Proceedings of the SPIE 16th Annual International Symposium on Aerospace/Defense Sensing, Simulation and Controls, Orlando, FL, April 1 - 5, 2002.

Value-Driven Behavior Generation for an Autonomous Mobile Ground Robot

Stephen Balakirsky* and Alberto Lacaze
Intelligent Systems Division
National Institute of Standards and Technology
Gaithersburg, MD 20899

ABSTRACT

In this paper, we will describe a value-driven graph search technique that is capable of generating a rich variety of single and multiple vehicle behaviors. The generation of behaviors depends on cost and benefit computations that may involve terrain characteristics, line of sight to enemy positions, and cost, benefit, and risk of traveling on roads. Depending on mission priorities and cost values, real-time planners can autonomously build appropriate behaviors on the fly that include road following, cross-country movement, stealthily movement, formation keeping, and bounding overwatch. This system follows NIST's 4D/RCS architecture, and a discussion of the world model, value judgment, and behavior generation components is provided. In addition, techniques for collapsing a multidimensional model space into a cost space and planning graph constraints are discussed. The work described in this paper has been performed under the Army Research Laboratory's Robotics Demo III program.

Keywords: Autonomous vehicle, path planning, 4D/RCS, hierarchical world model, Demo III, graph search, cost map.

1. INTRODUCTION

The autonomous vehicle planning system being developed for the Army Research Laboratory's Robotics Demonstration III[14] is designed to allow vehicles (both individual and groups) to move across the battlefield in an intelligent "military" fashion. This means that moving from point 'a' to point 'b' in a safe manner (from a vehicle roll-over or damage perspective) may not be good enough. In addition to reaching point 'b', the vehicle may need to perform specific military behaviors that are governed by rules from military doctrine such as formation maintenance or over-watch, while following other behaviors that are governed by showing a preference to avoid or seek out certain terrain features (roads, population centers, etc.) such as stealthy movement. In order to have our vehicle's behavior achieve this mix of rules and preferences, we have implemented a system that mixes a rule-base with value-driven cost evaluation to perform behavior generation.

To simplify our planning system's implementation, we have based our system on the National Institute of Standards and Technology's 4D-Real-time Control System (4D/RCS) architecture [1]. This is a hierarchical architecture that divides each level of the hierarchy into subsystem nodes, and further subdivides each node into the areas of perception, world modeling, value judgment, and behavior generation. This paper will address the *vehicle* and *section* levels of hierarchy for the *mobility* subsystem and will pay particular attention to the world modeling (WM), value judgment (VJ), and behavior systems (BG).

The remainder of this paper is organized as follows: Section 2 presents a discussion of prior work that is related to our current research; Section 3 details the theory of operation of our system, and Section 4 provides several examples. Finally, Section 5 provides conclusions and a discussion of areas for further work.

2. PRIOR WORK

In order to elicit intelligent behaviors from our mobile robot system, our planner uses a graph based planning paradigm to combine intelligent graph formation and search techniques, a flexible cost evaluation structure, and a rich world

* Authors' email addresses: firstname.lastname@nist.gov.

model. A large body of prior research exists in the area of graph based planning. For gross motion planning[1], this research includes topological systems that generally have a sparse graph of connected distinct situations, places, or landmarks [12,16,17] and cell based graphs that include more densely distributed nodes through the use of data structures such as grids [7,10] and cell trees [15,18]. In addition, our system has built upon domain independent graph based systems that formulate graph nodes and connections based on rule-bases [2].

The goal of the planning system is to travel from the node that contains the current location, to the node that contains the goal location while minimizing some cost function. This cost function produces a measure through which the "goodness" of plans may be compared. The actual function used is closely tied to the type and quantity of information that is maintained by a system. Many systems refer to this stored data as the system's world model. Systems in the literature include a wide range of approaches for ranking plan goodness. Some systems only maintain information on the validity of actions. These systems, as represented by the Graphplan system by Blum and Furst [2] and the FF system by Hoffman [9], attempt to minimize plan length by assigning a unit cost per valid action. The result is that the shortest plan (in terms of actions performed) is deemed the best plan. Other systems utilize obstacles [3,4,18], *a priori* traversibility [8,13], or a combination of the two [5] and attempt to find the shortest traversable path. Still others incorporate a more complex world model as well as more complex cost evaluation. Systems such as one by Saab and VanPutte [13], take into account such factors as path length, slope, and static obstacles. Sabb and VanPutte's system is able to find the path that expends the least amount of energy (assuming that traveling up and down hills uses more energy then traveling on flat ground).

While the above-mentioned systems produce valid plans for moving from point 'a' to point 'b', they are incapable of producing plans that exhibit a variety of movement techniques and behaviors without significant additional programming. In this paper, we will show how our cost evaluation and world model allow for the simple formulation of behaviors that may be dynamically varied.

3. THEORY OF OPERATION

The function of BG at every level of the 4D/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. Input to BG comes in the form of commands from the supervisory level BG, where the highest level BG is typically a human. These commands contain actions and goals for the system, and objectives that dictate the manner in which these actions are to be carried out and the goals met. The command may also contain constraints that limit the manner in which the actions may be carried out or the way that the planner operates. One example of a planner constraint would be a formation constraint that dictates a required configuration for a group of vehicles during the movement operation.

