Proceedings of the 1998 IEEE
ISIC/CIRA/ISAS Joint Conference
Gaithersburg, MD • September 14-17, 1998

Mobility Planning for Autonomous Navigation Multiple Robots in Unstructured Environments

Martial Hebert, Anthony Stentz, Chuck Thorpe

The Robotics Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh PA 15213

Abstract[1]

In this paper, we describe an approach to mobility planning for autonomous navigation of multiple vehicles unstructured environments. This approach is suitable for complex missions in which multiple goals with sequencing constraints may be specified. At the lowest level, this approach uses command arbitration in order to combine command recommendation from driving behaviors. Driving behaviors include local behaviors such as obstacle avoidance or goal tracking, and strategic behavior such as route planning to distant goals. Obstacle avoidance is performed by evaluating a fixed command set with respect to a terrain map built from 3-D sensors. Route planning is based on the D dynamic route planner which evaluates the command set based on the currently optimal path to the next goal. The path is updated frequently based on map updates from the obstacle avoidance sensors. On top of this behavior-based system, an additional planner called GRAMMPS, continuously analyzes the outputs of the route planners running on each of the vehicles and re-computes optimally the allocation of goals to vehicles in order to minimize time and distance travelled.*

This approach has be exercised extensively with two vehicles using stereo and ladar for obstacle detection and map building. Experiments were conducted over long distances with missions of up to nine intermediate goals in different configurations. We report of the implementation of this system and on its performance.

1 Introduction

Unmanned ground vehicles operate autonomously in natural, unstructured terrain. To accomplish this, they detect obstacles, plan paths, and build maps. In practical applications, multiple vehicles may operate simultaneously to carry out a common mission.

In an earlier paper [17], we described the first demonstration of an autonomous system with on-board, dynamic path planning combined with obstacle avoidance. At that time, the system was able to handle simple missions involving a single goal and a single vehicle. In realistic missions, however, several vehicles must coordinate their routes and complex mission plans may include multiple goal locations. Once multiple vehicles and multiple goals are used, the system must be able to make complex decisions such as allocating goals between the vehicles, and computing paths to many goals simultaneously in real time. Furthermore, the system must be able to accommodate a variety of different types of missions. For example, a mission might require all the vehicles to initially drive to a first goal location. e.g., a staging area, and then drive to a set of goals, e.g., observation points, in any order using any combination of vehicles while a different mission may require the vehicles to visit a set of goals in a specific sequence, e.g., to pick up supplies in a particular order.

In this paper, we report on experiments with a system that answers those needs by extending the capabilities described in [17]. The experiments were conducted with the two vehicles shown in Figure 1, operating in the environment shown in Figure 5. In the area of planning, the system described here demonstrates successful approaches to on-line route planning with multiple goals, on-line mission planning, e.g., allocation and ordering of goals, and to flexible mission specification.

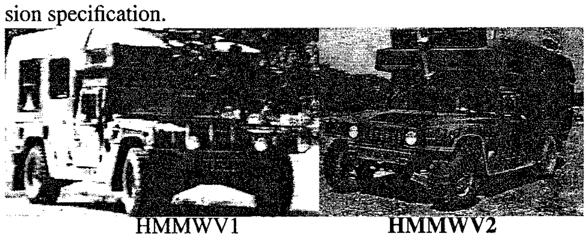


Figure 1: The two HMMWVs used in the experiments.

Beyond the planning requirements, the system must be able to integrate information from the different sensors in order to provide a consistent map representation to the planners. In the experiments described in this paper, two different sensors were used: a new, high speed, laser range finder, and a video-rate stereo machine. The experiments were also an opportunity to demonstrate those two state-of-the-art sensors and to demonstrate the ability of the system to accommodate different resolutions, fields of view, and ranges.

Although multi-vehicle planning systems have been reported by others [10][11][12], the experiments reported here are arguably the first demonstrations of vehicles with fully integrated systems operating in unstructured natural environments (see [4] for a survey of research in multi-robot control.)

