World Modeling and Behavior Generation for Autonomous Ground Vehicles

Stephen Balakirsky

Intelligent Systems Division
NIST

Alberto Lacaze

Electrical Engineering
University of Maryland

Abstract

In this paper we define an architecture for coarse vehicle motion planning within the Real-time Control System (RCS) hierarchy. Interaction between the planning system and world model will be discussed, and results of implementing this architecture on a real-world system will be presented.

1 Introduction

A long-standing goal has been to create a reference model architecture for intelligent controllers. The Real-time Control System (RCS) reference model architecture is one such architecture and it has been applied successfully to multiple diverse systems [1, 2]. The target systems for RCS are, in general, complex control problems. It has been shown [3] that the complexity of a control problem is reduced by the use of a hierarchical control system. In order to take advantage of this fact, RCS provides guidelines for the decomposition of the problem into sets of hierarchical control nodes. The decomposition into the hierarchy is guided by control theory that takes into account such items as system response times and planning horizons.

RCS has been further specialized into the application specific 4-D/RCS that is aimed at the design and implementation of control systems for intelligent, autonomous vehicles. The "4-D" portion of the name refers to the integration of the VaMoRs [4] approach to dynamic machine vision.

Each level of the 4-D/RCS hierarchy follows a "sense-model-act" paradigm and includes sensory processing (SP), world modeling (WM), value judgment (VJ), and behavior generation (BG) as depicted in Figure 1 [5]. In this paradigm, SP functions filter and extract information to feed into WM. WM maintains this information in a Knowledge Database (KD) over areas defined by the planning horizon of the level. The KD includes symbols and data structures containing information about entities, events, self, and knowledge of how the world behaves. In addition, WM may provide simulation facilities to estimate the state of the world at the present or some future time. The BG module utilizes these facilities in cooperation with the VJ to compute possible plans or courses of action. The role of the VJ module in the planning process is to compute costs, risks, and benefits of these courses of action. Finally, the BG module makes decisions or "acts" based on input from the VJ module, in cooperation with WM, and transmits these decisions to the next lower level of the hierarchy.

The 4-D/RCS hierarchy contains seven levels, each of which is modeled after the generic level depicted in Figure 1. The levels range in planning scope from "servo control" planning for the moves of an individual actuator, to "battalion control" planning for the movement of a large group of vehicles. Each level is designed to function in a particular spatial and temporal scope. For each level, the temporal scope is based on the response time required for control of the vehicle and the spatial scope is based on the required planning horizon. As one moves higher up the hierarchy (towards battalion control), the temporal and spatial scope increase while the resolution decreases. By using this scheme 4-D/RCS strives to maintain a constant level of complexity throughout the hierarchy. All of the levels of the hierarchy have similar components, however, the rest of this paper will concentrate on the vehicle level.

2 World Model

The vehicle level world modeling (VLWM) is responsible for maintaining a representation of the outside world and the vehicle self at a resolution that

