Integrating Disparate Knowledge Representations Within 4D/RCS

J. Albus, C. Schlenoff, R. Madhavan, S. Balakirsky, and T. Barbera

National Institute of Standards and Technology (NIST
100 Bureau Drive, Stop 8230
Gaithersburg, MD 20899
{james.albus, craig.schlenoff, rajmohan.madhavan, stephen.balakirsky, tony.barbera} @nist.gov

Abstract

In this paper, we show how the 4D/RCS architecture incorporates and integrates multiple types of disparate knowledge representation techniques into a common, unifying framework. 4D/RCS is based on the supposition that 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. 4D/RCS allow for the capability to capture knowledge in formalisms and at levels of abstraction that are suitable for the way that it is expected to be used.

In the context of applying the architecture to the control of autonomous vehicles, we describe the procedural and declarative types of knowledge that has been developed and applied, and value that each brings to the achieving the ultimate goal of autonomous navigation. We also look at symbolic vs. iconic knowledge representation, and show how 4D/RCS accommodates both of these types of representations and uses the strengths of each to strive towards achieving human-level intelligence in autonomous systems.

Introduction

During the past century, the neurosciences have provided deep insights into the anatomical, physiological, chemical, and computational basis of cognition. Neuroanatomy has described the structure and function of the basic computational element of the brain – the neuron, and produced extensive maps of the computational modules and interconnecting data flow pathways making up the anatomy of the brain. Behavioral psychology provides information about stimulus-response behavior and instrumental conditioning. Cognitive psychology is exploring how the brain represents knowledge; how it perceives objects, events, situations, and relationships; how it analyzes the past and plans for the future; and how it selects and controls behavior that satisfies desires and achieves goals

Over the last five decades, the invention of the electronic computer has brought rapid advances in computational power, making it feasible to launch serious attempts at building intelligent systems. Artificial intelligence and robotics have produced significant results in planning, problem-solving, rule-based reasoning, image analysis, and speech understanding. Autonomous vehicle research has produced advances in real-time sensory processing, world modeling, navigation, path planning, and obstacle avoidance. Research in industrial automation and process control has produced hierarchical control systems, distributed databases, and models for representing processes and products. Modern control theory has developed precise understanding of stability, adaptability, and controllability under various conditions of uncertainty and noise. Progress is rapid in each of the above fields, and there exists an enormous and rapidly growing body of literature in all of these areas.

What is lacking is a widely accepted theoretical framework that can integrate concepts from all of these different fields into a unified whole. Several frameworks have been proposed. One of the earliest was the ACT architecture [6]. 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 [14]. 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 [16,20]. 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

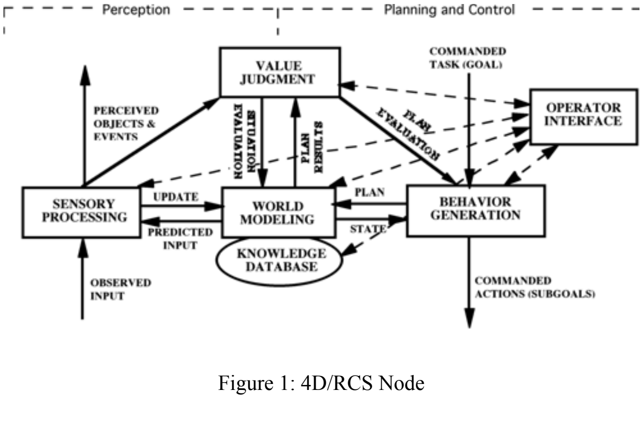


Figure 1: 4D/RCS Node

linguistics, theorem proving, and cognitive modeling.

Other cognitive architectures include Prodigy, ICARUS, IMPRINT (Improved Performance Research Integration Tool), EPIC (Executive-Process Interactive Control), and 4D/RCS (Real-time Control Systems). Like Soar, Prodigy uses search through a problem space to achieve goals cast as first-order expressions [19]. ICARUS is a reactive architecture that encodes knowledge as reactive skills [24]. IMPRINT is a task description language designed for the Army to capture the procedural specification of tactical behavior scenarios [8]. It contains a dynamic, stochastic, discrete-event network modelling 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 [7]. EPIC 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 [15]. 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 [5].

