The second, which is recent and unpublished work, is exploring grouping of pixels into objects to allow one to determine if what a sensor system is currently seeing anything similar to what the system saw before (i.e., establishing correspondence). The third is the PRIDE (PRediction In Dynamic Environments) framework, which predicts the location of moving objects in the environment. This is the highlighted example discussed below.

5.3. World Modeling Processes Highlighted Example: The PRIDE Framework

Many believe that the DEMO III XUV effort represented the state of the art in autonomous off-road driving [48]. This effort sought to develop and demonstrate new and evolving autonomous vehicle technology, emphasizing perception, navigation, intelligent system architecture, and planning. It should be noted that the DEMO-III XUV was tested only in highly static environments. It was not tested in on-road driving situations, which include pedestrians and oncoming traffic.

There have been experiments performed with autonomous vehicles during on-road navigation. Perhaps the most successful was that of Prof. Dr. Ernst Dickmanns [13] as part of the European Prometheus project in which the autonomous vehicle performed a trip from Munich to Odense (>1600 km) at a maximum velocity of 180 km/h. Although the vehicle was able to identify and track other moving vehicles in the environment, it could only make basic predictions of where those vehicles were expected to be at points in the future, considering the vehicle's current velocity and acceleration.

What is missing from all of these experiments is a level of situation awareness of how other vehicles in the environment are expected to behave considering the situation in which they find themselves. To date, the authors are not aware of any autonomous vehicle efforts that account for this information when performing path planning. To address this need, a framework, called PRIDE (PRediction in Dynamic Environments) was developed that provides an autonomous vehicle's planning system with information that it needs to perform path planning in the presence of moving objects [43]. The following section describes the "high-level" cost-based probabilistic prediction algorithms in detail.

5.3.1. The PRIDE Framework

As discussed in Chapter 1, the 4D/RCS architecture supports multiple behavior generation (BG) systems working cooperatively to compute a final plan for the autonomous system. The spatial and temporal resolution of the individual BG systems along with the amount of time allowed for each BG system to compute a solution are specified by the level of the architecture where it resides. In addition to multiple BG systems, multiple world models are supported with each world model's content being tailored to the systems that it supports (in this case the BG system). As such, it is necessary for the future location of moving objects to be determined differently (at different scales and resolutions) at the different levels of the architecture.

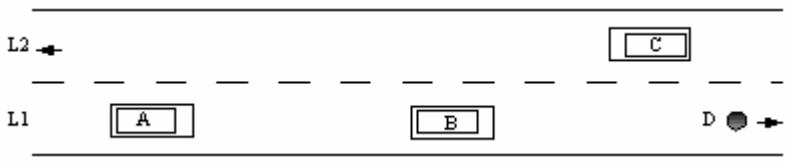


Fig. 11. The Situation.

To support this requirement, NIST has developed the PRIDE (PRediction In Dynamic Environments) framework. The underlying concept is based upon a multi-resolutional, hierarchical approach that incorporates multiple prediction algorithms into a single, unifying framework. This framework supports the prediction of the future location of moving objects at various levels of resolution, thus providing prediction information at the frequency and level of abstraction necessary for planners at different levels within the hierarchy. To date, two prediction approaches have been applied to this framework.

At the lower levels, estimation theoretic short-term predictions is used via an extended Kalman filter-based algorithm using sensor data to predict the future location of moving objects with an associated confidence measure [27]. At the higher levels of the framework, moving object prediction needs to occur at a much lower frequency and a greater level of inaccuracy is tolerable. At these levels, moving objects are identified as far as the sensors can detect, and a determination is made as to which objects should be classified as "objects of interest". In this context, an object of interest is an object that has a possibility of affecting the path in the planning time horizon. Once objects of interest are identified, a moving object prediction approach based on situation recognition and probabilistic prediction algorithms is used to predict where object will be at various time steps into the future. Situation recognition is performed using spatio-temporal reasoning and pattern matching with an *a priori* database of situations that are expected to be seen in the environment. This is discussed below.

The algorithms are used to predict the future location of moving objects in the environment at larger time planning horizons on the order of tens of seconds into the future with plan steps at about one second intervals. During the explanation of the algorithm, the following scenario will be used (Figure 11). This scenario is composed of three vehicles, two of which (A and B) are in lane L1 and moving to the right, and the third (C) is in lane L2 and moving to the left. In this scenario, D is a static object and is located in L1. Figure 12 graphically shows the overall process flow.

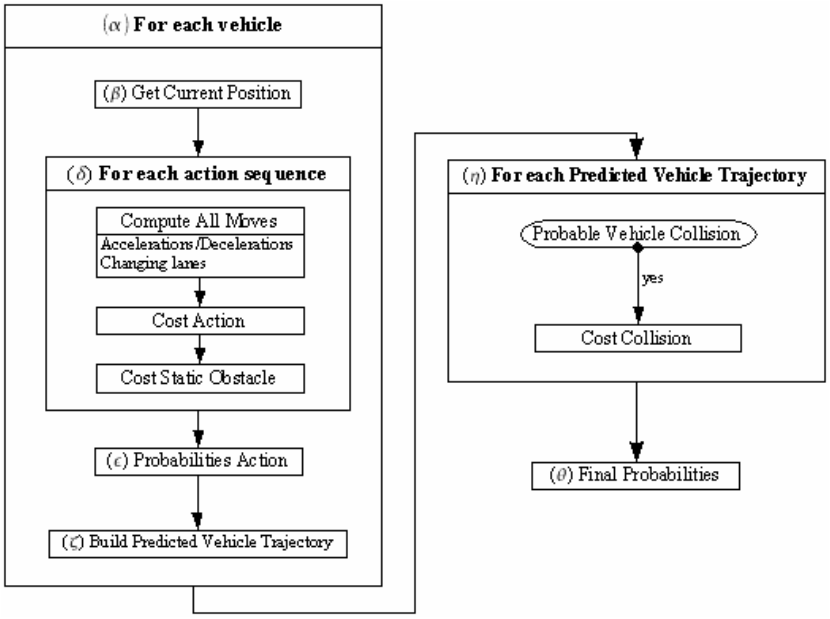


Fig. 12. The Moving Object Prediction Process.

The steps within the algorithm are:

1. For each vehicle on the road (α), the algorithm gets the current position and velocity of the vehicle by querying external programs/sensors (β).
2. For each set of possible future actions (δ), the algorithm creates a set of next possible positions and assigns an overall cost to each action based upon the cost incurred by performing the action and the cost incurred based upon the vehicle's proximity to static objects. An underlying cost model is developed to represent these costs.
3. Based upon the costs determined in Step 2, the algorithm computes the probability for each action the vehicle may perform (ε). At this step in the scenario, the possible actions/probabilities for the three vehicles are shown in Figures 13a-c:

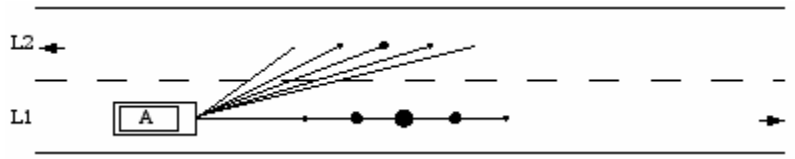


Fig. 13a: A-Vehicle Actions-Probabilities

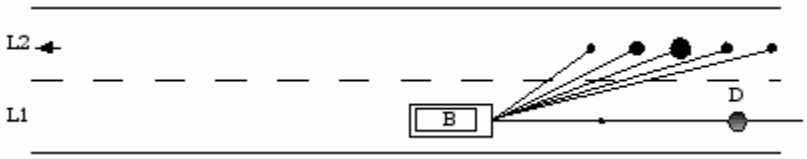


Fig. 13b. B-Vehicle Actions-Probabilities

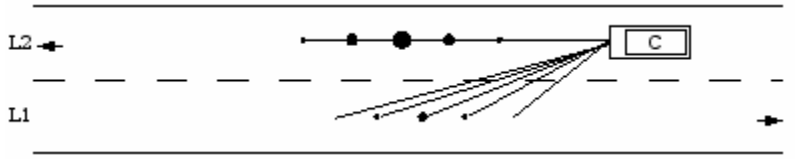


Fig. 13c. C-Vehicle Actions-Probabilities

where the size of each dot represents the relative probability with respect to the others.

4. Predicted Vehicle Trajectories (PVT) (ξ) are built for each vehicle which will be used to evaluate the possibility of collision with other vehicles in the environment. PVTs are a vector that indicates the possible paths that a vehicle will take within a predetermined number of time steps into the future.
5. For each pair of PVTs (η), the algorithm checks if a possible collision will occur (where PVTs intersect) and assigns a cost if collision is expected.

   In the scenario, for the vehicles A and C, Figure 14 shows two PVTs that cross, indicating that a collision is possible.

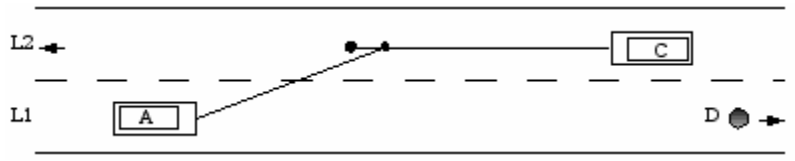


Fig. 14. Possible Collision between A and C

6. In this step, the probabilities of the individual actions (θ) are recalculated, incorporating the risk of collision with other moving objects, as shown in Figures 15a-c.

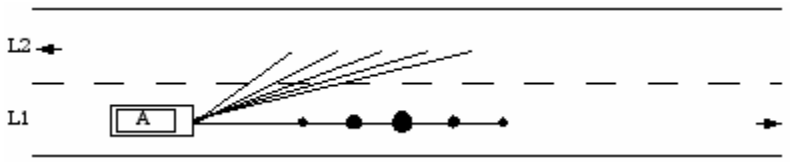


Fig. 15a. A-Vehicle Final Probability

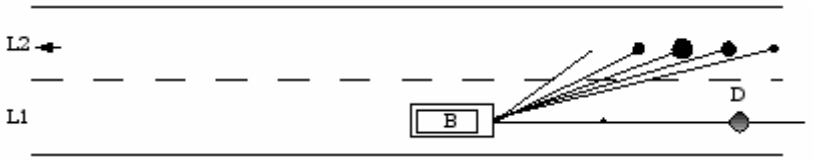


Fig. 15b. B-Vehicle Final Probability

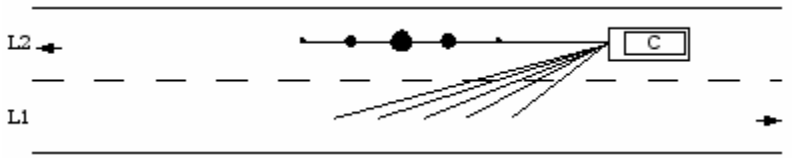


Fig. 15c. C-Vehicle Final Probability

At the end of the main loop, the future positions with the highest probabilities for each vehicle represent the most likely location of where the vehicles will be in the future. More information about the cost-based probabilistic prediction algorithms can be found in [43].

5.3.2. Experimental Results

The situation-based probabilistic prediction approach has been implemented in the AutoSim simulation package developed by Advanced Technology Research Corporation. AutoSim is a high-fidelity simulation tool which models details about road networks, including individual lanes, lane markings, intersections, legal intersection traversibility, etc. Using this package, typical traffic situations (e.g., multiple cars negotiating around obstacles in the roadway, bi-directional opposing traffic, etc.) have been simulated and predictions as to the future location of individual vehicles on the roadway have been made based upon the prediction of where other vehicles are expected to be (Figure 16).

At the point this chapter was written, numerous driving situations have been simulated using eleven costs to determine the probabilities of one action over another. In all driving situations, there were anywhere from one to three vehicles and obstacles placed at different random locations on the roadway. In all cases, there were no "road blocks", meaning that there was always at least one lane on the roadway that would allow a vehicle to pass. The initial velocity of the vehicles varied from being at rest to 30 m/s. Costs were incurred based on: 1) proximity to other objects in the environment as a function of the necessary stopping distance, 2) two costs associated with exceeding or going below the speed limit by a given threshold, 3) changing lanes, 4) not being in the rightmost lane, 5) five costs associated with acceleration profiles (constant velocity, slowly accelerating and decelerating, rapidly accelerating or decelerating), and 6) changing lanes where double yellow lines in the road exist. Using these costs, predictions up to ten seconds into the future at a rate of five predictions per second were achieved.

In the experiments, it was interesting to see how the vehicles performed when the "action sequences" were varied with respect to time. The action sequence is the amount of time that it would take the vehicle to perform a driving maneuver (e.g., changing lanes). For most of these experiments, a passing maneuver was used as the action sequence to be evaluated. When this time was set to a longer duration (e.g., 7 s), the vehicle was very slow to respond to unexpected events (e.g., a vehicle pulling into its lane quickly), which cause, in some cases, near collisions. Conversely, when the time for the action sequence was set to a shorter time (e.g., 2 s), the vehicle became jerky and could not make up its mind. For example, it would start a lane change and then quickly return to the original lane. Four seconds appears to be a good middle-ground to create realistic traffic patterns. The existence of these realistic traffic patterns is important to be able to truly assess the performance of autonomous vehicles in on-road driving scenarios.

Future work will create different action sequence timeframes for different types of driving maneuvers.
The introduction of action sequences into the prediction algorithms have resulted in dramatic increases in performance with respect to time. Before the concept of action sequences was introduced, predictions up to 5 s into the future for two vehicles at a rate of two predictions per second were possible. Once actions sequences were introduced (along with the rules which state which action sequences are valid and invalid), predictions up to 10 s into the future with two vehicles at a rate of 10 predictions per second were possible.

The purpose of these algorithms is to work in real time. The experimental results show that after seven seconds in the future, the algorithms become jerky and lose their real time functionality on a Pentium 4 CPU machine with a 1.8 GHz CPU and 512 MB of memory.

In addition, recent efforts have demonstrated the ability to use the results of the short-term prediction algorithms to strengthen/weaken the estimates of the long-term prediction algorithms. Based upon previous experiments, short-term prediction algorithms perform best when predicting on the order of a few seconds into the future and that the longer-term prediction algorithms are best at predicting on the order of several seconds into the future. Thus, there are opportunities for leveraging the predictions when both prediction algorithms provide valuable estimates during these times.

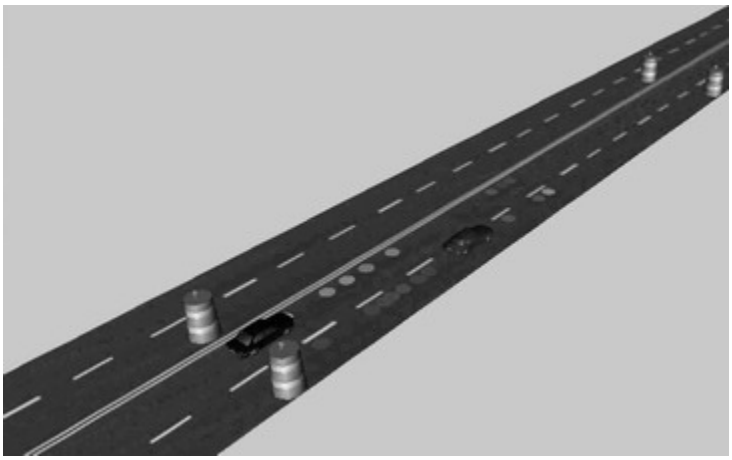


Fig. 16. Two Vehicles passing obstacles.

6. Discussion and Conclusion

The principles of how to choose one representation over another is never an easy problem and this chapter does not claim to provide all of the answers. However, using 4D/RCS helps to make this decision a little easier. As mentioned throughout the chapter, 4D/RCS is based on multi-representational approaches. There may be redundancy in the data/information due to the multiple representations. The intent is to have the right representation available for the various calculations and decision making processes. No one type of knowledge representation is adequate for all purposes. Davis [12] argues that representation and reasoning

at the symbolic level are inextricably intertwined, and that different reasoning mechanisms, such as rules and frames, have different natural representations that must be integrated in a representation architecture to achieve the advantages of multiple approaches to reasoning.

Redundancy in information usually involves extra effort in maintaining the integrity of the knowledge base, but 4D/RCS helps to address this through the tight coupling of different forms of information into higher level knowledge. For example, points cluster into lines which cluster into surfaces which cluster into objects. This tight coupling helps to ensure the integrity of the knowledge base, even when the same type of information is represented in different forms.

Often, multiple types of representation are used when performing a single behavior. Geometric knowledge may be used to probabilistically recognize a vehicle on the road while symbolic information might strengthen or weaken the probabilities based on known physical and behavioral characteristics of a vehicle. Topological and symbolic information may also be used to track the vehicle, based on the knowledge that vehicles tend to follow roadways, stop at some intersections, etc. The introduction of spatial data – often grid-based – integrated with symbolic data and parametric data in a multi-resolution hierarchical world model, enables the real time control of complex systems interacting with the real world, including the ability to deal with dynamic relationships of objects in space and time. This provides the ability for a moving vehicle to sense and correctly respond to unexpected obstacles and events. This is the essence of intelligent control.

Another important consideration of knowledge representation is complexity versus real utility. The more complex the representation, the more complete it (usually) is, but the longer it may take to access the knowledge within it. 4D/RCS addresses this challenge through its hierarchical nature and through its agent architecture. Agents are represented at a defined level with 4D/RCS and have a bounded set of knowledge and responsibility. The level of abstraction of knowledge that is captured and available to the agent is a function of the level in the hierarchy in which it resides. This bounded set of knowledge and responsibility ensures that the agent is not over inundated with information, but also ensures that enough information is necessary for it to accomplish its tasks. However, this does not imply that the complexity of the system stays consistent independent of the number of hierarchical levels and agents are added. Maximov and Meystel [31] show that when the number of hierarchical levels grow, complexity initially falls, and then starts growing again.

A third consideration is ensuring the representational choices allow for well-performing situation assessment. Similar to the previous consideration, 4D/RCS facilitates this through the use of multiple representations, information redundancy among the representations, and multiple levels of abstraction to ensure the scope/scale of the information aligns with the level of situation awareness that is desired. Another important aspect that allows for accurate situation assessment is the constant updating of knowledge. Sensor information is constantly fed into the world model to ensure it is as up-to-date as possible. This sensor knowledge, through world model processes, is constantly being analyzed and abstracted to achieve a higher level of situation assessment within the world model. These world model processes ensure that the most up-to-date information is always available to the behavior generation engines so the best possible decisions can be made.

4D/RCS architecture provides an excellent framework in which to integrate multiple knowledge representation approaches to build cognitive models and intelligent systems that significantly advance the level of intelligence that can be achieved. This chapter described how 4D/RCS supports multiple types of representations, ranging from iconic to symbolic and from declarative to procedural, and provided brief examples of how each of these representations are used in the context of autonomous driving. Also shown was how all of these knowledge representation formalisms fit into the node structure present at each level of the 4D/RCS hierarchy.

The efforts described in this chapter only begin to touch on all of the knowledge that is necessary to enable truly intelligent control, as compared to the knowledge that a human possesses when making decision. Much more work needs to go into the area of situational awareness, which focuses on extracting and reasoning about pertinent information about the environment to get a higher level picture of what is

actually occurring and what can be inferred from it. This is a challenging problem, and one that will be a focus of this research effort in the future.

References

1. Albus, J., "A Theory of Cerebellar Function," *Mathematical Biosciences*, Vol. 10, 1971, pp. 25-61.
2. Albus, J. S., *Brain, Behavior, and Robotics*, McGraw-Hill 1981.
3. Anderson, J., *The Architecture of Cognition*, Lawrence Erlbaum Associates, Mahwah, N.J., 1983.
4. Archer, R., Lebriere, C., Warwick, W., and Schunk, D., "Integration of Task Network and Cognition Models to Support System Design," *Proceedings of the Collaborative Technology Alliances Conference: 2003 Advanced Decision Architectures*, College Park, MD, 2003.
5. Archer, S. and Adkins, R., "IMPRINT User's Guide," 1999.
6. Balakirsky, S., *A Framework for Planning with Incrementally Created Graphs in Attributed Problem Spaces*, IOS Press, Berlin, 2003.
7. Balakirsky, S. and Lacaze, A., "World Modeling and Behavior Generation for Autonomous Ground vehicle," *Proceedings of the 2000 IEEE International Conference on Robotics & Automation*, 2000, pp. 1201-1206.
8. Barbera, T., Albus, J., Messina, E., Schlenoff, C., and Horst, J., "How Task Analysis Can Be Used to Derive and Organize the Knowledge For the Control of Autonomous Vehicles," *Robotics and Autonomous Systems Journal: Special Issue on the 2004 AAAI Knowledge Representation and Ontologies for Autonomous Systems Spring Symposium*, Vol. 49, No. 1-2, 2004, pp. 67-78.
9. Behnke, S., "Local Multi-Resolution Path Planning," *RoboCup-2003: Robot Soccer World Cup VII*, edited by B. Browning, D. Polani, A. Bonarini, and K. Yoshida Springer, 2004.
10. Borenstein, J. and Koren, Y., "Real Time Obstacle Avoidance for Fast Mobile Robots in Cluttered Environments," *Proceedings, 1990 IEEE ICRA*, Cincinnati, OH, 1990.
11. Cassimatis, N., Trafton, M., Bugajska, M., and Schultz, A., "Integrating Cognition, Perception And Action through Mental Simulation in Robots," *Robotics and Autonomous Systems*, Vol. 49, No. 1-2, 2004, pp. 13-23.
12. Davis, R., "What is in a Knowledge Representation?," *AI Magazine*, 1993.
13. Dickmanns, E. D., "A General Dynamic Vision Architecture for UGV and UAV," *Journal of Applied Intelligence*, Vol. 2, 1992, pp. 251.
14. Dijkstra, E. W., "A note on two problems in connection with graphs," *Numerische Mathematik*, Vol. 1, 1959, pp. 269-271.
15. Dunlaing, C. O. and Yap, C. K., "A "retraction" method for planning the motion of a disc," *Journal of Algorithms*, Vol. 6, 1986, pp. 104-111.
16. Etherington, D., "What Does Knowledge Representation Have to Say to Artificial Intelligence?," *Proceesings ate the AAAI*, 1997.
17. Gansner, E. and North, S., "An Open Graph Visualization System and Its Applications," *Software - Practice and Experience*, Vol. 00, No. S1, 1999, pp. 1-5.
18. Gonzalez, C., "ACT-R Implementation of an Instance-Based Decision Making Theory," *Proceedings of the Collaborative Technology Alliance Conference: 2003 Advanced Decision Architectures*, College Park, MD, 2003.
19. Harmelen, F. and McGuiness, D., "OWL Web Ontology Language Overview," *W3C web site: http://www.w3.org/TR/2004/REC-owl-features-20040210/*, 2004.
20. Harnad, S., "The Symbol Grounding Problem," *Physica*, Vol. 42, 1990, pp. 335-346.
21. Hong, T., Balakirsky, S., Messina, E., Chang, T., and Shneier, M., "A Hierarchical World Model for an Autonomous Scout Vehicle," *Proceedings of SPIE's 16th Annual International Symposium on Aerospace/Defense Sensing, Simulation, and Controls*, 2002, pp. 343-354.
22. Kieras, D. and Meyer, D. E., "An overview of the EPIC architecture for cognition and performance with application to human-computer interaction," *Human-Computer Interaction*, Vol. 12, 1997, pp. 391-438.
23. Kobilarov, M. and Sukhatme, G., "Time optimal path planning on outdoor terrains for mobile robots under dynamic constraints," *Technical Report CRES-04-009 USC*, 2004.
24. Laird, J. E., Newell, A., and Rosenbloom, P. S., "SOAR: An Architecture for General Intelligence," *Artificial Intelligence*, Vol. 33, 1987, pp. 1-64.

25. LaValle, S. M. and Kuffner, J. J., "Randomized kinodynamic planning," *1999 IEEE International Conference on Robotics and Automation, Volume 1*, 1999, pp. 473-479.
26. Liu, Y., Emery, R., Chakrabarti, D., Burgard, W., and Thrun, S., "Using EM to learn 3D models with mobile robots," *Proceedings of the International Conference on Machine Learning (ICML)*, 2001.
27. Madhavan R. and Schlenoff, C., "The Effect of Process Models on Short-term Prediction of Moving Objects for Autonomous Driving," *International Journal of Control, Automation and Systems*, Vol. 3, 2005, pp. 509-523.
28. Madhavan, R. and Messina, E., "Performance Evaluation of Temporal Range Registration for Unmanned Vehicle Navigation," *Proceedings of the 2004 Performance Metrics for Intelligent Systems (PerMIS) Workshop*, Gaithersburg, MD, 2004.
29. Madhavan, R. and Messina, E., "Iterative Registration of 3D LADAR Data for Autonomous Navigation," *Proceedings of the IEEE Intelligent Vehicles Symposium*, Columbus, OH, USA, 2003, pp. 186-191.
30. Martin, M. C. and Moravec, H., "Robot Evidence Grids," *Tech Report CMU-RI-TR-96-06*, Robotics Institute, Carnegie Mellon University, 1996.
31. Maximov, Y. and Meystel, A., "Optimum Architectures for Multiresolutional Control," *Proceedings of the IEEE Conference on Aerospace Systems*, Westlake Village, CA, 2003.
32. Minsky, M., "A Framework for Representing Knowledge," *MIT-AI Laboratory Memo 306*, 1974.
33. Minton, S. N., "Quantitative results concerning the utility of explanation-based learning," *Artificial Intelligence*, Vol. 42, 1990, pp. 363-391.
34. Moravec, H. and Martin, M. C., "Robot navigation by 3D spatial evidence grids," *CMU Mobile Robot Laboratory Internal Report*, 1994.
35. Moravec, H. P. and Elfes, A., "High Resolution Maps from Wide Angle Sonar," *Proceedings of the 1985 IEEE ICRA*, St. Louis, MO, 1985.
36. Newell, A. and Simon, H., *Human Problem Solving*, Prentice-Hill, Englewood Cliffs, 1972.
37. Newell, A. and Simon, H., *GPS; A Program that Simulates Human Thought*, McGraw-Hill 1963.
38. Nilsson, N. J., *Artificial Intelligence: A New Synthesis*, Morgan Kaufmann Publishers, Inc., San Francisco, 1998.
39. Nilsson, N. J., "A mobile automaton: An application of artificial intelligence techniques.," *Proceedings of the 1st International Joint Conference on Artificial Intelligence*, Washington DC, 1969, pp. 509-520.
40. Oskard, D., Hong, T., and Shaffer, C., "Real-Time Algorithms and Data Structures for Underwater Mapping," *Proceedings of the SPIE Advances in Intelligent Robotics Systems Conference*, Boston, MA, 1988.
41. Pearson, J. D., Huffman, S. B., Willis, M. B., Laird, J. E., and Jones, R. M., "A Symbolic Solution to Intelligent Real-Time Control," *Robotics and Autonomous Systems*, Vol. 11, 1993, pp. 279-291.
42. Provine, R., Uschold, M., Smith, S., Balakirsky, S., and Schlenoff, C., "Ontology-based Methods for Enhancing Autonomous Vehicle Path Planning," *Robotics and Autonomous Systems Journal: Special Issue on the 2004 AAAI Knowledge Representation and Ontologies for Autonomous Systems Spring Symposium*, Vol. 49, No. 1-2, 2004, pp. 123-133.
43. Schlenoff, C., Ajot, J., and Madhavan, R., "PRIDE: A Framework for Performance Evaluation of Intelligent Vehicles in Dynamic, On-Road Environments," *Proceedings of the Performance Metrics for Intelligent Systems (PerMIS) 2004 Workshop*, 2004.
44. Schlenoff, C., Balakirsky, S., Uschold, M., Provine, R., and Smith, S., "Using Ontologies to Aid in Navigation Planning in Autonomous Vehicles," *Knowledge Engineering Review*, Vol. 18, No. 3, 2004, pp. 243-255.
45. Schlenoff, C., Washington, R., and Barbera, T., "Experiences in Developing an Intelligent Ground Vehicle (IGV) Ontology in Protege," *Proceedings of the 7th International Protege Conference*, Bethesda, MD, 2004.
46. Schwartz, J. T. and Sharir, M., "A Survey of Motion Planning and Related Geometric Algorithms," *Artificial Intelligence*, Vol. 37, 1988, pp. 157-169.
47. Shapiro, D. and Langley, P., "Controlling physical agents through reactive logic programming," *Proceedings of the Third International Conference on Autonomous Agents 386-387*, ACM Press, Seattle, 1999.

48. Shoemaker, C. and Bornstein, J. A., "Overview of the Demo III UGV Program," *Proceedings of the SPIE Robotic and Semi-Robotic Ground Vehicle Technology Conference*, Vol. 3366, 1998, pp. 202-211.
49. Stentz, A., "Optimal and Efficient Path Planning for Partially-Known Environments," *Proceedings of the IEEE International Conference on Robotics and Automation*, 1994, pp. 3310-3317.
50. The OWL Services Coalition, "OWL-S 1.0 Release," http://www.daml.org/services/owl-s/1.0/owl-s.pdf, 2003.
51. Yahja, A., Stentz, A., Singh, S., and Brumitt, B. L., "Framed-Quadtree Path Planning for Mobile Robots Operating in Sparse Environments," *Proceedings 1998 IEEE International Conference on Robotics and Automation*, Vol. 1, 1998, pp. 650-655.

Chapter 5

Sensory Processing

Michael Shneier, Tsai Hong, Tommy Chang, and **Mike Foedisch**
National Institute of Standards and Technology (NIST)
{michael.shneier,tsai.hong,tommy.chang,mike.foedisch}@nist.gov

5.1. Introduction

Sensory processing (SP) provides information about the world that enables a robot to interact with its environment and react to events or changes that influence its task. The goal of SP is to build and maintain the internal representation of the world (World Model) that the behavior generation components of the robot can use to plan and execute actions that modify the world or the robot's position in the world. Since the world in general is not static, the SP algorithms must update the internal model rapidly enough to allow changes, such as the positions of moving objects, to be represented accurately. This places constraints on the sensors that can be used and on the processing that can be applied.

This chapter will address the sensory processing aspects of the 4D/RCS architecture as applied to autonomous mobile robots. It will cover the use of multiple sensors to build up a description of the environment around the vehicle and discuss the algorithms needed to interpret the sensory data. The chapter is organized as follows.

Section 5.2 covers sensors, sensor registration, and sensor fusion. Sensors in use include monocular and stereo color cameras and scanning and imaging LADARs. These sensors have to be registered so that data from the sensors can be combined. Registration involves accurately locating the sensors relative to each other and time- and position-stamping the outputs of each sensor so that the motion of the vehicle can be taken into account. Registration can also be carried out using a single sensor, whose output at different times can be combined if the positions of the sensor at successive times are known. Sensor fusion allows a richer description of the world and is particularly useful in associating range information with data from a monocular color camera, which would otherwise be difficult to locate in space because of its two-dimensional nature.

Obstacle detection is described in Section 5.3 and an algorithm is described for finding obstacles in range data. Obstacle features are described separately because obstacles are geometric entities detected using range information rather than color- or feature-based entities extracted from conventional imagery. Obstacles are features that lie above the ground or correspond to holes in the ground. They need to be detected well before the vehicle reaches them. It is necessary to know the true size of an obstacle so that it can be avoided; hence a three-dimensional analysis is required. This can be obtained from LADAR or stereo sensors, while a color camera may provide the best data to identify the obstacle.

Section 5.4 comprises the bulk of the chapter and describes a range of feature detection methods for specific features of interest. To build a rich description of the world, a variety of features need to be recognized. These include roads, road signs, water, other vehicles, pedestrians, etc. Special purpose processing is needed for each of these features, which must not only be detected, but tracked while they are within the immediate vicinity of the robotic vehicle.

Section 5.5 discusses learning and provides an algorithm for learning the traversability of terrain based on its appearance in color images.

Performance evaluation of SP algorithms is the subject of Section 5.6. Sensory processing plays a critical part in keeping the vehicle operating safely. Evaluating the performance of the SP algorithms provides a way to ensure that they work correctly and robustly enough in the real world. Performance evaluation involves testing the algorithms in realistic scenarios and quantitatively measuring how well they meet their specifications. Finally, the chapter ends with a brief conclusion.

World modeling and knowledge representation are presented in more depth in Chapter 4, but a basic overview is provided here as a basis for understanding the sensory processing, feature extraction, and tracking processes. The world model contains a representation of the current state of the world surrounding the vehicle and is updated continually by the sensors. A modified occupancy grid representation is used, with the vehicle centered on the grid, and the grid tied to the world. The world model thus scrolls under the vehicle as the vehicle moves about in the world. The world model is the system's internal representation of the external world. It acts as a bridge between SP and behavior generation by providing a central repository for storing sensory data in a unified representation and decouples the real-time sensory updates from the rest of the system. The world model process has two primary functions:

1. **To create a knowledge database (map) and keep it current and consistent.** In this role, it updates existing data in accordance with inputs from the sensors, and deletes information no longer believed to be representative of the world. It also assigns (multiple) confidence factors to all map data and adjusts these factors as new data are sensed. The types of information included in the map are state variables (e.g., time, position, orientation), system parameters (e.g., coordinate transforms, sensor to vehicle offsets, etc.), and lists or classes of sensed objects. The world model process also provides functions to update and fuse data and to manage the map (e.g. scrolling and grouping objects.)

2. **To generate predictions of expected sensory input** based on the current state of the world and estimated future states of the world. For the Demo III off-road autonomous driving application described in Chapter 10, very little *a priori* information is available to support path planning between the vehicle's position and a final goal position. The world model therefore constructs and maintains all the information necessary for intelligent path planning.

5.2. Sensors, Sensor Registration, and Sensor Fusion

Ideally, sensors would provide omni-directional views updated in real time with registered range and color information. In practice, this is not feasible, and a variety of compromises are made. While it is possible to make use only of cameras for both range (stereo) and color information, typical systems make use of LADAR sensors as well as cameras because LADARs can directly measure range. The fields of view of the sensors are usually limited, but sensors can often be actively pointed at areas of interest.

LADAR sensors include line-scan units, such as the SICK, which puts out a single plane of laser light spanning 100°-180° that can be mechanically scanned over a scene to build a range map. Range can be found to all points that intersect with the line based on time of flight of the light pulses. As described in Chapter 7, newer LADAR sensors directly image a scene without scanning, giving range data as well as color in a single instant [1]. Range resolution varies from a few millimeters to several centimeters, and measurable range varies from less than 10 m to more than 800 m, depending on the sensor used.

Modern cameras for computer vision applications can deliver 1024x768 images at 30 Hz (with Bayer filter-based color). This provides ample resolution for object recognition or for stereo-based range computation. With multiple sensors mounted on a robot, the issue arises of how to relate the information from each one to the others. This requires sensor registration, in which the relative positions and fields of view of the sensors are calibrated. The position and orientation of each sensor is measured. The sensors are represented in a common coordinate system (usually that of the robot). The fields of view can then be computed and overlaps used for sensor fusion. Registration of 3D LADAR both from UGVs and UAVs is discussed in

Chapter 6. This chapter describes the registration of a camera with a LADAR system and gives examples of how data fused from multiple sensors can be helpful in feature detection.

5.3. Obstacle Detection

A mobile robot needs to locate and characterize obstacles that prevent it from reaching its goal. Obstacles are of two kinds, those that stick up out of the ground (positive obstacles), and those that lie below the ground (negative obstacles). The robot needs to know where the obstacles are located and how big they are so that it can plan a path around them.

Finding obstacles in three-dimensional point clouds resulting from LADAR or stereo processing involves first determining the ground plane. In man-made environments or where the ground is reasonably flat, a surface can be fitted to points believed to be on the ground (e.g., points close to the vehicle, which is assumed to be on the ground). This surface can then be used as a reference against which points can be measured. Those that project above or below the surface by a large enough distance are considered to be obstacles. Where the ground is not flat, an alternative algorithm must be used: One such algorithm is described below for detecting obstacles in range images where the ground is not constrained to lie in a plane.

Obstacles are defined as objects that project more than some distance *d* above or below the ground. Positive obstacles are detected in the range images, while negative obstacles are detected in the world model map.

The positive obstacle detection algorithm works column by column in the range image [2]. The algorithm starts with a point, *g*, known to be on the ground. An initial ground value is assigned at the location where the front wheels of the vehicle touch the ground, known from INS and GPS sensors. Given point *g*, the algorithm processes upwards from the bottom pixel in the column to the top pixel, as follows:

1. Let p_i be the i^{th} pixel in the column, where pixel *0* is at the bottom of column. Let x_i, y_i, z_i be the Cartesian coordinates of p_i. Let *g* be the last known ground pixel in the column, initially obtained from the vehicle's position sensors. Compute the slope between the ground point, *g*, and the next pixel p_k. Pixel p_k is labeled a positive obstacle if

$$\frac{\left(z_k - z_g\right)^2}{\left(x_k - x_g\right)^2 + \left(y_k - y_g\right)^2 + \left(z_k - z_g\right)^2} \geq \sin^2(\alpha)$$

   where α is a predefined constant representing the maximum allowed slope. The value of $\sin^2(\alpha)$ is constant, and is pre-computed for efficiency.

2. Pixel p_k may fail the above test but still be a positive obstacle. This is because the slope test is a function of distance. The obstacle can be far from the current ground point due either to occlusion or to the resolution of the sensor which degrades as a function of distance. To resolve this ambiguity, the height of the obstacle is required to be greater than a constant, *H*. i.e., $\left|z_k - z_g\right| < H$.

3. If p_k is not an obstacle, it is assumed to be ground and replaces *g* as the current ground pixel. The process iterates up the column with each pixel being compared to the closest ground pixel.

4. If p_k is an obstacle, *g* is unchanged, and is compared with pixels $p_k, p_{k+1}, \ldots$ as above, until another ground pixel is found. When this occurs, point *g* is set to the new ground pixel value and the process continues from Step 1.

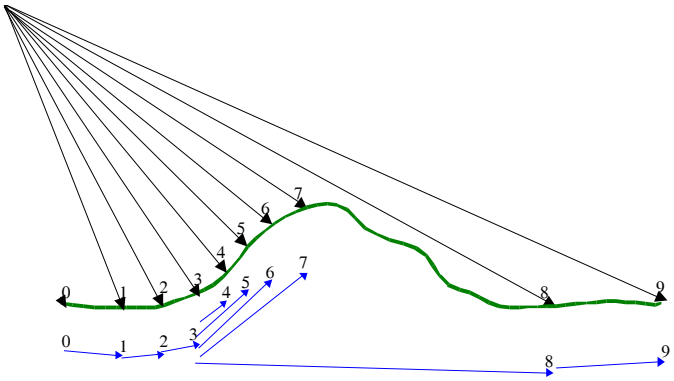


Figure 1. Positive obstacle detection.

In Figure 1, pixel 0 corresponds to the bottom of the vehicle wheel. Pixels 1, 2, 3, 8 and 9 are ground pixels. Pixel 4, 5, 6 and 7 are positive obstacles because they either satisfy step 1 or step2 or both. The direction vectors shown on the bottom of Figure 1 indicates the vectors in which the slopes are determined. In a way, the algorithm is analogous to flooding; pixels 1, 2, 3, 8 and 9 are flooded because they have shallow slopes.

The results of the positive obstacle detection are shown in Figure 2. The figure on the left is a LADAR scene of a wall obstructed by a truck on the far right. The objects in the foreground are low poles. The figure on the right shows in red the objects detected as positive obstacles in this scene.

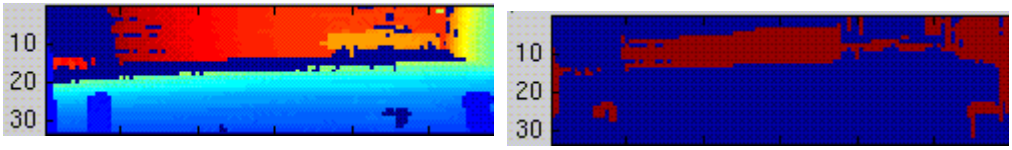


Figure 2. Raw LADAR image and detected positive obstacles.

The negative obstacle detection algorithm maintains its own high-resolution ground map centered on the vehicle. This ground map contains all the projected ground pixels detected by the positive obstacle detection module. The algorithm first identifies the pixels in the range image that potentially correspond to a negative obstacle (see algorithm details below). Based on the accumulated ground information in the ground map, the algorithm determines more precisely the dimension of the negative obstacle. Thresholds for depth and width are used to reject negative obstacles that are too small in dimension. For efficiency, the algorithm detects only the borders of negative obstacles. Steps 1 through 4 describe the algorithm in detail (Figure 3).

1. Let p_k be a ground point, and let *w* and *d* be the approximate width and depth of the negative obstacle. That is, $w = x_k - x_g$ and $d = z_g - z_k$. Let $proj_{p_k}$ be the map location corresponding to the projection of p_k onto the ground map. Let $d_{\min}$ be the minimum depth for an obstacle, and $w_{\min}$ the minimum width smaller than the vehicle wheel diameter.
2. If both $d < d_{\min}$ and $w < w_{\min}$ are true, then $proj_{p_k}$ is marked as ground.
3. If $proj_{p_k}$ is within a neighborhood corresponding to the area along the line of sight from the closest ground point on the map, then p_k is not labeled as a negative obstacle, but $proj_{p_k}$ is marked as a ground

cell.

4. When both $w \geq w_{\min}$ and $d \geq d_{\min}$ are true and no ground map cell exists within the neighborhood, p_k is labeled a negative obstacle, but $proj_{p_k}$ is not marked as a ground cell.

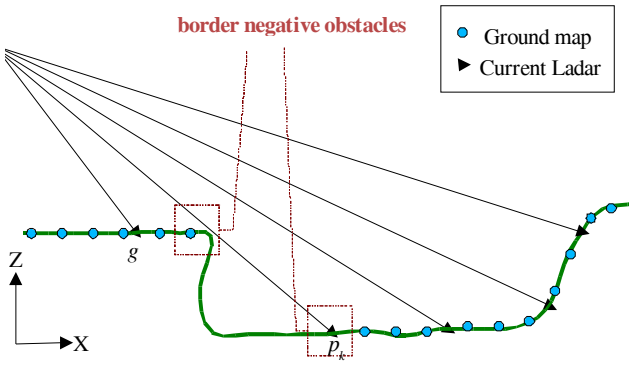


Figure 3. Negative obstacle border detection.

Figure 3 graphically describes this process. Circles represent points in the ground map. The triangular points represent current LADAR hits. The points enclosed in squares fulfill the requirements described in Step 4 above and are labeled as negative obstacle borders.

5.4. Feature Detection

Feature detection involves searching the image for regions that match a given criterion. Different features usually require separate searches using different algorithms and perhaps different sensors. How roads, road signs, curbs, and water features are detected is described briefly.

5.4.1. Road Detection

Neural Networks (NNs) are used for road detection by learning to differentiate the color distribution of road areas from other areas in the image. The *basic approach* consists of two steps: the NN training step and the road detection step.

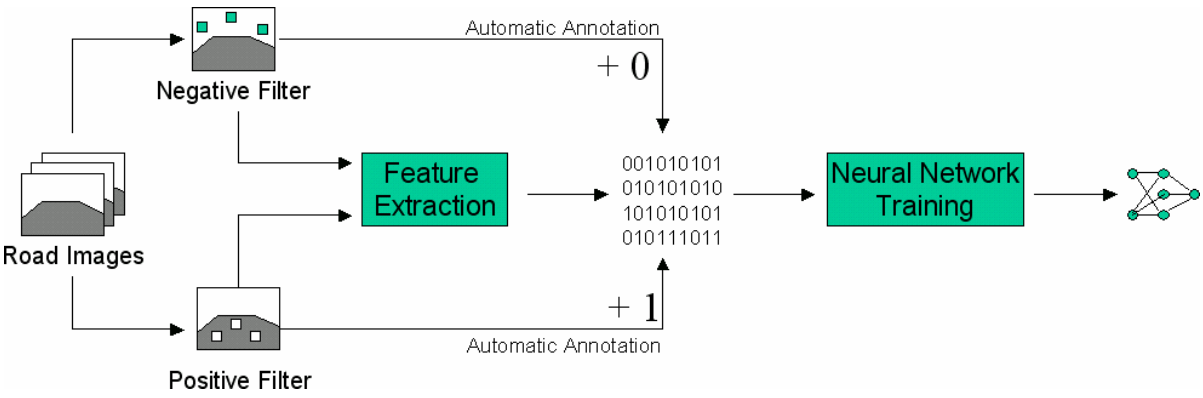


Figure 4. Overview of the NN training step.

Figure 4 gives an overview of the NN training step. First, feature data are extracted from areas of the image defined by filters (windows assumed to be entirely on the road or entirely off it). Then, a NN is trained on the feature data using the filter type as classification label. As introduced in [3], an "independent" color histogram consisting of 8 bins per channel is used [4]. This color histogram is computed for a 7x7-pixel window around each measuring point in the image. Measuring points form a sparse raster, for example by selecting every third point in the image (see the distribution of sample points in Figure 6). Additionally, the normalized *x* and *y* position values of the current point of consideration are included in the set of features, which results in a feature vector of 26 values.

The camera used as the sensor for detecting roads is positioned at the driver's viewpoint. The algorithm takes advantage of the fact that the road usually covers a trapezoidal area, which is centered in the lower part of the image.

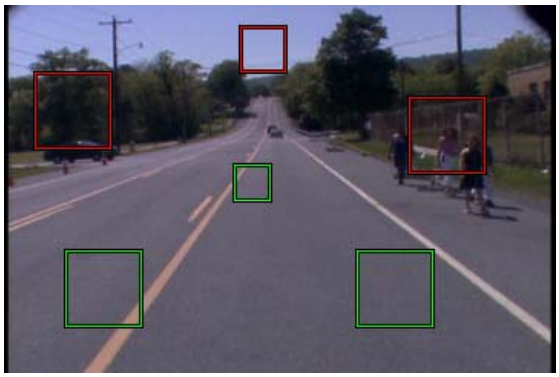


Figure 5. Example of three windows covering road area and another three covering non-road area.

Based on the estimated road location in the image, feature vectors are collected from pre-defined windows, which cover either road (road windows) or non-road areas (non-road windows). Feature vectors extracted from the windows are automatically labeled as either road or non-road depending on the type of the window. The example in Figure 5 shows three windows placed in the road area of the image and three windows in non-road areas.

The implementation uses a C++ based Neural Network library [5]. The NN receives 26 inputs (24 RGB histogram bins plus *x* and *y* coordinates) and consists of three layers. The first two layers contain four neurons each. The last layer is composed of one neuron, which generates the output. The NN employs back propagation learning [6].

For the road detection step the input image is overlaid on a raster of measuring points for which feature vectors are extracted. Each feature vector is processed by the NN, which was trained in the previous step. The resulting value is interpreted as either the road class or non-road class.

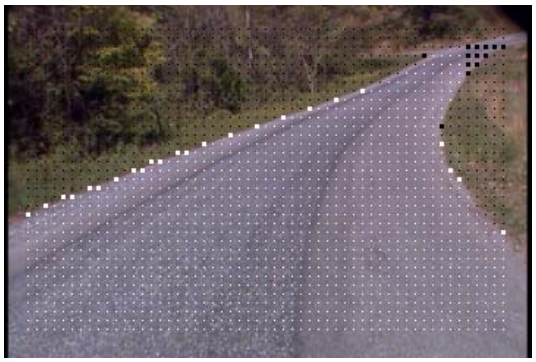


Figure 6. Sample result.

Figure 6 depicts a sample result where the area classified as road is drawn with white dots and the area classified as non-road is shown with black dots. The road detection accuracy has been enhanced by applying two post processing steps. First, noise is reduced by erosion and dilation methods. Second, only the largest detected region as the road area in the image is selected.

Initially, the road detection system relied on a NN trained only at the beginning of the course. Although the results were reasonable immediately after training the network, it was difficult to classify accurately using the same network when the environment and the road changed (e.g. due to changes in lighting) [7]. In order to solve this problem, continuous updates of the NN were added. For this purpose, new feature data are constantly collected from the feature extraction windows [8]. The re-training of the network takes about 600 ms. New feature data are collected for about one second, which enables the NN to be replaced roughly every two seconds.

Two enhancements to the *basic approach* have been developed. They integrate continuous learning into the road detection system. The approaches differ in how the positioning of feature extraction windows is handled. The first approach, called *fixed windowing*, follows the *basic approach* described above, and adds the updates of the NN every two seconds.

The windows are placed in the same fixed positions as in the previous approach, which causes problems in certain traffic situations. Figure 7 is a camera view when the vehicle is turning a corner. In this situation, some windows violate our assumption. One of the road windows is placed completely in the non-road area, and one of the non-road windows covers both road and non-road content. When extracting feature data in this situation, the NN learns using partially incorrect information. To overcome this problem, the *dynamic windowing* approach was developed, in which the positions of the road windows automatically adjust to the current road shape.

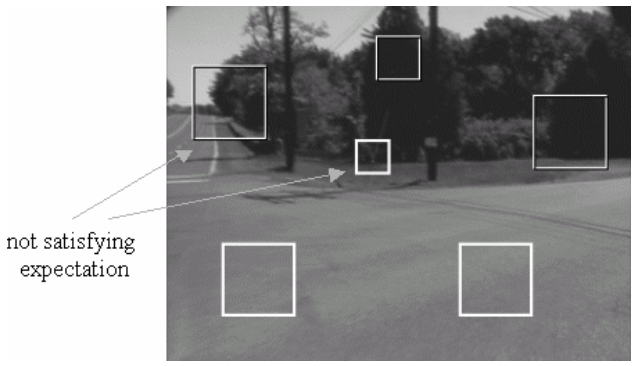


Figure 7. Example of windows violating the assumptions due to a change in the curvature of the road.

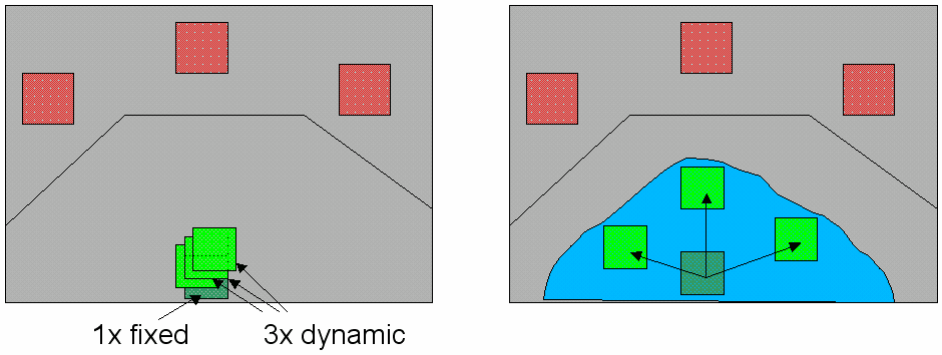


Figure 8. Approach for positioning dynamic windows.

The algorithm for the placement of dynamic road windows is as follows:

The three non-road windows stay at their fixed positions as before. On the other hand, instead of using three fixed road windows, four road windows are used. One window (the reference window) is placed at a fixed position in the lower center of the image. The other three windows dynamically move from the reference window's location in pre-defined directions.

When the road detection system is started, all windows are placed in the image as depicted in Figure 8 (left). From these locations, feature vectors are extracted and used for the initial NN training. Because the initial road classification result is based on the features extracted from the small area in the lower center of the image, the detected road area will appear as a mountain shape located around the reference window as shown Figure 8 (right).

Based on this initial road detection results, automatic placement of the three dynamic windows starts. Each window moves from the reference window's location in a pre-defined direction; one window moves upward, another window moves up and left and the third window moves up and right. The windows move in their pre-defined direction as long as they fully contain an area that was previously classified as road. From these new locations, feature vectors are collected and used to update the neural network. This process lets the detected road area grow until it fully covers the actual road area in the image.

5.4.1.1. Results

In order to analyze the performance of our road detection systems, the results of the *fixed windowing* and *dynamic windowing* methods were compared with the results of the *basic approach*. Each algorithm's performance is compared with manually annotated frames of video files. This allows computation of false positive and false negative ratios. False positives refer to non-road areas in the image that are classified by the system as road, while false negatives refer to road areas classified as non-road. The sum of both false positives and false negatives is used as an overall classification error calculated for each frame of the video sequence. After the error is calculated for each frame, the minimum and maximum classification error throughout the whole video sequence is determined and the average error is calculated. While the overall classification error per frame allows the performance of several algorithms to be compared on the same frame, the overall performance of each algorithm can be analyzed by comparing the minimum, maximum and average classification errors.

Abrupt Shadow Situation

Figure 9 shows the results for a situation in which shadows appear on the road. Two algorithms, the *dynamic windows* approach (*dwa*) and the *basic approach* (*ba*), handled the situation less accurately than the *fixed windows* approach (*fwa*).

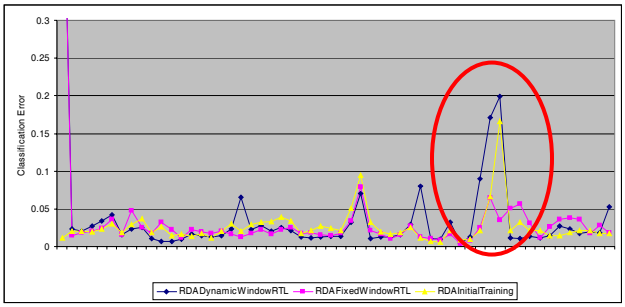


Figure 9. Classification error for abrupt shadow situation.

The reason for the bad performance of *ba* is that there was no shadow in the beginning of the course and therefore the NN was never trained on a shadowed road. Likewise, the reason for *dwa's* performance is that it cannot adapt to non-smooth changes. In contrast, the fixed road windows of *fwa* eventually (see arrow in Figure 10) covered the shadow area and allowed the NN to be trained on it.

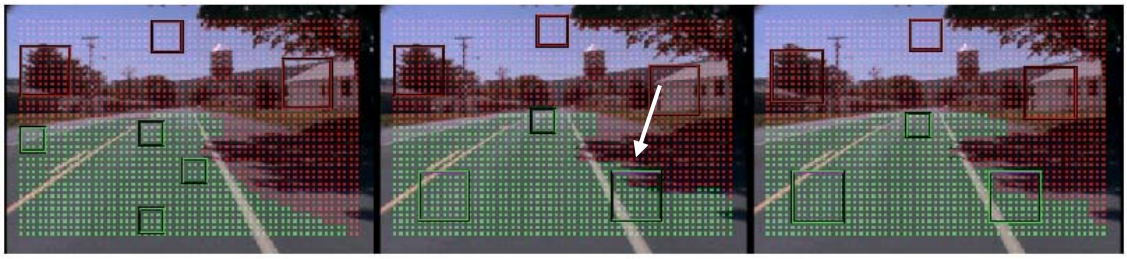


Figure 10. Abrupt shadow example for *dwa* (left), *fwa* (center) and *ba* (right).

Curvy Course Situation

Figure 11 shows the results for the situation of a curvy course. Both *dwa* and *ba* handled the situation more accurately than *fwa* did.

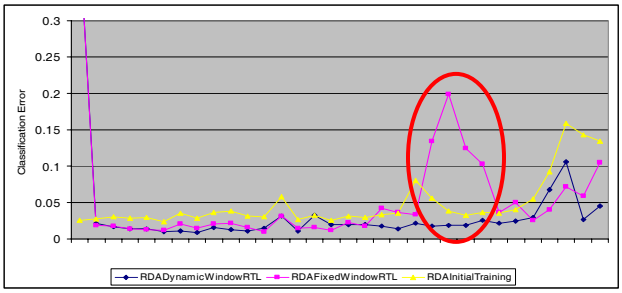


Figure 11. Classification error for a curvy course situation.

As the curvy road's appearance on the image differs from the road structure on which the location of the fixed windows was based, some of the windows violate the assumption that these windows will be either only over road areas or only over background areas. The contradictory feature data finally distort the NN (see center in Figure 12).

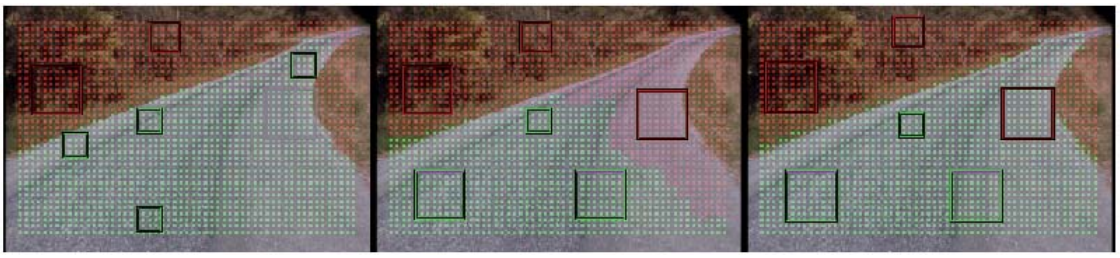


Figure 12. Curvy course example for *dwa* (left), *fwa* (center) and *ba* (right).

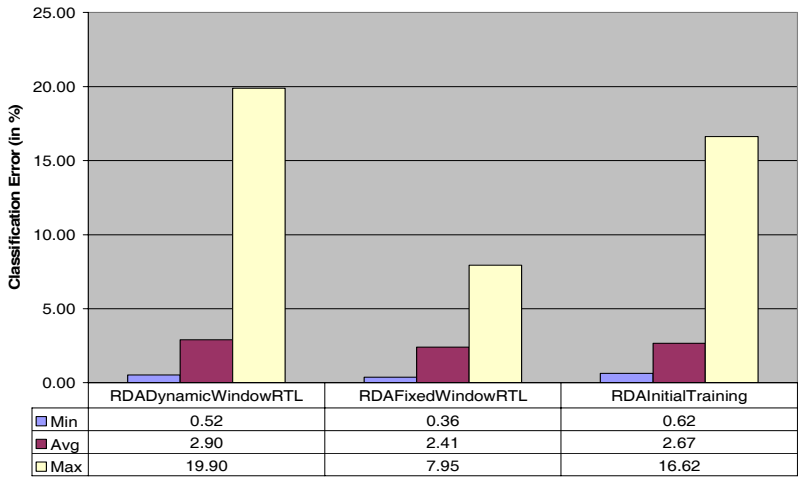
Based on the results of these two examples it can be seen that each approach, *dwa* and *fwa* has different advantages and disadvantages depending on the situation. Next, the algorithms are compared in terms of their minimum, maximum and average classification error.

Straight Road

Figure 13 is a collection of snapshots of the video sequence showing basically a straight road. Most of the course appears similar, but some abrupt shadows occur during the course.

Figure 13. Sample frames of the straight road example.

Figure 14 shows that *fwa* outperforms the other two algorithms. Both *dwa* and *ba* show a high maximum error, which is due to an abrupt shadow situation described earlier.



	RDADynamicWindowRTL	RDAFixedWindowRTL	RDAInitialTraining
Min	0.52	0.36	0.62
Avg	2.90	2.41	2.67
Max	19.90	7.95	16.62

Figure 14. Classification errors of *dwa* (left), *fwa* (center) and *ba* (right) for the straight road example.

Shadow Road

Figure 15 is a collection of snapshots of a video sequence showing a simple road with shadow throughout the whole course.

Figure 15. Sample frames of the shadow road example.

Compared to the classification errors of *ba* (see Figure 16), both of the new approaches show a slightly lower average classification error. However, the best average performance is given by *dwa* and the lowest maximum error by *fwa*. The examples show for a variety of road types that at least one of the new algorithms performs better (at least) on average. The evaluation of the algorithms shows that the static structure approach is better for sudden changes (like shadows) if the current road type complies

with the static windows' position. The dynamic approach, however, is better for smooth changes of the road's appearance as well as arbitrary road shapes. It can be concluded that the placement of feature extraction windows is the key for performance improvement. Therefore, a tight coupling of road detection and road recognition is suggested through a knowledge-based approach to road detection. Knowledge about the type of road in geometrical and topological terms would help to extract sample data for the adaptation purpose in a more informed and certain way. The knowledge of the exact location of the road in the image would prevent the extraction of erroneous sample data. This knowledge-based approach is being pursued in ongoing road detection work (see also [5], [6]).

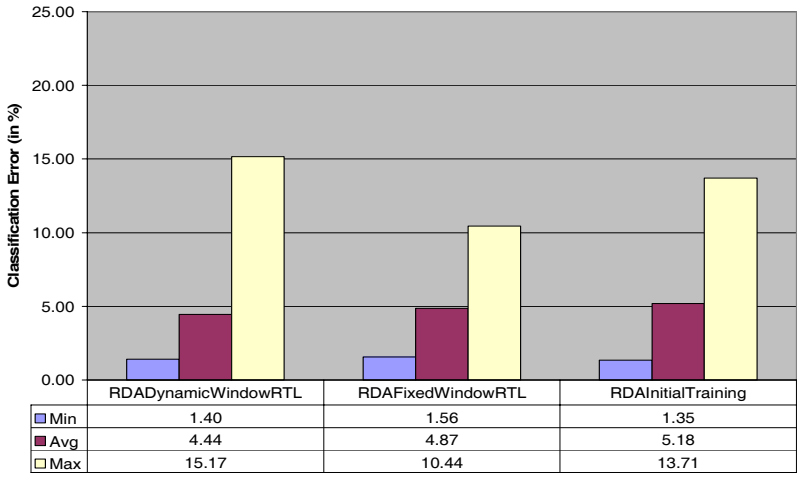


Figure 16: Classification errors of *dwa* (left), *fwa* (center) and *ba* (right) for the shadow road example.

5.4.2. Road Following Using Color Models

A new algorithm has been developed that segments road from background using color models [9]. The algorithm can be best compared with the SCARF[10] and UNSCARF[11] algorithms. Data are collected from a video camera mounted on a moving vehicle. In each frame, color models of the road and background are constructed through a scheme that makes a similar assumption about the shape and location of the road as the NN method (5.4.1), i.e., that the region in front of the vehicle is road. The color models are used to calculate the probability that each pixel in a frame is a member of the road class. Temporal fusion of these road probabilities helps to stabilize the models, resulting in a probability map that can be thresholded to determine areas of road and nonroad (Figure 17).

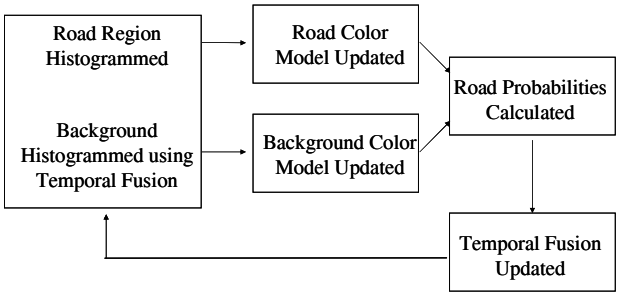
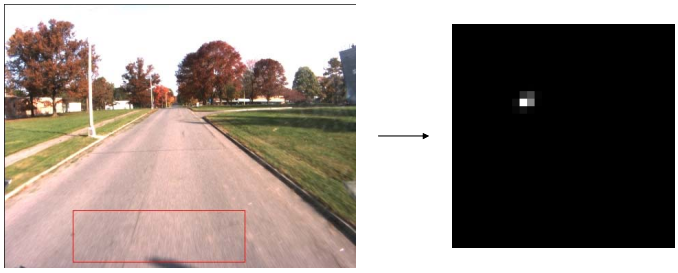


Figure 17. The system architecture after initialization.

The algorithm needs a method to represent the color distributions seen in road and background. For this color histograms are used. It was found that 30x30 histograms of normalized red (R) and green (G) gave

the best results. Normalized R and G are fairly robust to changes in illumination, while at the same time being fast to calculate. Normalized R and G are calculated as R / (R + G + B) and G / (R + G + B). Separate models are created for road and background. The major difference between the models is that for the road model, multiple color distributions are constructed, whereas the background is represented with a single color distribution model. Modeling the road with multiple color distributions helps to increase robustness. It allows the algorithm to adapt to non-homogeneous roads, shadows, illumination changes, and any other condition that causes spatial change in appearance. Having multiple color distributions allows the algorithm to learn and remember previously seen road conditions.



(a) **(b)**

Figure 18. (a) The box area is used to construct the initial road model. (b) The histogram of colors in the box.

A color distribution model for the road is implemented as a set of histograms created over time. A color distribution model starts off with a single histogram. New histograms may be added to update the color distribution model until a maximum number has been reached. At this point the oldest histogram is removed to make room for the new. This is a form of temporal integration carried out on color distributions, done in order to increase robustness, stability, and accuracy. While multiple histograms are constructed for the road, a single background histogram is constructed from frame to frame. The background color distribution model is a summation of a number of previous background histograms. If temporal fusion contains mistakes, a summation of previous results reduces their impact. The number of histograms in each color model controls how quickly the algorithm adapts to new data.

The first task is to model the road from just the region of the image assumed to be road (the box in Figure 18a). In the first frame, the region is histogrammed and used to create an initial color distribution model. Subsequent frames are processed as follows. At each frame, the region assumed to be road is histogrammed. The algorithm then can either update an existing color distribution model or create a new distribution. If the new histogram is too different from all existing distributions (determined by a threshold), a new color model is created. Otherwise, the histogram is used to update the model that is most similar.

Construction of the background model is simpler. The background is assumed to be where the road is not. To construct the background model, the algorithm looks at the previous segmentation result to determine areas of road and nonroad. The algorithm then randomly samples the nonroad areas to construct a color histogram, which is used to update the background color distribution model. Given the road and background models, the image must be segmented into road and background regions. This is done by computing, for each pixel, the probability that it belongs to each model. The pixel is then assigned to the most probable model. As there are multiple road models, multiple road probabilities are calculated at each pixel. The largest road probability is selected as the road probability for that pixel. The next step takes road probabilities from multiple frames and fuses them. The end result is a final probability map that is more consistent than individual probability maps constructed from single frames. The probability map is thresholded and regions that overlap the box are taken as road.

5.4.2.1. Algorithm Evaluation

For the evaluation of the color modeling algorithm, two sequences of images of roads were used. The first sequence is a video of the vehicle traversing a paved but unmarked road with fairly gentle curves (Figure 19 left). There is vegetation right up to the edge of the road in most places. The second sequence shows a similar road with more curves and with strong shadows that greatly change the appearance of the road (Figure 19 right). The first sequence contained 784 processed frames. The second contained 782 processed frames.

Figure 19. Left: frame from sequence 1. Right: frame from sequence 2 showing ground truth overlay.

As for the NN methods, statistics were gathered on the false positives, false negatives, and the sum of the two errors for each sequence separately (Figure 20). While these numbers give an overall impression of the performance of the algorithm, they don't by themselves capture the significance of the errors. In the graph for sequence1 in Figure 20, the false negative errors peak around frame 525 and stay relatively high through the end of the sequence. The images in Figure 21 show that this is due to the failure of the algorithm to detect the far part of the road as it goes around a bend. This could be because the road color changes as it gets further away from the vehicle, although this is not apparent from the images. Road colors might change smoothly towards the horizon, and a color model created close to the vehicle might therefore not cover the appearance of the road in the distance. Having more than one window as in the NN methods above would help overcome this problem.

It is also possible that the errors are a result of the temporal fusion step in the algorithm. Errors caused by temporal fusion are discussed further below. Looking at the lower graph of Figure 20, there is a peak in the false positive values around frame 375. As can be seen from the images in Figure 22, the errors occur at the edges of the road, at points where the road starts to curve. These are partly caused by temporal fusion and partly by the difference in road edge as judged by the human and as determined by the algorithm. These errors are largely insignificant for driving. If the locations of the errors in the images are along the edges of the road rather than in the middle, the practical value of the algorithm is much higher. The color model algorithm performs well in this way, as determined by visual evaluation of the errors overlaid on the video frames.