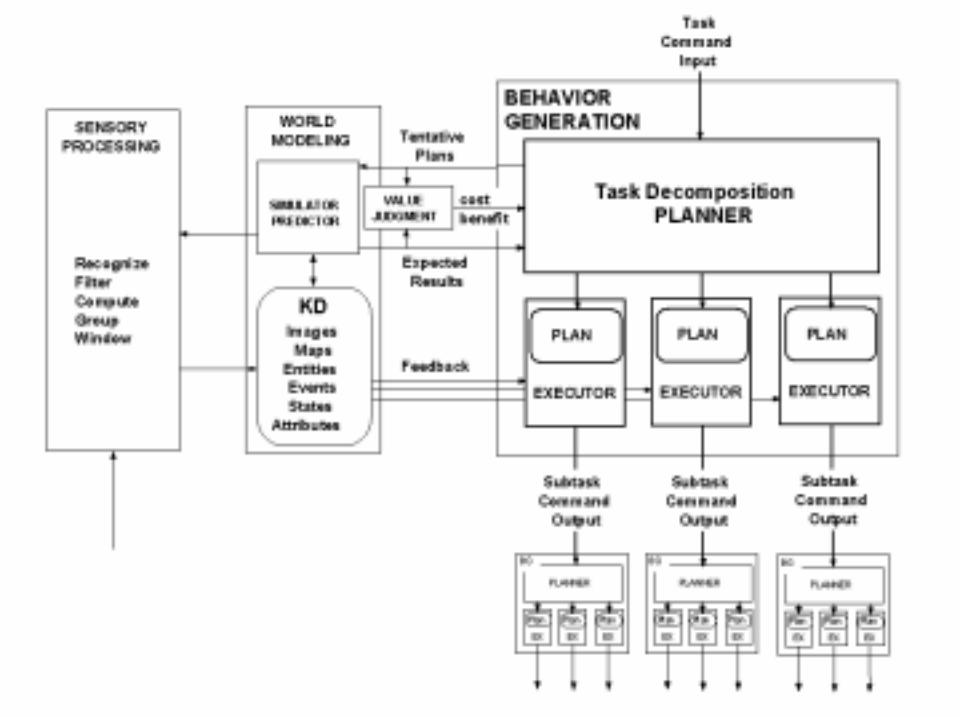

Figure 1: A typical 4-D/RCS computational level.

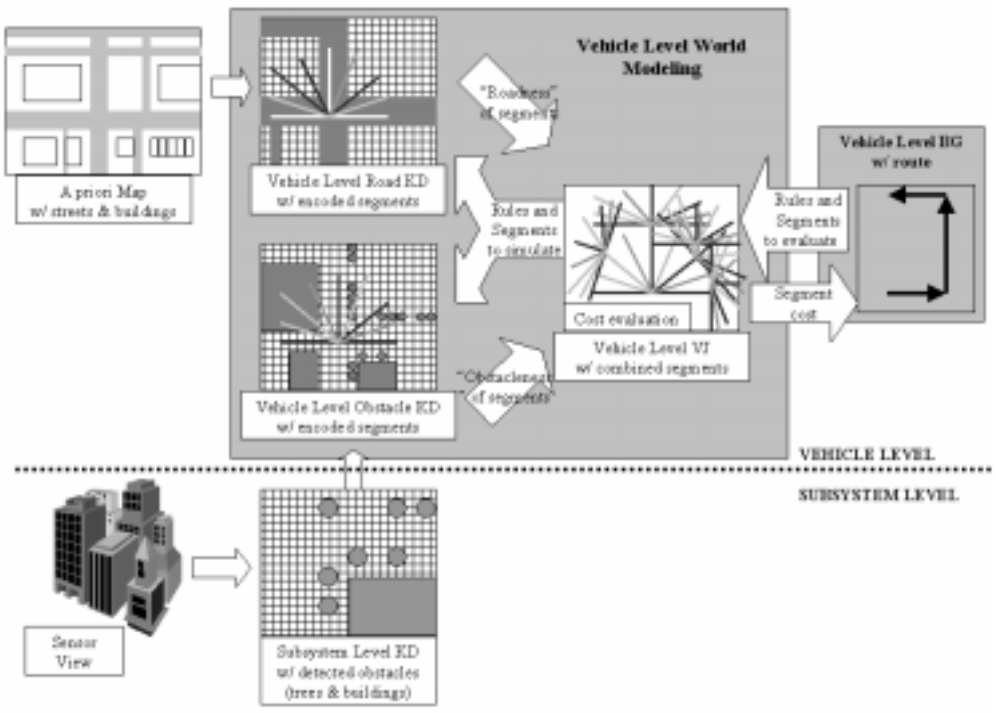

Figure 2: Simplified WM component layout.

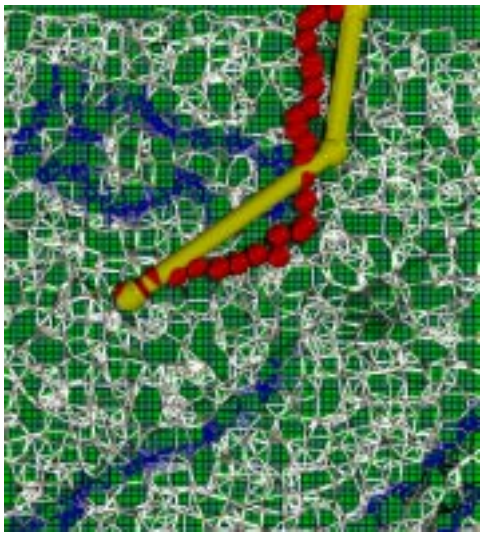

Figure 3: Vehicle level graph.

allows for the control of deliberative/reactive gross vehicle movements. The Vehicle Level (VL) has a subordinate level that performs similar tasks, but at a much finer resolution that includes such things as vehicle dynamics. BG tasks the VLWM to evaluate the cost/benefit of tentative plans. These tentative plans are passed into VLWM in the form of "plan segments", where many segments make up a complete vehicle level plan. VLWM simulates these plan segments and returns to BG a single value per segment that represents the cost/benefit. As shown in Figure 1, VLWM may be broken down into three components: the Vehicle Level Knowledge Database (VLKD), the Vehicle Level Simulator/Predictor (VLS/P), and the Vehicle Level Value Judgment (VLVJ) modules.