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.

Background of 4D/RCS

4D/RCS evolved from the bottom up as a real-time intelligent control system for real machines operating on real objects in the real world. The first version of RCS was developed as a sensory-interactive goal-directed controller for a laboratory robot [10]. The fundamental element is the control loop with a goal, a transition function, a feedback loop, and an action output such as a force, velocity, or position. Over the years, RCS has evolved into an intelligent controller for industrial robots, machine tools, intelligent manufacturing systems, automated general mail facilities, automated stamp distribution systems, automated mining equipment, unmanned underwater vehicles, and unmanned ground vehicles [2,11]. The most recent version of RCS (4D/RCS) embeds elements of Dickmanns [13] 4-D approach to machine vision within the 4D/RCS control architecture. 4D/RCS was designed for the U.S. Army Research Lab AUTONAV and Demo III Experimental Unmanned Vehicle programs and has been adopted by the Army Future Combat System program for Autonomous Navigation Systems [3,4].

4D/RCS 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), as shown in Figure 1. Throughout the hierarchy, interaction between SP, WM, VJ, BG, and KD give rise to perception, cognition, and reasoning. At low levels, representations of space and time are short-range and high-resolution. At high levels, distance and time are long-range and low-resolution. This enables high-precision fast-action response at low levels, while long-range plans and abstract concepts are being simultaneously formulated at high levels. The hierarchical approach also helps to manage computational complexity.

4D/RCS closes feedback loops at every level, through every node. SP processes focus attention (i.e., window regions of space or time), group (i.e., segment regions into entities), compute entity attributes, estimate entity state, and assign entities to classes at every level. WM processes maintain a rich and dynamic database of knowledge about the world in the form of images, maps, entities, events, and relationships at every level. Other WM processes use that knowledge to generate estimates and predictions that support perception, reasoning, and planning at every level. VJ processes 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.

Enabling Intelligence in Autonomous Vehicles

The Many Dimensions of Knowledge in 4D/RCS

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. This section will 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.

As mentioned previously, 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.

This knowledge is stored within the Knowledge Database (KD). The KD consists of data structures that contain the static and dynamic information that collectively form a model of the world. The KD contains the information needed by the world model to support the behavior generation, sensory processing, and value judgment processes within each node. Knowledge in the KD 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.

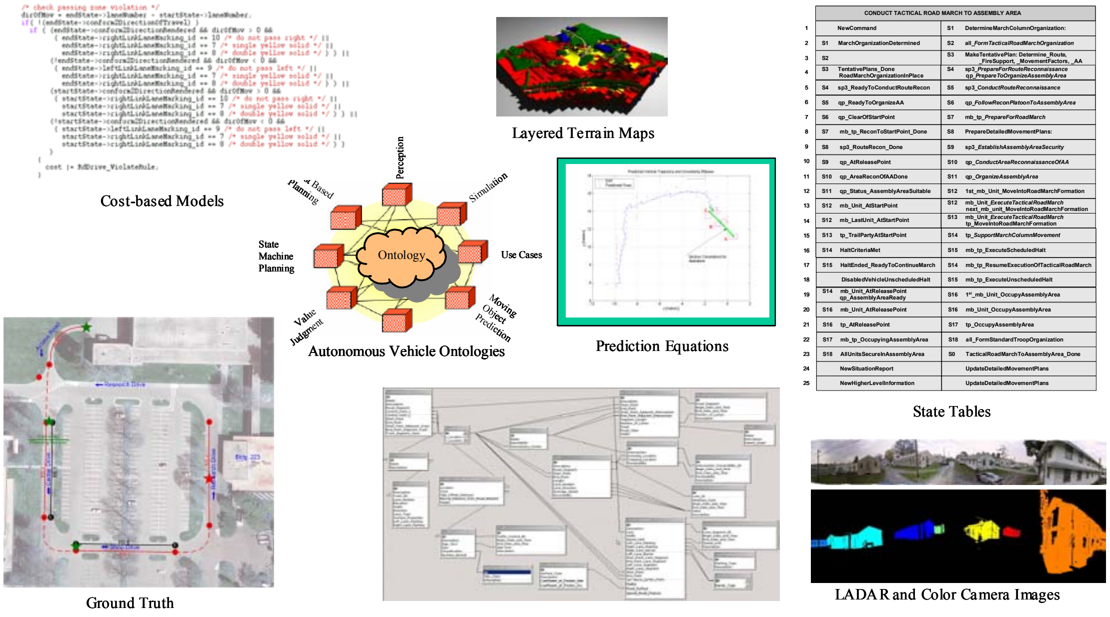
Figure 2 shows the many different types of knowledge representation formalisms that are currently being implemented within the 4D/RCS architecture as applied to autonomous driving. These formalisms range from iconic to symbolic and from procedural to declarative. Knowledge is captured in formalisms and 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. In the following subsections, we will describe some of the formalisms depicted, classifying knowledge as either procedural or declarative.

