Fusing Disparate Information Within the 4D/RCS Architecture

Craig Schlenoff, Raj Madhavan, Jim Albus, Elena Messina, Tony Barbera, Stephen Balakirsky
Intelligent Systems Division
National Institute of Standards and Technology
Gaithersburg, MD USA.
{craig.schlenoff, raj.madhavan, james.albus, elena.messina, tony.barbera, stephen}@nist.gov

Abstract - In this paper, we show how the 4D/RCS (Real-time Control System) architecture incorporates and integrates multiple types of disparate information into a common, unifying framework. 4D/RCS is based on the supposition that different information modeling techniques offer different advantages. 4D/RCS allows for the capability to capture information in formalisms and at levels of abstraction that are suitable for the way that they are expected to be used.

In the context of applying the 4D/RCS architecture to achieving the ultimate goal of the control of autonomous ground vehicle navigation, we describe the procedural and declarative types of information that have been developed and their respective values towards achieving the goal. Experimental results of information fusion within the 4D/RCS architecture are presented from the Demo III eXperimental Unmanned Vehicles (XUVs) in an extended series of demonstrations and field tests.

Keywords: 4D/RCS architecture, information representation and fusion, autonomous ground vehicle navigation

1 Introduction

The control of complicated automated robots has made us recognize the need for a diverse set of information which needs to be represented and reasoned over. Information ranges from parameters for equations, to metric spatial information, to declarative and procedural symbolic knowledge. Not only does this information need to be represented, but it also needs to be integrated into a common architecture that allows for fusion into a synergistic information source which can be seamlessly accessed and used.

This paper describes the 4D/RCS (Real-time Control System) and how it is being applied to the fusion of numerous information sources for autonomous ground vehicle navigation. In Section 2, we provide a brief background of 4D/RCS. In Section 3, we describe other efforts that have attempted to integrate different information formalisms, often in an attempt to mimic human intelligence, and describe how 4D/RCS is different. In Section 4, we describe some of the different types of procedural and declarative types of information represented in 4D/RCS. In Section 5, we describe some results and we conclude the paper in Section 6.

2 Background of 4D/RCS

4D/RCS evolved from the bottom up as a real-time intelligent control system for real machines operating on real objects in the real world. The first version of RCS was developed as a sensory-interactive goal-directed controller for a laboratory robot. The fundamental element is the control loop with a goal, a transition function, a feedback loop, and an action output such as a force, velocity, or position. Over the years, RCS has evolved into an intelligent controller for industrial robots, machine tools, intelligent manufacturing systems, automated general mail facilities, automated stamp distribution systems, automated mining equipment, unmanned underwater vehicles, and unmanned ground vehicles [1,8]. The most recent version of RCS (4D/RCS) embeds elements of Dickmanns [11] 4-D approach to machine vision within the 4D/RCS control architecture. 4D/RCS was designed for the U.S. Army Research Lab AUTONAV and Demo III Experimental Unmanned Vehicle programs and has been adopted by the Army Future Combat System program for Autonomous Navigation Systems [2,3].

4D/RCS consists of a multi-layered multi-resolutional hierarchy of computational nodes each

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

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

3 Related Work

There have been numerous attempts to develop architectures that aid in the integration of disparate information formalisms. One of the earliest was the ACT architecture [4]. ACT grew out of research on human memory. Over the years, ACT has evolved into ACT* and more recently, ACT-R. ACT-R is being used in several research projects in an Advanced Decision Architectures Collaborative Technology Alliance for the U.S. Army. ACT-R is also being used by thousands of schools across the country as an algebra tutor—an instructional system that supports learning-by-doing. Another well-known and widely used architecture is Soar [14]. Soar grew out of research on human problem solving, and has been used for many academic and military research projects in problem solving, language understanding, computational linguistics, theorem proving, and cognitive modeling. The spatial semantic hierarchy models and integrates knowledge for robot exploration and mapping [13]. Polyscheme develops a cognitive substrate that integrates multiple inference and representation abilities that can handle reasoning in most or all domains [10].

