2019 NDIA GROUND VEHICLE SYSTEMS ENGINEERING AND TECHNOLOGY SYMPOSIUM
AGS TECHNICAL SESSION
AUGUST 13-15, 2019 - NOVI, MICHIGAN

Off-Road Autonomous Mobility

Alberto Lacaze, Edward Mottern, and Bryan Brilhart

Robotic Research, LLC

ABSTRACT

Off-road autonomous navigation poses a challenging problem, as the surrounding terrain is usually unknown, the support surface the vehicle must traverse cannot be considered flat, and environmental features (such as vegetation and water) make it difficult to estimate the support surface elevation. This paper will focus on Robotic Research's suite of off-road autonomous planning and obstacle avoidance tools. Specifically, this paper will provide an overview of our terrain detection system, which utilizes advanced LADAR processing techniques to provide an estimate of the surface. Additionally, it will describe the kino-dynamic off-road planner which can, in real-time, calculate the optimal route, taking into account the support surface, obstacles sensed in the environment, and more. Finally, the paper will explore how these technologies have been applied to a wide variety of different robotic applications.

1. INTRODUCTION

Reliable autonomous mobility in an outdoor cross-country environment presents a set of challenges that are not common to on-road systems. Specifically:

- The support surface that the vehicle must traverse s not given and cannot assume to be flat as in the on-road environments. Vegetation and water make it challenging to estimate the support surface elevation.
- Unlike on-road scenarios, a-priori knowledge of the terrain is not available or is poor in most cases.
- Problems of localization and therefore registration are exacerbated by the fact that differential GPS is not available for many military applications.
- Real-time sensing and scene interpretation are costly and produce large onboard computational and bandwidth requirements.
- Classification of vegetation between traversable and non-traversable is at large an unsolved problem.
- Medium range sensing (50 to 200 meters) is very poor.

Developing platforms in off-road environments has the added challenge of the large amount of support personnel needed to test, deploy, and maintain the robotic platforms. Robotic Research is one of the few autonomy providers that not only has developed the software and hardware needed for the deployment of off-road robotic systems, but also, since its inception, emphasized the need to work in tough field conditions that mimic realistic military scenarios. This paper will present the approach and results.

2. ARCHITECTURE

Planning algorithms have been in literature for many years [1]. Hierarchical approaches to planning and representation have been studied

under the label of operations research for an even longer period of time. There are undeniable advantages to organizing the control system hierarchical structures [2]. Some of these advantages included:

- Reduction of computational burden.
- Homogeneous resolution within each level and therefore easier representation.
- Ability to re-plan at different rates at each level (slower re-planning at the top of the hierarchy and faster re-planning at the bottom of the hierarchy).
- Distribution of computational burden between levels (i.e. distributed computing).
- Efficient compression of representation.

In our case, we follow the RCS hierarchical architecture [3], [4]. RCS is a hierarchical control system divided into a multiplicity of levels of resolution. At the top of the hierarchy, the representation is coarse with large time and space horizons and slow re-planning cycles. At the bottom of the hierarchy, the opposite is true. Fast re-planning cycles at the bottom provide the stability and quick response; however, the scope is small in the time and space. In comparison with behavioral approaches, hieratical systems tend to create a more explicit world representation. Cost criteria are used to evaluate a model of the system traversing the predicted world representation to achieve the assigned goal. First, a very coarse behavior is generated at the top of the hierarchy, and then this same behavior is refined at each level of the hierarchy. Proponents of hierarchical architectures argue that applying cost evaluation criteria is much easier to resolve using a complete representation as opposed to dealing with multiple, sometimes contradicting, sets of behaviors. However, complex world representations and the complexity of testing plan combinations make the implantation of hierarchical systems challenging. RCS contains both reactive and deliberative (planning) components. Hierarchical architectures tend to lean toward planning solutions because they have a representation that allows the prediction components necessary for planning. Behavioral approaches tend to be more reactive in nature, which is sufficient for simple environments.

Within a hierarchy, sensing and perception loops tent to cluster the sensed data and, if needed, pass it to supervisor levels creating an upward flow of representation. On the other hand, commands tend to flow downward from coarse levels with large scopes, to the higher resolution levels with shorter scopes and faster control loops.

Figure 1 shows the outline of an RCS chain of command. Each level of the hierarchy (3 in the case of the figure) is composed of three different functional entities: Sensory Processing (SP), World Model (WM), and Behavior Generation (BG).

