Proceedings of the International Symposium on Intelligent Control, Vancouver, Canada, October 27 - 30, 2002.

# Knowledge Engineering for Real Time Intelligent Control

John M. Evans[1], Elena R. Messina, James S. Albus
National Institute of Standards and Technology
Gaithersburge, MD 20899-8230
elena.messina@nist.gov, james.albus@nist.gov

***Abstract* -- The key to real-time intelligent control lies in the knowledge models that the system contains. We argue that there needs to be a more rigorous approach to engineering the knowledge within intelligent controllers. Three main classes of knowledge are identified: parametric, geometric/iconic, and symbolic. Each of these classes provides unique perspectives and advantages for the planning of behaviors by the intelligent system. Examples of each from demonstration systems are presented.**

***Index terms*—knowledge engineering, intelligent control, intelligent systems, world models, knowledge representation, knowledge bases, software architecture**

## I. INTRODUCTION

The concept of intelligence in control applies to a variety of approaches to extending classical control theory that include learning, non-linear control, model-based control, and, in general, control of complex systems that will "do the right thing" when confronted with unexpected or unplanned situations [4]. It can be said that all "intelligent" systems have some knowledge of the system to be controlled or that they use some model of the system in calculating control outputs. In fact, the American Heritage Dictionary defines intelligence as "the capacity to acquire and apply knowledge."

Creating the knowledge – i.e., the model – of the system to be controlled is one branch of what is known as knowledge engineering. The real-time aspects of control make this problem domain different than other knowledge engineering problems such as large-scale ontologies. For example, there is a need for designing non-symbolic aspects of the system's knowledge, such as map-based world models. We argue that intelligent control requires several *different* types of knowledge and representation. Reference model architectures are useful in guiding the decisions in what type of knowledge is needed in the software and how it should be represented.

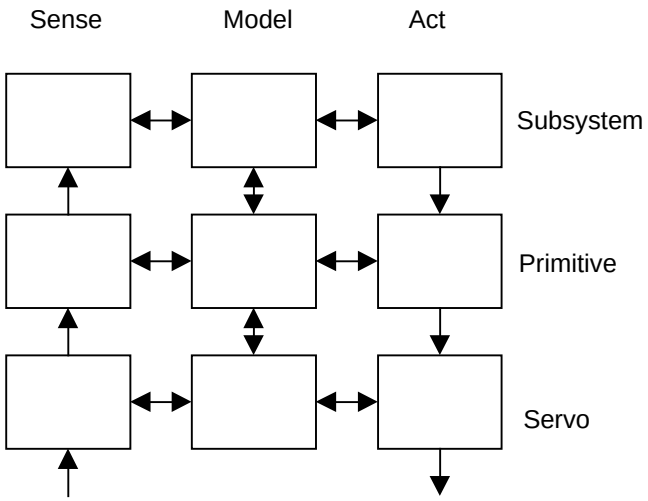


**Figure 1: General Framework for an Intelligent Control System**

## II. CLASSES OF KNOWLEDGE

A general framework for a model-based control system is shown schematically in Fig. 1. This framework shows a hierarchical control structure with a world model hierarchy explicitly interspersed between the sensor processing hierarchy and the behavior generation or task decomposition hierarchy, allowing for model-based perception and model-based control [1], [2]. Example labels for three of the levels (subsystem, primitive, and servo), as defined in [1] are shown. This paper presents an overview of the data needed for the world model hierarchy. We argue that there are three distinctly different classes of knowledge in such a control hierarchy: system parameters at the lower levels; maps, images and object models at the middle levels,; and symbolic data at

[1] Retired

the highest levels. We will consider each of these below. Note that each level may contain some or all of the classes of knowledge, but in general, there won't be use of symbolic knowledge at the lowest (servo) level, and the highest levels will mostly use symbolic knowledge. Traditionally, iconic, parametric, or numeric information is not addressed by knowledge engineering. We believe that it is necessary to consider this type of representation as well in designing the knowledge models for intelligent systems.

We can further distinguish knowledge that is learned or acquired, which we will call *in situ* knowledge, from knowledge that is pre-programmed or referenced from an outside data base, which we will call *a priori* knowledge. This provides a framework for considering learning and adaptive control.