The paper is organized as follows. First, we give an overview of the system architecture and of the major components of the system. Second, we describe in detail the experiments conducted with this system and walk through a step-by-step description of one particular mission. Third, we provide implementation details, e.g., parameters used, computation time.

1. The CMU UGV Group includes: B.Brumitt, P.Chang, J. Frazier, J. Hancock, M.Hebert, D. Huber, D. Langer, T. Stentz, C. Thorpe, T. Williamson, A. Yaha. This work was supported under DARPA/TACOM contract DAAE07-96-C-X075 monitored by U.S. Army TACOM.

2 Architecture Overview

The system is divided into a mobility element, which is responsible for driving the vehicle around obstacles and for planning paths to goal points, and a mission element, which is responsible for specifying the goal points, monitoring the execution of the mission, and generating any necessary updates to the mission. Those elements are implemented as sets of decentralized, asynchronous modules communicating through dedicated communication links, using the IPT inter-process communication package. The complete system architecture is shown in Figure 3.

Mobility

The system architecture is based on the behavior arbitration approach introduced in [15] and developed in the context of cross-country navigation systems in [9]. On each vehicle, a local obstacle avoidance module takes input from range sensors, constructs a local obstacle map, and outputs recommendations for steering and speed commands.

The local obstacle maps generated by the obstacle detection module are integrated into a global map maintained by a separate module called Intercom. In addition to maintaining the map for the vehicle on which it resides, Intercom is also responsible for exchanging map updates with the other vehicle. As a result, the two vehicles share the same map of the environment at all times.

The map representation is a large grid of cells labelled as obstacle or drivable. The map is used by a route planner, D*, to steer the vehicle toward goal locations. D* is a dynamic planner in that it is capable of continually updating the route to the goal based on update to the map from the obstacle detection module. Specifically, D* maintains an internal cost map in which the cost of driving from each cell to the goal is encoded; by modifying this cost map efficiently every time the map is updated, D* is able to compute the new optimal route to the goal. It has been shown in an earlier paper that D* is guaranteed to compute the optimal path to the given goal at all times [16].

Periodically, D* generates a set of steering recommendations based on the configuration of the cost map around the vehicle. More precisely, steering choices that guides the vehicle in directions with lower cost to the goal receive higher scores. A first algorithm for evaluating steering choices from cost maps was discussed in [17]. The algorithm used here is substantially different and leads to much smoother paths.

Recommendations from the local avoidance module and D* are lists of votes, one vote for each steering choice. Votes are normally continuous between 0 and 1, with 1 indicating a fully drivable arc. Either module can veto a steering choice by setting the vote to a veto value (-1). In particular, steering choices that would lead the vehicle outside of the field of view of the sensor are vetoed.

The votes are combined by an arbiter in order to generate a single driving command. A typical snaphost of the steering arbiter operation is shown in Figure 4.

Mission

The main module in the mission planning element is the GRAMMPS (Generalized Robotic Autonomous Mobile Mission Planning System) mission planning and execution module which is responsible for assigning goals to vehicles based on a mission description and the current cost maps. GRAMMPS is also a dynamic planner in that it is capable of efficiently changing the order or allocation of goals between the vehicle as the global obstacle map is updated.

The basic approach to GRAMMPS was introduced in [2]. The basic concept is shown in Figure 2: A separate D* is associated with each goal location in each vehicle. All the D*'s use the same obstacle map in order to update their cost maps. Periodically, GRAMMPS uses the costs of all the robots to all the goals provided by D* in order to compute the optimal assignment of goals to robots. This description of GRAMMPS using separate D* modules for each robot and each goal is convenient but, in practice, a single route planning process is used on each vehicle to compute the costs to a subset of the goals. The details of the search algorithm currently used in GRAMMPS can be found in a companion paper [3].