Other cognitive architectures include Prodigy, ICARUS, IMPRINT (Improved Performance Research Integration Tool), EPIC (Executive-Process Interactive Control), and 4D/RCS (Real-time Control Systems). Like Soar, Prodigy uses search through a problem space to achieve goals cast as first-order expressions [18]. ICARUS is a reactive architecture that encodes knowledge as reactive skills [22]. IMPRINT is a task description language designed for the Army to capture the procedural specification of tactical behavior scenarios [6]. It contains a dynamic, stochastic, discrete-event network modeling tool designed to help assess the interaction of soldier and system performance throughout the system lifecycle—from concept and design through field testing and system upgrades. IMPRINT has been integrated with ACT-R to model military behaviors [5]. EPIC is an architecture that models the detailed timing of human perceptual, cognitive, and motor activity, including the input/output characteristics of the nervous system connecting the higher level cognitive functions to the external world [12]. 4D/RCS is a control system architecture inspired by a theory of cerebellar function. 4D/RCS models the brain as a hierarchy of goal-directed sensory-interactive intelligent control processes that theoretically could be implemented by neural nets, finite state automata, cost-guided search, or production rules.

4D/RCS is similar to other cognitive architectures in that it represents procedural information in terms of production rules, and represents declarative information in abstract data structures such as frames, classes, and semantic nets. 4D/RCS differs from other cognitive architectures in that it also includes signals, images, and maps in its knowledge database, and maintains a tight real-time coupling between metric and symbolic data structures in its world model. 4D/RCS is also different in:
a) its focus on task decomposition as the fundamental organizing principle; b) its level of specificity in the assignment of duties and responsibilities to agents and units in the behavior generating hierarchy; and c) its emphasis on controlling real machines in real-world environments.

4 Enabling Intelligence in Autonomous Systems

4.1 The Types of Information Represented in 4D/RCS

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

As mentioned previously, 4D/RCS is a hierarchical architecture, and as such, supports information representation at different levels of abstraction. Traditionally, the lowest levels of the architecture
Layered Terrain Maps can be expected to evolve in the future under a variety of circumstances.

Figure 2 shows the many different types of information representation formalisms that are currently being implemented within the 4D/RCS architecture as applied to autonomous driving. These formalisms range from metric to symbolic and from procedural to declarative. Information is captured in formalisms and at levels of abstraction that are suitable for the way that it is expected to be used. Different information representation techniques offer different advantages, and 4D/RCS is designed in such a way as to combine the strengths of all of these techniques into a common unifying architecture in order to exploit the advantages of each. Real-time aspects are another dimension of knowledge representation which is not discussed here but can be found in [17]. In the following subsections, we will describe some of the formalisms depicted, classifying information as either procedural or declarative.

4.2 Procedural Information

Procedural information is the knowledge of how to perform tasks. Procedural information is different from other kinds of information, such as declarative information, in that it can be directly applied to a task. Within 4D/RCS, procedural information is primarily used for planning and control purposes. As such, we will describe two planning approaches that are currently being implemented in 4D/RCS and describe the information that underlies each.

Both planning approaches start with the same 4D/RCS methodology for determining the information needed by the world model to support the behavior processes within each node. In the KD includes the system’s best estimate of the current state of the world plus parameters that define how the world state can be expected to evolve in the future under a variety of circumstances.
that needs to be represented to accomplish the planning task. The methodology starts as follows:

1. The first step involves an intensive analysis of domain knowledge from manuals and subject matter experts, especially using scenarios of particular subtask operations. The output of the effort is a structuring of this information into a task decision tree consisting of simpler and simpler commands (actions/verbs) at simpler and simpler levels of task description.

2. The second step defines the hierarchical organization of agent control modules that will execute these layers of commands in such a manner as to reasonably accomplish the tasks. This is the same as coming up with a business or military organizational structure of agent control modules (people, soldiers) to accomplish the desired tasks. This step forces a more formal structuring of all of the subtask activities and responsibilities, as well as defining the execution structure.

At this point, the two approaches diverge in the procedure for determining the types of information necessary to accomplish the planning task. Subsequent steps are described in the following subsections: (a) represents the subsequent steps in the state machine planning approach and (b) represents the subsequent steps in the cost-based planning approach.

### 4.2.1 State Machine Based Planning (a)

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

3a. The third step clarifies the processing of each agent's input command through the use of rules to identify all of the task branching conditions with their corresponding output commands. Each of these command decompositions at each agent control module will be represented in the form of a state-table of ordered production rules (which is an implementation of an extended finite state machine (FSM)). The sequence of simpler output commands required to accomplish the input command and the named situations (branching conditions) that transition the state-table to the next output command are the primary information represented in this step.

4a. In the fourth step, the named situations that are the task branching conditions (determined in 3a) are defined in great detail in terms of their dependencies on world and task states. This step attempts to define the detailed precursor states of the world that cause a particular situation to be true.