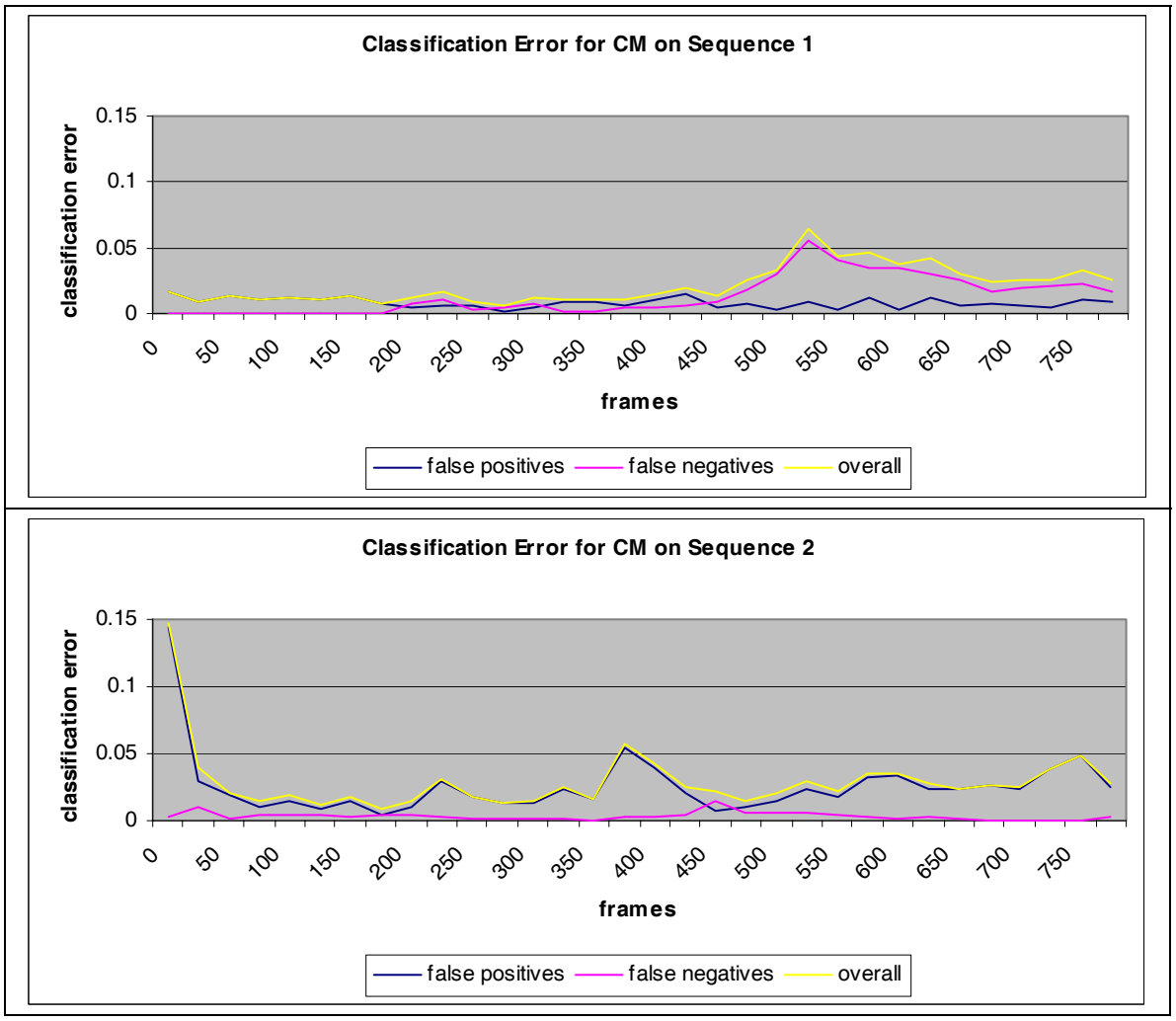


Figure 20. Error graphs for the Color Model Algorithm on the two sequences

Figure 21. Frames 400, 525, and 775 from Sequence 1 showing the false negative areas in red and the false positive areas in green.

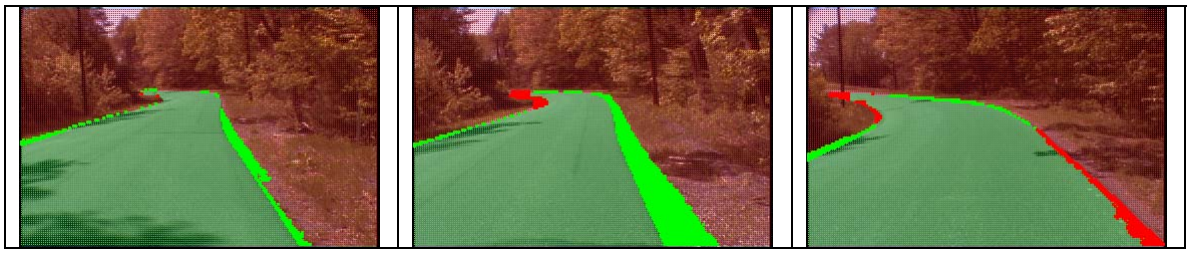


Figure 22. Left: Frame 350 in Sequence 2, just before the peak. Middle: Frame 375, at the middle of the peak. Right: Frame 450, at the end of the peak. False positives are shown in green, false negatives in red.

There is a delay caused by temporal fusion across frames, which gives rise to a small delay in updating the model. Motion of the vehicle during this time causes the mask for the road created in one frame to be applied to a later image, taken after the vehicle has moved slightly. The effect caused by left/right motion of the vehicle can cause errors at the edges of the image and gets worse with distance (Figure 23, left). It could be compensated for by monitoring the vehicle's motion and adjusting the temporal fusion parameters. A more difficult case arises when the vehicle passes a shadow, which hasn't yet been recognized as road. Behind the shadow some road will also be misclassified as non-road due to the temporal fusion delay (Figure 23, middle). The same effect occurs when the vehicle passes an intersection (Figure 23, right). A possible way to overcome this problem would be to project the temporal fusion buffer onto a flat plane in front of the vehicle and shift it towards the vehicle by an amount determined by the vehicle's motion. This would reduce the effect.

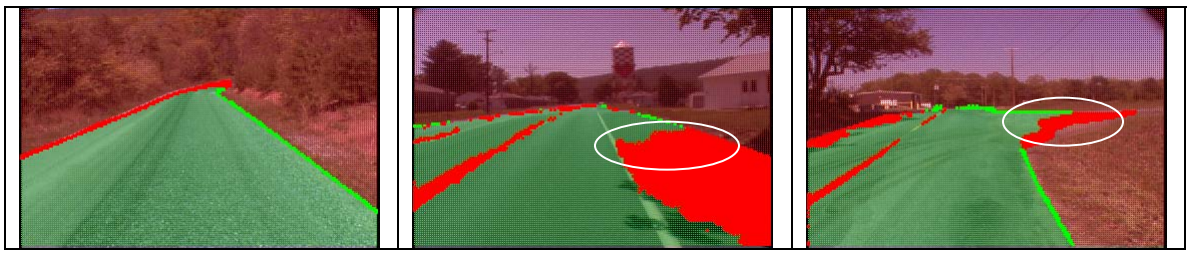


Figure 23. Left: Effects on road edge of temporal fusion. Middle: False negative after a shadow that has not yet been learned as road. Right: Similar error when the vehicle approaches an intersection.

5.4.3. Road Signs

For on-road driving, road signs provide useful information about the state of the road and legal driving actions. There has been a lot of research on road sign detection, much of which identifies candidate road sign regions using color and pruning them using shape criteria [12-23]. Piccioli et al. [12] used color and *a priori* information to limit the possible locations of signs in the image. They then extracted edges and looked for circular or triangular regions before applying a cross-correlation technique for recognizing the signs. In [13], a redness measure was used to locate stop, yield, and "do not enter" signs. This step was followed by edge detection and shape analysis to identify the sign. Escalera et al., [14], also started with color matching, which they followed with corner detection in which they looked for corners in specific relationships that correspond to triangular, rectangular, or circular signs. Classification made use of a neural network. In their approach to detecting stop signs, Yuille and his colleagues [15] corrected for the color of the ambient illumination, located the boundaries of the signs and mapped the sign into a frontoparallel position before reading the sign.

Huang and Hsu [16] used shape and color in a wide angle view to locate signs as circular or triangular shapes. They then controlled the camera to point at candidate sign locations for a closer view which was used to identify the sign based on matching pursuit filters. Another paper that describes actively controlling the camera for sign detection is [17]. Again, the researchers started with color and shape (from edges). They predicted the location of the sign and pointed the camera for a closer view. Signs were recognized and their contents read by template matching.

A decision tree method was used in [18] to detect and recognize signs without using color. Detection was based on shape using local orientations of image edges and hierarchical templates. The results were sent to a decision tree which either labeled the regions by sign type or rejected them. A method to detect speed limit signs is given in [19]. It was based on first using color to locate candidate signs, followed by a multi-resolution application of templates that looked for circular regions. Finally, the numbers were read to recognize the signs.

A very different approach is taken by Fleischer et al., [20], who used a model-based, top down approach. Predictions were made of locations in which signs could appear and shape, size, and color were used to specify edge models for the signs in the 3D world. Signs that were found were tracked through subsequent images using a Kalman filter.

Shaposhnikov et al., [24], made use of color segmentation using the CIECAM97 color appearance model. They then used histograms of oriented edge elements to determine the shape of the sign followed by location of the center of the sign. Signs were described by a set of descriptors which were matched with stored models to recognize the signs. Hsien and Chen [22] quantized colors in Hue, Saturation, Value (HSV) space and located candidate signs by projecting regions that could be signs onto the horizontal and vertical axes. Extracted sign regions were matched with templates using a Markov model, and the match with highest rank was regarded as the recognition result.

An impressive study focusing on sign detection in cluttered environments is that of Fang et al. [23]. Neural networks were used to locate regions that could be the center of signs. Both color (hue) and shape features were used. Candidate signs were tracked through time using a Kalman filter and signs were verified by a set of rules concerning the colors and shapes of the regions.

The approach described here uses color and shape to locate signs [25]. Recognition uses template matching, and signs are tracked over time. The method has the advantages of being fast and easily modified to recognize new classes of signs. The algorithm operates on a video stream taken from a camera on the roof of a test vehicle. Because the images are interlaced and the vehicle is moving, there is significant blur between fields in the images. To minimize this, images are subsampled and only every second line and column (one field) are used. Signs are detected in a multistage process, starting with segmentation based on color. Because of variation in the ambient illumination, it is not possible to search for regions with specific red, green, and blue values. To get around this problem, ratios of RGB colors are used, tailored to different classes of signs (warning, regulatory, informational, etc.). Several other authors suggested using the HSV color space instead of RGB (e.g., [22], [12]), but ratios of RGB colors worked better in this application and did not require color space conversion. For (orange) warning signs, a constraint for each pixel is that the red (r), green (g) and blue (b) values conform to the following.

$$r / g > \alpha_{warn} \;\&\; r / b > \beta_{warn} \;\&\; g / b > \gamma_{warn}$$

For predominantly red signs (stop, yield, no entry), the requirement is that

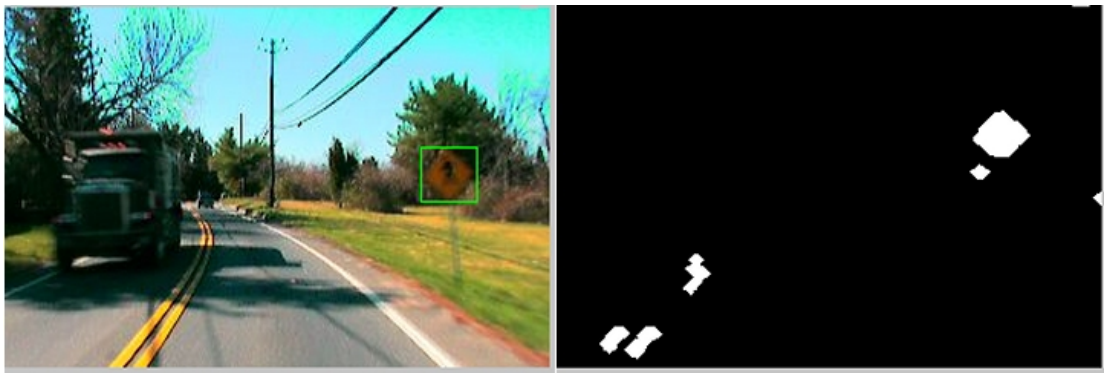
$$r / g > \alpha_{red} \;\&\; r / b > \beta_{red} \;\&\; g / b > \gamma_{red}$$

where α, β, and γ are constants. These constants are determined by sampling the red, green, and blue values of images of typical signs. The precise values of the constants are not critical for sign detection. For additional classes of sign, such as guiding signs on highways (green with white letters) a similar set of constraints would be defined, making it very easy to increase the range of signs recognized by the algorithm. This has been done for street name signs with good results.

The constraints are applied pixel by pixel to the image, which results in a binary image with 1's where pixels are candidates for belonging to a sign. The different classes of signs can all be combined into a single binary image, or each class can have its own binary image. In the latter case, more control is possible in the processing that follows this step. In this work, all classes of traffic signs were combined into a single binary image, but road name signs were treated separately.

Following the creation of the binary images, morphological erosion was performed to get rid of single pixels and two or three dilations are done to join parts of signs that may have become separated due to the presence of writing or ideograms on the signs. Figure 24 shows a scene taken from a video camera and the corresponding binary image for a warning sign (orange). The sign outline is clearly visible in the binary image. The sign outline was also used as a mask in the sign recognition stage.

The next step was to find connected components in the binary images and identify blobs that are likely to be signs. Properties of each component were computed, including the centroid, area, and bounding box.



(a) **(b)**

Figure 24. (a) shows a frame from the video with a detected sign. (b) shows the binary map for warning signs, with the shape of the sign clearly visible. The other blobs in the image are rejected as signs because they are either too small, are of the wrong shape, or are in a region where signs are not expected.

They were used in a set of rules that accepted or rejected each blob as a sign candidate. The rules required the area of the blob to be greater than a minimum and less than a maximum, the height to width ratio to be in a specified range, and the centroid to be in a restricted part of the image where signs could be expected to appear. The ratio of the area of the blob to the area of the bounding box was restricted to prevent blobs that are too thin from being accepted. Blobs that conformed to the rules were considered to be candidate signs and were tracked from image to image. If a blob was seen in three successive frames, it was confirmed as a candidate and went on to the recognition phase of the algorithm.

Recognition was achieved by template matching. A preprocessing step was first applied to each candidate sign. It masked out the background surrounding the sign which would otherwise interfere with the template matching. The results of the sign detection phase were used, since that step had already constructed a mask for the sign (Figure 24). Using this mask resulted in good segmentation of the sign region from the background (Figure 25). The masked candidate signs were scaled to a standard size (48x48 pixels) and compared with stored signs of the same size. The stored sign templates were taken from video sequences similar to those being recognized. Because there was a lot of variation in sign appearance, several stored templates were needed for each canonical sign.

Figure 25. Candidate signs before and after background masking.

Figure 26. Examples of recognized road signs.

5.4.3.1. Results

The algorithm was tested on video sequences recorded with a camera mounted on the roof of a vehicle driving on suburban roads at normal driving speeds. The results reported here are for a total of 23,637 frames containing 92 warning and stop signs. Figure 26 shows examples of recognized signs. Table 1 shows the results for three individual runs and the combined totals. As can be seen, the sign detection phase misses very few signs, but at the cost of substantial false detections. In the first data set, this is due to a single vehicle that was in front of the test vehicle in most of the frames.

The sign detection algorithm was sensitive to the brake lights of the vehicle, and almost all the false detections are related to this feature. The second data set was acquired in early spring, when the forsythia bushes were in flower. The color of the flowers is very close to that of warning signs, and the bulk of false detections were due to this coincidental color match. All of the false detections are rejected by the recognition phase. The false recognitions in the tables resulted when a correctly-detected sign was incorrectly recognized. The penalty for the extra detections is that the recognition process has to be run on each of them. This slows down the processing, although the algorithm still ran at over 20 frames per second on a 1.6 MHz Intel Pentium Mobile processor. This was fast enough to interact with the control system in real time.

Run	No. Frames	No. Signs	No. Detected	No. Recognized	False Detections	False Recognitions
1	10, 924	39	33	29	19	1
2	9, 439	45	43	38	34	5
3	3, 274	8	5	5	0	0
1, 2, 3	23, 637	92	81	72	53	6

Table 1 Performance of sign detection and recognition.

5.4.4. Curb Detection

Curbs are detected using a line-scan LADAR. The sensor is mounted on the bumper of the vehicle looking down at an angle to the road. The beam sweeps across the road and records a profile of the road surface. Curbs appear as regions that are higher than the road and that rise steeply. They are detected using the obstacle detection algorithm described in Section 5.3 and are the first obstacles detected when scanning outwards from directly in front of the vehicle in both directions. In Figure 27, the green section corresponds to road surface, the red to obstacles, and the yellow to high obstacles.

Curbs (or more accurately, road edges) can also be detected as the boundaries of the road surface detected as described in Section 5.4.1. Having two ways of detecting these features gives added confidence and makes the algorithms more robust.

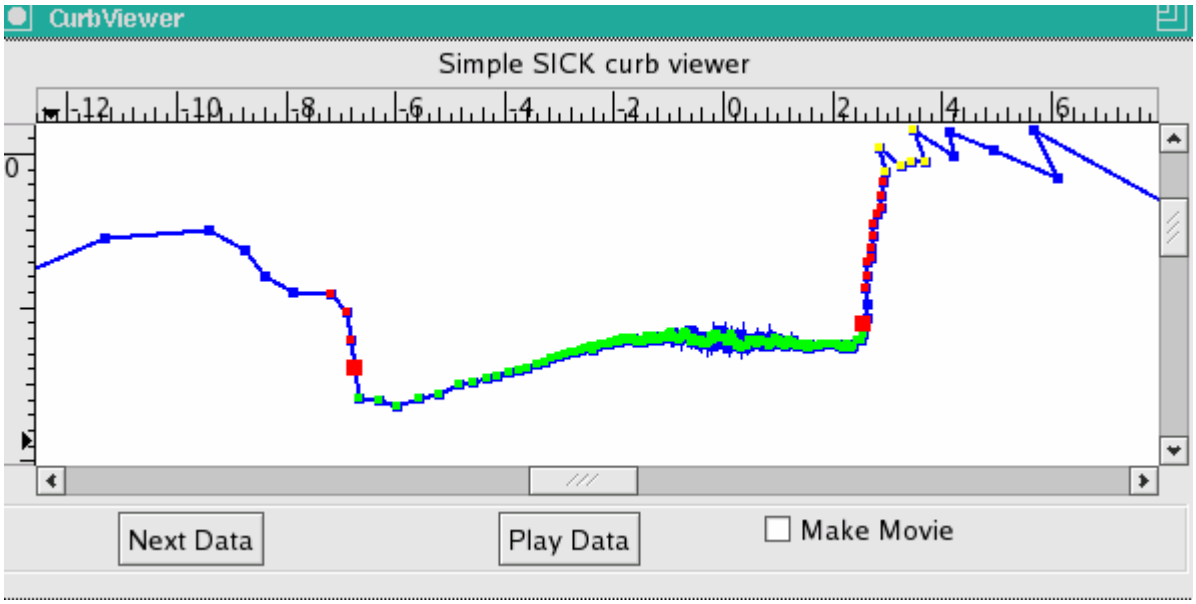


Figure 27. A sample line scan showing the road surface and the curbs.

5.4.5. Water (Puddle) Detection

Puddles are a particular problem for off-road mobile vehicles because they are hard to detect, they often mimic other features, and they may pose a serious hazard to the vehicle. It was also found that puddles appear as smooth, level surfaces that the motion planning system would very much like to traverse.

Finding puddles relies on sensor fusion of LADAR and color images. Puddles are tracked over time using predictions from the world model. The search for puddles begins in the LADAR domain (Figure 28). Advantage is taken of the fact that laser beams hitting a puddle at an oblique angle are reflected away from the sensor, and result in no data being returned. Such points show up as voids in the LADAR images, but are not the only sources of missing data. Other sources include negative obstacles (holes, ditches), occlusions, and out-of-range responses.

The puddle detection algorithm thus looks for voids in the data, and then scans a region surrounding them. Because a puddle must lie on the ground surface, a requirement is that there be ground points adjacent to the puddle and that the ground points on either side of the puddle (along a row of the image) have similar elevation. Since most of a puddle region is void, linear interpolation is used between the ground points on either side of the puddle to estimate the position of the puddle in the world model map. For every detected puddle point in the laser image, the puddle confidence is increased by a predefined constant, which depends on the reliability of the sensor. In experiments it was found that using laser alone was not sufficient to

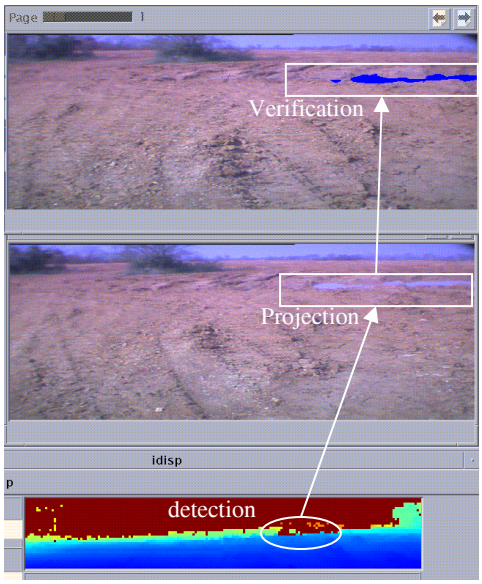


Figure 28. Bottom: Puddle detected in LADAR image. Middle: Window of attention projected into color image. Top: Verification.

uniquely identify a puddle. It was decided to look in the color image for supporting evidence to reduce false positives. When a puddle is detected in the laser data (with sufficient confidence in the map), a window is placed about the potential puddle region, and projected into the color image. In the color domain, the system tries to determine if the region has a similar color to what is above it. Often, this will be the sky, so a blue color will mean a puddle. At other times, however, the puddle may reflect trees, grass, or clouds. The algorithm searches for a match, but may fail if the puddle is reflecting something not in the image. When the puddle is verified, the puddle points then project into the map. The confidence of puddle in the map is increased by a predefined value that depends on the robustness of the color classification algorithms.

5.4.5.1. Projection

Since the sensors are mounted on a mobile platform, and the sensors themselves move, projections are not fixed, but must be computed each time they are needed. There are two kinds of projections: The LADAR data are projected into the world model, and points identified as puddles are projected into the color image space. Each sensor is at a known base position on the vehicle, and has a known sensor coordinate system (Figure 29). The vehicle is moving, however, and the world model maintains its representation in world coordinates, fixed on the ground. Thus, all coordinates must be converted from sensor to vehicle and from vehicle to world. Some of the sensors also move relative to their base position. The LADAR, for instance, rotates about its horizontal axis (tilts). This must be factored into the transformation. Finally, the sensors sample at different times, so a correction must be made for their relative positions in space when mapping between images.

The LADAR to world model coordinate transformation includes the LADAR-to-vehicle (including LADAR tilt angle) and vehicle-to-world-model transformations. A data point in the LADAR image can be transformed to the world model by the equation:

$$\begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} = H_v^w * H_l^v * \begin{bmatrix} X_l \\ Y_l \\ Z_l \\ 1 \end{bmatrix}$$

Where $[X_w \quad Y_w \quad Z_w \quad 1]$ and $[X_l \quad Y_l \quad Z_l \quad 1]$ are coordinates of a datum point in world model coordinates and LADAR coordinates respectively. H_v^w and H_l^v represent 4x4 transformation matrices from the vehicle to the world model and the LADAR to the vehicle.

The back projection from the world model to the color camera image includes world-model-to-vehicle, vehicle-to-camera and camera-to-image transformations.

Let f be the focal length of the camera. Then the projected image position is

$$\frac{Y_i}{f} = \frac{Y_c}{Z_c}$$
$$\frac{X_i}{f} = \frac{X_c}{Z_c}$$

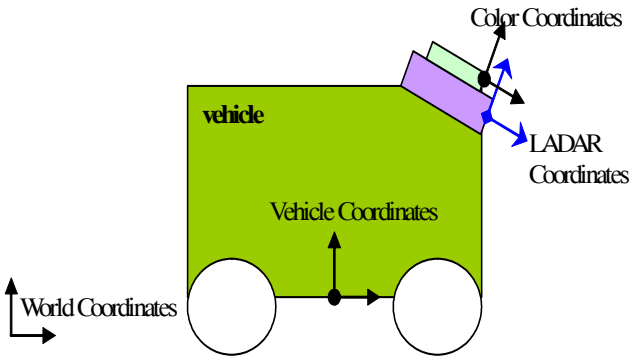


Figure 29. Coordinate systems on the mobile robot.

Prediction

Prediction is used to focus attention on regions that have previously been identified as interesting. It facilitates tracking, enables confidences in features to be updated, and allows information found in one sensor to influence processing in another. Prediction is mediated in the system by the world model. Since a grid representation fixed to the world is used, it is straightforward to project regions in the world model into each of the sensor coordinate systems. Currently, predictions are only made of where a feature is expected to occur, not what it may look like.

In the case of puddles, prediction plays an important role in LADAR processing. As the vehicle approaches a puddle, the angle at which a LADAR ray hits the water gets steeper, and at some point, the sensor starts to record a return from the puddle. Without knowing that the region had already been identified as a puddle, the sensor would start to indicate that the region was traversable and smooth, which would make it

a preferred location for the planner. Marking a region as already identified prevents this behavior. Note that puddles are unusual in that most features have their confidences increased by multiple views, whereas using the LADAR sensor to view puddles over time reduces their confidence. The behavior of the LADAR sensor in the neighborhood of other features that produce voids, such as holes and occlusions, is very different from that around puddles. This enables a distinction to be made over time that could not be made from a single view. Prediction ensures that corresponding features are tracked and labels remain consistent.

5.5. Learning

In the real world, robots cannot be expected to know everything they may encounter while carrying out their tasks. Thus, they must be able to adapt to new situations, and learn how to operate within a changing environment. To do so, they need a way to store pertinent information about the environment, recall the information at appropriate times, and reliably match stored information to newly-sensed data. They also must be able to modify the stored information to account for systematic changes in the environment. This section describes an approach to learning traversability based both on examples and on experience [26]. A fast learning algorithm is introduced that requires no training data to learn associations between appearance and traversability, and a histogram-based representation of models is described that provides a well-defined way of comparing the models and matching them with sensed data. The models are composed of color and texture features that do not rely on range data. This enables them to be used to classify regions for which no range data are available. The models are learned from data selected to be close together in space, making it more likely that they are from the same physical region. The learning module extends the 4D/RCS architecture by including learning of entities both in the maps kept by the World Model and as symbolically represented objects.

Note that the road detection and road following algorithms described in Sections 5.4.1 and 5.4.2 use learning to adapt to changes in the appearance of roads over time. In the neural network algorithms, the weights in the network are learned, while in the color model algorithm, new models replace old ones as the road appearance changes.

In order to apply the algorithm below, the vehicle must have at least a color camera, a range sensor that can measure range over an area (e.g., a stereo system) and an inertial navigation system that provides an estimate of the vehicle's position in space. The two vehicles for which the approach was developed have these sensors. NIST operates a High Mobility Multipurpose Wheeled Vehicle (HMMWV) that has several color cameras, including one mounted on top of an area-scanning ladar. The vehicle used for the Defense Advanced Research Project Agency's (DARPA) LAGR program (Learning Applied to Ground Robots) has twin color stereo systems (see Figure 30). Both of these platforms provide range and color information to the vehicle. Each vehicle also has navigation sensors that provide position estimates.

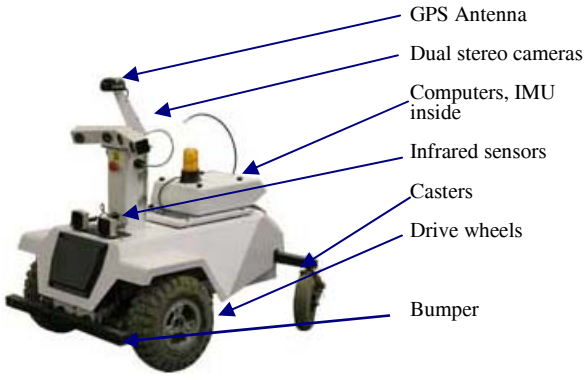


Figure 30. The DARPA LAGR vehicle, showing the sensors and actuators.

The problem faced by a robot of finding a path to a goal point is a feedback control problem. The sensed feedback information comes from the cameras, GPS, and other sensors. The actuators are the drive motors on the wheels. The on-board computer implements the feedback controller that drives the vehicle position (part of the state) to the goal position. It is for this reason that there are similarities between learning methods for robots and the field of adaptive control (sometimes called learning control). The closest relationships are to the area of "on-line approximation based feedback control" [27], and in particular the "indirect adaptive control strategy" where a parameterized nonlinear map is adjusted to represent the process (environment) and then control decisions are based on that map. Moreover, extended notions of adaptive control use learned models for planning and route selection by marrying ideas in adaptive and "model predictive control" [28]. Indeed, the map-based strategy here is an excellent example of how successful such approaches can be.

The notions of learning described here arose in the field of psychology. First, the most basic low levels of learning, represented by the notions of "habituation" and "sensitization" [29], are embedded in the algorithms. If the robot learns via multiple sensor inputs that an area is traversable, then it has been habituated to that input (it has learned to ignore information and go ahead and travel in a direction). Correspondingly, if the robot has learned that some sensory inputs correspond to a lack of traversability, then if such situations are encountered again the robot is sensitized and hence may not make the same attempts to travel through non-traversable areas as it did in the past. Such learning in the form of habituation and sensitization sets the foundation for the elements of "classical and operant conditioning" [29] that occur in the robot. Cell update strategies correspond to learning strategies where via repeated sensory inputs the system can learn to associate sensed features with a lack of traversability or good traversability so that the basics of classical conditioning are present. Indeed, the robot can exhibit the property of "blocking" since in learning it can initially use some sensed information to determine traversability, and then later when there are other learning opportunities, it will at times ignore new sensory information (model updates) since it is confident that for instance more sensory verification of the model is not needed. With respect to the "behaviorist" approach to operant conditioning, if the robot senses some scene and it has learned that certain features are associated with rewards (getting closer to the goal by making forward progress), it will try to apply the same actions that were successful before, leading to the "Thorndike's effect" similar to what occurs in a "Skinner box" [29]. And, such opportunities for conditioning can occur during a single attempt by the robot to find a goal point via storage, updating, and later use of information in our maps as the robot travels. Moreover, the learned maps can be used between trials so that on successive attempts the robot learns how to direct its behavior to succeed even faster; hence, a basic property of "speed-up" in the rate of reward acquisition seen experimentally in rats in mazes [29] can also be exhibited by the system. Finally, note that the use of maps is quite similar to the idea that animals and humans build (learn) and use "cognitive maps" of their environment for planning spatial movement ([30], [31], [32]).

5.5.1. The Algorithm

The algorithm analyzes range data to identify regions corresponding to ground and to obstacles. This information is used with color data to construct models of the appearance of the regions. The models include an estimate of the cost of traversing the regions. The models are then used to segment and classify regions in the color images. Associating regions with models enables traversability costs to be assigned to areas where there is no range data and thus no directly measurable obstacles. As the vehicle traverses the terrain, more direct information is gathered to refine the traversability costs. This includes noting which regions are actually traversed and adjusting the traversability of the associated models. It also involves adjusting the traversability of regions where the vehicle's mechanical bumper is triggered, and where the wheels slip or the engine has to strain to move the vehicle.

5.5.1.1. Building the Models

First, a local occupancy map is built, consisting of a grid of cells, each of which represents a fixed square region in the world projected onto a nominal ground plane.

An array of *201x201* cells is used, each of size 0.2 m square, giving a map of size 40 m on a side. The map is always oriented with one axis pointing north and the other east. The map scrolls under the vehicle as the vehicle moves. The model-building algorithm takes as input a color image, the associated and registered range data (*x, y, z* points) from a ranging sensor or from stereo, and the labels (GROUND and OBSTACLE) computed by the obstacle-detection algorithm of Section 5.3. It builds models by segmenting the color image into regions with uniform properties. When a data set becomes available for processing, the map is scrolled so that the vehicle occupies the center cell of the map. Each point is processed as follows.

1. If the point is not labeled as GROUND or OBSTACLE, it is skipped (Other labels can be treated without significant changes to the algorithm). Points that do not have associated range values are also skipped.

   Points that pass step 1 are projected into the map using the known *x*, *y*, and *z* values and the pose of the vehicle. If a point projects outside the map it is skipped. Each cell receives all points that fall within the square region in the world determined by the location of the cell, regardless of the height of the point above the ground. The cell to which the point projects accumulates information that summarizes the characteristics of all points seen by this cell. This includes color, texture, intensity, and contrast properties of the projected points, as well as the number of OBSTACLE and GROUND points that have projected into the cell.

   Color is represented by ratios R/G, R/B, and G/B rather than directly using R, G, and B. This provides a small amount of protection from the color of ambient illumination. Each color ratio is represented by an 8-bin histogram, representing values from 0 to 255. The values are stored in a normalized form, meaning that they can be viewed as probabilities of the occurrence of each ratio. Texture and contrast are computed using Local Binary Patterns (LBP) [33]. These patterns represent the relationships between pixels in a 3x3 neighborhood in the image, and their values range from 0 to 255. Similarly to the color ratios, the texture measure is represented by a histogram with 8 bins, also normalized. Contrast is represented by a single number ranging from 0 to 1, and intensity as another 8-bin histogram.

2. When a cell accumulates enough points it is ready to be considered as a model. In order to build a model, a minimum percentage (currently 95%) of the points projected into a cell must have the same label. If a cell is the first to accumulate enough points, its values are simply copied to instantiate the first model. Models have exactly the same structure as cells, so this is trivial. If there are already defined models, the cell must first be matched to the existing models to see if it can be merged or if a new model must be created.
   Matching is done by computing a score, *Dist*, as a weighted sum of the elements of the model, *m*, and the cell *c*. That is, $Dist(c,m) = \sum w_i f_i(c,m)$ where f_i is either a measure of the similarity of two histograms or, in the case of contrast, is the absolute value of the difference of the two contrast values, $f_{contrast} = \left| contrast_m - contrast_c \right|$. The histograms are always stored normalized by the number of points. Various measures f_h of the similarity of two histograms (discrete probability density functions) can be used, such as a Chi Squared test or Kullback-Liebler divergence. After trying these (and others) it was found that a sum of squared difference measure worked almost as well and is cheaper to compute. Thus, for each model histogram h_m and the corresponding cell histogram h_c,

$$f_h = \sum_{i=1}^{8} \left(h_{mi} - h_{ci} \right)^2$$

   Cells that are similar enough are merged into existing models; otherwise, new models are constructed. If the number of models exceeds a limit, merging of the most similar models is forced. Merging is a straightforward summation of histograms, each normalized by its number of

points. The merged contrast measure is computed as the weighted average of the two contrasts being merged. Figure 31 shows the histograms representing three different models.

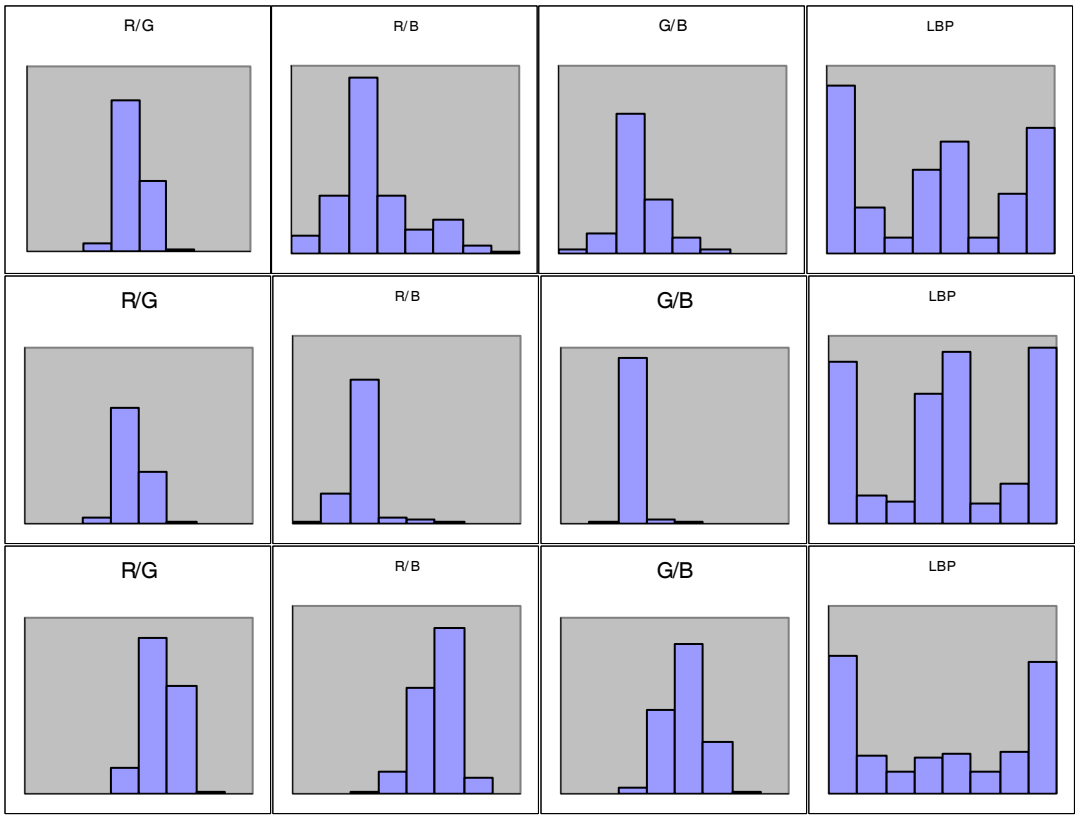


Figure 31. The histograms representing three different models. Models include other elements, such as contrast and traversability.

3. At this stage, there is a set of models whose appearance in the color images is distinct. Our interest is not so much in the appearance of the models, but in the traversability of the regions associated with them. Traversability is computed using three types of information. First, when a point is projected into a cell, it brings with it a label, either GROUND or OBSTACLE. Each cell accumulates a count of the number of GROUND and OBSTACLE points that have been projected into it. Second, the vehicle itself occupies a region of space that maps into some neighborhood of cells. These cells and their associated models are given an increased traversability weight because the vehicle is traversing them. Third, if the bumper on the vehicle is triggered, the cell that corresponds to the bumper location (and its model, if any) is given a decreased traversability weight. Cells and models that don't have known traversability from bumper hits or from being traversed are given traversability values computed as *numOBSTACLE / (numOBSTACLE + numGROUND)*. The traversability can be further modified by observing when the wheels on the vehicle slip or the engine has to work harder to traverse a cell.

4. When all the points in the input data have been processed, the occupancy map is sent to the World Model (WM) as follows: First, only cells that have values that have changed are sent. If a cell does not have an associated model, its local traversability measure is sent. If it does have a model, the traversability computed from the model is sent. This means that information learned in one region is propagated to other, similar regions. Note that the WM has no knowledge of the local models, and receives only traversability information rather than region identity. Each time new data come in, the process is repeated.

5. Periodically, a sweep is made of all the models. Each model is compared to all the others. If two models are similar enough, they are merged and the number of models is reduced accordingly.

5.5.1.2. Classifying scenes

The next step is to use the models to classify entire scenes. Here only color information is available, with the traversability being inferred from that stored in the models. The assumption is that regions that look similar will have similar traversability. The approach is to pass a window over the image and to compute the color and texture measures at each window location. The matching between the windows and the models operates exactly as it does when a cell is matched to a model in the learning stage. Windows do not have to be large, however. They can be as small as a single pixel and the matching will still determine the closest model, although with low confidence. In the implementation the window size is a parameter, typically set to 16x16. If the best match has an acceptable score, the window is labeled with the matching model. If not, the window is not classified. Windows that match with models inherit the traversability measure associated with the model. In this way large portions of the image are classified.

Sending the results to the World Model requires a 3D location to be associated with each point. For the part of the image used for model creation, only points that have associated range values are processed, so the problem does not arise. For the rest of the image the approach is based on two assumptions. One is that the ground is flat, i.e., that the pose of the vehicle defines a plane through the wheels. This allows windows that match with models that are created from ground points to be mapped to 3D locations. The second assumption is that all obstacles are normal to the ground plane. This allows obstacle windows to be projected into the ground plane and thus to acquire 3D locations.

Figure 32(a) shows an image taken during the learning process. The pixels contributing to the learning are shown in red for obstacle points and green for ground points. Figure 32(b) shows an image to be labeled with traversability values. Figure 32(c) shows the labeled image using the models built from previous traverses of similar terrain.

(a) **(b)** **(c)**

Figure 32. (a) An image from the model learning sequence. The green regions have been determined to be ground from the stereo data and are used to construct ground models. The red regions are used to construct obstacle models. (b) An image to be classified. (c) The classified image, using the previously-generated models. Yellow regions are classified as ground, while magenta regions are obstacles.

5.6. Performance Evaluation

Chapter 9 covers performance evaluation of intelligent systems as a whole. Here the evaluation of SP algorithms is discussed. Historically, performance evaluation has not been commonly practiced in the perception community. Periodically, efforts are made to persuade researchers to provide performance evaluations that can be substantiated, but only a few take up this challenge. As a result, performance evaluation is ad hoc in general and quite frequently completely absent from research papers.

In Europe, a number of formal programs have been developed that address performance evaluation of vision algorithms. Of these, ECVnet, an association of European vision researchers, had a subcommittee on Benchmarking and Performance Measures [34], although it now appears to be defunct. The German Association for Pattern Recognition (DAGM) established a Working Group on "Quality Evaluation of Pattern Recognition Algorithms", but it, too appears inactive [35]. The International Association for Pattern Recognition has a Technical Committee on Benchmarking & Software, which organizes performance competitions comparing algorithms for particular applications, such as fingerprint identification and document analysis [36]. There have also been a number of workshops on performance characterization and benchmarking of vision systems.

There are a number of publications that address how to evaluate the performance of vision algorithms, and a few examples of careful evaluations of particular algorithms or classes of algorithms. Approaches to performance evaluation can be classified into the following general categories, recognizing that more than one approach may be used in an evaluation:

Comparative Here an algorithm may be compared with others that attempt to address the same image-processing task, or its performance may be compared to "ground truth," or perhaps to human performance [37-42].

Analytic The theory behind the algorithm is examined to try to determine the limits to its operation. The computational complexity may be derived, or theoretical optimality may be determined under certain constraints. Frequently, the approach makes use of simplified input data to make the analysis feasible [43-46].

Performance The way the algorithm actually performs on test data is measured and execution times with different parameters may be reported [47-49].

Appropriateness to Task The algorithm is shown in the context of a particular application, and the constraints of the task are used to justify the selection of the particular algorithm. The performance of the task as a whole is taken as the evaluation of the algorithm [50,51].

Other, more informal measures include generality and acceptance. Perhaps the only real performance evaluation measure in common use is longevity. Algorithms that are accepted widely and implemented by many people for different applications can be considered good performers.

A large number of papers report excellent performance of their algorithms, based on small data sets. The success of the FERET program [42] was the inspiration for producing a large database of ground truth for the domain of mobile robotics. In this domain, sensors are mounted on board the moving vehicle, and the algorithms are constrained to run in real time (i.e., fast enough to provide data to control the vehicle). A rigid and reliable methodology has been developed for producing three different kinds of large databases of sensor data with ground truth. One method involves collecting ground truth data using a highly accurate Riegl (Riegl LMS Z210) LADAR sensor mounted on our instrumented HMMWV. The LADAR can characterize large areas of terrain and is registered with cameras that provide color information for each LADAR point. The position and time at which each sample is collected is recorded with an INS and GPS accurate to a few centimeters. Another set of data was obtained through a high-resolution aerial survey of the grounds of the NIST and surrounding area. The survey includes annotations providing labels for all the features. Lastly, an interactive method has been developed that reduces the amount of hand-labeling of features in image sequences to efficiently generate a large database of ground truth data. The performance evaluation of the road following algorithms used this last method of gathering ground truth.

The data sets are used to evaluate performance of algorithms objectively by comparing the output of the algorithms to the expected result derived from the ground truth. Given a large number of ground truth data sets from different environments, statistical evaluations are possible as well as the robust assessment of performance of algorithms. In the following, it is assumed that all sensors used to collect data and produce ground truth have been calibrated and necessary parameters such as the optical center of a device or the focal plane are known.

First to be discussed is the method for creating ground truth databases for sequences of color image data. It involves a human user, who annotates the data to supply the ground truth. Manually annotating sensor data

with ground truth is costly and time consuming. To minimize this effort, a semi-automatic ground truth application has been developed that reduces cost and time by requiring only occasional annotation. The user annotates the first image of a sequence by outlining and naming regions of interest (e.g., highway signs, vehicles). The computer then tracks the annotated regions through successive images, and the user observes how well each region is recognized and outlined by the computer. When the annotations start to diverge from the desired regions, the user intervenes and re-identifies the regions, retaining the same names. When new regions appear that the user wants to track, the same process of stopping the computer, annotating the regions, and restarting the tracking is followed. The annotation application can be used to outline regions with curved or polygonal lines, and several tracking algorithms can be used, depending on the objects in the images. The output of this process consists of the names, shapes and position coordinates of the targets in each image, stored in a relational database.

Figure 33 shows the starting frame of a sequence of color images. It shows road edges that were selected by a user constructing the ground truth. Figure 34 shows the results of automatic tracking. The tracking to this point is acceptable, and no user interaction is required. Figure 35 shows the situation when the automatic tracking is starting to drift. At this point, the user stops the tracking, resets the annotation, and lets the tracker continue (Figure 36).

Figure 33. The first frame of a sequence. The user has drawn the features to be tracked.

Figure 34. The computer tracks the features through a sequence of images.

Figure 35. In this frame, the automatic tracker has drifted enough to require human intervention.

Figure 36. The user re-initializes the features and automatic tracking continues.

The second method provides data for evaluating range sensors. It makes use of the high-resolution Riegl LADAR to construct a map of a region. The map can then be used for evaluating range sensors that have significantly lower resolution than the Riegl. A 5 cm x 5 cm spatial resolution grid is used to construct the ground truth map, but maps can be constructed at different resolutions (finer or coarser). This method has been used to gather ground truth for off-road terrain such as that shown in Figure 37.

Figure 37. An image mosaic used to provide color information for the high-resolution LADAR scanner.

Evaluating other range sensors involves mapping their data into the high-resolution map. The residual of the Riegl data and the other sensor data provides a measure of the performance of the sensor (relative to the Riegl). It is important to note that in order to map data from the sensor under test onto the Riegl data, the positions and orientations of the sensors must be known accurately. The current map resolution of 5 cm x 5 cm corresponds to a spatial tolerance of 5 cm. This method of constructing a map of a region can also measure how much information each successive LADAR image adds about the world. The ground truth maps can also be used to evaluate similar maps constructed with stereo algorithms [52].

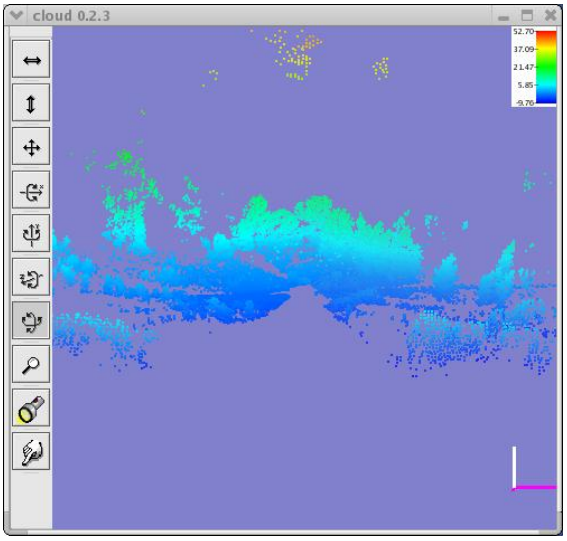


Figure 38. Range data from the Riegl LADAR. Color is used to represent elevation.

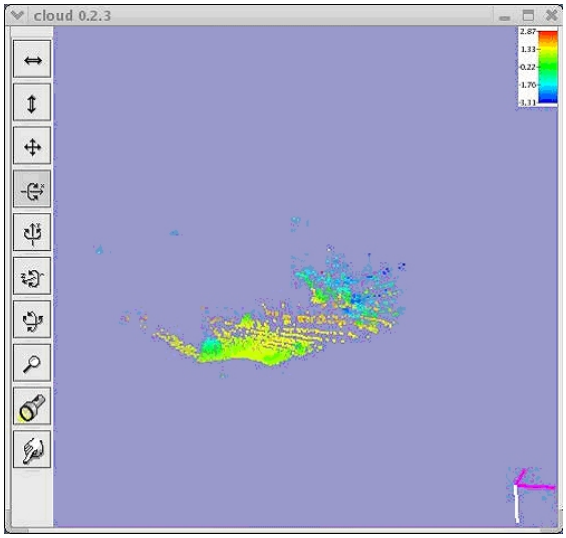


Figure 39. The sub-region seen by the GDRS LADAR, taken from the same position. Elevation is again represented by color.

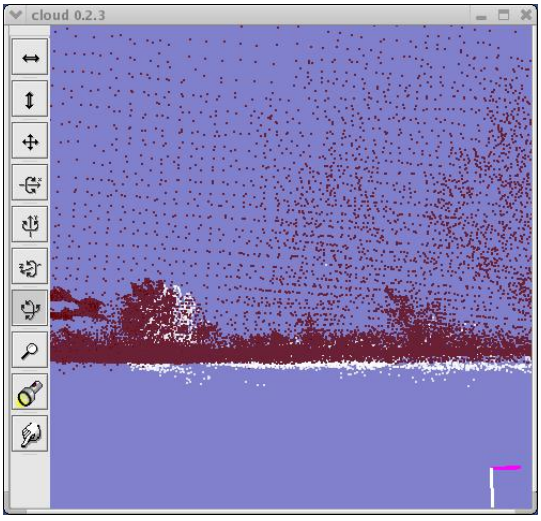


Figure 40. The result of overlaying the GDRS LADAR data on the Riegl data. The difference in measurement of the scene can clearly be seen.

Figure 38 shows the result of scanning the region in Figure 37 with the Riegl LADAR. Figure 39 shows the sub-region scanned with a different LADAR (GDRS). In Figure 40 the two scans are overlaid. The white region shows the mismatch due to the lower resolution and coarse range quantization of the GDRS LADAR with a small component due to registration error.

The third method of performance evaluation involves constructing a ground truth database of color and range images based on a high-resolution aerial survey combined with data from calibrated ground sensors such as cameras and LADARs. A survey of the NIST campus (234 hectares, or 578 acres) was commissioned, as well as of part of the surrounding urban area. The area includes roads, parking lots, traffic signs, buildings, trees, streams, fences, etc., as well as off-road terrain. All of these features were recorded and entered into a database of features and terrain elevation. Ground truth for each sensor can then be extracted from the database based on position and sensor model.

In order to produce the ground truth database from a sensor or set of sensors, each sensor is mounted on the NIST HMMWV or other calibrated vehicle with an accurate position sensor, or a tripod or other stationary mount whose position can be obtained accurately. If the sensors are to be used in real time autonomous driving, the vehicle is driven over the NIST grounds, preferably over the kind of terrain on which the sensors will be used and data is collected with associated time and position stamps [53].

A high resolution INS/GPS navigation sensor coupled with a differential GPS base station and post-processing of the position data enables determining the location of the NIST vehicle with an accuracy of about 4 cm in position and a few thousandths of a degree in orientation. The location of each sensor on the vehicle can also be precisely measured using the techniques described in [53]. This enables sensor data to be transformed into the vehicle coordinate system, or into the coordinates of the aerial survey of the NIST campus. The labels of sensed data can then be obtained from the ground truth.

With this procedure, a large ground truth database can be produced easily. Given a dataset captured in this way, one can borrow the evaluation procedure from the FERET program [42] to quantitatively evaluate the performance of sensor-processing algorithms such as segmentation, classification, and recognition algorithms. These algorithms produce labeled regions in an image. The regions can be assigned labels from the ground truth by projection into the *a priori* data (Figure 41-Figure 43). It then becomes a simple matter to determine the percentage of false positive and false negative labels of each algorithm and the correctness of the detected positions and shapes of the objects.

The ground truth data are also an excellent resource for verifying the accuracy of a LADAR sensor by taking samples from locations that contain surfaces or objects of known sizes, distances, and orientations. The response of the algorithm is then compared with the ground truth position, which is extracted from the database of prior knowledge based on the known position of the sensor and its field of view. Obviously, all measurements are limited by the accuracy of the *a priori* data and the accuracy with which the position and orientation of the sensor can be established with respect to the *a priori* data. A sample-by-sample measurement can be made, giving the range resolution and field of view of the sensor. Alternatively, feature-based measurements can be made, giving the accuracy with which the sensor can capture surfaces of different shapes and slopes. More detailed studies, such as trying to determine which part of the field of view of a single sample (e.g., laser beam) gives rise to the measured response, can also be made, but methods customized to the sensor are more reliable.

Three methods of producing and using ground truth data have been presented and applied to electro-optical and range sensors primarily used for autonomous mobile robots. The methods all rely on ground truth and are dependent on the accuracy with which it is represented and registered with the test samples. By careful measurement of the positions and orientations of the sensors at the time samples are taken, a good match with the ground truth can be established and quantitative measures of performance for sensors and SP algorithms can be made.

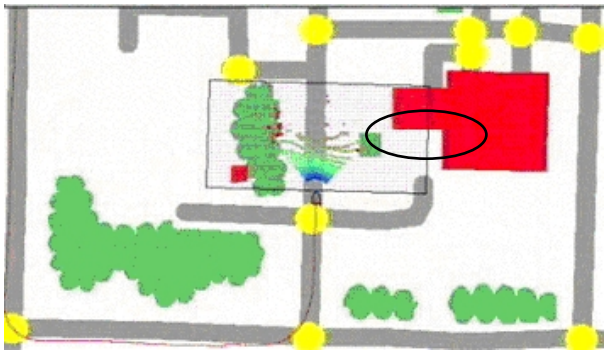


Figure 41. Feature map illustrating the *a priori* knowledge. The rectangle corresponds to the field of view of the LADAR sensor. Features included in the field of view are trees, roads, and part of a building.

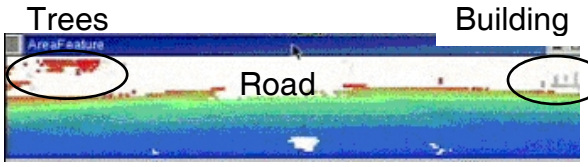


Figure 42. A LADAR image taken when the NIST HMMWV was at the location shown in Figure 41. The location in the real world is based on GPS and inertial information, which is also used to define the query sent to the *a priori* knowledge base.

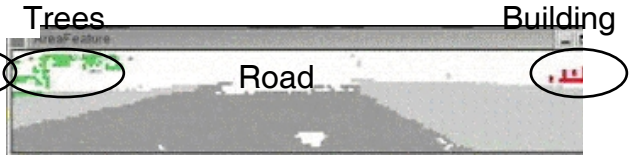


Figure 43. The predicted image generated by projecting the LADAR field of view into the *a priori* knowledge map. It is easy to label the LADAR image given the predictions in this image.

5.7. Conclusions

Most of the algorithms described in this chapter have related more to on-road driving than off-road driving, although our work is evenly divided between on-road and off-road applications. It has been found that driving on roads, particularly urban roads rather than highways, is significantly more challenging than driving cross country, although both present major problems for sensory processing. There are a number of rules that a vehicle driving on the road has to follow that complicate the sensory processing requirements and put constraints on the time available to carry out the sensory processing algorithms. These include the need to be aware of lane structure, road signs that modify vehicle behavior, pedestrians and other vehicles, and rules of the road that dictate when actions like turning are legal. All of these activities require real-time input from the control system and from sensors. Sensor input must be processed fast enough to ensure that the vehicle can operate at the same speeds as other traffic on the road. This is in contrast to off-road driving in which the main issues are determining traversability of the terrain, determining and maintaining the location of the vehicle in the world, and perhaps searching for features that are tactically important to a mission.

Current and future work is directed towards solving specific problems to enhance the capabilities of the sensory system in constrained environments. Two major thrusts are the emphasis on model-based methods and the incorporation of learning and adaptation into the algorithms. A third thrust is aimed at making the best use of multiple sensors, including newly developed range sensors such as those discussed in Chapter 7.

Models are being incorporated into the road detection and tracking algorithms. The models for the roads are derived from the NIST Road Network Database described in Chapter 4. The top-down information obtained from the road network is used to guide the bottom-up processing of sensor data. The left side of Figure 44 depicts the steps (low-level image processing, segmentation, classification) of a classical bottom-up approach for a recognition system. The segmentation step of the proposed system uses a trainable classifier to segment the image. The trainability allows the continuous update of the separation function between road and non-road areas in the image according to the current environment. After segmentation, another classifier is applied, which will recognize the type and structure of the road. In our case of a model-based approach, this classifier tries to fit models from a database of known road types to the segmentation results (in Figure 44 the road type "Straight Road" was recognized).

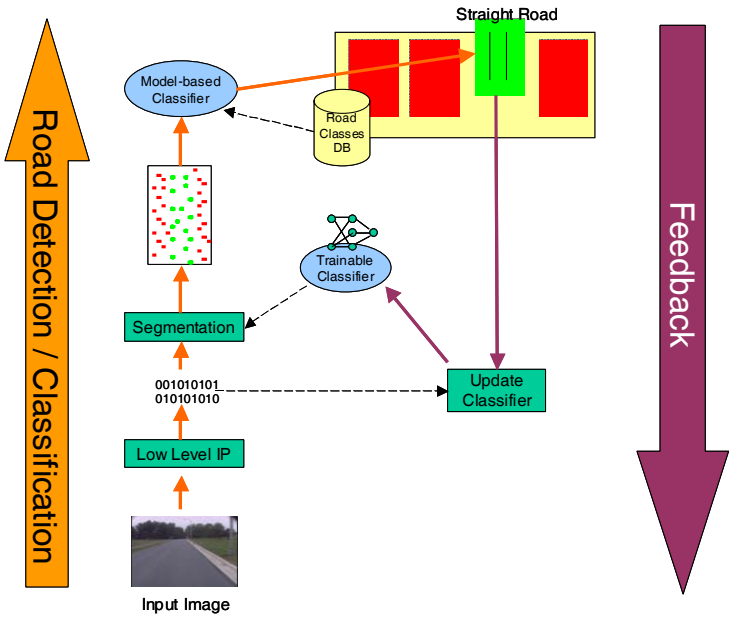


Figure 44. Model-based sensor processing.

The resulting structural and topological description of the current road will then be used to control the update of the trainable classifier used for segmentation. This feedback in terms of symbolic knowledge about the road structure allows an informed extraction of training samples, which describe road or non-road areas.

As was described in Section 5.5, learning is being applied to road following and off-road traversability analysis. In both cases, models are learned from experience and are then used to predict future appearance of similar features. Learning the traversability of regions makes use of both range and color information. As the vehicle traverses the terrain, more direct information is gathered to refine the traversability costs. This includes noting which regions are actually traversed and adjusting the traversability of the associated models. It also involves adjusting the traversability of regions where the vehicle's mechanical bumper is triggered, and where the wheels slip or the engine has to strain to move the vehicle.

Work on incorporating new sensors is focused on ways of using existing sensors in ways that emulate the capabilities expected to be available in the near to middle future. For example, high resolution sensors such as the Riegl described in Section 5.6 take a very long time to scan a scene, but provide data that is similar to that expected to be available from some of the new LADARs discussed in Chapter 7. Possible applications of such high resolution sensors are being explored using stop-action techniques in which a sequence of data sets is acquired from the slow sensor, and objects in the world are moved known amounts between each acquisition. This gives insight into the required resolutions and update rates for different types of environment. One environment that is being explored in detail using this approach is detection of pedestrians from a moving vehicle.

This chapter has provided an overview of some of the algorithms used by the SP module of the 4D/RCS architecture for the domain of autonomous mobile robots, and specifically for the Demo III program. All of the algorithms run in near real time, fast enough to update the WM several times a second. This is just a snapshot of the system. Over time, the SP hierarchy is being built up and the interactions between the SP and WM components of the architecture are being implemented.

References

[1] W. C. Stone, M. Juberts, N. Dagalakis, J. Stone, and J. Gorman, "Performance Analysis of Next-Generation LADAR for Manufacturing, Construction, and Mobility," National Institute of Standards and Technology, NISTIR 7117, May, 2004.

[2] T. Chang, T. Hong, S. Legowik, and M. Abrams, "Concealment and Obstacle Detection for Autonomous Driving," Proceedings of the Robotics & Applications Conference, Santa Barbara, CA, 1999, pp. 147-152.

[3] C. Rasmussen, "Combining Laser Range, Color, and Texture Cues for Autonomous Road Following," Proceedings of the IEEE International Conference on Robotics and Automation, 2002, pp. 4320-4325.

[4] M. J. Swain and D. H. Ballard, "Color Indexing," *International Journal of Computer Vision*, vol. 7, no. 1, 1991, pp. 11-32.

[5] A. Schmidt, "A Modular Neural Network Architecture with Additional Generalization Abilities for High Dimensional Input Vectors." Master's thesis, Manchester Metropolitan University, 1996.

[6] B. D. Ripley and N. L. Hjort, *Pattern Recognition and Neural Networks* New York, NY: Cambridge University Press , 1995.

[7] M. Foedisch and A. Takeuchi, "Adaptive Real-Time Road Detection Using Neural Networks," Proceedings of the International IEEE Conference on Intelligent Transportation Systems, 2004, pp 167-172.

[8] M. Foedisch and A. Takeuchi, "Adaptive Road Detection through Continuous Environment Learning," Proceedings of the Applied Imagery Pattern Recognition Workshop, 2004, pp 16-21.

[9] Ceryen Tan, Tsai Hong, Michael Shneier, and Tommy Chang, "Color Model-Based Real-Time Learning for Road Following," Proceedings of the IEEE Intelligent Transportation Systems Conference (Submitted), 2006.

[10] J. D. Crisman and C. Thorpe, "SCARF: A Color Vision System that Tracks Roads and Intersections," *IEEE Transactions on Robotics and Automation*, vol. 9, no. 1, Feb.1993, pp. 49-58.

[11] J. D. Crisman and C. Thorpe, "UNSCARF- A Color Vision System for the Detection of Unstructured Roads," Proceedings of the IEEE International Conference on Robotics and Automation, 1991, pp. 2496-2501.

[12] G. Piccioli, E. De Micheli, P. Parodi, and M. Campani, "A Robust Method for Road Sign Detection and Recognition," *Image and Vision Computing*, vol. 14, no. 3, 1996, pp. 209-223.

[13] L. Estevez and N. Kehtarnavaz, "A Real-Time Histographic Approach to Road Sign Recognition," Proceedings of the IEEE Southwest Symposium on Image Analysis and Interpretation, 1996, pp. 95-100.

[14] A. de la Escalera, L. e. Moreno, M. A. Salichs, and J. M. Armingol, "Road Traffic Sign Detection and Classification," *IEEE Transactions Industrial Electronics*, vol. 44, no. 6, 1997, pp. 848-859.

[15] A. L. Yuille, D. Snow, and M. Nitzberg, "Signfinder: Using Color to Detect, Localize and Identify Informational Signs," International Conference on Computer Vision (ICCV98), 1998, pp. 628-633.

[16] Chung-Lin Huang and Shih-Hung Hsu, "Road Sign Interpretation using Matching Pursuit Method," IEEE International Conference on Pattern Recognition (ICPR2000), 2000, pp. 1329-1333.

[17] Jun Miura, Kanda Tsuyoshi, and Yoshiaki Shirai, "An Active Vision System for Real-Time Traffic Sign Recognition," IEEE Conference on Intelligent Transportation Systems, 2000, pp. 52-57.

[18] Pavel Paclik and Jana Novovicova, "Road Sign Classification Without Color Information," Proceedings of the 6th Conference of Advanced School of Imaging and Computing, 2000, pp. 84-90.

[19] Lukas Sekanina and Jim Torresen, "Detection of Norwegian Speed Limit Signs," Proceedings of the 16th European Simulation Multiconference (ESM-2002), 2002, pp. 337-340.

[20] K Fleischer, H.-H. Nagel, and T. M. Rath, "3D-Model-Based-Vision for Innercity Driving Scenes," IEEE Intelligent Vehicles Symposium (IV'2002), 2002, pp. 477-482.

[21] Lubov N. Shaposhnikov, Alexander V. Podladchikova, Natalia Golovan, A. Shevtsova, Kunbin Hong, and Xiaohong Gao, "Road Sign Recognition by Single Positioning of Space-Variant Sensor Window," Proc. 15th International Conference on Vision Interface, 2002, pp. 213-217.

[22] Jung-Chieh Hsien and Shu-Yuan Chen, "Road Sign Detection and Recognition Using Markov Model," 14th Workshop on OOTA, 2003, pp. 529-536.

[23] C.-Y. Fang, S.-W. Chen, and C.-S. Fuh, "Road-Sign Detection and Tracking," *IEEE Transactions on Vehicular Technology*, vol. 52, no. 5, 2003, pp. 1329-1341.

[24] Lubov N. Shaposhnikov, Alexander V. Podladchikova, Natalia Golovan, A. Shevtsova, Kunbin Hong, and Xiaohong Gao, "Road SIgn Recognition by Single Positioning of Space-Variant Sensor Window," Proc. 15th International Conference on Vision Interface, 2002, pp. 213-217.

[25] M. Shneier, "Road Sign Detection and Recognition," SPIE Defense and Security Symposium, 2006.

[26] Shneier.M., T. Chang, T. Hong, and W. Shackleford, "Learning Traversability Models for Autonomous Mobile Vehicles," *Autonomous Robots (submitted)*, 2006,

[27] J. T. Spooner, M. Maggiore, R. Ordonez, and K. M. Passino, *Stable Adaptive Control and Estimation for Nonlinear Systems: Neural and Fuzzy Approximator Techniques* New York: John Wiley and Sons, 2002.

[28] K. M. Passino, *Biomimicry for Optimization, Control, and Automation* London: Springer-Verlag, 2005.

[29] M. Domjan, *The Principles of Learning and Behavior*, 4 ed. New York: Brooks/Cole Pub., 1998.

[30] T. R. Halliday and P. J. B. Slater, *Animal Behavior, Volume 1: Causes and Effects* New York: W. H. Freeman and Company, 1983.

[31] W. Schultz, P. Dayan, and P. R. Montague, "A Neural Substrate of Prediction and Reward," *Science*, vol. 275 1997, pp. 1593-1599.

[32] P. Gray, *Psychology*, 3 ed. New York: Worth Publishers, 1999.

[33] T. Ojala, M. Pietikcinen, and D. Harwood, "A Comparative Study of Texture Measures with Classification Based on Feature Distributions," *Pattern Recognition*, vol. 29 1996, pp. 51-59.

[34] Courtney, P. Benchmarking and Performance Evaluation . http://www-prima.inrialpes.fr/ECVNet/benchmarking.html . 3-25-1998.

[35] Faber, A. Quality Characteristics of Pattern Recognition Algorithms. http://www.dagm.de/DAGM/ag/wg.html . 1998.

[36] Lucas, S. IAPR TC-5 Benchmarking and Software. http://algoval.essex.ac.uk/tc5/Introduction.html . 2003.

[37] Bowyer K., C. Kranenburg, and S. Dougherty, "Edge Detector Evaluation Using Empirical ROC Curves," Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1999, pp. 354-359.

[38] T. B. Nguyen and D. Zhou, "Contextual and Non-Contextual Performance Evaluation of Edge Detectors," *Pattern Recognition Letters*, vol. 21 2000, pp. 805-816.

[39] L. Matthies, T. Litwin, K. Owens, K Murphy, D. Coombs, J. Gilsinn, T. Hong, S. Legowik, M. Nashman, and B. Yoshimi, "Performance Evaluation of UGV Obstacle Detection with CCD/FLIR Stereo Vision and LADAR," IEEE Workshop on Perception for Mobile Agents, 1998.

[40] J. A. Shufelt, "Performance Evaluation and Analysis of Monocular Building Extraction from Aerial Imagery," *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 21, no. 4, 1999, pp. 311-326.

[41] Wiedemannm C., Heipke, C., Mayer, M., and Jamet, O., "Empirical Evaluation of Automatically Extracted Road Axes," *Empirical Evaluation Techniques in Computer Vision* 1998, pp. 172-187.