In this approach to mission planning and execution, all the modules run asynchronously. In particular, D* continuously updates its cost maps and generates command recommendations independently from GRAMMPS or the obstacle avoidance modules. Typically, updates from the mission planner, route planner, and local avoidance planner are issued at increasingly higher rates, e.g., 0.5 Hz, 1 Hz, and 2 Hz, respectively.

The second part of the mission planning element is a user interface in which mission descriptions can be built and sent to GRAMMPS. A general interpreted language is used for describing the missions: Robots may be instructed to visit specific goals in a given order, or to visit a set of goals in any arbitrary order. A goal may be assigned to a specific vehicle or may be visited by any of the two vehicles.

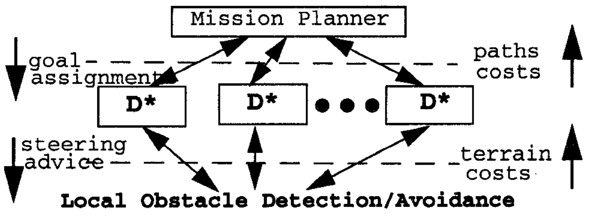


Figure 2: Conceptual view of GRAMMPS.

3 Experiments

The two-vehicle system described above was exercised in the field over a period of one month. Fragments of missions were first executed in order to evaluate the components. This section provides a detailed look at the execution of complex missions in real environments. Twelve complete missions, i.e., missions in which the vehicles visit all the goals specified in the mission description, were recorded and analyzed in detail. Each mission involved driving each vehicle up to 600 meters and visiting up to nine goals. The average speed was 1 m/s in all the missions.

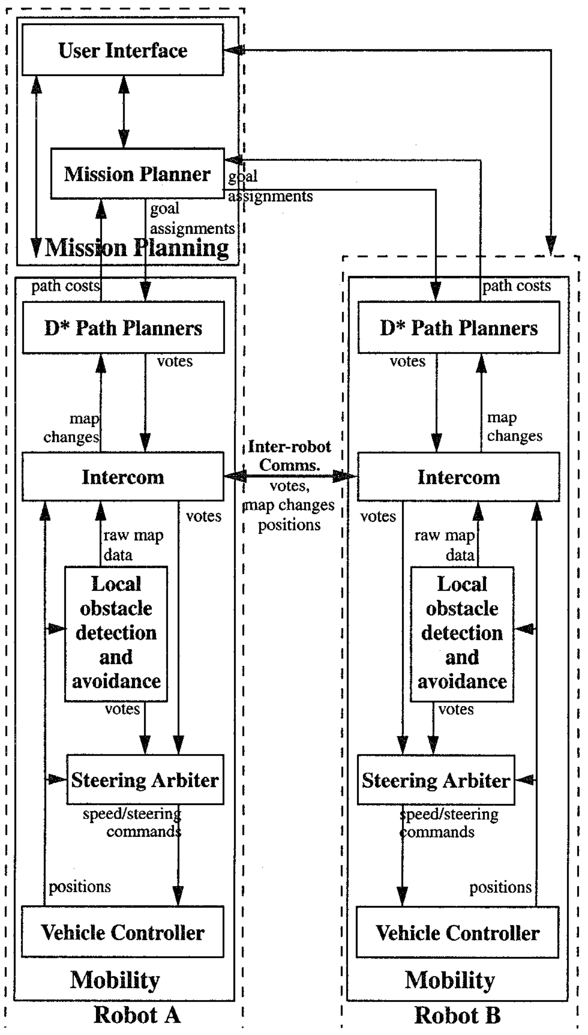


Figure 3: System architecture. The identical mobility systems reside on both vehicles. One of the robots supports the mission planning system. Mission planning could be moved to an off-vehicle base station if needed.