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

6a. The sixth step uses the context of the particular task activities to establish the distances and, therefore, the resolutions at which the above objects and entities must be measured and recognized by the sensory processing component. This step establishes a set of requirements and/or specifications for the sensor system at the level of each separate subtask activity.

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

### 4.2.2 Cost-Based Planning (b)

The cost based methodology concentrates on decomposing each of its assigned tasks into an optimal sequence of commands that will be assigned to its subordinates. This is accomplished through the incremental creation and evaluation of a planning graph [7]. Once again, the first two steps from section 3.2 must be performed and are then followed by:

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

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

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

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

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

4.3 Declarative Information

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

4.3.1 Symbolic Information

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

4.3.1.1 Ontologies

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

The first effort involves the use of ontologies as a mechanism to allow the planner in the behavior generation 4D/RCS component to better understand the costs and consequences of colliding with other objects. By representing the factors that could impact a path’s cost, an ontology can be used to reason over the information that is available to determine what the consequences of a collision would be. Further reasoning could then be performed to determine the cost of these consequences. This cost would then be fed back to the planner for consideration when deciding the “cheapest” plan for the system to execute.

The system is composed of an ontology of objects representing “things” that may be encountered in our current environment, in conjunction with rules for estimating the damage that would be incurred by collisions with the different objects in different situations. A series of tests were performed in a simulation environment with different variations of vehicles and obstacles. In all cases, the vehicle behaved appropriately, depending on the predicted damage that would be incurred by either taking evasive maneuvers to avoid the obstacles or driving through the obstacles when damage was not considered significant. More information about this effort can be found in [19].

The second effort is an ontology to model tactical behaviors, which is based upon the OWL-S specification (Web Ontology Language- Services) [23]. This is a joint effort currently being performed by DCS Corporation and NIST to develop an intelligent ground vehicle (IGV) ontology using Protégé. The goal of this effort is to develop a common, implementation-independent, extendable knowledge source for researchers and developers in the intelligent vehicle community that will:

- Provide a standard set of domain concepts along with their attributes and inter-relations;
- Allow for knowledge capture and reuse;
- Facilitate systems specification, design, and integration, and;
- Accelerate research in the field.

In this context, behaviors are actions that an autonomous vehicle is expected to perform when confronted with a predefined situation. The ontology is stored within the 4D/RCS knowledge database, and the behaviors will be spawned when situations in the world are determined to be true, as judged by sensor information and the value judgment components. The information which is modeled in the IGV Ontology is based upon the output of the 4D/RCS methodology described in Section 4.2.

At the time when this document was written, there were 489 classes, 213 properties (attributes), and 2674 instances modeled in the ontology. More information about this effort can be found in [21].

4.3.1.2 Databases

In addition to ontologies, databases have been developed to house symbolic information. The database that has received the most attention to date is the Road Network Database. For an autonomous vehicle to be able

1 Note that the databases described in this section are different than the knowledge databases described earlier.
to navigate a road network, it must be aware of and must respond appropriately to any object it encounters. This includes other vehicles, pedestrians, debris, construction, accidents, emergency vehicles, ... and it also includes the roadway itself. The road network must be described in such a way that an autonomous vehicle knows, with great precision and accuracy, where the road lies, rules dictating the traversal of intersections, lane markings, road barriers, road surface characteristics, and other relevant information.

The database includes detailed information about the roadway, such as where the road lies, rules dictating the traversal of intersections, lane markings, road barriers, road surface characteristics, etc. The purpose of the Road Network Database is to provide the data structures necessary to capture all of the information necessary about road networks so that a planner or control system on an autonomous vehicle can plan routes along the roadway at any level of abstraction. At one extreme, the database provides structures to represent information so that a low-level planner can develop detailed trajectories to navigate a vehicle over the span of a few meters. At the other extreme, the database provides structures to represent information so that a high-level planner can plan a course across a country. Each level of planning requires data at different levels of abstraction, and the Road Network Database must accommodate these requirements.

The database has been developed in a hierarchical fashion so as to support planning at various levels. Included in the database are information about:

- **Roads:** a stretch of travel lanes in which the name of the travel lanes does not change.
- **Road Segments:** a uni-directional stretch of roadway bounded by intersections.
- **Road Elements:** a uni-directional stretch of roadway bounded by any type of junction.
- **Lane Clusters:** a set of uni-directional lanes (with respect to flow of traffic) in which no physical attribute of those lanes change over the span of the lane.
- **Lane:** a single pathway of travel that is bounded by explicit or implicit lane marking
- **Lane Segment:** either straight line or constant curvature arcs.
- **Junctions:** two or more paths of transportation that come together or diverge, or a controlled point in a roadway
- **Intersections:** a type of junction in which two or more separate roads come together.

