Journal of Intelligent & Fuzzy Systems 14 (2003) 137–147
IOS Press

Knowledge engineering for real time intelligent control

Elena R. Messina*, James S. Albus, Craig I. Schlenoff and John Evans
Intelligent Systems Division, National Institute of Standards and Technology, 100 Bureau Drive, Stop 8230, Gaithersburg, MD 20899-8230, USA
Tel.: +1 301 975 3510; Fax: +1 301 990 9688; E-mail: {elena.messina, james.albus, craig.schlenoff}@nist.gov, john.evans@snet.net

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.

1. 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 [3]. 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, capturing, and using 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. In this paper, we argue that intelligent control requires several *different* classes of knowledge and representation.

In Section 2, we provide a categorization in which to classify different types of knowledge, and describe how knowledge representations in each class have been previously applied towards intelligent control. In Section 3, we describe efforts that attempt to use only single classes of knowledge, as described in Section 2, and the shortcomings that were encountered by these efforts. In Section 4, we explore integration consideration when trying to use inherently different classes of knowledge towards a unified world view. In Section 5, we discuss the implication of using a reference model architecture in guiding the decisions regarding what type of knowledge is needed in the software and how it should be represented. We conclude in Section 6.

2. 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 [2, 3]. Example labels for three of the levels (subsystem, primitive, and servo), as defined in [3] are shown. This paper presents an overview of the data needed for the world model hierarchy. We argue that there are

*Corresponding author.

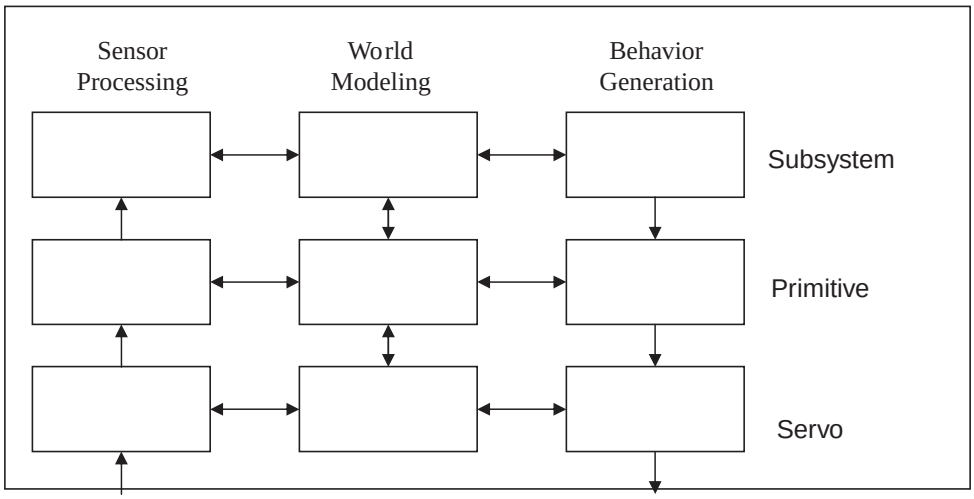

Fig. 1. General framework for an intelligent control system.

three distinctly different classes of knowledge in such a control hierarchy: sensory signals, state variables, and system parameters at the lower levels; spatial models (maps, images and objects) that represent geometric and dynamic knowledge at the middle levels; and symbolic data that represent mathematical, logical, linguistic, and procedural knowledge at the highest levels. Relationships between and among these three types of representations can be expressed as pointers. 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, and numeric information are not addressed by knowledge engineering. We believe that it is necessary to consider these types of representations 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 database, 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, tasks or behaviors (verbs). Modifiers include attributes of things (adjectives) and attributes of tasks (adverbs). 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 [8,17,20]. 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.

2.1. 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 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

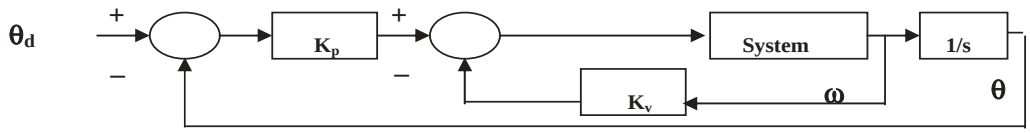

Fig. 2. PD servo control.

instabilities, stiction, and friction are best compensated for at this level.