[42] P. J. Phillips, H. Moon, S. A. Rizvu, and P. Rauss, "The FERET Evaluation Methodology for Face-Recognition Algorithms," *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 22, no. 10, 2000, pp. 1090-1104.

[43] K. Cho, P. Meer, and J. Cabrera, "Performance Assessment Through Bootstrap," *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 19, no. 11, 1997, pp. 1185-1198.

[44] P. Courtney, N. Thacker, and A. F. Clark, "Algorithmic Modelling for Performance Evaluation," *Machine Vision and Applications*, vol. 9, no. 5-6, 1997, pp. 219-228.

[45] N. Kiryati, H. Kالviainen, and S. Alaoutinen, "Randomized or Probabilistic Hough Transform: Unified Performance Evaluation," *Pattern Recognition Letters*, vol. 21, no. 13-14, 2000, pp. 1157-1164.

[46] R. Haralick, "Propagating Covariance In Computer Vision," Proceedings of the ECCV Workshop on Performance Characteristics of Vision Algorithms, 1996, pp. 1-12.

[47] E. E. Pissaloux, "Toward an Image Segmentation Benchmark for Evaluation of Vision Systems," *Journal of Electronic Imaging*, vol. 10, no. 1, 2001, pp. 203-212.

[48] J. Min, M. W. Powell, and Bowyer K., "Automated Performance Evaluation of Range Image Segmentation," Fifth IEEE Workshop on Applications of Computer Vision, 2000, pp. 163-168.

[49] S. C Coutre, M. W. Evens, and S. G. Armato II, "Performance Evaluation of Image Registration," Proceedings of the 22nd Annual EMBS International Conference, 2000, pp. 3140-3143.

[50] M. C. Shin, D. Goldgof, and K. W. Bowyer, "Objective Comparison Methodology of Edge Detection Algorithms Using a Structure From Motion Task," Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1998, pp. 190-195.

[51] H. Moon, R. Chellappa, and A. Rosenfeld, "Performance Analysis of a Simple Vehicle Detection Algorithm," *Image and Vision Computing*, vol. 20, no. 1, 2002, pp. 1-13.

[52] Larry Matthies, Todd Litwin, Ken Owens, Art Rankin, Karl Murphy, David Coombs, Jim Gilsinn, Tsai Hong, Steven Legowik, Marilyn Nashman, and Billibon Yoshimi, "Performance Evaluation of UGV Obstacle Detection with CCD/FLIR Stereo Vision and LADAR," 1998, pp. 658-670.

[53] M. O Shneier, T. Chang, T. Hong, and G. Scott H. Cheok, "A Repository of Sensor Data For Autonomous Driving Research," SPIE Aerosense Symposium, 2003, pp. 390-395.

Chapter 6
Temporal Registration of Sensed Range Images for Autonomous Navigation

R. Madhavan, T. Hong, and E. Messina
National Institute of Standards and Technology (NIST)
{*raj.madhavan*, *tsai.hong*, *elena.messina*} @*nist.gov*

1 Introduction

As discussed in earlier chapters, the 4D/RCS architecture specifies the simultaneous representation of information about entities and events in a hierarchical distributed knowledge database wherein information is presented in a form that is ideally suited for path planning and task decomposition. Maps are populated both with knowledge from *a priori* sources such as digital terrain databases, and with knowledge from sensors. The range and resolution of maps at different levels are specified to correspond to the range and resolution of planning algorithms. This limits the amount of computational power required to maintain maps and symbolic data structures with a latency that is acceptable for planning and reactive processes at each level.

Typically, the position estimation for Unmanned Ground Vehicles (UGVs) (for example, the Demo III program [37]; also see Chapter 10) relies on fusing Global Positioning System reported estimates with other on-board navigation sensors. The required accuracy of the GPS estimates cannot be guaranteed for the entirety of a particular mission as the direct line of sight to the satellites cannot be maintained at all times. GPS can be lost due to multipathing effects and terrain conditions, especially for on-road driving tasks in urban canyons or under tree cover. Sufficiently accurate vehicle positions are necessary to derive correct locations of sensed data towards accurate representations of the world model and for correctly executing planned trajectories and missions. In order to compensate for such unavailability and unreliability of GPS, another form of secondary position estimation becomes necessary. This chapter describes the development of registration algorithms to overcome this limitation.

The following reasons also warrant the need to develop robust registration algorithms:

- Within 4D/RCS, the use of *a priori* maps would enhance the scope of the world model. These maps may take a variety of forms including survey and aerial maps and may provide significant information about existing

topology and structures. In order to take advantage of this knowledge, these *a priori* maps need to be registered with sensor-centric maps [14]. Additionally, for incorporating *a priori* knowledge into the world model, some form of weighting is required and this depends on how well the *a priori* data and the sensed information are registered.

- There is also the need to generate higher resolution *a priori* terrain maps as the current survey maps are too coarse for off-road autonomous driving and also for maintaining up-to-date representations of the world even if the maps are of higher resolution.

- Another potential application for registering range images is the computation of *ground truth* as such registration is not dependent on time-based drift (unlike inertial navigation systems), vehicle maneuvers and terrain of travel. Such ground truth is necessary for evaluating performance of navigation algorithms and systems, as discussed in Chapter 9.

Active range sensing has become an integral part of any unmanned vehicle navigation system due its ability to produce unambiguous, direct, robust, and precise images consisting of range pixels, for example, using LADAR imagery. This is in direct contrast to passive sensing where the inference of range largely remains computationally intensive and not robust enough for use in natural outdoor environments. Depending on the speed of the vehicle, operating environment, and data rate, such range images acquired from a moving platform need to be registered to make efficient use of information contained in them for various navigation tasks including map-building, localization, obstacle avoidance, and control within the 4D/RCS architecture.

One of the following two approaches is commonly employed for matching range images to *a priori* maps [12]:

- *Feature-based Matching*: In this approach, two sets of features, $\mathcal{F}_i^1$ and $\mathcal{F}_j^2$, are extracted from two sets to be matched and then correspondences between features, $\mathcal{F}_{ik}^1$ and $\mathcal{F}_{jk}^2$, $k \in i, j$, that are globally consistent, are found. The displacement between the two sets can then be computed to deduce the sensor pose.

- *Pixel-based Matching*: This approach directly works on two sets of data points, $\mathcal{P}^1$ and $\mathcal{P}^2$, by minimizing a cost function of the form $\mathbf{F}(\mathbf{T}(\mathcal{P}^2), \mathcal{P}^1)$, where $\mathbf{T}(\mathcal{P}^2)$ is the second set of points subjected to a transformation $\mathbf{T}$. Any sensible cost is acceptable as long as its minimum corresponds to a *best* estimate of $\mathbf{T}$ in some sense. Usually, the minimization leads to an iterative gradient-like algorithm.

Lines and edges are two of the most widely used feature primitives. Matching between sensor observations and modeled features in a map have been considered as a search in an *Interpretation Tree* [9]. Drumheller extracts lines from sonar data and matches them against a room model to enable robot localization [6]. The complexity of the search problem is minimized by applying local constraints

(distances, angles and normal directions) to the set of possible pairings between observed and modeled features. The Hough transform is a shape detection technique which can be used to isolate features of a particular shape within an image or Time-of-Flight (TOF) sensor data. Schiele and Crowley extract line segments using the Hough transform from an occupancy grid and update the position of the robot using a Kalman filter [34]. Other researchers [7, 20] have combined odometric data and laser measurements using an Extended Kalman Filter (EKF) where the range weighted Hough transform is employed to extract lines from laser data. The resulting peaks are used as feature coordinates. Even though the Hough transform provides good results in indoor cluttered environments, it is restricted to operating in rectangular-shaped domains where no more than two predominant walls are present.

Pixel-based methods that attempt to minimize the discrepancies between sensed data and a model of the environment have been utilized for range registration. The attraction of these methods lies in the fact that the matching works directly on data points. Because the search is confined to small perturbations of the range images, it is computationally efficient. For example, Kweon et al. [19] compared elevation maps obtained from 3D range images to determine vehicle location. A similar point matching approach has also been adopted by Shaffer [35]. Cox [4] proposes a point matching method for an indoor robot named Blanche where scan-points from an optical rangefinder are matched to an *a priori* map composed of straight line segments. Blanche's position estimation system utilizes a robust matching algorithm which estimates the precision of the corresponding match/correction that is then optimally combined with odometric position to provide an improved estimate of robot position. Hoffman *et al.* employ a point matching algorithm for obtaining the inter-frame rotation and translation in a vision-based rover application [13]. Lu [21] finds corresponding points between two successive scans to compute the relative rotation and translation. An Iterative Dual Correspondence (IDC) algorithm is formulated based on closest point and matching range rules. Olson [30, 31] constructs a three-dimensional occupancy map of the terrain using stereo vision and iconically matches with a similar map to obtain the relative position between the maps enabling a mobile robot to perform self-localization.

The major drawback of the above approaches is that their use is limited to structured office or factory environments rather than unstructured natural environments. Straightforward correlation-based schemes (for e.g., see [41]), in general, are unable to handle outliers. As cross-correlation calculates the similarity, the two scans must be similar and thus this method cannot accommodate occlusions. This is easy to understand since if areas visible in one scan are not visible in another due to occlusion, then correlation of these scans may produce arbitrarily bad pose estimates. Also correlation usually places a high burden on computation especially when the scans are at different orientations.

In this chapter, algorithms for registering range images to overcome both the unavailability and unreliability of GPS within required accuracy bounds for UGV navigation are presented. Despite the apparent simplicity of the problem, to register range images from unmanned vehicles traversing unstructured

environments, the terrain of travel, sensor noise, and determination of accurate correspondences make it quite challenging. At the core of the registration process is a modified version of the well-known Iterative Closest Point (ICP) algorithm. These modifications render robustness to outliers, occlusions and false matches/spurious points. Results of the registration algorithm using real field data acquired from two different LADARs on the UGV are presented.

This chapter also evaluates the performance of the registration algorithm for position estimation of UGVs operating in unstructured environments. Field data obtained from two trials on UGVs traversing undulating outdoor terrain is used to quantify the performance of the algorithm in producing continual position estimates. Using the data obtained from the first trial, the iterative registration algorithm aids the position estimation process whenever GPS estimates are unavailable or are below required accuracy bounds. In the second trial, ICP is combined with a post-correspondence EKF to account for uncertainty present in the range images. For both the trials, the position estimates are then compared with those provided by ground truth to facilitate the performance evaluation of the registration algorithm. In addition, the importance of performance measures for assessing the quality of correspondences is described. These measures provide an indication of the quality of the correspondences thus making the registration algorithm more robust to outliers as spurious matches are not used in computing the incremental transformation. The registration algorithm is combined with one such performance measure and compared to the traditional ICP algorithm in terms of accuracy and speed.

Extensions to the ICP algorithm that make it possible to register range images obtained from a UGV to range images obtained from an Unmanned Aerial Vehicle (UAV) are then proposed. Registration of range data guarantees an estimate of the vehicle's position even when only one of the vehicles has GPS information. Temporal range registration enables position information to be continually maintained even when both vehicles can no longer maintain GPS contact. A Building Detection and Recognition (BDR) algorithm in urban environments using LADAR data is also presented. The motivation behind the development of the BDR algorithm is many fold:

World modeling: In 4D/RCS, the WM acts as a bridge between sensory processing and behavior generation by providing a central repository for storing sensory data in a unified representation [14]. Using the Knowledge Database (KD), it directly dictates behavior generation and in turn the level of intelligent planning that is achievable. Accordingly, it is necessary to have an underlying rich WM with a current and consistent KD which enables the UGV to analyze the past and plan for the future.

Registration across different sensing modalities: A continually updated and maintained WM will allow the sensors aboard the UGV to focus their attention on regions of future images where maximal useful information will be available. Complementary fusion and registration of information from different sensors offer distinct advantages over any one sensing modality.

Additionally, it is envisaged that the results of the BDR algorithm will be useful in the aforementioned air to ground registration scenarios.

This chapter is organized as follows: Section 2 details the ICP algorithm with suitable modifications for robustness. Section 3 presents the results of the proposed algorithm when applied for registering consecutive range images obtained from two LADARs mounted on a moving UGV. Section 4 evaluates the performance of the registration algorithm for position estimation and demonstrates the importance of performance measures with an example. Section 5 extends the modified ICP algorithm for air to ground registration by a hybrid feature-based registration approach. Section 6 describes the BDR algorithm and describes the results using LADAR range images obtained from a UGV traversing urban environments. Finally, Section 7 provides conclusions and describes further research work.

2 Temporal Registration Algorithm

The process of *registration* is termed as follows: *Given two sets of range images (model set:* $\mathbf{M}$ *and data set:* $\mathbf{D}$*): Find a transformation (rotation and translation) which when applied to* $\mathbf{D}$ *minimizes a distance measure between the two point sets.*

The goal can be stated more formally as:

$$min_{(\mathbf{R},\mathbf{T})} \sum_i ||\mathbf{M}_i - (\mathbf{R}\mathbf{D}_i + \mathbf{T})||^2 \quad (1)$$

where $\mathbf{R}$ is a rotation matrix, $\mathbf{T}$ is a translation vector and the subscript i refers to the corresponding points of the sets $\mathbf{M}$ and $\mathbf{D}$[1].

The ICP algorithm for registering 3D LADAR range images with suitable modifications is given in the next section. The registration algorithm is a modified variant of the well-known ICP algorithm [2]. At each iteration, the algorithm determines the closest match for each point and updates the estimated position based on a least-squares metric with some modifications to increase robustness. ICP and its variants have been widely used for registration purposes [33]. For autonomous vehicle navigation, ICP has been used for registration of range images for 3D terrain mapping and localization [17]. Other versions of the ICP algorithm have also been proposed for registration of range images in the presence of outliers [18, 26]. The modified algorithm is better suited for unstructured environments due to the robustness offered to outliers in the range data that is obtained from sensors aboard UGVs traversing undulating outdoor terrain.

2.1 Iterative Closest Point Algorithm

The ICP algorithm can be summarized as follows: Given an initial motion transformation between the two point sets, a pair of correspondences are developed

[1]Though it is not necessary that the model and data sets have the same number of points, after determining correspondences of data points with the model points, the number of model and data points used are the same. Hence only one index is used to denote both data sets.

between data points in one set and the next. For each point in the first data set, find the point in the second that is closest to it under the current transformation. It should be noted that correspondences between the two point sets is initially unknown and that point correspondences provided by sets of closest points are a reasonable approximation to the true point correspondence. From the pair of correspondences, an incremental motion can be computed facilitating further alignment of the data points in one set to the other. This find correspondence/compute motion process is iterated until a predetermined threshold termination condition.

In its simplest form, the ICP algorithm can be described by the following steps:

1. For each point in data set **D**, compute its closest point in data set **M**. In this chapter, this is accomplished via nearest point search from the set comprising $N_{\mathbf{D}}$ data and $N_{\mathbf{M}}$ model points.

2. Compute the incremental transformation (**R**,**T**) using Singular Value Decomposition (SVD) via correspondences obtained in step **1.**

3. Apply the incremental transformation from step **2.** to **D**.

4. If relative changes in **R** and **T** are less than a threshold, terminate. Else go to step **1.**

To deal with spurious points/false matches and to account for occlusions and outliers, the least-squares objective function in Equation (1) is modified and weighted such that [42]:

$$min_{(\mathbf{R},\mathbf{T})} \sum_i w_i \; ||\mathbf{M}_i - (\mathbf{R}\mathbf{D}_i + \mathbf{T})||^2 \tag{2}$$

If the Euclidean distance between a point x_i in one set and its closest point y_i in the other, denoted by $d_i \stackrel{\triangle}{=} d(x_i, y_i)$, is bigger than the maximum tolerable distance threshold D_{max}, then w_i is fixed to zero in Equation (2). This means that an x_i cannot be paired with a y_i since the distance between reasonable pairs cannot be very big. The value of D_{max} is set adaptively in a robust manner by analyzing distance statistics.

Let $\{x_i, y_i, d_i\}$ be the set of original points, the set of closest points and their distances, respectively. The mean and standard deviation of the distances are computed as:

$$\begin{aligned} \mu &= \frac{1}{N}\sum_{i=1}^{N} d_i \\ \sigma &= \sqrt{\frac{1}{N}\sum_{i=1}^{N} (d_i - \mu)^2} \end{aligned}$$

where N is the total number of pairs.

The pseudo-code for *Adaptive Thresholding* (AT) of the distance D_{max} is given below:

$$
\begin{array}{c}
\texttt{if } \mu < \mathcal{D} \\
\mathcal{D}^{itn}{}_{max} = \mu + 3\sigma\texttt{;} \\
\texttt{elseif } \mu < 3\mathcal{D} \\
\mathcal{D}^{itn}{}_{max} = \mu + 2\sigma\texttt{;} \\
\texttt{elseif } \mu < 6\mathcal{D} \\
\mathcal{D}^{itn}{}_{max} = \mu + \sigma\texttt{;} \\
\texttt{else } \mathcal{D}^{itn}{}_{max} = \epsilon\texttt{;}
\end{array}
$$

where itn denotes the iteration number. $\mathcal{D}$ is a user-defined parameter which indicates when the registration between the two data sets can be considered to be good. The pseudocode thus provides a procedure for statistically determining D_{max}. Accordingly, the modified algorithm is robust to relatively big motion between the two data sets and to outliers in the data.

During implementation, $\mathcal{D}$ was selected based on the following two observations:

1. If $\mathcal{D}$ is too small, then several iterations are required for the algorithm to converge and several good matches will be discarded, and

2. If $\mathcal{D}$ is too big, then the algorithm may not converge at all since many spurious matches will be included. The interested reader is referred to [42] for more details on the effect and selection of $\mathcal{D}$ and ϵ on the convergence of the algorithm.

At the end of this step, two corresponding point sets, $\mathbf{P_M}$:$\{\mathbf{p}_i\}$ and $\mathbf{P_D}$:$\{\mathbf{q}_i\}$ are available.

The incremental transformation (rotation and translation) of step **2.** is obtained as follows [1]:

- Calculate $\mathbf{H}$=$\sum_{i=1}^{N_D}(\mathbf{p}_i - \mathbf{p}_c)(\mathbf{q}_i - \mathbf{q}_c)^T$; ($\mathbf{p}_c$,$\mathbf{q}_c$) are the centroids of the point sets ($\mathbf{P_M}$,$\mathbf{P_D}$).

- Find the Singular Value Decomposition (SVD) of $\mathbf{H}$ such that $\mathbf{H} = \mathbf{U\Omega V}^T$ where $\mathbf{U}$ and $\mathbf{V}$ are unitary matrices whose columns are the singular vectors and $\mathbf{\Omega}$ is a diagonal matrix containing the singular values.

- The rotation matrix relating the two point sets is given by $\mathbf{R} = \mathbf{VU}^T$.

- The translation between the two point sets is given by $\mathbf{T} = \mathbf{q}_c - \mathbf{Rp}_c$.

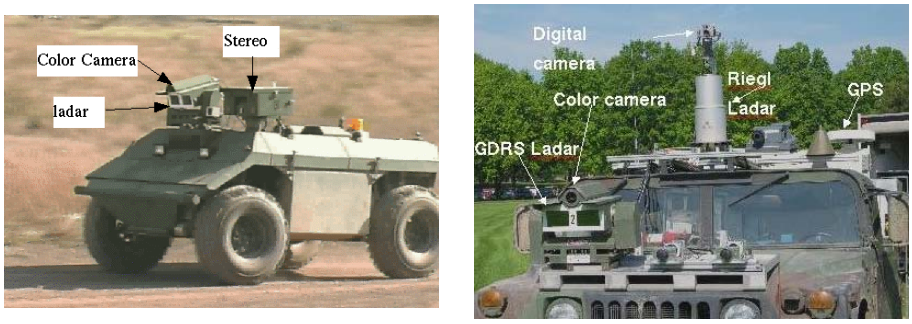
This process is iterated as stated in step **4.** until the mean Euclidean distance between the corresponding point sets $\mathbf{P_M}$ and $\mathbf{P_D}$ is less than or equal to a predetermined distance or until a given number of iterations is exceeded.

3 Temporal Registration of Sensed Range Images

In this section, the results of the temporal registration algorithm on two sets of LADAR data (henceforth referred to as UGVL1 and UGVL2) is presented.

3.1 Experimental Setup and Results

The experimental Unmanned Vehicle (XUV) shown in Figure 1(a) is a hydrostatic diesel, four-wheel-drive, four-wheel-steer vehicle. The military High Mobility Multipurpose Wheeled Vehicle (HMMWV) shown in Figure 1(b) is a one and one quarter ton, diesel-powered four-wheel-drive truck actuated with electric motors for steering, braking, throttle, transmission, transfer case, and park brake and sensors to monitor speed, engine RPM and temperature.



(a) (b)

Figure 1: The Demo III XUV shown in (a) can drive autonomously at speeds of up to 60 km/h on-road, 35 km/h off-road in daylight, and 15 km/h off-road at night or under inclement weather conditions. (b) shows the NIST HMMWV and its sensor suite.

Both vehicles utilize the 4D/RCS architecture using Neutral Message Language (NML) communications for autonomous navigation in unstructured and off-road driving conditions. The sensor suite of the XUV and the HMMWV consists of a pair of cameras for stereo vision, a Schwartz Electro-Optics LADAR, a stereo pair of Forward Looking Infra-Red (FLIR) cameras, a stereo pair of monochrome cameras, GPS, Inertial Navigation System (INS), a force bumper sensor and actuators for steering, braking and transmission. An integrated Kalman filter navigation system fuses observations from odometry, inertial and differential GPS sensors for position estimation.

UGVL1 (shown in Figure 1(a)) data was obtained during field trials as the XUV traversed rugged terrain with vegetation. The LADAR was mounted on this UGV on a pan/tilt platform to increase its narrow 20° field of view.

The range of the tilt motion is $\pm 30^\circ$ resulting in an effective field of view of about 90°. UGVL1 provides a range image of 32 lines $\times$ 180 pixels where each data point contains the distance to a target in the operating environment. The angular increment of this LADAR is $0.7^\circ \times 0.5^\circ$ in the horizontal and vertical directions, respectively. Utilizing knowledge about the LADAR mount position and calibration factors, the range information provided by the LADAR is transformed to cartesian coordinates.

UGVL2 (Riegl LADAR in Figure 1(b)) data was collected from a sensor mounted on the HMMWV as the vehicle traversed urban environments. The effective field of view is $80^\circ \times 330^\circ$ thus providing an almost panoramic view of the environment with an angular increment of 0.036°. The scan rate of UGVL2 is $1^\circ/\text{s} - 15^\circ/\text{s}$ providing 10000 pts/s with range up to 800 m thus making it much slower than that of UGVL1 but the resulting 3D range image is of a much higher resolution. For more details on the LADARs, see [36].

In the case of UGVL2, the 3D point cloud was acquired from two different view points whereas for UGVL1, the 3D point cloud represents scan points that were acquired between two consecutive vehicle locations. Additionally, for UGVL1, range image $\mathbf{D}$ was also translated 1 m along each of the (x,y,z) axes in addition to the translation and rotation that the image underwent due to the motion of the vehicle. It is important to note here that even though the range image points arrive in the same sequence for both the model and data sets, it is not guaranteed that both sets will have the same number of points as some facets of the LADAR data sets might return empty values.

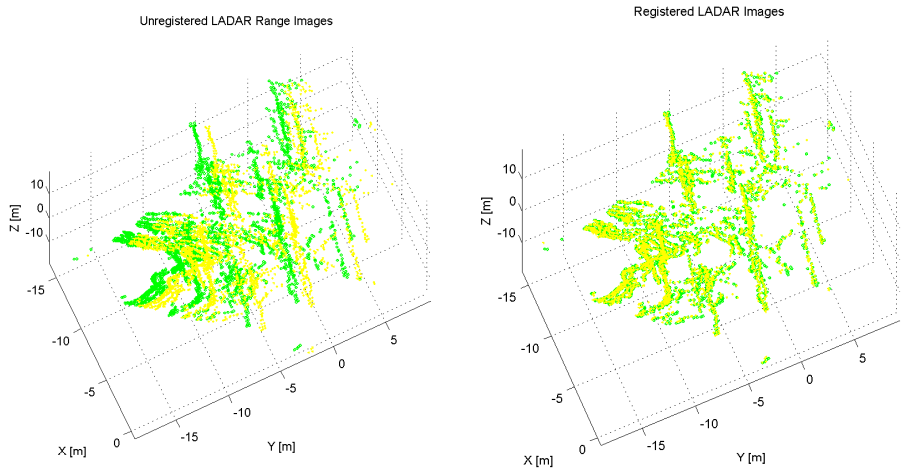
Figures 2(a)$-$(b) show the results when the modified ICP algorithm is used to register 3D range images obtained from UGVL1 and Figures 2(c)$-$(d) show that for UGVL2. The number of model ($\mathbf{M}$) and data ($\mathbf{D}$) points for the two LADARs are $\{2857, 2878\}$ and $\{125396, 123826\}$, respectively. As can be seen from Figure 2, the images are well registered. The closest point distances for UGVL1 before and after registration also prove that the images are sufficiently registered. The mean distances (m) after registration for the above three cases are $\{0.11, 0.84, 0.43\}$, respectively.

4 Performance Evaluation of Temporal Registration

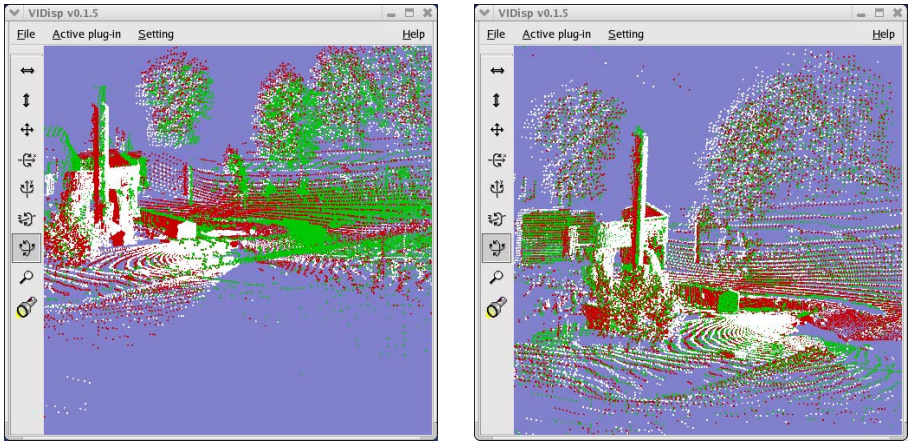
In this section, performance of the temporal registration algorithm (with adaptive thresholding) using two sets of field trials is evaluated.

4.1 Registration-aided Position Estimation

In this section, the position of an UGV operating in an unknown outdoor environment is estimated. The registration algorithm is used for aiding position



(a) (b)



(c) (d)

Figure 2: 3D LADAR range images before and after registration. (a) and (b) show the unregistered and registered UGVL1 range images, respectively. Here, the data range image (**D** denoted by '○' in yellow) was deliberately translated 1 m along the (x,y,z) axes in addition to the inherent translation to demonstrate the robustness of the registration algorithm. (c) and (d) show the unregistered and registered range images corresponding to two sets of UGVL2 range images where the model (**M**) range image is shown in green, unregistered and registered data (**D**) range images are shown in red and white, respectively.

estimation whenever GPS errors are above a predetermined threshold[2]. Whenever GPS position accuracy falls below the threshold, successive range images are registered with each other from the previous vehicle location (that is either available from dead-reckoning or the previous acceptable GPS estimate) to obtain the current vehicle location.

An EKF was used to fuse encoder, GPS and compass observations to arrive at ground truth position estimates. The EKF-based localization algorithm continually corrects the diverging dead-reckoning estimates based on external sensing information provided by GPS and compass corrections. Since the experiments were carried out in an outdoor environment with the UGV executing general motion (translation and rotation on all axes), sensor calibration is especially important to ensure accuracy of readings. For the encoder readings, external sensors (GPS and magnetic compass) were used to obtain calibration factors corresponding to the various axes. The correction factor for magnetic compass was obtained by looking up geodesic charts to determine the angle of magnetic variation corresponding to the longitude/latitude of the experiment's location. It should be noted here that the EKF pose estimate is always superior than that provided by GPS alone and thus has been considered as ground truth. Consequently, a better position fix is guaranteed even when GPS is subject to multipathing errors. The ground truth was obtained in a similar fashion as reported in [23].

Figure 3 shows the results of the position estimation using the registration algorithm. As mentioned earlier, registration of range images is used to aid position estimation when GPS reported positional errors exceed a given threshold. In Figure 3(a), the registration-aided position estimates are denoted by '+' and that of the GPS by '$\circ$'. The wheel encoder estimates are also shown by '$\times$' for comparison. The UGV is subject to slip and skid as a result of the undulatory nature of the terrain of travel. Accordingly, the errors in the wheel encoder estimates grow without bounds. The errors between the GPS and the registration-aided position estimates as compared with the ground truth are shown in Figure 3(b). The solid line represents the error in the registration-aided position estimates and that of the GPS estimate is shown in dashed-dotted line. It is evident that the registration-aided estimates are far superior than that of GPS alone.

4.2 2D Map-aided Position Estimation

A map-aided position estimation algorithm for computing accurate pose estimates for a UGV operating in tunnel-like environments is described in this section. A bearing-only laser was mounted on the roof of the vehicle so that it could detect strategically placed artificial landmarks (reflective stripes) in the trial environment. The exact position of the landmarks were made available from surveying using a digital theodolite. When the vehicle moves through the

[2]The error in the GPS positions reported were obtained as a function of the number of satellites acquired. As an alternative, the so-called *dilution of precision* measure associated with the GPS can be used for the same purpose [8].

environment, the presence of these landmarks is detected by using the observations from the laser. Thus, as the vehicle traverses the environment, a sequence of bearing observations to a number of fixed and known locations are made. Since the locations of these reflectors are known to the vehicle navigation system, the location of the vehicle can be computed from the bearing observations made. Utilizing bearing observations from a bearing-only laser in combination with dead-reckoning sensors (velocity and steering encoders and rate of change of orientation information from the inertial measurement unit), an EKF was employed to obtain ground truth [25]. Using the ground truth together with the information from a range and bearing scanning laser rangefinder, a map of the operating domain, represented by a polyline that adequately approximates the geometry of the environment, is obtained. The map building process relies on position estimation provided by artificial landmarks.

A combined ICP-EKF algorithm is used to match range images from a scanning laser rangefinder to the line segments of the polyline map [22]. For this application, ICP alone does not provide sufficiently reliable and accurate vehicle motion estimates. These shortcomings are overcome by combining the ICP with a post-correspondence EKF. Once correspondences are established, a post-correspondence EKF, with the aid of a nonlinear observation model, provides consistent vehicle pose estimates. The observation model relates line segments of the polyline map and range measurements provided by the laser rangefinder enabling the prediction of the range. Using this observation model, it was possible to discard ambiguous range measurements thus increasing the confidence in vehicle position estimates.

The ICP-EKF algorithm has several advantages. First, the uncertainty associated with observations is explicitly taken into account. Second, observations from a variety of different sensors can be easily combined as the changes are reflected only as additional observational states in the EKF. Third, the pixel-based algorithm does not require extraction and matching of features since it works directly on sensed data. Fourth, laser observations that do not correspond to any line segment of the polyline map are discarded during the EKF update stage thus making the algorithm robust to errors in the map.

The estimated vehicle positions (solid line) provided by the ICP-EKF algorithm along with the ground truth (denoted by $\circ$) is shown in Figure 4(a). The vehicle travels a distance of 150 m from right to left. The corresponding 2σ confidence bounds for the absolute error in x, y and ϕ are shown in Figure 4(b). It can be seen that the errors are bounded and thus the pose estimates are consistent. It is also clear that the estimated path is in close agreement with the ground truth.

4.3 Performance Measures

The correspondence determination process is the most challenging step of the iterative algorithm. Establishing reliable correspondences is extremely difficult as the UGV is subjected to heavy pitching and rolling motion characteristic of travel over undulating terrain. This is further exacerbated by the uncertainty of

the location of the sensor platform relative to the global frame of reference. In addition to these factors, noise inherently present in range images complicates the process of determining reliable correspondences.

One solution to overcome the above deficiencies is to extract naturally occurring view-invariant features, for example, corners, from range images. Such *ground control points* can then be used for establishing reliable registration with the ICP algorithm converging to the global minimum. The feature-based hybrid approach was shown to be effective in producing reliable registration for UGV navigation in rugged terrain and is described in Section 5.

For the map-aided position estimation scheme described in Section 4.2, the ICP-EKF algorithm failed to produce unambiguous correspondences with the map whenever variations in data sets were not unique. To enable ICP to produce accurate correspondences, a strategy to augment the ICP-EKF algorithm with artificial/natural landmarks was devised to provide external aiding. The proposed landmark augmentation methodology has been verified for the localization of a Load-Haul-Dump truck and resulted in the ICP-EKF algorithm producing reliable and consistent estimates [22].

The following measure towards performance evaluation of the registration algorithm for position estimation is proposed.

4.3.1 Mean Squared Error Measure

To indicate if the correspondences make sense the following measure is proposed:

$$\mathcal{P}_{mse} \quad = \quad \frac{1}{n}\sum_{i=1}^{n}\left[d\left(p_i, \ell_i\right)\right]^2$$

where p_i and ℓ_i are the i^{th} of n range data points and $d\left(p_i, \ell_i\right)$ is the distance from the p_i^{th} point to the ℓ_i^{th} point. Global minimum of the function will occur at the *true* pose of the vehicle.

At the true pose, all or at least the majority of the range data points will be close to their corresponding points, thus yielding a very low value for the correct solution. The problem with this measure is that it is difficult to decide if the pose is true in the presence of outliers and occlusions[3].

4.3.2 Results and Discussion

In this section, the combined utility of adaptive thresholding and the $\mathcal{P}_{mse}$ measure by using it to register 3D range images is illustrated. The registration results are then compared with direct ICP (i.e., without AT and $\mathcal{P}_{mse}$). For the

[3]Another measure that has been proposed is the Classification Factor ($\mathcal{P}_{cf}$). The problem with this measure is that it is not as sensitive as $\mathcal{P}_{mse}$ since it applies only for a certain local neighborhood. Thus $\mathcal{P}_{mse}$ can be used as a comparative performance measure whereas $\mathcal{P}_{cf}$ for pass/fail decisions for the correspondences before they are passed on for computing the incremental transformation. Additional performance measures have been developed and a detailed exposition is available from [24].

comparison, the same termination threshold condition is employed for both the algorithms.

Figure 5 summarizes the comparative results. Figures 5(a) and 5(b) show the registered LADAR images via the direct and combined ICP algorithms, respectively. The combined ICP needed 39 iterations whereas the direct ICP took 82 iterations[4] and the mean distances before and after registration were 0.07 m and 0.19 m for the two algorithms, respectively. Figures 5(c) and 5(d) show the closest point distance before and after registration. It is thus evident that the combined ICP algorithm is vastly superior to the direct ICP algorithm both in terms of accuracy and speed. Even though the $\mathcal{P}_{mse}$ metric is sensitive to outliers, the adaptive thresholding methodology in combination with the mean-squared error measure provides an acceptable means in inferring the validity of the position estimate.

5 Air to Ground Feature-based Registration

Another way to minimize the dependency on GPS for UGV navigation is to use aerial survey maps constructed using a downward-looking LADAR mounted on an Unmanned Aerial Vehicle (UAV). If the LADAR range images from the UGV can be registered to those from the UAV, then these results can serve as secondary position estimates in the event of absence or degradation of GPS.

In this section, a hybrid approach by combining the modified ICP algorithm with a feature-based method for registering two sets of LADAR range images is described. The approach is conceptually similar to Hebert *et al.* [11] who employ range imagery to compute vehicle displacement between two viewing positions by using a two-stage technique (feature matching followed by point matching). The advantage of this hybrid approach lies in the fact that the accuracy of the point matching technique is retained while keeping the computational burden under control as the feature-based method provides a good initial estimate for refinement.

The value of aerial imagery obtained via active range sensing for aiding ground vehicle navigation is being recognized within the UGV community. For example, in [40], aerial and ground views from unmanned vehicles are registered by extracting a geometrically consistent set of correspondences using surface signatures from which a registration transformation is estimated. It is not clear, given the computational burden associated with the extraction of surface signatures, whether this approach can be implemented in real-time. In [39], an aerial vehicle, a Flying Eye (FE), flies ahead of an UGV acting as a "scout" to detect difficult obstacles from an overhead perspective thus benefitting ground vehicle navigation. The above article briefly mentions the need for registering the data from the FE to the ground vehicle but the details of the registration process are not presented. The hybrid approach described in this chapter exploits the

[4]The difference in the complexity per iteration, for real-time implementation purposes, is almost negligible and thus for the direct and combined ICP algorithms, the complexity per iteration can be considered to be the same.

simplicity and speed of the iterative closest point algorithm thus lending itself to real-time implementation.

The underlying assumption in the iterative registration algorithm is that the rotation angle between the range images that need to be registered is not too large and also that these images are not too far apart. For the current case of UAV and UGV LADAR data, this assumption is overly restrictive and an aiding mechanism for the registration of the range images becomes necessary. Towards guaranteeing robust and accurate registration, the z translation value is first obtained by estimating the ground z (elevation) values on the UGV and UAV LADAR data in the vicinity of the UGV's current location. For the UGV, the ground values are obtained from the LADAR points that are within a given radius immediately in front of the vehicle and those for the UAV are obtained by finding the minimum of the LADAR values. Then the UAV and UGV LADAR data are projected into the base ground planes as depicted in Figure 6(a) and by using the Canny edge detector [3] the feature planes are constructed. The corner features are detected based on the intersections of lines formed by edges and are independently extracted from both LADAR data sets by considering those points that are above a given height from the ground as shown in Figure 6(b).

The two sets of the projected corner points (UAV LADAR set: $\mathbf{A}$ and UGV LADAR set: $\mathbf{G}$) are used to estimate a 2D translation. Given two sets of 2D corner points:

$$\mathbf{A} \triangleq \mathbf{a}_j = \begin{bmatrix} a_{1j} \\ a_{2j} \\ \vdots \\ a_{nj} \end{bmatrix};\ j = 1, 2,\ \cdots\ n; \quad \mathbf{G} \triangleq \mathbf{g}_k = \begin{bmatrix} g_{1k} \\ g_{2k} \\ \vdots \\ g_{nk} \end{bmatrix};\ k = 1, 2,\ \cdots\ n;$$

To find a translation along the x and y directions, the means of sets $\mathbf{A}$ and $\mathbf{G}$ are first computed using:

$$\bar{\mathbf{a}} = \frac{1}{n}\sum_{j=1}^{n} \mathbf{a}_j; \qquad \bar{\mathbf{g}} = \frac{1}{n}\sum_{k=1}^{n} \mathbf{g}_k;$$

The difference between the means of x, y and those between the aerial and ground z values provides a rough estimate of the required 3D translation between the two sets of LADAR data. The 3D translational offset when applied to the UGV range image enables the ICP algorithm to provide reliable registration results.

5.1 Experimental Setup and Results

The UAV LADAR produces a 3D range image at up to 6000 terrain pts/s within a 100 m scanning range [28, 27]. It provides an aerial survey map with significant information about existing topology and structures.

Figure 7 shows a top view of unregistered range images obtained from the UGV (in white) and UAV (in black) LADARs, respectively. Figure 8 depicts the results of the feature-based registration algorithm. Figure 8(a) shows a top view of the LADAR range images after applying the translation obtained using the corner features. Figure 8(c) shows the results of the iterative registration algorithm applied to the LADAR range images in Figure 8(a). Figures 8(b) and 8(d) show a magnified view of stages depicted in Figures 8(a) and 8(c), respectively. From Figures 7 and 8, it is evident that the LADAR range images are registered. For additional results, see [5].

6 Building Detection and Recognition

Several building detection algorithms are readily available for the detection and recognition of buildings from aerial LADAR/LIDAR (LIght Detection And Ranging) images [29, 32]. The adopted approach is to classify the data points according to whether they belong to the terrain, buildings or other object classes. The authors are not aware of algorithms that detect and recognize buildings for UGV navigation from ground-based LADAR data and believe that this work is the first of its kind.

The BDR algorithm consists of the following four main stages:

A1. First, ground detection is performed by using several small fixed areas of patches in front of the UGV to fit a plane for estimating and initializing the ground surface. Then, these 3D ground points are subtracted such that the points corresponding to objects above a certain height from the ground are available.

A2. Second, the projection distance to the ground plane is computed. Each range data point is projected to the ground surface which has a grid map representation of 10 cm resolution. The distance to the ground surface is stored in the grid data structure and is used to filter potential building segments.

A3. Third, an eight-connected component analysis on the projected grid map is used to group potential building segments.

A4. Finally, geometric properties are computed on each connected component and are used for building recognition.

6.1 Experimental Setup and Results

The data was obtained from the LADAR mounted on the HMMWV as shown in Figure 1(b) as the vehicle traversed urban environments. Figures 9(a) and (b) show a top-down view of the raw sensor data points before and after ground

subtraction[5]. Figure 9(c) (middle) shows the potential building segments. The intensity (top) and the panoramic color camera (bottom) images are also included in Figure 9(c) for comparison. Figure 9(d) depicts the 8 components resulting from the connected-component analysis. The final output of the algorithm after filtering small components is shown in Figure 9(e). The top figure shows a top-down view of the connected components projected onto the ground plane and the bottom figure shows that to the 3D point cloud.

In Figure 10, the left column shows the potential building segments, intensity, and color camera images, respectively, for three different sets of LADAR data. In the right column, the top and bottom figures show the top-down view of the connected components and their projection to the 3D point cloud, respectively, for the same sets of LADAR data. It is evident from Figures 9 and 10 that the buildings are reliably detected and recognized in the LADAR data.

Table 1 summarizes the false-positive rates of the BDR algorithm for four different LADAR data sets. Data sets #1 through #4 correspond to Figures 9(a)-(e), 10(a)-(b), 10(c)-(d), and 10(e)-(f), respectively. These data sets are representative of typical scenarios encountered in urban environments from which the buildings need to be detected. Whenever the BDR algorithm is unable to detect and recognize structures from the LADAR data as building or non-building, then that occurrence is deemed as a false-positive. For data sets #3 and #4, one of the buildings was not detected due to increased clutter in the environment.

Table 1: False-Positive rates of the building detection and recognition algorithm. Data sets #1 through #4 correspond to Figures 9(a)-(e), 10(a)-(b), 10(c)-(d), and 10(e)-(f), respectively.

Data Set	**No. of Buildings**	**No. of Other Objects**	**False Recognitions**
#1	8	0	0
#2	6	1	0
#3	4	5	1
#4	5	6	1

7 Conclusions and Further Research

Temporal registration of range images from unmanned ground and aerial vehicles was the main theme of this chapter. The need for such registration is motivated by several factors, the primary of which is the requirement to continually estimate the position of the unmanned vehicle within accuracy bounds

[5]The figures corresponding to the results of the BDR algorithm in this chapter are better viewed in color and are available from http://www.isd.mel.nist.gov/downloads/AIPR2004. The range images are shown in false color for better clarity (dark blue means no LADAR return).

dictated by a particular mission even when the GPS position estimates are unreliable or unavailable. By making suitable modifications to the ICP algorithm it was shown that the modified algorithm provides reliable and robust registration in rugged terrain and urban environments for registering successive range images obtained from two different LADARs on a UGV.

The evaluation of performance of the registration algorithm for position estimation of UGVs operating in unstructured environments was considered next. The temporal registration algorithm was used to aid the position estimation process and the resulting estimates were compared with ground truth to facilitate the performance evaluation for two sets of field data. Field data obtained from trials on UGVs traversing undulating outdoor terrain was used to quantify the performance of the algorithm in producing continual position estimates. In the first set of experimental results, registration-aided position estimates were generated whenever GPS estimates were unavailable or unreliable. For the second set of trials, the ICP-EKF algorithm was used for map-aided position estimation. In both cases, the presented results demonstrated the efficacy of the registration algorithm for position estimation. The importance of performance measures towards assessing the quality of correspondences required for accurate and efficient registration was detailed. The modified algorithm was combined with a mean-squared error measure to register 3D LADAR range images. The combined algorithm was then evaluated against the direct ICP algorithm. The accompanying results showed the superiority of the combined algorithm both in terms of speed and accuracy.

The temporal registration algorithm was then extended to register aerial images obtained from a UAV with those from the UGV. A hybrid approach was proposed to this end by combining the modified ICP algorithm with a feature-based method. The feature-based hybrid approach was also shown to be effective in producing reliable registration for UGV navigation. An algorithm for building detection and recognition from LADAR data was also presented in this chapter. The proposed BDR algorithm was tested on field data obtained from a UGV traversing urban environments and the resultant false-positive rates were found to be sufficiently reliable and efficient for use in temporal registration.

The results presented in the chapter demonstrated the potential of this approach lending itself to real-time implementation. For practical purposes, the sets of LADAR data utilized in this chapter can be assumed to be of the same resolution even though typically the aerial data tend to be of lower resolution than that of the UGV LADAR. To address this issue, schemes for use within the temporal registration algorithm that will inherently account for varying resolution in data sets that need to be registered are being developed. Towards this, corner detection schemes using the Harris [10] and SUSAN corner detectors [38] on the 3D projected base ground planes are being considered.

As ICP will only converge to the closest local minimum, the use of control points enabled to guarantee that this local minimum will correspond to the actual global minimum. If wrong convergence proves to be an issue in cases where control points cannot be established, stochastic optimization algorithms (e.g. **S**imulated **A**nnealing) can be used to alleviate this problem. SA is extremely

slow in converging to the global minimum and thus a hybrid algorithm that combines it with the proposed iterative algorithm would be more appropriate. As the convergence of the algorithm depends on an initial estimate, a sufficiently good initial estimate is required for superior registration. An initial estimate is almost always available for scenarios considered in this chapter as it can be obtained from either the vehicle's dead reckoning or GPS estimates.

Computing the correspondence is the most computationally expensive part of the algorithm. *kd-trees* have been proposed for faster correspondence where the complexity is reduced from $O(N_{\mathbf{D}}N_{\mathbf{M}}) \longrightarrow O(N_{\mathbf{D}}logN_{\mathbf{M}})$. Quaternions [16] (instead of SVD) have been considered to determine the 3D transformation but it results only in a slight improvement in the resultant registration for the tested field data.

In case of non-unique data sets (meaning less "structure" in the environment), extraction of naturally occurring control points was shown to be a good means of guaranteeing the convergence of the algorithm for reliable registration. In the field trials, the vehicle was driven at a top speed of 32 km/h for which case a good initial estimate is readily available and this immensely speeds up the correspondence determination. In the results reported in this chapter, reliable registration has been achieved in 10-12 iterations (at the most). This makes the proposed registration scheme very viable for real-time implementation.

The performance of the registration algorithm was evaluated for UGV navigation and measures to quantify the performance of the algorithm have been developed [24]. The quality of the 3D registration will significantly improve if the uncertainty of the LADAR range images are taken into account similar to the 2D case. Extension of these results to the 3D case are being investigated. To quantify the accuracy of the registration results, methods for estimating a covariance matrix of the error function that is minimized are being developed. The covariance matrix will be useful when fusing the position estimates obtained via registration with other sensors. At the time this chapter was written, the authors were in the process of obtaining LADAR data in areas where GPS accuracy degrades and then approaches its best estimate. Such data sets would be of immense value in evaluating the utility of the registration algorithm and performance measures.

Another interesting extension of the registration work described in this chapter is its use for a multi-robot team. When some robots do not have absolute positioning capabilities or when the quality of the observations from absolute positioning sensors deteriorate, another robot in the team with better positioning capability can assist other robots whose sensors have deteriorated or failed. This idea has been demonstrated for the localization of a team of robots where a distributed EKF-based cooperative localization scheme exploited heterogeneity of the available sensors is exploited in the absence or degradation of absolute sensors aboard the team members. The EKF-based localization algorithm was cast in a form such that the update stage of the EKF utilizes the relative pose information obtained using either a scanning laser range finder or a camera-based vision system. More details on the cooperative localization scheme can be found in [23].

In the BDR work described in this chapter, it has been assumed that the ground is relatively flat for ground detection. In scenarios where this assumption does not hold, the BDR algorithm may result in false-positives. However, in urban environments, the ground immediately in front of the UGV is almost always relatively flat. In cluttered environments, buildings can be wrongly grouped together with other objects thus increasing the false-positive rate of the algorithm. To counter this problem, the use of texture analysis using LADAR range values are being investigated. The presence of varying texture within a given range image is indicative of different classes of objects which can used for improving building detection and recognition. In addition, using color as an additional cue for feature detection is being considered.

The authors have not had an opportunity to subject the LADARs considered in this chapter to factors that might cause performance deterioration (for e.g. rain and fog). But these effects have been studied (on another LADAR) at NIST [15] in which the authors discuss the back scattering effect caused by fog that causes a spurious range value to be returned by the LADAR. There also exist research in the open literature on performance degradation of 2D LADAR under rainy conditions. For the LADAR used in the research reported in this chapter, these effects remain to be investigated.

References

[1] K. Arun, T. Huang, and S. Blostein. Least-Squares Fitting of Two 3-D Point Sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(5):698–700, 1987.

[2] P. Besl and N. McKay. A Method for Registration of 3-D Shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992.

[3] J.F. Canny. A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, November 1986.

[4] I. Cox. BLANCHE - An Experiment in Guidance and Navigation of an Autonomous Robot Vehicle. *IEEE Transactions on Robotics and Automation*, 7(2):193–204, 1991.

[5] A. Downs, R. Madhavan, and T. Hong. Registration of Range Data from Unmanned Aerial and Ground Vehicles. In *Proceedings of the Applied Imagery Pattern Recognition Workshop*, pages 45–50, October 2003.

[6] M. Drumheller. Mobile Robot Localization using Sonar. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(2):325–332, March 1987.

[7] J. Forsberg, U. Larsson, and A. Wernersson. Mobile Robot Navigation using the Range Weighted Hough Transform. *IEEE Robotics and Automation Magazine - Special Issue on Mobile Robots*, 21:18–26, March 1995.

[8] M.S. Grewal, L.R. Weill, and A.P. Andrews. *Global Positioning Systems, Inertial Navigation and Integration*. Wiley, 2001.

[9] W. Grimson and T. Lozano-Perez. Localizing Overlapping Parts by Searching the Interpretation Tree. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(4):469–482, July 1987.

[10] C. Harris and M. Stephens. A Combined Corner and Edge Detector. In *Proceedings of the Fourth Alvey Vision Conference*, pages 147–151, September 1988.

[11] M. Hebert, C. Caillas, E. Krotkov, I. Kweon, and T. Kanade. Terrain Mapping for a Roving Planetary Explorer. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 997–1002, 1989.

[12] M. Hebert, T. Kanade, and I. Kweon. 3-D Vision Techniques for Autonomous Vehicles. In *Proceedings of the NSF Range Image Understanding Workshop*, pages 273–337, 1988.

[13] B. Hoffman, E. Baumgartner, T. Huntsberger, and P. Schenker. Improved Rover State Estimation in Challenging Terrain. *Autonomous Robots*, 6(2):113–130, April 1999.

[14] T. Hong, S. Balakirsky, E. Messina, T. Chang, and M. Shneier. A Hierarchical World Model for an Autonomous Scout Vehicle. In *Proceedings of the SPIE International Symposium on Aerospace/Defense Sensing, Simulation, and Controls*, pages 343–354, April 2002.

[15] T. Hong, S. Legowik, and M. Nashman. Obstacle Detection and Mapping System. Technical Report NISTIR 6213, National Institute of Standards and Technology, Gaithersburg, MD 20899, U.S.A., August 1998.

[16] B.K.P. Horn. Closed Form Solution of Absolute Orientation using Unit Quaternions. *Journal of the Optical Society of America*, 4(4):629–642, April 1987.

[17] D. Huber, O. Carmichael, and M. Hebert. 3D Map Reconstruction from Range data. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 891–897, 2000.

[18] V. Koivunen, J. Vezien, and R. Bajcsy. Multiple Representation Approach to Geometric Model Construction from Range Data. Technical Report MS-CIS-93-66, GRASP Lab., University of Pennsylvania, 1993.

[19] I. Kweon and T. Kanade. High Resolution Terrain Map from Multiple Sensor Data. In *Proceedings of the IEEE International Workshop on Intelligent Robots and Systems*, pages 127–134, 1990.

[20] U. Larsson, J. Forsberg, and A. Wernersson. On Robot Navigation using Identical Landmarks: Integrating Measurements from a Time-of-Flight Laser. In *Proceedings of the IEEE Conference on Multisensor Fusion and Integration for Intelligent Systems*, pages 17–26, October 1994.

[21] F. Lu. *Shape Registration using Optimization for Mobile Robot Navigation*. PhD thesis, Dept. of Computer Science, University of Toronto, 1995.

[22] R. Madhavan and H. Durrant-Whyte. Terrain Aided Localization of Autonomous Ground Vehicles. *Special Issue of the Journal of Automation in Construction (Invited)*, 13(1):69–86, January 2004.

[23] R. Madhavan, K. Fregene, and L.E. Parker. Distributed Cooperative Outdoor Multi-robot Localization and Mapping. *Autonomous Robots: Special Issue on Analysis and Experiments in Distributed Multi-Robot Systems*, 17(1):23–39, July 2004.

[24] R. Madhavan and E. Messina. Performance Evaluation of Temporal Range Registration for Unmanned Vehicle Navigation. *Integrated Computer-Aided Engineering; Special Issue on Performance Metrics for Intelligent Systems (Invited)*, 12(3):291–303, 2005.

[25] R. Madhavan, E. Nettleton, E. Nebot, G. Dissanayake, J. Cunningham, H. Durrant-Whyte, P. Corke, and J. Roberts. Evaluation of Internal Navigation Sensor Suites for Underground Mining Vehicle Navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 999–1004, May 1999.

[26] T. Masuda and N. Yokoya. A Robust Method for Registration and Segmentation of Multiple Range Images. In *Proceedings of the Second IEEE CAD-based Vision Workshop*, pages 106–113, 1994.

[27] R. Miller, O. Amidi, and M. Delouis. Arctic Test Flights of the CMU Autonomous Helicopter. In *Proceedings of the Association for Unmanned Vehicle Systems International 26th Annual Symposium*, 1999.

[28] R. Miller and O. Amidi. 3-D Site Mapping with the CMU Autonomous Helicopter. In *Proceedings of the 5th International Conference on Intelligent Autonomous Systems*, June 1998.

[29] C. Nardinocchi, M. Scaioni, and G. Forlani. Building Extraction from LIDAR Data. In *Proc. of the IEEE/ISPRS Joint Workshop on Remote Sensing and Data Fusion over Urban Areas*, pages 79–83, 2001.

[30] C. Olson. Mobile Robot Self-localization by Iconic Matching of Range Maps. In *Proceedings of the 8th International Conference on Advanced Robotics*, pages 447–452, May 1997.

[31] C. Olson. Probabilistic Self-Localization for Mobile Robots. *IEEE Transactions on Robotics and Automation*, 16(1):55–66, February 2000.

[32] F. Rottensteiner. Automatic Generation of High-quality Building Models from Lidar Data. *IEEE Computer Graphics and Applications*, 23(6):42–50, November/December 2003.

[33] S. Rusinkiewicz and M. Levoy. Efficient Variants of the ICP Algorithm. In *Proceedings of the International Conference on 3-D Digital Imaging and Modeling*, pages 145–152, 2001.

[34] B. Schiele and J. Crowley. A Comparison of Position Estimation Techniques using Occupancy Grids. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1628–1634, May 1994.

[35] G. Shaffer. *Two-Dimensional Mapping of Expansive Unknown Areas*. PhD thesis, Carnegie Mellon University, 1992.

[36] M. Shneier, T. Chang, T. Hong, G. Cheok, H. Scott, S. Legowik, and A. Lytle. A Repository of Sensor Data for Autonomous Driving Research. In *Proceedings of the SPIE Unmanned Ground Vehicle Technology V*, April 2003.

[37] C. Shoemaker and J. Bornstein. The Demo III UGV Program: A Testbed for Autonomous Navigation Research. In *Proceedings of the IEEE ISIC/CIRA/ISAS Joint Conference*, pages 644–651, September 1998.

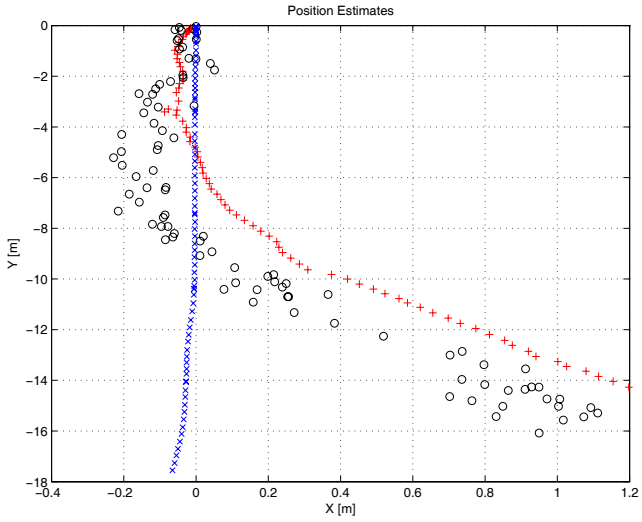
[38] S.M. Smith and J.M. Brady. SUSAN - A New Approach to Low Level Image Processing. *International Journal of Computer Vision*, pages 45–78, May 1997.

[39] A. Stentz et al. Real-Time, Multi-Perspective Perception for Unmanned Ground Vehicles. In *Proceedings of the AUVSI Unmanned Systems Conference*, July 2003.

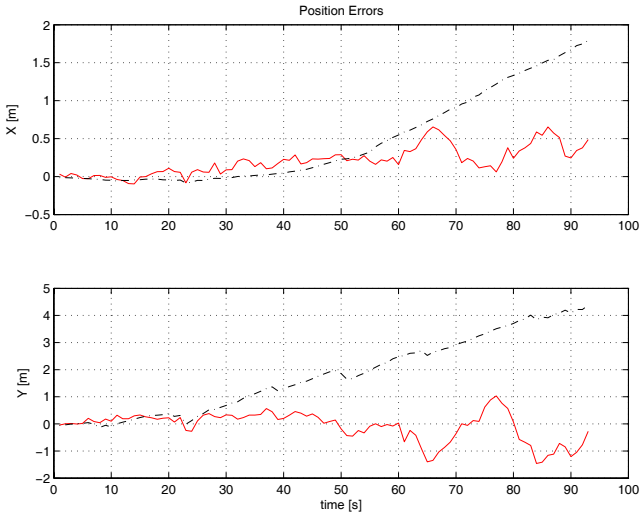
[40] N. Vandapel, R. Donamukkala, and M. Hebert. Experimental Results in Using Aerial LADAR Data for Mobile Robot Navigation. In *Proceedings of the International Conference on Field and Service Robotics*, July 2003.

[41] G. Weiβ, C. Wetzler, and E. von Puttkamer. Keeping Track of Position and Orientation of Moving Indoor Systems by Correlation of Range-finder Scans. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 595–601, September 1994.

[42] Z. Zhang. Iterative Point Matching for Registration of Free-Form Curves and Surfaces. *International Journal of Computer Vision*, 13(2):119–152, 1994.

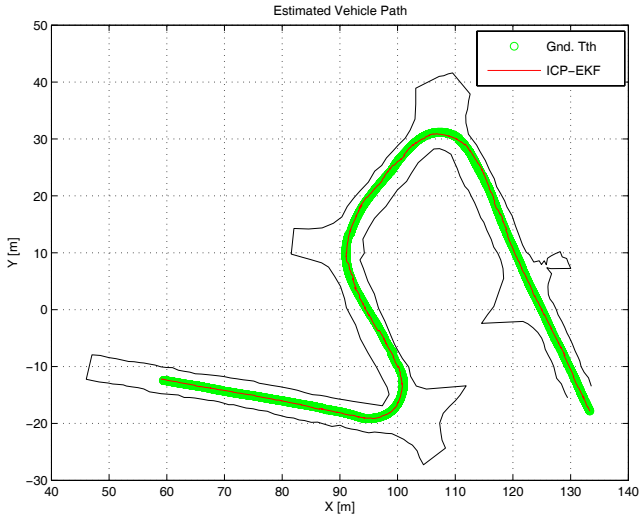


(a)

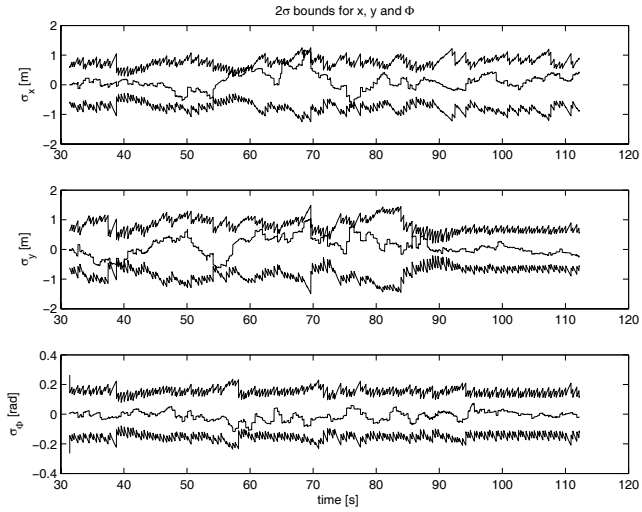


(b)

Figure 3: Registration-aided position estimation. The aided estimates are shown by '+' and that of GPS by '○'. The wheel encoder estimates shown by '×' are included for comparison in (a). In (b), position errors as compared to the ground truth is depicted; the solid line represents the error in the registration-aided position estimates and that of the GPS estimate is shown in dashed-dotted line.

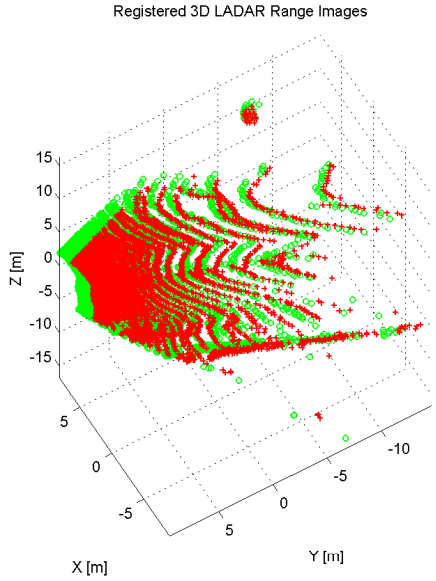


(a)

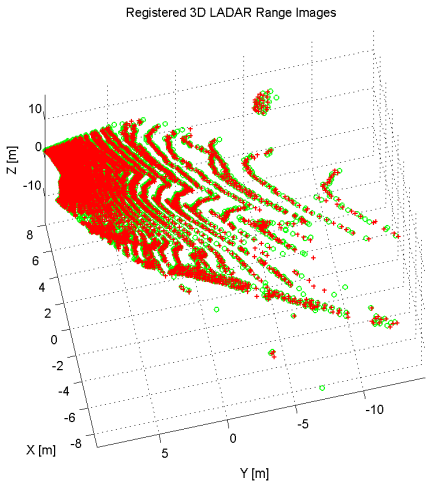


(b)

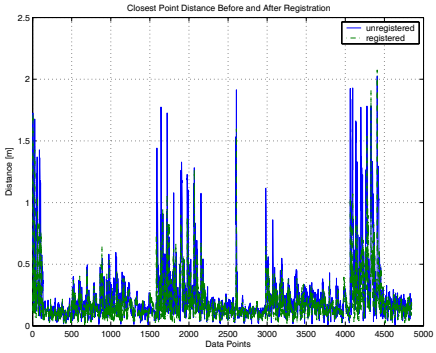
Figure 4: 2D map-aided position estimation. ICP-EKF estimated position the trial vehicle (solid line) and the ground truth (dotted line) are shown in (a). The 2σ confidence bounds are computed using the covariance estimate for the error in x, y and ϕ compared to the actual error computed with the ground truth estimates as depicted in (b).



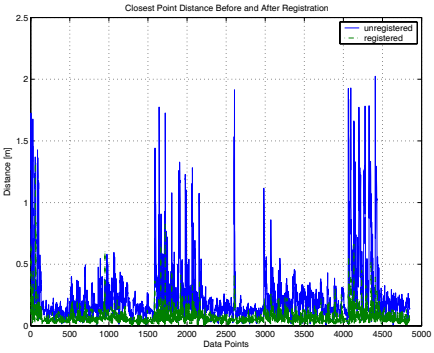
(a) Direct ICP



(b) Modified ICP with AT and $\mathcal{P}_{mse}$

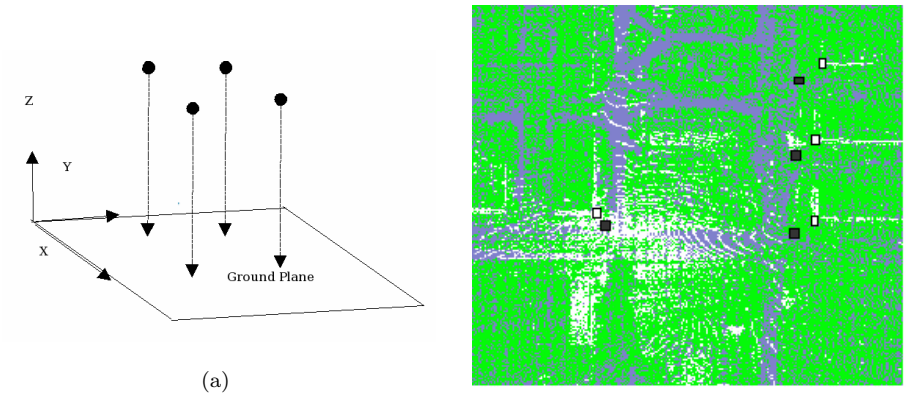


(c)



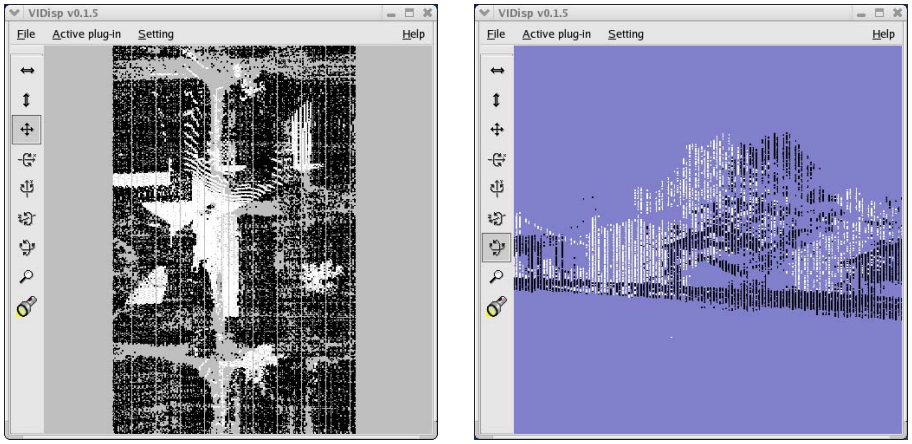
(d)

Figure 5: Illustration of 3D LADAR registration via the direct (w/o AT and $\mathcal{P}_{mse}$) and combined ICP algorithms. The model ('$\circ$') and data ('+') points before (a) and after (b) registration are shown. The registered (shown in dashed-dotted line) and unregistered (shown in solid line) closest point distances via the direct ICP (w/o AT and $\mathcal{P}_{mse}$) and the combined ICP algorithms, are shown in (c) and (d), respectively, corresponding to the registration of range images depicted in Figures (a) and (b).