This database has been applied at research efforts both with NIST and Boeing, in both simulated and real environments. More information about the Road Network Database can be found in [20].

### 4.3.2 Metric Information

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

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

Within 4D/RCS, maps enhance the scope of the world model. Such metric maps may take a variety of forms including survey and aerial maps and may provide significant information about existing topology and structures. The higher levels in the 4D/RCS control hierarchy include feature and elevation data from *a priori* digital terrain maps such as information about roads, bridges, streams, woods, and buildings. This information needs to be registered and merged with data from the lower level maps that are generated by sensors. Additionally, for incorporating *a priori* information into the world model, some form of weighting is required and this depends on how well the *a priori* data and the sensed information are registered. There is also the need to generate higher resolution *a priori* terrain maps as the current survey maps are too coarse for autonomous driving. Another potential application for registering sensor data is the computation of ground truth.

Towards registering LADAR (Laser Range Detection) range images to *a priori* maps, we have developed an iterative algorithm that can deal with false/spurious matches, occlusions and outliers for UGV (unmanned ground vehicle) navigation [16]. A hybrid iterative algorithm has also been developed for registering 3D LADAR range images obtained from unmanned aerial and ground vehicles [15]. Combined with a feature-based approach, the algorithm was shown to produce accurate registration for the two sets of LADAR data. Registration of the UGV LADAR to the aerial survey map minimizes the dependency on GPS for position estimation especially when the GPS estimates are unreliable or unavailable.
5 Results

Apart from the individual, isolated results included with each of the information formalisms described in Section 4, some of these information representation approaches were implemented in an overall experiment. Although all of the above mentioned representations were not available at the time that the experiment was run, the experiment did involve the fusion of multiple knowledge representation formalisms to achieve the accomplishments described below.

Experimental validation of the 4D/RCS architecture and the information representation within was provided by the performance of the Demo III eXperimental Unmanned ground Vehicles (XUVs) (Figure 3) in an extended series of demonstrations and field tests during the winter of 2002-2003. Although, in these demonstration, it was difficult to know the exact effect of each type of representation on the overall outcome, it is clear that the success of the demonstrations was closely tied to the information that was represented and the ability to fuse this information together into a common information source.

The XUVs were equipped with an inertial reference system, a commercial grade GPS receiver (accurate to about +/- 20 m), a LADAR camera with a frame rate of 10 frames per second, and a variety of internal sensors. Routes for XUV missions were laid out on a terrain map by trained Army scouts, and given to the XUVs in terms of GPS waypoints spaced over 50 m apart.

The XUVs operated completely autonomously until they got into trouble and called for help. Typical reasons for calling for help were the XUV was unable to proceed because of some terrain condition or obstacle (such as soft sand on a steep slope, or dense woods), and was unable to find an acceptable path plan after several attempts at backing up and heading in a different direction. At such a point, an operator was called in to teleoperate the vehicle out of difficulty.

During three major experiments designed to determine the technology readiness of autonomous driving, the Demo III experimental unmanned vehicles were driven 550 km, over rough terrain: 1) in the desert; 2) in the woods, through rolling fields of weeds and tall grass, and on dirt roads and trails; and 3) through an urban environment with narrow streets cluttered with parked cars, dumpsters, culverts, telephone poles, and mannequins. Tests were conducted under various conditions including night, day, clear weather, rain, and falling snow. The unmanned vehicles operated over 90% of both time and distance without any operator assistance. An extensive report of these experiments has been published [9], along with high resolution ground truth data describing the terrain where the XUVs experienced difficulties [24].

6 Conclusions and Future Work

We believe that the 4D/RCS architecture provides an excellent framework in which to integrate multiple information representation approaches to build cognitive models and intelligent systems that significantly advance the level of intelligence we can achieve. In this paper, we have described how 4D/RCS supports multiple types of representations, ranging from metric to symbolic and from declarative to procedural, and provided brief examples of how each of these representations are used in the context of autonomous driving. We also showed how all of these information representation formalisms not only fit into the node structure present at each level of the 4D/RCS hierarchy, but also played a role in the 4D/RCS methodologies.

Although this paper does not address every type of information formalism that exists, nor, for that matter, every type of information formalism that is being integrated in to the 4D/RCS architecture, it does show the robustness of using this architecture for information representation and fusion. Future efforts will continue to incorporate different information formalisms into 4D/RCS as well as extend the work that has already been described in this document.

References


