Proceedings of the American Association for Artificial Intelligent (AAAI), Fall Symposium Series, 1996, Massachusetts Institute of Technology, Cambridge, MA, November 9-11, 1996.

An Engineering Architecture for Intelligent Systems

James S. Albus

Intelligent Systems Division
National Institute of Standards and Technology
Building 220, Room B-124
Gaithersburg, MD 20899
albus@cme.nist.gov

Abstract

The Real-time Control System (RCS) is a reference model architecture for design and engineering of intelligent systems. It is intended to provide a theoretical framework for the development of standards and performance measures for intelligent systems, as well as engineering guidelines for the design and implementation of intelligent control systems for a wide variety of applications. RCS consists of a hierarchically layered set of intelligent processing nodes organized as a nested series of control loops. In each node, tasks are decomposed, plans are generated, world models are maintained, feedback from sensors is processed, and control loops are closed. In each layer, nodes have a characteristic span of control, with a characteristic planning horizon, and corresponding knowledge of detail in space and time. Nodes at the higher levels deal with high level management and planning, while nodes at lower levels deal with machine coordination and process control. RCS integrates and distributes deliberative planning and reactive control functions throughout the entire hierarchical architecture, at all levels, with all spatial and temporal scales.

Introduction

The Real-time Control System (RCS) is a reference model architecture for design and engineering of intelligent systems. It is intended to provide a theoretical framework for the development of standards and performance measures for intelligent systems, as well as engineering guidelines for the design and implementation of intelligent control systems for a wide variety of applications. RCS is based on the following axioms and definitions:

Axiom 1: The functional elements of an intelligent system are behavior generation, sensory perception, value judgment, and world modeling.

Df: **behavior generation**

behavior generation is the planning and control of action designed to achieve behavioral goals.

Behavior generation accepts task commands with goals and priorities, formulates and/or selects plans, and controls action. Behavior generation develops or selects plans by using a priori task knowledge and value judgment functions combined with real-time information provided by world modeling to find the best assignment of tools and resources to agents, and to find the best schedule of actions (i.e., the most efficient plan to get from an anticipated starting state to a goal state.) Behavior generation controls action by comparing current state feedback with the desired state specified by the plan and using a control law to compute the best action to null the difference. Behavior generation may also learn system models and optimize control laws.

Df: **sensory perception**

sensory perception is the transformation of data from sensors into internally meaningful and useful representations of the world.

Sensory perception accepts input data from sensors that measure states of the external world and states of the system itself. Sensory perception scales and filters data, computes observed features and attributes, and compares them with predictions from internal models. Correlations between sensed observations and internally generated expectations are used to detect events and recognize entities and situations. Variances between sensed observations and internally generated expectations are used to update internal models. Sensory perception also clusters, or groups, recognized entities and detected events into higher order

entities and events, and computes attributes of the higher order entities and events.

Df: **value judgment**

value judgment is the computation of cost, risk, and benefit of actions and plans; the estimation of the importance and value of objects, events, and situations; the assessment of reliability of information; and the calculation of rewarding or punishing effects of perceived states and events.

Value judgment evaluates perceived and planned situations thereby enabling behavior generation to select goals and set priorities. It computes what is important (for attention), and what is rewarding or punishing (for learning).

Df: **world modeling**

world modeling is the construction, maintenance, and use of internal representations of the world

World modeling maintains (short-term) dynamic and (long-term) static memory, including semantic and pragmatic relationships between entities, and links between symbolic and iconic representations. It answers queries from behavior generation regarding the state of the world, simulates results of possible future plans, and predicts sensory observations based on knowledge in the knowledge database. Interactions between sensory perception and world modeling compute transformations between descriptive and graphic representations and maintain the pointers that link signals to symbolic entities, and vice versa. This enables symbolic entity attributes to be updated from attributes in observed signals. It also allows predicted signals to be generated from symbolic entity frames. Predicted signals can be used by sensory perception to configure filters, masks, and windows for correlation and model matching, and for establishing correspondence between signal features. World modeling also maintains pointers that establish relationships (such as "is-part-of," "has-parts," "is-inside-of," "is-on-top-of," "is-caused-by," "is-used-for") between symbolic entities.