Learning or adaptive control systems [5,36] 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. We would not expect a robot to invent itself a torque loop in the field, although it could well change the gain or phase of a position or velocity loop as it learns to optimize a task.

2.2. Spatial 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", "metrical maps", or "patterns". This knowledge is 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 or real number 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 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.

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

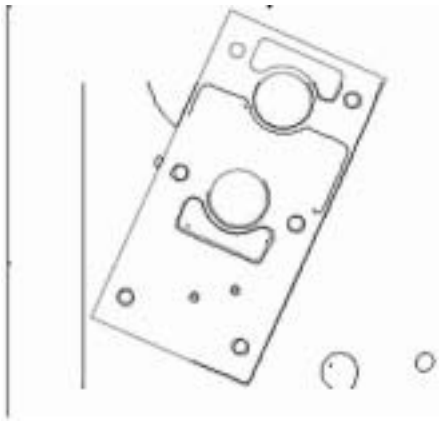

Fig. 3. Part pose computation.

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].

Figure 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) [16, 26].

For mobile autonomous robots, there are two main categories of spatial knowledge representation that are useful. These are sometimes referred to as metrical maps in the literature [21]. One captures what the sensors see (the view "out the windshield"). This may be two-dimensional images, as is the case for CCD cameras, or three-dimensional images, in the case of range sensors such as LADARs. Some mobile robots successfully accomplish their goals by planning based on a world model derived purely from the sensor image view. This is particularly true for road-following systems, such as those by Dickmanns [11] and Jochem [19].

Another spatial representation is akin to the "bird's-eye-view". Digital maps are a natural way of repre-

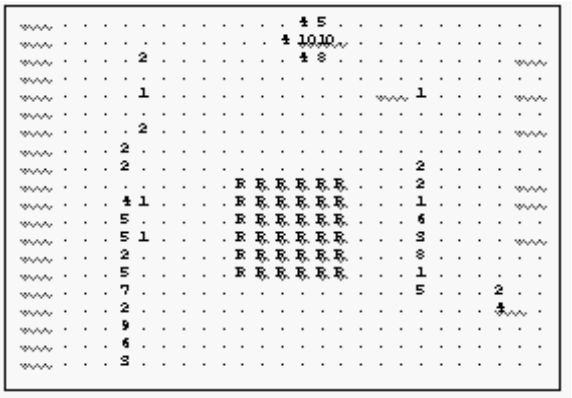

Fig. 4. Occupancy grid map for mobile robot navigating in a hallway and approaching an obstacle.

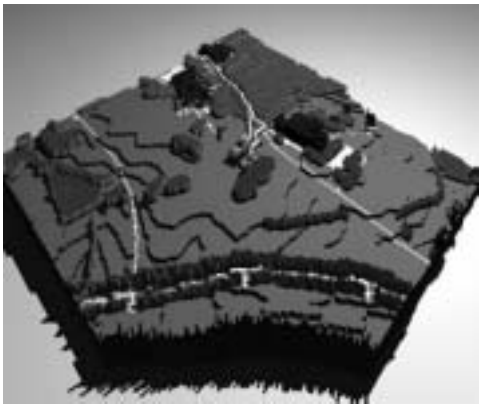

Fig. 5. Multi-featured digital terrain map.

```
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,(),(),(),$,(),
(#110),(),());
#110 = MATERIAL('aluminum','soft
aluminum',$,(),());
```

Fig. 6. STEP representation of a block.

senting 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 [6,9,27,30]. This is particularly useful with sensor modalities that are noisy or sensitive over wide angles such as sonar. Some systems [21] augment versions of these maps with topological information. This enables them to reduce the amount of data stored and relate individual local maps together into a more global one.

Figure 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. Figure 5 shows a higher level map for path planning for outdoor navigation. This map contains several feature layers, including elevation, vegetation, roads, buildings and obstacles.

2.3. 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 [10].