Procedural Knowledge

Procedural knowledge 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. Within 4D/RCS, procedural knowledge is primarily used for planning and control purposes. As such, we will describe two planning approaches that are currently being implemented in 4D/RCS and describe the knowledge that underlies each.

Both planning approaches start with the same 4D/RCS methodology for determining the knowledge which needs to be represented to accomplish the planning task. The methodology starts as follows:

1. The first step involves an intensive analysis of domain knowledge from manuals and subject matter experts, especially using scenarios of particular subtask operations. The output of the



```
/* check passing zone violation */
dirOfMov = endState->laneNumber - startState->laneNumber;
if( !(endState->conform2DirectionOfTravel) )
  if ( (endState->conform2DirectionRendered && dirOfMov > 0 &&
        ( endState->rightLinkLaneMarking_id == 10 /* do not pass right */ ||
          endState->rightLinkLaneMarking_id == 7 /* single yellow solid */ ||
          endState->rightLinkLaneMarking_id == 8 /* double yellow solid */ ) ) ||
       (!endState->conform2DirectionRendered && dirOfMov < 0 &&
        ( endState->leftLinkLaneMarking_id == 9 /* do not pass left */ ||
          endState->rightLinkLaneMarking_id == 7 /* single yellow solid */ ||
          endState->rightLinkLaneMarking_id == 8 /* double yellow solid */ ) ) ||
       (startState->conform2DirectionRendered && dirOfMov > 0 &&
        ( startState->rightLinkLaneMarking_id == 10 /* do not pass right */ ||
          startState->rightLinkLaneMarking_id == 7 /* single yellow solid */ ||
          startState->rightLinkLaneMarking_id == 8 /* double yellow solid */ ) ) ||
       (!startState->conform2DirectionRendered && dirOfMov < 0 &&
        ( startState->leftLinkLaneMarking_id == 9 /* do not pass left */ ||
          startState->rightLinkLaneMarking_id == 7 /* single yellow solid */ ||
          startState->rightLinkLaneMarking_id == 8 /* double yellow solid */ ) )
     )
  {
    cost |= RdDrive_ViolateRule;
  }
```