Sensory Processing collects clusters and classifies data supplied by the sensors. The results of SP are sent to the WM for archival and registration. Low level SPs (bottom of the hierarchy) tend to deal with pixels and higher levels deal with discrete entities such as concepts that reflect several layers of sensor fusion.

World Model serves as a repository of SP results as well as serves to the BG modules. It collects registers and fuses information across time, as well as being the keeper of all the a-priori knowledge. Since it keeps the models of the level's working space, it predicts the outcome of behaviors generated by the GB module.

Behavior Generation hypothesizes the sets of alternative plans that may satisfy the command sent from the level above and sends them to the WM for evaluation. Pans that satisfy the requirements of the supervisor as well as cost criteria sent with the command are then sent to the surrogate levels for further refinement and execution.

There are many subtleties of RCS that help to reduce complicity. The modular approach encourages reusability of code at different levels [5], [6].

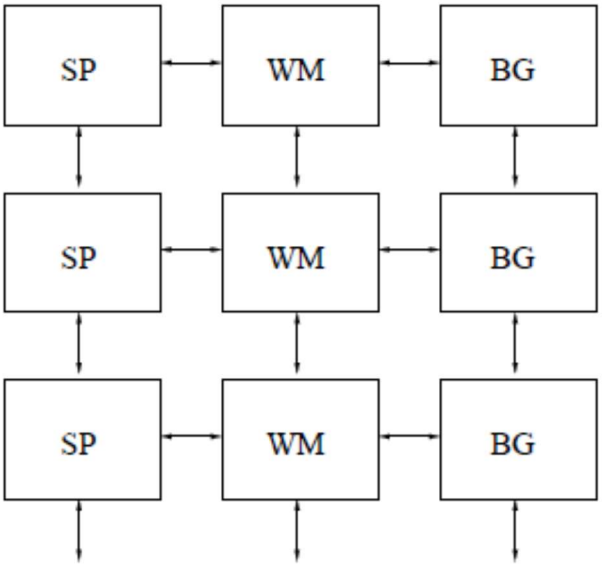

Figure 1: Outline of an RCS hierarchy

3. AUTONOMOUS MOBILITY

Figure 2 shows the different RCS levels used for autonomous vehicle control, as well as the different interfaces and suggestions for control cycle times. In this paper, we will concentrate on the Subsystem and Primitive levels for locomotion.

We assume that the Vehicle Level will generate coarse plans for the vehicle to follow, and that these paths will be sent to the Subsystem Level for execution.

At the bottom of the RCS hierarchy are the sensors and actuators. LADAR is the primary sensor used in our off-road platforms. This sensor provides about one hundred thousand measurements per second to a range of about 100 meters. Other sensors include color camera, inertial measurement, and Radar.

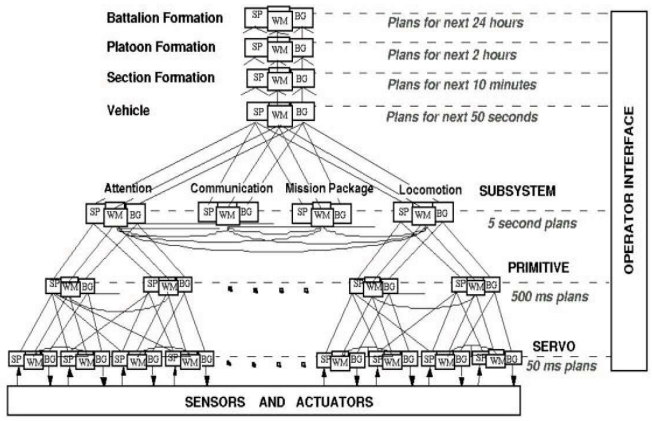

Figure 2: RCS Hierarchy for mobile platforms

4. SUPPORT SURFACE

In order to plan how to traverse the terrain in front of the vehicle, the BG modules need to be aware of the support surface. Determining the support surface is often a trivial task for on-road environments; however, in off-road environments, the presence of vegetation, dust, and water make the problem more difficult.