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 [18]. Note that this linguistic representation can be

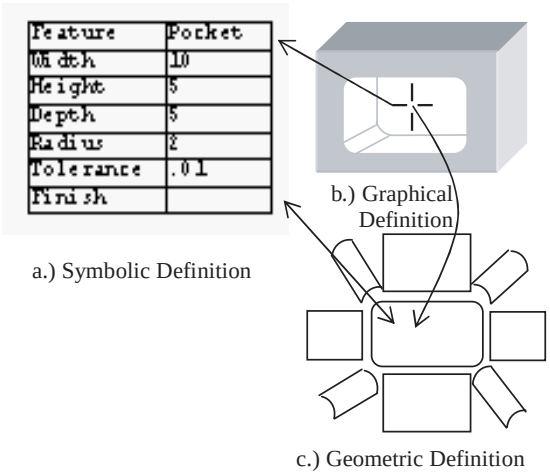

Fig. 7. Pocket Feature.

linked by pointers to 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.

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

In addition to capturing properties of objects, symbolic representations also provide a mechanism to capture rules that can govern the behavior of a system. Figure 8 shows a graphical depiction of a finite state machine that provides a high-level description of the rules that an autonomous vehicle must follow when navigating through traffic. This type of symbolic representation focuses on the states and transitions that are important to this type of behavior, and captures the rules that would cause the vehicle to transition from one state to another. For example, the vehicle would move from state "traversing" to state "exiting traffic" only when the condition "at destination" is met.

Another type of symbolic representation for representing rules is ontological. 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 [23] is an example of the latter, an effort to create a system capable of common sense, natural language understanding, and machine learning.

Ontologies provide mechanisms for reasoning over information. This includes being able to infer information that may not be explicitly represented, as well as the ability to pose questions to the knowledge base and receive answers in return. One way of enabling this functionality is to represent the symbolic information in the world model in a logic-based, computer-interpretable format, such as in the Knowledge Interface Format (KIF) representation [15].

Through the use of an inference engine or theorem prover, information represented in this format could be queried, and logically-proven answers could be returned. As an example, a manufacturer may want to know whether a given set of fixture positions is suitable to fully inspect a part. Assuming that the necessary inspection points, access volumes, and machine capabilities are represented in KIF, the manufacturer could enter in the fixture positions and the system could logically-prove whether those positions are sufficient to fully inspect the part. Future work will be exploring this area in more detail via the implementation of logic-based ontologies to represent the symbolic information in the control hierarchy.

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 geometric) 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.

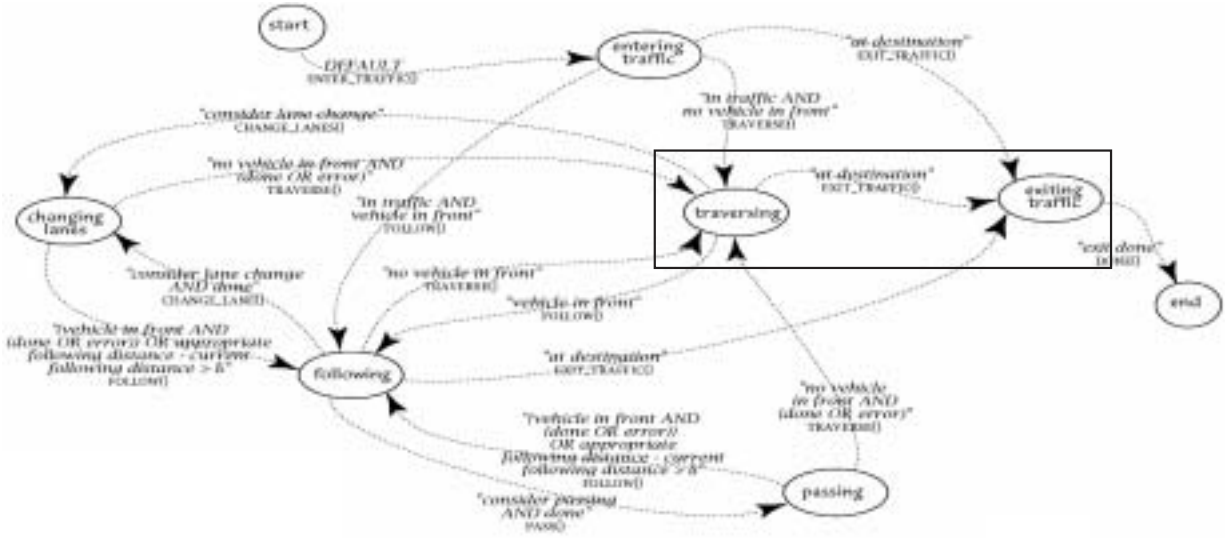

Fig. 8. Finite state machine representation of a driving activity.