The BG (or planning system) participates in a decision loop with the world model and value judgment to create and evaluate a planning graph. Once the graph has been evaluated and a final state chain has been chosen, it is sent to subordinate BG modules for execution. It is this state chain that determines the operational behavior of the system.

3.1. Plan Graph Creation

The plan graph that is used by BG contains nodes and edges and is created as a cooperative effort of the BG, WM, and VJ. A section of a simple 2-D planning graph is shown in Figure 1. As shown in the figure, the nodes of the graph represent possible discrete system states. The edges represent the transition that the system must undergo to change from one connected node to the next and the cost associated with the transition. There are many different connection strategies that appear in the standard plan graph search literature. For a 2-D planning graph, these would include 4-way (connect to north, south, east, and west), 8-way (add the diagonals), or *n*-way (connect by some spanning distance criterion). Our system uses a 2-D planning graph for the vehicle level (location of the vehicle in UTM coordinate frame), and a 4-D planning graph (location of two vehicles in UTM coordinate frames) for the section level. A spanning distance criterion in conjunction with a rule-base is utilized in order to connect the nodes via edges. The spanning distance criterion for node connections allows for a smoother final state chain then other connection techniques (this is because the distance criterion allows connections at a wider variety of angles the a set 4- or 8-way connection strategy), while the rule-base

[1] Where gross motion planning may be defined as "being concerned with problems involving free space much wider than the objects' sizes plus the positional error of the robot" [11] p. 220.

implements mandatory constraints on the plan (for example a node to node transition that will not conform to a proper formation will not be connected). It should be noted that each pair of connected nodes is connected by two edges, one for each direction of state transition. The reason behind this is that the cost for node traversal in one direction may be very different then the cost for the same traversal in the opposite direction. This may be seen when one drives down a one-way street (it is much more expensive to drive one way then the other). Once the graph has been constructed, a modified Dijkstra search [6] is performed on the graph in order to find the optimal path through the planning space.

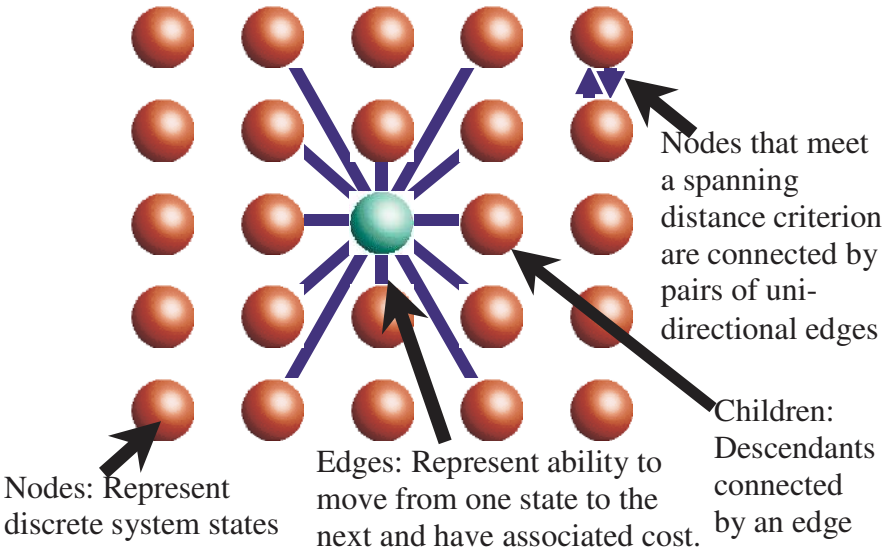


Figure 1: Section of a planning graph.

3.1.1. Vehicle Level Graph

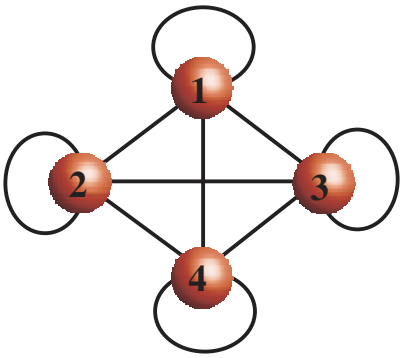
At the time of graph creation (which made be a preset phase, or may happen incrementally during graph exploration), the world model is queried for possible states that should be included as nodes in the planning graph. As outlined in section 3.2, a rule-base is used by the world model and value judgment module to determine a set of states for use as nodes in the planning graph. Depending on the specific objectives of the mission being planned, these states may also include pseudorandom states. For example, a plan that is constrained to drive only on roads would have planning graph nodes that only includes states whose geographic location lies on a road. A plan with a constraint that specifies that roads are "preferred" would include the same set of nodes as the "road only" graph with the addition of pseudorandom nodes distributed over all traversable off-road terrain. In our current implementation, these nodes are distributed in a fixed geographic grid and then perturbed by a random amount in both the northing and easting. This random perturbation has the effect of eliminating symmetries that may exist and providing for smoother final trajectories. The node selection rules may be fixed, or changed on a planning cycle or region basis.