For reasons of space, we can discuss only one mission in detail; a summary of mission results is provided in Section 5. This mission illustrates the basic capabilities of the mobility system such as goal re-ordering, route re-planning, and local obstacle avoidance. The example mission consists of eight goals. The mission description provided by the user is, in this case, a mix of mandatory orders, i.e., specific goals that must be visited by each vehicle, and optional orders, i.e., goals which can be visited in any order as decided by the run-time system. In order to facilitate the description of the mission, each goal is designated by a name. The robots HMMWV2 and HMMWV1 are instructed to visit goals INTER and EDGE3, respectively; the remaining goals may be visited in any order.

During this experiment, HMMWV1 and HMMWV2 travelled 670m and 683m, respectively. A total of 592 obstacle cells were seen in the map, with an additional 1500 potential field cells assigned around the obstacle cells. The run took a total of 16 minutes.

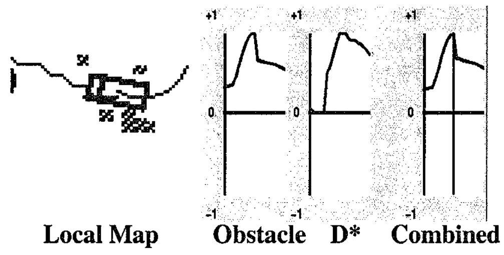


Figure 4: Arbitration results for the configuration shown at left. The votes from the obstacle avoidance module and D* are combined into a single distribution. The weight for D* is 8 times smaller than the weight for the obstacle avoidance module because the vehicle is in a cluttered area.

The approximate locations of six of the goals are shown in Figure 5 overlaid on a mosaiced image of the test area. The final goal location EXUNIT is slightly outside of this image to the right. HMMWV2 starts on a narrow path as shown in Figure 5; HMMWV1 start position is close to the final goal EXEUNT.

In order to show the major stages of this mission, we include below displays obtained by replaying the data recorded during the run. In all those displays, HMMWV1's path and vehicle icon are drawn using shaded lines, while HMMWV2's are drawn using black lines. Two types of paths are drawn. The path behind the vehicle is the path actually driven by the vehicle under combined control of D* and local obstacle avoidance; the path ahead of the vehicle is the best path planned by D* given the current obstacle map. The obstacles are shown as grey squares and the goal locations are indicated by their names.

4 Detailed Description

We describe below the details of the implementation of the system for the experiments described above. The purpose of this discussion is three-fold. First, we provide information on computation and communication requirements. Second, we describe as completely as possible the parameters used in the system, e.g., map resolution. Third, we emphasize algorithmic improvements over previous similar systems. For example, the D* vote generation and the steering arbitration algorithms described below are substantial improvements over previous implementations.

4.1 Planning and Map Management

As described above, the planning and map management responsibilities are distributed among three components, D*, Intercom, and GRAMMPS.

The D* route planner operated with a 1 meter resolution obstacle map in which obstacle cells are grown by a 2 meter potential field. Typical maps handled by D* are 500 meters on the side. D* is fast enough to update its internal cost map and to generate steering votes at 2Hz.