Axiom 2: World modeling maintains and uses a distributed dynamic store of knowledge, or knowledge database, that includes both a model of the environment and a model of the system itself.

Df: **knowledge database**

the data structures and the static and dynamic information that collectively form the world model.

The knowledge database stores information about the world in the form of state variables, symbolic entities, symbolic events, rules and equations, structural and dynamic models, task knowledge, signals, images, and maps.

Df: **state variable**

a numeric or symbolic variable that represent the current estimated value of a property or attribute of an object or event in world.

Df: **symbolic entity**

a data structure that represents a feature, object, or set that exists in the world, or in the world model. A symbolic entity can be a formal list, or frame, consisting of a list head with a name as an address, plus a set of attribute-value pairs, a set of recognition criteria, and a set of pointers that define relationships with other symbolic entities or events. These relationships can represent semantic or pragmatic meaning.

Df: **symbolic event**

a data structure that represents a state change, or a set of states or situations that occur at a particular time and place, or of a sequence of states, or situations, that transpire over an interval of time and space in the world. A symbolic event can be represented by a list, or frame, consisting of a list head, a set of attribute-value pairs, a set of detection criteria, and a set of pointers that define relationships with other symbolic events and entities.

Df: **rules and equations**

symbolic representations that express physical and mathematical laws that describe the way the world works and how things relate to each other in time, space, causality, and probability. Examples include If/Then rules, formulae in predicate calculus, differential equations, control laws, geometrical theorems, and system models.

Df: **structural models**

rules and equations that describe how physical structures are kinematically connected and how forces and stresses are distributed.

Df: **dynamic models**

rules and equations that describe how energy, force, and inertia interact with each other in time and space.

Df: **task knowledge**

knowledge of how to act in order to accomplish goals. Task knowledge includes skills and abilities required for manipulation, locomotion, communication, and attention. Task knowledge may also include a list of tools, materials, and information required to perform a task, as well as conditions required to begin or continue a task. For example, task knowledge may describe how to drill a hole, weld a seam, repair a TV, assemble an automobile, design

an engine, or plan a task. Task knowledge is primarily used by the behavior generation element.

Df: **signal**

output of a sensor that measures a phenomenon in the environment

Df: **image**

a two-dimensional array of attribute values. An image may consist of an array of brightness values from a CCD TV camera, or an array of attributes such as spatial or temporal gradients, stereo disparity, range, or flow that are computed from other images over spatial windows and temporal intervals. An image may also be generated by internal mechanisms (such as a computer graphics engine) from information stored in symbolic entity frames.

Df: **pixel**

a picture element. A pixel is the smallest distinguishable region in an image. A pixel may have attributes, such as the output of a photoreceptor in a CCD camera that corresponds to brightness or color. Pixel attributes may also include spatial or temporal gradients of brightness, range, stereo disparity, image flow magnitude and direction integrated over the area of the pixel.

Df: **maps**

two-dimensional arrays of attributes that are registered with known locations in the world. Map pixel attributes may include icons or names of symbolic entities.

Axiom 3: The knowledge database has two parts:

1. A long-term memory containing symbolic representations of all the entities, events, and rules that are known to the intelligent system.

2. A short-term memory containing both symbolic and image representations of those entities that are the subject of current attention.

Axiom 4: The functional elements and knowledge database of an intelligent system can be represented in an architectural node by a set of modules interconnected by a communication system that transfers information between them.

Df: **node**

a part of a control system that processes sensory information, maintains a world model, computes values, and plans and executes tasks. A node contains a Behavior Generation (BG) module, a World Modeling (WM) module, a Sensory Perception (SP) module, a Value Judgment (VJ) module, and a Knowledge Database (KD).

Figure 1 illustrates the relationships in a single node of the RCS architecture. The interconnections between sensory perception, world modeling, and behavior generation close a reactive feedback control loop between the observed input and the commanded action. The interconnections between behavior generation, world modeling, and value judgment enable task decomposition, planning, and reasoning about future actions. The interconnections between sensory perception, world modeling, and value judgment enable knowledge acquisition, situation evaluation, and learning. All the modules in an RCS node have input and output connections to an Operator Interface.