Of course, the geometric descriptions are necessary in order to avoid collisions, but that would be handled by a lower control level.

A number of methodologies and tools exist for analyzing and modeling knowledge at a symbolic level. CommonKADS [1] is one such tool which supports structured knowledge engineering, by enabling one to spot opportunities and bottlenecks in how organizations develop, distribute and apply their knowledge resources, and so gives tools for corporate knowledge management.

3. Different approaches to using knowledge for control

3.1. Control using only high-level symbolic knowledge

Much early robot work was carried out in the context of AI research using symbolic representations [22,29, 31]. This had the unfortunate result of uncoupling robotics from the geometry and dynamics of the real world, and focusing on purely symbolic approaches to perception, planning, and reasoning [13].

After struggling for the better part of two decades, the AI community turned away from robotics to expert systems, knowledge representations, and problem solving in the symbolic domain. 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 [34,35].

3.2. 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 [7].

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 [25]. However, Brooks and others 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 Albus [3], Arkin [4] or Thorpe [33] have attacked more complex problems.

3.3. Control with multiple levels of knowledge

Intelligent systems with multiple levels and types of representations are in the minority.

Kuipers and others have elaborated the Semantic Spatial Hierarchy (SSH), which is inspired by human

cognitive modeling. The SSH [21] contains both qualitative and quantitative representations in a hierarchy. Sensor and control level information is based on various types of control laws leading to locally distinctive states. Local geometric maps with their individual frames of reference are constructed at the control level. Above this is a causal level, which derives discrete models of action from the control level. A topological level contains an ontology of places, paths, and regions, which connects the various local metrical maps into a patchwork, which can be merged into a single global frame of reference.

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 [11] 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 sparse way points on a low resolution map by an Army scout. The XUV used its on-board sensors to create high definition, multiresolution maps of its environment and then navigated successfully through very difficult terrain.

This is basically a demonstration of the use of multiresolutional 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.

4. Integration considerations

Representing multiple classes of knowledge within an intelligent control system introduces the challenge of integrating fundamentally different representations into a single, unified knowledge base. This knowledge base must behave as a single, cohesive entity, and as such, there must be seamless information exchange and interoperability between all knowledge sources. In the case of autonomous mobility, as alluded to in Section 2, parametric knowledge may be stored as a set of numbers in a computer program representing the values of the state variables, the iconic knowledge may be a set of digital terrain maps represented as two-dimensional arrays, and the symbolic knowledge may be a set of tokens with pertinent attributes stored in a database.

4.1. Integration within a single representation

There are integration challenges within a single representation, as well as among disparate representations. Building off of the autonomous mobility example above, within solely the symbolic level, one must integrate a priori information about the types of entities that one expects to see in the environment with instances of the entities that are identified with on-board sensors as they are encountered. When both of these pieces of information are represented in database format, association of database keys is often sufficient enough to provide the necessary integration.

Within solely the iconic level, one must integrate processed sensed data about the environment with a priori terrain maps. This is a difficult challenge due to the noise associated with sensed data as well as the varying level of resolution between a priori maps and the sensed data. In addition, one must integrate two or more sensed images, which may be taken by two different sensors, or by the same sensor at different times. Often described as "data registration", researchers are actively addressing this challenge [12,24].