Cost-based Models

Layered Terrain Maps

Autonomous Vehicle Ontologies

Prediction Equations

	CONDUCT TACTICAL ROAD MARCH TO ASSEMBLY AREA			
1		NewCommand	S1	DetermineMarchColumnOrganization:
2	S1	MarchOrganizationDetermined	S2	all_*FormTacticalRoadMarchOrganization*
3	S2		S3	MakeTentativePlan: Determine_Route, _FireSupport, _MovementFactors, _AA
4	S3	TentativePlans_Done RoadMarchOrganizationInPlace	S4	sp3_*PrepareForRouteReconnaissance* qp_*PrepareToOrganizeAssemblyArea*
5	S4	sp3_ReadyToConductRouteRecon	S5	sp3_*ConductRouteReconnaissance*
6	S5	qp_ReadyToOrganizeAA	S6	qp_*FollowReconPlatoonToAssemblyArea*
7	S6	qp_ClearOfStartPoint	S7	mb_tp_*PrepareForRoadMarch*
8	S7	mb_tp_ReconToStartPoint_Done	S8	PrepareDetailedMovementPlans:
9	S8	sp3_RouteRecon_Done	S9	sp3_*EstablishAssemblyAreaSecurity*
10	S9	qp_AtReleasePoint	S10	qp_*ConductAreaReconnaissanceOfAA*
11	S10	qp_AreaReconOfAADone	S11	qp_*OrganizeAssemblyArea*
12	S11	qp_Status_AssemblyAreaSuitable	S12	1st_mb_Unit_MoveIntoRoadMarchFormation
13	S12	mb_Unit_AtStartPoint	S12	mb_Unit_*ExecuteTacticalRoadMarch* next_mb_unit_MoveIntoRoadMarchFormation
14	S12	mb_LastUnit_AtStartPoint	S13	mb_Unit_*ExecuteTacticalRoadMarch* tp_MoveIntoRoadMarchFormation
15	S13	tp_TrailPartyAtStartPoint	S14	tp_*SupportMarchColumnMovement*
16	S14	HaltCriteriaMet	S15	mb_tp_ExecuteScheduledHalt
17	S15	HaltEnded_ReadyToContinueMarch	S14	mb_tp_ResumeExecutionOfTacticalRoadMarch
18		DisabledVehicleUnscheduledHalt	S15	mb_tp_ExecuteUnscheduledHalt
19	S14	mb_Unit_AtReleasePoint qp_AssemblyAreaReady	S16	1st_mb_Unit_OccupyAssemblyArea
20	S16	mb_Unit_AtReleasePoint	S16	mb_Unit_OccupyAssemblyArea
21	S16	tp_AtReleasePoint	S17	tp_OccupyAssemblyArea
22	S17	mb_tp_OccupyingAssemblyArea	S18	all_FormStandardTroopOrganization
23	S18	AllUnitsSecureInAssemblyArea	S0	TacticalRoadMarchToAssemblyArea_Done
24		NewSituationReport		UpdateDetailedMovementPlans
25		NewHigherLevelInformation		UpdateDetailedMovementPlans

State Tables

Ground Truth

Databases

LADA R and Color Camera Images

Figure 2: Knowledge Representations in 4D/RCS

effort is a structuring of this knowledge into a task decision tree consisting of simpler and simpler commands (actions/verbs) at simpler and simpler levels of task description.

2. The second step defines the hierarchical organization of agent control modules that will execute these layers of commands in such a manner as to reasonably accomplish the tasks. 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 and responsibilities, as well as defining the execution structure.

At this point, the two approaches diverge in the procedure for determining the types of knowledge necessary to accomplish the planning task. Subsequent steps are described in the following subsections.

State Machine-Based Planning (a)

The state machine-based methodology concentrates on the task decomposition as the primary means of understanding the knowledge required for intelligent control. Once the previous two steps are performed, the procedure proceeds as follows:

3a. The third step clarifies the processing of each agent's input command through the use of rules to identify all of the task branching conditions with their corresponding output commands. 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 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 step.

4a. In the fourth step, the above named situations that are the task branching conditions are defined in great detail in terms of their dependencies on world and task states. This step attempts to define the detailed precursor states of the world that cause a particular situation to be true.

5a. In the fifth step, we identify and name all of the objects and entities together with their particular features and attributes that are relevant to defining the above world states and situations. Current efforts are exploring the use of ontologies and databases to represent this information.

6a. The sixth step uses the context of the particular task activities to establish the 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.

As described in the previous four steps, this approach relies on various knowledge representation formalisms, such as state tables, production rules, situation representation, ontologies, databases, and sensor images.