Axiom 5. The complexity inherent in intelligent systems can be managed through hierarchical layering.

Intelligent systems are inherently complex. Hierarchical layering is a common method for organizing complex systems that has been used in many different types of organizations throughout history for effectiveness and efficiency of command and control. In a hierarchical control system, higher level nodes have broader scope and longer time horizons, with less concern for detail. Lower levels have narrower scope and shorter time horizons, with more focus on detail. For example, Behavior Generating modules in RCS nodes at the upper levels in the hierarchy make long-range plans consisting of major milestones. At lower levels, Behavior Generating modules successively refine the long-range plans into short term tasks with detailed activity goals. At lower levels, Sensory Processing modules process data over local neighborhoods and short time intervals. At higher levels, they integrate data over long time intervals and large spatial regions. At low levels, World Model knowledge is short-term and fine grained, while at the higher levels it is broad in scope and generalized. At every level, feedback loops are closed to provide reactive behavior, with high-bandwidth fast-response loops at lower levels, and slower more deliberative reactions at higher levels.

At each level, state variables, entities, events, and maps are maintained to the resolution in space and time that is appropriate to that level. As detail is geometrically increased at each successively lower level in the hierarchy, the range of computation is geometrically decreased. As temporal resolution is increased, the time horizon decreases. This produces a ratio of range to resolution that remains relatively constant throughout the hierarchy. As a result, at each level, behavior generating functions make plans of roughly the same number of steps. Sensory perception functions compute entities that contain roughly the same number of sub entities. At higher levels, plans, perceived entities, and world modeling simulations are more complex. But, there is more time available between

replanning intervals for planning to occur. Thus, hierarchical layering keeps the amount of computing resources needed in each node reasonably small and relatively constant.

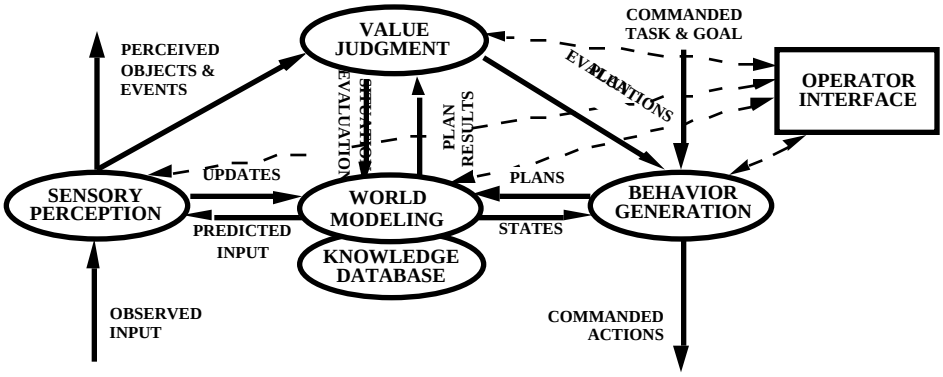

Figure 1. A node in the RCS reference model architecture. The functional elements of an intelligent system are behavior generation (planning and control), sensory perception (filtering, detection, recognition, and interpretation), world modeling (store and retrieve knowledge and predict future states), and value judgment (compute cost, benefit, importance, and uncertainty). These are supported by a knowledge database, and a system architecture that interconnects the functional modules and the knowledge database. This collection of modules and their interconnections make up a generic node in the RCS reference model architecture. Each module in the node may have an operator interface.

Axiom 6. The complexity of the real world environment can be managed through focusing attention.

Df: **focusing attention**

the commitment of computational resources to processing selected sensory data.

Intelligent systems typically operate in a real-world environment which is infinitely rich with detail, but the computational resources available to any intelligent system are finite. No matter how fast and powerful computers become, the amount of computational resources that can be embedded in any practical system will be limited. Fortunately, at any point in time and space, most of the detail in the environment is irrelevant to the immediate task of any particular node in the intelligent system. Therefore, the key to building practical intelligent systems lies in understanding how to focus the available computing resources on what is important, while ignoring what is irrelevant.