Our approach has the following steps:

- LADAR range measurements are geometrically transformed into a voxel representation, centered around the vehicle and collected as the vehicle moves through the terrain.
- The voxels record both "hits" and "transparencies." For each range measurement one hit occurs per laser beam at the measured distance. Whereas several transparencies occur along every laser beam, starting at the LADAR and extending just short of the measured distance. Voxels with recorded transparencies are partially if not completely transparent.
- A robust minimum (height) is used between the hits and transparencies counts to determine the support surface.
- A robust ratio between the transparencies and hits is used to determine the "density" of a particular voxel. We assume that a less transparent voxel will be more likely to be solid than a more transparent one. This is an assumption that is not always true (i.e. glass or steel wire). Therefore,

other attributes from the color cameras are used to better classify the traversability of the voxels.

Figure 3 shows the support surface found using the above described method for a terrain with shoulder height grass. The inverted cones show the path taken by the vehicle. The bottom of the cone is the location of the center of the navigation unit where the wheels should meet the support surface. The cones are 1 meter high. The blue surface is the predicted support surface. The walls that cross the path of the vehicle are the cross cuts of the voxel representation. This was done because if all voxels were displayed the support surface would not be viewable. The color of the cross cuts represents the number of hits (middle) and transparencies (bottom). The darker the red (green) color of the voxel, the more hits (transparencies) the particular voxel received. For most of the data collected, the error between the predicted support surface and the actual support surface does not exceed 0.25 meters for this kind of vegetation. The denser the vegetation the harder it is for laser beams to penetrate; and therefore, larger errors could be generated. The errors are monitored online as the vehicle compares predicted and actual elevation and makes adjustments to attitude and speed accordingly. Each voxel in the figure is 0.4 x 0.4. 0.1 meters.

The voxel representation is only used at the Primitive Level since the number of hits available in the Subsystem Level is not large enough to determine an accurate support surface. The LADAR does not give consistent measurements of the horizontal plane beyond 20 meters because of the shallow angle of incidence.

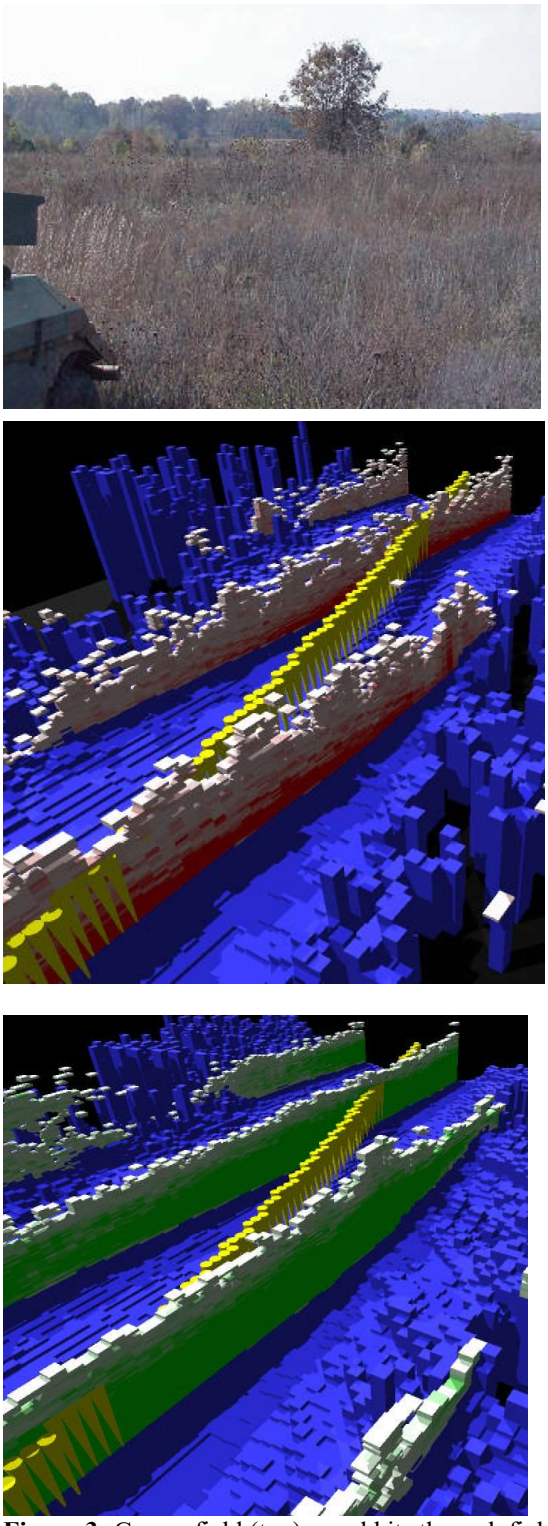

Figure 3: Grassy field (top) voxel hits through field (middle) and transparencies (bottom)collected for a vehicle traversing shoulder height vegetation

5. PLANNING ALGORITHM

A support surface is only calculated in the Primitive Level. Therefore, the strategies used for calculating the cost traversal at the Primitive and Subsystem Level are significantly different.