Cost-Based Planning Representations (b)

The cost based methodology concentrates on decomposing each of its assigned tasks into an optimal sequence of commands that will be assigned to its subordinates. This is accomplished through the incremental creation and evaluation of a planning graph [9]. Once again, the first two steps from section 3.2 must be performed and are then followed by:

3b. The third step develops an action model that delineates how each of the subordinate's commands will affect the system state at the current level of resolution. This allows a simulation system to experiment with various command options in order to obtain the state transitions that are required to fulfill the level's goals.

4b. The fourth step develops a set of user constraints and objectives that could affect the cost/benefit ratio of performing a given action or occupying a given state. For example, the cost/benefit of running a red light would be substantially different for a casual driver than it would be for a person driving their wife to the hospital to deliver a baby.

5b. Step 5 examines the potential state variable transitions that have been identified along with the potential user constraints and objective in order to construct a cost function that will be utilized by the value judgment module during the graph expansion process.

By developing the state transition simulator from step 3 we are able to incrementally build a planning graph based on potential actions that a subordinate may take. The cost function developed in step 5 may then be used to evaluate the individual arcs of the planning graph in order to control the expansion order and find the cost optimal path through the planning graph.

Similar to the state-based planning approach, this approach relies on various knowledge representation formalisms, such as state tables, cost-based models, situation representation, ontologies, databases, and sensor images.

Declarative Knowledge

Declarative knowledge is represented in a format that may be manipulated, decomposed, and analyzed by reasoners. Unlike procedural knowledge, it does not describe how to perform a given task. Instead, it provides the ability to use knowledge in ways that the system designer did not foresee. Two classes of declarative knowledge that is captured within 4D/RCS are symbolic knowledge and iconic knowledge. In the follow two subsections, we describe details about these two types of knowledge representations.

Symbolic Knowledge

Symbolic representations use symbols to represent events or objects in the world. Two types of symbolic representations that are being implemented within 4D/RCS are ontologies and relational databases.

Ontologies represent key concepts, their properties, their relationships, and their rules and constraints within a given domain. Ontologies often focus more on the meaning of concepts than on the terms that are used to represent them. Two efforts have focused on the development of ontologies for autonomous navigation.

The first is an ontology for roadway driving, which is used to determine if objects in the environment are potential obstacles to our vehicle. The system is composed of an ontology of objects representing “things” that may be encountered in our current environment, in conjunction with rules for estimating the damage that would be incurred by collisions with the different objects in different situations. Automated reasoning is used to estimate collision damage, and this information is fed to the route planner to help it decide whether to avoid the object. More information about this effort can be found at [21].

The second is an ontology to model tactical behaviors, which is based upon the OWL-S specification (Web Ontology Language- Services) [25]. In this context, behaviors are actions that an autonomous vehicle is expected to perform when confronted with a predefined situation. The ontology is stored within the 4D/RCS knowledge database, and the behaviors will be spawned when situations in the world are determined to be true, as judged by sensor information and the value judgment components. More information about this effort can be found in [23].

In addition to ontologies, databases have been developed to house symbolic information. The database that has received the most attention to date is the Road Network Database. [22] The database includes detailed information about the roadway, such as where the road lies, rules dictating the traversal of intersections, lane markings, road barriers, road surface characteristics, etc. The purpose of the Road Network Database is to provide the data structures necessary to capture all of the 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 provides 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 provides 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.

Iconic Knowledge

Iconic knowledge is often spatial in nature and can be defined as 2D or 3D array data in which the dimensions of the array correspond to dimensions in physical space. The value of each element of the array may be Boolean data, real number, or vector data representing a physical property such as light intensity, color, altitude, range, or density. Each element may also contain spatial or temporal gradients of intensity, color, range, or rate of motion. Each element may also contain a pointer to a geometric entity (such as an edge, vertex, surface, or object) to which the pixel belongs.