The problem of distinguishing what is important from what is irrelevant can be approached from two perspectives: top down and bottom up. Top down, what is important is related to behavioral goals. The intelligent system is driven from high-level goals to focus attention on objects specified by the task, using resources identified by task knowledge as necessary for successfully accomplishing task goals. Top down goals and high level perceptions generate expectations of what should be encountered during the evolution of the task.

Bottom up, what is important is the unexpected, unexplained, unusual, or dangerous. The lower level sensory perception functions detect variations between what is expected and what is observed. These low level error signals are processed first by control laws in behavior generating modules in lower level nodes that generate corrective actions to bring the process back on track. However, if low level reactive control laws are incapable of resolving the differences between expectations and observations, errors filter up to higher levels where plans may need to be revised and goals restructured. The lower levels also compute attributes of signals or images that may indicate problems or emergency conditions, such as limits being exceeded on position, velocity, acceleration, vibration, pressure, force, current, voltage, or thermal sensor signals.

In either case, hierarchical layering provides a mechanism for focusing the computational resources of the lower levels on particular regions of time and space. Higher level nodes with broad perspective and long planning horizon determine what is important, while the lower levels detect anomalies and attend to details of correcting errors and following plans. In each node at each level, computing resources are committeed to issues relevant to the decisions that must be made within the scope of control and time horizon of that level.

At the top of the hierarchy, categorical imperatives influence the selection of goals and the prioritization of tasks throughout the enterprise. At intermediate levels, tasks with goals and priorities are received from the level

above, and sub tasks with sub goals and attention priorities are output to the level below.

At each level, global goals are refined and focused onto more narrow and higher resolution sub goals. At each level, attention is focused into a more narrow and higher resolution view of the world. The effect of each hierarchical level is thus to geometrically refine the detail of the task and the view of the world, while only linearly