At the Primitive Level, the support surface is used to determine the stability as well as the roughness of the ride through several potential plans. At the Subsystem Level (past 20 meters) only sensed obstacles and a-priori data are used. The LADAR reasonably detects vertical obstacles up to 60 meters.

In both cases, the BG modules are graph based searches. [7] shows in more detail how these graphs are generated. Figure 4 shows the Primitive Level (curvy paths in the center) and the Subsystem Level (straight path in the outskirts) graphs. The top figure shows the graph used for an initial condition where the vehicle has the wheels pointed straight ahead, while the bottom picture shows the graphs used when the vehicle has the wheels placed on the smallest turning radius.

5.1. Primitive Level

The Primitive Level graph computed offline (Figure 4) contains some of the parameters needed to calculate the cost to traverse each potential path. For example, the length and the side accelerations. Cost must also be added to account for traversing the local terrain. This includes the support surface and the "density" of the terrain that must be traversed to follow each trajectory. Therefore, the BG module uses the WM representation to evaluate the cost of traversal. Figure 5 shows how vehicle masks are placed along each of the Primitive Level trajectories. These masks are used to compute pitch and roll along each path. The masks are also used to compute the amount of "density" that the vehicle must traverse to follow that trajectory. Other terms in the cost function include the roughness that each wheel will have to follow across the path, as well as checks for protruding objects that my hit the undercarriage of the vehicle.

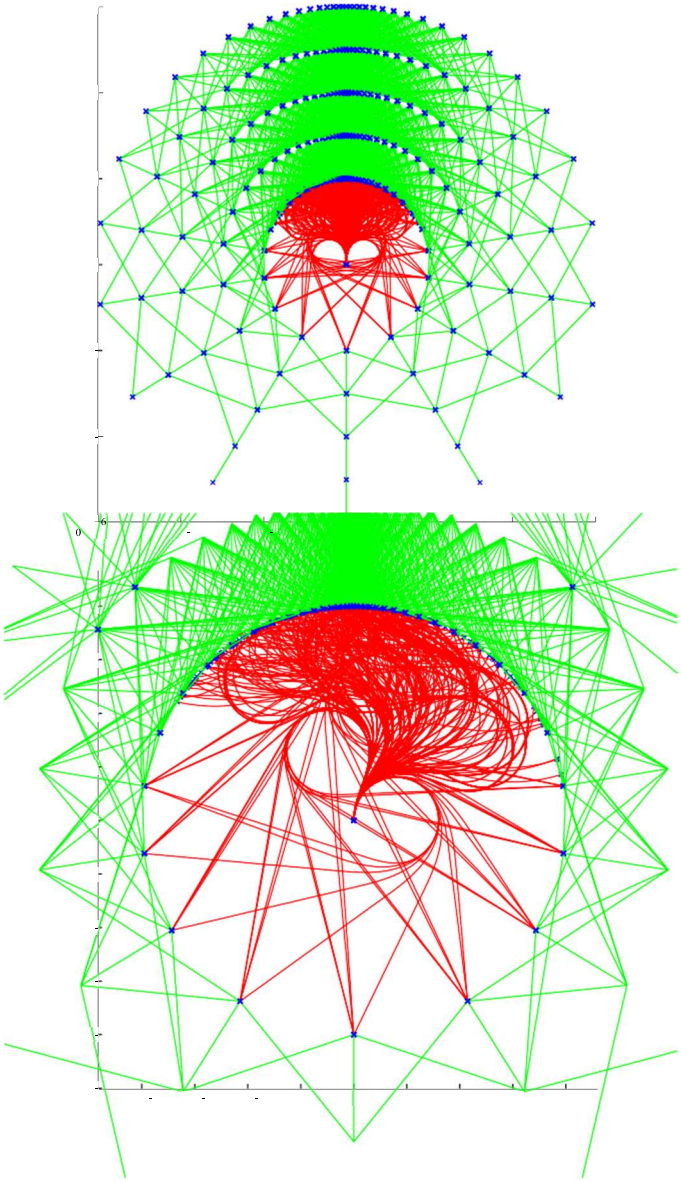

Figure 4: Egographs for the Primitive and Subsystem Level

Another important factor in whether a particular tile has already been seen. The cost evaluation will assign larger costs to trajectories that place the wheels of the vehicle in tiles that have never been seen by the sensor because these tiles have unknown elevation and may be holes or ditches. As these unknown elevation tiles get closer to the vehicle, the cost of the trajectories that cross them increases nonlinearly with distance.