4.2. Integration among disparate representations

Similar challenges exist when integrating knowledge captured in different representations. Although the representations differ, there will undoubtedly be direct correlations between the data in each representation. In the case of object recognition [32], information that can be inferred by analyzing the data stored in the iconic grid structure must be compared to the class attributes stored in the symbolic knowledge base to determine if there is a correspondence. For example, if a cluster of occupied cells in a spatial representation can be grouped into a single object, one can create an object frame and link all the pixels in the spatial representation to the object frame. This object frame contains a list of object attributes that are measured properties of the cluster of pixels in the spatial representation. Depending on the information that is stored in the spatial representation, one may be able to tell the object's dimensions, average color, velocity, location, etc. Based

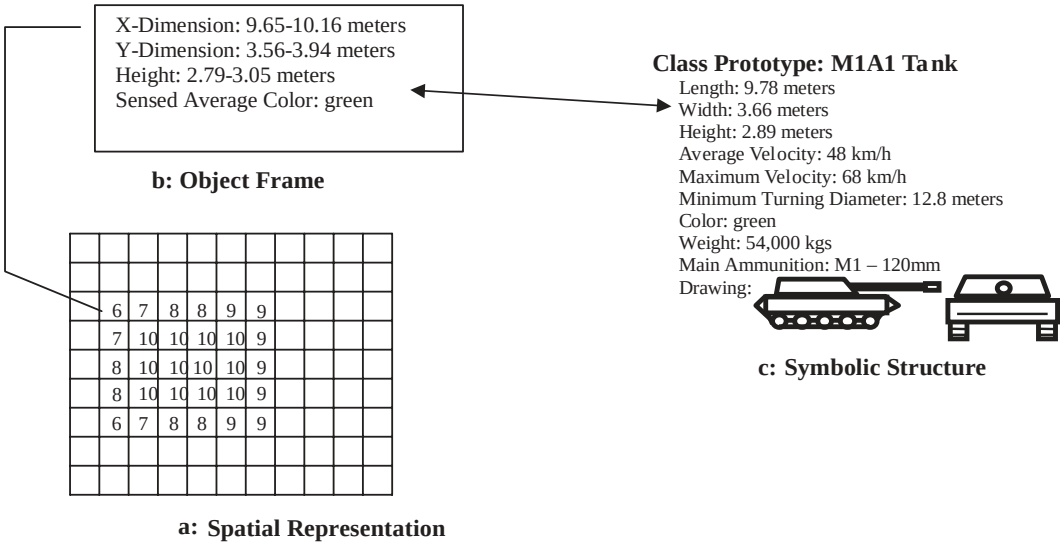

Fig. 9. Integrating a symbolic representation with a spatial representation.

on this information, one can compare the attributes of an observed object to attributes of a class prototype of objects that are expected to be seen in the environment. If a correspondence is found (within a desired threshold), links are established between the object frame and the class prototype in the database. This is the process of classification. Links established through the classification process are bi-directional pointers. Thus, class names and class attributes can be linked back to the object frame, and from there back to the pixels in the spatial representation.

Figure 9 shows an example of integrating a spatial representation with a symbolic representation. In Fig. 9a, the number in the cells represent the probability that the cell is occupied, with 10 being the greatest. Other information is stored in each cell that is not shown in Fig. 9a, such as the color and the height of the object that is occupying that cell. In Fig. 9b, the information in the spatial representation is processed and stored as a list of attributes in an object frame. This involves clustering cells that appear to be part of the same object, and determining overall characteristics of that object. The cluster of cells have an overall X-dimension of between 9.65–10.16 meters, an overall Y-dimension of 3.56–3.94 meters, an average height of 2.79–3.05 meters, and an average color of green. The perceived attributes are then compared to a priori attributes stored in a list of class prototypes as shown in Fig. 9c to determine if there is correspondence. In this case, there appears to be a clear match between the observed attributes measured from the sensed data and the attributes in the class prototype of a M1A1 tank. Therefore, links are created between the class prototype and the cells in the spatial representation.

Although the above scenario is an oversimplified example, it shows the steps that need to be accomplished to establish a link between stored class prototypes and objects observed in the world. These links would ground the symbolic representations in the world model to the objects in the world.

The above approach can be extended to deal with moving objects. To deal with moving objects, one must continually track the motion of the object, and update the pointers from spatial representation to object frame and class prototypes as time progresses. This process can be facilitated by including velocity in the list of object attributes.