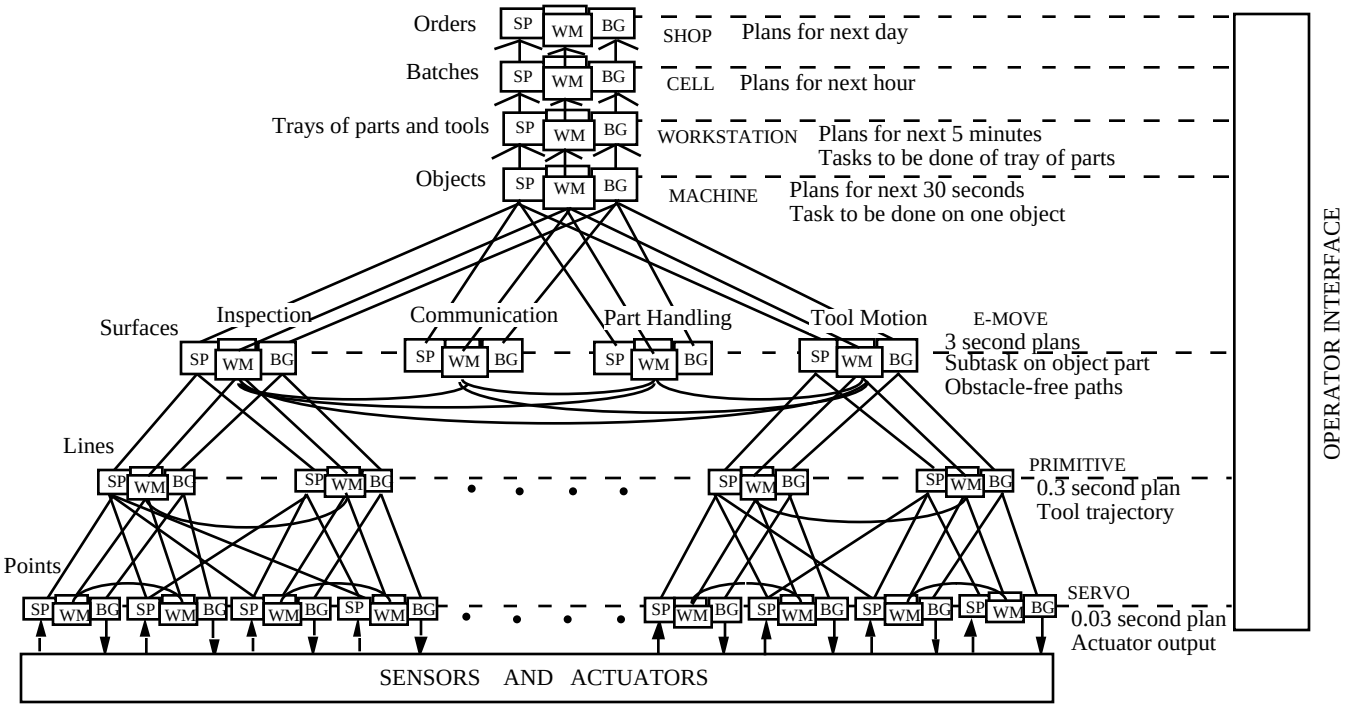

Figure 2. The RCS reference model architecture for an intelligent machining center. Processing nodes are organized such that the BG modules form a command tree. Information in the KD is shared between WM modules in nodes within the same subtree. KD modules are not shown in this figure. On the right, are examples of the functional characteristics of the BG modules at each level. On the left, are examples of the type of entities recognized by the SP modules and stored by the WM in the KD knowledge database at each level. Sensory data paths flowing up the hierarchy typically form a graph, not a tree. VJ modules are hidden behind WM modules. An operator interface provides input to, and output from, modules in every node.

increasing the computational power required of the intelligent system.

At the bottom of the hierarchy and external to the control system, are actuators that act on the world environment, and sensors that transform events in the world into information signals for the control system. The external world environment contains a variety of real objects, such as materials, tools, machines, and fixtures as well as other intelligent agents, and forces of nature, all of which have states and may cause events and situations to occur.

The generic node illustrated in Figure 1 can be used to construct a distributed hierarchical RCS reference model architecture. Depending on where the generic node resides in the distributed hierarchical structure of the RCS architecture, it might serve as an intelligent controller for an actuator, a subsystem, a production machine, a workstation, a manufacturing cell, a shop, or facility. Or it might represent the functionality of a human operator, worker, or management unit at any level in the corporate enterprise.

An example of an RCS reference model architecture for an individual intelligent machine (a flexible machining center) is shown in Figure 2. This diagram consists of a hierarchy of control nodes, each of which contain modules corresponding to the functional elements illustrated in Figure 1. Each node consists of a behavior generation (BG), world modeling (WM), sensory perception (SP), and knowledge database (KD) module (not shown in Figure 2.) Most nodes also contain a value judgment (VJ) module (hidden behind the WM module in Figure 2.) Each of the

nodes is, therefore, an intelligent controller. An operator interface may access modules in all nodes at all levels.

Figure 2 illustrates a machining center with four subsystems: tool motion, part handling, communication, and inspection. Each of the four subsystems have one or more mechanisms, each having one or more actuators and sensors. For example, the manipulation subsystem may consist of a spindle and tool path controller with several axes of continuous-motion path control, plus a tool changer consisting of a tool carousel and a manipulator, each having two or more actuators and sensors. The part handling subsystem might be a pallet feeding mechanism, which consists of a conveyor and buffer, pallet shuttle, and a pair of turn tables. The communication subsystem might consist of a message encoding subsystem, a protocol syntax generator, and communications bus interface. The inspection subsystem might consist of mechanisms that use cameras and touch probes to detect and track objects, surfaces, edges and points, and compute trajectories for probe drive motors, and pan, tilt, and focus actuators.

The operator interface provides the ability for the operator to start or stop the system at any time, to single step, to override the feed rate of the motion controller, to actuate or monitor any of the tool change or pallet feeding mechanisms. Using the operator interface, a human operator is able to run diagnostic programs in the case of failures, display control programs (plans) while they are being executed, generate graphics images of tool paths, display volumes to be removed from parts by NC programs, and portray shaded color images or wire frame models of parts with dimensions and tolerances indicated with overlays.