There is yet a third means of differentiation of types of knowledge, which is to distinguish knowledge of things (nouns), and knowledge of actions or tasks or behaviors (verbs). This becomes very useful at higher levels in considering the interaction of autonomous machines with complex environments, where appropriate behaviors depend upon the nature of the objects encountered in the environment; another application where this distinction arises is generative process planning for assembly or machining or inspection.[16][9][18] A distinction between object models (things) and behavior models (actions) also helps the system designer in matching the sensor processing and world modeling specifications to the control task specifications.

### *A. Parametric Level Knowledge*

The lowest levels of any control system, whether for an autonomous robot, a machine tool, or a refinery, are at the servo level, where knowledge of the value of system parameters is needed to provide position and/or velocity and/or torque control of each degree of freedom by appropriate voltages sent to a motor or a hydraulic servo valve. The control loops at this level can generally be analyzed with classical techniques and the "knowledge" embedded in the world model is the specification of the system functional blocks, the set of gains and filters that define the servo controls for a specific actuator, and the current value of relevant state variables. These are generally called the system parameters, so we refer to knowledge at this level as parametric knowledge.

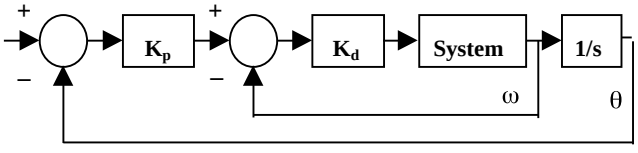


**Figure 2: PD Servo Control**

Fig. 2 shows a traditional PD servo control for a motor of a robot arm. All six or seven motors that drive the arm will have basically the same servo control, but each will have different parameters because there are different size motors driving different loads at different points in the arm. Any errors that deal with a single degree of freedom, such as ball screw lead errors, contact instabilities, and stiction and friction are best compensated for at this level.

Learning or adaptive control systems [6],[28] may allow changes in the system parameters and even autonomous identification of the system parameters, but the topology of the control loops is basically invariant and set by the control designer. No robot is going to invent itself a torque loop in the field, although it could well change the gain of a position or velocity loop as it learns to optimize a task.

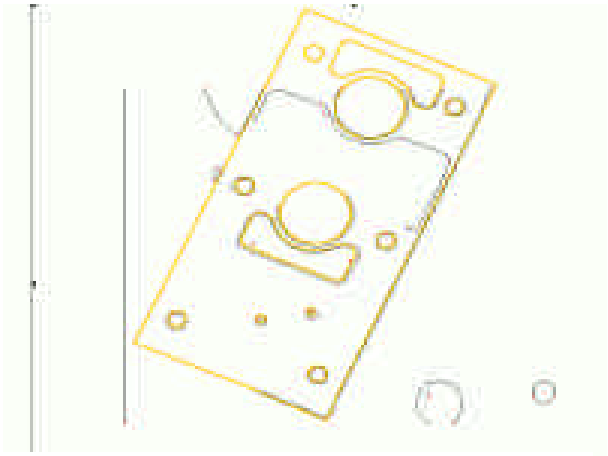

**Figure 3: Part Pose Computation**

### *B. Iconic or Geometric Level Knowledge*

Above the servo level are a series of control loops that coordinate the individual servos and that require what can be generally called "geometric knowledge," "iconic knowledge,." or "patterns." Iconic knowledge 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 or real number data representing a physical property such as light intensity, color, altitude, or density. Examples of iconic knowledge include includes digital terrain maps, 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.

For industrial robots, machine tools, and coordinate measuring machines, the first level above the servo level deals with the kinematics of the machine, relating the geometry of the different axes to allow coordinated control. Linear, circular and other interpolation and motion in world or tool coordinates is enabled by such coordination. The "knowledge" here may be the kinematic equations or Jacobian coefficients that define the geometric relationships of the axes, or the mathematical routines for interpolation or coordinate transformations. It is at this level that systematic multi-dimensional geometric errors such as non-orthogonality of axes of a machine tool and Abbe offset errors are considered. [3]. Fig. 3 shows an investigation of fixtureless inspection, in which a part is placed on the table of an inspection machine without a fixture and the pose of the part is determined by matching an image of the part (dark edges) with a predicted image derived by rotating and translating a CAD model of the part (light edges) [21],[15].

For mobile autonomous robots, digital maps are the natural way of representing the environment for path planning and obstacle avoidance, and provide a very powerful mechanism for sensor fusion since the data from multiple sensors can be represented in a common format [14]. Digital terrain maps are essentially two-dimensional grid structures that are referenced to some coordinate frame tied to the ground or earth. 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. 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.