Once the nodes have been selected, the graph must then be connected by edges. This is once again a cooperative process between the BG, WM, and VJ modules of the system. A rule-base is consulted to specifically connect or prevent connection of particular node pairs, and a spanning distance criterion is computed to add additional connections. Rule-base forced connections take place in situations such as vector road maps. In a vector road map, a geographic distance that is quite large relative to the resolution of the BG system may separate adjacent road vertices. Because of this large geographic separation, a spanning distance criterion (such as a distance) would fail to connect the adjacent nodes. In order to assure that adjacent road nodes are connected, rules are put into place to specifically connect them.

3.1.2. Section Level Graph

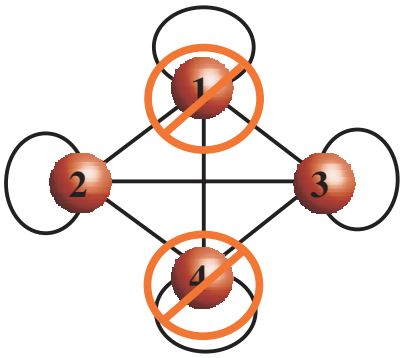
The section level graph is constructed in a similar manner to the one used by the vehicle level. However, instead of each node representing the geographic location of a single vehicle (northing, easting), each node now represents the location of two vehicles (n_1, e_1, n_2, e_2) as shown in Figure 2a. This added complexity of the node representation would lead to an

increase of n^2 in the number of nodes and at least e^2 in the number of edges (depending on the connection strategy used) over the 2-D graph, where n is the number of 2-D nodes and e is the number of edges. In order to reduce this large number of nodes and edges, additional rules are applied based on the particular mission being performed.



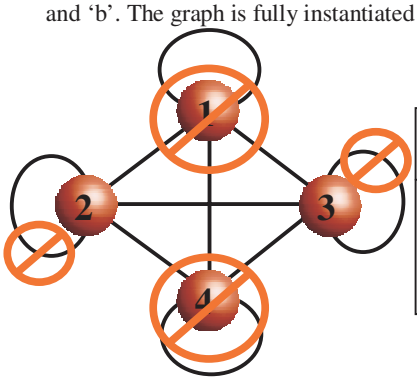
State	Location v1	v2
1	a	a
2	a	b
3	b	a
4	b	b

a) 2 vehicle-planning graph for possible geographic locations 'a' and 'b'. The graph is fully instantiated and fully connected.



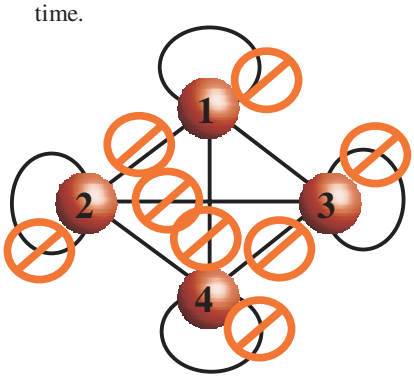
State	Location v1	v2
1	a	a
2	a	b
3	b	a
4	b	b

b) Constraint: two vehicles may not be at same location at same time.



State	Location v1	v2
1	a	a
2	a	b
3	b	a
4	b	b

c) Additional constraint: At least on vehicle must move.



State	Location v1	v2
1	c	a
2	b	a
3	a	c
4	a	b

d) Three locations in graph, additional constraint: Only one vehicle may move at a time.

Figure 2: Examples of constraints on section level graph.

3.2. World Model and Value Judgment

The world model contains the knowledge and tools required by the BG module to formulate valid plans for the vehicle. It contains multiple feature or attribute class specific knowledge layers with each layer consisting of vehicle centered maps (grid or vector based), rule-bases, and a simulation system. The vehicle-centered maps contain information for the computation of graph edge costs and have an extent and resolution that is dictated by the level of the 4D/RCS architecture where they reside. At the vehicle level for Demo III, the maps cover an extent of 1km at a resolution of 4m per grid cell (for the grid based maps), while the section level has maps that extend to 10km at a resolution of 40m per grid cell (for the grid based maps).

The rule-base contains rules that pertain to graph creation, and the rules may apply globally (across time and space), or be limited in their scope. At the time of graph creation, the world model is queried for possible states that should be included as nodes in the planning graph. Constraints and rules are applied at each knowledge layer, and candidate states are passed to value judgment for consideration. Value Judgment then combines these states by applying further global constraints (again, fixed or dynamic), and passes the resulting nodes into BG for graph creation. Since the rules eliminate world states from possible consideration as graph nodes, they may be used to shrink the planning space as well eliminate certain patterns of behavior.

For example, our vehicle level of the mobile robot system contains an *a priori* map that represents features such as roads, buildings, and forested areas. In this system, the world model will receive a request for graph nodes along with

information about the intent of the plan (e.g. this plan is for an on-road vehicle, so plans must be on a road). The *a priori* knowledge layer will search its database for states that conform to the mission specific rules and pass these states as a 2-D grid of quantized locations to VJ. These states may be distributed over the entire ground surface except for areas of dense trees for an off-road system, or may only cover road surfaces for an on-road system. The global rules in the VJ then eliminate any state received from another layer that was not also received from the *a priori* layer.

Another example would be the section level constraint that the two vehicles may not occupy the same location at the same time. As shown in Figure 2b, this single constraint would eliminate a large percentage of the graph nodes. Since there is no longer a node in the planning graph that represents the simultaneous combination of vehicle 1 at location 'a' and vehicle 2 at location 'a', the behavior pattern of having both vehicles at the same location at the same time has been totally eliminated.