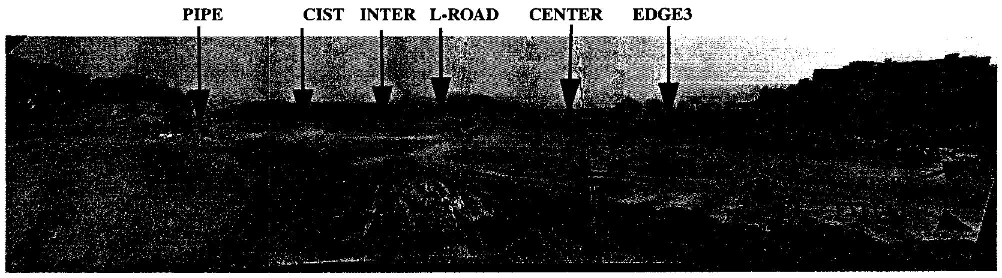
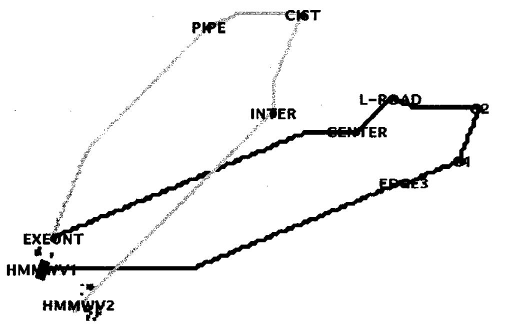
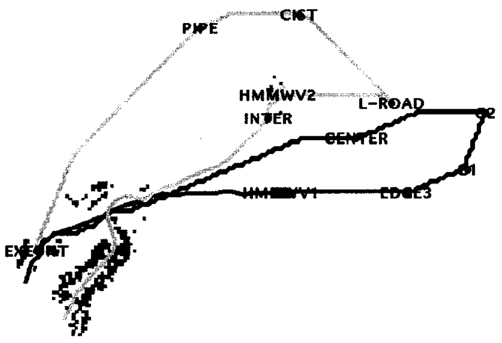


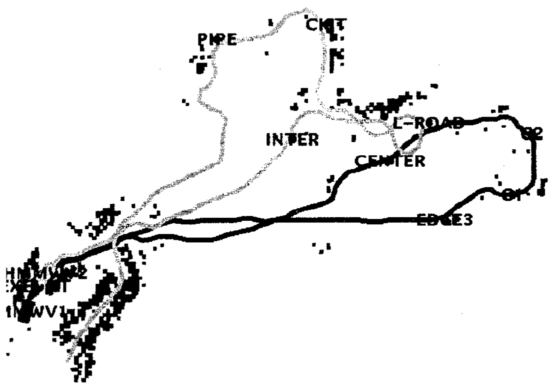
Figure 5: Actual position of the goals used in the mission described below; total distance travelled from starting point to final goal is approximately 500 meters



Initial configuration: Obstacles are known only in the immediate neighborhood of the vehicles at their starting positions. HMMWV2 plans its route from the starting location to the first goal, INTER, which is imposed by the mission plan, through two intermediate goals, ending at the final goal, EXEUNT. Similarly, HMMWV1's plan goes first to the mandatory goal EDGE3 and to four intermediate goals before going back to EXEUNT. GRAMMPS generates the allocation of goals between the two vehicles.



Change in goal allocation: Shortly after HMMWV2 reaches goal INTER, GRAMMPS decides that goal L-ROAD should be switched to HMMWV2's plan. This event occurs because newly detected obstacles just past INTER forced HMMWV2 to move closer to L-ROAD than had been initially planned. As the goal switching event occurs, the routes are immediately recomputed, leading to seamless transition between goal distributions.



Mission completed: Both HMMWVs complete the mission. The

start of the mission; the paths are also substantially different. The paths shown here are optimal based on the information available at the time they were generated. Were these runs to be performed again, given the complete obstacle map shown above, the initial paths and goal ordering would be globally optimal and, therefore, would be very different from the ones shown above.

This vector of steering votes is computed by averaging the cost-to-goal of cells along each steering arc, weighted by the length of the segment of the arc contained in each cell. This approach is different from the one used in [17] and leads to substantially smoother vote distributions.

In addition to collecting map data and transmitting map data from one vehicle to the other, Intercom is also responsible for checking for interference between vehicles. Specifically, Intercom maintains a 10 meter buffer zone around each vehicle. One of the vehicles is stopped if the buffer zone is violated. The map management algorithm is temporarily suspended on the stopped vehicle so that the moving vehicle is not included in the map. More sophisticated algorithms can be found in the literature on collision avoidance in fleets of multiple robots [10][1]; we did not attempt to implement an optimal strategy for interference avoidance.