Examples of iconic knowledge used within 4D/RCS include digital terrain maps, sensor images, models of the kinematics of the machines being controlled, and knowledge of the spatial geometry of parts or other objects that are sensed and with which the machine interacts in some way. This is where objects and their relationship in space and time are modeled in such a way as to represent and preserve those spatial and temporal relationships, as in a map, image, or trajectory.

Within 4D/RCS, maps enhance the scope of the world model. Such iconic maps may take a variety of forms including survey and aerial maps and may provide significant information about existing topology and structures. The higher levels in the 4D/RCS control hierarchy includes feature and elevation data from *a priori* digital terrain maps such as information about roads, bridges, streams, woods, and buildings. This information needs to be registered and merged with data from the lower level maps that are generated by sensors. 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 autonomous driving. Another potential application for registering sensor data is the computation of ground truth.

Towards registering LADAR (Laser Range Detection) range images to *a priori* maps, we have developed an iterative algorithm that can deal with false/spurious matches, occlusions and outliers for UGV (unmanned ground vehicle) navigation [18]. The iterative registration algorithm can be summarized as follows: Given an initial motion transformation between two 3D point sets, a set of correspondences are developed between data points in one set and the next. For each point in the first data set, we find the point in the second that is closest to it under the current transformation. It should be noted that correspondence between the two points sets is initially unknown and that point correspondences provided by sets of closest points is a reasonable approximation to the true point correspondence. From the set of correspondences, an incremental motion can be computed facilitating further alignment of the data points in one set to the other. This correspondence/compute motion process is iterated until a predetermined threshold termination condition.

A hybrid iterative algorithm has also been developed for registering 3D LADAR range images obtained from unmanned aerial and ground vehicles [17]. Combined with a feature-based approach, the algorithm was shown to produce accurate registration for the two sets of LADAR data. Registration of the UGV LADAR to the aerial survey map minimizes the dependency on GPS for position estimation especially when the GPS estimates are unreliable or unavailable.

Results

Experimental validation of the 4D/RCS architecture and the knowledge representation within has been provided by the performance of the Demo III eXperimental Unmanned ground Vehicles (XUVs) in an extended series of demonstrations and field tests during the winter of 2002-2003.

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 the vehicle planned to drive. The LADAR detected the ground out to a range of about 20 m, and detected 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 over 50 m apart.

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 in 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 were driven 550 km, over rough terrain: 1) in the desert; 2) in the woods, through rolling fields of weeds and tall grass, and on dirt roads and trails; and 3) through an urban environment with narrow streets cluttered with parked cars, dumpsters, culverts, telephone poles, and manikins. Tests were conducted under various conditions including night, day, clear weather, rain, and falling snow. The unmanned vehicles operated over 90 % of both time and distance without any operator assistance. An extensive report of these experiments has been published [12], along with high resolution ground truth data describing the terrain where the XUVs experienced difficulties [26].

It should be noted that the Demo III 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 world model, and the development of perception, world modeling, and planning algorithms for operating in the presence of moving objects is a topic of current research.

Conclusion

We believe that the 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 we can achieve. In this paper, we have 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. We also show how all of these knowledge representation formalisms not only fit into the node structure present at each level of the 4D/RCS hierarchy, but also play in the role in the 4D/RCS methodologies.

We also believe that autonomous driving is an excellent topic for continued research on intelligent systems for the following reasons:

- It is a problem domain for which there is a large potential user base, both in the military and civilian sectors. This translates into research funding.

- It is a problem domain where physical actuators and power systems are readily available. Wheeled and tracked vehicle technology is mature, inexpensive, and widely deployed.
- It is a problem domain for which the technology is ready. The invention of real-time LADAR imaging makes it possible to capture the 3-D geometry and dynamics of the world. This has broken the perception barrier. The continued exponential growth rate in computing power per dollar cost has brought the necessary computational power within the realm of economic viability. This has broken the cost barrier. Intelligent control theory has advanced to the point where the engineering of intelligent systems is feasible. This has broken the technology barrier.
- It 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 in rewarding successful locomotion behavior. It is therefore, not unreasonable to suspect that building truly 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.