5. Considerations in design of knowledge and its representation

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, 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 processes that perform Sensory Processing (SP), Behavior Generation (BG), World Modeling (WM), and Value Judgment (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 to feedback to modify planned actions so that goals are accomplished despite unexpected events. In the sensory processing side of the hierarchy, information derived from observations by subordinate levels is filtered and processed upward to more abstract levels, using a priori knowledge of objects and situations to interpret the incoming data in detecting events, recognizing objects, and developing situation awareness. The sensory processing results are used to update the world model at each level; planning is thus carried out against the best possible representation of the external world.

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. 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 Primitive level, each RCS Node receives commands with goal points about 500 ms in the future. The primitive level computes dynamic trajectories expressed in terms of vehicle heading, speed, and acceleration and sends commands to the servo level about every 50 ms.

At the Subsystem level, an application-specific RCS node for autonomous mobility 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, potential 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.

At the Vehicle level, all subsystems on an individual vehicle are coordinated. These may include mobility, communication, weapons, and reconnaissance subsystems. Maps extend to 500 meters, with resolution of about 4 meters. Plans extend to a time horizon of about one minute into the future, and may be recomputed every 5 seconds.

Higher still in the hierarchy is the Section Level. This is the controller for a group of 2 or more of individual 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 2–5 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 type of information is largely symbolic in nature and may be amenable to rule or case-based reasoning tools (such as [28]). At the Section Level, a Value Judgment function may 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 km/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.

At every level (except the servo level), map-based (graph search) planning and symbolic reasoning tools may be used. At the higher levels, symbolic knowledge will become more important, but at all levels, map-based knowledge will be useful for planning and decision making.

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., digital terrain maps and descriptions of enemy vehicles and capabilities.

6. Conclusion

No one type of knowledge representation is adequate for all purposes. Davis [10] 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, which is the essence of intelligent control of mobility systems. The Army XUV program provides a leading edge demonstration of the value of this approach.

References

[1] CommonKADS General Information, http://www.commonkads.uva.nl/frameset-commonkads.html, 2003.
[2] J. Albus, 4-D/RCS: A Reference Model Architecture for Demo III, *NISTIR 5994*, Gaithersburg, MD, 1997.
[3] J. Albus, R. Lumia, J. Fiala and A. Wavering, NASREM: The NASA/NBS Standard Reference Model for Telerobot Control System Architecture, *Proceedings of the 20th International Symposium on Industrial Robots*, Tokyo, Japan, 1989.
[4] R. Arkin, Navigational Path Planning for a Vision-Based Mobile Robot, *Robotica* **7** (2003), 49–63.
[5] K. Astrom and B. Wittenmark, *Adaptive Control*, Addison-Wesley 1995.
[6] J. Borenstein and Y. Koren, Real Time Obstacle Avoidance for Fast Mobile Robots in Cluttered Environments, *Proceedings, 1990 IEEE ICRA*, Cincinnati, OH, 1990.
[7] R.A. Brooks, A Robust Layered Control System for a Mobile Robot, MIT AI Lab, A. I. Memo 864, Sept. 1985.
[8] S. Brooks and R. Greenway, Using STEP to integrate design features with manufacturing features, *Computers in Engineering Conference*, New York, NY, 1995, pp. 579–586.
[9] J.L. Crowley, Navigation for an Intelligent Mobile Robot, *IEEE Journal of Robotics and Automation* **RA-1**(1) (1985), 31–41.
[10] R. Davis, What is in a Knowledge Representation? *AI Magazine* (1993).
[11] E.D. Dickmanns, A General Dynamic Vision Architecture for UGV and UAV, *Journal of Applied Intelligence* **2** (1992), 251.
[12] M.D. Elstrom, P.W. Smith and A. Abidi, Stereo-Based Registration of LADAR and Color Imagery, *Proceedings of the SPIE Conference on Intelligent Robots and Computer Vision XVII*, Boston, MA, 1998.
[13] D. Etherington, What Does Knowledge Representation Have to Say to Artificial Intelligence? *Proceesings ate the AAAI*, 1997.