All of these parameters are taken under consideration in order to calculate the costs of each trajectory. They are computed as the search is underway. Therefore, trajectories (or edges of the graph) that are not opened by the search algorithm do not get evaluated. Re-planning at this level is done at 4-10 Hz.

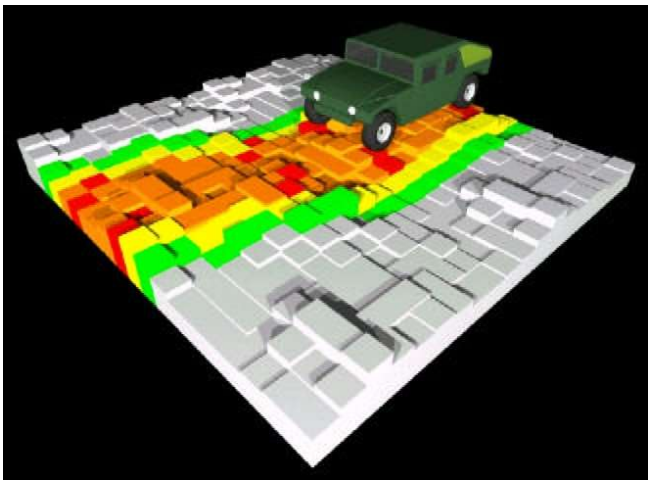

Figure 5: Placing vehicle masks across the paths to compute cost

5.2. Subsystem Level

The Subsystem Level representation only contains obstacles and a-priori data. In cases where only 30 meter DTED data is available, elevation is ignored as its accuracy is not sufficient for planning. The obstacles at this level are found by a cumulative probabilistic measure of the obstacles reported by the LADAR and stereo systems. The detections are accumulated in the world model as the vehicle is moving through the terrain. The trajectories used by this level are straight line approximations. It is not necessary to generate kinematically correct trajectories since the Primitive Level will have many opportunities (planning cycles) to refine the path selected by this level. The planner finds the optimal shortest obstacle free path available in the graph. The graph used by the Subsystem Level can be seen in Figure 4.

Figure 6 shows both levels at work. In this example, the vehicle was crossing a bridge in a wooded area. The top figure shows the Vehicle Level plan (purple trajectory) that misses the bridge. This trajectory was done by using 30 meter DTED. Because of the inaccuracies of the a-priori data and the GPS drift, the Vehicle Level plan only marks the path coarsely across the bridge. The Subsystem and Primitive Level of pith pictures show areas that have not been seen by any of the sensors.