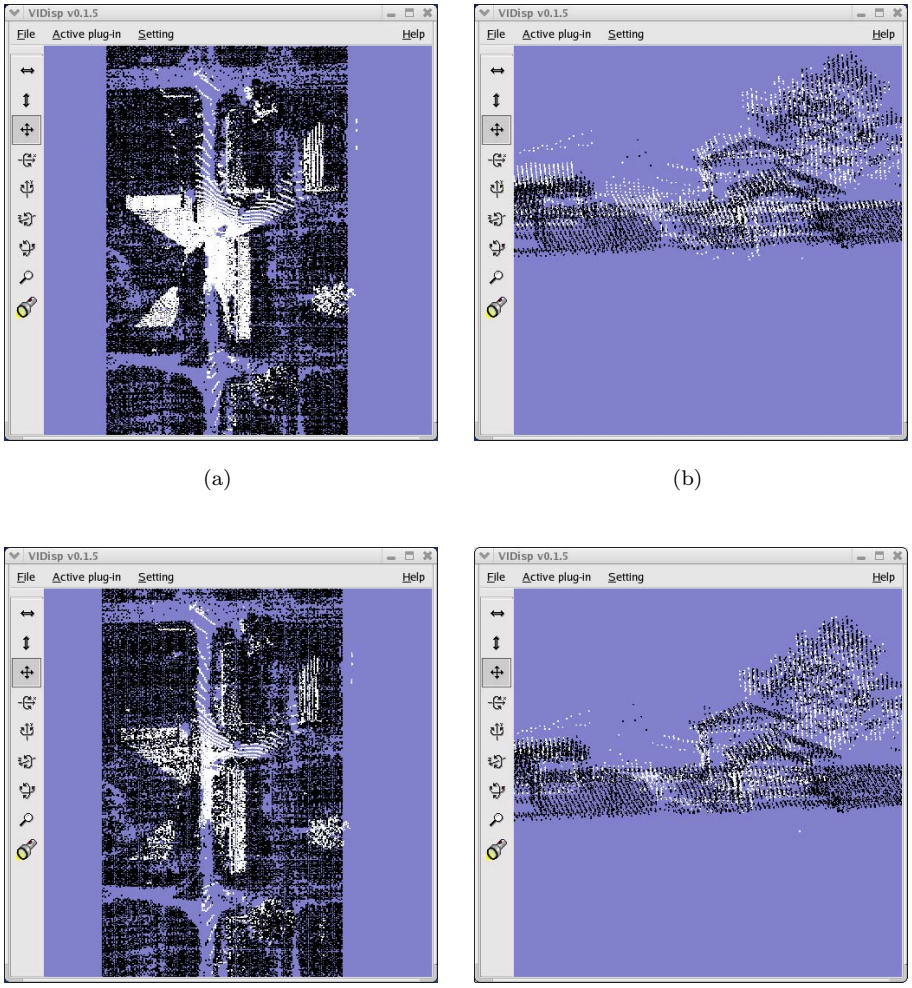
(a) (b)

Figure 6: Projection of LADAR data to base ground planes is shown in (a). The extracted features (corners) from the UGV (white) and UAV (black) LADARs are shown in (b) as white and black squares, respectively.



(a) (b)

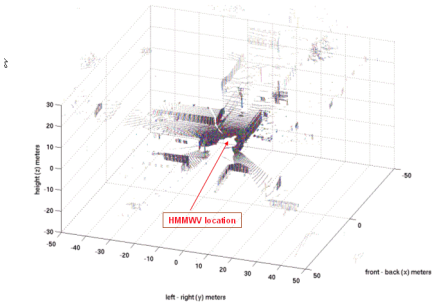
Figure 7: A top view of unregistered UAV (black) and UGV (white) LADAR range images is shown in (a). A magnified side view of (a) is shown in (b).



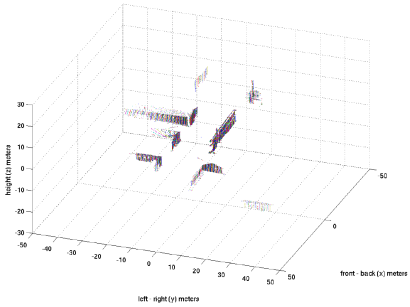
(a) (b)

(c) (d)

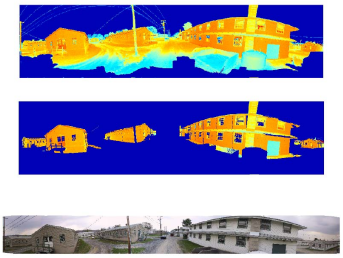
Figure 8: A top view of the feature-based translation obtained using the extracted corners is shown in (a) and a magnified side view of the same is shown in (b). (c) shows a top view of the registered UAV (black) and UGV (white) LADAR range images obtained by utilizing the feature-based translation results and (d) is a magnified view of (c). See text for further details.



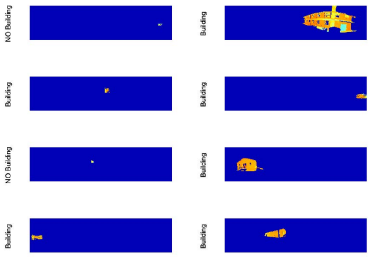
(a) 3D point cloud before ground subtraction



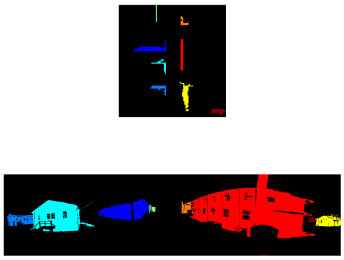
(b) 3D point cloud after ground subtraction



(c) Potential buildings, intensity, and color camera images

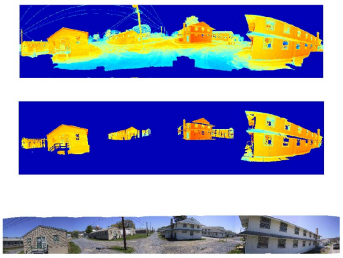


(d) 8 components

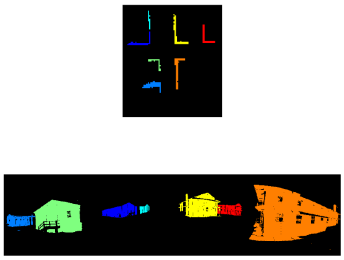


(e) Final output of the BDR algorithm

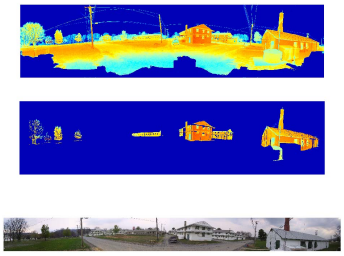
Figure 9: Results of the building detection and recognition algorithm.



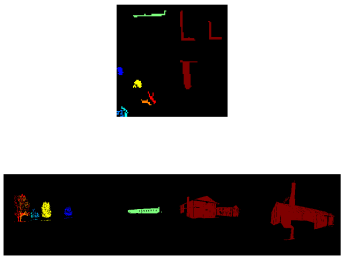
(a) Data set #2



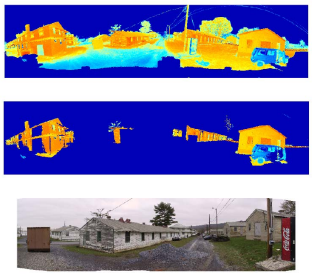
(b) Data set #2



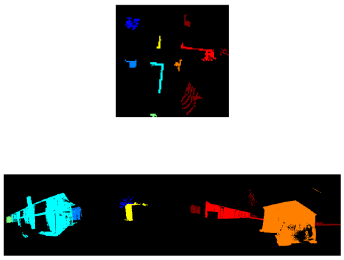
(c) Data set #3



(d) Data set #3



(e) Data set #4



(f) Data set #4

Figure 10: Building detection and recognition for three different sets of LADAR data.

Chapter 7

Advanced LADAR for Driving Unmanned Ground Vehicles

Maris Juberts, Anthony Barbera, and **Sandor Szabo**
National Institute of Standards and Technology (NIST)
{maris.juberts,tony.barbera,sandor.szabo}@nist.gov

1. Overview Of Needs And Capabilities Of LADAR for Autonomous Driving

The U.S. Department of Defense (DOD) has initiated plans for the deployment of autonomous robotic vehicles in various tactical operations expected to start in 2010. Several programs, including the Future Combat Systems (FCS) program, have received significant funding in order for this to take place. Envisioned are manned and autonomous Unmanned Ground Vehicles (UGVs) as well as manned and unmanned air vehicles performing cooperative tactical missions. Some of the tactical missions being considered for UGVs include: reconnaissance, active or passive surveillance, communication relay, mine detection or clearing, targeting, search and rescue, supply, terrain control/denial, forward observation, and lethal or non-lethal missions. These missions will require the vehicles to drive autonomously over open terrain and on roads which may contain traffic, obstacles, military personnel as well as pedestrians. UGVs must therefore be able to detect, recognize and track objects and terrain features in very cluttered environments.

As described in Chapter 1, safe driving at operational speeds requires the autonomous UGVs to perceive and model the 3D environment in real-time in order to support tactical reasoning, path planning, obstacle avoidance, and dynamic vehicle motion control. There are several sensor modalities that can directly provide 3D range imaging of a scene, including sonar, radar, structured light, and LADAR. As pointed out, sonar and radar have poor angular spatial resolution at most ranges, and sonar is slow and structured light is effective only at short ranges. Only LADAR has demonstrated the ability to provide real-time 3D range images, reliably and with sufficient resolution, that can be used to model the 3D environment within about 50 m of the sensor for off road autonomous driving. One of the advantages of LADAR is that data segmentation and grouping, needed for distinguishing objects from background data, can be performed much more easily and robustly in the 3D space provided by LADAR point clouds. To achieve autonomous driving performance in normal traffic at operational speeds requires that LADAR provides real-time images at ranges out to more than 100 m and with spatial resolution approaching that of human perception. This technology has evolved and is close to delivering these requirements. What is needed is a well funded development effort to build reliable, high performance LADARs at reasonable cost.

Although several LADAR sensors exist at the time of this writing which have successfully been implemented and demonstrated to provide somewhat reliable obstacle detection and can be used for path planning and path selection, they tend to be limited in performance (primarily resolution and maximum range), are affected by obscurants (dust, fog, grass, foliage), and are quite large and expensive. One such mobility LADAR is reported in Appendix A of a NIST publication[1]. An example of where the available UGV LADAR technology falls short is shown in Figure 1. Although the flatness of the road is visible in the display of the 3D range image of the road the vehicle is following, it has very limited range information from the road and terrain further ahead in the scene. In addition, although the trees and brush along the sides of the road are seen as obstructions, it would be very hard to use that information for classifying or identifying what those obstructions are. Limitations in available UGV LADAR spatial resolution (about 0.25^o per pixel) are also evident in Figure 2. It is a range image taken with a Riegl LMS Z420 LADAR, set at a resolution of 0.2^o per pixel, of two persons standing at a distance of 100 m from the camera. Even though target returns are obtained from that distance, there are only about 4 pixel hits on each person. This is insufficient for recognizing that the returns are from a person. Another example of available technology limitations is shown in Figure 3. The figure displays the range image of a scene taken in a parking lot. It is very hard to identify any detected known objects in the scene. In the future, it is

expected that unmanned vehicles must be able to locate available parking spaces on their own and to be able to park autonomously.

Figure 1. (left): Digital camera image of road and trees; (right) the same scene as viewed by Demo III real-time LADAR. Where does the technology fall short? The vehicle is effectively myopic. LADAR is low resolution (75 mm to 150 mm range accuracy) and short range (< 40 m returns from horizontal surfaces). This LADAR does not penetrate dust and smoke. Planning is not optimized for road following.

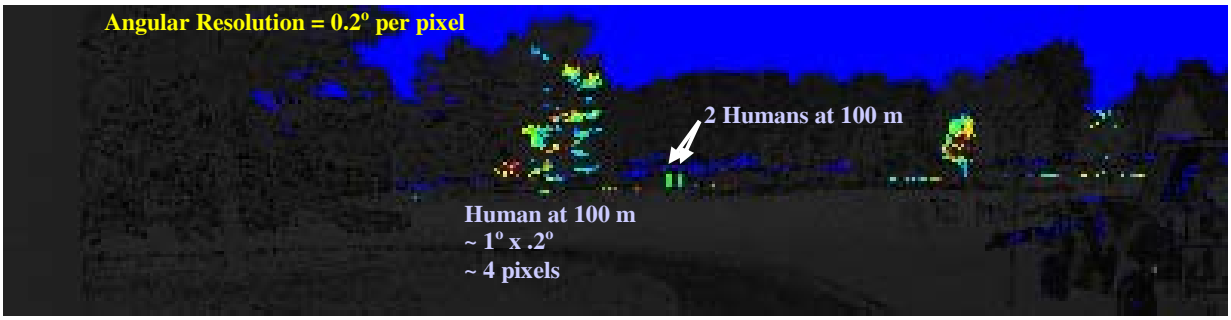


Figure 2. Detection of two human targets at 100 m. Range image taken with Riegl LMS Z420 LADAR scanner set to an angular resolution of 0.2° per pixel.

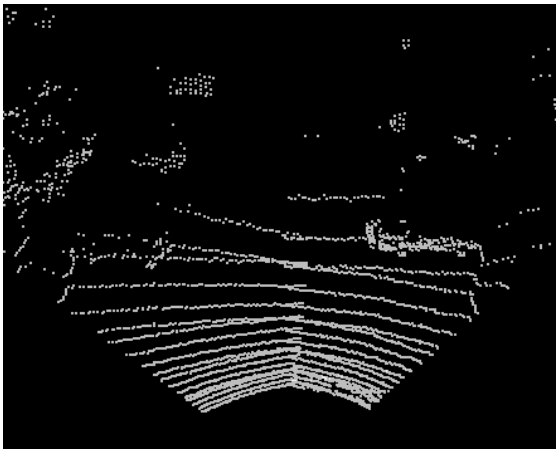


Figure 3. Typical point cloud of parking lot produced by a coarse UGV LADAR.

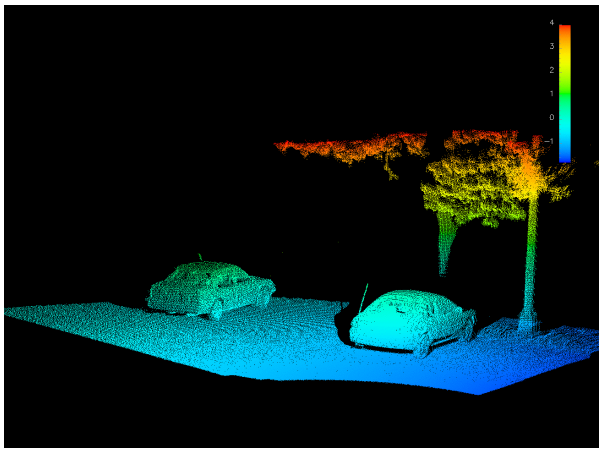


Figure 4. High resolution point cloud image of parking lot. Image is color coded in vertical (z) dimension.

What denser, high resolution range data can provide is clearly evident in the range image taken of a parking lot with a high resolution LADAR camera as shown in Figure 4. Detection and identification of cars, tree trunks, overhanging branches, man-made structures (light pole) and even curbs are highly possible from this kind of data. It must be pointed out, however, that the data was taken with a high resolution scanning LADAR that required several minutes to generate the range image. In addition, even though considerable effort and funding has been provided by the DOD R&D community, nearly all of the development has been for target detection (ATR) and tracking from various flying platforms. Participation in the Army and DARPA sponsored UGV programs has helped NIST to identify requirement specifications for LADAR to be used for on and off-road autonomous driving. This chapter describes the expected requirements for advanced LADAR for driving Unmanned Ground Vehicles and presents an overview of proposed LADAR design concepts and a status report on developments in next generation scannerless LADAR and advanced scanning LADAR which may be able to achieve the stated requirements. Although the primary use of LADAR has been in the military robotics programs, LADAR is also beginning to show up in other industries as well. This chapter also briefly describes the needs and applications of LADAR sensors in the transportation industry and the industrial material handling industry. Examples of real-time range images taken with existing LADAR prototypes are presented.

The Department of Transportation's Automated Highway Systems program in the mid '90s was completed in 1997 with a successful demonstration of autonomous highway vehicles in San Diego, California[2]. However, interest in highway automation by the Government and the transportation industry, since that time, has taken a back seat to improvements in highway safety. Since traffic fatalities in the U.S. have remained steady at around 42,000 per year, the transportation industry and the Government are looking at advanced technology to improve vehicle safety and reduce traffic accidents. In particular, the DOT has been looking at on-vehicle sensor-based driver warning systems for rear-end and road departure crash prevention. In 2005, the DOT initiated a new program to fund the development and to evaluate the performance of an Integrated Vehicle Based Safety System (IVBSS) that provides warnings in three crash scenarios: rear-end, road departure and lane change. The IVBSS will most likely use a collection of video cameras and range sensors to determine potential crash conditions.

Figure 5 illustrates the ways in which LADARs might be employed in automotive safety applications. A vehicle mounted LADAR is expected to provide the following measurement attributes:

1. Determine location of obstacles (range and azimuth)
2. Determine size of obstacles
3. Determine range rates to obstacles
4. Perform measurements at maximum highway speeds (120 km/h)
5. Detect obstacles at far range (> 65 m)
6. Sufficient Field Of View (FOV) to view and detect obstacles in path of vehicle
    - rear-end sensing – FOV must cover road in front to measure curvature of road
    - road-departure sensing – FOV must cover shoulder in forward direction of vehicle and take into account the curvature of the road and obstacles directly to the side (e.g., jersey barriers)
    - lane-change sensing – FOV must cover adjacent lane directly to the side of the vehicle and possibly to the rear to detect passing vehicles

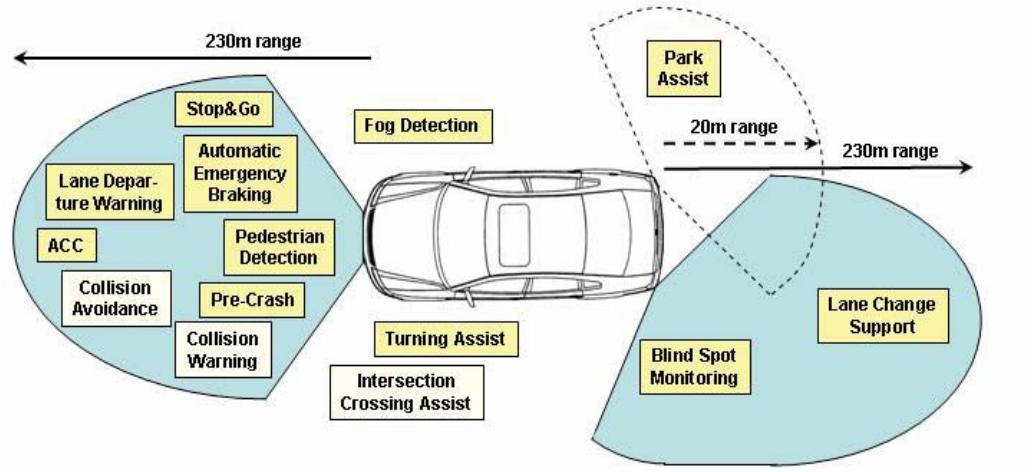


Figure 5. LADARs for automotive safety and driver assist applications (figure from IBEO Automotive Sensor GmbH[14]).

The material handling industry is also considering using advanced LADAR sensors for Automatic Guided Vehicle (AGV) autonomous navigation, non-contact safety systems and for automation of material handling[3]. One way of increasing cost efficiency in the manufacturing and material handling industries is to increase the speeds that AGVs operate at. Vehicle speeds for indoor operation as high as 2 m/s are being targeted. This however increases the possibility of accidental collisions with personnel and other stationery or mobile equipment. The most widely used AGV safety systems use contact bumpers, however, this may not be sufficient to prevent injury or damage at these higher operating speeds. Another approach which is gaining acceptance is to use 2D LADAR line scanners as non-contact safety bumpers and for 3D image generation. This improves the safety bumper approach, however the vehicle and sensor have to be moving in order to generate a 3D scene. Other real-time, non-contact imaging devices are being looked at to fill the need for improving obstacle detection and avoidance, real-time path planning, and mapping functions. A good possible solution to this need are low cost, compact, solid-state Focal Plane Array 3D LADAR imagers. These are now becoming available in the U.S. and in Europe. These cameras can provide the AGV with real-time/dynamic (at frame rates of 10 Hz or higher) 3D range images for world modeling of the operational environment. An example of a single frame 3D range image, taken with a CSEM SwissRanger II, camera is shown in Figure 6. Actually this 3D scene is a result of combining range image data from two side-by-side cameras which produce an effective wide horizontal Field of View (FOV) of close to 90^o. The vertical FOV is close to 45^o. By placing additional cameras around the vehicle, a 360^o ring of perception in 3D around the vehicle can be achieved at close to video frame rates. This technology could open up new application for recognition of personnel, materials and for automation of material handling and assembly.

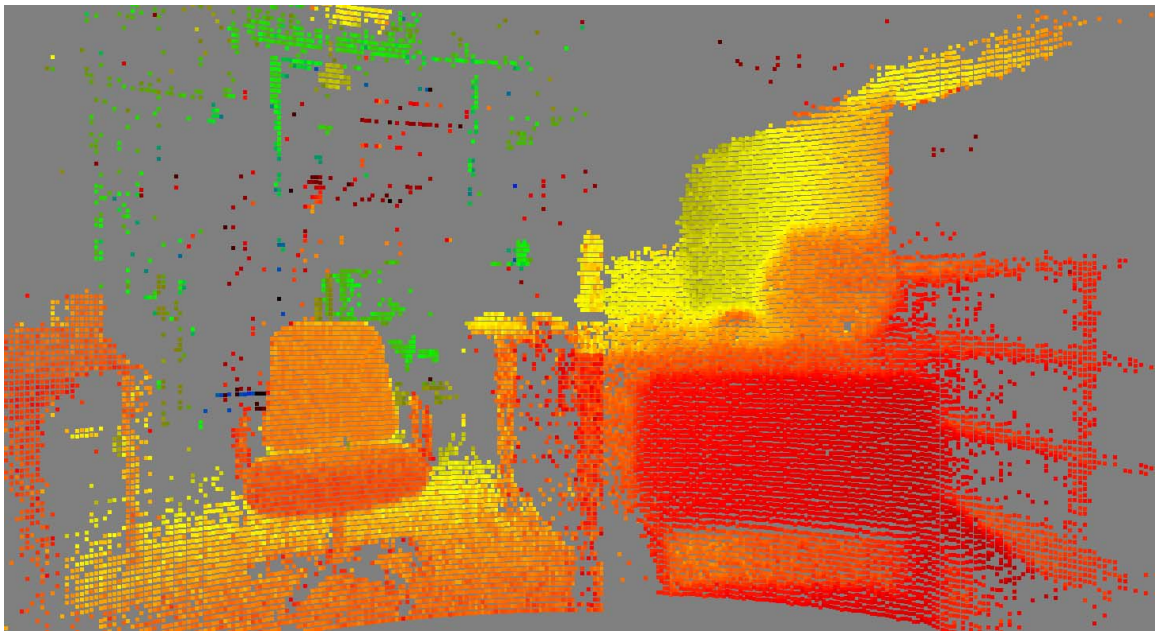


Figure 6. Image taken with two side-by-side solid state CSEM – SwissRanger 2 cameras[13].

This chapter is organized as follows:

Section 2 discusses expected requirements for LADARs to be used for on and off-road autonomous driving. Section 3 describes example potential solutions to the requirements that were offered by four winners of a NIST Phase I Broad Agency Announcement (BAA) contract to develop LADAR concept designs. Section 4 provides examples of existing real-time advanced LADAR prototypes available from other developers. Section 5 discusses the steps being taken to further identify/refine LADAR sensor requirements for FCS UGV missions and recommends what steps are needed to meet these requirements.

2. Expected Sensor Requirements

2.1 Earlier Baseline Requirements

Even though considerable effort and funding for advanced LADAR development has been provided by the DOD R&D community, nearly all of the development has been for Automatic Target Recognition (ATR) and tracking from various flying platforms. This includes LADAR technology development which allows for foliage penetration, permitting detection of targets hidden under trees and camouflage netting. Although much of the development has contributed significantly toward furthering the performance of LADAR sensors, it has not addressed the needs for autonomous driving with unmanned ground vehicles. LADAR sensors for ground vehicles have their own particular requirements. This includes having a very broad dynamic range. This is the ability of the sensor to detect and recognize obstacles/objects/terrain features which are at very close range (< 1 m) and at more than 100 m, all in a single frame of data.

Participation in the Army Demo III program helped NIST to identify requirement specifications for LADARs to be used for on and off-road autonomous driving. At that time, NIST envisioned the need for two types of LADAR range imaging sensors for this type of application - one having a wide FOV (40° x 90°) with a resolution of about 0.25° or better per pixel, and the second a foveal LADAR having a narrow FOV of approximately $1/10^{th}$ of the wide FOV with a resolution of about 0.05° or better per pixel. The intent was to make the foveal LADAR quickly steerable to points-of-interest positions within the wide peripheral angle FOV LADAR at a rate of at least 3 saccades (point-to-point moves) per second. Both types of LADAR sensors were expected to have a resolution of about 5 cm or better in range, and be able to detect the ground plane out to a distance of better than 50 m and vertical surfaces out to a range of at least 100 m. Frame rates of higher than 10 Hz were required. Both types of LADAR were expected to be eye safe and be provided with the capability of penetrating dust, fog, grass and light foliage (either by sensing multiple returns or looking for the last return), and be able to operate in full sunlight conditions. Small size and low cost were also emphasized as important requirements.

The initial NIST Broad Agency Announcement (BAA), which contained details on the expected requirements, was released in June of 2002 and is included in the NIST publication[1] as Appendix B. There was a good industry response to the announcement with 15 proposals being submitted. A unanimous decision was made by the proposal reviewers to make four awards for Phase I at the end of September 2002. The four awards went to (listed in alphabetical order):

- Advanced Scientific Concepts Inc., Santa Barbara, California
- Coherent Technology Inc., Lafayette, Colorado
- Lockheed Martin Missiles and Fire Control, Dallas, Texas
- Raytheon Missile Systems, Tucson, Arizona

Synopses of some of the main features of each design were prepared by each contractor for inclusion in the NIST report[1]. These are provided in Section 3 of this chapter. Because of the proprietary nature of the designs, however, the contractors desired to keep much of the proposed design information confidential. Although sponsorship and funding for the next phase of development never materialized, the need of advanced LADARs for use on UGVs still exists. Over the last few years NIST received numerous requests for information pertaining to the status of the original BAA and about the progress in advanced LADAR development in general.

2.2 Requirements Based on Analysis of Autonomous On-road Driving

The DARPA Mobile Autonomous Robot Software (MARS) On-Road Driving Project, described in Chapters 2 and 4, funded NIST to do a task analysis of autonomous on-road driving. If one looks at an on-road driving task, it is very difficult to identify the sensors and the processing required because different driving tasks have significantly different resolution, classification, and identification requirements. For the task of the vehicle driving down a road, the sensor system has to be able to identify large objects moving nearby, their direction, speed and acceleration, their position in the lanes and the state of the brake and turn signal indicator lights on other vehicles. Figure 7 shows a typical scene that a driver may encounter when driving through an intersection. The sensor processing system has to be able to identify: the position and velocity of turning cars; the position and velocity of oncoming cars; pedestrians; traffic signals; the position

and velocity of vehicles in the direction of travel; road edges; intersection edges; lanes and the position and velocity of the driven vehicle in its lane.

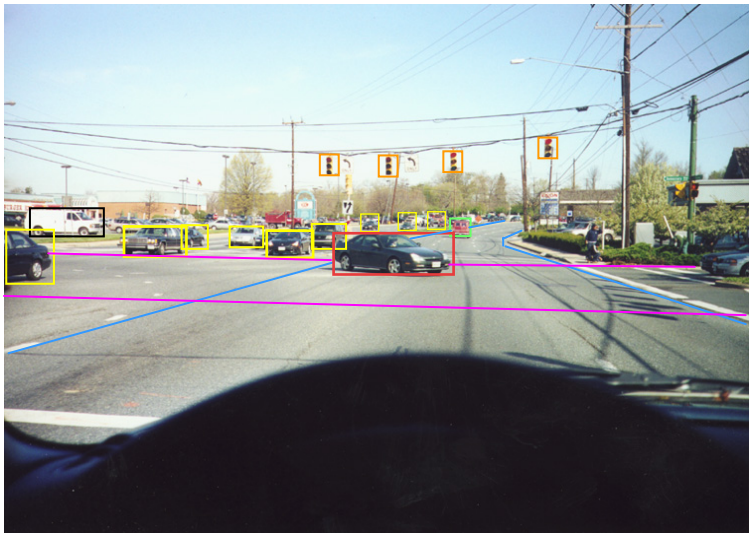


Figure 7. Perception needed for driving on roads.

Chapter 2 described how the NIST developed 4D/RCS methodology and reference architecture was used in the MARS project for the task analysis of autonomous on-road driving. It further described how this task decomposition representation is used as the framework to further specify the world model attributes, features, and events required for proper reasoning about the driving scenario subtask activities. These world model specifications, in turn, are used as the requirements for the sensory processing system. These requirements identify those things that have to be measured in the environment, including their resolutions, accuracy tolerances, detection timing, and detection distances for each subtask. This methodology concentrates on the task behaviors explored through example scenarios to define a task decomposition tree that clearly represents the branching of tasks into layers of simpler and simpler subtask activities. There is a named branching condition/situation identified for every fork of this task tree. These conditions become the input conditions of the "if-then" rules of the knowledge set that define how the task is to respond to input state changes. Detailed analysis of each branching condition/situation is used to identify antecedent world states and these, in turn, are further analyzed to identify all of the entities, objects, and attributes that have to be sensed to determine if any of these world states exist. An example given is the subtask activity to "Pass_Vehicle_In_Front" on a two lane road. This is just one of over 170 subtask activities so far identified for on-road driving. After evaluating possible branching conditions/situations and the associated world states for this subtask, the objects that have to be observed to recognize the world states are identified along with the minimum sensing resolutions needed for each particular subtask. At a speed of 120 km/h, the minimum passing zone is 200 m or more. In order to detect objects in this passing zone, a minimum resolution of 0.05° to 0.09° is required. This is consistent with the LADAR BAA requirement specifications mentioned earlier. At higher speeds, the passing zone will be longer and the minimum resolutions smaller, but this sets a reasonable sensor requirement for this particular subtask.

Analysis of high resolution LADAR for driving was conducted as part of the MARS project. Figure 8 shows a LADAR range image taken with a long range, high resolution range imager of vehicles on a road and at an intersection on the NIST grounds. For this scene, cars were detected and localized for ranges out to 62 m, however, longer ranges (out to 200 m) for the detection of a road and the cars on a road were possible. The project concluded that the perception problem for on-road driving is tractable using LADAR image processing techniques.

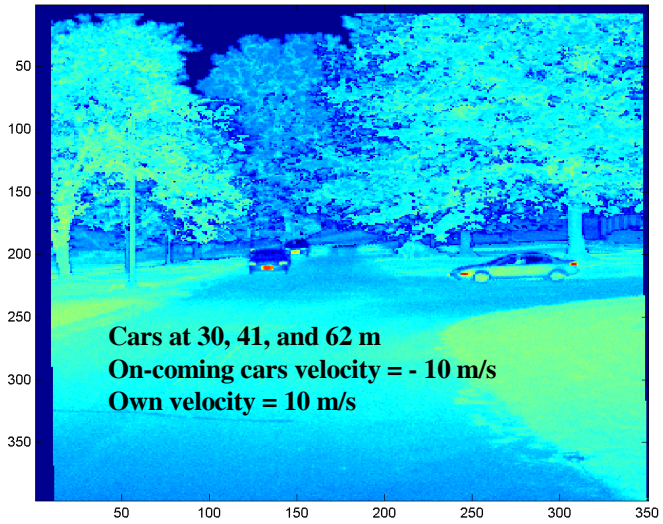


Figure 8. High resolution LADAR range image of vehicles on a road.

2.3 Updated Requirements of LADAR for UGVs to be used in FCS

Early in 2005, DARPA tasked several government labs versed in advanced LADAR technology to conduct an analysis of LADAR performance requirements for future FCS UGV autonomous driving applications. Although the main objective of the task was to define requirements for LADAR stealth, only the updates to LADAR sensor performance requirements for autonomous driving will be presented here. In this study NIST was tasked to leverage the existing performance requirements established for the NIST BAA in next generation LADAR, and to update the requirements for tactical UGV missions. NIST updated the requirements based on guidance from DARPA and on the outcome of an ARL funded project to establish perception and autonomous driving requirements for a tactical Road Reconnaissance mission.
Initial study results have indicated that the original NIST BAA requirements still stand but need some minor changes and additions. Figure 9 is a conceptual diagram of a single LADAR sensor or dual sensors intended for autonomous UGV driving needs. A Wide FOV (WFOV) (40° x 120°), coarse resolution (0.25°) LADAR is needed for peripheral vision and a Narrow FOV (NFOV) (4° x 12°), fine resolution (0.025° or better) is needed for saccadic foveal perception. The intent is to steer the foveal LADAR to areas of interest within the field-of-regard of the peripheral LADAR sensor at a rate of 3 - 10 saccades per second. The higher resolution is necessary to detect, classify and track objects and personnel on or near the path taken by the vehicle at distances up to 200 m, when the vehicle is operating at top speed. Some sort of image stabilization or image motion compensation must be provided for the high resolution camera.

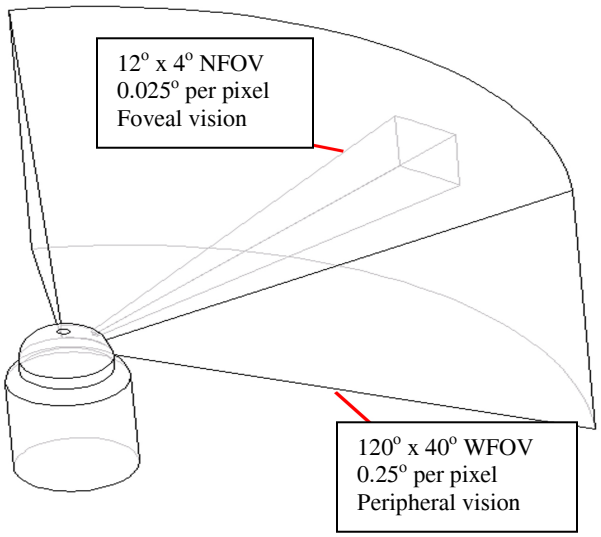


Figure 9. NIST conceptual diagram of a single LADAR sensor or dual sensors for FCS UGV autonomous driving needs

In addition to the increased resolution for the foveal LADAR, there are some other suggested changes and additions:

1. Performance in range measurements should be stated as follows:
   Range uncertainty (standard deviation plus bias) must be ± 5 cm.
   Range resolution should be at least 15 cm or better
2. In addition to range data, provide intensity and color data for improved object classification and recognition.
3. Measurement requirements should also include: Concertina wire detection, thin wire and object detection, and detection of rocks hidden in grass and foliage. This will require understanding the capability of algorithms and processing software needed to perform automatic terrain and object detection and classification. A couple of publications[4,5] describe research which was conducted in this topic area.

As UGVs near deployment in military operations for FCS, there is a growing need for high speed driving safety. In a study conducted for ARL, NIST has generated some initial perception performance requirements for LADAR at distances out to 100 m. The LADAR sensor must be able to detect and identify a person in the path of the vehicle in time for the vehicle to stop or avoid hitting the person. Tests conducted at NIST with a long range, variable resolution, high performance LADAR (Riegl LMS Z420) have concluded that LADAR with coarse resolution (0.2° to 0.25°) can detect objects the size of a human at 100 m, but cannot identify them. When the resolution was set to 0.02° (foveal LADAR perception), it was possible to apply segmentation approaches to identify a person as shown in Figure 10. There are 600 pixels on the target at 100 m. In addition, by combining the range image with color, fewer than 100 range pixels on target may be required to identify a person at ranges past 100 m.

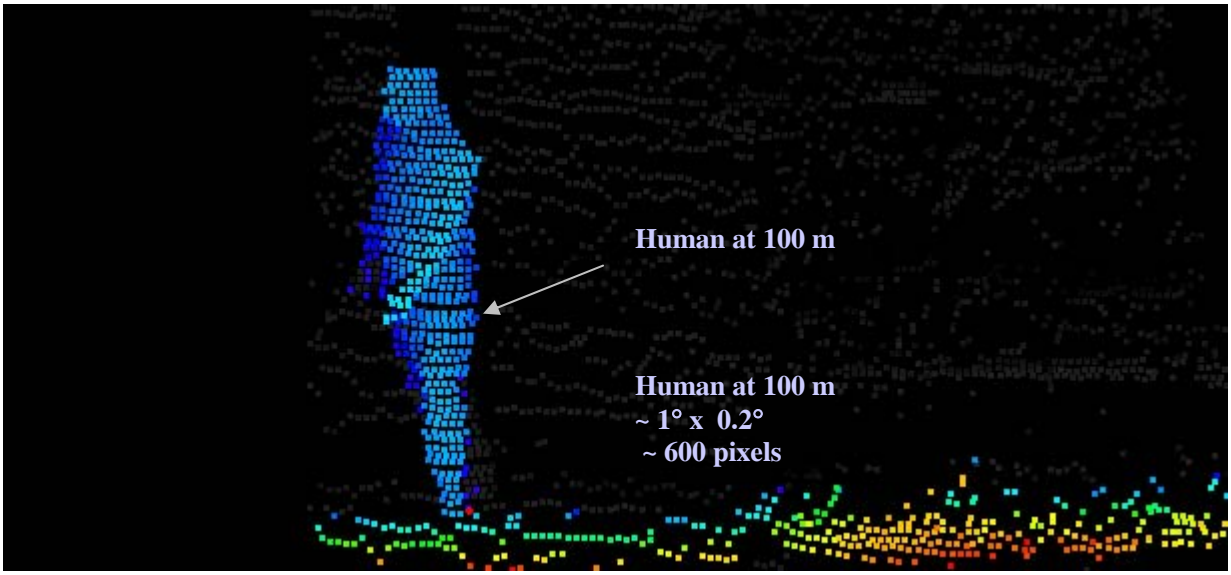


Figure 10. Recognition of a human target at 100 m. Range image taken with Riegl LMS Z420 set to an angular resolution of 0.02° per pixel. This is approximately the resolution of unaided human foveal vision.

3. Example Potential Solutions to Advanced LADAR Requirements

The following are technical synopses offered by the original four NIST Phase I BAA winners on their proposed designs for a next generation LADAR for driving UGVs. Although the BAA did not proceed to Phase II, the participants have continued to make progress and some have provided progress reports which are included following the synopses.

3.1 Advanced Scientific Concepts Inc. (ASC)

3.1.1. Synopsis of ASC's CUGVEL

The objective of the Advanced Scientific Concepts Inc.'s (ASC) contract effort was the design of a 3D Flash LADAR system that could meet the specifications discussed in section 2.1 of this chapter. The proposed ASC system is called the Compact Unmanned Ground Vehicle LADAR (CUGVEL). The ASC designs used Commercially Off The Shelf (COTS) parts as much as possible to reduce cost and a compact laser was used to reduce volume, weight and power. The designs used no mechanically moving parts for laser scanning and only a single laser pulse was needed to capture the entire FOV with a hybrid 3D FPA. Figure 11 shows a typical ASC hybrid 3D FPA configuration. It shows how a Readout Integrated Circuit (ROIC) is bump bonded to a solid-state detector array (such as a Silicon, InGaAs or HgCdTe PIN or APD detectors). Each unit cell, or pixel, contains circuitry which independently counts time from the emission of a laser pulse from the camera, to the detection of the reflected pulse from a surface. In this manner, each pixel captures its independent 3D information in a scene (angle, angle, range). Additional circuitry is available in each pixel to capture temporal information from the returning pulses. Twenty sampling circuits are provided in the ASC FPA design which helps in detecting objects which are obscured by smoke, foliage, etc.

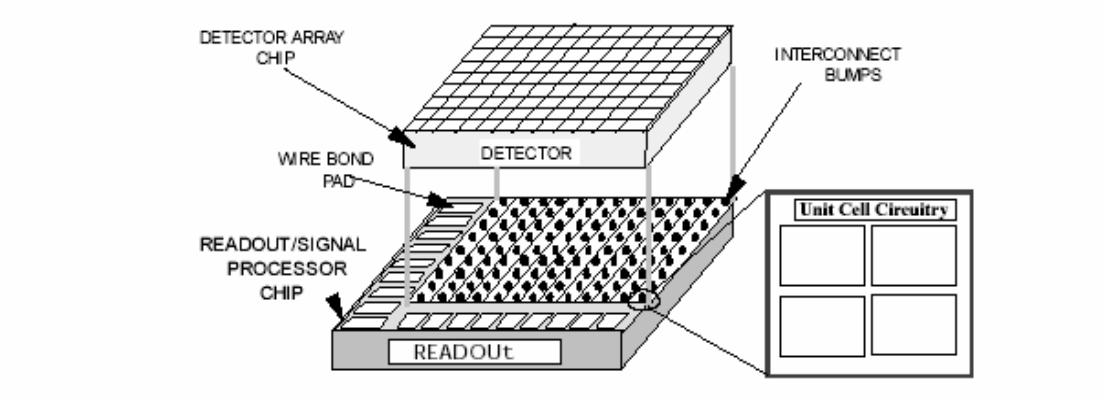


Figure 11. ASC 3D FPA Hybrid Design. ROIC bump bonded to detector array chip.

As designed with off-the-shelf optics, all the 3D imaging systems use two aperture systems whether they are WFOV, NFOV or a combination of the two: an aperture for the laser-transmit optics and an aperture for the 3D imaging, receive optics. Figure 12 illustrates a possible camera configuration for the ASC pulse TOF WFOV design[6]. The drive and output electronic circuit boards, as well as the laser transmitter, are inside the camera housing. The same configuration and the same 3D FPA (with longer focal length optics) can also meet the needs for a NFOV stand-alone sensor. Single aperture CUGVEL systems are possible at an increased cost with potentially reduced weight and volume. There are advantages for having two WFOV and NFOV imaging systems; objects that require fine resolution can be investigated completely independently, without affecting the WFOV frame rate. In addition the optics are simpler. The disadvantage

is the increased weight and volume of two separate systems. A combined WFOV and NFOV system is also possible; however, this concept would have to be further evaluated.

Figure 12. Possible packaged configuration for the standalone WFOV CUGVEL. Estimated weight is 1.8 kg; the COTs optics are a large fraction of the weight.

3.1.2 Progress Report on ASC Flash LADAR Development

In 2004 ASC reported[6] that their prototype camera, shown in Figure 12, which uses their FPA 3D 128X128 InGaAs PIN array (operates with a 15 μJ, 1570 nm pulsed laser), had been used to take Flash range images at distances out to about 40 m. Several unprocessed range images are shown in Figures 13 and 14. These figures were provided by ASC. Another publication[7] reviews progress at ASC in applying their 3D Flash LADAR to longer range applications (300 – 600 m). Various lasers, with pulse energies up to 45 mJ at a wavelength of 1.5 μm, were used for the longer range operation. Figures 15 and 16 show a top down range image and a rotated 3D range image respectively of a building and cars taken from a flying platform at an altitude of over 300 m. This work, with a newer version of their Flash LADAR, illustrates the ability of the Flash LADAR to be used for object identification and 3D mapping from a flying platform.

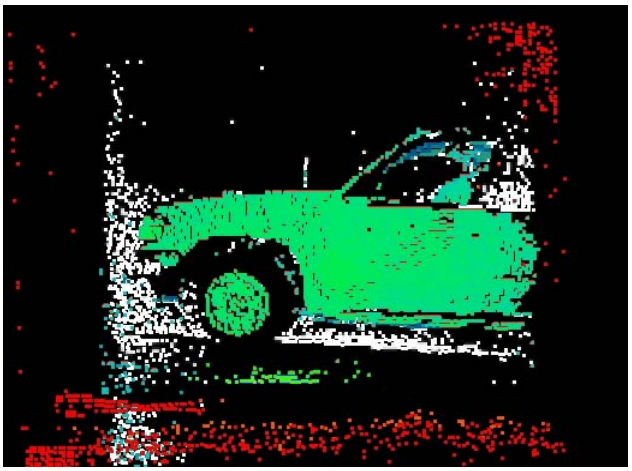


Figure 13. Car at distance of about 40 m outdoors.

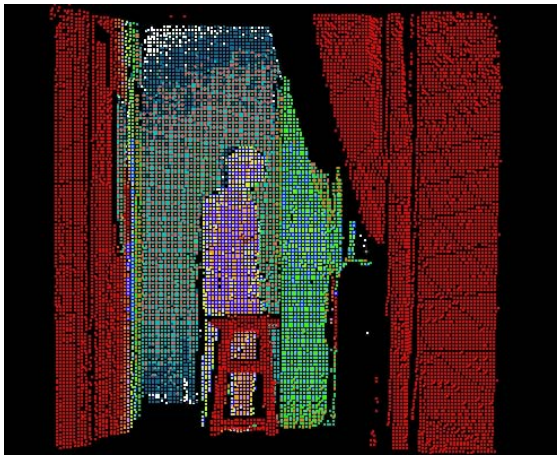


Figure 14. Person sitting on a bench (approx. 15 m)

ASC reports that they are now using a 128 X 128 APD detector with their ROIC electronics. They have demonstrated a times 8 improvement in gain over the InGaAs PIN detector array and are now able to achieve even longer range operation, out to over 2000 m, at a 20-30 Hz frame rate. The new camera imager that they have built around this detector includes a registered color imager in the common aperture path. It could also be configured to include an IR detector array. These features would enhance the object detection, classification and recognition abilities of this camera.

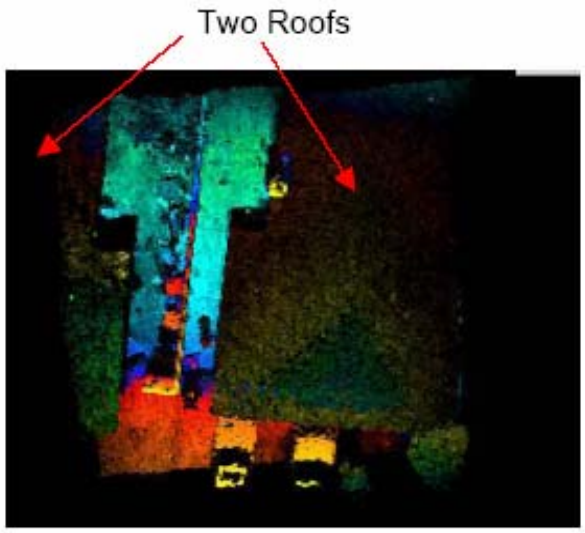


Figure 15. Down looking 3D Image of Two Roofs. Color denotes range.

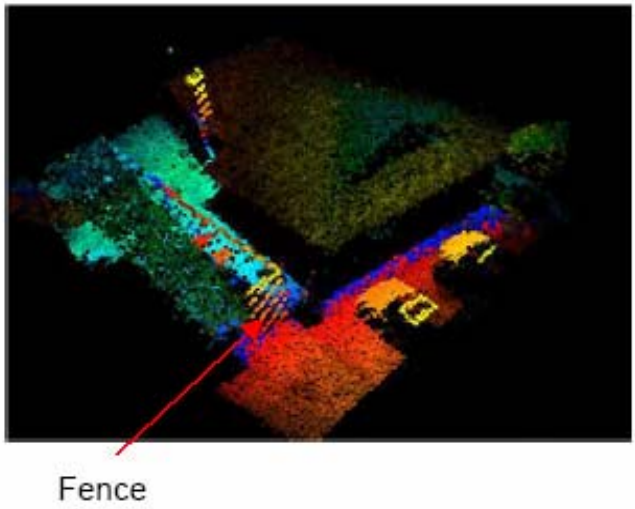


Figure 16. Rotated Figure 15 3D Image. Shows presence of cyclone fence and automobile height.

3.2 Coherent Technologies Inc. (CTI)

3.2.1 Synopsis of the CTI FM-CW LADAR Architectural Approach

CTI has designed an innovative, coherent FM-CW LADAR for an Unmanned Ground Vehicle (UGV) 3D imaging sensor. The proposed approach is a departure from conventional direct and coherent detection designs. It offers significant size, weight, power, and performance advantages over direct detection systems such as pulsed Time-of-Flight (TOF) or Amplitude-Modulated Continuous Wave (AM-CW) waveforms. It also offers cost reduction over conventional pulsed coherent detection designs. For the NIST-specified UGV sensor, the compact coherent FM-CW system meets or exceeds all narrow field-of-view (FOV) requirements (few cm range resolution, 200 m range, IFOV < 1 mrad (0.05°), frame rate ~ 20Hz, > 9° x 9° FOV) and provides pan/tilt mosaics for the wide FOV (40°x90°) operation with moderate frame rates. The narrow FOV high-resolution sensor meets the most stringent requirements while also providing higher sensitivity, superior countermeasure/jam/spoof/damage resistance, and greater range-scalability compared to alternative direct detection architectures. The following is a list of some of the main specific advantages offered by the CTI approach:

- The innovative optical (not RF) homodyne receiver architecture offers a path to very high range resolution (0.75 cm to 3 cm) through high effective bandwidths (10 GHz to 40 GHz).
- The homodyne receiver reduces signal processing burdens by more than 10 times versus conventional broadband signal processing approaches.
- High quantum efficiency (80 %) shot noise-limited performance is obtained by the coherent transceiver where mixing function in the optical domain as opposed to the RF domain, where AM-CW transceivers operate.
- The miniature (few mm size) aperture significantly simplifies the scanner design, thereby enabling high-speed raster scans with a compact low power lower cost laser system.

• The laser power is transmitted with 100 % duty cycle, enabling full utilization of the energy to generate target returns and receiver signal.

• Costly master oscillators were eliminated in the design, making a coherent transceiver practical for this moderate volume, cost-sensitive market.

• Highly linear frequency modulation is not required by this sensor design.

• Optically mixing the FM-CW waveform does not encounter the 3 dB SNR loss that is inherent in AM-CW transceivers (single sideband demodulation) and other RF domain mixing receivers.

• Countermeasure resistance is unmatched for the coherent transceiver. The proposed coherent transceiver can reject large in-spectral band jamming signals by Doppler filtering. Immunity to high intensity jamming is conferred by the fact that the local oscillator is already intentionally driving the receiver to the saturation limit of the detectors.

• A coherent transceiver is inherently more immune to background interference.

• Coherent transceivers offer significantly higher dynamic range than direct detection transceivers. A coherent transceiver detects the field amplitude, not the intensity of an echo as direct detection transceivers do. Coherent detection receivers typically have an 80 dB dynamic range vs. 40 dB for direct detection systems.

• The proposed design utilizes small pixel count focal plane arrays, but can utilize large ones as they become available.

• Speckle is mitigated in the proposed high bandwidth (> 4 GHz) transceiver.

There have been significant technical advances in all coherent transceiver subsystems, thereby making the proposed system tractable. Figure 17 shows a preliminary conceptual drawing of the coherent FM-CW laser radar sensor. The sensor dimensions are 42 cm x 17 cm x 18 cm (L,W,H). The design can be made smaller with additional effort. Smaller in many cases, means lower cost, motivating size reduction.

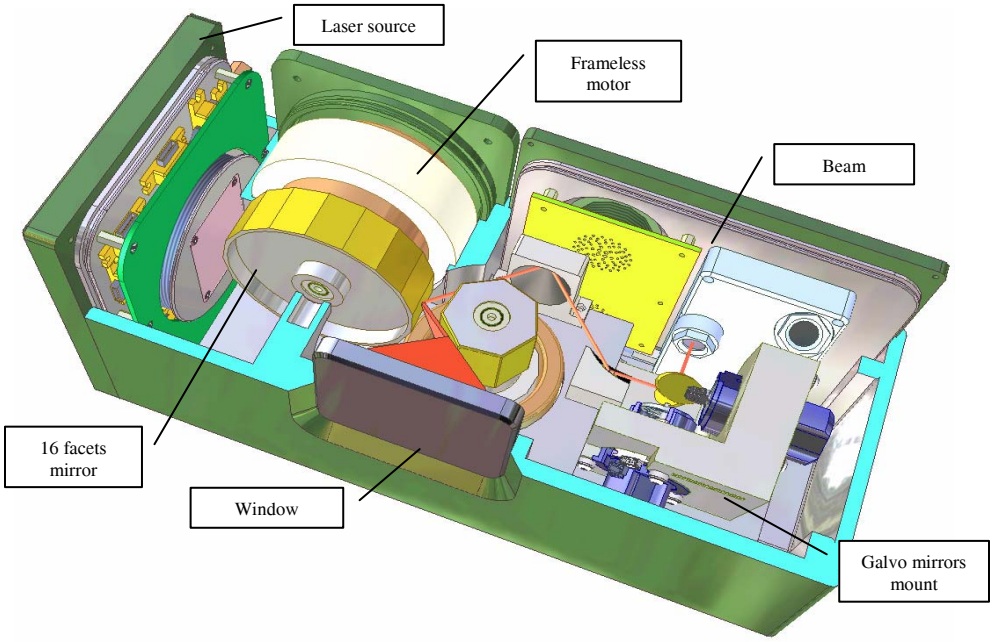


Figure 17. Mockup of Coherent Technologies Inc. FM-CW UGV LADAR.

3.2.2 Progress Report on CTI FM-CW Coherent LADAR Development

CTI reports that they are developing a 32 X 32 pixel array detector for coherent range imaging which could be applied to the needs of improved real-time perception for autonomous UGV applications. This imager will be swept over the area of regard in order to generate a full range image (capable of 0.02°/pixel spatial resolution). Because it uses a very small

aperture size, it may be necessary to provide a short, built it stand-off distance – for eye safety reasons. However, because of the much higher quantum limited detector capability, much lower laser power is needed in comparison to direct detection techniques. An added feature of using the coherent approach comes from its natural ability to directly measure velocities of moving objects using the Doppler shift information. The new coherent LADAR array detector will be ready for testing before the end of 2005.

3.3 Lockheed Martin Missiles and Fire Control

3.3.1 Synopsis of Lockheed Martin's Traditional Scanning, Pulsed Laser Time-of-Flight LADAR Design

A simple system that can meet the shorter-range requirements for autonomous navigation can be constructed using commercial parts and non-developmental items, such as Lockheed Martin's mature laser pulse signal processing electronics. A dual-axis scanned, small linear array receiver was proposed based on experience. This approach led to the proposal of a design that is small, flexible, and inexpensive. Items such as high-repetition rate, low-pulse-energy lasers, detectors, scanners, and processors are commercially available. An added benefit is the ability to change the scan rates to adjust resolution, FOV, and frame rates to meet mission needs without requiring complex gimbaled and/or optical designs.

The basic components of a LADAR sensor are the transmitter, receiver, optical system, and system electronics. The transmitter is a compact, high pulse rate laser (or laser diode). The optical system fully scans the laser to create an image with fine angular resolution. It also provides the collecting aperture for the InGaAs (or Silicon) detector receiver. The system electronics control the laser, scanner, process the laser returns, and provides data to update the terrain database. Figure 18 shows the proven Lockheed sensor system architecture.

A core technology of the system is Lockheed Martin's existing, flight-tested, signal processing electronics to capture and analyze laser pulses. This technology is referred to as the Pulse Capture Electronics (PCE). It is a well-established direct detection approach that accurately determines relative reflectivity (scene intensity) under varying conditions of range, atmospheric attenuation, obliquity, multiple returns, and noise. This is accomplished by matching a programmed template with the entire return pulse data to minimize the effects of signal strength variation, noise and distortions. Figure 19 shows LADAR pulse processing with the Pulse Capture Electronics.

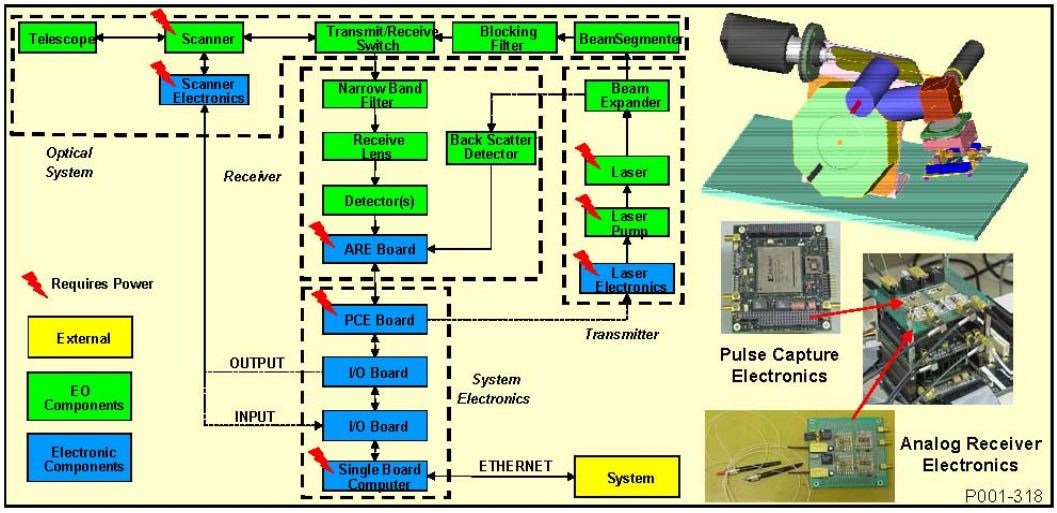
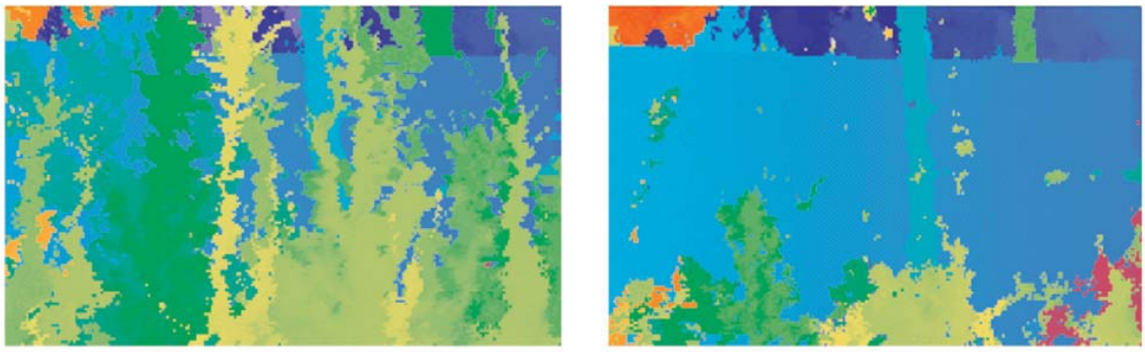


Figure 18. Lockheed LADAR concept uses proven system architecture, signal processing electronics, commercial components, and standard interfaces.



First Pulse Sees Foliage Second Pulse Sees Wall Behind Foliage

Figure 19. LADAR Pulse Processing. Pulse signal processing has first / best / last pulse logic for improved, single-frame, imaging performance through foliage.

3.3.2 Progress Report on Lockheed Martin UGV LADAR Development

Lockheed Martin has developed a couple of breadboard LADAR sensors using internal IR&D funding and plans to use them for FCS autonomous ground vehicle applications. They are based on the miniaturized polygonal scanning concept proposed in the Phase I part of the NIST LADAR BAA. The two breadboards have the following specifications:

Breadboard no. 1: Uses a 50 kHz laser and collects 0.25° pixels over a 90° X (up to) 40° FOV image. They typically operate it at a 90° X 30° FOV to get a 1 Hz scanning rate for ranges out to 50 m.

Breadboard no. 2: Uses a 200 kHz laser and collects 0.10°pixels over a 90° X (up to) 40° FOV image. They typically operate it at a 90° X 22° FOV to get a 1 Hz scanning rate for distances up to 1000 m. This system was being upgraded to operate with a linear array. This would increase the scanning rate to 4 Hz for a 90° X 30° FOV. This represents a range data collection rate of approximately 1,000,000 pixels per second.

Some initial image data collected with breadboard no. 1 is shown in Figure 20. Since the system is capable of detecting and processing three returns from each laser pulse, the corresponding three range images are displayed.

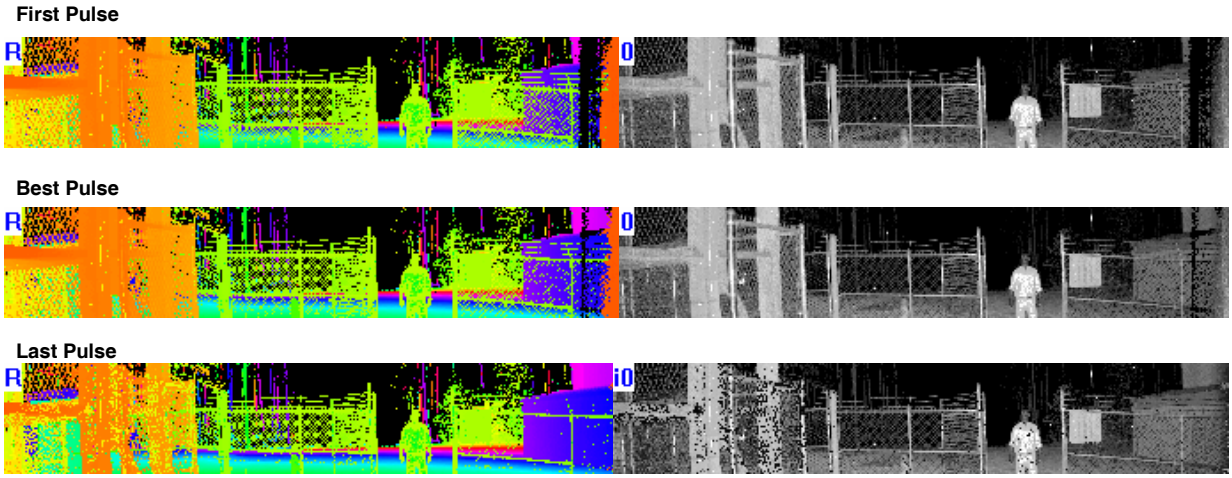


Figure 20. Range image data taken with Lockheed Martin UGV LADAR breadboard no. 1. Range data on the left and corresponding reflectance image data on the right.

3.4 Raytheon Missile Systems

3.4.1 Synopsis of Raytheon's FPA Flash LADAR Design Concept

Raytheon is developing a 256 x 256 pixel HgCdTe Flash LADAR avalanche photodiode (APD) detector array and a multi-pulse processing Read-Out-Integrated-Circuit (ROIC) for the Air Force research Laboratory (Eglin A.F.B., FL) for a seeker application. APD's and ROIC's have been designed, fabricated, hybridized and tested. All functionality has been verified. Operability's in excess of 97 % have been achieved. Figure 21 shows elements of the detector buildup as well as the final detector configuration on a Leadless Chip Carrier (LCC). Table 1 lists the detector performance requirements. The detector records amplitude of the first pulse, and times of arrival of the first and second pulse returns. The ROIC performs several functions including global bias, individual bias adjustment for each pixel, timing ramp generation, and signal pre-amplification. Raytheon is now under an AFRL Dual Use Science and Technology (DUST) contract to extend the ROIC to three pulse-return capability, as well as improve range accuracy and resolution by a factor of 2. To achieve these improvements, the DUST ROIC is being designed with 0.18 µm geometry, whereas the AFRL/MNGS Flash LADAR ROIC has been designed with 0.35 µm. First demonstration of DUST ROIC performance was in late 2004.

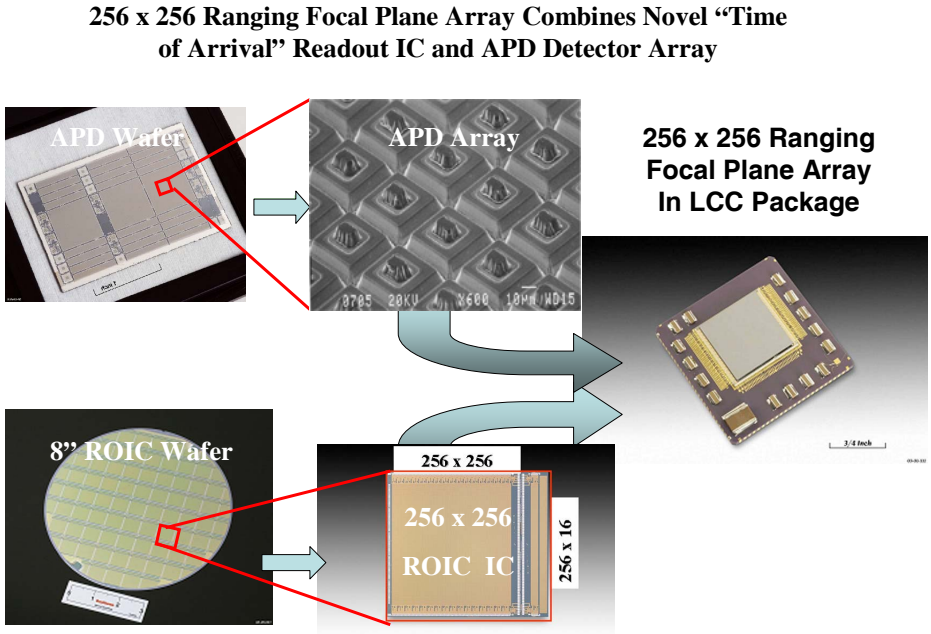


Figure 21. Raytheon FPA technology for a fully functional, advanced, 256 x 256 Flash LADAR detector array.

Table 1: AFRL / MNGS
Flash LADAR Detector Requirements

Geometric Parameters	
Format	256 x 256
Pixel Pitch (µm)	60
Detector Optical Area (µm x µm)	35 x 35
EO Parameters	
Wavelength (µm)	1.55
NEP System (pW/rt-Hz)	0.1

ROIC TIA Noise Current (pA/rt-Hz)	0.5
NEPD APD (pW/rt-Hz)	0.08
NEP APD (nW)	0.52
Bandwidth (MHz)	100
Idark (pregain) (nA)	<10
K (electron to hole ionization ratio)	0.1
Gain, M	5-10
Fex	4
QE %	>90
Fill Factor %	>80
Crosstalk %	<1

Figure 22 depicts Raytheon's AFRL/MNGS Flash LADAR system concept. This brassboard sensor will be fully computer controlled, provide real-time imagery display at 10 frames per second, perform burst mode data collection/transmission at 20 frames per second, and output all LADAR data to the AFRL/MNGS ATA data processor.

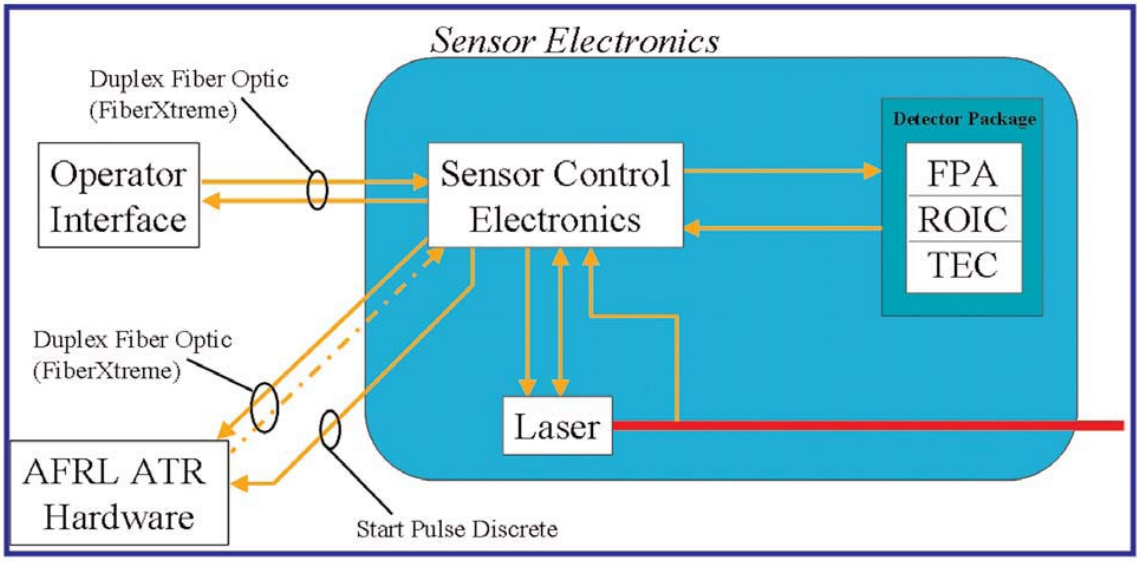


Figure 22. System layout for Raytheon's AFRL/MNGS Flash LADAR brassboard system. Gimbals and scanning hardware are not required to collect LADAR imagery over a large field-of-view.