[14] J. Evans and B. Krishnamurthy, HelpMate, the trackless Robotic Courier: A Perspective on the Development of a Commercial Autonomous Mobile Robot, *Autonomous Robotic Systems* (1998), 182.
[15] M. Genesereth and R. Fikes, Knowledge Interchange Format, *Stanford Logic Report Logic-92-1*, Stanford University, 1992.
[16] J. Horst, A Lexical Analogy to Feature Matching and Pose Estimation, *NISTIR 6790*, Gaithersburg, MD, 2002.
[17] L. Huagno, J. Cuiyun and H. Jianan, A Knowledge-Based Approach for Object Classification for Assembly, *Proc. IEEE International Conference on Intelligent Processing Systems*, Beijing, China, 1997.
[18] ISO 10303-21, Industrial automation systems and integration – Product data representation and exchange – Part 21: Clear Text Encoding of the Exchange Structure, Geneva, Switzerland, 1994.
[19] T. Jochem and D. Pomerleau, Vision-Based Neural Network Road and Intersection Detection, *Intelligent Unmanned Ground Vehicles* (1997).
[20] T. Kramer, H. Huang, E. Messina, F. Proctor and H. Scott, A Feature-Based Inspection and Machining System, *Computer Aided Design* (2001).
[21] B. Kuipers, The Spatial Semantic Hierarchy, *Artificial Intelligence* **119**(1–2) (2000), 191–233.
[22] J.E. Laird, A. Newell and P.S. Rosenbloom, Soar: An Architectue for General Intelligence, *Artificial Intelligence* **33** (1987), 1–64.
[23] D. Lenat, R. Guha, K. Pittman, D. Pratt and M. Shephard, CYC: Toward Programs with Common Sense, *Communications of the ACM* **33**(8) (1990), 30–49.
[24] R. Madhavan, G. Dissanayake, H. Durrant-Whyte, J. Robetrs, P. Corke and J. Cunningham, Issues in Autonomous Navigation of Underground Vehicles, *Journal of Mineral Resources Engineering* **8**(3) (1999), 313–324.
[25] P. Maes and R. Brooks, Learning to Coordinate Behaviors, *Proceedings AAAI*, 1990, pp. 796–802.
[26] E. Messina, J. Horst, T. Kramer, H.M. Huang, T. Tsai and E. Amatucci, A Knowledge-Based Inspection Workstation, *Proceedings of the 1999 IEEE International Conference on Information, Intelligence, and Systems*, 1999.
[27] H.P. Moravec and A. Elfes, High Resolution Maps from Wide Angle Sonar, *Proceedings of the 1985 IEEE ICRA*, St. Louis, MO, 1985.
[28] H. Munoz-Avila, D. Aha, D. Nau, R. Weber, L. Breslow and F. Yaman, SIN: Integrating Case-Based Reasoning with Task Decomposition, *Proceedings of the 2001 IJCAI*, 2001.
[29] A. Newell and H. Simon, *GPS; A Program that Simulates Human Thought*, McGraw-Hill, 1963.
[30] D. Oskard, T. Hong and C. Shaffer, Real-Time Algorithms and Data Structrues for Underwater Mapping, *Proceedings of the SPIE Advances in Intelligent Robotics Systems Conference*, Boston, MA, 1988.
[31] J.D. Pearson, S.B. Huffman, M.B. Willis, J.E. Laird and R.M. Jones, A Symbolic Solution to Intelligent Real-Time Control, *Robotics and Autonomous Systems* **11** (1993), 279–291.
[32] C. Schlenoff, Linking Sensed Images to an Ontology of Obstacles to Aid in Autonomous Driving, *Proceedings of the 18th National Conference on Artificial Intelligence: Workshop on Ontologies for the Semantic Web*, 2002.
[33] C. Thorpe, Vision and Navigation for the Carnegie Mellow NavLab, *IEEE PAMI* **10**(3) (1988).
[34] R. Volpe, T. Estlin, S. Laubach, C. Olson and J. Balaram, Enhanced Mars Rover Navigation Techniques, *Proceedings of the IEEE International Conference on Robotics and Automation*, San Francisco, CA, 2000.
[35] G. Wasson, D. Kortenkamp and E. Huber, Integrating Active Perception with an Autonomous Robot Architecture, *Robotics and Automation Journal* **29** (1999), 175–186.
[36] M. Zah, Model-Aided Stability Control on Machine Tools, *Proceedings of the 2001 IEEE/ASME International Conference on Advanced Intelligent Mechantronics*, Como, Italy, 2001.