In Figure 2, three levels of control are shown above the node representing the individual machine controller. This represents the chain of command that exists in the manufacturing environment above the individual machining center. There, of course, may be more than three levels above the machine. In Figure 2, the nodes at the upper levels may have the same degree of branching that exists at the lower levels, but this is not shown in order to simplify the diagram.

The horizontal curved lines between WM modules represent the sharing of state information between nodes within sub trees in order to synchronize related tasks.

The functionality of each level in the RCS reference model hierarchy is defined by the characteristic timing, bandwidth, and algorithms chosen for decomposing tasks and goals at each level. Typically these are design choices that depend on the dynamics of the processes being controlled. The numbers shown in Figure 2 are representative of those appropriate for a machining center. For other types of systems, different numbers would be derived from different system design parameters.

Applications

Over the past two decades, the RCS architecture has been used in the implementation of a number of experimental projects. These include:

A Horizontal Machining Workstation

This project was part of the NBS Automated Manufacturing Research Facility (AMRF) (Albus, *et al.*, 1982). It implemented an intelligent control system for a robot with a structured-light machine-vision system, a machine tool, an automatic fixturing system, and a pallet shuttle. The robot included a quick change wrist, a part handling gripper with tactile sensors, and a tool handling gripper for loading and unloading tools in the machine tool magazine. The discrete event elements were represented as state-tables, and a wide variety of sensory interactive behaviors were demonstrated. These included locating and recognizing parts, determining the orientation of parts presented in trays, and automatically generating part handling sequences for part and tool loading and unloading (Wavering and Fiala, 1987).

A Cleaning and Deburring Workstation

This project was also part of the AMRF. It included two robots, a set of buffing wheels, a part washer/dryer machine, and a variety of abrasive brushes. Part geometry was input from a CAD database. Deburring parameters such as forces and feed rates were selected from a menu by an operator. Discrete event part handling sequences were automatically planned and executed for loading parts in a vise, and turning parts over to permit tool and gripper access. Continuous force sensing and control algorithms were used to modify the planned paths so as to compensate for inaccuracies in robot kinematics and dynamics (Murphy, *et al.*, 1988.)

An Advanced Deburring and Chamfering System

This project integrates off-line programming, real-time control, and active tool technologies in a hybrid control system. It automatically grinds precision chamfers on complex parts manufactured from hard materials such as aircraft jet engine components. The workstation consists of a grinding tool mounted on a micro positioner with computer controlled force and stiffness parameters, integrated with a 6 degree-of-freedom robot, and an indexing table for part fixturing. Part geometry is derived from

standard IGES CAD data formats. Edge selection is performed by a human operator. Required tool force is automatically generated by formula using the cutting depth, feeds, and speeds input by the operator. Under a cooperative research and development agreement, a prototype production cell is currently being tested at Pratt & Whitney's East Hartford, CT site (Stouffer, *et al.*, 1993.)

NBS/NASA Standard Reference Model Architecture for the Space Station Telerobotic Servicer (NASREM)

This project was sponsored by NASA Goddard Space Flight Center. NASREM was used by Martin Marietta to develop a control system for the space station telerobotic servicer. NASREM compliant algorithms have been developed for force servoing, impedance control, and real-time image processing of robotic and telerobotic systems at NIST, Martin Marietta Denver, Lockheed Palo Alto, Goddard, and at a number of university and industry labs in the United States and Europe (Albus, *et al.*, 1989.)

Coal Mining Automation

This project transferred the RCS architecture and methodology to a team of researchers in the U.S. Bureau of Mines, and in turn, to the commercial mining industry. A comprehensive mining scenario was developed starting with a map of the underground region to be excavated, the machines to be controlled, and the mining procedures to be applied. Based on this scenario, a hybrid control system with simulation and animation was designed, built, and demonstrated. The same control system was later demonstrated with an actual mining machine and sensors (Huang, *et al.*, 1991.)