Table 2 highlights the advantages Flash LADAR has over conventional scanned LADAR sensors for the UGV application. Raytheon believes the detector array developed under the AFRL/MNGS Flash LADAR program will serve well in demonstrating UGV LADAR technology, and that the subsequent DUST array will serve to generate a high performance, low cost, UGV LADAR production system.

Table 2: Advantages of Flash LADAR for UGV Applications

- Overall system simplicity and ruggedness
- Data collection rates reaching 2-Million pixels/second
- Fine stabilization and motion compensation not required
- No motion artifacts in collected imagery
- Most electronics processing able to be carried out on single integrated circuit

3.4.2 Progress Report on Flash LADAR Development

At the time of this writing, Raytheon was in the process of completing the effort funded by the Air Force Research Laboratory, Munitions Directorate, Eglin A.F.B, FL, to generate a brassboard Flash LADAR seeker. Dr. William Humbert is the AFRL/MNGS program manager. AFRL intends to use the brassboard seeker for development of Flash LADAR autonomous target acquisition (ATA) algorithms. To generate the seeker, Raytheon is developing a 256 x 256 pixel HgCdTe flash LADAR avalanche photodiode (APD) detector array, a multi-pulse processing read-out-integrated-circuit (ROIC), and Flash LADAR seeker system components and architecture.

3.5 Summary of Main Features and Salient Characteristics of Proposed Advanced LADAR Concepts

Table 3 summarizes the main features and salient characteristics of each of the proposed concept designs offered by the original NIST Phase I BAA winners.

Table 3: Summary of example potential solutions for driving UGVs

Company	Type of LADAR	Main Features & Salient Characteristics
Advanced Scientific Concepts	2D FPA "Flash"	1. scannerless 2. system simplicity & ruggedness 3. highest potential data collection rate 4. leverages other agency sponsored R&D 5. meets NFOV & WFOV imaging requirements 6. high immunity to motion artifacts 7. small sensor size 8. high cost savings in large volume production 9. 128 x 128 operational arrays available for tests 10. stores 20 consecutive return pulse samples 11. excellent obscurant penetration
Raytheon Missile Systems	2D FPA "Flash"	Same as for Advanced Scientific Concepts items 1. through 8. 9. largest FPA array being tested (256 x 256) 10. stores first and last laser pulse returns 11. some obscurant penetration capability
Coherent Technologies Inc.	Coherent FM-CW	1. scanning sensor with small linear array 2. highest potential detector sensitivity 3. low laser power requirements because of 2 4. highest potential dynamic range and resolution 5. highest potential range image quality 6. outstanding obscurant penetration 7. optimized for NFOV imaging requirements. and data rate
Lockheed Martin Missile & Fire Control	Scanning pulsed laser Time-of-Flight	1. scanning sensor with small linear array 2. based on existing LADAR built for flying ATR 3. good data rate – uses multiple detector array 4. short term delivery possible 5. stores three laser pulse returns 6. good obscurant penetration capability 7. programmable FOV – NFOV and WFOV combined in one sensor possible

4. Examples Of Existing Real-Time Advanced LADAR Prototypes

This section describes several additional advanced LADAR prototypes (FPA and Scanning) which have been built by the developers to demonstrate use of such sensors for improving perception in autonomous and semi-autonomous vehicles for autonomous navigation, non-contact safety systems, autonomous material handling, and other industrial, commercial, transportation and military applications.

4.1 MIT Lincoln Laboratory

Work at MIT Lincoln Lab has taken another approach to FPA development. In order to achieve enhanced ionization which is responsive to the arrival of a single photon, they have developed a "Geiger-mode" (GM) avalanche photodiode (APD) array that is integrated with fast CMOS time-to-digital converter circuits at each pixel[8]. When a photon is detected there is an explosive growth of current over a period of tens of picoseconds. Essentially, the APD saturates while providing a gain of typically > 10^8. This effect is achieved by reverse-biasing the APD above the breakdown voltage using a power supply that can source unlimited current. Over the last couple of years movies have been taken with a GM APD 32 x 32 FPA LADAR. However, since it is not possible to include the real-time movies in this text, a couple of range images taken with earlier prototypes are provided. These are shown in Figures 23 and 24. These figures were provided by MIT/LL. Several different sensor cameras have been built and demonstrated over the past few years. Some of that work is described in SPIE conference publications[9,10]. They used solid-state Fiber-Pumped lasers (530 nm and 780 nm), operated at fast firing rates (8 - 10) kHz, required between 15 μJ and 30 μJ per pulse illumination, and used pan/tilt and scanning mechanisms to cover a larger FOV.

MIT also reports[11] that they have extended their earlier work with silicon based APDs by developing arrays of InGaAsP/InP APDs, which are efficient detectors for near-IR radiation at 1.06 μm. 32 x 32 pixel arrays, with 100 μm pitches.. Figure 25 shows the key elements that are integrated into a package for Geiger-mode operation. In the figure, light enters the array from the top and is focused by the microlens array onto the detector array. About 70-80% of the light is captured and focused onto the detector pixel elements. The overall detector efficiency with the microlenses is in the order of 30-35% thus considerably increasing detector efficiency. The thermoelectric cooler is necessary to reduce the darck current rates in order to keep the LADAR time-of-flight gates open for times in excess of 1 μs. This is adequate for many LADAR applications. MIT has also demonstrated 32 x 32 pixel InGaAs/InP arrays 1.5 μm operation and plan to extend this work into the 2-4 μm wavelength region.

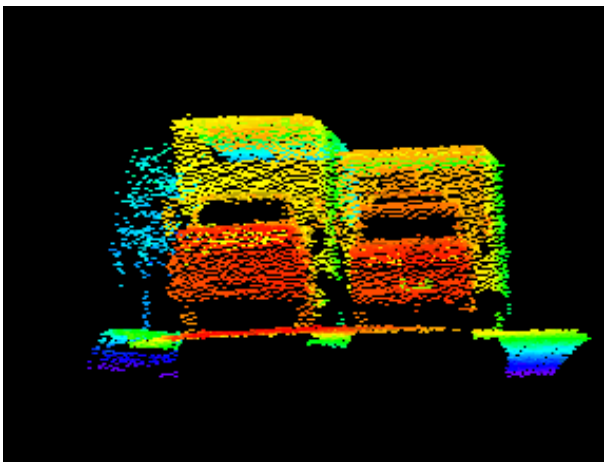


Figure 23. Two trucks at a distance of 150 m.

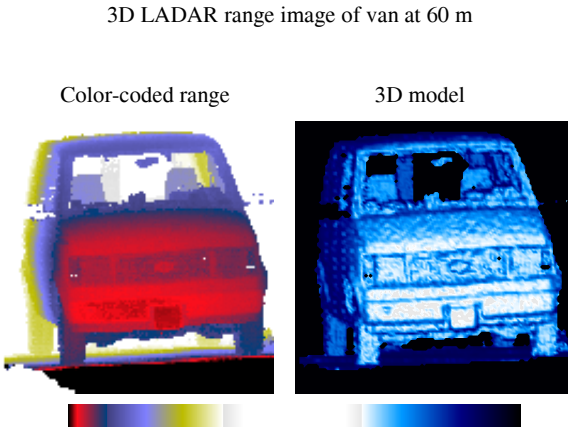


Figure 24. Chevy Van at 60 m (courtesy MIT/LL).

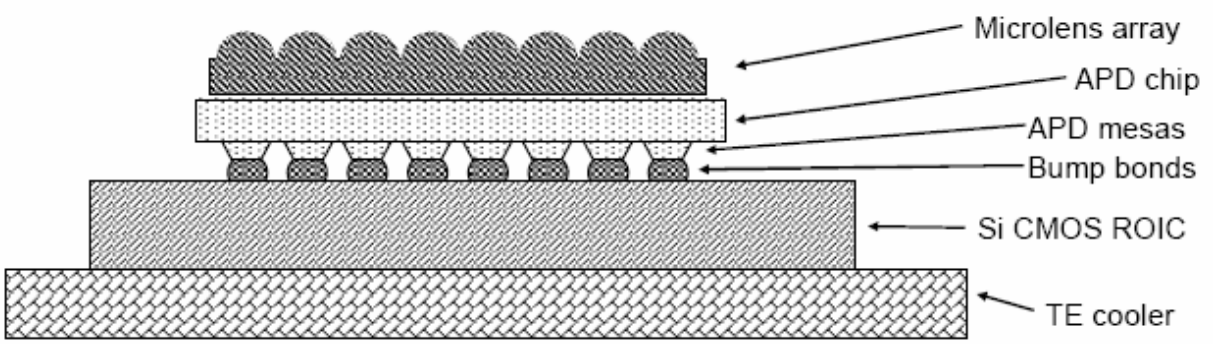


Figure 25. Cross section showing key elements integrated into package for Geiger-mode APD arrays (courtesy MIT/LL).

4.2 CSEM (Zurich, Switzerland)

Still another FPA approach has been developed by CSEM in Zurich, Switzerland. Their research has concentrated on developing compact, robust, low cost, and real-time 3D image cameras, which take advantage of available custom CMOS/CCD technology. This unique combination permits the optimal use of the strengths of both CMOS and CCD technologies is terms of performance. A prototype miniature 3D Time-of-Flight (TOF) camera, the SwissRanger 2 (SR-2), has been built and is available for experimentation purposes[12,13]. The SR-2 is a stand-alone range image camera system which includes the illumination unit, a 3D optical sensor, and control electronics in a very compact unit. It is shown in Figure 26. The emitted optical signal (LED array, 870 nm) is a continuous wave signal which is modulated in amplitude. The IR signal is modulated at a frequency of 20 MHz. This represents an ambiguity interval of 7.5 m for the range measurements. The reflected signal from the scene travels back to the camera, where the TOF is measured by recording the phase delay between the signals. The signal phase is detected by synchronously demodulating the incoming optical signal by cross correlation with the demodulation signal. Sampling response at the detector at intervals of $\Pi/2$, or four equally spaced temporal points, allows for the calculation of the phase delay. The phase delay is measured by each pixel. In other words, each pixel is able to measure the TOF. This results in a complete distance/range map at frame rates approaching 30 Hz where the spatial resolution is defined by the number of pixels in the FPA. For the SR-2, this happens to be better than 0.25° per pixel (124 x 160 pixels for a FOV of about 30°). At the time of this publication, the CMOS detector was not able to suppress the effects of high intensity ambient light, therefore operation in bright sunlight was not possible. Pixels with ambient suppression are in the process of development by both CSEM and the German group at PMD (see below). Figure 27 is a photo of various size obstacle targets set up in a hallway at NIST. Figures 28 and 29 are the SR-2 reflectance and range images, respectively.

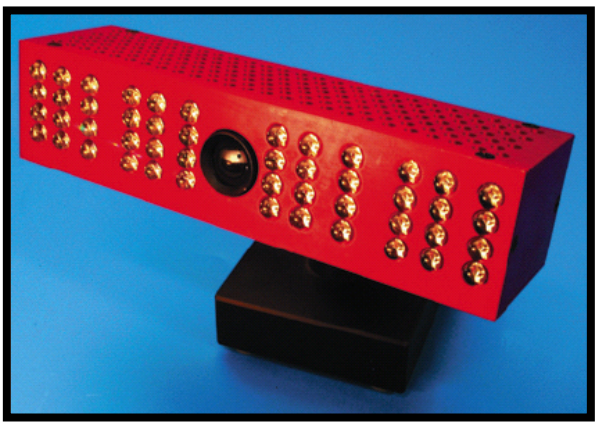


Figure 26. CSEM SwissRanger 2 prototype camera (Figure provided by CSEM).

Figure 27. Photo of various size obstacle targets in hallway.

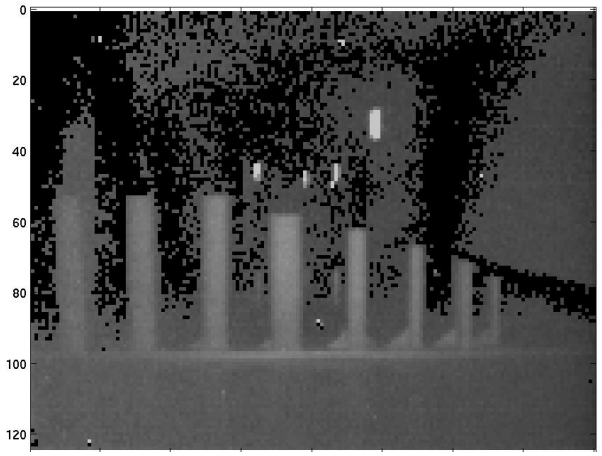


Figure 28. SR-2 active illumination reflectance image.

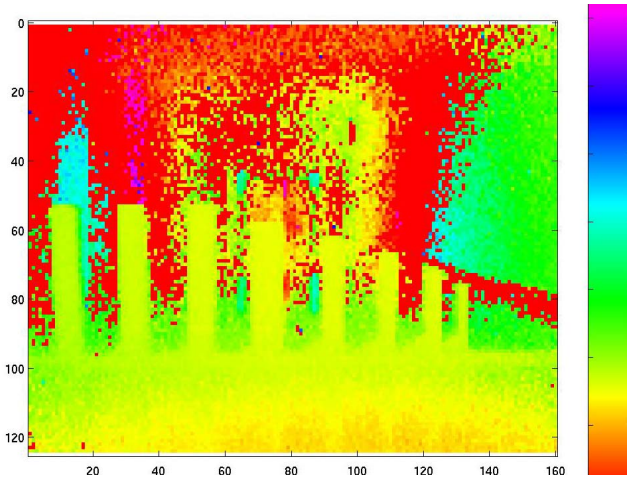


Figure 29. SR-2 processed range image (color coded in range).

4.3 PMD Technologies (PMDTec) in Siegen, Germany

Another continuous wave Amplitude Modulated (AM) phase-based FPA LADAR camera was under development by PMDTec in Germany. Similar to CSEM, it too uses CMOS technology to lower costs, and, because of the single 20 MHz modulation frequency, is limited to 7.5 m (ambiguity interval) operation. The early prototype, now named PMD [vision] 1k-S has been available since 2004. Since the process for measuring range at each pixel is identical to that used by CSEM, it will not be further described here. There are some differences however. The pixel array size is 64 x 16, frame rates of up to 50 Hz are possible, but it is somewhat larger in size when compared to the CSEM SR-2. However, a key feature in the design is the addition of active Suppression of Background Intensity (SBI) circuitry at each pixel. Depending on the intensity of the illumination source and the level of ambient light conditions, tests conducted by the company indicate that an SBI performance as good as 40 dB can be achieved. This implies that operation outdoors, even in sunlight, may be possible. Example images provided by PMDTec taken with the 1k-S camera are shown in Figures 30 and 31. Figure 30 is the reflectance image and Figure 31is the 3D range image.

Figure 30. Active illumination reflectance image taken with PMD [vision] 1k-S prototype camera.

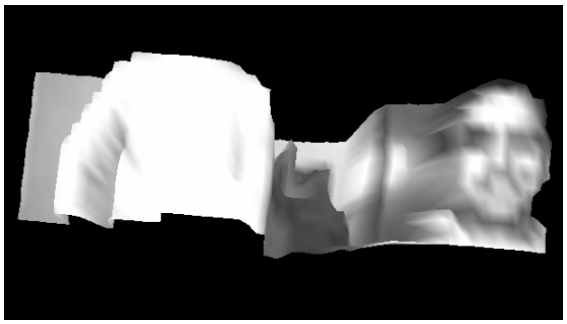


Figure 31. Processed 3D range image taken with the 1k-S camera. Reflectance image superimposed on range image.

In 2005 PMD announced the release of two new camera prototypes, the PMD [vision] 3k-S and the PMD [vision] 19k. The 3k-S 3D range camera is similar to the 1k-S camera, but has a larger 64 x 48 array size and also has the SBI circuitry for each pixel in the array. The newer 19k 3D range camera has a pixel array size of 160 x 120 and is very similar to the CSEM SR-2 camera and has a maximum frame rate of this camera is 10 Hz.

4.4 Canesta, Sunnyvale, California

Canesta, a U.S. company, has introduced competing CMOS-based Focal Plane Array 3D cameras that perform similarly to the CSEM SR-2 and PMD [vision] 3D range camera prototypes. They also use the continuous wave AM phase based approach for measuring time-of-flight (essentially range) at each pixel in the array. At the time of this writing, three Canesta Development Kits were available for research and development purposes. Model DP 203 is a 30° FOV camera, while DP 205 and DP 208 are 50° FOV and 80° FOV cameras respectively. All three have a 64 x 64 array size. Similarly to the PMD Tec cameras, these cameras also have background illumination suppression circuitry built into each pixel in the array. These cameras typically are set to operate at the 7.5 m (ambiguity interval), but the modulation frequency can be easily changed so that the cameras operate over a different ambiguity interval range.

4.5 IBEO Automobile Sensor GmbH, Hamburg, Germany

A LADAR product for automotive safety applications was announced[14] by IBEO (www.ibeo-as.de). The IBEO "Alasca" sensor uses multi-return pulsed TOF range measurements which are scanned across the horizontal FOV. The sensor utilizes a four–layer/plane approach where the returns from the four layers are processed in parallel as the mirror scans the scene horizontally. Although the sensor does not use FPA technology, the sensor can produce a wide 240° horizontal FOV (using two sensors in front) at a resolution of 0.25°, 0.5° or 1.0°, and a 3.2° vertical FOV at a resolution of 0.8° at a 10 - 40 Hz frame rate. Distance range measurements can be resolved to 1 cm and range accuracy can vary as much as ± 5 cm. The multi-target return detection capability was incorporated in the design to allow for the detection of objects on the road even in heavy rain. This model sensor utilizes a 905 nm laser which has a range of 0.3 - 80 m. A 200 m sensor is expected to be released in the future.

The short term objective for this sensor is to introduce preventive safety features/functions in automobiles to assist drivers in unsafe driving conditions. The long term objective is to provide vehicles with full automated collision avoidance capabilities. The Alasca Gen 1 model is limited to 80 m range operation but has been tested and evaluated for automated stop-and-go driving, automated emergency braking, pedestrian detection, pre-crash detection - collision warning, and for other applications such as automated intersection crossing assistance, turning assist, parking assist and blind spot monitoring. The Alasca Gen 2 model, which is expected in the future, will increase detection range to 230 m, will provide fog detection and measurements in fog, will provide for automatic detection and tracking of lane markings and road edges, and provide full ACC capability for stop-and-go driving at speeds of 0 - 180 km/h. Figure 5 shows how LADARs (like the the Alasca) can be used for a variety of automotive safety and driver assist applications. In the illustration the vehicle is fitted with a single sensor (150° opening angle) and 230 m range detection. Figure 32 shows how a test vehicle was instrumented with a multi-layer laserscanner for analyzing the performance of the sensor under realistic driving conditions.

Figure 32. IBEO test vehicle fitted with an Alasca LADAR.

5. Conclusions

Based on the analysis of autonomous on-road driving for the MARS project and autonomous on and off-road driving for FCS (DARPA and ARL) UGV mission applications, some conclusions on LADAR baseline sensing requirements can be drawn. The required sensor resolutions will vary depending on which subtask activity is analyzed and on the speeds of the vehicles involved. Much analysis work has been completed for off-road navigation, however, this work is continuing in an attempt to identify the sensing requirements for all of the on-road driving and on-road subtask activities and the resulting required sensor resolutions. Additionally, on-road military tactical behavior driving tasks are now being analyzed to determine the sensing needs for high speed driving safety. This work will aid in identifying those sensor capabilities that are most important for both near-term and long-term future development of autonomous driving systems.

Potential solutions for meeting the established LADAR requirements were presented in Section 3 with inputs from four leading LADAR developers (winners of the NIST Phase I next generation LADAR for UGVs BAA contract). Although sponsorship and funding for the Phase II part of the development never materialized, the companies have continued to make progress and provided updated status reports on their capability. Section 4 described additional LADAR developers and prototypes which offer potential solutions for improving perception performance of autonomous and semi-autonomous vehicles in the military and civilian sector industries.

Since the release of the original BAA LADAR requirements specification in June of 2002 through the time of publication of this book, NIST has not seen the announcement of any new LADAR products that can meet all of the stated requirements. However, as reported in this chapter, prototype advanced LADAR cameras are becoming available for experimentation which have the potential for a variety of ground vehicle mobility applications in the near future. NIST intends to continue encouraging and tracking these and other new developments, and working with other government agencies and industry to measure LADAR performance and investigate potential new applications. Although some improvements, such as spatial resolution and increased maximum range, of available mobility LADAR are being reported from ongoing work for FCS, not all of the stated requirements are expected to be achieved in the next six years. Therefore, NIST encourages additional funding support for further improving sensor capabilities and for the development of new advanced, high performance sensors based on next generation technology emerging from and reported by leading LADAR developers.

Acknowledgements

The authors of this chapter would like to thank the following individuals and organizations for contributing information and expert opinions on the needs and solutions of advanced LADAR for UGV applications. In particular the authors would like to thank James Albus, Mike Shneier, and Tsai Hong of the Intelligent Systems Division at NIST and Stefan Baten of EADS Dornier for their insight into LADAR performance and requirements for UGV autonomous driving applications. Information on LADAR designs and development research was provided by: Robert Lange and Bernd Buxbaum of PMDTec GmbH; Peter Seitz and Nicolas Blanc of CSEM; Richard Marino and Rick Heinrichs of Lincoln Lab; Duane Smith of Coherent Technologies Inc.; Roger Stettner of Advanced Scientific Concepts; Pat Trotta of Raytheon Missile Systems; Bruno Evans of Lockheed Martin. In addition, the following individuals at NIST should be credited for providing some of the photos and figures used in the report: James Albus; Tsai Hong; Tommy Chang; Peter Russo; Gerry Cheok; William Stone.

References

1. W.C. Stone, M. Juberts, N. Dagalakis, J. Stone, J. Gorman, “Performance Analysis of Next Generation LADAR for Manufacturing, Construction, and Mobility”, National Institute of Standards and Technology, NISTIR 7117, May 2004.

2. “National Automated Highway System Research Program: A Review” TRB Special Report 253, Transportation Research Board, Washington, DC 1998.

3. R. Bostelman, T. Hong, R. Madhavan, "Obstacle Detection using a Time-of-Flight Range Camera for Automated Guided Vehicle Safety and Navigation", Integrated Computer-Aided Engineering Journal, Special Issue: Performance Metrics for Intelligent Systems, Volume 12, Number 3, 2005.

4. N. Vandapel, M. Hebert, "Finding Organized Structures in 3D LADAR Data", Army Science Conference, November, 2004.

5. M. Hebert, N. Vandapel, "Terrain Classification Techniques from LADAR Data for Autonomous Navigation", Collaborative Technology Alliance conference, May, 2003.

6. R. Stettner, et.al., "Eye-safe laser radar 3D imaging", Proceedings SPIE, Defense & Security, Volume 5412, April, 2004, Orlando, Florida.

7. R. Stettner, et.al., "Large format time-of-flight focal plane detector development", Proceedings SPIE, Defense & Security, Volume 5791, April, 2005, Orlando, Florida.

8. R. Marino, et.al., "A Compact 3D Imaging Laser Radar System Using Geiger-Mode APD Arrays: System and Measurements", Proceedings SPIE, Aerosense, Volume 5086, April, 2003, Orlando, Florida.

9. R.W. Cannata, et.al., "Obscuration measurements of tree canopy structure using a 3D imaging ladar system", Proceedings SPIE, Defense & Security, Volume 5412, April, 2004, Orlando, Florida.

10. R. Marino, et.al., "High-resolution 3D imaging laser radar flight test experiments", Proceedings SPIE, Defense & Security, Volume 5791, April 2005, Orlando, Florida.

11. J.P. Donnelly, et.al., "1-μm Geiger-Mode Detector Development", Proceedings SPIE, Defense & Security, Volume 5791, April, 2005, Orlando, Florida.

12. R. Lange, et.al., "Demodulation Pixels in CCD and CMOS Technologies for Time-of-Flight Ranging", Proceedings of the SPIE, Volume 3965A, pp.177-188, San Jose, 2000.

13. T. Oggier, et al., "An all-solid-state optical range camera for 3D real-time imaging with sub-centimeter depth resolution (SwissRangerTM)", Proceedings of the SPIE, Vol. 5249 No. 65, 2003.

14. U. Lages, "Laser Sensor Technologies for Preventive Safety Functions", 12th International Symposium ATA EL, June 2004, Parma, Italy.

Chapter 8

Standards-Based Architectural Framework for Intelligent Autonomous Vehicles

Hui-Min Huang, Elena Messina, and **James Albus**
National Institute of Standards and Technology (NIST)
{hui-min.huang,elena.messina,james.albus}@nist.gov

1 Introduction

The 4D/RCS architecture addresses the problem of intelligent control at three layers of abstraction: (1) conceptual framework (2) reference model architecture and (3) engineering guidelines. Chapter 1 provided comprehensive descriptions for the first two layers. The other earlier chapters focused on particular aspects of the architecture, including the corresponding implementation issues. This chapter focuses on the third layer, the engineering guidelines. Particular attention is paid to the roles that standards play in the process of implementing intelligent systems using an architecture-based approach.

In order to characterize this 4D/RCS engineering approach from an architectural perspective, the following two terms must be defined first:

> Software architecture:
> "*the structure that identifies, defines, and organizes components, their relationships, and principles of design; the assignment of functions to subsystems and the specification of the interfaces between subsystems [1].*"
>
> Software architectural framework:
> *"a set of guidelines and descriptions of approaches for the development and evaluation of software architectures."*

These two concepts anchor this chapter. The chapter is outlined as follows: Some important perspectives for software architectures and for engineering standards are provided in Section 1. These are followed, in section 2, by a summary of the major government and private industry initiatives in architectural frameworks for guiding the development of large software system programs. In section 3, the authors attempt to establish a framework for assessing the standards availability and for analyzing the standards needs in the area of intelligent unmanned systems (UMS) control. In sections 4 through 7, the authors use the established framework to identify, describe, and analyze the relevant standardization efforts. The authors point out some architectural aspects that can benefit from standardization. The authors also discuss whether and how some of the ongoing research efforts, including 4D/RCS might evolve into standards. 4D/RCS's ultimate objective is for itself to be an integral part of the UMS standards portfolio. Section 8 describes an implementation example that uses some of the standards. A summary follows, in section 9.

1.1 Orientations for Architectures

It is not until the last two decades that the issue of software architecture began to receive wide recognition as a key ingredient to the success of software intensive systems. The federal government has since established major, high-level architectural initiatives [2, 3]. Industry and academia have also launched major architectural research and development efforts [4, 5, 6, 7]. However, one must keep in mind that engineering systems are developed for different purposes: to process and organize information, to execute particular business processes, or to generate electromechanical behaviors. The architectural principles

behind the different systems, therefore, must focus on the intended purposes. For example, 4D/RCS is a model-based, behavior-oriented architecture. A key aspect is the set of tasks that the control entities perform. On the other hand, the overall architecture for the U.S. Department of Transportation's Intelligent Transportation System (ITS) [8] involves many aspects, such as traffic management, statistics, and finances. The corresponding sub architectures should be built based on these key aspects.

1.2 National Standards Strategy

The National Standards Strategy for the United States (NSS) and the later United States Standards Strategy (USSS) [9] include calls for increasing government use of the standards and improving the responsiveness of the standards system to consumers' needs. This motivates NIST to actively participate in architectural standardization efforts. The authors view that architectural implementations using standard interfaces, standard components, common definitions of terms, and standard metrics significantly facilitate interoperable or portable systems and contribute to sharable and reusable software components. These, in turn, improve the competitiveness of U.S. products.

1.3 Characteristics of Standards

There are several characteristics of standards that must be understood.

In the U.S., standards participation must be voluntary. Standardization processes must be open, transparent, consensus based, responsive, and meet the needs of the constituencies [9]. Standards apply only when the participants willingly create and/or adopt them. A team could decide to create or adopt certain conventions and call them standards for the team. There are also voluntary groups like Open Robot Control Software (OROCOS) [10] and Orca [11] that seek to develop common robotic control software but do not seem to formally standardize the results. Informal standards of this sort are referred to as *de facto* standards and can be just as effective as formal standards in certain circumstances, particularly when they are voluntarily adopted by participant in a particular special interest group. Included in this informal category are efforts by consortia that aim at component portability and maintenance benefits without their products standards, such as the Automotive Open System Architecture (AUTOSAR) [12], Coupled Layer Architecture for Robotic Autonomy (CLARAty) [13]. Yet, some other consortia, such as Object management Group (OMG) [14], Open Group [15], and OPC [16] may consider formal standards generation as a part of their objectives.

Most commonly, though, standards are generated by the formal Standards Development Organizations (SDOs). SDOs must follow the guidelines and procedures published by the American National Standards Institute (ANSI). ANSI coordinates and administers domestic standards, as well as represents the United States in the International Organization for Standardization (ISO). It is, therefore, advisable to reference standards by their hosts or sponsors, e.g., a DoD standard, ASTM (formerly known as the American Society for Testing and Materials) International standards, etc.

Various levels of conformity can be designated for the published standardization reports. For example, IEEE publishes the following types of documents [17]:

- Standard--for specifying mandatory requirements
- Recommended Practice
- Guide--for providing information
- Trial-Use (for any of the above)--when the document will be published for a limited period (two years or less) before it becomes an official IEEE document.

The Society of Automotive Engineers (SAE) Aerospace Council publishes the following types of documents: Aerospace Material Specification, Aerospace Standard, Aerospace Recommended Practice, Aerospace Resource Document, and Aerospace Information Report. Other SDOs have similar setups.

2 Architectural Framework Standards

This section describes some of the major architectural framework standards that are relevant to the standards framework presented in this chapter. A framework in this context is the support structure within which other standards can be organized and developed.

2.1 Federal Enterprise Architecture (FEA)

Federal Enterprise Architecture (FEA) [2] describes an infrastructure that aimed at providing services to the public. FEA employs a set of reference models, namely, Performance Reference Model (PRM), Business Reference Model (BRM), Service Component Reference Model (SRM), Technical Reference Model (TRM), and Data Reference Model (DRM).

The Service Platform and Infrastructure of the TRM includes embedded technology and software engineering as parts of the service components. This would be the type of service that 4D/RCS provides.

The PRM identifies high-level, key measurement areas, including mission results, customer results, processes and activities, technology, and human capital. Most of these are at the policy level. The engineering performance metrics that concern the researchers is a detailed subset of the PRM.

Overall, UMS architectures, including 4D/RCS could be considered a detailed subset of the FEA.

2.2 Department of Defense Architectural Framework (DODAF)

DODAF [3] specifies that integrated architectures be represented in the following three architectural views, which are inter-related: operational, systems, and technical standards. An additional "all" view addresses issues that are common to all the three views. Each view is further realized by a collection of products. These effectively created a "three plus one" view structure, which is summarized as follows:

A. Operational View

The Operational View addresses the tasks and activities, operational elements, and information exchanges required to accomplish missions. The specific products include: high-level operational concept, operational node connectivity, operational information exchange matrix, organizational relationships, operational activity model, rules model, state transitions, event-traces, and logical data model.

B. Systems View

The Systems View describes systems and interconnections supporting the operational activities. The specific products include: systems interface and communications, inter-systems matrix, systems functionality, mapping operational activities to systems functions, systems data exchange matrix, systems performance parameters, systems evolutions and technology forecast, rules, state transitions, and event-traces for system, and physical schema.

C. Technical Standards View

The Technical Standards View provides the technical systems implementation guidelines for developing engineering specifications, establishing common building blocks, and developing product lines. The products include a collection of the technical standards, options and implementation conventions, rules, and criteria.

D. All View

The All View specifies two products: overview and summary information and integrated dictionary.

These identified products serve as valuable references for 4D/RCS in developing architectural applications. For example, node connections, state transitions, dictionaries are among the key products for 4D/RCS.

2.3 Standards for Levels of Abstraction

There are multiple standards that describe the multiple layers of architectural abstraction for UMS. The authors use them to identify the architectural standards needs.

The Open Systems Interconnect (OSI) model, developed by the ISO contains the following seven-layer of abstraction: Application, Presentation, Session, Transport, Network, Data link, and Physical.

The Internet Protocol Suite (also referred to as the TCP/IP suite) contains the following layers of abstraction: Application, Transport, Network, and Data link.

The DoD TRM, based on the SAE General Open Architecture (GOA) model, described four layers: application software, system services, resource access services, and physical resources.

The three models can easily be mapped. The authors will use the GOA/TRM model for the stated purposes of this chapter.

2.4 IEEE 1471-2000

This standard, entitled "Recommended Practice for Architectural Description of Software-Intensive Systems," identifies the roles of architecture in a software engineering community that includes systems, various stakeholders, and the operational environments [18]. This standard also specifies that architectures be described using a set of views and composed with a set of models. The standard helps to identify the entities that should be involved in the architectural development for UMSs.

However, IEEE 1471-2000 does not cover the next level details of specific models or views as DODAF does. Also, although IEEE 1471-2000 correctly identified various application scopes for the architectures: those for single and new system, those for existent systems, and iterative architecture for evolutionary systems, it does not emphasize a particular type, namely, reference architecture, of which 4D/RCS is one. IEEE 1471-2000 provides a different perspective for architectures, and, as such, it could complement the aforementioned architectural framework standards.

3 UMS Standards Needs from an Architectural Framework Perspective

Standards definition is a critical architectural framework issue. The authors use a two-dimensional model to describe an architectural framework, depicted in Figure 1. The vertical axis is adopted from the SAE GOA/DoD TRM. In other words, from the vertical perspective, the architectural standards should cover three major layers of abstraction: hardware (computer, sensor, actuator, chassis, power), operating environments and resource handling, as well as applications and application support software.

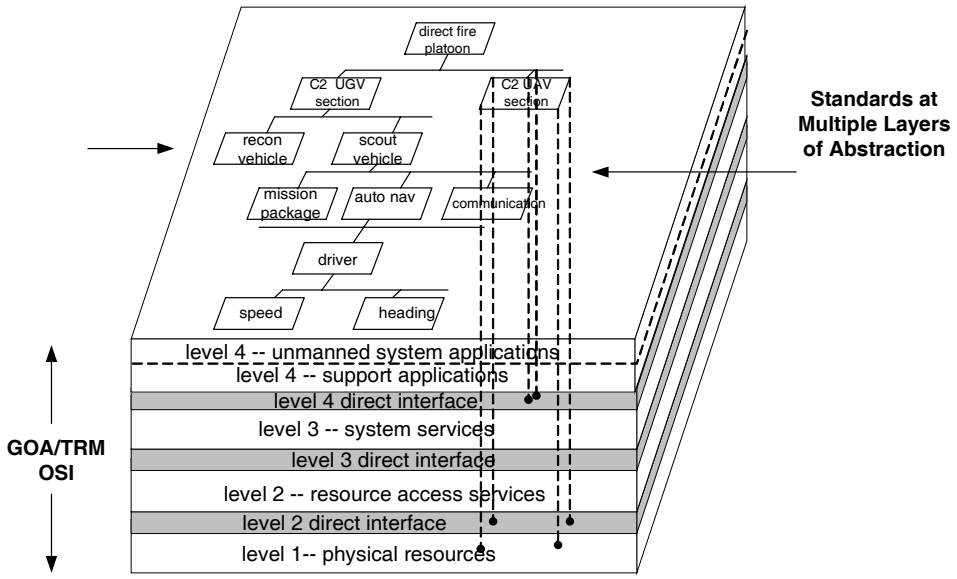


Figure 1: Architectural Framework for Standards Analysis.

Horizontally, the architectural standards should cover other following aspects relevant to the standardization effort. These include:

1. organization of architectural applications
    a. functional components
    b. inter-component connections
    c. governing rules for organizing the components and connections
2. knowledge models for tasks, entities, events, images, maps, and costs
3. execution models
    a. solution paradigms such as state-transitions, search
    b. messaging protocols such as send/receive acknowledgement, periodical vs. per request
    c. inter-component dependence such as dynamic configuration, sequential vs. concurrent
4. metrics
    a. functional requirements, such as missions and tasks
    b. systems performance specification on computers and on electric/mechanical systems
5. processes and tools
    a. terminology
    b. development methods through out life cycle, including domain knowledge mining and orchestration, system partition
    c. tools and environments for representation and for simulation
    d. testing, verification and validation, user interface

The following sections address the UMS standards needs and availability based on this framework.

4 Application Layer Standards

This layer concerns application software for the unmanned vehicle systems. Standards efforts and needs are focused on the following types:

- Architectural Connectors—the interfaces that convey information among the architectural components
- Architectural Components—the computing units that utilize the Connectors to support intelligent systems

- Data Models—the information that support intelligent decision making
- Execution Model—the real-time decision making processes

4.1 Existent Architectural Connectors Standards

The ongoing standardization efforts in this category include SAE AS-4 and North Atlantic Treaty Organization (NATO) Standardisation Agreements (STANAG). The Unmanned Systems committee, designated as AS-4, of the Aerospace Avionic Systems Division, Aerospace Council of the SAE is developing a message specification for UMS. AS-4 was migrated from the Joint Architecture for Unmanned Systems Working Group (JAUS WG), originally chartered by the Office of the Secretary of Defense (OSD). NIST is a charter member of JAUS WG.

The AS-4 committee advances the JAUS documents. The Unmanned Systems Message Specification (UMSMS) is the next version of the message specification section in the JAUS Reference Architecture, Version 3.2. Within the UMSMS, there are six categories of messages, namely, command, query, inform (responses to queries), event setup, event notification (responding to event setups), and node management. Each message is assigned a code, i.e., a message ID. Each message class is assigned a code range, or message space. Additional message space is reserved for applications that require messages that are not yet specified in the UMSMS. These are called User Defined Messages. Proven User Defined Messages should be submitted to the WG for consideration. Messages begin with a standard, 16-byte header that includes the following fields:

- message properties for specifying such properties as message priority, acknowledgement of receipt, whether user-defined or standard message, and the version of RA or UMSMS that the message conforms to,
- source and destination IDs for the involved components, nodes, and subsystems,
- data control, for indicating data size and single/multi packet transactions, and
- sequence number for serializing the messages.

There is a service connection feature in the messaging that support regular, periodic data transmission of the Inform or Command class messages. The UMSMS also supports the broadcast type of messaging.

4D/RCS systematically lists sets of command vocabulary and identifies interfaces among architectural components. 4D/RCS could, therefore, serve as a reference for the UMSMS.

The NATO Standardisation Agency (NSA) has published a series of standards, called STANAGs. The title of STANAG 4586 is "Standard Interfaces of UAV Control System (UCS) for NATO UAV Interoperability." The document describes interfacing standards, such as the Data Link Interface (DLI), that provides common messages and mechanisms for UAVs; Command and Control Interface (CCI), which defines messages on airspace control, air traffic control, mission planning, etc.; and the Human Computer Interface (HCI), which is for operator control, monitoring, warnings, cautions, and advisories, communications management, post mission reporting, etc.

The document covers both the data content and the protocols, which are identified as the top three layers in the authors' framework of study. These, to some extent, overlap the aforementioned SAE AS-4 work in the domain of UAV. Efforts have begun to investigate the possibility of integrating the two.

4.2 Further needs and paths for interfacing standards

Command interface is essential to UMS. In DoD, Services publish standard tasks, such as Air Force Task List (AFTL) [19] and Army Universal Task List [20]. These might serve as good starting points for developing architectural command interfaces. In addition, DoD has message standards to facilitate information sharing, including Tactical Digital Information Links (TADIL) [21] and Joint Variable Message Format (JVMF) [22].

The Access 5 Project [23], aimed at developing "recommendations for the safe and reliable integration and routine access of High Altitude Long Endurance Remotely Operated Aircraft Systems in the National Airspace System" offers opportunities for architectural standardization.

Given all these standardization activities, several issues will have to be addressed. A decision is needed regarding whether to coordinate amongst or actually consolidate multiple standards. Especially for command structure standards, it may be necessary to optimize them for robotic systems versus human soldiers. Another major issue is how generic to make messages. Depending on the domain (aerial, ground, underwater, etc.), it may be more efficient and effective to make the interfaces be domain-specific.

4.3 Architectural Components Standard

The ongoing standardization efforts in this category include SAE AS-4. The aforementioned JAUS Reference Architecture (RA) (Section 4.1) contains component specification. In this specification, the components are categorized into six types, namely, command and control, communication, platform, manipulator, and environment sensing. Each component has a unique identifier (ID) and performs a single function. Each component has a specified set of messages it sends and receives.

The updated version of JAUS RA contains a limited set of components. Components play a limited role in JAUS. They are not covered in the JAUS compliance plan, and, hence, are not enforceable. Another issue with the component specification is that the guidelines for functionally organizing components are not provided[1]. However, the authors regard these guidelines as essential architectural concerns.

There are other architectural efforts that emphasize component organization. The concept of an Observe, Orient, Decide, and Act (OODA) loop [24] has existed since the 1980s. The loop process features many lines of cross-referencing, feeding back, and feeding forward as well as rich interactions with the environment. The OODA loop is strikingly similar to the construct of a 4D/RCS node that includes the functions of sensory processing, world modeling, value judgment, and behavior generation. 4D/RCS takes a further step and describes that the functional loop be replicated throughout all the nodes of a system.

4D/RCS and OODA can serve as organizational frameworks for the JAUS components and messages.

4.4 Data model standards

Unmanned system control utilizes a large number of different data types. The data types include scalars, vectors, arrays, symbols, strings, pointers, lists, frames, and graphs. Information in the database includes signals, state variables, names, characters, numbers, attribute values, relationships, images, maps, rules, equations, and recipes. Knowledge includes structural and dynamic models that describe how the world behaves, and task knowledge that describes how to perform tasks, what tools to use, what resources are needed, and what information is required. State variables define estimated conditions in the world. Attributes describe properties. Entities and events can be represented in frames that contain lists of state variables, attribute values, and relationships. Images contain information about the position of entities in the world, and maps can provide geometric representations of terrain overlaid with labels, icons, and text that contain information necessary for situation assessment and planning of action. Therefore the standards requirements for data applying to unmanned systems are enormous. Some key applicable standards efforts are underway.

Geographic data is an area that has a lot of standardization efforts, although a lot of the results may need to be tailored to the computing requirements of the UMS real-time control. Federal Geographic Data Committee (FGDC) [25] is a U.S. Federal Government interagency committee for developing the National Spatial Data Infrastructure (NSDI), including geospatial data standards. The developed standards include:

[1]JAUS's coverage of this aspect is limited to a topological view, stating that a node contains a number of components.

- Spatial Data Transfer Standard (SDTS) - for sharing earth-referenced spatial data among heterogeneous computer systems [26]. SDTS also includes a Computer Aided Design and Drafting (CADD) Profile that supports exchange of geospatial data contained within CADD systems with other geoprocessing systems [27], and
- Vegetation Classification Standard [28].

National Geospatial-Intelligence Agency (NGA), formerly National Imagery and Mapping Agency (NIMA), maintains a set of geospatial data standards such as Universal Transverse Mercator (UTM) [29], Digital Terrain Elevation Data (DTED) [30], Raster Product Format (RPF), and Vector Product Format (VPF).

World geodetic system 1984 (WGS 84) [31,32], an earth-fixed global frame, is widely used. WGS 84 was adopted by the International Civil Aviation Organisation (ICAO) as its geodetic reference standard. NATO NSA took the approach of adopting many of these data model standards as its STANAGs [33]. There is also a National CAD (computer-aided design) Standard [34] that aims at facilitating the integration of building design data.

In terms of standards for detailed mechanical parts, the Standard for the Exchange of Product Model Data (STEP) is a comprehensive ISO standard (ISO 10303) that describes how to represent and exchange product information. STEP includes many "parts" ranging from mechanical design representations, systems engineering data representation, to product life cycle support. It is conceivable that some aspects could be extended for modeling the relevant objects in the UMS architectures.

All of these efforts are potentially useful to the problem of UMS control, yet, as stated before, a lot more standards may be needed to support the rich knowledge bases required for intelligent system control.

4.5 Execution Model Standards

Architectures should describe consistent models for execution, as those would support system integration efforts. 4D/RCS, for example, specifies the default model for execution as one in which all control nodes may execute concurrently. The intelligence is distributed among all the nodes in the system. Messaging within 4D/RCS should be non-blocking and reception of all messages should be acknowledged. Other execution models have been defined for unmanned systems. The key point is that a consistent standard is necessary for collaborating systems or intra-system components.

5 System Service Layer Standards

The standards at this layer facilitate the interoperability of unmanned system components. 4D/RCS applications should operate regardless of which system service layer standards the host computer implements. Several major efforts are underway in this area. These standards address similar technical issues. Therefore users should reference the one(s) that is/are most beneficial to their programs. Summaries for these standards are provided below.

U.S. Army Weapon System Technical Architecture Working Group (WSTAWG) Operating Environment (OE) Application Program Interface (API) specifications chartered to develop technical standards for war fighting embedded systems for the U.S. Army. One of the Integrated Product Teams (IPT) under WSTAWG develops the Operating Environment (OE) Application Program Interface (API) specification. The API set serves as a foundation for portable, distributed real-time embedded weapon systems applications.

The API set covers several areas. The operating system (OS) services form an operating system shell facilitating portability to various operating systems. The resource access services standardize and isolate the dependencies of the physical services layer from the OE system services. The covered issues include common interface definition, error handling, configuration, data distribution, timer, synchronization, etc.

The WSTAWG Weapon System Common Operating Environment (WSCOE) IPT aims at developing a common development environment consisting of standards based products to serve as a template for new or upgraded Army manned and unmanned weapon systems. At the time of this writing, 4D/RCS was being considered a part of the architectural base for WSCOE.

The SAE AS-4B Transport Specification defines the protocols employed for the transport of the UMSMS messages for all supported low layer protocols and media on a computer network. A preference is to use the Internet Protocol Suite (TCP/IP Suite) that employs the best-effort, unreliable delivery of the User Datagram Protocol (UDP) over Internet Protocol (IP) for most of the UMSMS messaging. Transmission Control Protocol (TCP) may be used for the assured delivery of a small volume of safety critical messages. 4D/RCS applications should operate if this standard is what the host computer implements.

The NATO STANAG 4586 covers messaging protocols. TCP/IP, HTTP, FTP, and Network Time Protocol were specified. A comparison between this standard and the SAE AS-4 Transport specification may be beneficial.

OMG Middleware [14] is a set of middleware specifications that aim at providing interoperability in heterogeneous computer systems that may involve different platforms, operating systems, and programming languages. The OMG specifications include Common Object Request Broker Architecture (CORBA) and Data Distribution Service (DDS).

6 Performance Metric Standards

Performance requirements are typically application specific. Architectural metrics must be specified according to the software and hardware performance requirements, for example, "cycle time of Y ms for the control nodes" and the functions or behaviors that the resulting electro-mechanical system is expected to perform, including issues such as accuracy and reliability, for example, to deliver supplies to an area of a particular size. However, it could be very beneficial if there were sets of standard definitions, metrics, and measurement processes that user could use to either specify their architectures or evaluate the systems under development. These standards could be established in a generic way but allow for instantiations applicable to specific systems.

There have been a lot of studies on software metrics. The relevant standards include [35, 36]. In addition, a collection of bibliography is provided in [37]. Mills, in [38], assessed a collection of software metrics that might involve measuring the size (lines of code, function points), complexity (nodes and branches in logical flows), quality (mean time between failure), level of effort, etc.

On the functional/behavioral performance, a general area that receives a significant amount of attention is the adaptability, robustness, and reliability of unmanned systems. Unmanned systems are anticipated to operate in uncertain and changing environments which may be unsafe for humans. As such, unmanned systems must be able to handle certain levels of uncertainty with certain levels of performance consistently. These give rise to the following two standards oriented projects.

6.1 Performance Metric Standards for Urban Search and Rescue (US&R) and Bomb Disposal Robots

The Science and Technology (S&T) Directorate of the Department of Homeland Security DHS has initiated an effort with the NIST to develop comprehensive, performance based standards related to the development, testing, and certification of US&R robotics [39] [40]. NIST has conducted a series of workshops attended by the first responders to conduct a US&R robot requirements analysis [**Error! Bookmark not defined.**]. The analysis also outlined test methods for the requirements. The test methods are a significant part of the resulting standards This project has been migrated to be under ASTM International and is designated as ASTM E54.08.01.

In a similar vein, National Institute of Justice has initiated an effort with NIST to develop performance standards for bomb disposal robots. Chapter 9 of this book describes these issues in further detail.

6.2 Autonomy Levels for Unmanned Systems Framework

NIST initiated and is coordinating an Autonomy Levels for Unmanned Systems (ALFUS) *ad hoc* working group effort. The goal is to define standard terms and metrics for specifying the autonomy requirements and for evaluating the autonomy capabilities of intelligent unmanned systems [41, 42, 43]. The group defined that the autonomy capability of UMS is measured by the missions that the UMS is capable of performing in certain types of environments and requiring certain types of human intervention. Adaptability to changes in mission operations and environmental conditions is essential. Three sets of metrics have been defined as follows:

- Mission complexity could be measured with the metrics of: levels of subtasking, levels of decision making, levels of collaboration, knowledge and perception requirements, planning and execution performance, independence level, situation awareness level, etc.
- Human independence can be measured with the metrics of: interaction time, interaction types, robotic initiation of communication, etc.
- Environmental difficulty can be measured through relative solution space, levels of dynamicity and understandability of the environment, etc.

Work is underway to define measuring scales for all of these metrics.

Autonomy levels can be considered as a subset of the general performance metrics for unmanned systems. The authors are also exploring extending the ALFUS model to facilitate measuring the general performance of intelligent systems. The authors believe that, conceptually, most of the ALFUS metrics sets are applicable to measure either autonomy levels or general system performance. What may need to be further developed includes different sets of scales. For example, the ability to avoid certain obstacles is a contributing factor both to the vehicle's autonomy level and to the vehicle's general performance specification.

7 Process and Tools Standards

Common processes and tools facilitate sharing of architectural components. This section assesses the status of relevant tools and process support standards.

7.1 Terminology

Terms and definitions are among the critical initial steps in architectural development efforts. They provide a common vernacular within the community for communication and for system description. The following are some of the published terminology reports:

- Department of Defense Dictionary of Military and Associated Terms [44]
- NATO STANAG 4586 Annex A, Terms and Definitions
- AIAA Recommended Practice - Terminology for Unmanned Aerial Vehicles and Remotely Operated Aircraft (R-103-2004e) [45]
- 4D/RCS Definitions [**Error! Bookmark not defined.**]
- Autonomy Levels for Unmanned Systems Terminology [46]
- ASTM F 1490 – 04a: Standard Terminology Relating to Search and Rescue
- ASTM F 2395 – 05: Standard Terminology for Unmanned Air Vehicle Systems

In addition, the aforementioned US&R Robotic standards effort is in the process of generating a terminology document, which is planned to be submitted to ASTM for standardization.

7.2 Development methods

Large, intelligent systems need to be developed in a systematic way, requiring sets of comprehensive guidelines or methodologies. The following are among the established methods:

- ANSI/GEIA EIA-632, Processes for Engineering a System
- AIAA Guide for Reusable Software: Assessment Criteria for Aerospace Applications
- Chapter 2 of this book describes a methodology for developing 4D/RCS architectural applications

7.3 Tools and Software Environments

UMS architectural modeling could utilize the various representation or language standards. Developers need to identify the key aspects of the systems, including tasks, data, and processes to be developed before selecting best suitable modeling tools or representations.

The following are some of the standards that are either available or evolving:

- OMG Interface Definition Language (OMG IDL), which is ISO/IEC 14750
- OMG Unified Modeling Language (UML) and Systems Engineering Modeling Language (sysML) [14]
- SAE Architecture Analysis & Design Language (AADL) [47]
- IDEF [48]
- Web ontology language (OWL) [49]

In addition, NIST researchers also experiment with several 4D/RCS tools that facilitate the architectural development. The tools include:

- Neutral Message Language (NML) [50,51]: a uniform Application Programming Interface (API) to the communication functions of the intelligent system components. NML employs a mailbox (fixed size) model and supports many widely-used protocols such as interprocess shared memory, interprocessor backplane global memory, and internet networking. NML support various communication modes such as queued and non-queued as well as blocking and non-blocking. NML provides language bindings for C++ and Java.

- RCS Design Tool: "A tool written as a java applet that allows programmers to create RCS applications graphically and generates source code [51,53]"

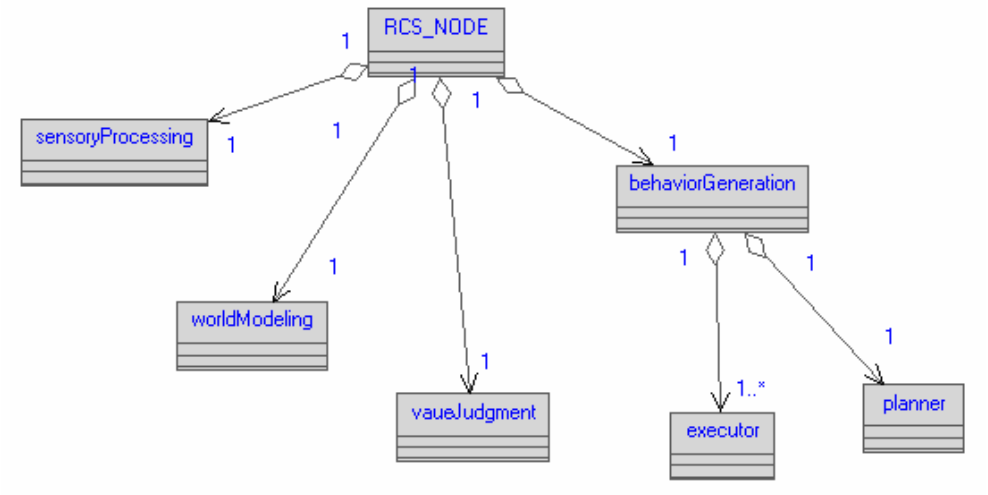


Figure 2: RCS Node Composition in UML.

A set of tools such as this one can help ensure that engineers develop control systems that are compliant with a given architecture. Such tools can inherently limit the choices of modules, messages, and the underlying communications and/or computational implementation to those that conform to a standard specification. The next section also provides an example, using generic templates, of a method to help assure compliance to a standard for UMS architecture.

8 A 4D/RCS Implementation Reference Model

Standards-based software tools accelerate the creation and enhancement of intelligent unmanned systems. Usage of tools also provides a basis for component reuse. Standards based commercial tools, such as those based on the Unified Modeling Language (UML) are being experimented with for the feasibility of facilitating a 4D/RCS development infrastructure. The following describes a set of generic templates that might be used for particular system development.

The following UML models were developed to represent 4D/RCS: RCS Node, shown in Figure 2, RCS Node Functions, shown in Figure 3, and RCS Node Functional Model, in Figure 4. Note that all the modeled functions are stubs that are intended for instantiation for the applications.

As described in the earlier chapters, an RCS node consists of four functions, namely, sensory processing, world modeling, value judgment, and behavioral generation. Behavior generation is further decomposed into the planner and executor functions (see Figure 2). All the node functions, as well as the node itself, can be implemented with sets of generic functions. Figure 3 illustrates the concept. RCS applications are distributive in nature. They can be executed cyclically on their own. The authors employed a base class, called cyclicProcess to realize this concept. In the cyclic process, a preprocess, a decision process, and a post process would be executed in order by the unitCycle function. The cycle time is specified by the cycleTime variable.

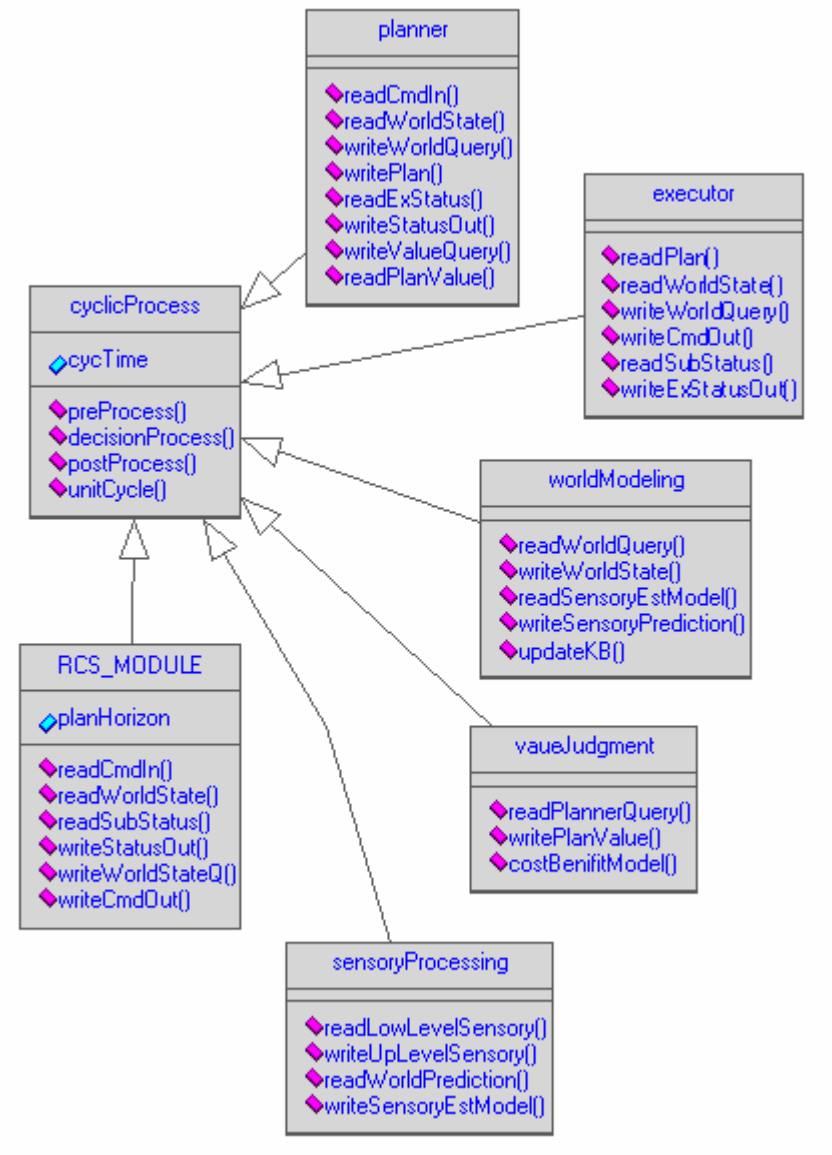


Figure 3: RCS Node Function Code Frames in UML.

All the node functions inherit the functionality of cyclicProcess. The node function templates contain additional functions, including reading from and writing to particular communication channels. These read and write activities described in the early chapters of this book and are illustrated in Figure 4.

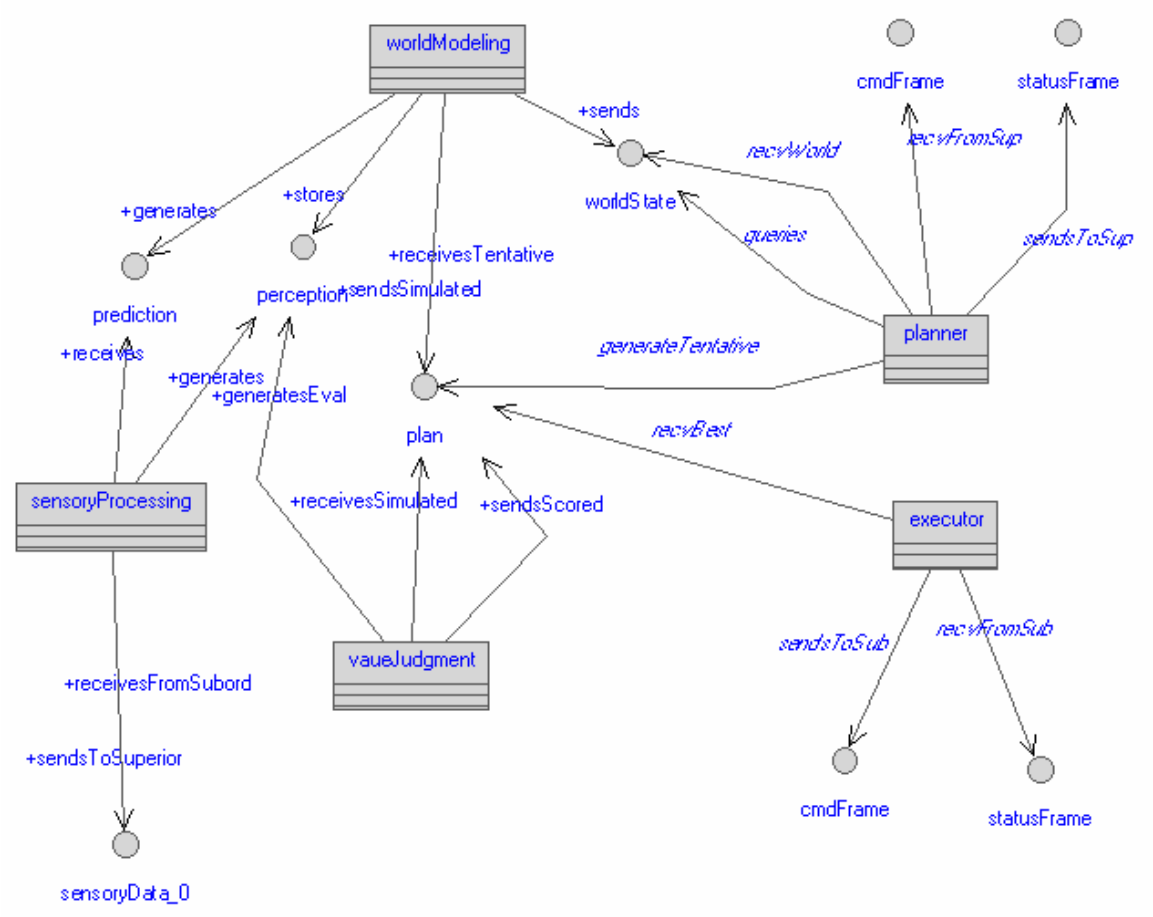


Figure 4: 4D/RCS Node Functional Model in UML.

9 Conclusion

An architectural standards framework based on 4D/RCS has been described. This framework was used to assess intelligent, unmanned system standards that were either established or being developed. The authors also identified *de facto* standards. Architectural standards needs were also highlighted through this assessment process. The authors' findings included that standards needs in the area of behavioral performance metrics might be worth its focus from the architectural community.

References

1 James Albus et al., 4D/RCS: A Reference Model Architecture For Unmanned Vehicle Systems, Version 2.0, NISTIR 6910, Gaithersburg, MD, 2002, http://www.isd.mel.nist.gov/projects/rcs/

2 Executive Office of the President of the United States, FY07 Budget Formulation FEA Consolidated Reference Model Document, http://www.whitehouse.gov/omb/egov/a-1-fea.html

3 DoD Architecture Framework Working Group, DoD Architecture Framework Version 1.0, http://www.defenselink.mil/nii/doc/DoDAF_v1_Volume_I.pdf, http://www.defenselink.mil/nii/doc/DoDAF_v1_Volume_II.pdf

4 http://www.iasahome.org/iasaweb/appmanager/home/home

5 Carnegie Mellon Software Engineering Institute, Software Architecture for Software-Intensive Systems http://www.sei.cmu.edu/architecture/index.html
6 IEEE Recommended Practice for Architectural Description of Software-Intensive Systems ANSI/IEEE Std 1471, http://www.pithecanthropus.com/~awg/public_html/browsing_library.html
7 Foundation for Physical Intelligent Agents, FIPA Abstract Architecture Specification, http://www.fipa.org/specs/fipa00001/SC00001L.html#_Toc26668600, December 3, 2002.
8 United States Department of Transportation, National ITS Architecture Version 5.1, http://www.iteris.com/itsarch/
9 American National Standards Institute, United States Standards Strategy, http://ansi.org/standards_activities/nss/usss.aspx?menuid=3, December 8, 2005.
10 http://www.orocos.org/index.html
11 http://orca-robotics.sourceforge.net/index.html
12 http://www.autosar.org/find02_ns6.php
13 R. Volpe, et al., "The CLARAty Architecture for Robotic Autonomy," IEEE Aerospace Conference, Big Sky, Montana, March 10-17, 2001, http://robotics.jpl.nasa.gov/
14 http://www.omg.org/
15 http://www.opengroup.org/
16 http://opcfoundation.org/Default.aspx/01_about/01_whatis.asp?MID=AboutOPC#top
17 IEEE Standards Style Manual
18 ANSI/IEEE Std 1471-2000, Recommended Practice for Architectural Description of Software-Intensive Systems, IEEE, New York, New York, 2000
19 Air Force Doctrine Document 1-1, 12 August 1998
20 U.S. Army Field Manual No. 7-15.
21 http://www.fas.org/irp/program/disseminate/tadil.htm
22 Dalrymple, E., "U.S. Army Develops High Quality, Extremely Low Cost Digital Message Parser," Software Technology Support Center Magazine, www.stsc.hill.af.mil/crosstalk/2002/02/dalrymple.pdf , February 2002
23 Access 5 Project, Functional Requirements Document for High Altitude Long Endurance (HALE) Unmanned Aircraft Systems (UAS) Operations in the National Airspace System (NAS), January 2006.
24 http://www.belisarius.com/boyd.htm
25 http://fgdc.gov/
26 http://mcmcweb.er.usgs.gov/sdts/whatsdts.html
27 http://www.fgdc.gov/standards/documents/standards/sdts_cadd/
28 http://www.fgdc.gov/standards/documents/standards/vegetation/vegclass.pdf
29 http://erg.usgs.gov/isb/pubs/factsheets/fs07701.html
30 National Imagery and Mapping Agency, http://www.fas.org/irp/program/core/dted.htm
31 http://www.wgs84.com/
32 http://home.hiwaay.net/~taylorc/toolbox/geography/geoutm.html
33 STANAG 4586, Annex B, section 4.3
34 http://www.nationalcadstandard.org/whyncs.html
35 Dictionary of Measures to Produce Reliable Software. IEEE Std 982.1-1988, New York, April 1989
36 A Software Quality Metrics Methodology. IEEE Std 1061-1992, New York, March 1993
37 http://irb.cs.uni-magdeburg.de/sw-eng/us/bibliography/bib_main.shtml
38 Mills, E. E., Software Metrics, SEI curriculum Module SEI-CM-12-1.1, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, 1988.
39 http://www.isd.mel.nist.gov/US&R_Robot_Standards/
40 Messina, E. and Jacoff, A., "Performance Standards for Urban Search and Rescue Robots," Proceedings of the 2006 SPIE Defense and Security Symposium, Orlando, FL, April 2006.
41 http://www.isd.mel.nist.gov/projects/autonomy_levels/
42 Hui-Min Huang, Kerry Pavek, James Albus, Elena Messina, "Autonomy Levels for Unmanned Systems (ALFUS) Framework: An Update," Proceedings of the SPIE Defense and Security Symposium 2005, Conference 5804, Orlando, Florida, March 2005.
43 Hui-Min Huang, Elena Messina, Ralph English, Robert Wade, James Albus, and Brian Novak, "Autonomy Measures for Robots," Proceedings of the 2004 ASME International Mechanical Engineering Congress & Exposition, Anaheim, California, November 2004.

44 Department of Defense Joint Publication 1-02, http://www.dtic.mil/doctrine/jel/doddict
45 http://www.aiaa.org/content.cfm?pageid=363&id=1189&Type=StoreProduct&LayerID=51
46 Huang, H., Autonomy Levels for Unmanned Systems Terminology, NIST Special Publication 1011, Version 1.1, http://www.isd.mel.nist.gov/projects/autonomy_levels/terminology.htm
47 http://www.aadl.info/
48 Mayer, Richard J., et al., *IDEF Family of Methods for Concurrent Engineering and Business Re-engineering Applications*, Knowledge-Based Systems, Inc., 1992
49 http://www.w3.org/2004/OWL/
50 Shackleford, W., Proctor, F.M., Michaloski, J.L., *The Neutral Message Language: A Model and Method for Message Passing in Heterogeneous Environments*, Proceedings of the 2000 World Automation Conference, Maui, HI, June 11 - 16, 2000, (2000)
51 http://www.isd.mel.nist.gov/projects/rcslib/
53 Veysel G., Moore, M., Passino, K., Shackleford, W., Proctor, F., and Albus, J., *The RCS Handbook: Tools for Real-time Control Systems Software Development*, Wiley Series on Intelligent Systems, John Wiley & Sons, Inc., June 2001.