The VLKD receives information from lower levels through the use of sensor processing and information from higher levels through filters. The actual control and low-level processing of raw, high-resolution, sensor data are performed at the level subordinate to the vehicle level (the subsystem level). The vehicle level receives the subsystem level's SP representation, and performs processing to convert this data into a resolution and form appropriate for the vehicle level. For example, high-resolution obstacle data will be converted into mobility corridor estimates. Information received from higher levels includes such items as a priori map information and designated constraints on individual vehicles. This information is filtered, and the relevant information for this particular vehicle is stored in the VLKD.

The VLKD may be viewed as containing many "overlays", each of which has knowledge about a particular attribute or feature in the world. In an effort to reduce off-processor communication bandwidth and latency between the VLS/P and VLKD, the VLKD has been designed as a modular system with individual overlays residing with individual sensor processing modules. This has the additional advantage of allowing for distributed computing.

The VLS/P utilizes the information from the VLKD in cooperation with VLVJ to produce the cost map that is used by BG. The VLS/P must first discretize the plan segment into steps that match the resolution of the vehicle level. The plan segment is then simulated so that information about the vehicle self may be produced. The VLS/P combines this predicted information about the vehicle self (e.g. vehicle speed, heading, etc.) with known constraints from higher levels (e.g. avoid roads and populated areas) to form queries about individual plan segments into the VLKD. These queries are sent to the individual VLKD overlays, and the results are collated for processing by the VLVJ module.

3 Value judgment

The VLVJ is a rule-based module that evaluates the cost/benefit of plan segments based on information provided by the VLS/P and the supervisor. The VLVJ has two distinct types of rules: those that apply to individual VLKD overlays to form intermediate results and those that apply to the combination of these intermediate results to form a single final cost/benefit to be passed back to the BG. The rules that apply to

the individual VLKD overlays utilize modifiers such as "contains", "near", and "far", and information about the vehicle self and constraints to produce a conformity measure to the rule. For example, a mobility rule that would be sent to the "roads" overlay of the VLKD may state to only allow driving on roads (contains roads). Constraints from the upper levels may include items such as obey traffic laws and avoid major highways. Predicted information about the vehicle self would include information on predicted vehicle speed and heading. All of this information would be combined to evaluate each step of the plan segment. The step would be said to "conform" if it met the rule and all of the constraints, and would be said to "not conform" if it violated the rule or any constraint (for example driving the wrong way on a one-way road). The conform/not conform decision need not be binary. The rule may contain a conformance curve to be applied to the feature as it strays from the desired value. For example, a rule may be to drive as fast as possible. A plan step with a vehicle speed equal to the speed limit and a plan step with a speed slightly below the speed limit would both conform. However, the higher speed step would conform "more" and receive a higher conformity value based on the conformance curve. Conformance values of the individual steps of the segment are then combined into a single conformance value. This may be accomplished in many ways, including using the minimum, maximum, or average value.

The second class of rules is used to combine the intermediate conformance results into a single, final, cost/benefit value for the plan segment. These rules allow for complex behaviors to be performed by the vehicle. The rules may take the form of binary arithmetic over rules (e.g. rule 1 conforms and rule 2 conforms or rule 3 conforms) or may be a weighted combination of conformance values (e.g. conformance to rule 1 is very important and conformance to rule 2 is slightly important). Each plan segment is evaluated by these rules and a single cost/benefit value is passed back to BG.

4 Behavior Generator

The function of BG at every level of the 4-D/RCS hierarchy is the same: to create ordered time tagged sets of actions to be performed by the subordinate levels and to execute these actions.