An Nuclear Submarine Maneuvering System

This ARPA sponsored project demonstrated the design and implementation in simulation of maneuvering and engineering support systems for a 637 class nuclear submarine. The maneuvering system involves an automatic steering, trim, speed, and depth control system. The system demonstrated the ability to execute a lengthy and complex mission involving transit of the Bering Straits under ice. Ice avoidance sonar signals were integrated into a local map using a CMAC neural network memory model (Albus, 1975). Steering and depth control algorithms were developed that enabled the sub to avoid hitting either the bottom or the ice while detecting and compensating for random salinity changes under the ice by making trim and ballast adjustments. The submarine engineering support system demonstrated the ability to respond to an emergency such as a lubrication oil fire by reconfiguring ventilation systems, rising in depth to snorkel level, and engaging the diesel engines for emergency propulsion (Huang, *et al.*, 1993.)

A U.S. Postal Service Automated Stamp Distribution Center.

This RCS system demonstrated the ability to route packages through a series of carousels, conveyors, and storage bins, to maintain precise inventory control, provide security, and generate maintenance diagnostics in the case of system failure. The stamp distribution center was designed and tested first in simulation and then implemented as a full-scale system. The system contained over 220 actuators, 300 sensors, and ten operator workstations. An even larger and more complex RCS system for controlling a general mail facility is still under development (ATR Report, 1994.)

Multiple Autonomous Undersea Vehicles

This DARPA sponsored project developed an intelligent control system for a pair of experimental underwater vehicles designed and built by the University of New Hampshire. The RCS control system included a real-time path planner for sonar-guided obstacle avoidance, and a real-time map builder for constructing a topological map of the bottom. A series of tests was conducted in Lake Winnipasaki during the fall of 1987 (Herman and Albus, 1988.)

Unmanned Ground Vehicles

Two versions of a RCS for unmanned ground vehicles have been implemented on an Army HMMWV light truck. One version enables the vehicle to be driven remotely by an operator using TV images transmitted from the vehicle to an operator control station. This version has a retrotraverse mode that permits the vehicle to autonomously retrace paths previously traversed under remote control, using GPS and an inertial guidance system (Szabo, *et al.*, 1990.)

A second version has demonstrated the ability to drive the HMMWV automatically using TV images processed through a machine-vision system with a real-time model matching algorithm for tracking lane markings. A World Model estimate of the lane markings is compared to observed edges in the image, and a new estimate is computed every 15 milliseconds, with pipeline latency of less than 150 milliseconds. The RCS real-time vision processing system has enabled this vehicle to drive automatically at speeds up to 100 km/hr (60 mph) on the highway, and at speeds up to 55 km/hr (35 mph) on a winding test track (Schneiderman and Nashman, 1994.)

Planning and Control for a Spray Casting Machine.

The RCS architecture has been applied for planning and control of the automated Spray Casting Machine "OSPREY" which has been developed and manufactured by MTS Corporation (Minneapolis, MN) in cooperation with

Drexel University. The system has three levels of resolution (Cleveland and Meystel, 1990.)

An Autonomous Mobile Vehicle.

An autonomous vehicle was assembled and tested by Drexel University in 1984-1987. The goal of the effort was to investigate the RCS architecture with four levels of resolution: "Planner-Navigator-Pilot" on the top of the lower level control of steering and propulsion (Meystel, 1991.)

An Open Architecture Enhanced Machine Controller

The RCS reference model is currently being used as the basis for an open architecture Enhanced Machine Controller (EMC) for intelligent control of manufacturing equipment, such as machine tools, robots, and coordinate measuring machines. A prototype EMC has been installed and is being evaluated in the General Motors Powertrain prototype production facility in Pontiac Michigan under a DoE-TEAM/NIST-EMC government/industry consortium. The goal of this effort is to develop application program interface (API) standards for open architecture controllers (Proctor and Michaloski, 1993.)

Current work at NIST and elsewhere is pursuing more complex implementations of RCS. Work is also in progress to develop an engineering design methodology and a set of software engineering tools for developing RCS systems.

References

Albus, J.S., McLean, C.R., Barbera, A.J. and Fitzgerald, M.L. (1982). Architecture for Real-Time Sensory-Interactive Control of Robots in a Manufacturing Facility. *Proceedings of the Fourth IFAC/IFIP Symposium -- Information Control Problems in Manufacturing Technology*.