The mobile robot literature references occupancy grids as a specific approach to building quantized local maps with some measure of certainty applied to contents of each grid element [22] [24][7] [10]. This is particularly useful with sensor modalities that are noisy or sensitive over wide angles such as sonar.

Fig. 4 shows a typical local map from a mobile robot navigating through an indoor environment. The robot’s position at the center is indicated by marking the occupied cells with “R.” The numbers in certain cells indicate the degree of confidence that there is an obstacle occupying that cell. Fig. 5 shows a higher level map for path planning for outdoor navigation. This map contains

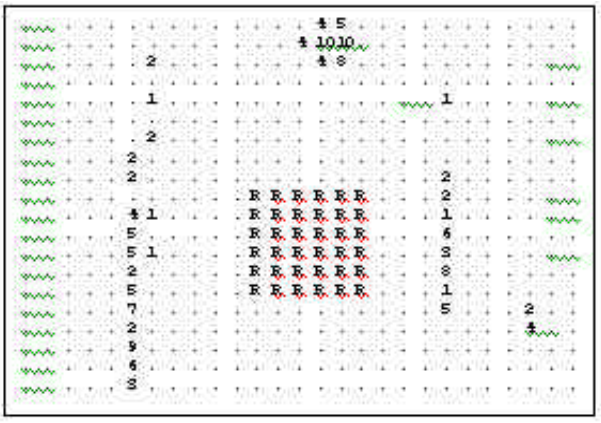

**Figure 4: Occupancy Grid Map for Mobile Robot Navigating in a Hallway and Approaching an Obstacle**

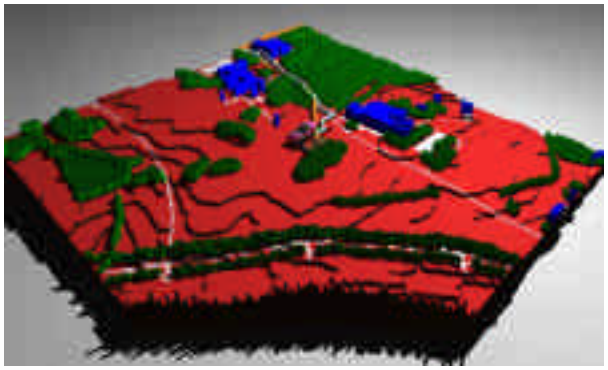

**Figure 5: Multi-featured Digital Terrain Map**

several feature layers, including elevation, vegetation, roads, buildings and obstacles.

### *C. Symbolic Knowledge*

At the highest levels of control, knowledge will be symbolic, whether dealing with actions or objects. It is at this level that a large body of relevant work exists in knowledge engineering for domains other than real-time control, such as formal logic systems or rule based expert systems. Whether the knowledge is represented in terms of mathematical logic, rules, frames, or semantic nets, there is a formal linguistic structure for defining and manipulating and using the knowledge. A good presentation of different concepts of knowledge representation is found in Davis [11].

An example of a formal description of a solid model of a part is shown in Fig. 6. A block is being described using International Standards Organization Standard for the Exchange of Product Model Data (STEP) Part 21 [17]. Note the fundamentally different nature of this linguistic representation from a geometric representation where, for example, a block might be represented by equations of six

planes with bounding curves and a coordinate transformation matrix to position the block within a given coordinate system.

```
DATA;
#10 = BLOCK_BASE_SHAPE(#20,#30,#70,#80);
#20 = NUMERIC_PARAMETER('block Z
dimension',50.,'mm');
#30 = ORIENTATION(#40,#50,#60);
#40 = DIRECTION_ELEMENT((0.,0.,1.));
#50 = DIRECTION_ELEMENT((1.,0.,0.));
#60 = LOCATION_ELEMENT((62.5,37.5,0.));
#70 = NUMERIC_PARAMETER('block Y
dimension',75.,'mm');
#80 = NUMERIC_PARAMETER('block X
dimension',125.,'mm');
#90 = SHAPE((),#10,());
#100 = PART('out','rev1','','simple
part','insecure',(),#90,(),(),(),$,(),
(#110),(),());
#110 = MATERIAL('aluminum','soft aluminum',$,(),());
```