Another cooperative operation between BG, WM, and VJ is the procedure of connecting graph nodes with edges. The process is initiated by the BG module, which may connect the entire graph at initialization time, or may incrementally build and connect the graph during the search procedure. In either case, the BG sends a node to the WM for connection. The world model first builds a list of possible connections by applying a spanning distance criterion to the node in question and its neighbors. This spanning distance is used to ensure that close neighboring nodes have the opportunity to be connected. Additional nodes that fail the spanning distance criterion, but are listed in the nodes "must connect" list are also added for consideration. The node's "must connect" list is built and maintained by the world model and contains information from vector databases such as road network connections.

This list of possible connections is then passed to the individual layers of the world model which may veto connection decisions based on their own internal rules. Finally, the VJ module collects the results and returns the node connections to BG. Examples of possible rules are shown in Figure 2c and Figure 2d. In Figure 2c, an edge constraint that requires at least one vehicle to move during each time step is imposed. Figure 2d, demonstrates how a simple version of bounding may be implemented by requiring only a single vehicle to move at any time instance.

```
cost = (  costWeight->offset +
          costWeight->obstacle * detectObs +
          costWeight->unknown * detectUnknown +
          costWeight->road * aRoads +
          costWeight->building * aBuildings *  detectUnknown +
          costWeight->forest * aForest  * detectUnknown +
          costWeight->nearRoad * aNearRoads +
          costWeight->nearBuilding * aNearBuildings +
          costWeight->nearForest * aNearForest +
          costWeight->risk * layerValues[2] +
          costWeight->frontSlopeMax * abs(maxFrontSlope) +
          costWeight->sideSlopeMax * abs(maxSideSlope) +
          costWeight->frontSlopeAvg * avgFrontSlope +
          costWeight->sideSlopeAvg * avgSideSlope +
          costWeight->field * fieldConformance +
          costWeight->pathLength ) * segLength;
```

Figure 3: Typical cost function.

The above formulation depicts how certain behaviors may be entirely eliminated from the vehicles vocabulary through the use of rules. However, we must still have a technique that allows the vehicles to exhibit "preferences" for certain behaviors. For example, the graph depicted in Figure 2d will cause a simple form of bounding. However, to have military utility, it is not good enough to have a vehicle move out into the open and then stop while its partner moves. Instead, we

would like to have the vehicle move to a concealed location from which the partner's route will be visible. Soft constraints or preferences such as these are elicited through the use of our cost function. This cost function is divided between the WM and VJ.

The world model does not compute costs associated with graph edge transitions (this is the job of VJ). It does, however, use a plan simulator to compute "conformance" to knowledge layer specific attributes as well as resource consumption and production. Conformance measures represent basic statistics on graph edges. For example, the *a priori* map layer computes (along with other measures) the maximum and average slope that a particular graph edge encounters and the risk of enemy detection while traversing a particular edge. Resource requirements refer to resources that must be available to traverse a graph edge, or are made available by the edge traversal. An example of this would be a requirement for a road detection algorithm to be running before an off-road to on-road transition is performed.

The results of these simulations are returned to the VJ module for integration into a final cost. This final cost must be formulated in such a way that the system behavior is consistent with the supervisor's commanded goals and objectives. In our current implementation, using logical rules and weights to combine individual results as shown in Figure 3 performs this. For this cost function, a weight vector (the *costWeight* vector) is used to set the importance of the various simulation results. In addition, precedence is given to sensed information over *a priori* information through the use of a logical operator. This can be seen through the use of the Boolean flag *detectUnknown* which causes *a priori* obstacle information to be discounted when it is false (meaning that the sensors have seen this area). Further experimental results are presented in the next section.

4. EXAMPLES

The following examples are taken from code run on our actual system with real data (for the vehicle level examples) and simulated data (for the section level examples). In both cases, the states may be viewed as the physical locations that the vehicle(s) may occupy. The system must plan waypoints, and paths between these waypoints that meet the overall objectives specified by the supervisor.

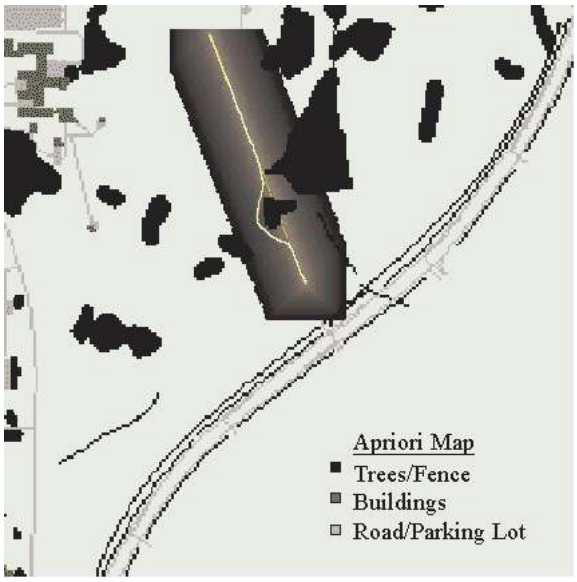
These objectives may be region based (e.g. perform surveillance over a region while not being detected), line based (e.g. move from way-point to way-point in a stealthy or fast or safe manner, and point based (e.g. be at this location by this time). Through a combination of rules and cost weights, the system will design a planning graph that provides these various behaviors.