Chapter 9

Performance Evaluation of Autonomous Mobile Robots

Elena Messina, Adam Jacoff, and **Harry Scott**
National Institute of Standards and Technology (NIST)
{elena.messina,adam.jacoff,harry.scott}@nist.gov

1. Introduction

As the nation's metrology laboratory, NIST has been involved in measuring performance of various technologies related to unmanned ground vehicles for decades [23] [11] [12]. A comprehensive effort was initiated in 1999 to develop measures of performance for intelligent systems. This primarily, but not exclusively, focused on the software within the systems. Emphasis was placed on unmanned ground vehicles, since that was a technology with which NIST had extensive experience. Over the years, NIST has been collaborating with Army Research Laboratory, DARPA, DOT, and other organizations to develop focused measures of performance for unmanned systems and subsystems. This chapter discusses the overall philosophy guiding the performance evaluation work at NIST, describes the required infrastructure to support a measurement program for unmanned ground vehicles, and provides examples of applied performance evaluations.

The approach espoused by NIST in evaluation of mobile robots is based on both cognitive principles and domain-specific requirements. In this chapter, \ the terms "intelligent system" and "cognitive system" are used interchangeably. A definition of cognition is "the mental acquisition of knowledge through thought, experience, and the senses" [18]. Basic understanding of animal and human cognition has helped shape the 4D/RCS architecture [3]. Therefore the approach to evaluating systems developed using 4D/RCS exercises basic functions that have a cognitive foundation: sensing and modeling the world and planning behavior to achieve goals. Equally important is the development of tests that are driven by the constraints of the domain that the system must function within and by the tasks that the system must be able to accomplish. Both the cognitive and task-based performance evaluation approaches are useful in measuring the performance of non-RCS-based systems. Regardless of their underlying architecture, systems must have some capacities for sensing their environment, creating a model of what the state of the world is, and planning – either reactively or deliberatively or both. And of course, ultimately, systems must be able to accomplish the tasks that they are required to perform.

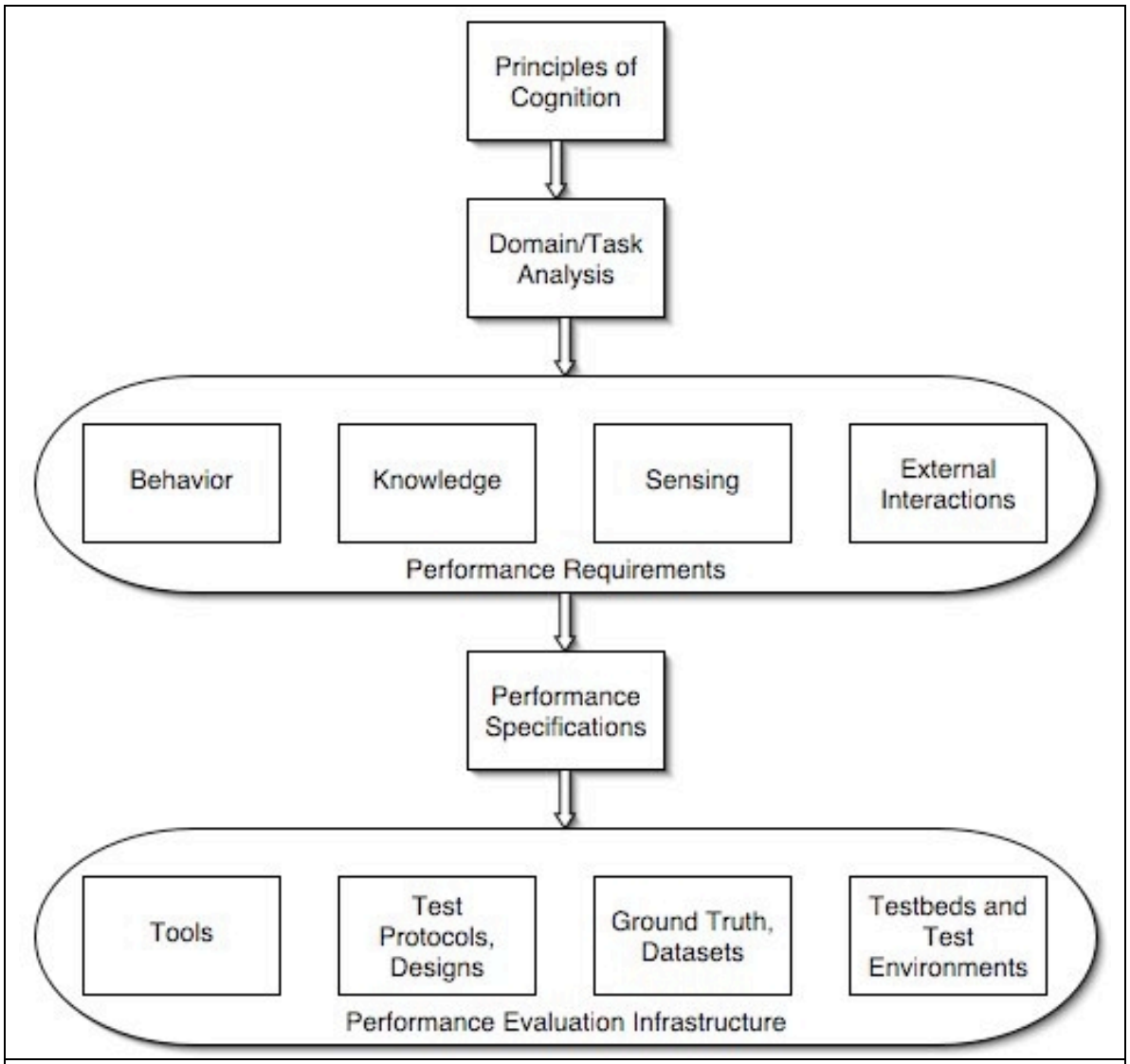


Figure 1: The Performance Evaluation Process for Intelligent Autonomous Vehicles. The overall process starts with principles of cognition (concepts about perception, types of memory, feedback loops, etc.) From those basic principles, one can derive a cognitive architecture such as 4D/RCS (not shown in this diagram). The cognitive foundations can be used to structure a domain/task analysis. Starting with the types of tasks that a system will be expected to perform, the behaviors it will need to have are derived, followed by the knowledge requirements to support the behaviors (i.e., what will it need to know about in order to make decisions – both deliberative and reactive). The knowledge requirements drive the sensory perception requirements and the system's external interactions, be they with other agents or humans. The output of this step is a set of performance specifications for the overall system and for subsystems, including sensors, world models, planners, etc. An infrastructure is needed to support evaluations of the system. Tools in this infrastructure include simulations and data capture. The performance specifications are used to derive test designs and test methods and set the requirements for datasets, ground truth, arena designs, and other supporting artifacts.

Figure 1 shows a high-level picture of the performance measurement process and its various stages. The process begins with an understanding of the principles of cognition. This foundation provides guidance in the development of an implementation architecture and methodology, such as 4D/RCS, which is discussed in-depth elsewhere in this book, as well as in the derivation of the tests. The methodology includes a task and domain-based analysis, which results in behavioral, knowledge, sensing, and other system requirements. The supporting infrastructure for performance evaluation, comprising test

designs/protocols, tools including simulators and visualizers, ground truth and sensor datasets, and testbed environments. In this chapter, examples are provided of how this process has been applied by NIST and ARL to evaluate performance of unmanned vehicles.

A methodology for deriving the control architecture for a mobile robot (or other complex system) is described more fully in Chapter 2. A parallel methodology is applied to derive appropriate performance metrics for the system. The relevant point for performance evaluation is that the derivation of the evaluation procedures is also driven by the requirements of the tasks that the system is to perform. The design and the evaluation process both begin with a rigorous analysis of what the system is supposed to do. Measures and constraints are defined at this time, ideally in close consultation with the end users and domain experts (also known as subject matter experts or SMEs). The system's overall capabilities, such as being submersible or able to carry a payload of a certain weight, are key aspects that have to be evaluated. However, these physically-based performance tests are relatively straightforward to design and carry out. More challenging to evaluate are the individual tasks and overall missions that the system is to perform. Is the system supposed to be able to support a route reconnaissance or try to find victims in a collapsed building? The mission definition and environmental conditions lead to specific and unique performance requirements. SMEs are asked to describe what they themselves do, and to start defining what they would like the robot to be able to do as well. This is especially true in the vast majority of instances where the robot is not a replacement for the human, but rather is intended to assist him/her or augment their capabilities. The evaluations span performance by the entire system as well as individual components. It is essential to be able to characterize the subsystems, including sensing and locomotion, as well as the entire system, in order to understand the capabilities and limitations that will be encountered when deploying a system.

Besides defining metrics and tests for measuring the system's ability to accomplish overall mission goals and individual tasks, it is essential to be able to isolate subsystems and measure their individual strengths and weaknesses. If the system fails to perform a task or a mission under test, the engineers need to understand the exact mechanism or chain of events that led to the failure. This can be accomplished by several means. Capturing results from the system under test and comparing them to ground truth is a central strategy in the performance evaluation of autonomous vehicles. The trace of a vehicle's path and its own representation of the world and its location within it can be compared to actual positions in the world through the use of methods described in Section 3.

Having well-characterized components and subsystems is also necessary for enabling reuse of existing work in new implementations. System engineers can determine whether a particular component is suited to their application or even choose among various candidate approaches if there is quantitative data on the performance of the component under known conditions. Targeted subsystem tests, either standalone or as part of an overall test arena, can provide insights into the particular capabilities of sensors or algorithms.

This chapter is organized as follows. Section 2 contains a discussion of testbeds that NIST has designed to evaluate the performance of mobile robots. The testbeds encompass portable, physical arenas as well as virtual representations. A discussion of the approaches used to measure performance of urban search and rescue research robots through the use of testbeds and competition metrics is included in this section. Testing approaches for evaluating the capabilities of robots to be potentially fielded for homeland security applications, such as urban search and rescue and bomb disposal conclude Section 2. Section 3 discusses infrastructural capabilities for capturing performance data. In particular, tracking technologies are described for indoor and outdoor applications. Section 4 gives a detailed overview of the Technology Readiness Level (TRL) assessment that was performed on the Autonomous Mobility component of the ARL Demo III XUV. This assessment is a distillation of the methodology and infrastructure developed at NIST for evaluating performance of unmanned vehicles. Section 5 briefly touches on some additional, more theoretical performance measurement efforts that involve the greater community of researchers and practitioners. Chapter conclusions are listed in Section 6.

2. Testbeds and Test Methods

Measuring the performance of robots in controlled, repeatable, and reproducible ways is an essential requirement. Controlled experiments refer to being able to characterize and manage the conditions under which the system is tested. For example, the amount of moisture in the air, temperature, and lighting may be important to control. Repeatability of an experiment refers to being able to conduct the experiment under the exact same conditions (both the external environmental ones) and with the same starting state of the system each time. Reproducible means that the tests and related artifacts and instrumentation are defined adequately enough that another organization can duplicate the tests elsewhere. This can be accomplished through the use of specifically engineered testbeds. This section describes testbeds that have been designed by NIST to measure robot performance in targeted application domains. Metrics and instrumentation to capture performance results are an integral part of the testbed approach. The testbeds described are only a subset of a spectrum possible. Shown in Figure 2, there is a range of testbeds that includes virtual ones (described below), qualifying arenas (described in this section), reality arenas (some of which are described in Section 2.3), and outdoor arenas.

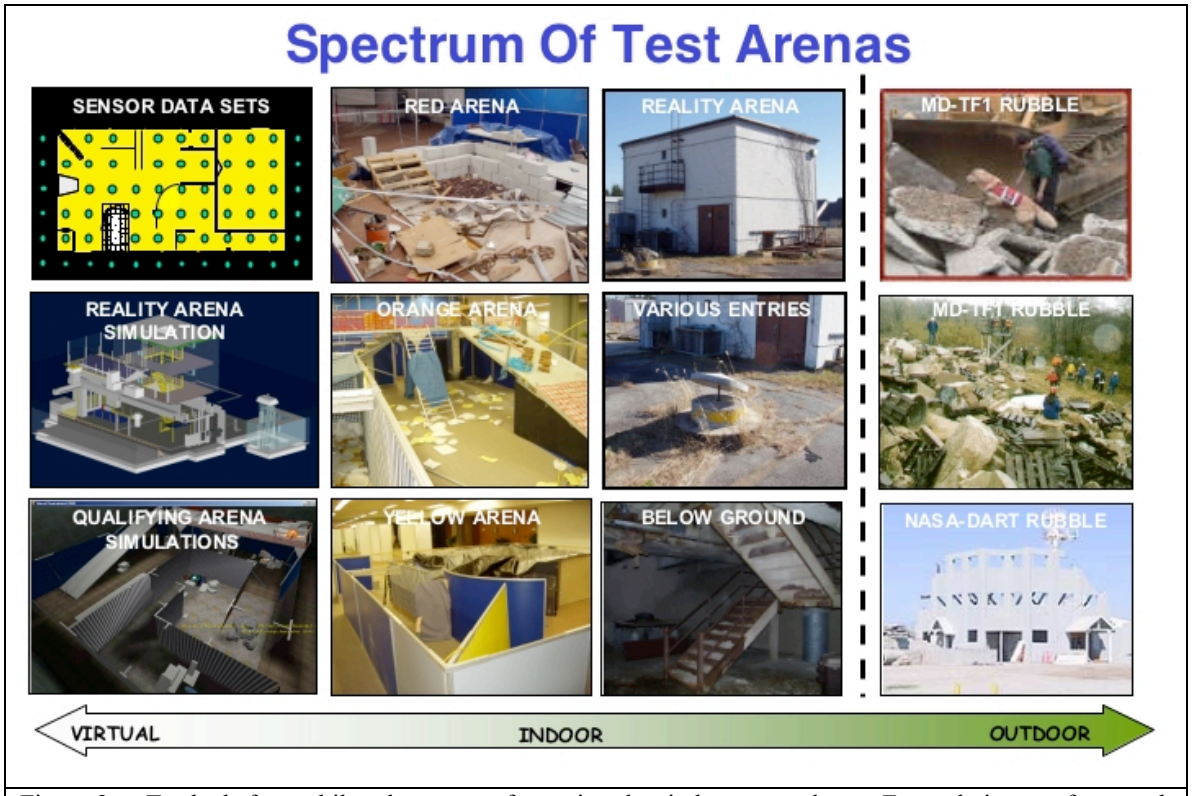


Figure 2: Testbeds for mobile robots range from virtual to indoor to outdoor. Example images from each category are shown.

2.1 Testbeds for Urban Search and Rescue Robot Competitions

Partly supported initially by the DARPA Tactical Mobile Robots (TMR) and later by the Mobile Autonomous Robots Software (MARS) program, NIST became involved in a mobile robot competition hosted by the American Association for Artificial Intelligence (AAAI) starting in 2000. The competition encouraged participants to contribute to the field of Urban Search and Rescue (USAR) robotics and provided the competitors with a sense of what a real USAR situation involves. The goal of the Rescue Robot Competition was to increase awareness of the challenges involved in search and rescue applications, provide objective evaluation of robotic implementations in representative environments, and promote collaboration between researchers. It required robots to demonstrate their capabilities in mobility, sensory

perception, planning, mapping, and practical operator interfaces, while searching for simulated victims in a maze of increasingly difficult obstacles. The competition rules and format were adopted by the RoboCup international organization [15] [4], which placed the event in an international forum.

The competitions were hosted within the Reference Test Arenas for Urban Search and Rescue Robots, which were developed by NIST and were proliferated globally [13]. The arenas provided a continuum of increasingly difficult representations of collapsed buildings that had simulated victims scattered throughout. There were three areas of increasing difficulty within the arenas labeled yellow, orange, and red, with red being the most challenging. They were also referred to as "qualifying arenas" since a robot had, at minimum, to traverse all three arenas without difficulty before being considered qualified to attempt more realistic test environments.

Figure 3 shows images of versions of the arenas. A maze of walls, doors, and elevated floors provided various tests for robot navigation and mapping capabilities. Variable flooring, overturned furniture, and problematic rubble provided obvious physical obstacles. Sensory obstacles, intended to confuse specific robot sensors and perception algorithms, provided additional challenges. Intuitive operator interfaces and robust sensory fusion algorithms were highly encouraged to reliably negotiate the arenas and locate victims.

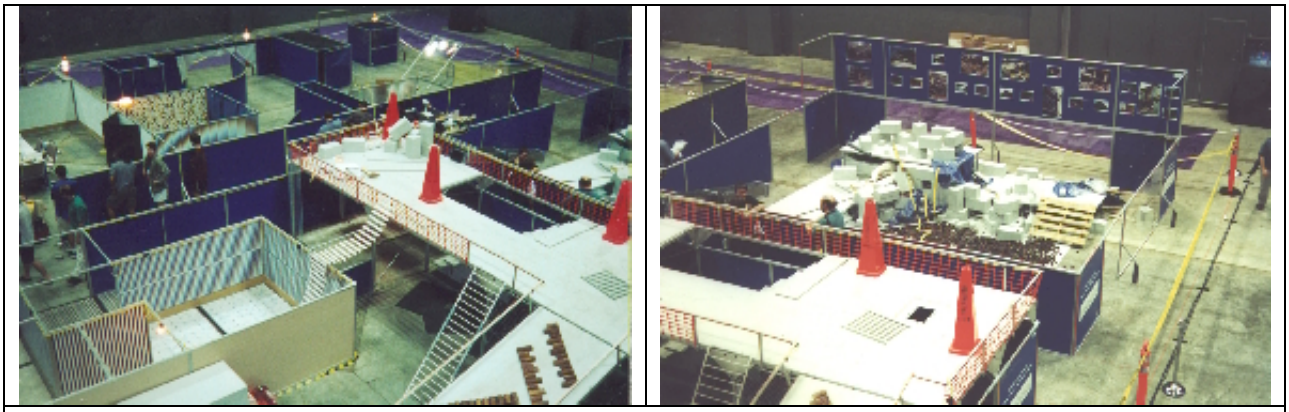


Figure 3: NIST Reference Test Arenas for Urban Search and Rescue Robots, shown as deployed in a competition.

Robot teams were scored according to a performance metric that quantified robot capabilities pertinent to USAR applications for the purposes of comparison between diverse robotic implementations and team strategies. The rules of these competitions evolved each year to encourage robots to negotiate complex and collapsed structures, find simulated victims, determine their condition and location, and generate human readable maps to enable victim recovery [14]. The associated performance metric also evolved as it attempted to quantify and encourage these and other robot capabilities pertinent to urban search and rescue applications. The rules and performance metric focused on the basic USAR tasks of setting up an operator station, safely negotiating the complex and collapsed structures in the arenas, allowing clear comparisons of diverse robotic implementations, and demonstrating reasonable operational stamina.

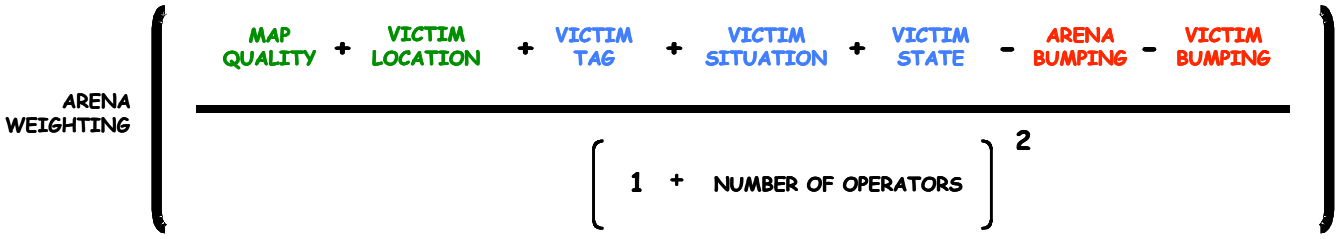


Figure 4: The performance metric used for scoring competitions

Figure 4 shows the performance metric for the competitions. Perception of detailed victim information through multiple sensors was encouraged by the metric. It also encouraged teams to minimize the number of operators, which could be achieved through use of better operator interfaces and/or autonomous behaviors that allowed effective robot control of multiple robots. Finally, arena weightings accounted for the difference in difficulty negotiating each arena. The more difficult the arena, the higher the arena weighting (score) for each victim found.

- Up to 50 points were available for each victim found based upon a variety of factors. However, points were also deducted for errant victim identifications or uncontrolled bumping of victims or arena features. The penalties were meant to encourage confidence in reported results and promote safe operation within dangerous environments. The performance metric's point allocations were as follows:
    - Up to 20 points per victim applied toward a paper-based map of the environment generated immediately after the completion of each mission. Map quality and accuracy (victim location) were each worth up to 10 points.
    - Up to 15 points per victim applied toward sensory perception of simulated victims to reward individual sensor capabilities and correct interpretation of particular combinations of sensor readings. These included correct identification of victim tag (a numeric ID placed on the simulated victim or in its vicinity), definition of victim's situation (lightly trapped, in a void space, or entombed), and its state (conscious, semi-conscious, or aware).
    - Up to 15 points per victim apply toward demonstrating advanced mobility via remote teleoperation or other control modes. These points were awarded through correct identification of the victim tag and victim situation, both of these requiring not just sensory perception capabilities, but sufficient maneuverability and agility for the robot to circumnavigate the location of the victim in order to appropriately assess its situation (by viewing its surroundings) and finding its tag, which was placed in places near the victim that were not readily accessible.
    - Penalties were assessed for each instance in which the robot exhibited uncontrolled bumping of arena features (-5 points) or of the victims (-20 points).
- To discourage a large number of operators being required, the total number obtained from the previous scoring measures was squared in the above equation. The number of operators counted any person who entered the operator station during a mission.
- Finally, the overall resulting points were weighted by arena factors, which accounted for the difference in difficulty negotiating each arena; the more difficult the arena, the higher the arena weighting (score) for each victim found. Factors were 0.5 for victims found in Yellow, 0.75 for those found in Orange, and 1.0 for those found in Red.

The Reference Test Arenas for Urban Search and Rescue Robots were produced by applying the methodology described above in the introduction. A task-specific understanding was developed for the performance requirements. In this case, the robot had to perform tasks to aid human rescuers in exploring collapsed and compromised buildings and trying to locate victims. The robots were presented with a series of challenges that were abstractions of a real-world collapse and that were isolated (especially within the yellow and orange arenas) so as to enable researchers to understand their system's strengths and

weaknesses. Having an immature robot fail when trying to explore a very complex and realistic collapsed building representation does not yield any insights into what caused the failure and what needs to be improved – other than "it was too hard." The robots were to find victims that exhibited a variety of signs of life. The redundancy in the victim's sensor signatures mirrored the various signs of life that humans emit and encouraged the fusion of multiple sensors by the robot developers. The combination of available signs of life not only indicated the presence of a victim, but also their state: unconscious, semi-conscious, or aware. The victims were placed according to statistics from actual earthquakes: surface (50%), lightly trapped (30%), void (15%), or entombed (5%) [17].

To gain understanding of the characteristics of the robot's components, there were tests that isolate the individual elements. For example, the arenas included a variety of distinct challenges aimed at particular types of sensors. Some obstacles were clear acrylic, making them invisible to color or black and white cameras, but not necessarily to certain range sensors. Other obstacles were made of acoustic tiles, which absorbed emissions by sonar sensors. Wall coverings with regular patterns, such as vertical stripes, were used to confuse stereo algorithms. A system's ability or inability to deal with these particular obstacles revealed the strengths and weaknesses of the different approaches.

As the capabilities of the robots matured, the challenges within the competitions and hence the arenas evolved. Communication dropouts, which are pervasive in the real world, were introduced. More focus on robot autonomy led to new competition rules that enforced totally independent operation by the robots within certain portions of the arena. The competition rules and arena configurations continued to evolve as the technologies and algorithms matured and approached being deployable.

2.2 Virtual Testbeds

Virtual versions of NIST's Reference Test Arenas for Urban Search and Rescue Robots were developed to provide the research community with an efficient way to test algorithms independent of the costs associated with maintaining functional robots and traveling to one of the permanent arena sites for validation and practice. The effort was two-fold and covered two of the infrastructural elements for performance evaluation: sensor datasets and simulations. Both of these elements were aimed at providing researchers with the ability to develop and evaluate algorithms without needing access to robot hardware, including sensors, or to the physical instantiations of the reference test arenas.

Sensor data sets that were captured systematically within the actual arenas were made available to researchers to allow them to process arena sensor data during the development of their algorithms, while freeing them from the costs and maintenance associated with real robot hardware. Data from range-imaging sensors such as LADARs and from digital color cameras were captured. The virtual arena helped quicken the pace of software development and evaluation while maintaining a direct correlation to the real world, representative environment. Once a researcher's software was tested and deemed likely to succeed, it was downloaded to an actual robot equipped with the exact same sensors used to capture the original data set, and allowed to run in the actual arenas.

The University of Pittsburgh and Carnegie Mellon University developed USARSim, a realistic simulation of the Reference Test Arenas for Urban Search and Rescue Robots using a game engine graphics environment [21] [22]. This pseudo-dynamic simulation of NIST's arenas supported hardware-independent algorithm development with simulated sensor signatures and added the ability to virtually design and test new robotic mechanisms and sensor configurations. Figure 5 shows an image of the Orange arena modeled within USARSim. Starting in 2006, the RoboCup Rescue organization initiated a new Rescue competition using USARSim-based venues.

Figure 5: NIST Orange Arena modeled in USARSim.(courtesy of University of Pittsburgh and Carnegie Mellon University).

USARSim was integrated into a broader infrastructure that supported system and component evaluations. The Mobility Open Architecture Simulation and Tools (MOAST) environment conformed to the NIST 4D/RCS architecture and allowed simulated and real architectural components to function seamlessly in the same system [7]. This permitted not only the development of individual components, but also allowd component performance metrics to be developed and for the components to be evaluated under repeatable conditions. The environment was composed of high-fidelity and low-fidelity simulation systems, a detailed model of real-world terrain, actual hardware components, a central knowledge repository, and architectural glue that tied all of the components together. The need for a simulation environment is critical to the development, testing, and analysis of algorithms for mobile robotics. Simulation has benefits that include reduced competition for scarce resources (i.e. multiple agent configurations can be preset and remain constant), no risk of harm to personnel or equipment, the ability to add as yet undeveloped capabilities to subsystems, and the ability to perform repeated tests over vast and varied terrains. In a properly configured simulation environment, an individual code module or agent can be tested and understood before moving to real hardware.

The MOAST environment strove to seamlessly integrate simulation subsystems with real robotic hardware subsystems. The goal was to allow the individual subsystems to each perform in the area where and when they did best. For example, simulation systems could replicate multiple platforms for the development of multi-platform behaviors. They allowed for repeatable events, and provided detailed system/event logging. In addition, by simulating the results of sensor processing, the potential benefits of detecting new features or utilizing novel sensing paradigms could be measured. However, as noted, there is no substitute for real mobility, sensing, and communications. Therefore, when available, real system components/subsystems were able to plug into the MOAST environment and replace simulated subsystems. This was made possible through the architectural glue of the environment. This glue included a reference model architecture with well-defined interfaces and communications protocols, and detailed specifications of individual subsystem input/output. The 4D/RCS reference model architecture was selected for the MOAST reference model architecture. All communications between modules was accomplished over Neutral Messaging Language (NML) channels that functioned as the communication medium. By minimizing the jump from the virtual to the real world arenas, and allowing hardware-independent testing of concepts, this tool quickened the pace of development of more capable systems. The ability to repeat experiments under controlled

conditions and to capture results from algorithms and subsystems (which could be output via NML channels and easily written to files) greatly facilitated performance evaluation of a system or component.

2.3 Testbeds and test methods for homeland security robots

The performance evaluation approach undertaken for the USAR robot competitions was a platform for expansion into other areas. Lessons learned were applied to technology readiness level assessments, described in Section 4. Outside sponsors asked NIST to develop performance metrics and standards for USAR robots and for bomb-disposal robots. Programs to address these needs led to the development of requirements, test methods, and test environments that lie closer to the reality arenas in the spectrum illustrated in Figure 2.

Both of these programs closely followed the performance evaluation process in Figure 1. SMEs (in this case urban search and rescue task force members and bomb squad members) defined the tasks that the robots are to perform and set the performance objectives and thresholds. The deployment mission, environmental conditions, logistical constraints, and other considerations all helped define the performance requirements. Test protocols, artifacts, and environments were defined based on these requirements. These tests were aimed at the component or function level. For overall system evaluation, exercises in which the robots were run by responders in representative environments were also part of the testing program. These were akin to the technology readiness level assessments, which are described in Section 4 below.

The Department of Homeland Security initiated a program in 2004 to develop performance metrics and standards for USAR robots. This led to an in-depth definition of the performance requirements for robots by the intended user community. Federal Emergency Management Agency (FEMA) Task Force members were the principal contributors in the definition process. The initial set of performance requirement categories is shown in Table 1. The initial set of individual requirements within these categories totaled over 100, but continued to grow. Performance objectives and thresholds were defined for each individual requirement within a category. These objectives and thresholds varied according to the robot platform or deployment category, which is another set of domain-specific definitions developed in conjunction with the end users (SMEs). Over a dozen individual robot or deployment categories were identified, spanning the ground, aerial, and underwater domains. Test protocols and artifacts were developed to measure the performance of robots per their category's ranges. For example, speed requirements were much lower for peek-bots, are small, throwable robots that are meant to have limited mobility, than for wide-area survey robots that cover large distances quickly. Knowing the target platform's intended use is critical to devising the testing protocols and supporting apparatus.

Table 1: Main USAR Robot Performance Categories

Human-System Interaction	Pertaining to the human interaction and operator(s) control of the robot
Logistics	Related to the overall deployment procedures and constraints in place for disaster response
Operating Environment	Surroundings and conditions in which the operator and robot will have to operate
Safety	Pertaining to the safety of humans and potentially property in the vicinity of the robots
System:	Overall physical unit comprising the robot. This consists of the sub-components below:
- Chassis	The main body of the robot, upon which additional components and capabilities may be added. This is the minimum set of capabilities (base platform).
- Communications	Pertaining to the support for transmission of information to and from the robot, including commands for motion or control of payload, sensors, or other components, as well as underlying support for transmission of sensor and other data streams back to operator

- Mobility	The ability of the robot to negotiate and move around the environment
- Payload	Any additional hardware that the robot carries and may either deploy or utilize in the course of the mission
- Power	Energy source(s) for the chassis and all other components on board the robot
- Sensing	Hardware and supporting software which sense the environment

Figure 6 is an example of the type of test that was developed for USAR robots. Based on the requirements from the various intended end users, a test method using eye charts (or other "standard" visual cues) positioned at various locations on a wall near the floor and near the ceiling, both in front of the robot and to its side is shown in this figure. Conditions, such as the lighting, communications signal quality, and distance of robot to the chart, as well as the orientation of the robot, are varied. From this basic set up several permutations are possible. Measurements taken include the smallest line of text that can be read by the operator from any given position and the time required to be able to complete the successful reading of this line.

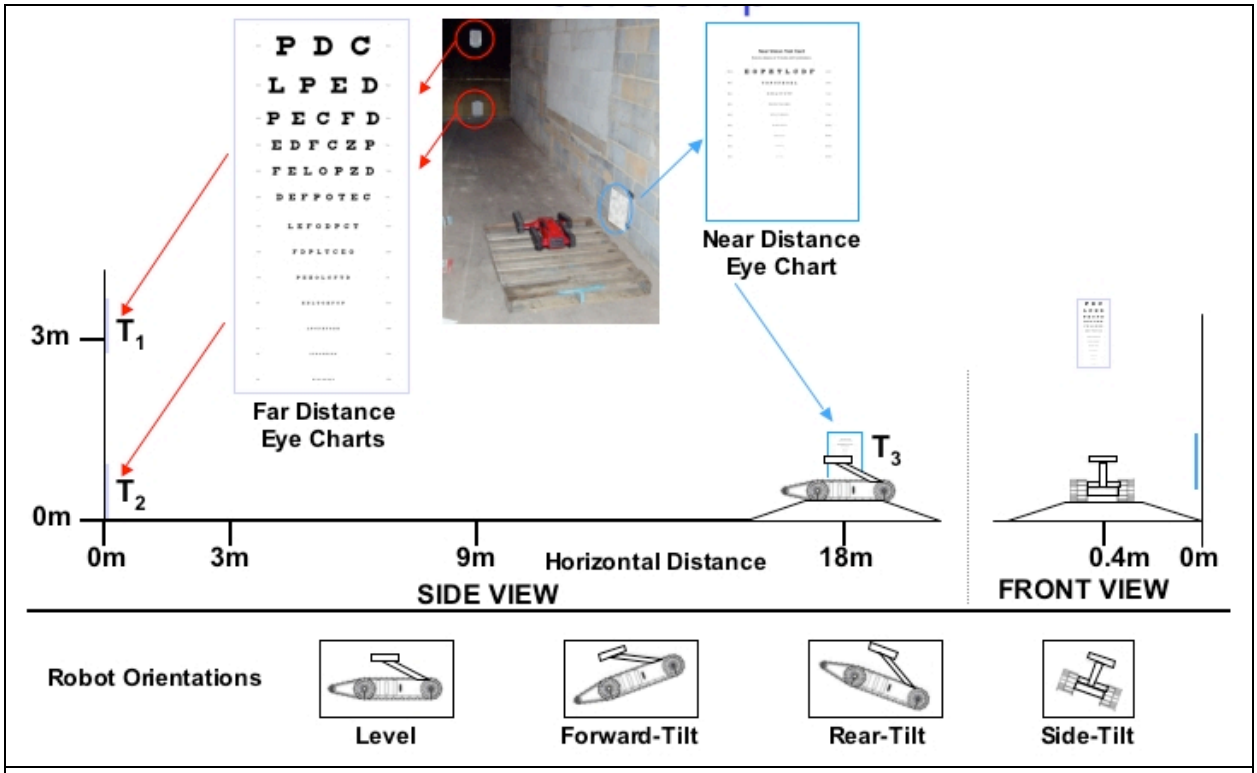


Figure 6: Visual Acuity Test for Mobile Robots. Robot is positioned by operator (remotely located) at various specified locations. The smallest line of text legible by the operator is noted. The robot's orientation is varied. Eye charts are placed at different elevations in front of the robot as well as to its left or right. The amount of time necessary for the operator to identify the chart and read its text is also noted, as this is an indicator of the ease of positioning of the robot's sensors. Lighting conditions may also be controlled.

Another major homeland security robot performance evaluation effort at NIST was aimed at bomb disposal applications, under sponsorship by the National Institute of Justice. Although already widely deployed by

civilian bomb squads and the military, there were no standard performance evaluations for these robots. NIST worked with civilian and military subject matter experts to define the tasks and acceptable ranges of performance by these robots. Numerous new artifacts and challenges were added to the existing NIST reference test arenas for mobile robots addressing the bomb disposal task. A large number of the additions provided manipulation challenges, as bomb disposal robots have arms and must handle a variety of objects, as well as aim and fire disruptors at suspected explosive devices. Figure 7 shows two of the bomb-disposal evaluation tests.

Figure 7: Bomb Disposal Test Examples. The robots must search a variety of locations for suspicious devices (potential bombs). In some cases, they must manipulate the environment, as shown on the right, where the robot lifts the couch in order to be expose the device to the cameras that the operator is viewing.

2.4 Discussion Regarding Design of Testbeds

It is hard to define a formal methodology by which to translate an application domain's challenges into testbed elements. For the USAR testbeds used in competitions, described in Section 2.1, aspects that were critical to the performance of the robots were informally enumerated by experts who had experience in robotics and search and rescue. Prior knowledge of what challenged robot sensing, mobility, planning, as well as operator situation awareness was used to generate various elements of the testbeds. For example, sensor tests within the arenas were meant to expose weaknesses commonly found in particular sensors and algorithms and in the process, educate the research community how to avoid those pitfalls. Clear partitions and mirrors confused operators if they were driving the robots through a maze-like arena.

The types of tasks that the robot is to perform must be understood in order to design appropriate tests. If the robot is to search for victims, then the tests must provide representative victims. The types of features that identify a potential victim include: human form and/or clothing, sound, movement, heat, and carbon dioxide. Therefore the testbeds included elements that exhibited a combination of these features. The robot developers should in turn incorporate the appropriate sensors for achieving the task. For example, knowing that heat emissions are present where there are victims should encourage robot developers to include sensors that detect heat. Of course, not all instances where there is heat necessarily mean that there is a victim. Additional sensors need to be brought to bear to confirm that it is not a lamp or other inanimate heat source.

In the development of standard performance test methods, the process is more structured and rigorous. There is a requirements-development stage, in which the various performance needs are captured from subject matter experts. The SMEs may also help define (as in the case for USAR robots), how the performance is measured. Another important factor is the expected normal operating ranges for a particular measurement. The result is a set of standard test methods, i.e., a definition of activities that the robot is supposed to perform, artifacts required to support the activities (also referred to as "props" by responders), and measurement techniques. The positioning of the visual acuity charts, described in

Section 2.3, for instance, depends on how a particular robot may be deployed. Some robots will operate in confined spaces, hence their sensors are not expected to have a long range visibility. The eye charts would be positioned more closely to these robots than for robots that are to assess a building's structural stability by flying or hovering beside it and trying to estimate sizes of structural cracks.

In general, NIST's philosophy is to encourage dissemination of test arenas, artifacts, and methods as widely as possible to educate and challenge those who build robots and related technologies. Therefore, an additional consideration when designing tests is to try and make them as easy to replicate as possible. This may not always be feasible, but it is nevertheless to be considered while making design decisions. Economical and readily-available materials are strongly preferred. Figure 8 shows two examples of test artifacts that are made from commonly available materials and can be replicated easily.

The use of "virtual testbeds" is another consideration. Although simulation environments are very useful tools for performance evaluation, they are not a panacea and must be used with an understanding of their limitations. Simulations cannot completely reflect the complexity and noise found in the real world. Simulated sensor feeds, even if degraded in order to avoid being unrealistically "clean," do not represent the type of sensor input that a system will encounter from a real sensor in the real world. Similarly, communication subsystems have myriads of random degradations that cannot be accurately modeled in most simulations. Environmental models cannot capture the detail and variability present in the world. It is practically impossible to capture the level of detail of an environment (e.g., each blade of grass that a ground vehicle must go over) and the physical interactions that occur due to the modeling effort and the resulting strain on the processing power of the computer running the simulation. Similarly, the actuation of the robot cannot be completely modeled. Therefore, developers must be aware of the limits of the simulation's power in validating and verifying their system's performance: it is an important and effective first step, but if a real systems is to be built, much further testing will be needed in the real world.

Figure 8: Examples of readily-reproducible test methods. On the left is a "directed perception" test artifact that is constructed of a set of cardboard boxes taped together. Holes cut out in the boxes are positioned at different locations within the stack. Target items, such as glow sticks, are placed in some of the boxes. The task is for the robot (or operator controlling the robot) to clear the entire stack, i.e., search each box and determine whether there is an object inside or not. On the right are "random step fields," which are easily manufactured from standard lumber and can be used to provide mobility testing as well as non-planar surfaces from which to execute other tests, such as visual acuity or directed perception.

3. Ground Truth Infrastructure

The supporting infrastructure for a performance measurement program relies on instruments that have higher accuracy and resolution than the systems being tested. Effective evaluation of vehicle performance depends on ground truth about the world, the vehicle, and what the vehicle senses. NIST has developed capabilities that address this need.

In most of the performance evaluation procedures, precise measurements of vehicle position and orientation are required. For outdoor applications, GPS-based methods are used. For indoor applications, NIST can deploy a tracking system that does not rely on GPS.

3.1 Localization and Tracking of Vehicles in an Indoor Environment

In an effort to capture quantitative performance data for robots under test within the NIST Reference Test Arenas for Urban Search and Rescue Robots, a commercial Ultra Wide Band (UWB) radio frequency system was adapted to perform non-line-of-sight localization and tracking of each robot. The system used four (or more) networked receivers with antennas in known locations around the perimeter of the arenas to locate actively chirping radio frequency tags affixed to each robot for the duration of the test. The tags were golf-ball size transmitters sending small data bursts, essentially containing tag identifiers and a time-stamp, in short pulse radio frequency waveforms that allowed the set of receivers to use time difference of arrival computations to determine the location of the tag. The system, which tracked the location of multiple tags (robots) simultaneously, operated at a center frequency of 6.2 GHz with a bandwidth of approximately 1.25 GHz, and the tags transmitted at power levels of roughly 30 mW.

2D location experiments, both static and dynamic, initially assessed performance characteristics such as range and accuracy within the multi-path environments of the NIST test arenas and found static accuracies over one minute time intervals to be well within 0.20 m across arenas roughly measuring at least 20 m x 10 m. Although the instantaneous accuracies from a single chirp were often much larger than that due to occlusions in the environment, the average position over multiple chirps, five or more at roughly 1 Hz, was sufficient for the application. However, to allow tracking of moving robots and generally improve the accuracy of all reported locations, data filters helped discern indications of the robot's general motion and dwelling locations. One filter suppressed positions deemed to be noise by looking for and discarding sudden movements of more than 1 m/s, faster than the robots could reasonably traverse the complex environments, along with sudden reversals of direction from chirp to chirp, which were physically unlikely. Another filter considered the simultaneously reported location of three tags affixed to any given robot to use the average of the two closest tag locations as the reported location, discarding the third location as spurious. These filters effectively improved the overall tracking performance for robots moving at roughly 0.5 m/s, which was effective for most robots in these complex environments.

A graphical user interface was also developed to make the system easy to set up, calibrate, and use in a variety of test scenarios, including the annual USAR robot competitions held in various locations around the world. The tracking system displayed each robot's 2D position, updated roughly once per second and superimposed on an overview image of the test environment or arena (Figure 9). The display also had time indicators to synchronize with other video streams, and information regarding the robot and operator(s) involved in the test.

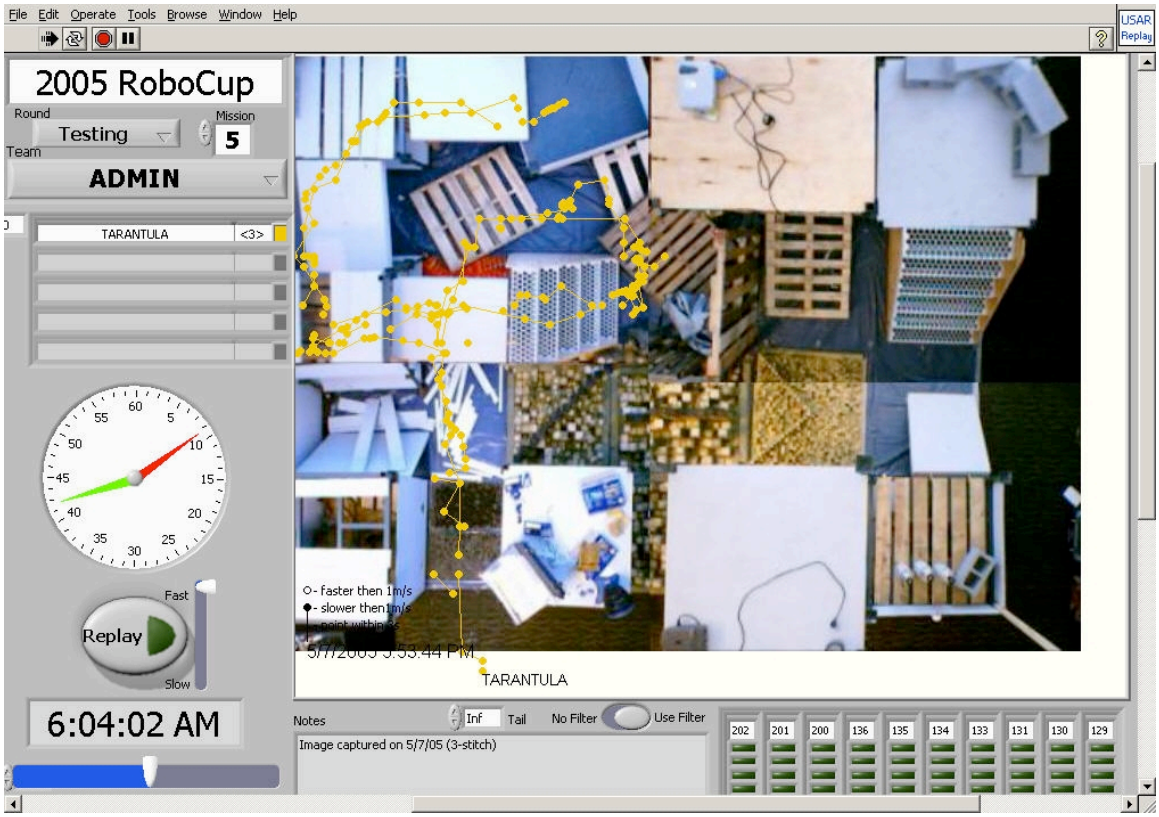


Figure 9: This is a screen capture from a tracking movie showing the path of a robot (yellow dots connected by lines when timely positions are reported) overlaid onto an overview image of one of the more complex Reference Test Arenas for Urban Search and Rescue Robots, which includes stairs, elevated floors, and pallets of random step fields. The surrounding tracking system receivers with antennas are located just out of view in the corners of the image.

In addition to tracking robots within the USAR test arenas, the tracking system was deployed to capture the performance of both robots and emergency responders in a collapsed structure training exercise hosted at NASA's Disaster Assistance and Response Team (DART) facility at Moffett Field, California in May 2004. This exercise/workshop brought together emergency responders and technology developers around situationally relevant rescue scenarios to learn about each other's needs, requirements, and capabilities. Two separate rescue scenarios were instrumented for tracking: a rubble pile search and a victim extraction from a semi-collapsed structure. The system was able to track emergency responders on top of the rubble pile, but not within the voids and tunnels underneath the rubble; the limited power of the small tags was insufficient to penetrate the reinforced concrete rubble. However, tracking data was captured for emergency responders during the victim extraction scenario within the semi-collapsed structure. Each emergency responder was outfitted with two tags, one on either side of their helmet, prior to entering the scenario. The antennae had to be judiciously configured to assure coverage within this structure. Figure 10 is a screen capture of a movie file showing the current position and previous paths of three responders overlaid onto the floorplan of the intact section of the building. Each track has a variable length tail to delineate the path traversed in a set time period.

The goal is to use such tracking movies, and associated time-stamped position data, to provide the basis for quantitative performance metrics to support development of new technologies and their inclusion into existing operational strategies for emergency responders. Objective performance data such as this can help developers understand the successes and failures of their systems, and provide eventual purchasers with clear indicators of success. Examples of performance metrics for USAR robots within well-defined arenas

and obstacles are: search rates, percentage of search area traversed, proximity to simulated victims when identified, and other quantitative measures of performance.

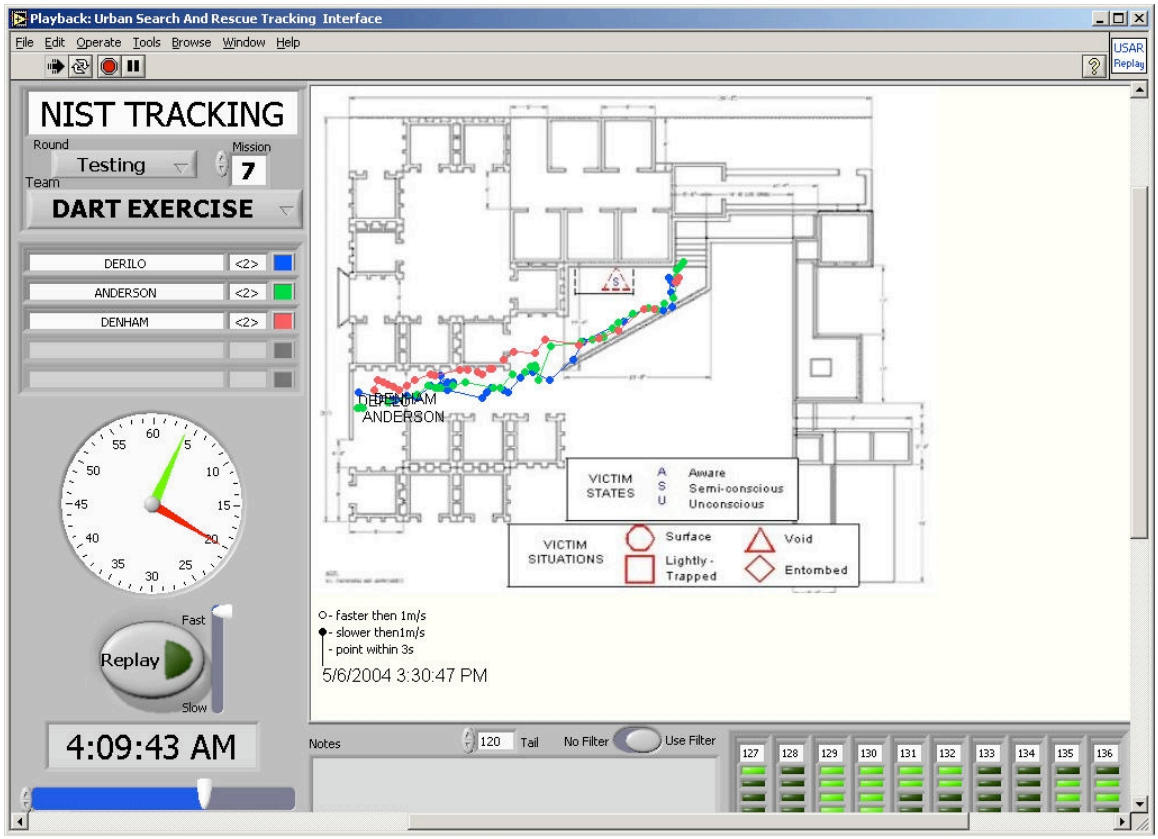


Figure 10: This is a screen capture from a tracking movie of three emergency responders during a training exercise to extract a victim from inside a simulated collapsed structure. Each emergency responder wore two active tags on their helmets while conducting the exercise. The track of their positions is shown by colored (red, green, blue) dots connected by straight line segments.

3.2 Localization and Tracking of Vehicles in Outdoor Environments

Further ground truth measurement infrastructure was developed in support of autonomous vehicle research and development, and vehicle safety system analysis for lane departure warning systems. This infrastructure was developed, in part, to support of Army Research Laboratory and Department of Transportation projects. In the outdoor realm, the GPS provided an important element of a high precision ground truth system

NIST's measurement system consisted of components on the vehicles and components off the vehicles. The off-vehicle component was a differential GPS base station installed on the tallest NIST building. The base station antenna location was surveyed to millimeter level accuracy. The GPS unit used this known location and its currently computed location to provide correction messages to the roving receivers in the area. Proper operation of the base station was confirmed by evaluating position reports of a rover GPS receiver located over National Geodetic Survey markers on the NIST campus.

The vehicle (rover) components were installed on two NIST testbed vehicles. A HMMWV and a sedan were equipped with an Applanix high precision position and orientation system. These systems provided full position and orientation solution for the vehicles at rates up to 200 Hz by integrating a dual frequency GPS, a wheel encoder, and an inertial measurement unit (IMU) that was based on gyros and

accelerometers. A second GPS was used to aid in determining heading. This system was capable of supporting reports of position, roll, pitch, and heading, as well as velocities and accelerations, all time tagged with microsecond precision. Figure 11 illustrates the NIST HMMWV high-precision position measurement system.

The absolute (world) position accuracy of the solution was limited by the quality of the GPS solution. A typical position solution uncertainty from the Applanix unit operating stand-alone was on the order of a few meters. Improvement to cm level performance was accomplished via two methods. The first method involved logging the rover GPS data and post-processing it with logs from the nearby differential GPS reference base station. Once the corrections from the base station were applied, the rover data was further post-processed with other logged data from the Applanix to produce the smoothed best estimate of trajectory data file. This produced position solutions with 2 cm level uncertainty, depending on the quality of the logged GPS data. The data quality was influenced by environmental factors such as tree cover, buildings and other obstructions to the satellite line-of-sight, as well as the current satellite constellation geometry. Advantages of this technique were that no radio link is needed during the test, and that potentially, slightly improved results over real-time approaches were possible. Disadvantages included the delay in obtaining the reference station data, and the complexity of post-processing.

The second method employed a FreeWave data radio on the rover vehicle to receive the differential corrections transmitted from the differential GPS base station described earlier. These messages were passed to the Applanix unit. This permitted the Applanix to operate in a real-time kinematic (RTK) mode, achieving cm level accuracy in real-time (no post-processing required). The Applanix unit used an algorithm called Inertial RTK (IRTK), which featured a tightly coupled integration of the GPS information. In this method, low level GPS data was used rather than full position solutions from the GPS component. IRTK performed better in challenging environments with frequent, intermittent satellite signal blockages. In the best conditions, consisting of good visibility to a sufficient number of GPS satellites from both the rover and base station, position uncertainties of about 3.5 cm were typical. This required the GPS portion of the system to be operating in its highest performance mode, Integer RTK. On the NIST grounds, this mode was commonly observed continuously in relatively open areas (fields, parking lots), and sporadically in more cluttered areas.

With slightly more challenging environmental elements, the system operated in Float RTK mode, providing uncertainties on the order of 15 cm. This was typical around buildings, trees, etc. On the NIST grounds, it was common for the system to switch quickly back and forth between Integer and Float RTK during a drive on the campus. More challenging areas, denser tree cover, canyons between buildings, etc., generally precluded RTK operation, in which case the system reverted to Code-based differential GPS mode with 1 m uncertainty, or in the most severe cases, no differential correction solution, accompanied by several meter uncertainty. And, of course, it was possible to receive no GPS signal at all in some situations (complete tree cover, indoors, etc.), in which case the inertial solution was aided only by the wheel encoder. The solution degraded with time and distance traveled until GPS reception was reacquired. The advantages of this method included immediate results without post-processing complexity and delays. Disadvantages included requirement for real-time data link from the base station, and small potential decrease in solution quality.

Finally, a second GPS receiver was acquired that could be configured to operate as a rover in RTK mode. This was useful for logging high precision position data of a rover vehicle where the tests could be conducted in a favorable environment (open field, etc.) and where the full capability (eg., roll, pitch, etc) of an Applanix-type unit was not required. A complete, portable position logging system consisting of the GPS receiver and antenna, a data radio receiver and antenna, and battery power supply was configured in a small, portable enclosure for this purpose. Again, centimeter level performance was possible in appropriate conditions. This GPS unit could also be configured to serve as a base station for use at other locations, or used as a field unit for surveying specific locations.

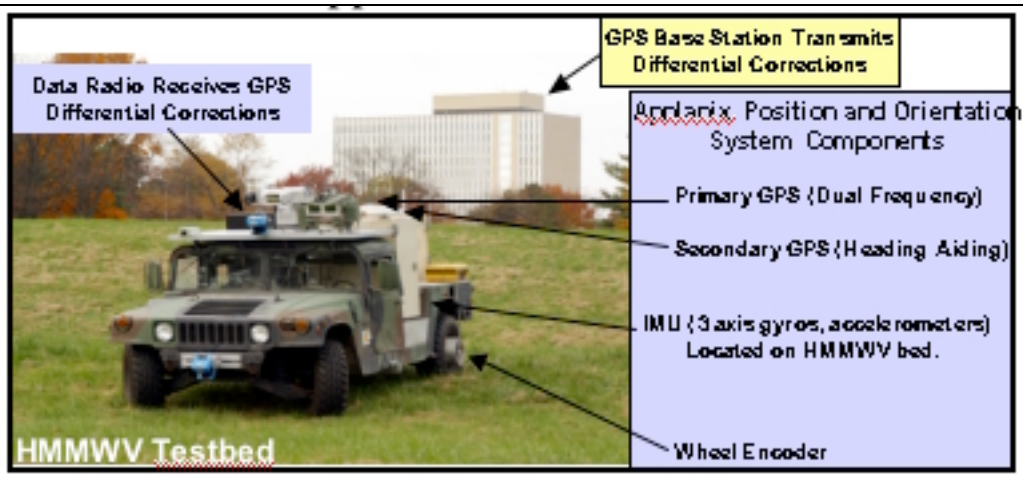


Figure 11: NIST HMMWV Testbed shown with High Precision Vehicle Location Measurement System.

3.3 Ground Truth Through Digital Terrain and Feature Maps

Another essential tool in the ground truth arsenal is high precision site data. This enables comparison not only of the vehicle's position with actual locations on the campus map but also of what the vehicle senses in its surroundings (and its interpretation of the objects' locations and characteristics) with what the object is in reality and where it is placed in the world. NIST had its entire campus and some surrounding areas surveyed to produce a very detailed terrain model. This terrain model consisted of a bare earth elevation array with post spacing of 45 cm (1.5 feet) and root mean square error (RMSE) of 15 cm (6 inches), color orthophotography with pixel resolution of 7.5 cm (0.25 feet), and comprehensive vector data that captured features in the environment. The vector data included features such as all road edges, parking lots, parking lot stripes, buildings, sidewalks, lamp posts, and signs. Tools were developed that can track a vehicle's position in real time (capturing the results of the high-precision position measurement system described above) and overlay its trace onto the NIST map. Results of onboard sensory processing could be visualized and evaluated. For a vehicle performing road-following behavior, its sensed road edges were overlaid on the actual ones to determine how accurately the on-board processing is positioning them in its maps. Figure 12 shows the road edges that were computed by a NIST vehicle using sensors overlaid over the relevant section of the high-resolution NIST campus map.

In Section 4.2, another approach to generating ground truth is discussed. Selective capture of detailed data of areas of interest can be performed to clarify certain vehicle performance issues. Terrain data can be captured and characterization of such parameters such as roughness, slope, or even vegetation density can be computed. This knowledge can shed light on what types of terrains or environments a vehicle can negotiate or not.

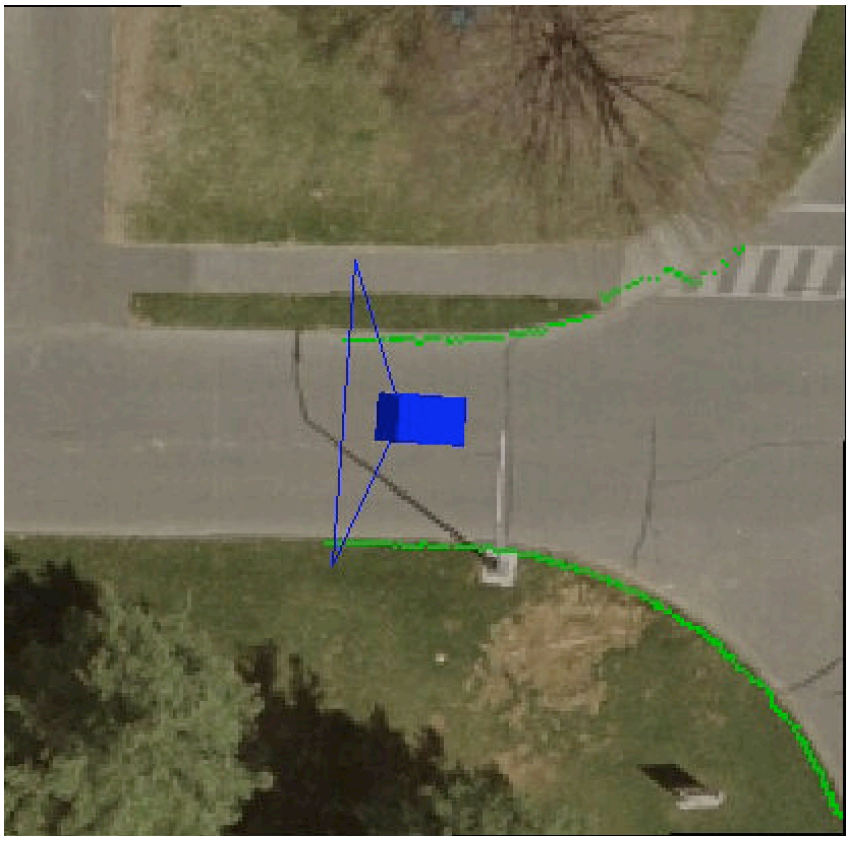


Figure 12: Overhead view of high precision ground survey data for NIST campus showing tracking of vehicle position and detected road edges (shown in green). The sensor field of view is the blue triangle in front of the blue box representing the vehicle. The extracted road edges can be compared with the ground truth using this data.

3.4 Discussion of Ground Truth Infrastructure

Simply put, one cannot measure the performance of an unmanned vehicle unless there is ground truth. Without knowing where the vehicle is, how fast it is moving, what features exist in the environment, and what the other relevant external conditions are, it is futile to try to measure how well the vehicle did and it is extremely difficult to diagnose problems.

One of the key pieces of knowledge necessary is where the vehicle went and when it went there. This section has discussed technologies for tracking vehicles in indoor and outdoor environments, including areas which have limited GPS. Unless the environment provides clean lines of sight to the satellites (for GPS) or antennae (for UWB), several provisions have to be put in place to still have acceptable localization results. Examples of some of the approaches to mitigating the localization errors were described.

The location on the world or within an indoor environment where the vehicle went can be enriched by knowing also what the surrounding conditions were at each location. In the test arenas primarily aimed at USAR robots, ground truth is established because these are manufactured environments and hence known to the evaluation team. Overhead photo imagery and two or three-dimensional models may be available of the arenas. For outdoor locations, it is more difficult to gain ground truth knowledge of the environment. Two approaches can produce high resolution terrain characterization. One is an overhead capture of high-resolution digital terrain elevation and exhaustive features of an entire site via aerial flyovers. Another is

judicious, targeted capture of high resolution data in areas of interest, typically from a ground-based perspective. This is discussed in more detail in Section 4.2. A high-resolution digital aerial survey of a site is very expensive and time-consuming and becomes obsolete after the site changes. Portions of the NIST campus survey became obsolete shortly after capture due to new construction. Nevertheless, the richness of features, such as curb heights and parking lot paint stripes, and size of area captured, make this a very useful tool in the ground truth and performance evaluation arsenal. Of course, this site is used frequently for evaluation of unmanned ground vehicles. Being able to overlay the vehicle's position and displaying portions of its world model over the campus features has been extremely valuable for testing new algorithms and sensors. For sites that are used only for a limited time period, it is more efficient to capture selected ground truth, using imaging, positioning, and other sensors. The data captured may have to be post-processed to generate usable ground truth, for example slopes and roughness. To gain understanding of a vehicle's performance under different situations, this is a worthwhile effort, as is seen in the next section.

4. Case Study: Technology Readiness Evaluation of Autonomous Mobility

The above performance evaluation methodologies and some of the tools were used in the evaluation of the Army's Demo III [19] XUV's Autonomous Navigation System (ANS). This evaluation was one of the most rigorous and comprehensive ever conducted on intelligent autonomous ground vehicles. In 2002 and 2003, the Army Research Laboratory conducted a study to determine if the autonomous navigation subsystem developed under Demo III had reached Level 6 in the TRL scale developed by NASA [8]. Autonomous navigation controls the movement of the vehicle from waypoint to waypoint using onboard sensors to detect and avoid hazards and obstacles. The TRL scale ranges from 1 to 9, with 9 signifying a system that has been proven in actual missions. TRL 6 is defined as "component and/or breadboard validation in a relevant environment." The Future Combat Systems program, discussed in more detail in Chapter 10, expected to rely on vehicles capable of traversing environments including Urban, Open Rolling Arid, and Mixed Open Rolling Vegetated, but does not specify exactly how these terrains are defined. The selection of a particular terrain that is considered "relevant" on which to test the vehicles strongly affects the outcome of the tests. Unless there is some objective measure of terrain difficulty, the tests have limited value. NIST developed terrain characterization methods to provide quantitative evaluation of the paths traversed by vehicles in carrying out their missions. The terrain characterization provided ground truth. This allowed for comparisons between autonomous vehicles and human drivers and between different types of vehicles. To determine if autonomous mobility has reached Technology Readiness Level 6, some measure was needed of vehicle performance over terrain of known difficulty.

This section begins with a description of the TRL 6 experiments. This is followed by a discussion of the terrain characterization work that accompanied the TRL 6 data collection. Both of these are interrelated elements within the holistic performance evaluation view presented in this chapter. The TRL experiment definition was driven by the application-specific needs – both the mission types and the environments. This led to the selection of the courses over which the robotic vehicles were required to run. Extensive data capture of the vehicles' performance provided meaningful information to the developers and program managers of the strengths and weaknesses of the various components. The TRL experiments necessitated an extensive ground truth infrastructure, which was further augmented by the terrain characterization effort.

For the TRL 6 exercises, NIST administered similar field experiments in three different terrain types: rolling/arid, rolling/vegetated, and urban. Two courses were set up at each terrain site to provide different levels of difficulty. The test vehicles were provided with GPS waypoints for over 650 missions, ranging from 500 m to 2 km in length, which covered over 500 km of autonomous driving through the three terrain types. For each mission, several types of performance data were captured, including: vehicle log files, operator control unit log files, handwritten observer log sheets, and several streams of video to capture vehicle performance, operator workload, and operator interface screens. After completion of vehicle missions at each site, NIST attempted to characterize the terrain difficulty of the courses using an instrumented HMMWV with differential GPS, inertial sensors, cameras, and laser range scanners to capture detailed terrain features in an effort to quantify the terrain difficulty. The following is an introduction to the approach used to administer these field experiments, collect performance data, and

characterize the terrain. A more comprehensive discussion is available in the "Autonomous Mobility Technology Assessment Final Report," which includes vehicle performance results [9].

4.1 Technology Readiness Level Assessment Experimental Framework

The purpose of these field experiments was to evaluate the functionality of a test vehicle's autonomous navigation system to conduct tactically relevant missions in a variety of repeatable terrains with an emphasis on collection of statistically significant performance data. As noted above, TRL 6 is defined as validation performed in a relevant environment. The definition of what constitutes a "relevant environment" is somewhat vague, so NIST relied on subject matter experts, military officers or noncommissioned officers, to select the sites used to conduct these experiments, which were representative of rolling/arid, rolling/vegetated, and urban environments. The emphasis on statistically significant data collection required development of a formal experimental design to isolate key variables, introduce randomization, and adhere to accepted statistical practice.

A secondary focus for these experiments considered the need for occasional intervention by a human operator under certain conditions when the vehicle recognized it could not cope with an obstacle or terrain feature either because of limitations of the autonomous navigation system or the vehicle mobility itself. Since a soldier conducting other mission duties is expected to perform such interventions in eventual operations, the operator workload involved in remotely teleoperating the robot in difficult terrain is particularly important.

Overall responsibility for the conduct of the field experiments resided with the Test Director and Test Leaders, who ensured adherence to the experimental design, maintained a safe operating environment, and administered day-to-day operations for the roughly two weeks at each site necessary to execute the experimental plan, collect data, and characterize the terrain. The Test Leaders led each vehicle column in the field and were directly responsible for collection and archiving of all experimental data. NIST engineers and scientists also performed terrain characterization of the test courses after completion of missions at each site.

4.1.1 Field Experiment Sites and Course Development

The purpose of these field experiments was to assess the functionality of the test vehicle's autonomous navigation system to conduct tactically relevant and repeatable missions in a variety of terrains. To establish tactical relevance SMEs, military officers or noncommissioned officers from the Weapons and Materials Research Directorate of ARL selected the sites used to conduct the experiments and to define the specific courses at each site. The sites chosen are illustrated in Figure 13.

Figure 13: The field experiments were conducted in three different terrains: rolling/arid, rolling/vegetated, and urban.