**Figure 6: STEP Representation of a Block**

Linguistic representations provide ways of expressing knowledge and relationships, and of manipulating knowledge, including the ability to address objects by property. Tying symbolic knowledge back into the geometric levels provides the valuable ability to identify objects from partial observations and then extrapolate facts or future behaviors from the symbolic knowledge. In the manufacturing domain, using a feature-based representation (which is symbolic) is reasonable at the generative planning level (Fig. 7a). Graphical primitives (Fig. 7b) that relate to the geometry can be tied to features

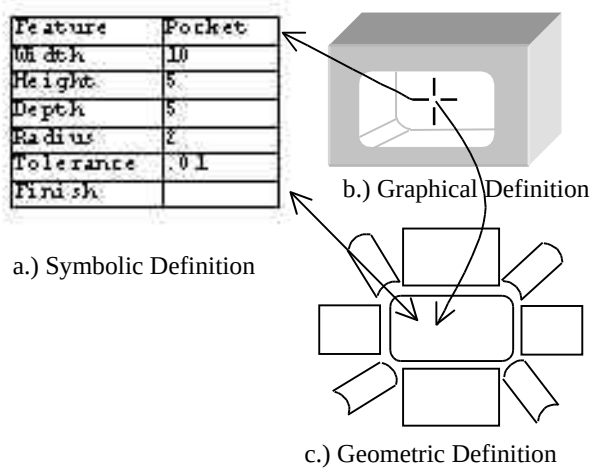


**Figure 7: Pocket Feature.**

to let users easily pick a feature (such as a pocket) by selecting on a portion of it on the screen. The geometric representation of each edge and surface that comprise a feature (Fig. 7c) can be tied to the feature definition in order to facilitate calculations for generating the tool paths.

Ontologies are definitions and organizations of classes of facts and formal rules for accessing and manipulating (and possibly extending) those facts. There are two main approaches to creating ontologies, one emphasizing the organizational framework, with data entered into that framework, and the other emphasizing large scale data creation with relationships defined as needed to relate and use that data. Cyc [19] is an example of the latter, an effort to create a system capable of common sense, natural language understanding, and machine learning.

Linguistics is useful for human-machine communication and for sharing and exchanging information amongst robots. Many of the results of such formal methodologies can be useful to control applications. Formal methods can be used to prove correctness and completeness of the knowledge representation. Higher-level behaviors and environmental situations are more readily and efficiently expressed using linguistic (versus numeric) representations. For instance, at the higher levels of control, describing the environment near an autonomous driving vehicle by only naming objects is more compact than an enumeration of a series of surfaces and their mathematical descriptions. Of course, the geometric descriptions are necessary in order to avoid collisions, but that would be handled by a lower control level.

## III. DIFFERENT APPROACHES TO USING KNOWLEDGE FOR CONTROL

### *A. CONTROL USING ONLY HIGH LEVEL SYMBOLIC KNOWLEDGE*

Much early robot work was carried out in the context of AI research in Computer Science departments at major universities. This had the unfortunate result of uncoupling robotics from controls engineering, mechanical engineering, and sensor engineering in other departments and focused on high level issues of perception, planning, and reasoning.[13]

After struggling for the better part of two decades, the AI community turned away from robotics to more fruitful applications. Little of this early work ever found practical application, although recent work which couples higher

level planners or agents to real systems has found new advocates, particularly for space applications. [26][27]

### *B. CONTROL USING ONLY LOW-LEVEL KNOWLEDGE*

The behavioralist school of robotics, as started by Rodney Brooks at MIT, rejected the idea of purely symbolic control as sterile and irrelevant to robots that could effectively interact with the real world. Brooks proposed using insects as a model, defining the controls as a series of reactive behaviors that directly related sensor inputs to behaviors through finite state machines. More complex behaviors were able to inhibit or subsume simpler lower level behaviors, hence this was called a subsumption architecture [8].

Some significant accomplishments were achieved, including the learning experiment that Brooks carried out to demonstrate that a hexapod with a network of controllers could learn to walk with the appropriate tripod gait [20]. However, Brooks and others have explicitly rejected the concept of a world model, arguing that the world was its own model, and as a result behavioralist or reactive systems have not been applied to any problems of great complexity. Hybrid systems, such as the deliberative-reactive systems proposed by Arkin [5] or Thorpe [25] have attacked more complex problems. However, these systems tend to be brittle since the behaviors that are learned and the decision rules for merging of these behaviors tend to be specific to the training environment.