4.1. Vehicle level

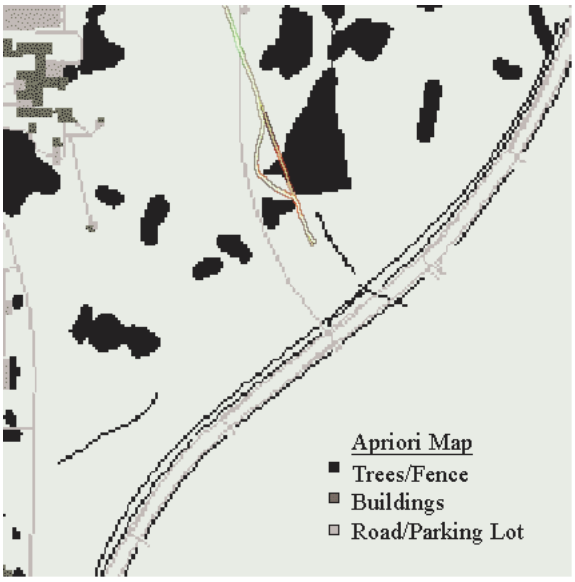
As detailed in previous sections (and see also [10]), the architecture provides for a very detailed, rich, world model (the Knowledge Database (KD) layers) that underlies the Value Judgment (VJ). For the current Demo III system, these KD layers include an *a priori* KD layer, a derived line-of-sight (LOS) KD layer, a derived slope KD layer, a supervisor's constraint KD layer, and a sensed obstacle KD layer.

4.1.1. Database layers

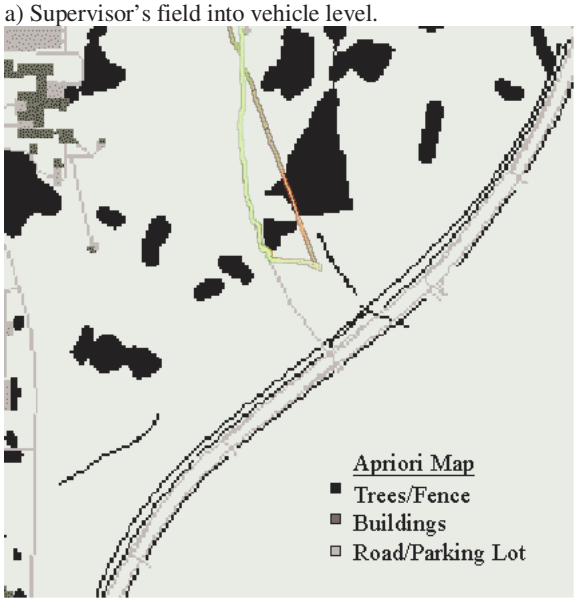
The *a priori* KD layer provides feature information that is available from high-resolution digital map products (currently 10 m per cell for Demo III). The layer's conformance simulator computes the conformance of a trajectory segment to a particular feature (e.g. forest, roads, lakes, buildings, soil types, etc.) as well as the conformance of a trajectory being "close to" a particular feature. In addition, the predicted length of time to traverse the trajectory is computed. This information is then passed to the resource consumption simulator which computes which feature based algorithms must be active for this edge to be traversed. The conformance to a feature and being close to a feature are returned as the percentage of the trajectory that contains the feature, or is close to a feature, respectively. The definition of what constitutes being "close to" a feature is a user-defined parameter.



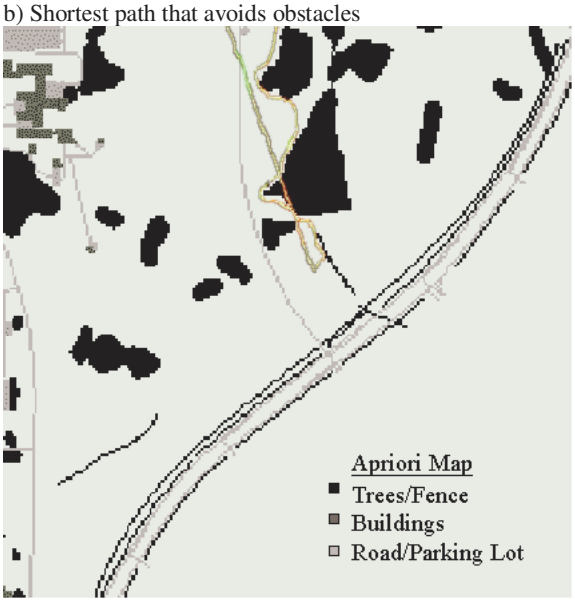
a) Supervisor's field into vehicle level.



b) Shortest path that avoids obstacles



c) Avoid obstacles, and prefer roads.



d) Avoid obstacles, and follow tree-line.

Figure 4: Varied system behaviors are possible through simple changes in the system cost function.

The traversal time is calculated by simulating the vehicle traversing the soil types and features specified in the map data at user defined average speeds for each soil and feature type. Algorithm activation is specified by a rule-base that is implemented as a look-up table.