To simulate likely tactical conditions as closely as possible, courses were defined by SMEs from aerial photographs. During this process, the SMEs were instructed to utilize paths that would likely be used during military operations and to make maximal use of the diversity of terrain available at each site. Two courses were established at each site, with each course characterized by a different level of terrain difficulty. The course traversing less difficult terrain was designated as the "Gold" course and the course traversing significant terrain challenges was designated as the "Black" course. To minimize the time required to reset the vehicles between trials on the large rolling/arid and rolling/vegetated sites, closed loop courses were used. Once each course was outlined onto the photographs, a safety officer drove the course to ensure that there were no safety hazards. Then a series of waypoints, approximately 100 m apart, were designated to ensure that the vehicles would stay close to the intended terrain features. GPS coordinates for each of the waypoints were identified and used to form individual mission paths with randomized start points, direction, and lengths of 500 m, 1000 m, or 2000 m for the principal experimental design. Longer missions were performed as excursion missions for the experimental design, along with several other interesting mission variations discussed in Section 4.1.3. Based on the terrain features available at each site, the SMEs defined the Rolling/Arid Gold Course as a 6.8 km loop, and the Rolling/Arid Black Course as a 4.4 km loop. The Rolling/Vegetated Gold Course was defined as a 5.0 km loop, and the Rolling/Vegetated Black Course was a 7.2 km loop (see Figure 14). In addition to the test courses, separate "practice courses" were designated to allow vehicle testing and operator training for the soldiers.

The urban courses had designated waypoints at every intersection of a 500 m x 200 m barracks complex, with five east-west avenues and eleven north-south roads or alleys – 55 waypoints in all. The limited size of the urban area required that both the Black and Gold courses share the same physical area and features. The unaltered urban area served as the Urban Gold Course. Additional obstacles such as junk cars, trash dumpsters, gravel, and scrap woodpiles placed among the existing urban features served as the Urban Black Course. The missions defined in the experimental plan were conducted on the Gold Course first. After the additional rubble was placed, the Black Course missions were conducted.

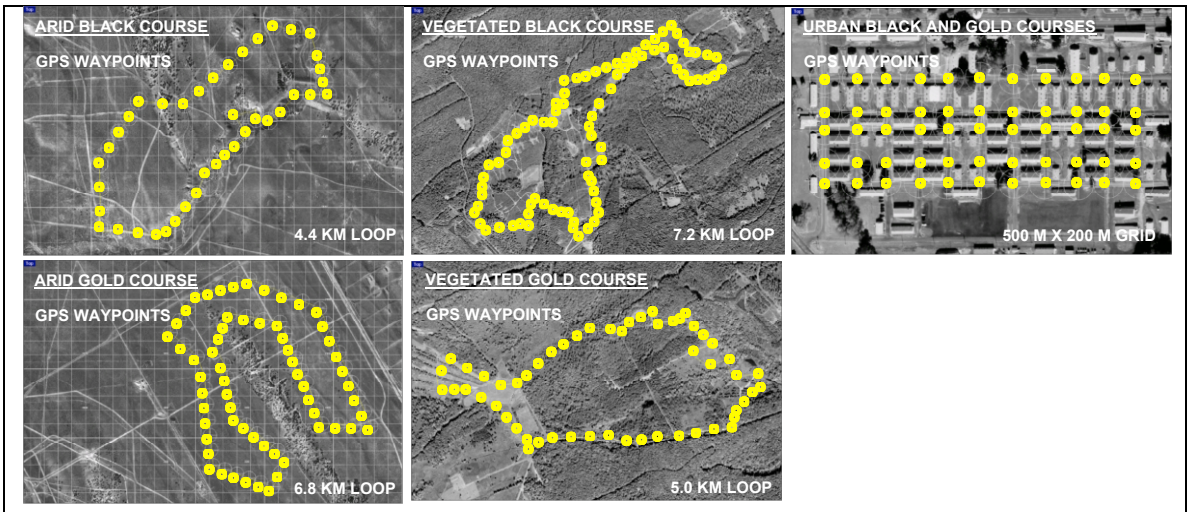


Figure 14: Test courses defined by sets of GPS waypoints placed to ensure that the vehicles stay close to tactically relevant terrain features. Both test courses from each of the rolling terrain sites are shown: the Rolling/Arid Black Course (4.4 km loop) and Rolling/Arid Gold Course (6.8 km loop), the Rolling/Vegetated Black Course (7.2 km loop) and Rolling/Vegetated Gold Course (5.0 km loop). The Urban Black and Gold Courses used the same waypoints (every intersection in a 500m x 200m area) and were differentiated by additional obstacles for the Black Course.

4.1.2 Test Vehicles

The prototype autonomous vehicle used to conduct the field experiments was the XUV developed for the ARL Demo III program by General Dynamics Robotic Systems (GDRS) and others (see Figure 15). The XUVs were four-wheel drive, four-wheel steer, vehicles that used actively nodding (front) and stationary (rear) LADARs along with GPS and inertial sensors to navigate. Several cameras were also mounted around the vehicle to enable teleoperative interventions.

Each mission was conducted with a column of vehicles including the XUV, a Safety HMMWV, and an Operator Control Unit (OCU) HMMWV. The Safety HMMWV, which always stayed in close proximity to the XUV, contained a Safety Operator who could remotely stop the XUV if considered unsafe in any way and the Test Leader, who was responsible for all operations in the field. The OCU HMMWV was always located some distance away from the XUV but within radio range to allow teleoperative interventions. The operator within the OCU sat behind a "blind" to limit direct sight of the XUV and the surrounding terrain. However, the driver of the OCU HMMWV could verbally assist the operator with direct surveillance during designated Line of Sight (LOS) missions. During Non-Line-Of-Sight missions, the operator worked only from the information, images, and video communicated from the XUV. A human factors observer also rode in the OCU HMMWV to capture operator workload assessments during each intervention when they occurred. Two concurrent vehicle columns operated continuously during the field experiments, one on each course, to enable efficient capture of vehicle performance data at each site.

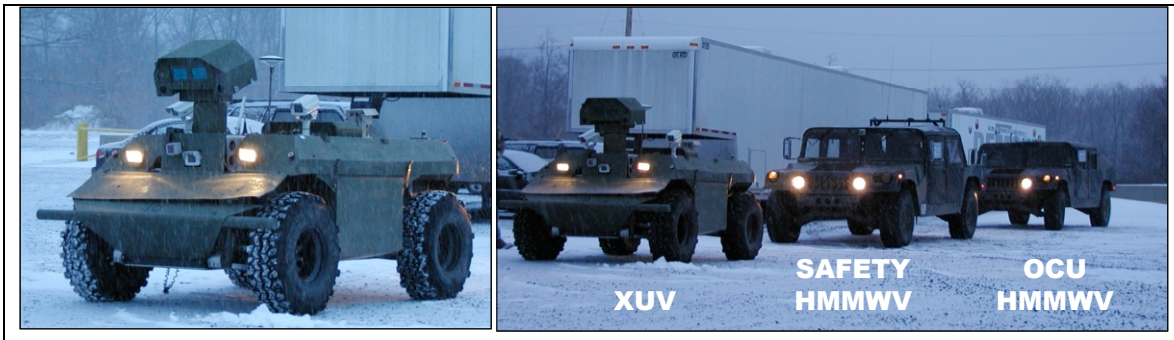


Figure 15: The vehicle columns used to conduct each mission included an XUV, a Safety HMMWV that could remotely stop the XUV if considered unsafe in any way, and an Operator Control Unit (OCU) HMMWV located within radio range for teleoperative interventions.

4.1.3 Experimental Design

Statisticians from the Computational and Information Sciences Directorate of ARL developed a formal experimental design to isolate key mission variables, introduce randomization, and adhere to accepted statistical practice. The "Code of Best Practice Experimentation," a publication of the Command and Control Research Program (CCRP) [1], also guided many aspects of the approach to formalizing the field experiments. The SMEs, statisticians from NIST, and a "Red Team" of distinguished experts knowledgeable in experimental design, conduct, and analysis also provided guidance. This resulted in a statistically significant set of missions to assess the vehicle's autonomous mobility while capturing comprehensive performance data regarding key mission variables.

The three-site experimental design provided a variety of relevant operational terrains: rolling/arid, rolling/vegetated, and urban. Within each site a principal experimental design was established to study five factors: terrain difficulty, mission distance (or type of mission at the urban site), maximum vehicle speed, line-of-sight operation for teleoperative interventions, and XUV/Team effects since multiple experimental vehicles and associated operators were used to gather data. After completion of the principal experimental design, excursion missions were conducted to capture less statistically rigorous data regarding alternative mission profiles and other variables of interest.

The overall experimental design called for 656 missions over the three terrain types. At each of the two rolling terrain sites, 182 missions were planned totaling over 200 km of autonomous driving (400 km over two sites). The principal design for the rolling sites allocated 144 missions, using 500 m, 1000 m, or 2000 m mission lengths to address the questions of primary importance, with an additional 38 excursion missions. The approach adopted was to randomize the presentation of course features and expect varying XUV behaviors to provide different course looks. Twenty-four missions were performed each day by two concurrent XUV teams. Each team conducted randomized sets of six missions in the morning and in the afternoon, with randomly selected variables such as a starting waypoint for the set, direction around the course loop (clockwise or counter-clockwise), and lateral displacement from the first mission waypoint (5 m, 10 m, or 15 m) to vary the actual start point. The final waypoint of each mission in the set was the starting waypoint for the following mission, along with a random direction and lateral displacement.

For the urban site, due to its limited size, the experimental design called for 192 missions in the principal design and 108 excursion missions, totaling roughly 150 km. Thirty-two missions were performed each day as two concurrent XUV teams conducted randomized sets of eight missions in the morning and in the afternoon. The factors of the principal design aligned as closely as possible with the rolling terrain experiments, including terrain difficulty, maximum vehicle speed, line-of-site teleoperative interventions, and XUV/team. However, mission distance was replaced with mission complexity, using "assault" or "patrol" type missions from one end of the urban area to the other. An additional factor of five specific

routes representing assault or patrol type mission complexity levels was necessary to appropriately reflect the statistical design structure.

The principal experimental design for all three terrain types provided an emphasis on collection of statistically significant performance data. The excursion missions investigated alternative mission profiles and provided initial feedback on other tactical variables of interest. Some examples of excursion mission sets included: endurance missions (7 km), 100% teleoperative missions, day vs. night missions, soldier XUV operators, no intermediate waypoints, no GPS coverage, fully blockaded urban streets, and others. A more comprehensive discussion is available in the formal report, which includes vehicle performance results and analysis of outcomes [9].

4.1.4 Operator Interventions

Although the primary goal of the field experiment was to evaluate autonomous mobility, XUV operator interventions were allowed but strictly controlled. The XUV operators, located in the OCU HMMWV within radio range of the XUV, were provided guidance that missions were to run autonomously unless a qualified reason for operator intervention was noted on the OCU user interface, and that teleoperation of the vehicle was to be minimal. The XUV operator was allowed to monitor status messages transmitted from the XUV while it was maneuvering autonomously, along with a simple display that showed the position of the XUV with respect to upcoming waypoints. Once a qualified status message or call for help was initiated by the XUV, the XUV operator could take over teleoperation of the vehicle until the XUV was safely away from the present obstacle and not likely to back up into the same obstacle. Prior to each intervention, the XUV operator had no access to video from the XUV cameras, and never had line-of-sight to the XUV or surrounding terrain (a blind was set up in the OCU HMMWV to limit visibility).

An example of a typical intervention was when the XUV backed up three times, as it was programmed to do, to reevaluate the terrain ahead when confronted with imposing or confusing obstacles. If, after three backups at the same location, the XUV ANS was unable to circumvent the obstacles, the XUV sent a message to the OCU requesting operator assistance. In total, the XUV could report 14 such qualified intervention conditions (e.g. loss of traction, loss of GPS fix, pitch or roll exceeds 20°, etc.) for which operator intervention was allowed. Human factors experts from the Human Research and Engineering Directorate of ARL, riding along in the OCU HMMWV, conducted operator workload assessments while NIST researchers captured video of all operator actions for every intervention.

4.1.5 Measures of Performance

Several forms of performance data were captured after each mission including: XUV log files, Operator Control Unit log files, handwritten Test Leader log sheets, and several streams of video to capture vehicle performance, operator workload, and operator interface screens. The computer-readable log files were parsed to determine the vehicle's state at locations along the mission path. The after-action briefing graphics showed the mission start point, the path, and state of the XUV for each mission; yellow paths for autonomous operations, blue paths for teleoperative interventions, and green paths for autonomous backups by the XUV (see Figure 16). The mission classification is shown at the mission end point. Additional images and interesting details of the path, such as multiple autonomous backups or teleoperative interventions, which were particularly helpful to the engineers involved in developing the vehicle behaviors, were also included in the graphics.

There were four end-of-mission classifications that divided into successful, or "complete," missions and incomplete missions. Missions were classified as incomplete if one of three end-of-mission conditions occurred: Stop, E-Stop, or Fault. These classifications are described below.

Mission Complete: A mission was classified as Complete if the XUV autonomously navigated to with 20 m of the final designated waypoint (GPS coordinates) in an ordered list of mission waypoints provided prior to the mission. The XUV was expected to pass within 20 m of each waypoint provided, but was allowed to autonomously cut corners as long as no hazardous conditions or property boundaries were encountered. Complete runs could include administrative stops (A-stops) as well as intermittent

teleoperative interventions. Fig. 16a shows an example of the path traced out by the vehicle during a mission that ran to completion.

Mission Stop: When the XUV became stuck or disabled during the normal course of operations, not involving any safety concerns, it would call for an intervention by the OCU Operator. If the OCU operator was unable to teleoperatively extricate the XUV from a situation that caused the intervention, the mission was considered a "Stop." The OCU Operator essentially conceded that the mission could not be completed, and the vehicle was moved to the next mission start point. An example of a Mission Stop, along with the location at which the vehicle was stuck and close up pictures of the XUV and surroundings, is shown in Fig. 16b.

Mission E-Stop: The Safety Operator or Test Leader located in the trailing Safety HMMWV could halt operation of the XUV to prevent risk to personnel or damage to the XUV or other property. At times, upon closer examination of the surrounding conditions, the Safety Officer realized that an unsafe condition did not actually exist, or that the perceived hazard was temporary. An example would be when the XUV was cresting a precipitous hill and the Safety Operator was not in a position to observe the potential path. If an unsafe condition did not actually exist, then the stop was declared an Administrative Stop (A-Stop) and autonomous operation would continue without penalty. If the Safety Operator determined that the XUV could not continue safely, the end-of-mission classification was an "Emergency Stop" (E-Stop) and the vehicle was moved to the next mission start point. Video and still pictures of the conditions surrounding each E-Stop were captured to document the cause, as shown in Fig. 16c.

Mission Fault: Missions that were initiated with a clearly undiagnosed mechanical failure from the previous mission (immediately evident after mission start), or suffered from data collection issues or other administrative errors not associated with the vehicles autonomous mobility, were repeated after addressing the cause of the problem. An example is shown in Fig 16d, where an incorrect waypoint was entered in the vehicle mission definition. Damage incurred by the XUV during a mission did not qualify as Fault criteria. Lessons learned from Fault missions provided ongoing refinement to the standard operating procedures so as not to replicate preventable errors. Data captured from the Fault missions did not contribute to the overall vehicle performance results but were included in an Appendix of the final report.

Graphical representations of individual mission paths on each course provided significant insight into the vehicle's behaviors and ability to effectively, and repeatably, negotiate certain terrain features. Since the field exercises included so many missions, with start points randomly distributed around each course, the vehicle covered most sections of the courses autonomously in one mission or another (see Figure 16). So the autonomous path summaries shown in yellow generally trace the entire course. All information surrounding teleoperative interventions and all autonomous backing maneuvers were tracked and similar graphical summaries were generated for the final report (not shown here). Such graphics readily showed areas where the vehicle was proficient at negotiating the terrain, while clearly indicating the location of problematic for the vehicle's sensors or mobility. The most severe challenges to the vehicle resulted in E-Stops, which were often specific obstacles or scenarios that the vehicle either could not negotiate of could not avoid. They are location and the number of E-Stop instances at each location were also shown on the summary graphics. For example, positive obstacles such as fallen trees directly along the prescribed path would high-center the vehicle and negative obstacles such as culverts near certain intersections in the Urban course went undetected and became hazards for the XUV.

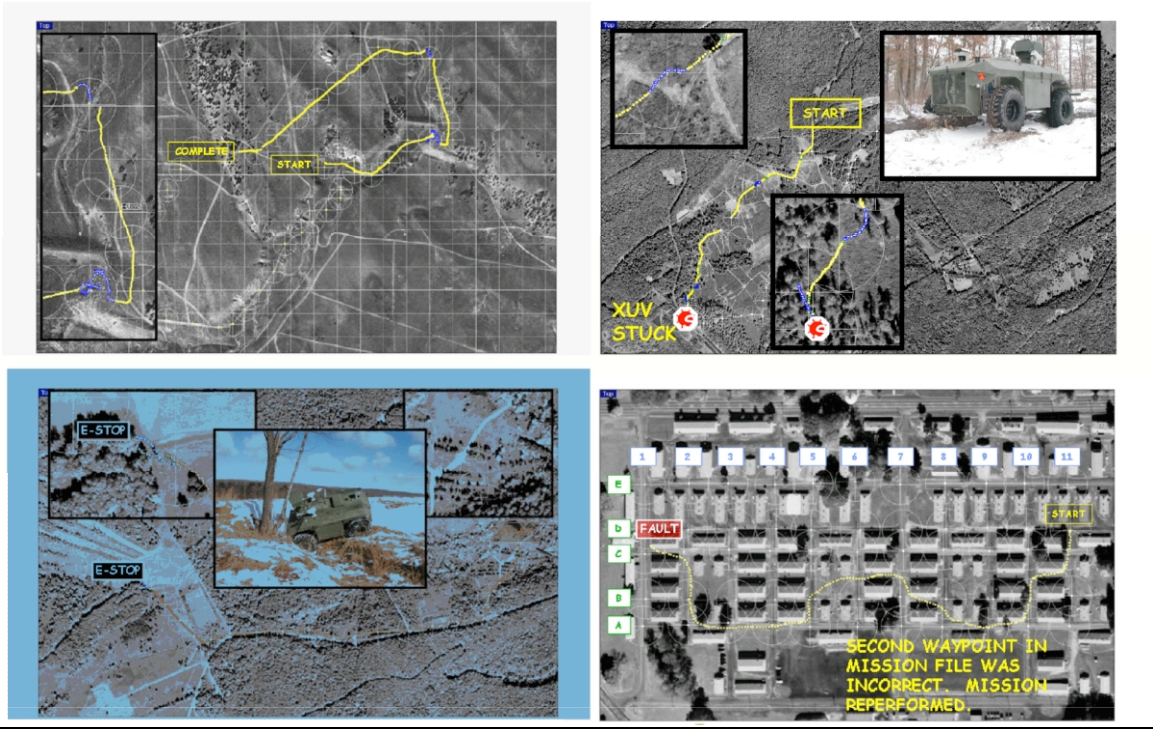


Figure 16: The four end-of-mission classifications; a) Mission Complete – the vehicle achieved the goal point either completely autonomously or with periods of teleoperative interventions, b) Mission Stop – the Operator conceded the vehicle was unrecoverably stuck, c) Mission E-Stop – the Safety Operator stopped the vehicle, either to protect the vehicle or other property, and would not let the mission resume for safety reasons, d) Mission Fault – the mission was erroneously set up to run, either with an undiagnosed mechanical failure from the previous mission or administrative errors in data collection (faulted missions were re-run).

Table 2 is a summary of the three field experiments by site and shows an ambitious schedule that produced a comprehensive archive of performance data to assess the state of technology for autonomous mobility. For each of the terrain types in which the XUV autonomous navigation was tested, the total number of missions, as well as the distance traveled by the XUV and time required for accomplishing the missions is listed in Table 2. Further detail is also provided in terms amount of autonomous navigation performed by the XUV: number of kilometers that were driven autonomously (along with percentage of overall travel this represented) and number of mission hours that were conducted autonomously (and corresponding percentage of overall mission time). This comprehensive process clearly identified strengths and weaknesses in vehicle capabilities. The approach taken, based on mission-relevant environment selection, using statistically significant numbers of experiments, and bolstered by qualitative and quantitative data that characterize the successful and unsuccessful trials, produced a body of data that could be analyzed to understand the autonomous navigation subsystem's capabilities under certain conditions. The subset of capabilities that were identified as needing improvement can be re-evaluated on the same courses, at a much-reduced overall cost, to quantify improvements as they occur. Such rigorous methods of evaluation are necessary to objectively assess and improve upon the state of capabilities for autonomous ground vehicles.

RESULTS OF FIELD EXPERIMENT BY SITE							
	Missions	Distance (km)			Time (hr)		
Terrain Type		*Total*	*Autonomous*	*% Autonomous*	*Total*	*Autonomous*	*% Autonomous*
Rolling/Arid	**177**	203.8	199.3	97.8%	34.3	31.9	92.9%
Rolling/Vegetated	**181**	203.4	188.1	92.5%	39.6	33.1	83.5%
Urban	**288**	152.5	148.9	97.6%	25.5	22.7	89.0%
Total	**646**	**559.7**	**536.3**	**95.8%**	**99.4**	**87.7**	**88.20%**

Table 2: Results of Field Experiment by Sites.

4.2 Terrain Characterization

The goal of terrain characterization is to provide quantitative evaluation of the paths traversed by vehicles in carrying out their missions. This allows comparisons between autonomous vehicles and human drivers and between different types of vehicles. To determine if autonomous mobility had reached TRL 6, some measure was needed of vehicle performance over terrain of known difficulty. The selection of a particular terrain on which to test the vehicles strongly affected the outcome of the tests. Characterizing the terrain was also necessary in order to assist in the understanding of the types of environments that the vehicles are able to deal with. The techniques that NIST has developed and applied to measure terrain are described below.

Terrain was characterized using sensors on the NIST HMMWV and on a trailer that measures soil mechanics. The trailer was developed by the Army Tank-automotive and Armaments Command Research and Development Engineering Center Robotics Mobility Laboratory. For the TRL-6 terrain characterization, the HMMWV was driven manually. Besides the high-precision position measurement system described in Section 3, the HMMWV had several other sensors mounted onboard. Sensors included a GDRS imaging LADAR mounted on a tilt platform, a color camera mounted on top of the LADAR on the tilt platform, and a Riegl high-resolution scanning LADAR that provided the prime source of range data for terrain characterization. In the initial phases of the program, a ring of cameras was mounted concentric with the Riegl LADAR to provide color data, and in later phases, a digital camera attached to a pan-tilt unit mounted above the Riegl LADAR provided panoramic images of the terrain. In some of the tests, a SICK line-scanning LADAR was mounted on the back of the HMMWV pointing straight down at the ground to provide a measure of the topography with precision sufficient to measure the depths of ruts in the ground left by the HMMWV wheels.

In order to combine data from multiple sensors, it is first necessary to calibrate the sensors and register them to a common coordinate system. Registration also enables the spatial relationships between successive samples to be computed. Chapter 6 describes the registration process in more depth. Preparing for data collection includes calibrating the sensors and accurately measuring their positions and orientations on the vehicle.

Data were collected in two primary modes. One was while the vehicle was driving normally, and the other was with the vehicle stationary. Some of the sensors did not run in real time, so could only be used when the vehicle was not moving. The trade-off between the two modes was that while data acquired in real-time approximated more closely the actual driving conditions, they were less accurate and usually of lower resolution than data from the slower sensors when the vehicle was stopped.

A critical part of data collection for mobile vehicle applications is to record the vehicle's position and orientation (pose) and the time at which each sample is acquired. This enables data collected from multiple sensors to be registered, and also allows the data for a complete mission to be compiled into a reconstruction of all the terrain that was traversed (Figure 17). The Applanix system discussed previously was used in this data collection.

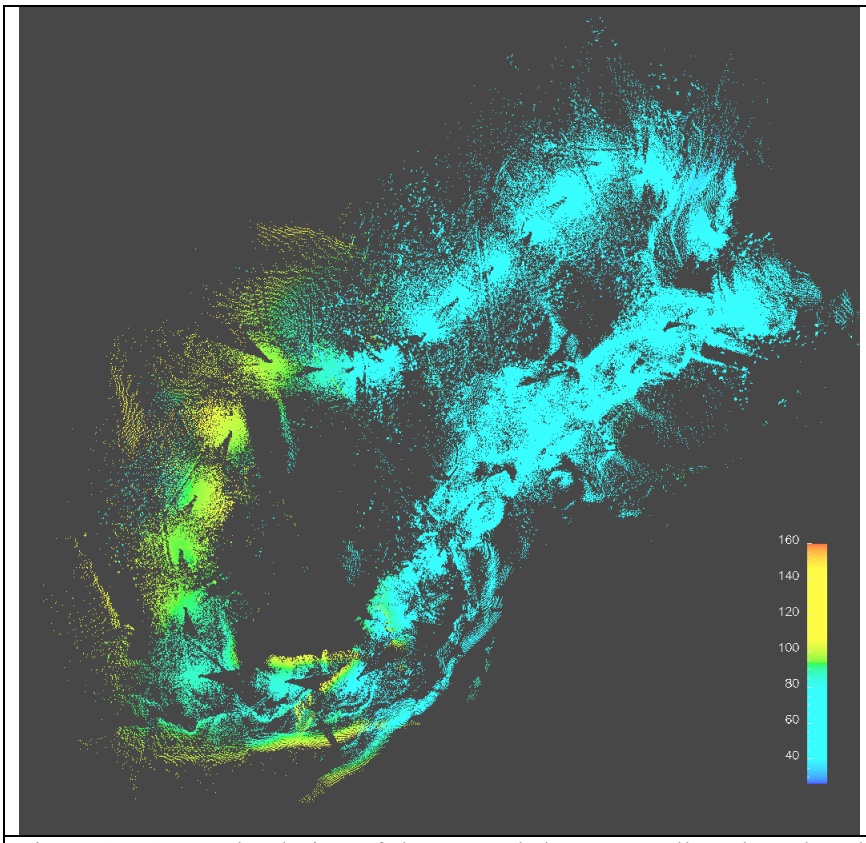


Figure 17: An overhead view of the coarse ladar scans collected on the Black course at Tooele Army Depot. The colors encode the height of the terrain.

Three sets of data were collected for each course. First, the vehicle was driven over the course collecting data with the real-time sensors (real-time LADAR, color video camera, and, in some of the runs, SICK LADAR data). Next, the vehicle moved to the first waypoint on the course. Starting from this point, and moving 10 m to 150 m between samples, scans were taken of the terrain using the Riegl LADAR, GDRS LADAR, panoramic digital camera and a set of six cameras arranged in a ring around the Riegl. The scans were not taken at the highest resolution the Riegl LADAR can measure, but still provided much more accurate information than the real-time sensors. The navigation data were also stored to provide the position and attitude (roll, pitch, heading) of the data collection vehicle at the time the sample was collected. The entire course was sampled in this way. Finally, a set of high-resolution scans was taken of the difficult or interesting locations on the course.

To collect real-time data, the NIST HMMWV was driven over the course being characterized. A human driver navigated by following a visual display of the course which shows the waypoints laid out for the TRL experiment and the HMMWV's trajectory. The sensors were started at the same instant in time and navigation and timing data were collected with the sensory data. The sensors used for real-time data collection were a GDRS LADAR (the primary sensor used by the XUV), a color video camera mounted on top of the GDRS LADAR, and the INS/GPS position data from the Applanix unit augmented by a differential GPS base station. In some cases, SICK LADAR data were also collected.

Non-real time data were collected for the entire course at a relatively coarse resolution. Higher resolution data are gathered at locations that required an emergency stop for the XUV or where the vehicle displayed "interesting" behavior, such as backing up, suddenly changing direction, or performing an unanticipated intelligent maneuver.

Having captured copious amounts of data in order to characterize the terrain and understand the vehicle behavior, a great deal of *post facto* analysis occurred in a qualitative manner through use of visualization tools. The navigation data were post-processed to provide the highest accuracy in the position and orientation of the HMMWV when each data sample was acquired. The real-time data were processed and inserted into a world model that represented the terrain in a grid, with elevation values in each grid cell. This enabled the data to be visualized in a form close to that available to the XUV. A movie showing the terrain changing as the vehicle moved provided valuable insight into what the vehicle knew about its surroundings at each point on the course. It must be noted, however, that the sensor data captured on the HMMWV was only an approximation of that used by the XUV during its runs.

Additional analyses performed on the data included computation of vegetation density, ground surface slope, and ground surface roughness for regions of interest. These analyses were based on data captured from the onboard range and pitch (orientation) sensors as well as from the measurements taken from the soil mechanics trailer. Figures 18 and 19 illustrate analyses that were performed of an area where the vehicle had difficulty navigating autonomously. In the site where data were collected for arid rolling terrain, there was a particularly difficult region at a dry dam. Figure 18 shows the dam region with the locations where high-resolution LADAR scans were taken. Two scans were combined to give a high-resolution topographic map of the region and a set of measurements was made to analyze the region. Figure 19 shows a combined scan, annotated as follows. The dotted line shows two traversals by the XUV. One ended in success and the other in failure. The failed path indicates that the XUV was misled by its inability to see far enough ahead to choose a good path. The slope of the ground along the path it chose is not as steep as the successful path near the top of the dam, but becomes steeper near the tree. When the XUV approaches the steeper section, it is going sideways along the slope, and an emergency stop was initiated to prevent it from tipping over.

The successful trajectory required the vehicle to take a path that initially appeared steeper than the alternatives, but the vehicle could traverse it sine it did not have to travel sideways on the steeper slope. The angles shown in the figure for the slopes on either side of the dam wall were computed by fitting surfaces to the wall and measuring their slopes. The measurements match the pitch recorded by the navigation system very well.

This example of the analyses performed in conjunction with the overall TRL experimental assessment shows the type of insight that can be gained through the understanding of the environment in which the vehicle must operate and of the data that are available to the vehicle in order to model its environment. Performance evaluations are not just measures of success or failure, but rather windows into the algorithms and components within a system and how they function in different circumstances.

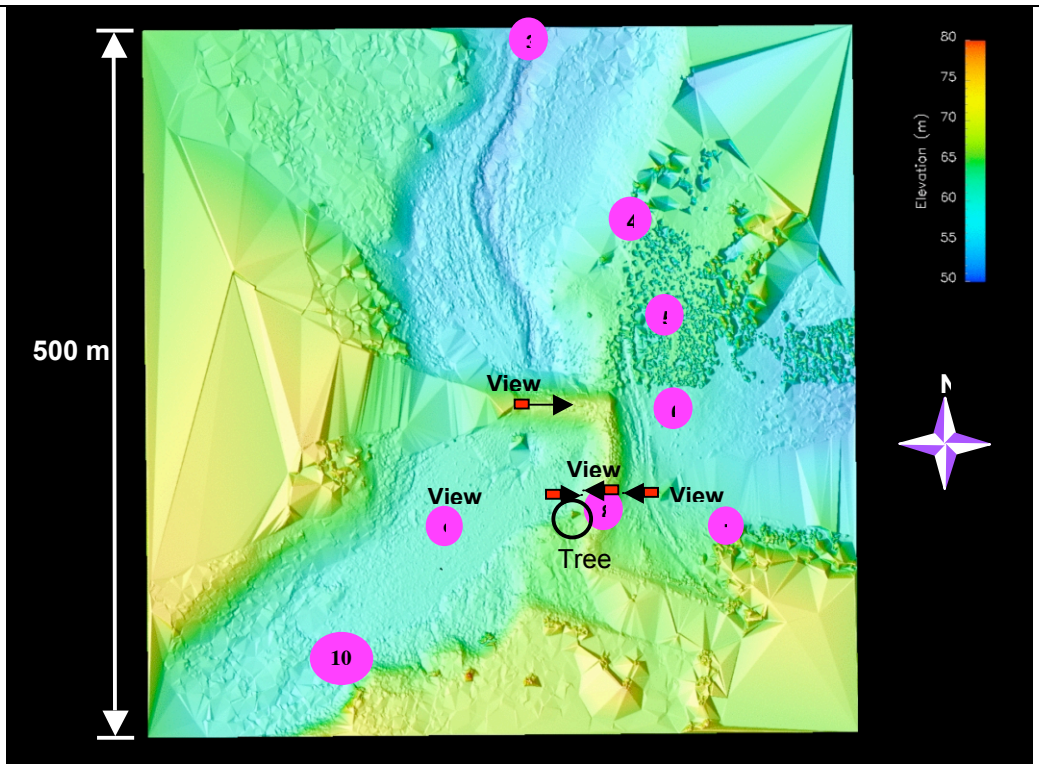


Figure 18: Dam region, Tooele. Locations where high resolution LADAR scans were taken. The numbered circles are waypoints on the black course. The circle near waypoint 8 corresponds to a tree that resulted in several emergency stops.

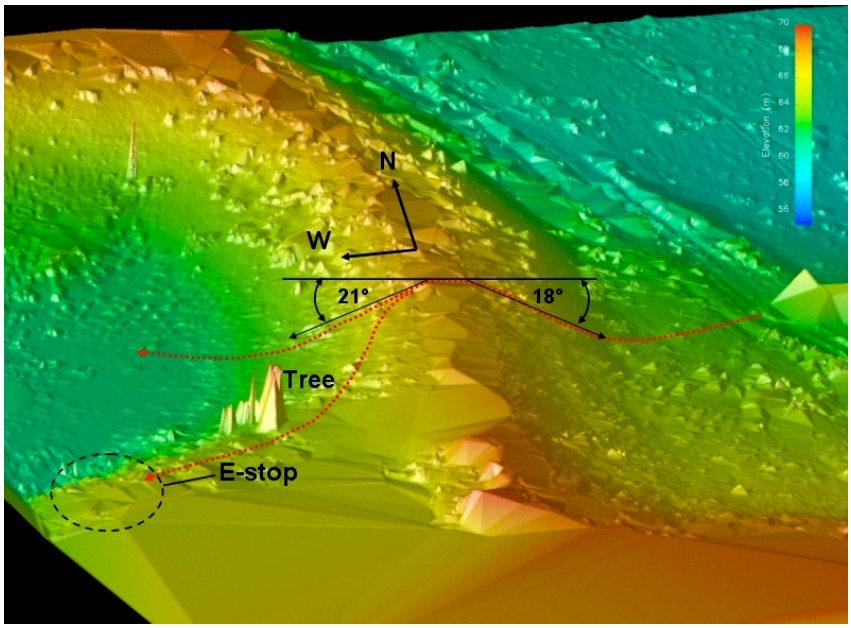


Figure 19: Dam region: Dotted lines show alternate paths taken by the XUV. One path ends in an emergency stop, while the other traverses the region successfully.

5. Performance Metrics for Intelligent Systems

The sections above have described examples of applied performance evaluations that were designed and implemented following a NIST-developed process. There is additional work that extends beyond NIST and looks at the problem of performance measurement in a more rigorous yet broader framework. One particular effort that brings together a broader group of intelligent systems researchers and developers is described briefly in this section.

As the community of engineers and computer scientists developing unmanned systems grows, it is imperative to arrive at a common understanding of how to define the performance requirements for a system and how to measure whether and how well the system meets the requirements. Alternative approaches – both in terms of algorithms and in components such as sensors – must be characterized through a commonly accepted method. To help address this need for a forum in which the unmanned systems developers share results and move towards a common body of work, NIST initiated a workshop series. With support from DARPA, an annual workshop series was launched in 2000 which convened researchers and practitioners and allowed them to exchange ideas and results pertaining to how to measure the performance of a variety of intelligent systems. Known as PerMIS (Performance Metrics for Intelligent Systems), the proceedings from this series of workshops have documented the progression from mostly theoretical views in the early years to more applied approaches that have been put into practice in the recent past.

Especially in its earlier years, PerMIS provided a forum for publication of more theoretical aspects of performance evaluation. Approaches to generic measures of intelligence through diverse schemes have been presented over the years. Readers are directed to the proceedings for more information [16].

As the field matured, increasing numbers of presentations became more application-oriented. Analyses of the performance of algorithms that underlie many mobile robot systems have been presented. These include formal comparisons of several leading optimization algorithms (random search, simultaneous perturbation stochastic approximation, simulated annealing, and evolutionary computation) [20], establishing guidance to practitioners for when to use or not use a particular method. Balakirsky and Kramer discussed various means for evaluating the performance of algorithms, using Dijkstra's graph search as an example, in [5] and [6].

Numerous papers have been presented that focus on evaluating the performance of mobile robots and are directly pertinent to this book's themes. For example, Albus explored the features of intelligence required by unmanned ground vehicles [2]. Several sessions dealt with mobility, planning, human-robot interaction, and simulation support for unmanned vehicles. Papers in the series also provided initial impetus to the Autonomy Levels for Unmanned Systems (ALFUS) described in Chapter 8. See for example [10].

6. Conclusions

The measurement of the performance of a ground vehicle, be it teleoperated, autonomous, or a combination of both, is an essential part of the overall progress in the technologies pertaining to unmanned robotic systems. A process for deriving the necessary measures, tests, and supporting infrastructure that is based on cognitive foundations and on domain-specific task analysis has been developed over many years at NIST. The key components within an overall testing program are

1. The development of the behavioral, knowledge, and sensing requirements for the system, beginning with a definition of the system's required application domain, tasks and missions. Through analyses of these, performance specifications are developed in conjunction with the end-user community.

2. Testbeds that provide situationally relevant as well as targeted challenges for the unmanned system. Testbeds can include physical instantiations that range from simplified representations of the target environment to realistic environments in which to test near-fieldable systems as well as virtual instantiations that include sensor data and simulation testbeds.
3. An infrastructure that enables repeatable, controlled, and reproducible measurement procedures. This infrastructure must support measurement of the vehicle and subsystem's performance against ground truth. Instrumentation that tracks mobile vehicles' positions and orientations is critical. Also important are advanced sensors that capture the environment in which the vehicle executes a mission and can be used to characterize the environment and provide insight into the vehicle's functioning. High-resolution elevation and feature maps also provide means of evaluating the vehicle's performance by providing ground truth.

Much work must still be done. As the unmanned ground vehicles become more sophisticated, the testing requirements become more elaborate. Additional instrumentation and more fully developed testing scenarios and environments must be developed. As the expected functionality of the systems grows to include more advanced mission-specific capabilities, the spectrum of contingencies that the systems have to be able to handle grows exponentially. It will be much more challenging to capture all the possible situations and test the vehicle against them. A solution may be to characterize the envelope of operation within which the vehicle is expected to function and devise tests that target the extrema of the envelope to measure where the points of failure are.

References

1. Alberts, D., Hayes, R., "Code of Best Practice for Experimentation," Department of Defense (DoD) Command and Control Research Program (CCRP) Publication Series, 2002.
2. Albus, J., "Features of Intelligence Required by Unmanned Ground Vehicles," *Proceedings of the Performance Metrics for Intelligent Systems Workshop,* NIST Special Publication 970, Gaithersburg, MD, August 2000.
3. Albus, J. and Meystel, A., Engineering of Mind: An Introduction to the Science of Intelligent Systems, Wiley and Sons, 2001.
4. Asada, M. et al. "An Overview of RoboCup-2002 Fukuoka/Busan" *AI Magazine*, 24(2): Summer 2003.
5. Balakirsky, S. and Kramer, T., "NOT(Faster Implementation ==> Better Algorithm), A Case Study," *Proceedings of the 2003 Performance Metrics for Intelligent Systems Workshop,* NIST Special Publication 1014, Gaithersburg, MD, September 16-18, 2003.
6. Balakirsky, S., and Kramer, T.R, "Comparing Algorithms: Rules of Thumb and an Example," *Proceedings of the 2004 Performance Metrics for Intelligent Systems Workshop,* NIST Special Publication XXX, Gaithersburg, MD, August 16-18, 2004.
7. Balakirsky, S., Scrapper, C., Messina, E., "Mobility Open Architecture Simulation and Tools Environment," *Proceedings of the 2005 IEEE Knowledge-Intensive Multi-Agent Systems International Conference*, Waltham, MA, April 2005.
8. Camden, R., Bodt, B., Schipani, S., Bornstein, J., Phelps, R., Runyon, T., French, F., And Shoemaker, C. Autonomous Mobility Technology Assessment Interim Report. Army Research Laboratory (ARL-MR-565). 2003.
9. Camden, R., Bornstein, J., French, F., Shoemaker, C., Bodt, B., Schipani, S., Runyon, T., Jacoff, A., Lytle, A., "Autonomous Mobility Technology Assessment Final Report," Army Research Laboratory Technical Report (ARL-TR-3471), April 2005.
10. B. Clough, "Metrics, Schmetrics! How the Heck do you Determine a UAV's Autonomy Anyway?," *Proceedings of the 2002 Performance Metrics for Intelligent Systems Workshop,* NIST Special Publication 990, Gaithersburg, MD, August 2002.

11. Herman, M., Report of the ARPA/NIST Workshop on Performance Evaluation of Unmanned Ground Vehicle Technologies, NISTIR 5237, National Institute of Standards and Technology, Gaithersburg, MD August 1993.
12. Herman, M. et al., Recommendations for Performance Evaluation of Unmanned Ground Vehicle Technologies, NISTIR 5244, National Institute of Standards and Technology, Gaithersburg, MD, August 1993.
13. Jacoff, A., Messina, E., Evans, J., "Performance Evaluation of Autonomous Mobile Robots," *Industrial Robot* 29:3, May 2002.
14. Jacoff, A. and Weiss, B., Messina, E., "Evolution of Metrics and Performance for USAR Competitions," *Proceedings of the 2003 Performance Metrics for Intelligent Systems Workshop,* NIST Special Publication 1014, Gaithersburg, MD, September 16-18, 2003.
15. Kitano, H., Asada, M., Kuniyoshi, Y., Noda, I., Osawa, E., Matsubara, H., "RoboCup: A Challenge Problem for AI," AI Magazine 18(1): Spring 1997, 87-101.
16. Messina, E. and Meystel, A., PerMIS Proceedings 2000-2004 (NIST Special Publications 982, 990, 1014, 1036).
17. Murphy, R., Casper, J., Micire, M. and Hyams, J. "Assessment of the NIST Standard Test Bed for Urban Search and Rescue," *Proceedings of the Performance Metrics for Intelligent Systems Workshop,* NIST Special Publication 970, Gaithersburg, MD, August 2000.
18. Oxford English Dictionary, Compact Edition.
19. Shoemaker, C. M. And Bornstein, J. A. The Demo3 UGV Program: A Testbed For Autonomous Navigation Research. Proceedings Of The IEEE International Symposium On Intelligent Control. Sept.,1998. Gaithersburg, Md.
20. Spall, J., Hill, S., Stark, D., "Towards an Objective Comparison of Stochastic Optimization Approaches," Proceedings of the Performance Metrics for Intelligent Systems Workshop, NIST Special Publication 970, Gaithersburg, MD, August 2000.
21. Wang, J., Lewis, M., and Gennari, J. "A Game Engine Based Simulation of the NIST Urban Search and Rescue Arenas." *Proceedings of the 2003 Winter Simulation Conference,* New Orleans, LA.
22. Wang, J., Lewis, M. and Gennari, J. (2003). Interactive Simulation of the NIST USAR Arenas. Proceedings of the 2003 IEEE International Conference on Systems, Man, and Cybernetics, Washington, DC, October 5-8.
23. Wheatley, T. and Michaloski, J., Configuration and Performance Evaluation of a Real-Time Robot Control System: The Skeleton Approach, Proceedings of the IEEE International Conference on Systems Engineering, Pittsburgh, PA, August 1990.

Chapter 10

Development of Semi-Autonomous Robotic Ground Vehicles: DoD's Ground Robotics Research Programs: Demo I through Demo III

Charles Shoemaker
Army Research Laboratory
cshoe@arl.army.mil

1. Introduction

The U.S. Army is actively engaged in developing and fielding its first generation of semi-autonomous, fieldable, robotic ground robotic vehicles. This is arguably a revolutionary course of action which has generated many new challenges and opportunities. This chapter will describe the research, technology, and field exercises that cleared the way and provided foundation technology for the deployable system oriented programs that followed. Specifically, the integrated research efforts in machine perception, intelligent control, and man machine interface will be described.

The motivations for these investments in robotics evolved over the last 20 years influenced by growing field experience with the technology and by the changing challenges faced by the United States Army. From 1989 to 1999, the Office of the Secretary of Defense (OSD) directly managed the autonomous mobility development efforts for Unmanned Ground Vehicles within the Department of Defense's (DoD) Joint Robotics Program (JRP). During this period it was said that robotics would address "dirty, dull and dangerous jobs." Current perceptions are that the use of robotics will have a major positive impact on the Army's ability to project force over long distances, save lives in hazardous missions, and permit troops to control larger areas through use of semi-autonomous robots. In many instances the robots will operate kilometers away from friendly ground forces; in others they may maneuver only a few meters away from the ground forces. Disruption or removal of explosive devices is a major application for currently deployed teleoperated robots. As autonomy is implemented reconnaissance applications, convoying of vehicles enabling crew rest or logistics resupply, and use of robotic vehicles that could follow squads of soldiers helping to reduce the amount and weight of equipment soldiers must carry (frequently exceeding 45 kg) are additional major focal points of the current program. Placing significant machine intelligence and through it, autonomous functionality, on robotic platforms enables equally important, but less obvious, benefits such as increasing the survivability of systems through removal of the crew to a safe location. This design approach will enable a single soldier or crew to control larger areas of terrain through supervision of multiple vehicles.

Teleoperation (with a few innovative exceptions described later in this chapter) requires near real time communication of very large amounts of information to the operator. Typically amounting to data rates of megabits per second, even after compression, this ordinarily requires use of transmission wavelengths shorter than 0.2 m that require the transmitter and receiver to stay within line of sight of each other. Other alternatives include use of fiber optic tethers and satellite communications (SATCOM). When fiber optics are used new challenges arise including the likelihood of damage to long tethers due to the potential for the tether to be tangled or cut by other vehicles operating in the same area. Alternatively, SATCOM, at the present time, is a very expensive option, which tends to limit its use to very low data rates more suited to control of semi-autonomous systems. Further, data latencies in signals processed through networks of multiple satellites (such as the TDRIS network in use by NASA) can be on the order of 250 to 750 ms. Although this time lag does not present problems for autonomous mobility, it is problematic when an operator is directly in the control loop.

On-board processing can dramatically reduce the amount of data which must pass through communication links and can reduce the impact of latency on the operator's ability to control the unmanned system(s).

Since the amount of data that must be passed is orders of magnitude lower than that in teleoperation, on-board processing can enable use of secure tactical data links through use of longer wavelengths that provide propagation beyond line of sight. Finally, the availability of systems with these capabilities may enable entirely new tactics and operational capabilities such as the use of mobile unmanned systems inserted behind enemy lines to collect information without risk to the armed forces.

2. Initial Technology and Programs

Some of the earliest programs in the modern era of Army autonomous UGV development were executed by the Human Engineering Laboratory (HEL) at Aberdeen Proving Ground, by the Defense Advanced Research Projects office (DARPA), and at TACOM, the Tank Automotive Command in Warren, MI.

From the robotics program's inception in 1980, HEL robotics management personnel developed a strategic alliance with the National Bureau of Standards (NBS) which in 1991 became the National Institute of Standards and Technology (NIST). The early HEL/NIST efforts addressed logistics applications for robotics while those at TACOM produced a prototype and type classified engineering package for a Robotic Obstacle Breaching Assault Tank (ROBAT). The ROBAT was a 45 ton turret-less M60 tank with a line charge mission package. A line charge is a linear array of explosives it is typically deployed (pulled across) a section of minefield by a short-range rocket. Once deployed on a sector of minefield it was remotely detonated, blasting a clear lane through the minefield. Once the cleared lane was produced, the ROBAT vehicle was teleoperated through the lane "proofing it" using mine rollers that were the width of the tank itself (i.e. proving that it is clear), and marking the edges of the cleared lane for following forces that could now traverse the cleared, proofed, and marked lane. ROBAT was teleoperated via a fiber optic or radio frequency communications link.

The HEL effort developed a vehicle equipped with a large powerful manipulator arm and wrist designed to enable autonomous unloading and transfer of palletized loads of logistics materiel. In use, the vehicle was parked in the center of an outdoor, "in the field" analog of an industrial "work cell". This program, titled the Field Material-handling Robot (FMR) developed a custom built, autonomous, 10 m long, manipulator with 6 degrees of freedom and a maximum payload of approximately 1000 kg as shown in Figure 1.

Figure 1: Field Material Handling Robot with 10 m reach, 1000kg payload, and NBS Real Time Control System for sensory-interactive control.

Martin Marietta Aero & Naval Systems located in Middle River, Maryland held the role of system integrator. MM's responsibilities also included manipulator design and fabrication, and system electronics and software. MM's subcontractor Moog Hydraulics designed the heavy duty, hydraulically actuated manipulator "wrist". The Standard Manufacturing Company provided the trailing arm-drive vehicle on which the manipulator was mounted. A team led by NBS, developed the sensors and software enabling autonomous control of the entire FMR system. Of greatest significance, the FMR represented the first "beyond the factory floor" application of the NBS developed RCS, a hierarchically configured, sensory interactive, intelligent system architecture employed state tables and a common sensory processing-world model–behavior generation modular design [1].

The RCS software was used to autonomously guide the end effector's fork tines into palletized materiel and rapidly transfer the pallets onto other trucks moving into and out of the work cell. The FMR's smart, sensor-equipped, fork tine, end effector (which superficially resembled the front end of a forklift truck typically used in warehouse environments) utilized ultrasonic, proximity, and optical sensors to identify the openings in the pallets into which the tines of a conventional forklift are normally inserted to lift and transfer pallets. In addition to the NBS leaders, the smart end effector development team included Tooele Army Depot and the Human Engineering Laboratory who assisted in the development and testing of the autonomous pallet engagement capability utilizing a very large industrial robot with a 200 kg payload (a Unimate 4000 series industrial robot) as a stand-in for the much larger FMR. The FMR's testing was completed and the unit was accepted by the Army in 1989, but autonomous ammunition repackaging in the field from long term inexpensive pallet storage configuration to expensive, precision magazines that would fit directly into weapon systems is an approach that has yet to be adopted by Army logisticians. The FMR was, however, the first successful military application of the RCS. As will be described later in this chapter, this architecture, in a more evolved form, now forms the foundation for the Army's first generation intelligent autonomous ground vehicles.

3. New Direction: Autonomous Mobility

This period from the mid to late 80's saw the initiation of autonomous ground vehicle technology development and demonstration programs by 3 groups:

- Defense Advanced Research Project Agency (DARPA) – Developers of the Autonomous Land Vehicle (ALV) with Martin Marietta Denver Aerospace and Carnegie Melon University (CMU) and a number of other contractors and universities
- Tank Automotive Research Development and Engineering Command (TARDEC) partnering with National Aeronautics Space Administration (NASA) Jet Propulsion Lab (JPL) from Pasadena, California Carnegie Melon University of Pittsburgh, PA, and FMC Inc. of San Jose, California, and
- Army's Laboratory Command (LABCOM), the parent agency of HEL with NBS.

3.1. The ALV Program

During the mid 80's DARPA initiated a set of programs to demonstrate the utility of a new generation of advanced computing technology known as the Strategic Computing program. In the vernacular of the time, the hardware component of this effort was referred to as developing "a super computer in a soup can." Three military applications programs were established to demonstrate the utility and progress of the high performance products of this program against a background of extremely challenging applications for the technology. These application programs addressed:

- global weather forecasting
- a virtual co-pilot for fighter pilots known as the Pilot's Associate
- a program focused on autonomous mobility for land vehicles known as the Autonomous Land Vehicle program.

ALV was the first program in DOD to focus major resources (exceeding $10 million) on autonomous mobility. The DARPA assessment that autonomous mobility for unmanned ground vehicles was an

extremely challenging computational problem was borne out in the field. Martin Marietta was competitively selected as the prime contractor. Although DARPA was the sponsor of the effort, the Army's Engineer Topographic Lab. was the Army contracting agent for the effort. Several universities e.g. University of Massachusetts, Carnegie Mellon University, University of Maryland and several companies such as Hughes Research Labs, Standard Manufacturing company and the Environmental Research Institute of Michigan (ERIM) focused their energies on this program. The vehicle, a large (about half the size of a school bus), very mobile chassis was developed by the Standard Manufacturing Company who had used the same suspension approach "trailing arm drive" on the previously discussed FMR program. Multiple Silicon Graphics workstations were employed for processing of images produced by stereo cameras and an early LADAR that was 1.8 m (nearly 6 feet) in length was used as an obstacle detection system (among other approaches such as stereo vision). The objective was to generate 3D images, and from them accurate maps, of the terrain immediately ahead of the vehicles. Individual cameras on pan/tilt heads were used to identify road edges for lane following. The vehicle was large enough to house researchers along with the workstations and was able to traverse roads at speeds of approximately 11 km/h while searching for obstacles about the size of hay bales. In addition to the early image processing technology, the program did generate a research community focused on autonomous mobility which became closely affiliated with the group associated with DARPA's long term Image Understanding program.

3.2. LABCOM's Team Program

The Army Laboratory Command was established in the mid 80's and was comprised of a variety of Army laboratories (exclusive of the Army's Corps of Engineers Labs). The LABCOM effort establishing a team comprised of HEL, the Ballistics Research Lab, the Harry Diamond Lab, Tooele Army Depot, Oak Ridge National Laboratory (ORNL) and NBS. This program was termed the Technology Enhancements for Autonomous Machines (TEAM) program.

Figure 2: Artist Concept of TEAM vehicle developed by LABCOM.

This team developed an all electric actuation and control package for HMMWVs which had the unique design feature of selectable robotic control. The operator could sit in the vehicle and drive it as an ordinary manned vehicle or he could install a few cotter pins, flip a few switches and convert it to teleoperated operation. This system also utilized an inertial navigation unit from the Palladin howitzer, the Modular Azimuth Positioning System (MAPS) which combined fiber optic gyros, accelerometers, and wheel

odometry to allow teach-playback mobility (the term "retrotraverse" was coined by LABCOM to describe this function) with path retrace accuracies on the order of 0.1% of distance traveled.

Once again the TEAM vehicles (see Figure 2) were controlled using RCS now configured for control of vehicles with autonomous mobility and automatic target acquisition (not manipulator control). This system also provided a mode of teleoperated driving over compressed data links that allowed multiple "dialable" control by the operator of a number of selectable data compression parameters including level of resolution, frame rate and numbers of grey scales. This permitted experimental evaluation of the performance decrements associated with various types and levels of data compression. Early results demonstrated that compression down to levels near 60 kbps was highly desirable because it enabled use of communication links with good propagation characteristics beyond line of sight. This technique however forced drivers to slow down to 16 km/h or less due to the poorer image quality it afforded.

TARDEC efforts during this time focused on conventional stereo vision and computer assisted teleoperation techniques, and autonomous control based on ALV technology. Each of these three techniques was implemented on a separate HMMWV. The Computer Aided Remote Driving (CARD) technique developed by JPL used stereo cameras to build a 3D model of the terrain ahead (acquired while the robot was stopped). The operator could view the 3D scene using shuttered goggles providing him "depth cues" and designate a string of way points for the vehicle. Because the waypoints were identified in three dimensional space, enough information was available to allow the robot's planning software to define the path and allow the robotic HMMWV to drive through the trajectory linking the waypoints autonomously. Much the same technique has been used on planetary rovers by JPL. For the NASA missions the low speeds, comparatively short ranges and intermittent stops required to acquire data were not a severe handicap. Alternatively, the ability to perform this function using only a pair of still frame images enabled this mode of operation to be supported over NASA's Deep Space Network.

3.3. The OSD Joint Robotics Program

As the 80's ended, Congressional Language was created that vested OSD's Land Warfare Office with the responsibility for all ground robotics efforts in the Department of Defense. Approximately $20 million were split between:

1. early efforts to develop and deploy a teleoperated weapon or reconnaissance package equipped small wheeled vehicle less than 375 kg platform the Tactical Unmanned Ground Vehicle (TUGV), and
2. research efforts focused on advancing technology required for autonomous robotic vehicles.

The original roadmap OSD created for the research component of the Joint Robotics Program described a program to run between 1989 and 1991 titled Demo I and a second effort which would ramp up as Demo I concluded termed Demo II.

4. DEMO I

In concept, Demo I merged the ongoing LABCOM and TARDEC efforts into a program that demonstrated six robotic HMMWVs (see Figure 3) with three types of autonomous mobility, and in one instance with automated reconnaissance and target acquisition [2]. The LABCOM group included the members of the TEAM program described above (HEL, Ballistics Research Lab, Harry Diamond Lab, Tooele Army Depot, ORNL and NBS). Each of the HMMWVs was controlled by a team member located in a large command and control vehicle the Unmanned Ground Vehicle Control Testbed (UGVCT) built for TARDEC by FMC Corporation. This system was also referred to as the Remote Command Center (RCC). The LABCOM vehicles could also be controlled using a smaller operator workstation "the table top controller." Their technology was derived from LABCOM programs which had built an intelligence fusion station the Combat Information Processor (CIP). Each approach to autonomous mobility was demonstrated on a separate HMMWV platform.

Figure 3: OSD Demo I Robotic HMMWV technology testbeds; Testbed primary developers from left: HDL, BRL, TARDEC, TARDEC, NIST, JPL

The HMMWVs were configured as follows:

HMMWV #1
All electric "optionally manned" robotics package the "LABCOM actuation package" which integrated steering, brake, throttle, transmission, transfer case, and parking brake, all of the electronics for control were located in an environmentally conditioned module located between the front seats, this HMMWV was equipped with a video compression package allowing the operator to independently set, frame rate, resolution, and color as opposed to black and white imagery for driving, Using this package the HMMWVs could be driven in a low data rate (64 kbps Vs 60 megabits for uncompressed color video teleoperation mode at low speed e.g. 16 km/h). The system also used an RCS implementation for vehicle control developed by NIST. The Army provided funding for this extension of RCS sponsorship for semi-autonomous robotic vehicle control [3] which included an automatic target acquisition module using 4 CCD cameras providing a field of view of 20 degrees. Operation of this module, which employed RCS, required the HMMWV to be parked and used target motion as the discriminator for target detection. Moving targets used for Demo I were located at modest ranges of up to 1 km.

HMMWV #2 was similar to HMMWV #1 and included a turret capable of pointing a Modular Laser Integrated Engagement Systems on a separate turret that could "fire" a laser at distant targets equipped with receivers capable of triggering a smoke charge to simulate an accurate target acquisition and engagement.

HMMWV #3 was developed by FMC Corp for TARDEC. It used an actuation package developed by Kaman Electronics originally developed for installation on target vehicles used on Army test ranges. It employed, the previously described Computer Aided Teleoperation (CARD) developed by JPL.

HMMWV #4 used the same actuation design as HMMWV #4 but integrated early autonomous road following technology derived from the ALV program to road identify edges.

HMMWV #5 was similar to HMMWV #1, used RCS, and integrated and featured a "retrotraverse" mode of autonomous mobility that allowed an operator to drive a route up to a few kilometers and repeat it autonomously on command. This system relied on a ring laser gyro/triad of accelerometer inertial

reference system which also utilized an odometry sensor to detect wheel rotation. A new experimental turbine engine powered smoke generator developed by the Army was integrated into this system.

HMMWV #6 was similar to HMMWV #3 but used an improved version of CARD that supported longer path lengths, and was integrated with an Army Single Channel Ground Air Radio System (SINCGARS) tactical radio.

4.1. Control Stations

The control stations used for DEMO I included the Robotic Command Center (RCC) and the Table Top Controller (TTC).

The RCC utilized an M109 Self Propelled Howitzer chassis as the vehicular platform. It integrated two driver work stations and a commander's station. It was designed to support all of the mobility modes outlined for the previously described Demo I HMMWVs. Mission package software was included to support reconnaissance and surveillance, decoy operations, chemical detection, battlefield obscuration (smoke), convoy following and anti-armor weapon systems. Sun workstations were used to support path planning performed on a digital terrain database. 3D goggles were also integrated to present stereo imagery to operators for operational modes such as CARD. In the RCC, drivers used a workstation much like that used to control a tank integrating a steering yoke and brake pedals for control. The Table Top Controller was based on a Sun Single Processor Scalable Processor (SPARC) for a Combat Information Processor, for data fusion and a "suitcase" controller for control of mobility functions.

4.2. DEMO I Conclusion

The OSD vision and that of the program management staff began with an array of individual robotic vehicle programs along with many programs focused on robotic subsystems. At its conclusion in late spring of 1991, Demo I presented the robotics community and Senior DoD officials with an integrated review of the state of the art in mobile robotics. It allowed evaluation of several modes of vehicle control including advanced, computer-aided teleoperation, semi automated control modes such as CARD, and highly autonomous modes of control such as retrotraverse and autonomous road following. It also presaged investigation of many of the most challenging issues the Army faced as it approached robotics for Future Combat Systems including levels of autonomy, operator workload, efficient modes of human intervention as a means to intervene in the control of autonomous vehicles experiencing difficulty, and multiple vehicle control. Both the RCC and the table top controller supported simultaneous control of two vehicles. The RCC added the dimension of controlling moving vehicles from a moving mobility platform, within which the operators' were located. This capability was demonstrated by TARDEC during trials at Camp Roberts, California.

As the concluding event for Demo I, five demonstration days were held during a 2 week period on the Army's rugged hill test site in Churchville, Maryland. The individuals controlling the demonstration vehicles were Army civilians and contractors. Many of the component technologies developed in Demo I were transitioned some in integrated packages, others individually, to follow-on efforts. Project Mustang (see below) was the best example of this type of program continuity. Demo I was declared a success by the OSD sponsors who stated that it "set the standard for the next major robotics program Demo II".

4.3. Project Mustang

From the beginning of Demo II through Demo A an Army officer stationed at DARPA served as the Demo II Program Manager (PM). The Request For Proposal (RFP) issued by DARPA to select the Demo II integration contractor included a requirement for a capstone demonstration at Ft. Hood, Texas. This demonstration was originally envisioned as an event using a large amphitheater-like range as a setting rather than a process involving military technology users in the effort. During 1992, the Demo II PM position was vacated as the first PM retired from the Army. A second DARPA PM (who had been the Demo I program manager) was selected by OSD and DARPA senior management. The positive experiences connected with use of map based interfaces during Demo I resulted in a decision by the new

Demo II PM to seek a tighter connection with military subject matter experts. At the same time an Army officer who had obtained a graduate degree in Mechanical Engineering with a concentration in Robotics from the Massachusetts Institute of Technology was appointed as an Armored Battalion Commander at Ft Hood. These two individuals working with the III Corps Commander LTG Paul Funk and Division Commander BG Eric Shinseki (later the Chief of Staff of the Army and the chief architect of the robotics intensive Future Combat Systems (FCS) Army transformation program) developed a program plan that closely connected individuals with armored cavalry scout experience to the Demo II program.

This interaction involved two processes:

1. A near term effort (Project Mustang) that brought robotics capabilities into the field with Armored Cavalry Scouts, and
2. Scout involvement in Demo II events and the culminating Demo II event at Ft. Hood.

For Project Mustang, Demo I assets were used to develop a testbed for an autonomous robotic scout. Specifically a Demo I HMMWV (HMMWV #2) was equipped with improved acoustic based automatic target acquisition. A laser engagement system was added (see Figure 4) and a simulated weapon system was integrated onto the laser pointing turret.

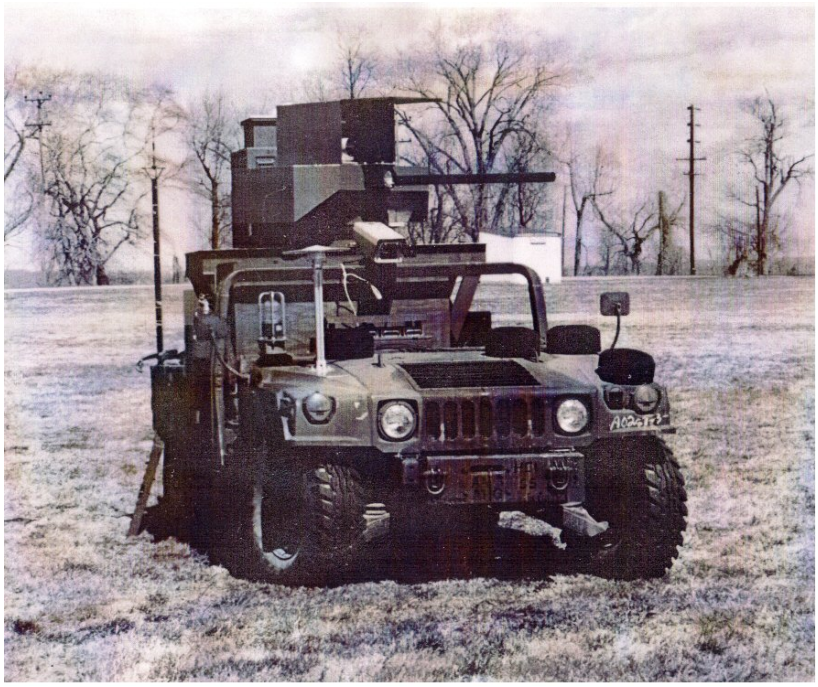


Figure 4: Project Mustang configured HMMWV with SATCOM, laser engagement and a weapon system visual modification.

The Demo I TTC was integrated into a HMMWV (ambulance configuration). An advanced Sun computer was used for digital terrain data base planning tools. GPS technology was integrated with the Demo I developed inertial reference system for an extended range retrotraverse capability. The first Project Mustang experiment titled "Blackjack Challenge" took place in Spring 1993. The soldiers' ability to quickly adapt to the graphical user interfaces was very impressive and led to a decision in Fall 1993 to conduct a second experiment "Blackwolf Challenge" to enable Armor Scouts to serve as the operators of the robots instead of tactical advisors in Fall 1993. Mobile communications experts at NASA's Jet Propulsion Laboratory were brought in to integrate a Ka band satellite communications capability using NASA's Advanced Capability Technology- Satellite (ACTS).

During this period Army Scouts from Ft. Hood participated in several Demo II workshops (see below) providing insights into the Army's conduct of scout operations. These insights and the technology required

to enable them were subsequently integrated into the Demo II platforms. Two aspects of autonomous control: inherently low data rate requirements and robustness to time lags in the SATCOM network (latency) enabled the integration of the ACTS satellite data link with the HMMWV. At the operator control unit end of the data link a satellite dish approximately 2 m in diameter was pointed at the geosynchronous satellite. On the HMMWV vehicle a one degree of freedom tracking antenna was located in a small plastic dome about 0.3 m in diameter and mounted on the highest point on the HMMWV. This supported continuous tracking while the HMMWV was on the move and prevented structures on the vehicle from interfering with the line of sight path required for signal transmission. The ACTS system provided approximately 125 kbps of data rate with latencies in the system of as much as 1.5 s depending on other users of the network and the location of NASA relay satellites in the Tracking and Data Relay Satellite System (TDRSS).

A major improvement in operational utility resulted from the use of the ACTS system since line of sight between two ground systems was no longer a communications constraint. There were relatively few tall trees or structures in the test areas used at Ft. Hood. This enabled the military operators to place the robotic HMMWV with near complete freedom.

The military operators found the operator control unit easy to use and with informal instruction could teach the vehicle path segments to be repeated using retrotraverse, and adjust parameters in the RSTA system that used a 3-5 micron Forward Looking Infra Red system (FLIR) as its primary visual sensor. This FLIR used Indium Antinimide as the detector within the imaging detector element. The RSTA module also included visual CCD cameras and a stepper motor driven pan tilt turret. A Northrop developed acoustic target detection system was integrated and helped to cue the imaging RSTA sensors on the HMMWV.

In combination, these subsystems enabled an operator to plan a predetermined trajectory between observation points as much as 1 km apart (limited by the “0.1% of distance traveled” limitation of the MAPS system and the approximately 30 m accuracy of conventional (non-differential) GPS that was integrated on the Blackwolf Challenge system to reduce the effects of drift on inertial sensor data [4]. The retrotraverse capability was ideally suited to the dispensing of tactical smoke. Shifting winds create a need to move smoke generators to alternate locations. Many modern tactical smoke systems employ particulates added to the basic smoke. These provide rapid attenuation of signals within the visible spectrum, and also interfere with signal transmission in many other areas of the electromagnetic spectrum. An ability to use an onboard reference system for navigation which does not require imaging of local terrain features can be very useful for tactical operations in battlefield obscuration missions. One major limitation of unaided retrotraverse is the inability to detect changes in the environment since the original path was traveresed. Movement of dismounted troops, vehicles and changes in the terrain caused by artillery and other weapon systems create real limitations for unaided retrotraverse.