### *C. CONTROL WITH MULTIPLE LEVELS OF KNOWLEDGE*

The most significant and complex autonomous mobile robot built to date is the Army's Experimental Unmanned Vehicle (XUV) being developed for scout missions (reconnaissance, surveillance, and target acquisition (RSTA) missions). The architecture for this vehicle is called 4D/RCS, merging the work of Dickmanns in Germany on road following [12] and the work of Albus at NIST [2]. Both use data from multiple sensors to build a world model and then use that model for planning what the vehicle should do.

The Army XUV has successfully navigated many kilometers of off road terrain, including fields, woods, streams and hilly terrain, given only a few way points on a low resolution map by an Army scout. The XUV used its on-board sensors to create high definition multi-resolution maps of its environment and then navigated successfully through very difficult terrain.

This is basically a demonstration of the use of multi-resolutional maps as a means of knowledge representation for sensor fusion and path planning in autonomous mobile robots. Over the next several years symbolic knowledge will be added to enable tactical behaviors and human-machine interaction. This will create a machine that will indeed be considered intelligent. A brief discussion of some of the design aspects of knowledge content and representation in building such as system are presented in the following section.

## IV. CONSIDERATIONS IN DESIGN OF KNOWLEDGE AND ITS REPRESENTATION

There are several design considerations when implementing the world model for an intelligent controller. A high level discussion only is possible within the space constraints of this paper. A key concept is that a single, monolithic world model is too restrictive in terms of performance and capabilities.

In this brief summary, we elaborate on the description of the 4D/RCS autonomous scout vehicle software architecture. 4D/RCS integrates the functional elements, knowledge representations, and flow of information so that intelligent systems can analyze the past, perceive the present, and plan for the future. A reference model architecture is essential for guiding the design and engineering of complex real-time control systems.

The 4D/RCS architecture is a hierarchical control structure, composed of RCS Nodes, and with different range and resolution in time and space at each level. The functionality of each level in the 4D/RCS hierarchy is defined by the functionality, characteristic timing, bandwidth, and algorithms chosen by Behavior Generation processes for decomposing tasks and goals at each level. Hierarchical layering enables optimal use of memory and computational resources in the representation of time and space. At each level, state variables, images, and maps are maintained to the resolution in space and time that is appropriate to that level. At each successively lower level in the hierarchy, as detail is geometrically increased, the range of computation is geometrically decreased. Also, as temporal resolution is increased, the span of interest decreases. This produces a ratio that remains relatively constant throughout the hierarchy.

Each RCS Node contains the same functional elements, yet is tailored for that level of the hierarchy and the node’s particular responsibilities. An RCS Node contains Sensory Processing (SP), Behavior Generation (BG), World Modeling (WM), and Value Judgement

(VJ). At every level of the control hierarchy there are deliberative planning processes that receive goals and priorities from superiors and decompose them into subgoals for subordinates at levels below. At every level, reactive loops respond quickly to feedback to modify planned actions so that goals are accomplished despite unexpected events. At every level, sensory processing filters and processes information derived from observations by subordinate levels. Events are detected, objects recognized, situations analyzed, and status reported to superiors at the next higher level. The sensory processing results are stored in the world model for that particular level.

At every level, sensory processing and behavior generation processes have access to a model of the world that is resident in a knowledge database. This world model enables the intelligent system to analyze the past, plan for the future, and perceive sensory information in the context of expectations. At each level, cost functions enable value judgments and determine priorities that support intelligent decision making, planning, and situation analysis. The cost functions can be dynamic and are determined by current commands, priorities, user preferences, past experiences, and other sources.

Therefore, the design of the knowledge requirements at each level is driven by the responsibilities of that level. What commands will an RCS Node be able to execute? What is its required control loop response time? What spatial scope does it need to understand? What types of entities does it have to deal with?

At the servo level, an RCS Node receives commands to adjust set points for vehicle steering, velocity, and acceleration or for pointing sensors. It must convert these commands to motion or torque commands for each actuator and issue them at high frequencies (e.g., every 5 ms). The planning horizon is about 50 ms. The knowledge used at the servo level is primarily single-valued state variables: actuator positions, velocities, and forces, pressure sensor readings, position of switches, and gear shift settings.

At the higher Subsystem level, the Autonomous Mobility node, which is part of the vehicle's locomotion controller hierarchy, generates a schedule of waypoints that are sent to the subordinate Primitive controller. Commands that the Autonomous Mobility RCS Node accepts include directives to follow a schedule of waypoints to avoid obstacles, maintain position relative to nearby vehicles, and achieve desired vehicle heading and speed along the desired path. Knowledge used at this level supports planning movement through 3D terrain, hence digital terrain maps (which are forms of iconic knowledge), with multiple registered attribute layers are appropriate.