The Line Of Sight (LOS) KD layer provides an estimate of the risk of detection that the vehicle will incur and an estimate of the areas that the vehicle will be able to survey while traversing a given trajectory. The layer utilizes a LOS calculation in order to compute this value. A LOS calculation is a calculation that determines if an object of a given target height at a given target location may be seen from a given observation height at a given observation location. Our calculation takes into account both ground elevation contours and features (buildings, forests, etc.).

In essence, the layer computes two fields of view (a field of view is a 360º line of sight out to the expected detection range) for each cell that the vehicle is expected to traverse. One field of view is performed with an observation height that matches the vehicles sensor height and a target height of the ground level, and the other with an observation height that matches the vehicles height and a target height of the expected enemy's sensor height. Through the use of these calculations, the KD layer is able to compute the number of locations that will be able to see the predicted vehicle location, and the number of these locations that the vehicle will have been able to see first. This information, coupled with information on known or predicted enemy locations, is used to formulate the risk of being in any particular location. The layer also stores which areas will be visible (and thus surveyed) from the vehicle trajectory. The simulator for this layer computes the average and maximum risk values along the given edge. There is no resource consumption simulation for this layer.

The slope KD layer provides an estimate of the danger of vehicle rollover while traversing the terrain. The layer utilizes a pre-computed database that contains the slope from the current cell to its eight neighbors. Vehicle specific information on acceptable front, back, and side slope is compared to the slope that the vehicle will experience when traversing the given terrain. The conformance of the trajectory to a "safe" trajectory (in terms of a slope safety margin) is returned as the conformance measure.

The slope KD layer allows for the demonstration of the benefit of our WM over traditional grid based cost maps. In a traditional grid based cost map, the cost of traversing through a cell would be based on the worst case slope for that cell and the smallest of the front, back, and side slopes permitted by the vehicle. By simulating the actual trajectory that the vehicle will take, we are able to add directionality into the conformance criteria (thus separating front, back, and side slopes). Therefore, a cell traversal with a high, but acceptable front slope that is allowed by our WM may be ruled out by a traditional cost map because the slope violates the vehicle's lesser side slope constraints. There is no resource simulation for this layer.

Path tolerances are sent to the vehicle level from its supervisor. The supervisor's constraint KD implements these tolerances as a variable width, variable slope field that surrounds the commanded path. By changing the field width, the user can specify how far the vehicle is allowed to stray from the commanded path. By specifying the field slope, the user can specify how rapidly the vehicle should maneuver back to the commanded path. The simulator for this layer computes an average and maximum penalty for each plan segment based on the field strength in the region where the vehicle is traversing. This field may be seen in Figure 4a.

The sensed obstacle KD layer provides information on mobility corridors thru obstacles sensed by the vehicle. The layer utilizes information that flows up from lower levels of the RCS hierarchy. The subordinate layer in the RCS hierarchy sends an occupancy grid to the sensed obstacle KD layer. This grid contains indications of areas that are not traversable by the vehicle, as well as areas that have an unknown traversal value. This layer returns the conformance of a trajectory to a path that is known to be traversable and demonstrates the ability of the WM to utilize information from lower levels of the 4D/RCS hierarchy.

4.1.2. Results

Figure 4b through Figure 4d show samples of plans that have been generated by this system. This data was obtained by using the current vehicle location as the plan start point, and a user defined ending location as the goal. The figures represent three planning cycles where the users goals were changed between each cycle. There were no code changes between the cycles, and the start and goal locations remained static. The only change was to a user provided weight vector for the cost function. The actual cost function used may be seen in Figure 3.

In Figure 4b the weight vector was tuned to find the shortest distance path while avoiding obstacles and conforming to the supervisor's field. As may be seen in the figure, the computed path deviates from the straight-line commanded path to avoid the area of dense trees. The computed path then quickly joins the commanded path as a result of the supervisor's field.

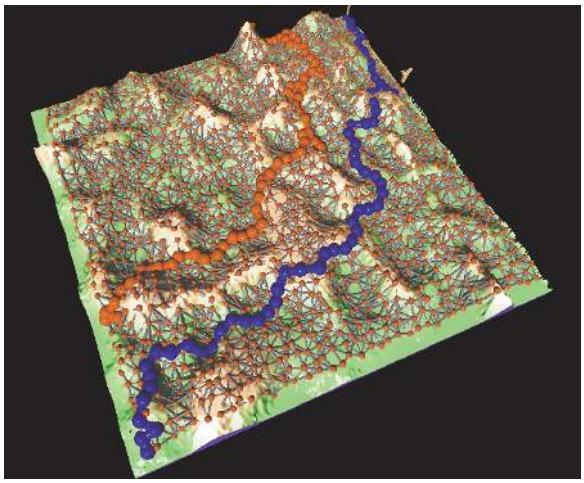
Figure 4c shows the system operating with a weight vector tuned to perform road following. The computed plan turns on the road edge-finding algorithm before the robot is expected to encounter the roadway. Road following is then enabled to

maintain the path along the road. This figure was generated exactly one planning cycle after the previous figure. The only change to the system was the cost weight vector.