The GRAMMPS mission and execution planner updates the mission at 0.5 Hz or less. Updates consist of the ordering of goals to be visited by each vehicle. In the experiments described above, GRAMMPS resides on HMMWV1 and communicates the plan updates to the local D* module as well as to the system resident on HMMWV2. However, it is important to note that the mission planner, and its user interface, could reside at a base station outside of the vehicles.

4.2 Speed Control

Vehicle speed is controlled from three sources: mission directives may alter vehicle speed, for example by stopping at goals; maximum speed may be limited depending on the current steering radius in order to avoid tipover; and the speed should be adapted based on the complexity of the terrain, i.e., the more cluttered the terrain, the lower the speed.

In the current implementation, speed-based mission directives are only binary directives, that is, the vehicle is either stopped at a goal or waiting for the other vehicle, or it is driven at the speed commanded by the mobility system. Because of the low vehicle speeds used in the experiments, adaptation of speed as a function of turning radius was not necessary. The source of speed control was the obstacle avoidance system. The basic idea is that if the environment is cluttered, or, equivalently, if the steering votes have low values, then the speed should be reduced.

In the current speed control algorithm, the speed is the maximum desired speed multiplied by a speed factor s_O which is computed as: $s_O = v_{max} - \alpha (1 - r_{max})$, where v_{max} is the maximum steering vote in the current vote distribution, and r_{max} is the proportion of votes that are close to v_{max}. This algorithm controls speed with the desired behavior, as illustrated by the three main cases shown in Figure 6: If both v_{max} and r_{max} are large, the vehicle can maneuver over a wide range of arcs and, as a result, s_O is close to 1 and the commanded speed is close to the maximum speed. If v_{max} is high but r_{max} is small, the vehicle has a clear path but it has less room to maneuver and s_O is decreased. If both v_{max} and r_{max} are small, the vehicle is driving in a cluttered environment and s_O is decreased even further. Finally, this approach to speed forces the vehicle to stop if the environment is too cluttered, i.e., v_{max} and r_{max} are too small, even if no steering arcs are actually vetoed.

Experimentally, this approach to speed control allowed for fast driving across the obstacle-free areas of the map, e.g., in the segment between PIPE and EXEUNT in the earlier example, and to drive at reduced speed near obstacle regions such as the tree line near L-ROAD.

4.3 Steering Arbitration

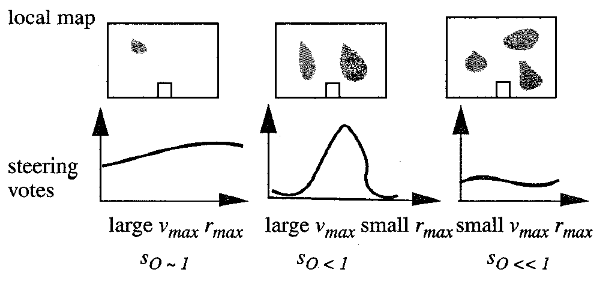
The steering arbiter used 21 arcs regularly spaced between the minimum and maximum turning radii of -7m and 7m. New commands were issued at 2Hz based on the most recent sets of votes from the obstacle avoidance and D* modules.

This implementation of the steering arbiter is similar to the one originally described in [9]. However, as indicated above, a critical difference is the ability to adjust the relative weights of voting modules depending on the environment. Specifically, the votes are combined using a linear combination: $vi = w_{D*} v_i^{D*} + w_O v_i^O$, where v_i^{D*} and v_i^O are the votes from D* and obstacle avoidance, respectively, and w_{D*} and w_O are the weights. If the vehicle is in an obstacle-free area, then D* should have control over the vehicle since the votes from the obstacle avoidance module do not carry any information except for the veto votes outside of the sensor's field of view. On the other hand, if the vehicle is driving in a cluttered area, then the contribution of D* should be decreased, or even eliminated, in order to give control of the vehicle to the obstacle avoidance module.