Planning for mobility at this level is concerned with obstacles (both positive and negative, i.e., holes), elevation, potentially roads, if it is to follow roads, and observability, if it is to perform stealthy movements. A cost-based search through a graph whose nodes are derived from elements of the regular terrain grid is used to find the lowest-cost path that achieves the specified objectives. The map-based format also provides a convenient "receptacle" for registering and fusing information from multiple sensors with each other and with *a priori* information, such as from digital terrain maps. The subsystem level of the hierarchy outputs a new plan about every 500 ms, and the planning horizon at this level is about 5 s into the future. The spatial scope is roughly 50 m, with a resolution of about 40 cm. The extents of the space considered are based on the planning horizon and vehicle velocity. The grid resolution is based on engineering considerations, like computational resources available and what resolution the onboard sensors can provide.

Higher still in the hierarchy is the Section Level. This is the controller for a group (2 or more) individual scout vehicles. The section RCS Node is responsible for assigning duties to the individual vehicles and coordinating their actions. Orders coming into the section level are tactical maneuvers, including mission goals, timing and coordination requirements. The planning horizon is 10 minutes into the future, and new plans are sent to subordinates about every minute. Knowledge at the Section Level includes digital terrain maps, typically covering about 2 km, at low resolution (30 m), with multiple attribute layers, such as roads (of various types), vegetation, fences, buildings, as well as enemy locations and militarily significant attributes. Enemy locations may be noted within the grid-based map, but more extensive symbolic information about the situation is associated with the grid locations. The symbolic information could include details about the enemy force such as number of soldiers, weapons, and estimated travel direction. This group of information is best kept in a separate knowledge structure and may be amenable to rule or case-based reasoning tools (such as [23]). At the very least, the Value Judgement function will need to convert the knowledge that " a band of 23 soldiers and 1 tank is moving toward location x, y with 60 % probability at velocity of 10 miles/day" into a set of costs that can be tied to the map grid and utilized by the graph-based search to generate the vehicle plans. Therefore, this level is an example where map-based (graph search) planning and symbolic reasoning tools may be used.

At most of the levels, there is some combination of *a priori* knowledge and *in situ* knowledge. At lower levels concerned with mobility, the maps are primarily sensor-

generated, however, there may be pre-computed kinematically correct steering curves that are overlaid on the planning graph. At higher levels, more *a priori* knowledge is used, e.g., as digital terrain maps and descriptions of enemy vehicles and capabilities.

## V. CONCLUSIONS

No one type of knowledge representation is adequate for all purposes. Davis [11] 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. We go further and argue that there is a requirement for integrating iconic and parametric knowledge with multiple types of symbolic knowledge and that, as Davis argues, there is a basic need for a representational architecture to provide a basis for intelligent control, which we have presented above.

The introduction of iconic data, 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. The Army XUV program provides a leading edge demonstration of the value of this approach.

## VI. REFERENCES


[1] Albus, J.S., Lumia, R., Fiala, J., Wavering, A., "NASREM -- The NASA/NBS Standard Reference Model for Telerobot Control System Architecture", Proceedings of the 20th International Symposium on Industrial Robots, Tokyo, Japan, October 4-6, 1989.

[2] Albus, J., "4-D/RCS: A Reference Model Architecture for Demo III," NISTIR 5994, Gaithersburg, MD, March 1997.

[3] ANSI/ASME B5 TC52 Committee, "Interim Revision of Methods for Performance Evaluation of Computer Numerically Controlled Machining Centers." ASME standard B5.54, Version 5.0, May 2001.

[4] Antsaklis P. J., "Defining Intelligent Control", Report of the Task Force on Intelligent Control, P.J Antsaklis, Chair, IEEE Control Systems Magazine, pp. 4-5 & 58-66, June 1994.

[5] Arkin, R., "Navigational Path Planning for a Vision-Based Mobile Robot," Robotica, 7: 49-63.

[6] Åström, K. and Wittenmark, B., Adaptive Control, Addison-Wesley, 1995.

[7] Borenstein, J., and Koren, Y., "Real Time Obstacle Avoidance for Fast Mobile Robots in Cluttered Environments." Proceedings, 1990 IEEE ICRA, Cincinnati, OH, 1990.