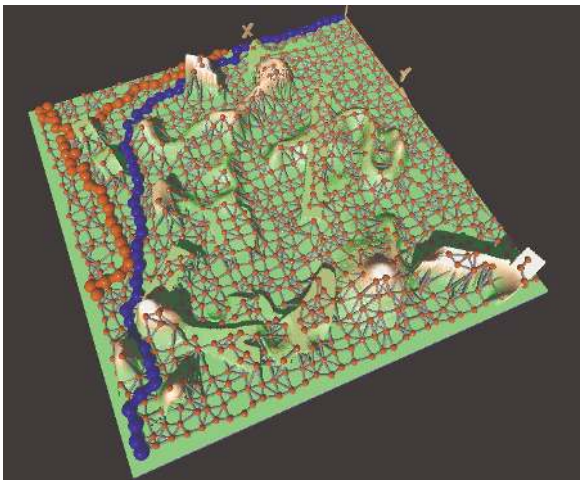
The final figure (Figure 4d) represents the system tuned to follow a tree line. Having a low overall cost for the terrain near dense trees caused this behavior. Having the vehicle seek areas of low observability could have caused a similar behavior.

4.2. Section level

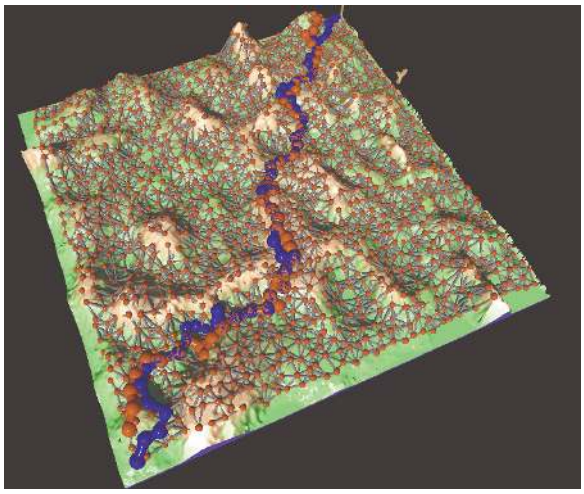
At the section level of the 4D/RCS architecture, we have investigated several different movement techniques through simulation. These include joint optimal vehicle movement for formations, having both vehicles arrive at the goal at the same time, simultaneous vehicle movement with the vehicles constrained to maintain a closeness constraint (never too close, and never too far), and a single vehicle moving at a time (bounding behavior).



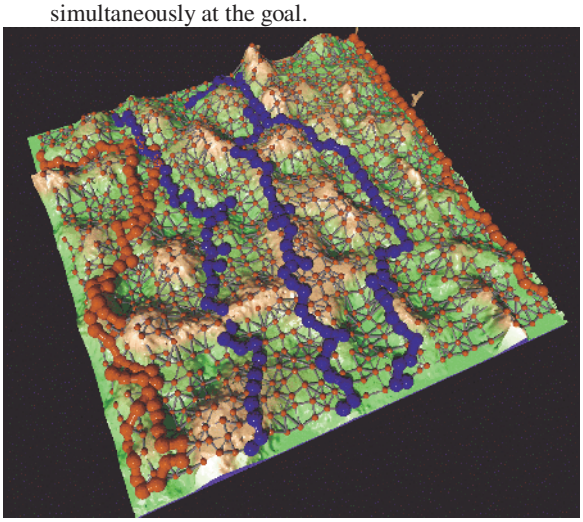
a) Joint optimal paths for formations.



b) Both vehicles travel at constant speed and arrive simultaneously at the goal.



c) Bounding behavior.



d) Platoon level.

Figure 5: Example two vehicle movement techniques possible through rule application at the section level.

In the section level, the planning spaces are significantly larger than at other levels as the number of vehicles increase the dimensionality of the problem. This is not only a problem for autonomous vehicles; it is also a problem for human

driven vehicles as well. The way that this problem was solved with human driven vehicles was to organize them into formations that reduce the freedom of each vehicle. The same approach was taken in the autonomous case. In this case, the constraints presented in the previous section, strongly restrict the multidimensional space so that planning within the constraints can be done in real time.

The advantages of planning in these multidimensional spaces are that the resulting paths are not only optimized for one vehicle, they are optimal for the group. This is a big distinction from the standard approach where the path is generated for one vehicle and then the other vehicles are attached to the first with some control (or reactive) law that maintains a certain distance from the first vehicle.

Figures 5a through 5c show the variety of behaviors that can be generated by simply changing the constraints on the space. In these examples the cost evaluation was not modified, only different constraints where used to segment the spaces. All three examples show the collapsing of the 4-dimensional space generated by the (x,y) position of both vehicles into a two and a half dimensional elevation. This elevation determines the cost of traversing the space. The size of the map shown is approximately 5 km on the side.

In Figure 5a the constraints are set so that the vehicles should be no more than 1 km apart from each other and no less than 0.8 km apart. This is assuming that they should not be so close as to be disabled at the same time, but close enough to stay within radio range.

Figure 5b shows the path of two vehicles that have the additional constraint of being asked to maintain the same speed throughout the path. In this case, one of the vehicles follows an optimal path, however, the second vehicle moves around the first vehicle to maintain the constrained distances.

Figure 5c shows bounding over-watch. Only one of the vehicles is allowed to move at any given time. In the example, one vehicle moves, while the other is in an observation position. Then the roles are changed. A set cost is added to stopping which makes the leaps longer.