In order to perform its function, BG at a level must:

- receive commands from the supervisor level BG. These commands contain actions and goals for the level
- interpret the set of constraints and cost evaluation functions relevant to the assigned command
- create sets of alternative behaviors that may achieve the goal
- evaluate these behaviors based on predicted outcomes
- select the best behavior given the supervisor's constraints and cost evaluation criteria and the time allocated to find it
- execute these behaviors by sending them to the subordinates in a common language

In order to accomplish these tasks, BG has at its disposition VJ and WM. From the point of view of BG, the purposes of WM and VJ are to give an accurate representation of the state of the world and provide a comparison between alternate behaviors so that it can select the best one. This may include simulation and prediction of the changes in the world caused by the generated behavior. The source, accuracy, and freshness of the information in WM are irrelevant to BG except in the manner that they influence the result of the evaluation of cost for the requested behavior.

BG can be divided functionally into the Planner (PL) and Executor (EX). The planner's primary function is creating alternative behaviors. Since intelligent systems are generally highly dimensional, there may exist a large or infinite number of alternative behaviors. Generally, the evaluation of these alternatives is the computational bottleneck of the level. In order to avoid this bottleneck, the planner must carefully select the alternatives to include satisfactory solutions but not overburden the WM. To constrain the number of alternatives to be created and evaluated, the PL makes use of the constraints available. These constraints may come from different sources, which include:

- goal assigned by the supervisor, which can be represented as regions in the state space
- current state of the world and the self represented in the WM
- characteristics and capabilities of the subordinates (or resources)
- computational timing constraints set by the level's cycle

The creation (or pruning) of the alternatives varies depending upon which of these constraints is valid for the particular application. In some applications, certain constraints are more important than others, and therefore they may create separate functional modules for the determination of these alternatives. For example, resource allocation and scheduling can be considered the pruning of alternatives that do not follow resource and timing constraints respectively.

The executor takes the result of PL and feeds it to the subordinates. The subordinates for the level may be other levels or actual hardware actuators in the highest resolution levels. The executor can be viewed as an interface between the two levels and generally is composed of classical control techniques to guarantee stability between levels.

In the interest of consistency and modularity, RCS reuses modules at different levels whenever possible. Of course, in actual implementation, these modules may be instantiated in different hardware, and they may work with different levels of resolution. The VLBG is no exception, and similar planners may be shared at different levels. Specifically, in VLBG the cycle of execution for mobility purposes is as follows:

1. A command is received from the vehicle level's superior (the section level), which comes as an order to move along a particular path. The section level may be coordinating different vehicles therefore it may send allowed corridors to traverse, or sets of zones to stay away from.

2. Along with this command there may be a cost evaluation function (or reference to one) so that the VLBG knows how to compare behaviors. Example include how much risk to accept, or a desired completion time.

3. The VLBG sends the cost evaluation function to VLWM.

4. The VLBG requests from the VLWM special interest states (SIS) given the current goal and command. An example of these may include roads, bridges, known observation posts, etc.

5. The VLBG creates a set of possible actions and sends them to the VLWM and VLVJ. In our case these actions are sets of movements where the initial and final state are assigned. They may include SIS but the set of actions is not restricted to them. Internally to VLPL and unknown to the VLWM and VLVJ, these actions are connected into graphs that allow the planner to concatenate actions and costs. In cases where costs cannot be concatenated, complete paths through these graphs can be sent. In our application, placing points in a grid-like fashion over the space of interest and randomly shifting them creates these candidate actions. This procedure allows the inclusion of SIS and has the benefit that pseudo randomly distributed points over featureless areas avoids the common symmetry problems associated with grids.

6. These lists of actions are passed to VLWM and VLVJ. The VLWM first refines the cost evaluation function passed from the VLBG. For example, the amount of risk to take will be combined with a priori knowledge of the probability of enemy contact and inter-visibility lines and will result in a behavior being generated. In this case it may be staying close to trees while avoiding roads. The VLWM then evaluates the cost of being in the initial state, traversing from the initial state to the final state, and being at the final state by using the simulator and VLVJ. The costs evaluated are then sent back to the VLBG.

7. VLPL compares and strings groups of actions to create trajectories. In our case, the optimal trajectory within the given graph is found.

8. VLPL sends the trajectories to VLEX to execute, and the re-planning cycle starts again.

5 Current Implementation

The current implementation of the 4-D/RCS vehicle level was developed for the Office of the Secretary of Defense's Robotics Demo III program. This program will culminate in late summer of 2001 with a user appraisal involving four robotic vehicles performing complex, autonomous behaviors. These behaviors will mimic activities performed by a scout platoon and will include both individual vehicle behaviors (cross-country driving at 9 m/s, road following at 18 m/s, etc.) and group behaviors (bounding over-watch, cooperative target tracking, etc.).