[8] Brooks, R., "A Robust Layered Control System for a Mobile Robot." IEEE Journal Robots & Automation, RA-2, April, 1986, p.14.

[9] Brooks, S. L. and Greenway R.B., "Using STEP to integrate design features with manufacturing features," In A. A. Busnaina, editor, ASME Computers in Engineering Conference, pages 579–586, New York, NY 10017, September 17- 20, Boston, MA 1995. ASME. SIGGRAPH and the IEEE Computer Society, ACM Press. Salt Lake City, Utah.

[10] Crowley, J.L., "Navigation for an Intelligent Mobile Robot." IEEE Journal of Robots and Automation, 1, p31, 1985

[11] Davis, R., et al. 1993. "What is in a Knowledge Representation?" AI Magazine, Spring 1993.

[12] Dickmanns, E.D., "A General Dynamic Vision Architecture for UGV and UAV." Journal of Applied Intelligence, **2**, p.251, 1992.

[13] Etherington, D., "What Does Knowledge Representation Have to Say to Artificial Intelligence?" Proceedings of the AAAI, 1997.

[14] Evans, J., and Krishnamurthy, B., "HelpMate, the trackless Robotic Courier: A Perspective on the Development of a Commercial Autonomous Mobile Robot." p.182 in Autonomous Robotic Systems, Ed. A. T. de Almeida and O. Khatib, Springer Verlag Lecture Notes in Control and Information Sciences 236, 1998.

[15] Horst, J., "A Lexiccal Analogy to Feature Matching and Pose Estimation," NISTIR 6790, Gaithersburg, MD 2002.

[16] Huaguo, L., Cuiyun, J., and Jianan, H., "A Knowledge-Based Approach for Object Classification for Assembly." Proc. IEEE International Conference on Intelligent Processing Systems, Beijing, China, 1997.

[17] ISO 10303-21:1994; Industrial automation systems and integration – Product data representation and exchange - Part 21: Clear Text Encoding of the Exchange Structure; ISO; Geneva, Switzerland; 1994.

[18] Kramer, T., Huang, H., Messina, E., Proctor, F., Scott, H., "A Feature-Based Inspection and Machining System," Computer-Aided Design, August 2001.

[19] Lenat, D. B., Guha, R., Pittman, K., Pratt, D., Shepherd, M., CYC: Toward Programs with Common Sense," Communications of the ACM, 33(8):30:49, August 1990.

[20] Maes, P., and Brooks, R., "Learning to Coordinate Behaviors." Proc. 1990 AAAI, Boston, 1990, p.796.

[21] Messina, E., Horst, J., Kramer, T., Huang, H.M., Tsai, T.M., Amatucci, E., "A Knowledge-Based Inspection Workstation," Proceedings of the IEEE Intn'l. Conference on Intelligence, Information, and Systems, Bethesda, MD., Oct. 31-Nov. 3, 1999.

[22] Moravec, H.P., and Elfes, A., "High Resolution Maps from Wide Angle Sonar," Proceedings of the 1985 IEEE ICRA, St. Louis, MO, 1985.

[23] Muñoz-Avila, H., Aha,D., Nau,D. Weber, R.,Breslow, L., Yaman, F., "SIN: Integrating Case-Based Reasoning with Task Decomposition," Proceedings of the 2001 IJCAI, August, 2001.

[24] Oskard, D., Hong, T., Shaffer, C., "Real-Time Algorithms and Data Structrues for Underwater Mapping," Proceedings of the SPIE Advances in Intelligent Robotics Systems Conference, Boston, MA, November 10-11, 1988.

[25] Thorpe, C., et. al, "Vision and Navigation for the Carnegie Mellow NavLab, IEEE PAMI, 10, 3, May 1988.

[26] Volpe, R., T. Estlin, S. Laubach, C. Olson, J. Balaram, "Enhanced Mars Rover Navigation Techniques." Proceedings of the IEEE International Conference on Robotics and Automation, San Francisco CA, April 24-28 2000.

[27] Wasson, G., Kortenkamp, D., and Huber, E., "Integrating Active Perception with an Autonomous Robot Architecture," Robotics and Automation Journal , Vol. 29, pp. 175-186, 1999.

[28] Zah, M., Model-Aided Stability Control on Machine Tools, Proceedings of the 2001 IEEE/ASME International Conference on Advanced Intelligent Mechantronics, Como, Italy, July, 2001.