Figure 5d shows 6 vehicles organized in groups of 2. This is an example where both the platoon level and 3 independent section levels are shown. The section level is planned first, which in this case a highly constrained 6 dimensional space is used to search, and then the resulting 3 paths are used to constrain the path of the independent section levels. For the example, the 3 different constrains shown in Figures 5a to 5c where used (one per section).

5. CONCLUSIONS

In this paper we have presented a rich world model and planning system that is capable of constructing single and multi-vehicle plans. These plans may exhibit different behaviors that can be changed on a cycle-by-cycle basis without the need to generate separate behavior modules. The different behaviors may be achieved by simply changing the cost vector or rule-base that is used in the plan graph creation and evaluation. However, further work remains to be done in incorporating even more features from the world into the world model and developing a larger set of rules and weight vectors. In addition, to date all of the behaviors performed by the system have been tuned by hand. There exists a rich repository of data that could be made available to a learning system to improve upon these hand-tuned behaviors. A learning system could be used in the graph node selection process as well as in the edge cost evaluation. In addition, different forms of learning could be experimented with. A reinforcement signal could be developed for reinforcement learning, or the system could operate as normal and observe its own simulation results to implement supervised learning.

REFERENCES

1. Albus, J. S., "RCS: A Reference Model Architecture for Intelligent Control," *Computer*, Vol. 25, No. 5, 1992, pp. 56-59.

2. Blum, A. L. and Furst, M. L., "Fast Planning Through Planning Graph Analysis," *Artificial Intelligence*, Vol. 90, No. 1-2, 1997, pp. 281-300.

3. Branicky, M. and Newman, W., "Rapid Computation of Configuration Obstacles," *Proceedings of the IEEE International Conference on Robotics and Automation*, Vol. 1, 1990, pp. 304-310.

4. Brooks, R. A. and Lozano-Pérez, T., "A Subdivision Algorithm In Configuration Space For Findpath With Rotation," *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, Vol. 2, 1983, pp. 799-806.

5. Crowley, J. L., "Navigation for an Intelligent Mobile Robot," *IEEE Journal of Robotics and Automation*, Vol. RA-1, No. 1, 85 A.D., pp. 31-41.

6. Dijkstra, E. W., "A note on two problems in connexion with graphs," *Numerische Mathematik*, Vol. 1, 1959, pp. 269-271.

7. Grevera, G. J. and Meystel, A., "Searching for an Optimal Path Through Pasadena," *Proceedings IEEE International Symposium on Intelligent Control*, 1988, pp. 308-319.

8. Haigh, K. Z., Shewchuk, J. R., and Veloso, M. M., "Exploiting Domain Geometry in Analogical Route Planning," *Journal of Experimental and Theoretical Artificial Intelligence*, Vol. 9, 1997, pp. 509-541.

9. Hoffmann, J.. FF: The Fast-Forward Planning System. AI Magazine 22[3], 57-62. 2001. AAAI Press.
Ref Type: Magazine Article

10. Hong, T., Balakirsky, S., Messina, E., Chang, T., and Shneier, M., "A Hierarchical World Model for an Autonomous Scout Vehicle," *Proceedings of SPIE's 16th Annual International Symposium on Aerospace/Defense Sensing, Simulation, and Controls*, 2002.

11. Hwang, Y. K. and Ahuja, N., "Gross Motion Planning - A Survey," *ACM Computing Surveys*, Vol. 24, No. 3, 1992, pp. 219-291.

12. Kuipers, B., "The Spatial Semantic Hierarchy," *Artificial Intelligence*, Vol. 119, No. 1-2, 2000, pp. 191-233.

13. Saab, Y. and VanPutte, M., "Shortest Path Planning on Topographical Maps," *IEEE Transactions on Systems Man and Cybernetics*, Vol. 29, No. 1, 1999, pp. 139-150.

14. Shoemaker, C. and Bornstein, J. A., "Overview of the Demo III UGV Program," *Proceedings of the SPIE Robotic and Semi-Robotic Ground Vehicle Technology Conference*, Vol. 3366, 1998, pp. 202-211.

15. Singh, S., Simmons, R., Smith, T., Stentz, A., Verma, V., Yahja, A., and Schwehr, K., "Recent Progress In Local and Global Traversability For Planetary Rovers," *Proceedings IEEE International Conference on Robotics and Automation*, Vol. 2, 2000, pp. 1194-1200.

16. Thrun, S. and Bücken, A., "Integrating Grid-Based and Topological Maps for Mobile Robot Navigation," *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, AAAI Press, 1996, pp. 944-951.

17. Wallner, F., Kaiser, M., Friedrich, H., and Dillmann, R., "Integration of Topological and Geometrical Planning in a Learning Mobile Robot," *Proceedings of the IEEE/RSJ/GI International Conference on Intelligent Robots and Systems '94*, Vol. 1, 1994, pp. 1-8.

18. Yahja, A., Stentz, A., Singh, S., and Brumitt, B. L., "Framed-Quadtree Path Planning for Mobile Robots Operating in Sparse Environments," *Proceedings 1998 IEEE International Conference on Robotics and Automation*, Vol. 1, 1998, pp. 650-655.