The idea of dynamic weight adaptation originates in the more general concept of adapting of the driving behavior based on the environment in a way similar to the speed adaptation algorithm described above. In fact, the weight adaptation algorithm used in those experiments is based on the speed factor, s_O generated by the obstacle avoidance module. More precisely, the weights are defined as $w_O = f(s_O)$, where f is a linearly decreasing function of s_O between a non-zero minimum value for w_O, to ensure some contribution of obstacle avoidance even if the speed is close to the maximum speed, and a maximum value less than 1, to ensure that D* introduces a small amount of bias toward the goal even in cluttered environments, as shown in Figure 7.

4.4 D* Extension

One problem with the regular grid representation is that it represents map data inefficiently. Natural terrains are usually of sparse and fractal forms. Also, natural terrains are often not completely known in advance. It is generally the case that we encounter large areas having no obstacles and that we assume the unknown part of the terrain to be empty. This implies that unknown environments will be encoded sparsely during the initial explorations and that many areas will remain sparsely populated even during execution. Moreover, the regular grid cell representation only allows eight angles for direction, resulting in an abrupt changes in path direction and an inability, in some cases, to generate a straight diagonal path through empty areas.



local map
steering votes
large v_{max} r_{max}
$s_O \sim 1$
large v_{max} small r_{max}
$s_O < 1$
small v_{max} r_{max}
$s_O << 1$

Figure 6: Speed adaptation in three cases: obstacle-free, narrow corridor, and cluttered environment; for each case, the local map is shown at the top and the corresponding steering vote distribution is shown at the bottom.

One way to solve those problems is to use a quadtree data structure to represent the environment, instead of using regular size cells. A quadtree is based on the successive subdivision of region into four equally sized quadrants. The criterion for splitting a region into four smaller regions is that if the region still contains obstacles, it is subdivided until either a subregion free of obstacles is found or the smallest grid cell is reached. In the latter case, an occupied cell is marked as an obstacle cell, else it is marked as a free cell.

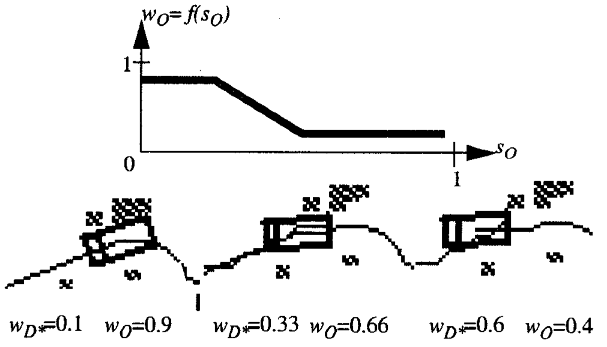


Figure 7: Adaptation of arbiter weight as function of the density of obstacles: (top) weight adaptation function; (bottom) example sequence in which w_{D*} increases as the vehicle drives away from the obstacles.

The use of quadtrees addresses the memory efficiency aspect but it does not address the issue of path smoothness. For that, higher resolution cells are added around the perimeter of each quadtree region. We use these border cells as the cells upon which a path planner performs its search. This augmented quadtree representation is called a framed-quadtree. The framed-quadtree representation permits many angles of direction, instead of just eight angles as in the case of regular grid representation. A path can be constructed between two border cells lying far away from each other. Most importantly, the path generated on a framed quadtree more closely approximates the exact optimal path, especially for sparse and unknown environments.

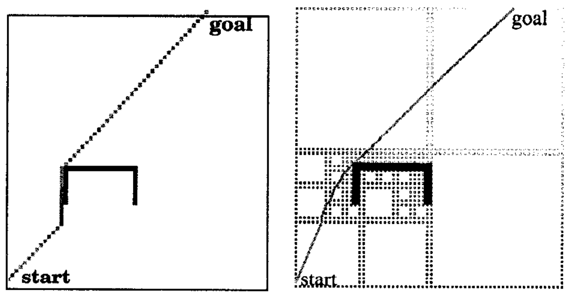


Figure 8: Path generated using a regular grid structure (left) and path generated using the framed quadtree structure.