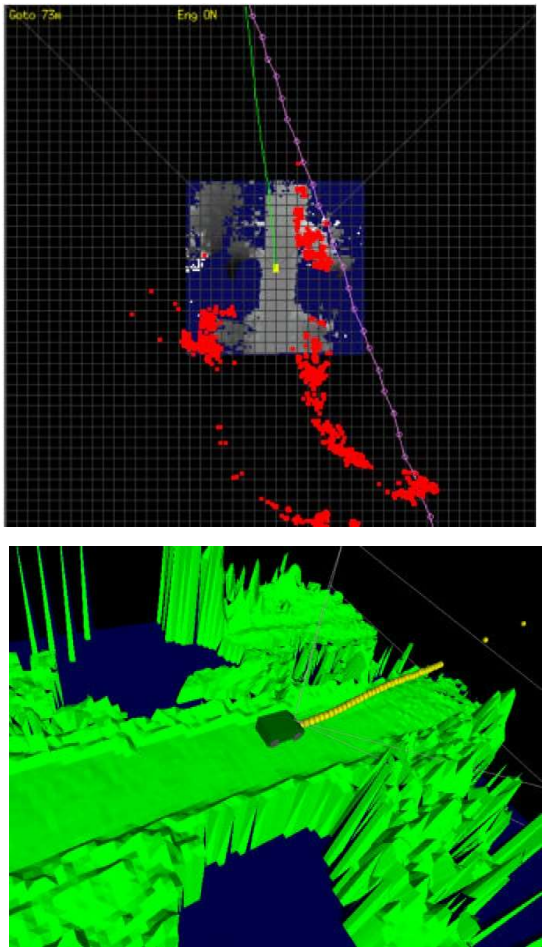

Figure 6: Primitive and Subsystem Levels traversing a bridge. Top and 3-D view

6. HIGH MANEUVERABILITY PLANNER

In some cases, such as physical blockages or barriers in a path, it is necessary to plan paths outside the geometry of the planned trajectory. In

these scenarios, the High Maneuverability Planner (HMP) is used. The HMP generates trajectories for the robot that avoid obstacles while meeting the constraints imposed by the operator (no-go areas, speed limits, etc.). The input to the HMP is a moving robot centered 3-D representation of the local environment. It outputs a trajectory to be followed by a path tracking module on the robot that controls wheel speeds and steering. The HMP combines all sensor information into a single representation of the environment that is the utilized to evaluate the cost of performing different actions. The resulting trajectories are sequences of vehicle state/time pairs that the vehicle control processes follow. Among other things, the state information includes the desired vehicle position and attitude. By including kino-dynamic constraints, HMP can protect the robot from accidents involving over-steering as well as collisions with obstacles. When conditions require invoking the HMP, the resulting path is used to temporarily augment the previous plan. Figure 7 illustrates the interaction of the HMP with other planning methods.

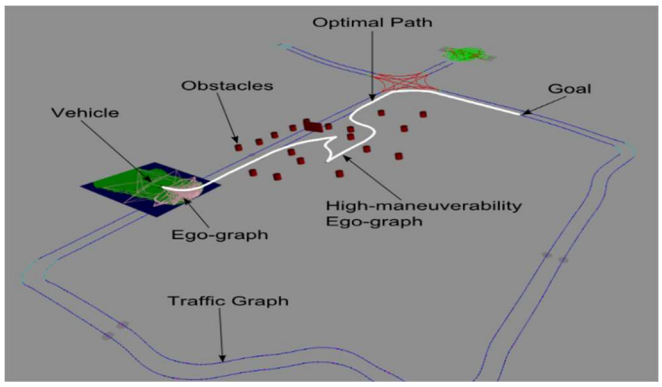

Figure 7: Interaction between HMP and other planning methods.

7. RESULTS

During the US Marine Corp Warfighter Lab (MCWL) Intuitive Robotic Control of Autonomous Behaviors (IROC) experiment in 2017, Robotic Research demonstrated a fully autonomous MUTT vehicle traversing a gauntlet of obstacles without the use of GPS. MCWL provided feedback that the Robotic Research autonomous vehicle, shown in Figure 8, completed the course faster than any other autonomous vehicle to date.

Figure 8: Robotic Research autonomous MUTT vehicle traversing gauntlets at IROC (Muscatatuck, Indiana)

8. CONCLUSIONS

- The challenges that can be found in traversing cross country are not the same that are available for on-road environments.
- RCS hierarchical architecture is well suited for off-road mobility.
- Novel support surface computations and planning algorithms have been developed to deal with the challenges of the environment.
- Kino-dynamic planning has been developed to deal with the most complex and/or dynamics environments.

1. REFERENCES

[1]T. Perez-Lozano, “Spatial planning: a configuration space approach,” *IEEE Trans. Computers,* vol. 32, 1983.

[2]Y. Maximov and A. Meystel, “Optimum design of multiresolutional hierarchical control systems,” in *Proceedings of IEEE Int’l Symposium on Intelligent Control,* Glasgow U.K., 1992, pp. 514-520.

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

[4]J. Albus, “An intelligent system architecture for manufacturing (ISAM),” Tech. Rep., NIST, 1996.

[5]J. Albus and A. Meystel, *Engineering of Mind,* Wiley Series of Intelligent Systems, 2001

[6]J. Albus and A. Meystel, *Intelligent Systems: Architecture, Design and Control,* Wiley Series on Intelligent Systems, 2002.

[7]A. Lacaze, Y. Moskovitz, N. DeClaris, and K. Murphy, “Path planning for autonomous vehicles driving over rough terrain,” in *International Conference of Intelligent Systems,* NIST, 1998.