References

1. Albus, J., "A Theory of Cerebellar Function," *Mathematical Biosciences*, Vol. 10, 1971, pp. 25-61.
2. Albus, J., "The NIST Real-time Control System (RCS): An approach to Intelligent Systems Research," *Journal of Experimental and Theoretical Artificial Intelligence*, Vol. 9, 1997, pp. 157-174.
3. Albus, J. and et.al., "4D/RCS Version 2.0: A Reference Model Architecture for Unmanned Vehicle Systems," NISTIR 6910, National Institute of Standards and Technology, Gaithersburg, MD, 2002.
4. Albus, J. and Meystel, A., *Engineering of Mind*, John Wiley & Sons, Inc. 2001.
5. Albus, J. S., *Brain, Behavior, and Robotics*, McGraw-Hill 1981.
6. Anderson, J., *The Architecture of Cognition*, Lawrence Erlbaum Associates, Mahwah, N.J., 1983.
7. 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.
8. Archer, S. and Adkins, R., "IMPRINT User's Guide," 1999.
9. Balakirsky, S., *A Framework for Planning with Incrementally Created Graphs in Attributed Problem Spaces*, IOS Press, Berlin, Germany, 2003.
10. Barbera, T., Albus, J., and Fitzgerald, M., "Hierarchical Control of Robots Using Microcomputers," *Proceedings of the 9th International Symposium on Industrial Robots*, Washington DC, 1979.
11. Barbera, T., Fitzgerald, M., Albus, J., and Haynes, L. S., "RCS: The NBS Real-Time Control System," *Proceedings of the Robots Conference and Exposition*, Detroit, Michigan, 1984.
12. 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.
13. Dickmanns, E. D., "An Expectation-Based Multi-Focal Saccadic (EMS) Vision System for Vehicle Guidance," *Proceedings of the 9th International Symposium on Robotics Research (ISRR'99)*, Salt Lake City, 1999.
14. 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.
15. 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.
16. Laird, J. E., Newell, A., and Rosenbloom, P. S., "SOAR: An Architecture for General Intelligence," *Artificial Intelligence*, Vol. 33, 1987, pp. 1-64.
17. Madhavan, R., Hong, T., and Messina, E., "Temporal Range Registration for Unmanned Ground and Aerial Vehicles," *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, New Orleans, LA, USA, 2004, pp. 3180-3187.
18. 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.
19. Minton, S. N., "Quantitative results concerning the utility of explanation-based learning," *Artificial Intelligence*, Vol. 42, 1990, pp. 363-391.
20. Newell, A. and Simon, H., *Human Problem Solving*, Prentice-Hill, Englewood Cliffs, 1972.
21. Provine, R., Uschold, M., Smith, S., Balakirsky, S., and Schlenoff, C., "Observations on the Use of Ontologies for Autonomous Vehicle Navigation Planning," *To Appear in the Robotics and Autonomous Systems Journal: Special Issue on the 2004 AAAI Knowledge Representation and*

Ontologies for Autonomous Systems Spring Symposium, 2004.

22. Schlenoff, C., Balakirsky, S., Barbera, T., Scrapper, C., Ajot, J., Hui, E., and Paredes, M., "The NIST Road Network Database: Version 1.0," National Institute of Standards and Technology (NIST), 2004.
23. Schlenoff, C., Washington, R., and Barbera, T., "Experiences in Developing an Intelligent Ground Vehicle (IGV) Ontology in Protege," *Submitted to the 7th International Protege Conference*, Bethesda, MD, 2004.
24. 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.
25. The OWL Services Coalition, "OWL-S 1.0 Release," http://www.daml.org/services/owl-s/1.0/owl-s.pdf, 2003.
26. Witzgall, C., Cheok, G., and Gilsinn, D., "Terrain Characterization from Ground-Based LADAR," *Proceedings of the PerMIS '03 Workshop*, National Institute of Standards and Technology, Gaithersburg, MD 20899, 2003.