In order to execute the RSTA mission the operator identified targets of interest on the ground using knowledge of the vehicle’s orientation and pointing of the turret housing. Once oriented, the sensors enabled a systematic search of areas of interest using a combination of change detection, moving target detection, and high thermal contrast as automatic target acquisition strategies. Since the vehicle was being used in a tactical exercise it was essential that it be both detectable by opposing force “players” and have a realistic signature equating to its use of a simulated 40 mm cannon. The use of a Modular Integrated Laser Engagement System (MILES) provided this capability. The vehicle could be knocked out of action by other systems firing coded laser energy at an array of MILES detectors mounted around the vehicle. Similarly, it could fire its own laser (boresighted with the FLIR) at targets, triggering a flash and smoke signature on the HMMWV intended to represent the type of signature a weapons system of that caliber selected would possess. This replicates the potential for the vehicles location to be disclosed once it fired; and, indeed, during the actual “engagements” carried out during several nights of BlackWolf challenge it was quickly determined that if the HMMWV fired it was quickly detected and “killed”. On the other hand, if it was used as a forward sensor to detect and relay information including location and type of probable targets it was found to be a valuable asset, frequently the first to identify incoming targets. The soldier operator was signaled when a likely target was detected. He was shown a still frame target since the 150 kbps data rate supported by ACTS was much lower than that required for live video.

This exercise and the soldiers' demonstrated aptitude to quickly learn to control the system resulted in a unanimous agreement among the Demo II principals to fundamentally change the nature of the interaction planned for Demo II at Ft. Hood. Rather than a demonstration being manned by expert contractors at Ft. Hood, a three year long interaction was planned and executed with military scouts attending major Demo II workshops and providing insights regarding their tactical skill and experience. When Demo II was conducted at Ft. Hood during the Summer of FY96 the soldiers were the operators of the robotic systems rather than spectators.

5. DEMO II

5.1. Introduction-Through Demo Alpha

The Demo II program was the second element of the original OSD vision for robotics research [5]. Its primary focus was autonomous mobility. OSD developed a teaming relationship with DARPA for this program with OSD and DARPA each contributing funds (in the ratio dollars of approximately $2 of OSD funds for each $1 of DARPA funds). A Memorandum of Understanding (MoU) was entered into between the two agencies specifying the shared investment approach along with a stipulation that the Demo II Program would be managed by a DARPA Program Manager in that agency's Software Intelligent Systems Technology Office. Management of Demo II began at about the same time period as Demo I but its funding profile was the inverse of that for Demo I.

While Demo I made extensive use of existing robotics assets and used a group of Army Labs as the integrator, Demo II followed a much different and more traditional industry approach in that a major aerospace contractor, Martin Marietta (the ALV lead contractor) was competitively selected as the integration contractor. However, in a major departure from the aerospace model, many other organizations, that were perceived to have valuable technologies to contribute were selected as "co-contractors" (in contrast to sub-contractors). Many had been principals in ALV. Significant contributors to the program included:

- **Academic co-contractors** e.g. Carnegie Melon University, Colorado State University, University of Maryland, University of Massachusetts, University of Michigan, Georgia Institute of Technology, University of Pennsylvania, and Cornell University;
- **Corporate organizations** e.g. STA Inc., Jet Propulsion Lab, AAI, Odetics Corporation, Sarnoff Research Center, Amber Corporation, Hughes Research Labs, Lear Astronics, Rockwell, Environmental Research Institute of Michigan, Honeywell, Advanced Decision Systems, and Alliant Tech Systems; and
- **U.S. Government agencies** e.g. Engineer Topographic Lab, Army Night Vision Electro-optics Sensors Directorate, TARDEC, and ARL.

Although many of the ALV team members contributed to Demo II there was very little reuse of Demo I assets or technology. New actuation and control approaches were adopted; RCS was not used as an integrating architecture. A Demo II "tiger team" developed a software architecture the team described as incorporating hierarchical and "reactive" behavior based characteristics.

Demo II organized its primary schedule milestones into a series of technology demonstrations termed Demo Alpha (A) (1993), Bravo (B) (1994), and Charlie (C) (1995) conducted at the Martin Marietta facility and test site at Waterton, Co. The culminating Demonstration, Demo II, was held at Ft. Hood Texas in 1996.

Demo Alpha's milestones were closely tied to development and demonstration of the basic automotive infrastructure for the Demo II mobile robotics testbed vehicles designated the Surrogate Semi-autonomous Vehicles or, more simply the SSVs. The first SSV was developed and demonstrated at this time. During Demo Alpha the first SSV was demonstrated in a teleoperated mode without autonomous functionality. Basic RSTA camera pointing was demonstrated. The Waterton, Colorado site was an outstanding

demonstration venue. It included a dedicated building for the Martin Marietta personnel involved in Demo II which included a high bay area for vehicle assembly, and integrated testing. Moreover there was nearly a square mile of spectacular rugged terrain located within a short distance of the integration building and a viewing area from which much of the site could be seen by VIPs.

The terrain included dirt and paved roads, rugged hills, meadows, and enjoyed microwave relay data links through the site that had originally been installed in support of ALV. The vehicles' location throughout the Waterton site tracked on a high resolution Digital Terrain Elevation Database (DTED V) 1 m resolution database originally generated for ALV this terrain had also been digitized to levels of resolution below 1 m and placed in a digital terrain database. Much of the perception, and intelligent control technology used throughout Demo II was developed by a team from CMU's Robotics Institute. During Demo A, autonomous road following was demonstrated using a subsystem termed Autonomous Land Vehicle in a Neural Network (ALVINN). The SSV was teleoperated onto a road (dirt or paved) and the ALVINN modes was manually selected. The vehicle would then drive autonomously down the road using images collected from a single camera. Images were grabbed using Datacube vision processors and processed using the ALVINN software. Between the on-road segments, Demo A used a low data rate teleoperation technique developed by CMU. This mode of control, Supervised Telerobotics Incremental Polygonal Earth Geometry, was more frequently referred to as STRIPE.

STRIPE provided the operator a still frame image of the scene ahead of the vehicle. Lacking stereo or other range data, only a 2D view was available. Data integrating the missing range information became available once the vehicle mounted cameras were displaced as the vehicle drove from point to point following the points the operator designated. The vehicles planning software was utilized to build a trajectory between the points once the vehicle (and cameras) had displaced. STRIPE used a single camera and a single still frame requiring only very limited incremental data rate capacity for the operator to generate the original waypoints. The operator manually switched back and forth among the mobility modes in Demo A. Demo A also used a GPS system integrated through a Kalman filter integrating odometry, and inputs including data from a flux gate magnetometer compass and two inclinometers.

Demo A also served as a significant "rallying point" for members of the Image Understanding community developed by DARPA over many years. Indeed a parallel program DARPA's BAA 93-01 Advanced RSTA Technology Effort was put in place as a funded effort during Demo Bravo. This effort brought forth a new group of "co-contractors" focused specifically on RSTA technology; this effort developed techniques for motion stabilization (RSTA) on the move, adaptive Forward Looking Infra Red system (FLIR), target detection, advanced sensor fusion including FLIR, targeting LADAR, video, and polarization with advanced senor based planning. Although not covered in detail in this chapter, these efforts provided significant new options to the MM system integrator [6].

5.2. Demo Bravo

A major focus of efforts leading up to Demo B held in Summer 1994 was to enable basic autonomy over cross country terrain using stereo vision, to upgrade planning capabilities enabling the vehicle to shift among the various modes of control automatically, to integrate a much improved Operator Control Unit, and to provide significantly upgraded RSTA capabilities. The Demo Bravo vehicle was a standard "slant back" four seat HMMWV modified to increase available volume for onboard computing by nearly an additional 20% [7] (see Figure 5).

Figure 5. A pair of Demo II robotic HMMWV testbeds (Demo Bravo-Charlie configuration).

Demo II Bravo comprised a two day demonstration which for the first time included significant participation by officers and Non-Commissioned Officers (NCOs) from Ft. Hood based on the Project Mustang initiative.

A single robotic HMMWV was controlled by an NCO supported by a MM engineer. The OCU was located in an S-250 communication shelter (Figure 6) mounted in the cargo bed of the HMMWV which was used from Demo B throughout Demo II as a dedicated OCU vehicle, i.e. it did not support conventional tactical functions.

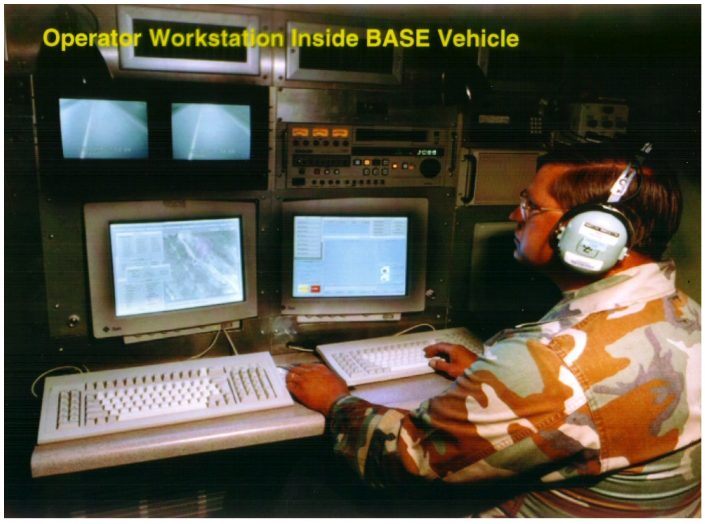


Figure 6. Interior of Demo II operator control unit located inside an S-250 communications shelter carried on a manned HMMWV.

The OCU used Sun workstations complemented by an array of smaller monitors. For the first time in the OSD managed robotics program, the OCU developed for Demo B incorporated a graphical user interface integrated into a digital terrain database. The symbology displayed was identical to that used by the Army to plan manned scout missions. Typically, a military mission is planned on a map, employing a symbol known as a Phase Line, to synchronize movements of multiple elements of a maneuver force. At this location manned forces wait until other elements catch up or move to previously designated locations. In the Demo B OCU, the phase line was identical pictorially to that used by the military and could trigger a similar pre-programmed wait function.

Demo B took place on a route nearly 3 km in length. During the first kilometer, three types of sensor based mobility were demonstrated during clear-weather daylight conditions. The first two modes included dirt and paved road following using a single black and white Charge Coupled Device (CCD) daylight camera. In this instance single lane roads were traversed. No lines of any type were painted on the road. The camera's output was processed to identify the road edge. Contrast between the road and the surrounding terrain was the primary information used to detect the edge. This type of processing enabled the robotic HMMWV to move along the road at a maximum of 25 km/h. The system shifted smoothly from movement on the dirt section of the road to the paved section. A third transition took place as the HMMWV drove off the road at a location coincident with a previously identified GPS location. The vehicle drove sequentially through multiple way points. This was the third mode of control. At the beginning of the run landmarks on the horizon were matched with those generated during a previous run. Because GPS (when the location of the receiver is not moving) provides only location data and not orientation data, this initial matching was a valuable source of calibration information. During this period the vehicle was not performing obstacle detection and could easily collide with an object such as a fallen tree or another vehicle on the road. Once the initial mobility modes were demonstrated, basic RSTA capabilities using the FLIR as the primary sensor were performed. An extended mobility run using only Global Positioning System waypoints as a reference was also demonstrated. The vehicle was directed to return to the road as the next element of the demonstration.

The demonstration continued employing a mode of control enabling the operator to intervene to assist the autonomous vehicle if it encountered difficulties or if the operator wished to make a decision based on information not processed by the SSV e.g. selecting a fork in the road based on the operators perception of road surface qualities. STRIPE was used to support this mode of control. Depending on sight lines this information could be used to drive the robot distances up to 25-40 m.

The next demonstrated mobility mode utilized stereo vision to detect obstacles and place them in a "map" structure. This subsystem, developed by JPL, was part of the research it was pursuing for implementation on planetary rovers. The Demo II system used two fixed camera's for stereo; they were based approximately 30 cm apart (the stereo baseline) on a bar located above the HMMWVs windshield. Potential obstacles were detected using disparity information resulting when the two images were registered. Trigonometric analysis of this information was used to generate obstacle maps. Artificial intelligence based planners were used to modify the vehicle trajectory as required, departing from the baseline path only to the extent required to avoid the obstacle and return to the original trajectory generated by the operator's initial planned path.

In Demo B processor implementation, the considerable computational load entailed by these processes resulted in processing rates on the order of 3 frames per second. For a vehicle of the size of the HMMWV these experiments resulted in vehicle velocities less than 10 km/h. Obstacles 1-2 m in diameter (simulated boulders made of fiberglass) were used to allow rapid repositioning and prevent damage to the vehicle if stuck. During Demo B obstacle detection ran only when the vehicle was moving cross country.

Several basic modes of RSTA capability were performed using a FLIR developed by the Amber Corporation known as the Radiance model. These FLIR approaches used thermal contrast and target motion as the two primary discriminators and as was the case in Project Mustang the vehicle had to be stationary in order to use the RSTA capability.

5.3. Demo C

This group of events took place during July 1995. From a program standpoint this effort focused on demonstration and evaluation of subsystem capabilities to enable decisions regarding the culminating demonstration at Ft. Hood a year later. Although the majority of the technologies demonstrated represented efforts to make the B component technologies more robust, several new capabilities were integrated and shown especially those involving multiple (two) vehicles. Formation control was shown that utilized GPS integrated with an inertial reference system as a reference (not local sensing). This technique would maintain relative vehicle positions with precision and distance controllable independently. A second new capability termed "map sharing" allowed the obstacles located in the map framework of the lead HMMWV to be communicated to a second HMMWV. This capability allowed the lead vehicle's "experience" to benefit that of subsequent followers. For example, if the lead vehicle encountered a dead end situation the followers could avoid the problem area. This second new capability was the first demonstration in a military ground robotics program of an emergent "tactical behavior." This movement termed "Hill Cresting" was invoked to provide a level of capability beyond movement from point A to point B (Point B was typically selected through a line of sight analysis based on the digital terrain database).

In a true tactical operation, it is frequently undesirable to move to the crest of a hill. In that situation a vehicle could be silhouetted against a high contrast sky background. In the hill cresting mode a point an arbitrary distance (typically 20 m) short of the desired observation point, was selected. The SSV moved to that location and began to pulse its laser rangefinder in the direction of the expected target. Short distance, incremental, vehicle displacement continued resulting in relatively short range distances being read by a laser rangefinder. As the vehicle neared the top of the hill a suddenly shift to distant ranges roughly equivalent to the range of the notional targets is detected. The point at which the rapid shift takes place represents the location of the vehicle at what is termed the "military crest" of the hill. This integrated use of mobility and RSTA sensors (in Demo II comprised of the Amber FLIR, a visible zoom camera, and laser rangefinder) was unprecedented and presaged later developments with similar objectives.

DEMO II Charlie included three days of subsystem evaluations in preparation for a critical downselect to the technology components of the concluding DEMO II event in 1995. It concluded with an integrated demonstration similar to that performed in Demo Bravo. The integrated demonstration was under the control of a military scout from III Corps at Ft. Hood, but included simultaneous control of two HMMWVs from the HMMWV mounted OCU. The culminating Demo Charlie demonstration was observed at close range (from a following HMMWV) by senior personnel from the US Army's Infantry School at Ft. Benning, Georgia. The demonstration was compelling enough to cause skepticism to shift to advocacy from this important organization during the span of the 45 min. demonstration. The Demo II robotic SSVs were configured so that there was room for a driver and passenger in the front two seats of the vehicle. Since the vehicles were, in essence, rolling laboratories there was every expectation that during the testing of the vehicles personnel would be located in the driver and or passenger seat. This was not unique to Demo II but it had been true of ALV, Demo I and Project Mustang. While Demo B had been performed with the Robotic HMMWV unmanned, Demo C was conducted with a person in the passenger seat. This individual could not steer the vehicle even if he wished to but the passenger could hit a safety kill switch or rapidly reboot electronics if necessary. As Demo II C concluded the program manager solicited concepts for candidate demonstration "vignettes" to be performed at Ft. Hood the following summer. This was the first use of the term vignette associated with a robotics demonstration and related to an approach that integrated significant new technical capabilities into a representative tactical scenario.

The culminating demonstration for the Demo II program took place at Ft. Hood, Texas during July 1996 under the supervision of a second program manager from ARL (The first ARL PM had been tasked by OSD to develop plans for Demo III). Three demonstration vignettes based on activities during Demo C were selected, implemented and conducted. Significantly, the early interactions with the scouts of Ft. Hood had matured into a longer term teaming relationship with the Mounted Maneuver Battle Lab at Ft. Knox, KY.

Demo II was more than a technology demonstration (see testbeds in Figure 7). Close interaction with the Army's Training and Doctrine Command (TRADOC) had changed what was planned as a technology

demonstration into a Battle Lab Warfighting Experiment. This involved significantly more structure, preparation, and sophisticated data analysis than a typical technology demonstration. The summer heat at Ft. Hood which on occasion exceeded 104 degrees F, proved to be a significant new challenge to test personnel and the robotic equipment. The electronics on the vehicle were comprised primarily of commercial grade components placed in environmentally controlled shock, and electromagnetic interference isolated enclosures. The three vignettes selected for demonstration were an Urban Assault mission, a Forward Observer mission, and a mounted Scout operation.

Figure 7: DEMO II HMMWV technology testbeds. From left: Demo II-A. SSV, Demo II-OCU workstation in S-250 shelter mounted on HMMWV, Demo II-C SSV, Demo II-C SSV, Demo II-C SSV.

The Urban assault mission required the simultaneous deployment of all 3 SSVs around the perimeter of the Ft. Hood Military Operations in Urban Terrain (MOUT) site. From vantage points originally selected using the digital terrain display on the operator interface, FLIRs were utilized to detect and pass information regarding the location of "enemy" soldiers in the town.

The Forward Observer mission required a single unmanned SSV to move to a previously selected location inside an Artillery impact area for a mission that would call in live fires on a target selected by the operator of the SSV. The vehicle operated in the obstacle detection mode during the mission and successfully detected and negotiated a difficult (negative obstacle) dry stream bed as it moved to the GPS location designated by the operator. The Amber FLIR was used to detect difficult non-moving targets. It was used in conjunction with the vehicle's GPS unit was used to generate data required for a "call for fire."

The mounted scout mission was the most complex of the three missions. It was performed in a "force-on-force" mode meaning that the robotic vehicles were utilized by a friendly "Blue" manned scout force which employed them against a "Red" or enemy force. The exercise was conducted in a fairly flat maneuver area containing scrub trees up to 2.5 m in height and brush. This terrain was typical of that found in many of Ft. Hood's maneuver areas and was used for manned vehicle training operations. Observers from OSD and many other Government agencies observed the exercise from a nearby mesa.

The terrain analysis tools developed as part of Demo II proved highly effective in enabling the Blue Force to maneuver the SSVs to positions of tactical advantage. The Blue Force used the SSVs to "probe" the enemy force using routes that were concealed from the view of the Red force by fairly subtle terrain features. The SSVs were employed with obstacle detection functioning full time. Improvements in

processing algorithms and hardware enabled the SSVs to operate at speeds up to a maximum of approximately 8 km/h when the obstacle avoidance subsystem was in use. Obstacle detection based on the previously described stereo vision system proved capable of detecting the trees that were the primary mobility challenge.

Seen from a distance 1 km away by the observers, the outlines of the SSVs were virtually indistinguishable from the manned HMMWV scout vehicles. However, the difference in movement between manned and unmanned HMMWVs was readily observable. The experienced scouts tended to move rapidly from one position of concealment to the next using bursts of speed (only marginally faster than the SSV robotic vehicles). In contrast, the SSVs moved with a relatively linear velocity profile. Negative obstacles such as holes, ditches etc. proved much harder to discriminate with the stereo technology. Not withstanding these limitations, the SSVs were believed to be very useful by the Blue Force whose only real "complaint" was that the FLIRs on the SSVs did not have adequate range. They were limited to detecting red force vehicles at 600-750 m and had definitely not been a major area of technical emphasis within the program.

The Commanding General of Ft. Knox's U.S. Army Armor Center found, in the preface to the experiment's final report, that "This experiment found significant potential value added to the warfighter in increased situational awareness, reducing risk to manned platforms, increasing the tempo of operations and protecting the force". A corollary finding by the Integrator of Mounted Battlespace was that the "Biggest payoff for "the" maneuver force is "a UGV that requires minimal (near zero) involvement after mission assignment".

6. DEMO III

During the concluding year of Demo II OSD tasked the Army Research Lab's Robotics Program Office (RPO) to formulate a proposal for the next technology development and demonstration in robotic vehicle evolution namely Demo III [8]. A workshop to assist in this process was held at the Institute for Defense Analyses in Alexandria, Virginia. This meeting was hosted by the ARL RPO and was part of an effort to develop an updated, long range strategic technology vision for the OSD Joint Robotics Program. ARL sought and received participation from the DoD "user" community with representation from each of the three Services and the Marine Corps. Army participants included ARL, TARDEC, the Aviation and Missile Command (AMCOM), NASA's JPL, NIST, DARPA, and Academia [9].

The group identified a number of technology challenges believed to be critical for future DoD application of robotics. These included:

- the need for UGVs to be able to keep up (speed-operational tempo) with maneuver units,
- obstacle detection and avoidance,
- image stabilization,
- image understanding,
- situation awareness,
- navigation and landmark recognition,
- cooperative target detection,
- dynamic world modeling,
- reactive behavior and re-planning,
- system architecture standardization, and
- communication with robotic vehicles.

This information in concert with extensive prior experience applying the technology through Demo I, Project Mustang, and Demo II, was integrated into a proposal to OSD for a new effort titled by OSD as Demo III. Specifically, the program's technical objectives were aligned against the following challenge areas:

- autonomous mobility,
- intelligent system architectures,
- man machine interface,
- mission packages (reconnaissance and target acquisition to permit realistic field exercises),

- communications, and
- mobility platforms.

Target capabilities of the program as proposed by the RPO to OSD were:

- autonomous scout vehicles with off-road cross country speeds of 25 km/h in daylight conditions and 16 km/h at night,
- an operational focus on the scout mission,
- a robotic scout vehicle smaller than a HMMWV (based on user feedback from Demo II),
- multiple vehicle control with levels of autonomy enabling one operator to control 4 platforms,
- in order to focus maximum resources on critical technology development, selection of mobility platforms was limited to off the shelf commercial or military vehicles or an adaptation of an existing platform,
- similarly options for actuation and low level control system were limited to adaptations of existing systems,
- use of the 4D/RCS. Alternatives to this architecture were considered if documentation supporting alternate selections was provided
- use of digital terrain not to exceed DTED II i.e. levels of terrain resolution limited to (30 m postings), and
- use of tactical radio systems with data rates not to exceed 125 kbps. Typical ranges for the tactical radios were required to be in the 15-25 km range.

The ARL RPO developed a plan to place resources for a competitive procurement for Demo III at the Army's TARDEC. TARDEC is the Army's land vehicle system integrator. RPO personnel believed that a renewed involvement of that group in robotics was critical to enable the transition of Demo III component technologies to eventual deployment. (At the conclusion of Demo I, TARDEC activity in robotics was significantly reduced for the duration of Demo II). Senior TARDEC personnel supported this proposal and assisted in the development of the required procurement documentation and processes. Over the life of Demo III ARL transferred approximately $22 million to TARDEC.

A Request for Proposals (RFP) was published in 1998 to begin the competitive procurement process for Demo III. In November 1998, TARDEC awarded the Demo III contract to a team led by Robotic Systems Technology, teamed with SAIC (the SAIC team included many of the Lockheed Martin Demo II group), Sarnoff Labs, and Perceptek. RST was acquired by General Dynamics Robotic Systems during 1999. Demo III's Preliminary Design Review (PDR) was held on 28-30 July 1998. Between the contract award and the PDR tight connections were established between the Demo III contractor team and ARL's separately funded Concerted Technology Thrust (CTT). CTT members included ARL, NIST, JPL, and Ft. Knox's Mounted Maneuver Battlespace Battle Lab. Approximately 12 months after the Demo III contract award the program's Critical Design Review (CDR) was held from 17-19 November 1998.

Demo Alpha utilizing 2 XUVs was conducted less than 10 months after the CDR. Given the complexity of the technology development and integration this was an exceedingly short period of time to build the first 2 prototype vehicles and demonstrate them before a VIP audience. The XUV platforms, true to the requirements in the original RFP, were adaptations of an existing platform. This vehicle the Modular Detection and Response-External (MDARS-E) platform had been developed for PM Physical Security Equipment.

6.1. Demo III Technical Characteristics

6.1.1. Demo III Robotic Platforms

The Demo III vehicles known as eXperimental Unmanned Vehicles (XUVs shown in Figure 8) utilized 4 wheel drive and 4 wheel steer (to meet the tight turning radius requirements). The wheels were hydrostatically driven by a hydraulic pump powered by a VW NATO spec Turbo-Diesel engine. The platforms were Ackerman steered. Each wheel had independent suspension. Six sealed electronics

enclosures, cooled using folded fin technology were placed in a configuration with 3 on each side of the vehicles. Total XUV weight was between 1150 and 1300 kg (approximately 2500 and 2800 lb).

Figure 8: XUV Chassis.

6.1.2. Demo III Autonomous Mobility

The Demo Alpha XUV configuration utilized daylight CCD and cooled thermal imagers (FLIRs) in a stereo configuration to meet day and night requirements. Demo's Bravo and Charlie included a custom built LADAR. The first generation of the Demo III LADAR utilized scanner components developed by Schwartz Electro Optics. Much of the design expertise and all of the software for processing LADAR information was developed by NIST and GDRS using, respectively NIST's experience with the Dornier 1 Hz EBK LADAR acquired in 1995 by ARL, and GDRS' MDARS-E experience with a laser line scanner. While early versions of the LADAR were mounted, shock isolated but fixed, to the chassis, later versions were mounted on a dedicated pan and tilt head. Stereo sensors were mounted on an independent pan and tilt platform. Still later, LADAR, stereo CCD and stereo FLIR were all integrated into a single enclosure with pan and tilt motors.

6.1.3. Demo III Software Control Architecture

Five levels of the 4D/RCS were implemented during Demo III. This system provided the onboard "intelligence" for the vehicle to react appropriately in complex, dynamic situations. It combined existing, *a priori* information, primarily map and operator control unit inputs, with real time feeds from multiple sensors. Real time information from sensors included data from navigation sensors including INS/GPS, inputs from the vehicle sensors e.g. steering angle and wheel slip/odometry, speedometer, LADAR data, stereo vision data, CCD, FLIR, and radar data for obstacle detection and collision avoidance. A critical dimension of this control schema is the task decomposition which takes place at each level of the system. This breaks down extremely complex problems such as maneuvering a group of unmanned system through unknown, changing terrain and situations in building block fashion into solvable problems. This decomposition makes complex problems tractable and supports the superimposition of order in the form of rules of engagement, and relationships between events including cause and effect relationships. Relative priorities for mission objectives and the scheduling of events necessary to accomplish them are integrated as well. The development of multi-resolution maps, corresponding to different levels of the control hierarchy, is a critical mechanism for integrating real time information with pre-existing data.

The 4D/RCS system, and the maps and planning tools it integrates, support reasoning through multiple scales, temporally, spatially and tactically [10]. This means that in the future high levels of 4D/RCS implementation will enable the development and execution of "big picture" plans supporting combined operations of humans and robots over tens of kilometers, through hours of operation, involving large

numbers of men and machines e.g. Battalion level exercises might include 400-600 men and dozens of manned and unmanned vehicles. At the other end of the time horizon the robotic vehicles have to generate commands with time horizons measured in milliseconds and provide the data necessary to control actuators e.g. steering and brakes, to avoid obstacles that appear as the vehicle is traveling along at high speed. Between these two extremes, 3 or 4 intermediate levels of control levels are required to control a single platform. As one goes up the levels of the control hierarchy vehicles may be controlled individually or in groups. The groups need not be heterogeneous i.e. they can span UAVs and UGVs. By Demo Charlie in Oct 2001 the five levels that were controlled using the RCS framework included the:

- Servo level
- Primitive level,
- Subsystem planner e.g. mission package,
- Vehicle planner, and
- Section planner.

Within each level of the 4D/RCS architecture, a WM exists that includes all of the *a priori* information appropriate to that level's time and distance horizon. A Sensory Processing module interprets raw sensor data needed for decisions at the specified level, and finally, a Behavior Generation Module accesses all of the available information within the World Model including that from real time sensory interaction with the surrounding terrain, other vehicles, and the robot's own state to make decisions as appropriate. Significantly, for the operator interface discussion to follow all of the information at each level of the hierarchy is readily available in state-table form for operators who (figuratively) need to enter into the system to understand the present situation.

In the years prior to Demo III, interactions among NIST, ARL and later the Bundeswehr University (UBM) [11] in Munich (the German Defense University), the RCS architecture had been application engineered to become well matched to requirements for a robotic scout vehicle. Prior to DEMO III there was a working vision tested on the Demo I HMMWV at NIST and on German autonomous navigation (AUTONAV) testbed vehicles at UBM and Dornier Aerospace as to how to support "tactical behaviors" in the RCS and later the 4D/RCS architecture [12]. General Dynamics Robotics Systems too had in-depth experience with 4D RCS, one of the company's principals had been a research associate at NIST. The company had applied the architecture effectively in prior programs. Through the previously described CTT program, ARL supported a dedicated tech transfer effort to enable implementation of 4D/RCS into Demo III and provided the means for NIST and ARL to collaborate on the development of the Demo III LADAR. Because of the first experience with the Dornier LADAR the first generation LADAR developed under Demo III is frequently referred to as the Gen. II LADAR.

Although much of Demo III was oriented on autonomous mobility involving movement from point to point, the vision of tactical behaviors would later be used to expand upon tasks such as the aforementioned "hill-cresting" of Demo II and the later much more complex "route reconnaissance" mission capabilities developed by NIST in conjunction with ARL and Ft. Knox.

6.1.4. Man Machine Interface

The man machine interface, frequently referred to as the Operator Control Unit (OCU), as seen in Figure 9, used throughout Demo III was based on the "look and feel" of the Demo II interface. For Demo III, the interface was reduced in size to fit in the front passenger seat windshield area of a HMMWV, with a couple of suitcase sized enclosures containing additional computing hardware in the cargo area of the HMMWV as opposed to the S-250 communications shelter configuration used in Demo II. As specified in the RFP, the primary objective for the Demo III interface was to support the simultaneous mobility and mission control of four XUVs. This was consistent with the tactical doctrine which was at the time projected for future operations. There was, therefore, minimal emphasis on providing the means for an operator to intervene in a mission. This situation was to change significantly with the new doctrine envisioned by FCS (to be described later in this chapter).

Figure 9: Demo III Operator Control Unit mounted in front passenger windshield area of HMMWV.

The emphasis on planning and executing a mission for 4 XUVs based on a DTED II database, and conducted over a 56 kbps communication link had profound implications for the Demo III OCU. Section level planning (as specified in the 4D/RCS hierarchy) enabled simultaneous control of multiple XUVs. The interface used was a reconfigurable, multi-function display that utilized touch screen inputs with context sensitive map and planning features so that at any point in a mission the screen specific to that mission presented the operator with appropriate data interpretation, control, and display and replanning tools. This type of interface used to advantage the state table base of 4D/RCS. Once again, military symbology was used to facilitate transfer of the mission planning skills soldier operators already possessed, to the control of autonomous UGVs. A ruggedized keyboard was also present for the use of the system in the field as a research testbed.

Forewarned of hazardous situations resulting from other military systems that inadvertently blocked the OCU HMMWV driver's vision through the windshield, the Demo III OCU designers enabled the entire OCU to be pivoted out of the way and stowed in a compact configuration between the driver and passenger seats which would not block lines of sight. As was the case for the ARL components of Demo I, Project Mustang, and Demo II, operator inputs to the OCU were made only when the OCU HMMWV was parked. (TACOM's Robotic Command Center component of Demo I was unique in that it did support teleoperation and semi-autonomous control of UGVs while the command center was on the move). Later programs such as ARL's Collaborative Technology Alliance in Robotics (R-CTA) specifically emphasized control while on the move.

6.1.5. Navigation References

DEMO III used navigation references combining military GPS units known as Precision Lightweight GPS Receiver (PLGR) with a Smith's Industry inertial reference unit that used tuned mechanical gyros as the inertial reference. These navigation references were combined though a Kalman filter to improve the accuracy, update rate, and resistance to drift of the system beyond that available individually from these components.

6.1.6. Communications System

All communications to and from the XUVs (with the exception of redundant safety radios developed by Tooele Army Depot near Salt Lake City, Utah) were performed using the Army's Near Term Digital Radio Systems (NTDR). This was a packet switched digital radio operating in the UHF frequency band. As used in Demo III's final Charlie configuration the network topology of this radio system provided each of the 4 platforms with approximately 60 kbps of data rate. The UHF propagation characteristics of this radio provided superior communication over long distances (up to 15 km) on sparsely foliated terrain. In the deciduous forest area of Ft. Indiantown Gap PA where the final Demo III demonstration took place, communication of 3-5 km was a dramatic improvement over the line of sight limited digital modem radios (1-2 GHz frequency) that had been used earlier during Demo Bravo and Demo Alpha.

6.1.7. RSTA Mission package

As mentioned earlier, the scouts operating Demo II vehicles during the Battle Lab. Warfighting Experiment were clearly disappointed in the capabilities the Demo II RSTA sensors had provided them. This problem tended to mask the performance of the vehicles when used in a scout mission. In response to this, the government RFP for Demo III specified a RSTA package capable of acquiring and classifying vehicles and dismounted personnel at long distances, while the XUV was stationary and on the move. This chain of events and GDRS's market survey led them to specify the WESCAM 14QS (WESCAM is a trademark product name…the company is now a business unit of L3 communications) a fully gimbaled and stabilized RSTA package for the purpose of meeting these requirements. This was a high-end system originally designed for use in manned and unmanned aircraft.

For the ground mobility application planned for DEMO III the manufacturer recommended an additional level of shock and vibration isolation. This RSTA system included an InSb FLIR, laser rangefinder and color camera with zoom lens. To the optical sensors GDRS added acoustic target acquisition capabilities which were incorporated with active noise cancellation technology to reduce the ambient noise floor and increase the sensitivity of the acoustic array for targets generating steady state e.g. the roar of a tank engine and the sound emitted by its road wheels, or impulse noise signatures e.g. noise pulses generated when a weapon is fired. The acoustic sensors aided with target classification (e.g. is the moving target a tank or a truck?) and could be used to generate target location information which could be used to slew the optics toward the general area of a target.

7. DEMO III Performance

Demo Alpha was conducted in October 1999 at the Perryman test area of Aberdeen Proving Ground [13]. Perryman is flat with long (5 km) straight high speed paved roads and multiple dirt roads and trails that wind through dense stands of trees and brush. Stereo implemented by JPL was the obstacle detection mode available at the time and vehicle speeds were approximately 8-12 km/h while in this mode. As originally programmed in preparation for the Alpha exercise, the vehicle would come to a stop if it could not perceive a clear route forward. As a response to this frequently encountered problem a code change was made that caused the vehicle to back-up an arbitrary distance and take a new "look" at the terrain. The back-up distance could be specified. Ten meters was a frequently specified distance. In many instances, this simple maneuver provided the vehicle a slightly different "look" at the terrain than did the original route. Frequently (obviously dependent on the terrain) this enabled the vehicle to find a clear way forward without any input from the operator.

Negative obstacles such as holes and ditches were quite difficult to detect with the stereo technology. During Demo Alpha, two vehicles maneuvered simultaneously and independently. Unlike Demo I and II before it, the XUVs were not designed to carry a passenger, so each vehicle was truly unmanned, although chase vehicles with people onboard followed behind the XUVs for safety purposes. Basic RSTA was demonstrated as were OCU screens. This demonstration served primarily as an integration point forcing basic system integration to take place early in the program allowing more time and resources for Demo III's primary research tasks. Remarkably, GDRS and the team had two vehicles designed and built, with

basic functionality operational in front of a VIP audience approximately twenty one months after contract award.

Demo Bravo was conducted at Ft. Knox, KY on Range 10. This terrain was used to train tank crewman and had been significantly eroded by many years of 70+ton tank maneuvers. Additionally, the course was also characterized by significant elevation change. Despite the much more difficult terrain, major performance improvements were seen in the autonomous mobility demonstrated at Demo Bravo. These improvements were directly attributable to the use of the new LADAR and its key processing software populating obstacle maps with data, much higher in resolution, and more frequent in update rate, to the 4D/RCS architecture. The increasing complexity and volume of the data generated by sensor systems such as the LADAR imposed new burdens on the planners used to select appropriate routes for the vehicle. Research in this area [14] also addressed optimality across multiple levels in the multi resolution system used to describe the vehicles world model. The LADAR (a Gen. II model with the Gen. I designator being reserved for the Dornier EBK system used within the ARL/NIST/UBM AUTONAV program) provided rapid update (10 Hz as compared to the 1 Hz stereo and EBK systems) and provided accurate, unambiguous range data. The autonomous mobility planning software which combined perception data, the vehicle's location, the on board digital terrain model, and the nominal path originally planned for the mission with other operator selectable parameters had also been considerably improved since Demo II. This resulted in improvements in vehicle speed of up to 15-20 km/h with the obstacle detection and avoidance perception system functioning.

Demo Bravo also demonstrated perception based vehicle following on and off road. Adaptive velocity control with the vehicle adjusting its speed based on its initial mission plan and the local environment was also demonstrated for the first time. The WESCAM units were also demonstrated in an early high data rate mode of operation and although there were some cooling problems observed during the trials in late summer heat at Ft. Knox, moving target detection capabilities were impressive.

Demo Charlie took place during October 2001, on schedule, as outlined in the original RFP to industry. Six days of demonstrations were performed at Ft. Indiantown Gap, PA. Fort Indiantown Gap was selected for the demonstration due to the wide variety of rugged Appalachian foothill terrain it encompassed [15]. It was also accessible from the Pentagon, approximately an hour and a half by military helicopter and four hours by car. The terrain included significant relief with open and densely vegetated fields, multiple dirt roads, trails and tree lines. There were also several one lane bridge stream crossings. Each day of demonstrations featured what came to be called the "ride-behind experience" during which an observer was placed in a HMMWV that closely followed the maneuvering XUVs as they crossed the terrain (see Figure 10). From this vantage point one could appreciate the challenges of the terrain while simultaneously viewing an engineering display in the HMMWV that provided a visualization of the terrain as detected by the autonomous mobility sensors. The operator's planned path and the instantaneous path generated in response to obstacles could also be seen allowing visitors to anticipate the XUVs next move. This visualization developed during the Demo Bravo time period was especially useful to the team members who would integrate and evaluate new software and sensors into the system.

At the beginning of each demonstration, plans for 4 vehicle XUV missions were generated in the operator interface by a soldier operator. All 4 XUVs were operated simultaneously by this single soldier. Vehicles were assigned several waypoints corresponding to the path the operator selected. Frequently the planning tools themselves would generate the necessary waypoints if the operator chose to identify only a few points (such as the end point of a mission). In this instance the operator chose an endpoint for the mission and assigned weighting factors to a number of planning constraints to actually generate the planned trajectory. Typical weighting factors could include parameters such as "prefer road" or avoid slopes exceeding X%. Named Areas of Interest (NAI) for reconnaissance were also identified during this process. This generated the requirement to define Observation Points (OP) and led to the demonstration of a new tactical behavior that utilized a three step process, extending the hill cresting maneuver of Demo II, to identify and then maneuver to an OP. Initially terrain analysis was performed at the OCU to identify positions with appropriate sight lines and ranges from the target. Once the vehicle approached the designated location, it would survey the direction of interest with the LADAR to determine if any brush was in evidence, once that eventuality was ruled out the vehicle would fire a laser rangefinder in the direction of the target and if

an appropriate range return was seen the OP location would be considered confirmed. Fully autonomous RSTA capabilities were demonstrated at Demo Charlie.

Once autonomous mobility brought the XUV to the planned location, the system would generate multiple high magnification narrow field of view still frames, within a specified area of regard. These would be reconstituted as a wide field of view image "mosaic" back at the operator interface. This capability combined a combination of terrain reasoning, motion detection, and continuous tracking of multiple "targets" moving and stationary, with an ability to provide the operator a still frame "image chip" of the target allowing him to make a positive identification of a target. While he was assessing the target the system would continue to track it and other targets. This entire sequence was performed over the low data rate communications link. The significant amount of on-board XUV image processing necessary for these functions was performed using specialized image processing hardware and software developed by the Sarnoff Corporation.

7.1. **Mobility Performance**

Individual vehicle runs during a demonstration were on the order of 2 km in length. Obstacle detection ran continuously during these exercises and the adaptive velocity control of the vehicles enabled them to increase speed in open areas where a path could be followed that closely corresponded to the intended route of the system. Similarly, if the terrain did not allow movement in open areas in the direction of the next waypoint forcing the XUV to move through more difficult terrain, its speed would be reduced. Once diverted from the nominal trajectory generated by the planner the LADAR would actively "look" at the terrain near the robot to identify paths enabling a return to the original trajectory. In general, the XUVs were programmed not to move into areas they could not "see" (generate obstacle detection data.) After the primary "runs" of Demo III were completed, the vehicles were shown in a leader-follower move. These runs were based on a string of GPS waypoints generated by the leader. The following XUV would maneuver through these points. Since obstacle detection mode was operating, if a vehicle or some other barrier was put in place to block the path generated by the leader, the follower would autonomously depart from the original path, maneuver around the obstacle and return to the leader's path in as short a distance as the local topography allowed [16].

During 2001, senior Joint Robotics Program technology managers in the Office of the Secretary of Defense decided to transfer JRP resources for technology development to the Army. Based on the results of the robotics research programs described earlier in this chapter and early results from Demo III the decision was made by OSD to place these funds under the control of ARL which in turn placed these resources under the management control of its Robotics Program Office. Senior Army research management further committed to provide an equivalent amount of Army funds to underwrite the Army's robotics program. The agreement between the Army and OSD specified that these resources be used to support research providing autonomous mobility capabilities to the Army, Navy, Air Force, and Marine Corps.

Figure 10: Demo III XUV Demo III Charlie configuration-2001.

7.2 Results of Demo III

Demo III, and its constituent technology and team members have had a significant impact on military robotics. From its inception, government and industry program management worked to make good use of prior, relevant work in the field. This and the enduring efforts of the Demo III and CTT team members enabled the rapid development of capable testbed vehicles and core technology in perception, intelligent control, and man machine interface that has been the foundation for the ongoing Collaborative Technology Alliance in Robotics. XUV testbed vehicles are in use not only by ARL, but by CERDEC for countermine research, AMRDEC for weapons integration activities, by NIST, and by TARDEC for advanced mobility applications such as Leader-Follower applications (see Figure 11).

Perhaps most significantly, during the 2003-2004 period, the platforms and technology were used in the conduct of an intensive series of field exercises, which characterized with unprecedented experimental rigor the performance of its autonomous mobility functionality on three different terrain sets. These terrain sets included high desert, Appalachian foothills in the dead of winter, and maneuvers in military housing areas configured to represent many of the mobility challenges associated with operations in urban areas. Moreover, the experimental results of over 600 mobility runs demonstrated that the technology was capable of operating nearly 90% of the time in an autonomous mode. These experiments also addressed the workloads associated with operator interventions when the XUV technology could not cope with a particular challenge triggering the vehicle to contact the operator. During the remaining 10% of the time (and distance) of the experimental runs, the remote operator was able to extricate it from difficulty over a low data rate communications link, essentially teleoperating it for distances on the order of 50 m. These interventions produced modest operator workloads as rated by the operators on NASA's TLX workload scale. A discussion of the TRL exercises is included in Chapter 9. The reader is referred to an ARL Report titled "Autonomous Mobility Technology Assessment Final Report" for a much more complete description of the experiment and its results [17].

During 2004, the vehicles were independently tested by the Army's Lead System Integrator (LSI) as an element of the FCS program. The LSI test results buttressed the experimental conclusions that the technology had reached a level of robustness that merited significant new Army investment. At this writing the Army is making the largest investment in its history in tactical autonomous robotic systems. Moreover, it has committed major investment in the Autonomous Navigation System (ANS). A major contract addressing the development of a true military system for application to a variety of Army vehicles was awarded to GDRS in 2004. This effort builds directly on the technology developed the long term Army/OSD research investment in the 4D/RCS system and multiple generations of LADAR that were matured through Demo III, its companion integration contract, the concerted Technology Thrust, and the Robotics-CTA.

The ANS program is designing, building and evaluating a new generation of autonomous mobility sensors and software configured in a fully ruggedized "Mil-Spec" package. The fundamental technology built through the OSD-Army program from Demo III to the present CTA is the backbone of the ANS. It is seeing reuse in the architecture, perception and intelligent control baseline for ANS. The ANS will provide perception and intelligence to three vehicles in the FCS era, the robotic MULE, the Armed Robotic Vehicle and a number of variants of the Manned Ground Vehicle. Other programs such as MDARS are moving the technology forward toward early deployment this decade. These developments will carry autonomous mobility forward in a pivotal role in response to the defense challenges this nation will face in this new century.

Figure 11. View of Current Generation XUV-2006.

References

1. J. S. Albus, C. McLean, A .J. Barbera, and M .L. Fitzgerald, An architecture for real-time sensory interactive control of robots in a manufacturing environment, *4th IFAC/IFIP symposium on Information Control Problems in a manufacturing environment,* Gaithersburg, MD, October 1982.

2 C.M. Shoemaker, *Robotic Vehicle Mobility: Technology and Research Testbeds, IVHS* America, 1993.

3. Szabo, S., Scott, H., Murphy, K., Legowik, S. A., and Bostelman, R. A., (1992) "High level mobility controller for a remotely operated unmanned land vehicle", *Journal of Intelligent and Robotic System,* 5, pp. 63-77.

4. Murphy, K. N., Juberts, M., Legowik, S., Nashman, M., Schneiderman, Scott, H., and Szabo, S. Ground Vehicle Control at NIST: from teleoperation to autonomy, Proceedings *of the 7th Annual Space Operations, Applications and Research Symposium,* Houston, TX, August 3-5, 1993.

5. Munkeby, S., Shoemaker, C., Chun, W., "Unmanned Ground Demo II Program", 1994 *IEEE National Telesystems Conference*, San Diego, CA, May 1994.

6. Chun, W., and Jochem, T., "Unmanned Ground Vehicle Demo II: Demonstration A", *Unmanned SYSTEMS: The Magazine of the Association for Unmanned Vehicles Systems,* Volume 12, Number 1., Winter 1994. Also in SPIE Volume 2352, Mobile Robots IX, Boston, MA, Nov. 1994.

7. Chun, W., Lynch, R., Shoemaker, C., and Munkeby, S., "UGV-Demonstration B", *Unmanned SYSTEMS: The Magazine of the Association for Unmanned Vehicles Systems* Volume 13, Number 3, Summer 1995.

8. C. M. Shoemaker, *The Use of Unmanned Ground Vehicles: Demo II Lessons Learned,* Unmanned Vehicles Symposium, Paris, France 1997.

9. C .M. Shoemaker, J. Bornstein, *Demo III Program: A Testbed for Unmanned Ground Vehicle Autonomous Navigation,* IEEE Symposium on Intelligent Control 1998.
10. Albus, J., Lacaze, A., and Meystel, A. Multiresolutional planning with minimum complexity, *Proceedings of the 1997 International Conference on Intelligent Systems and Semiotic, Gaithersburg, MD,* pp.151-156, 1997.
11. E. Dickmanns, "Vehicles capable of dynamic vision," in *Proceedings of the International Conference on Artificial Intelligence,* pp.1557-1592, 1997.
12. Albus, J. (1998) "4D RCS: a reference model architecture for Demo III", Version 0.1, NISTIR *5994, National Institute of Standards and Technology, Gaithersburg, MD*.
13. Shoemaker, C., Bornstein, J., Myers, S., Brendle, B., "Demo III: Department of Defense Testbed for unmanned ground vehicle mobility", *Proceedings of the SPIE Col. 3693, AeroSense Session on Unmanned Ground Vehicle Technology,* Orlando, FL, 1999.
14. Lacaze, A., (2002) Hierarchical Planning Algorithms, *Unmanned Ground Vehicle-IV,* Proceedings of SPIE Vol. 4715.
15. Murphy, K., Abrams, M., Blakirsky, S., Chang, Tommy., Hong, Tsai., Lacaze, A., and Legowik, S. "Intelligent Control for Off-Road Driving" *First International NAISO, Congress on Autonomous Intelligent Systems,* Deakin University, Geelong, Australia, 2002.
16. Bornstein, J.A., "Army Ground Robotics Research Program", Unmanned Ground Robotics IV-Proceedings of SPIE Volume 4715, 2002.
17. Camden, R., Bodt, B., Schipani, S., Bornstein, J., Runyon, T., French, F., Shoemaker, C., Jacoff, A., and Lytle, A., "Autonomous Mobility Technology Assessment Final Report", (2005) *Army Research Laboratory Technical Report ARL-TR-347*.

Chapter 11

Epilog

The fundamental processes of cognition, perception, knowledge representation, reasoning, decision making, planning, and control are understood, at least in principle. There is a large and growing community of researchers in the cognitive sciences. The computational mechanisms of the brain and the mental processes of mind are the subject of intensive scientific investigation.

A number of futurists [1,2] have predicted that machine intelligence comparable to that of the human brain will be available on laptop-equivalent computers before the middle of this century. The development of human level intelligence in machine systems for manufacturing, transportation, construction, agriculture, mining, and health care will dramatically improve productivity, increase quality, reduce costs, and accelerate the rate at which goods and services can be produced. Intelligent weapons systems will revolutionize warfare. Truly intelligent machine systems will enable a new industrial revolution that will transform the modern socio-economic system. The potential benefits are inestimable. Intelligent systems may become the signature technology of the 21st century, comparable in importance to the automobile, airplane, and nuclear power in the 20th century.

There are good reasons to believe that research in autonomous vehicle systems is an important waypoint on the path to human equivalent machine intelligence. There are a number of reasons for this:

First, autonomous driving is a problem domain for which there is a large potential user base, both in the military and civilian sectors. This translates into significant and sustained funding for research and development.

Second, autonomous driving is a problem domain where physical actuators and power systems are readily available. Wheeled and tracked vehicle technology is mature, inexpensive, and widely deployed.

Third, autonomous driving is a problem domain for which the technology is ready. The development of real-time LADAR imaging makes it possible to capture the 3D geometry and dynamics of the environment. Combined with video cameras and radar technology, LADAR imaging enables a solution to the perception problem. The continued exponential growth rate in computing power per dollar cost is bringing the necessary computational power within the realm of economic practicality. This enables a solution of the computational problem. Cognitive modeling and intelligent control theory has advanced to the point where a scientific understanding of intelligent systems is emerging. This enables a solution of the system engineering problem.

Finally, autonomous driving is problem domain of fundamental scientific interest. Locomotion is perhaps the most basic of all behaviors in the biological world. Locomotion is essential to finding food and evading predators throughout the animal kingdom. The brains of all animate creatures have evolved under the pressures of natural selection that rewards successful locomotion behavior. It is therefore, not unreasonable to suspect that building intelligent mobility systems will reveal fundamental new insights into the mysteries of how the mechanisms of brain give rise to the phenomena of intelligence, consciousness, and mind.

There are, of course, contrary opinions. There are some who contend that autonomous driving does not involve a significant cognitive component. The argument is sometimes heard: "How hard can it be? My sixteen-year-old daughter can do it." True, but this grossly underestimates the cognitive capabilities of the sixteen-year-old human brain. A sixteen-year-old human has phenomenal capabilities for perceiving the world, understanding complex situations, judging distances and velocities, comprehending signs and signals, and performing maneuvers such as changing lanes, passing, merging, and negotiating intersections.

These are significant cognitive capabilities. To fully understand how the sixteen-year-old human brain works would be a scientific breakthrough of the first magnitude.

Other say "Driving is simply getting from point-A to point-B." True, but this is not simple. Between point-A and point-B there are many uncertainties, and many complex and potentially lethal situations that require mature judgment and robust rapid-fire decision-making. The driver of a car must have the ability to focus attention, sense the environment, segment objects of interest, track moving objects, and avoid collisions. For example, cars on a typical two-lane country road routinely pass within two meters of each other traveling at high speeds in opposite directions. Every such encounter involves the potential for a fatal crash. Every curve and hill requires the ability to assess road conditions and judge the speed at which the vehicle can safely travel. One of the reasons many inexperienced drivers are involved in traffic accidents is that they have not yet fully mastered these cognitive skills. Among the reasons that even experienced drivers have accidents is that their attention wanders for only a moment.

Still others contend that there is no cognitive component because "Driving is something I can do without thinking". However, this fact in itself opens a window onto one of the great mysteries of cognition. That a complex task such as driving a car appears easy to an experienced driver, and can be accomplished without significant cognitive workload, is among the most important features of the human brain. Once a skill is thoroughly mastered, it requires minimal attention, and the conscious mind is free to focus on other things. The brain pushes well-learned cognitive functions down to computational modules that reside below the level of conscious attention. This gives the impression that the computational load has disappeared. But this is an illusion. The computational load has simply been shifted downward in the control hierarchy to echelons that that are below the level of conscious attention. All of the computations must continue to be done. They are simply done at control echelons that do not burden the higher centers of consciousness. Understanding how the brain accomplishes this push-down feat is one of the most important research problems in cognitive science. Solving this problem will enable new theories of learning, knowledge representation, decision-making, planning, and control.

Current research at NIST is focused on the following three aspects of autonomous vehicle research:

1) Autonomous driving on normal roads and streets, e.g., driving on country roads and city streets with on-coming traffic, negotiating intersections with traffic signals and pedestrians, and maneuvering in and out of parking spaces,
2) Autonomous tactical behaviors for teams of real and virtual autonomous military ground and air vehicles cooperating in the performance of elements of a route reconnaissance mission, and
3) Performance measures, testing, and evaluation of sensors, algorithms, and performance of intelligent systems and subsystems in real and virtual environments.

Elsewhere in government and industry laboratories around the world, major research programs are focused on building machines that are capable of flying airplanes, driving ground vehicles, and navigating undersea vehicles. Well funded military programs are addressing not only the mechanical and low level control of mobility systems, but the higher level functions of mission planning and tactical behaviors that involve collaboration between multiple manned and unmanned systems engaged in deadly competition with intelligent and determined enemy forces. Both military and commercial efforts are addressing the problem of driving autonomous vehicles safely and efficiently in traffic on city streets and highways while observing rules of the road, obeying traffic signs and signals, and avoiding collisions with other vehicles, pedestrians, animals, and road debris.

Progress is rapid. Capabilities of autonomous vehicles are advancing dramatically. New sensing technologies, increasing computational power, better understanding of the fundamental processes, and system architecture of intelligent systems are developing at an astonishing rate of speed. Significant economic and military applications will emerge in the next 10 to 15 years. For the military, autonomous vehicles will save lives, provide force multiplication, reduce training costs, improve operational readiness, and enable performance that exceeds human limitations. In the civilian sector, intelligent vehicle technology will reduce traffic fatalities, reduce the cost of accidents, increase traffic throughput, and improve transportation services.

To date, experimental unmanned ground vehicles have demonstrated the ability to follow a series of widely spaced GPS waypoints to destinations many kilometers away, and to do so in a way that avoids obstacles and driving hazards, while satisfying constraints and minimizing cost and risk along the way.

Current autonomous capabilities for off-road driving include:

- Driving on trails through the woods
- Driving cross country through rolling hills and fields of tall grass and weeds
- Planning routes that uses military maps to minimize cost and risk, and satisfy spatial and temporal constraints

Current autonomous capabilities for on-road driving include:

- Driving on well-marked freeways at high speed
- Driving on road through urban areas with rubble and parked cars
- Driving on dirt roads through the desert following GPS waypoints
- Driver warning of lane departures and impending collisions
- Planning routes that uses commercial road network databases to plan routes to distant destinations

Current autonomous driving capabilities *do not* yet include:

- Driving safely at normal speeds in the presence of moving objects, such as cross-traffic, on-coming traffic, pedestrians, and animals
- Obeying rules of the road in complex situations
- Negotiating intersections with traffic signals and 4-way traffic
- Understanding and obeying hand signals from human police
- Understanding complex military situations and responding with appropriate tactical behaviors (e.g., run, hide, communicate, attack, or continue current task)
- Learning from experience or take advice from subject matter experts

These capabilities will require considerable further research and development. The ability to drive safely in the presence of moving objects will require advances in sensors, that can enable autonomous vehicles to see 100 m or more with sufficient angular resolution to recognize human forms, and sufficient range resolution and frame-rates to track high speed on-coming traffic on narrow, winding, and hilly roads. It will require the ability to recognize and reason about traffic situations that include intersections, traffic signs and signals, other vehicles, pedestrians, animals, other vehicles, construction barriers, and on-road debris. It will require the ability to focus attention on what is important, and to segment objects of interest from a background of clutter under weather conditions that include rain, snow, ice, and road spray; and under lighting conditions that include daylight and darkness, glare from on-coming headlights, a background filled with clutter such as lighted advertisements, and rising or setting sun. It will require advances in world modeling to enable simultaneous tracking and prediction of the probable future trajectories of 30 or more moving objects. And it will require situational awareness, cognitive understanding, intuitive reasoning, and robust decision-making in a complex environment that is filled with fast-moving, potentially deadly encounters that can occur unexpectedly at any time.

Both autonomous driving and tactical behaviors require an enormous amount of a priori knowledge and skill, combined with the ability to sense and react to unexpected events in the environment. They require the ability to perceive the current situation, to build and maintain an internal model of the external world, to reason about what is represented in the world model, to imagine and anticipate the future, to make decisions under uncertainty, to formulate plans, to react to unanticipated situations, and to learn from experience. Autonomous driving on normal roads requires the ability to read signs and symbols, to understand traffic signals, and to interpret gestures by traffic police. Autonomous tactical behavior of teams requires the ability to perceive and understand the tactical situation, to know the roles, responsibilities, and capabilities of other team members, and to decide what appropriate action to take under the circumstances. Understanding how the human brain performs these tasks will provide deep insight into the fundamental elements of cognition.

Autonomous driving and tactical behaviors represent a major scientific and engineering challenge. Yet, there appear to be no fundamental scientific barriers. As we noted at the beginning of this chapter, the basic processes of perception, knowledge representation, reasoning, decision-making, planning, and control are understood, at least in principle. Sensing and perception capabilities are developing rapidly. Computational power is available. And a solid understanding of architectures that enable robust system design, engineering, and testing have been achieved.

What remains is a monumental effort in terms of scientific and engineering manpower, supported by large amounts of funding over a period of decades. The level of effort may be comparable to that required for building the atom bomb, or sending men to the moon. But these are well within the capability of the nation and the world to afford. Given the potential benefits of both military and civilian application, it seems likely that the necessary resources will be forthcoming.

References

1. Moravec, H. "Towards Automatic Visual Obstacle Avoidance", Proceedings of the Fifth International Conference on Artificial Intelligence, Cambridge, MA, pp. 584, August, 1977.
2. Kurzweil, R., "The Singularity is Near: When Humans Transcend Biology", Viking Adult, September, 2005.

Intelligent Vehicle Systems: A 4D/RCS Approach
Glossary

1) An ***Architecture*** is the structure that identifies, defines, and organizes components, their relationships, and principles of design; the assignment of functions to subsystems and the specification of the interfaces between subsystems.

2) A ***Bayer*** filter is a color filtering array for arranging RGB colors on a monochromatic grid. The term most often refers to a common mosaic of color filters used on many single chip digital cameras. Each sensor pixel is covered by red, green and blue colored filters. The Bayer filter has twice as many green pixels as red or blue because of the human eye's greater resolving power with green light.

3) ***Behavior Generation (BG)*** is planning and control of actions designed to achieve behavioral goals.

4) ***Classification*** is the categorization of objects according to measured features or *a priori* knowledge.

5) ***Global Positioning System (GPS)*** is a satellite-based system that works with ground based receivers to provide time, velocity and position information at the receiver at various levels of service quality.

6) ***Ground Truth*** is a reference data set, describing locations or objects in the world, used for comparison with data collected from a system-under-test in order to establish performance attributes of the system.

7) A ***Histogram*** is the result of a data transformation in which each data element is added to a cell in an array based on one or more measured features. The cells in the array contain a count of the number of elements that match a particular feature set.

8) An ***Intelligent System*** is a system with the ability to act appropriately in an uncertain environment.

9) ***Intelligent Vehicle Systems*** typically consist of a variety of sensors, actuators, navigation and driving systems, communications systems, mission packages, and weapons systems controlled by an intelligent controller.

10) ***Knowledge Database (KD)*** is the data structures, the static and dynamic information that collectively form the world model.

11) ***LADAR*** is an acronym for LAser Detection And Ranging. A LADAR is an active optical sensing system that is used to obtain multiple distance measurements of a scene either in a scanning or scannerless mode. Individual distances are obtained by measuring the time-of-flight (TOF) of a laser pulse or from the phase difference of an Amplitude Modulated (AM) or Frequency Modulated (FM) Continuous Wave (CW) laser or other light source.

12) ***Learning*** is the process of acquiring knowledge or skill through study, experience or teaching. It is a process that depends on experience and leads to long-term changes in behavior potential. Behavior potential describes the possible behavior of an individual (not actual behavior) in a given situation in order to achieve a goal.

13) A ***Map*** is a simplified depiction of a space, a navigational aid which highlights relations between objects within that space. Most usually a map is a two-dimensional, geometrically accurate representation of a three-dimensional space. In this book, a map is usually represented by a two-dimensional array projected onto a surface in the world, with cells in the array containing information about the associated region of space in the world.

14) A ***Mission*** is the highest level task assigned to the system.

15) ***Mission Planning*** is the process to generate tactical goals, a route (general or specific), commanding structure, coordination, and timing for one or teams of unmanned systems. The mission plans can be generated either in advance or in real-time. The mission plans can be generated either by operators or by the onboard software systems in either centralized or distributed ways. The term ***dynamic mission planning*** refers to onboard, real-time mission planning.

16) ***Modes of Operation***
    - In the ***fully autonomous mode*** of operation, the unmanned system is expected to accomplish its mission, within a defined scope, without human intervention.
    - In the ***semi-autonomous mode*** of operation, the unmanned system and/or a human operator conduct a mission which requires various levels of human-robot interaction.
    - In the ***teleoperation mode*** of operation, the human operator, using video feedback and/or other sensory feedback, either directly controls the actuators or assigns incremental goals, waypoints in mobility situations, on a continuous basis, from off the vehicle and via a tethered or radio linked control device. In this mode, the unmanned system may take limited initiative in reaching the assigned incremental goals.
    - In the ***remote control mode*** of operation, the human operator, without benefit of video or other sensory feedback, directly controls the actuators of the unmanned system on a continuous basis, from a location off the vehicle and via a tethered or radio linked control device using visual line-of-sight cues. In this mode, the unmanned system takes no initiative and relies on continuous or nearly continuous input from the user.

17) A **Neural Network** or an Artificial Neural Network is an information processing paradigm loosely modeled on the way biological systems, such as the brain, process information.

18) ***Obstacle*** Any physical entity that opposes or deters passage or progress, or impedes mobility in any other way. Any obstruction designed or employed to disrupt, fix, turn, or block the movement of an opposing force, and to impose additional losses in personnel, time, and equipment on the opposing force. Obstacles can be natural, manmade, or a combination of both. They can be positive, negative (e.g., ditches), or groupings (e.g., areas with high security alert) and can be moving or stationary.

19) An ***Ontology*** is a controlled vocabulary that describes objects and the relations between them in a formal way and has a grammar to allow using these terms to express something meaningful within the specified domain of interest. The potential of an ontology is to be able to store the entire set of task knowledge on a computer in a computer interpretable format to support human and automated inquiries and builds of simulators and controllers.

20) ***Perception*** is the capability of unmanned system in sensing and building an internal model of the environment within which it is operating, and assigning entities, events, and situations perceived in the environment to classes. The classification (or recognition) process involves comparing what it observed with the system's *a priori* knowledge.

21) ***Performance Evaluation*** is the process of data collection and analysis to determine quantitatively and unambiguously how well a system or component meets the operational, computational, or functional requirements.

22) A ***Pixel*** is a shortened form of "picture element". The smallest unit of a digital picture.

23) ***Planning*** is a process of generating and/or selecting a plan to accomplish a task.

24) ***RCS Methodology*** is a formalized set of analysis procedures used to identify the relevant knowledge about multi-resolutional task decisions, their conditional situations/world states, and their measurable real world entities in a manner so that this knowledge set can be represented in a format conducive to implementation in a hierarchical real-time control system.

25) **A *Reference Model Architecture*** is an architecture in which the entire collection of entities, relationships, and information units involved in interactions between and within subsystems and components are defined and modeled.

26) ***Registration*** is the process of transforming different sets of data, in different coordinate systems, acquired by sampling the same scene or object at different times, with different sensors, or from different perspectives, into one coordinate system. Registration is necessary in order to be able to compare or integrate the data obtained from different measurements.

27) A ***Saccade*** is a term used to describe rapid intermittent eye movement (changes in eye position) which occurs when the eyes fix on one point of interest after another in the visual field. This term can also be applied to describe the rapid pointing (usually with a pan-tilt platform) of an electro-optical visual perception system.

28) ***Sensor*** is taken to mean a device that responds to a stimulus, such as heat, light, or pressure, detects and/or generates a signal measures, and/or records physical phenomena, and indicates objects and activities by means of energy or particles emitted, reflected, or modified by the objects and activities.

29) ***Sensor fusion*** is the process of combining sensory data or data derived from sensory data from disparate sources such that the resulting information is in some sense better than would be possible when these sources were used individually. The term better in that case can mean more accurate, more complete, or more dependable, or refer to the result of an emerging view, such as stereoscopic vision (calculation of depth information by combining two-dimensional images from two cameras at slightly different viewpoints).

30) ***Sensory Processing (SP)*** is a set of processes that operate on sensor signals to detect, measure, and classify entities and events and derive useful information about the world.

31) A ***Situation*** is the identified pattern of entities/world states that the World Modeling has computed that matches or is a precursor to one of the relevant input conditions of a rule in a state table. It usually implies a value judgment operation in the context of the task to determine its existence, for e.g., the situation of Conditions_Good_To_Pass is an evaluated situation that is relevant to the task to Pass_Vehicle_In_Front and it is dependent on a great number of entities and world states.

32) A ***State Table*** is a structured mechanism used within the Behavior Generation component of an RCS node that groups all of the procedural rules relevant to a particular input command for this node. These rules are ordered in their execution by the addition information of states which encode the execution sequence history.

33) A ***Task*** is named activity performed to achieve or maintain a goal. Mission plans are typically represented with tasks. Task performance may, further, result in subtasking. Tasks may be assigned to operational units via task commands.

34) ***Task Decomposition*** is a process by which a task given to a BG process at one level is decomposed into a set of sequences of subtasks to be given to a set of subordinate BG processes at the next lower level.

35) ***Technology Readiness Level (TRL)*** is a measure used to assess the maturity of evolving technologies prior to incorporating that technology into a system or subsystem.

36) ***Terrain*** refers to the physical features of the ground surface, including the subsurface. These physical features include both natural (e.g., hills) and manmade (e.g., pipelines) features. Major terrain types are delineated based upon local relief, or changes in elevation, and include: flat to rolling, hilly and mountainous. Other important characteristics used to describe the terrain include: hydrologic features (e.g., swamps), vegetation characteristics (e.g., forests) and cultural features (e.g., cities).

37) An ***Unmanned System*** is an electromechanical system, with no human operator aboard, and is able to exert its power to perform designed missions. May be mobile or stationary. Includes categories of Unmanned Ground Vehicles (UGV), unmanned aerial Vehicles (UAV), Unmanned Underwater Vehicles (UUV), Unmanned Surface Vehicles (USV), and Unattended Ground Sensors (UGS).

38) ***Urban Search and Rescue (US&R, USAR)*** (A) The term used to define the strategy, tactics, and operations for locating, providing medical treatment, and extrication of entrapped victims. (B) A task force equipped with necessary tools and equipment and the required skills and techniques for the search, rescue, and medical care of victims of structural collapse.

39) ***Value Judgment (VJ)*** is a process that computes value, determines importance, assesses reliability, and generates reward and punishment.

40) A ***World Model*** is an internal representation of the world.

41) ***World Modeling (WM)*** is a set of processes that construct and maintain a world model.