5 Conclusion

We reported in this paper on experiments with a system for autonomously driving two vehicles based on complex mission specifications. We showed that the system is able to plan local paths in obstacle fields based on sensor data, to plan and update global paths to goals based on frequent obstacle map updates, and to modify mission execution, based on the updated paths to the goals. This system is the first multi-vehicle and multi-goal system demonstrated in real, natural environments with this degree of adaptation. Moreover, because it is built on the earlier design of distributed, behavior-based architectures, the system can be easily extended to accommodate new modules or additional vehicles.

References.

[1] Arora, S. Polynomial-time Approximation Schemes for Euclidean TSP and other Geometric Problems. In *Proc. IEEE FOCS*. 1996. pp. 2-13.

[2] B. Brumitt. Dynamic Mission Planning for Multiple Mobile Robots. In Proc. In *Proc. ICRA'96*. May 1996.

[3] B. Brumitt. GRAMMPS: A Mission and Execution Planner for Multiple Robots in Unstructured Environments. Submitted to ICRA'98.

[4] Cao, U.Y., et al. *Cooperative Mobile Robotics: Antecedents and Directions*. Autonomous Robots 4.pp 7-27. 1997.

[5] Froelich, C., M. Mettenleiter, F. Haertl. "Imaging laser radar (LIDAR) for high-speed monitoring of the environment." SPIEProceedings of the Intelligent Transportation Systems Conference,October 1997.

[6] Froelich, C., J. Hancock, R. Sullivan, D. Langer. "High-performance Imaging Laser Radar." Submitted to ICRA'98.

[7] Hancock, J., E. Hoffman, R. Sullivan, D. Ingimarson, D. Langer, M. Hebert. "High-performance laser range scanner." SPIE Proceedings of the Intelligent Transportation Systems Conference, October 1997.

[8] M. Hebert. Pixel-Based Range Image Processing. In *Proc. ICRA'94*. May 1994.

[9] D. Langer, J. Rosenblatt, M. Hebert. *A Behavior-Based Approach to the Autonomous Navigation Systems*. IEEE Transactions on Robotics and Automation, vol.10, no. 4. 1994.

[10]Le Pape, C. A Combination of Centralized and Distributed Methods for Multi-Agent Planning and Scheduling. In *Proc. ICRA*. pp. 488-493. 1990.

[11] Mackenzie, D.C., Arkin, R.A, Cameron, J.M. Mutliagent Mission Specification and Execution. *Autonomous Robots 4*. pp. 29-52. 1997.

[12]Mataric, M. J. Reinforcement Learning in the Multi-Robot Domain. *Autonomous Robots 4*. pp.73-83. 1997.

[13]M.K. Morganthaler. UGV Mission Planning. *RSTA for the UGV: Providing Eyes for an Autonomous Vehicle*. O. Firschein and T. Strat Ed. Morgan Kaufman Publ. 1997.

[14]Parker, L. *Heterogeneous Multi-Robot Cooperation*, Ph.D. Thesis, MIT, Feb. 1994.

[15]J.K. Rosenblatt. DAMN: A Distributed Architecture for Mobile Navigation. In *Proceedings of the 1995 AAAI Spring Symposium on Lessons Learned from Implemented Software Architectures for Physical Agents*. H. Hexmoor & D. Kortenkamp (Eds.). AAAI Press, Menlo Park, CA. 1995.

[16]A. Stentz. Optimal and Efficient Path Planning for Unknown and Dynamic Environments. *International Journal of Robotics and Automation*. Vol. 10, No. 3. 1995.

[17]A. Stentz, M. Hebert. *A complete navigation system for goal acquisition in unknown environments*. Autonomous Robots. Kluwer Academic Publishers. 1995.