As may be seen from Figure 2, the current VLWM consists of three independent modules. The first two modules, the vehicle level road KD and vehicle level obstacle KD, are both based on a generic KD/Simulation module. To customize the generic module for a given KD overlay, an input filter or sensor processing routine and a simulation routine must be written. These modules accept simulation rules and segments to simulate as their input, and send back a measure of how well the segments conformed to the

rules. For example, if darker segments show higher conformity, the rules "avoid obstacles" and "avoid roads" would generate the evaluation shown in Figure 2.

The third module shown is the VJ module. This module acts as the communications front-end for the entire system. It receives segments and rules from the BG and passes the required information onto the KD overlays. It then receives the results of the simulation and creates a final cost to be passed back to the BG module based on rules for combining the output of the individual KD overlays.

The current implementation of the VLBG is based upon a graph search technique derived from Dijkstra search. The VLBG creates a graph where each node corresponds to a state that the vehicle may visit. These nodes are a combination of randomly thrown points and special interest states like roads and bridges. An example of such graph can be seen in Figure 3.

In Figure 3, the car is located at the center of the map. The size of the map is about 500 meters on a side. The grid-like background shows the internal representation of the VLWM, and the light colored segments are possible actions where the vehicle will move from one end of the segment to the other. The dark colored segments represent segments of very high cost as evaluated by the VLVJ. The reason for the high cost in this case is that they cross through walls, fences, or forest. The VLWM representation (underlying grid) is composed of 15,625 cells while the graph to be searched is only composed of around 2000 nodes, so the computational advantages are evident. In this case, the path found by the VLBG can be seen as a set of marbles driving NW. The distance between the nodes in the found trajectory is under 50 meters, therefore the subordinate level can find the accurate (and dynamically correct) trajectory within its map, and avoid obstacles that cannot be seen in the coarse representation of this level. The lighter colored trajectory represents the path sent by the supervisor level.

6 Conclusions

Our system has advantages over more traditional grid-based planning systems in both cost map generation and plan generation. Traditional techniques calculate cost maps in a grid fashion. In these cases, directionality and time of arrival at a location are not available at the time of the calculation of cost. Therefore the cost functions where the directionality or time is important cannot be evaluated. Our cost map generation has the advantage of working on path segments instead of grid cells. VJ and WM have at their disposal the complete initial and final state description for an action. This gives us the ability to have an accurate picture of our vehicle self at each step of the segment. The vehicle self includes things such as velocity and direction of travel, and time of arrival at a location. This knowledge may be of fundamental importance in the cost evaluation. A simple example would be that the WM has the knowledge that an area will be exposed to artillery during a certain time window, but perfectly safe at other times. Because the complete descriptions of initial and final states are available to the VJ, the modification and development of cost evaluation can be performed in an efficient manner.

The planning system has the advantage of replanning cycles that are computationally inexpensive. The evaluation of cost of future movements, in general, is more computationally expensive than the graph search associated with selecting the path. Our VLBG only re-evaluates the cost of traversal where new information is received (most likely inside the sensor range). Because the number of nodes in the graph is small with respect to the number of cells in the grid of the underlying map, it is easy to see that the search performed has computational advantages over grid or field techniques. In addition, within the graph created by SIS and random points the path found is optimal. In our case, the number of nodes in the search graph does not need to be large because the subordinate level will decide the fine movement. Thus optimal search technique can be applied.

References

[1] J.S. Albus. *Brain, Behavior, and Robotics.* McGraw-Hill, 1981.

[2] J. Albus. Outline for a theory of intelligence. *IEEE Transactions on Systems, Man, and Cybernetics*, 21:473–509, 1991.

[3] J. Albus, A. Meystel, and A. Lacaze. Multiresolutional planning with minimum complexity. *Intelligent System and Semiotics*, 97.

[4] E. Dickmanns. The seeing passenger car VaMoRs-P. *International Symposium on Intelligent Vehicles*, 1994.

[5] J. Albus. 4-D/RCS reference model architecture for unmanned ground vehicles. In *SPIE*, volume 3693, Orlando, Fl., 1999.