Albus, J.S., McCain, H.G., and Lumia, R. (1989). NASA/NBS Standard Reference Model for Telerobot Control System Architecture (NASREM). *NISTTN 1235*, National Institute of Standards and Technology, Gaithersburg, MD.

Albus, J.S. (1991). Outline for a Theory of Intelligence. *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 21, No. 3, pp. 473-509

Albus, J.S. (1993). A Reference Model Architecture for Intelligent Systems Design. In: *An Introduction to Intelligent and Autonomous Control*, (Antsaklis, P.J., and Passino, K.M., (Ed.)), pp. 27-56, Kluwer Academic Publishers, Boston

ATR Report. (1993) Stamp Distribution Network, *USPS Contract Number 104230-91-C-3127 Final Report*, Advanced Technology & Research Corp, Burtonsville, MD., 20866-1172

Cleveland, B., Meystel, A. (1990). Predictive Planning + Fuzzy Compensation=Intelligent Control. *Proceedings of the 5th IEEE International Symposium on Intelligent Control*, Philadelphia, PA.

Herman, M. and Albus, J.S. (1988). Overview of the Multiple Autonomous Underwater Vehicles (MAUV) Project. *Proceedings of IEEE International Conference on Robotics and Automation*, Philadelphia, PA.

Huang, H.M., Quintero, R. and Albus, J.S. (1991). A Reference Model, Design Approach, and Development Illustration toward Hierarchical Real-Time System Control for Coal Mining Operations. In: *Advances in Control & Dynamic Systems*, (C.T. Leondes (Ed.)), **Vol. 46, part 2 of 5,** pp. 173-254 Academic Press, San Deigo, CA.

Huang, H.M., Hira, R. and Quintero, R. (1993). A Submarine Maneuvering System Demonstration Based on the NIST Real-Time Control System Reference Model. *Proceedings of the 8th IEEE International Symposium on Intelligent Control*, Chicago, IL.

Meystel, A. (1991). *Autonomous Mobile Robots: Vehicles with Cognitive Control*, World Scientific, Singapore

Meystel, A. (1993). Nested Hierarchical Control. In: *An Introduction to Intelligent and Autonomous Control*, (Antsaklis, P.J., and Passino, K.M. (Ed.)), pp. 129-161, Kluwer Academic Publishers, Boston

Murphy, K.N., Norcross, R.J. and Proctor, F.M. (1988). CAD Directed Robotic Deburring. *Proceedings of the Second International Symposium on Robotics and Manufacturing Research, Education, and Applications*, Albuquerque, NM.

Proctor, F. and Michaloski, J. (1993). Enhanced Machine Controller Architecture Overview, *NISTIR 5331*, National Institute of Standards and Technology, Gaithersburg, MD.

Schneiderman, H. and Nashman, M. (1994). Visual Tracking for Autonomous Driving. *IEEE Transactions on Robotics and Automation*, **Vol. 10, No. 6,** p.769-775

Senehi, M.K., Kramer, T.J., Michaloski, J., Quintero, R., Ray, S.R., Rippey, W.G., Wallace, S. (1994). Reference Architecture for Machine Control Systems Integration: Interim Report. *NISTIR 5517*, National Institute of Standards and Technology, Gaithersburg, MD.

Stouffer, K., Michaloski, J., Russell, R. and Proctor, F. (1993). ADACS - An Automated System for Part Finishing. *Proceedings of the IECON '93 International Conference on Industrial Control and Instrumentation*, Maui, Hawaii.

Szabo, S., Scott, H.A., Murphy, K.N. and Legowik, S.A. (1990). Control System Architecture for a Remotely Operated

Unmanned Land Vehicle, *Proceedings of the 5th IEEE International Symposium on Intelligent Control*, Philadelphia, PA.

Wavering, A.J. and Fiala, J.C. (1987). Real-Time Control System of the Horizontal Workstation Robot, *NBSIR 88-3692*, National Institute of Standards and Technology, Gaithersburg, MD.