Experiments in Autonomous Driving With Concurrent Goals And Multiple Vehicles

Barry Brumitt, Martial Hebert, and the CMU UGV Group

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

Abstract[1]

In this paper, we report 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, e.g., the ordering of the goals, based on the updated paths to the goals.

Two recently developed sensors are used for obstacle detection: a high-speed laser range finder, and a video-rate stereo system. An updated version of a dynamic path planner, D, is used for on-line computation of routes. A new mission planning and execution monitoring tool, GRAMMPS, is used for managing the allocation and ordering of goals between vehicles..*

We report on experiments conducted in an outdoor test site with two HMMWVs. Implementation details and performance analysis, including failure modes, are described based on a series of twelve experiments, each over 1/2 km distance with up to nine goals.

This system is the first multi-vehicle and multi-goal system to be demonstrated in real, natural environments with this degree of generality.

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 [20], 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 [20]. 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.

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 [12][13][14], the experiments reported here are arguably the first demonstrations of vehicles with fully integrated systems operating in unstructured natural environments (see [5] 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. Because most of the components are derived from previously published approaches or are described in papers submitted separately, we concentrate on the algorithmic improvements over earlier versions in the Architecture and Implementation sections.

HMMWV1

HMMWV2

Figure 1: The two HMMWVs used in the experiments.

2 Architecture

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

1. The CMU Group includes: Barry Brumitt, Peng Chang, Jim Frazier, John Hancock, Martial Hebert, Daniel Huber, Dirk Langer, Tony Stentz, Chuck Thorpe, Todd Williamson, Alex Yahja.

communication package. The complete system architecture is shown in Figure 2.

Mobility Element

The system architecture is based on the behavior arbitration approach introduced in [17] and developed in the context of cross-country navigation systems in [18][11]. 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.

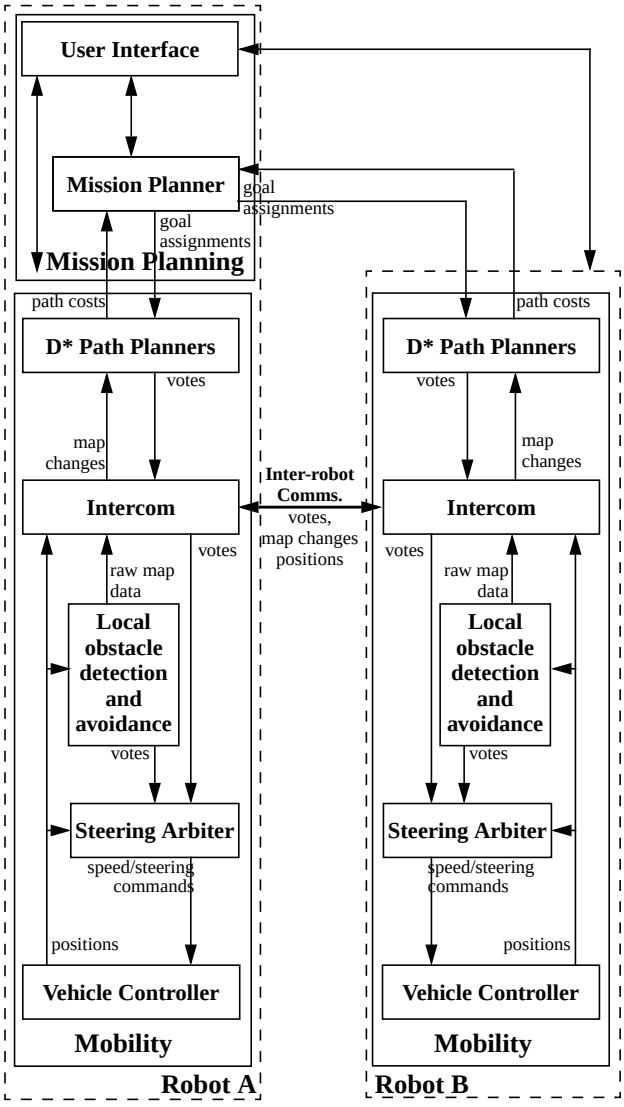


Figure 2: System architecture. The identical mobility systems reside on both vehicles. One of the robots supports the mission planning system. Mission planning could be moved to an off-vehicle base station if needed.

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 computethe 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 [19].

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 [20]. 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. This approach to command-based arbitration is based on the approach introduced in [18]. A typical snaphost of the steering arbiter operation is shown in Figure 3.

A long-standing issue with this approach is that a weighting scheme for combining votes between the two modules must be devised. Previous implementation of this approach used fixed weights, e.g., a large weight for local obstacle avoidance and a smaller weight for D*. The arbiter used in this paper incorporates a new approach for weight adaptation based on the local obstacle map. Empirically, weight adaptation leads to tremendous improvements in performance by allowing the obstacle avoidance system to assume total control in cluttered regions.

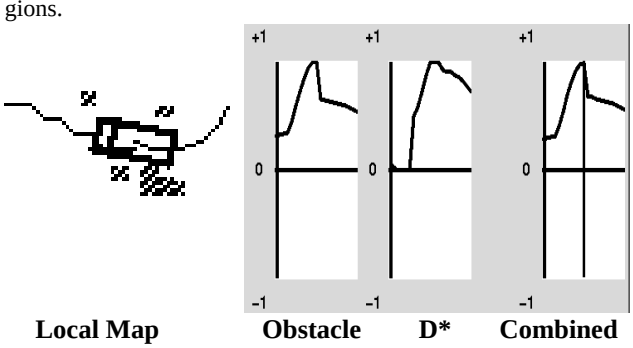


Figure 3: Arbitration results for the configuration shown at left. The votes from the obstacle avoidance module and D* are combined into a single distribution. The weight for D* is 8 times smaller than the weight for the obstacle avoidance module because the vehicle is in a cluttered area.

Together, D*, the arbiter, and the obstacle avoidance module constitute the mobility element of the system. The mobility element is identical on both vehicles, except for the range sensors used for detecting obstacles and for minor differences in the vehicle controllers.

Mission Planning Element

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 [3]. The basic concept is shown in Figure 4: 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 [4].

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. The complete language specification is described in [4]. This approach to mission description permits the specification of a large class of missions that includes most combinations of vehicles and goals that are relevant to practical applications.

Together, the user interface and the GRAMMPS mission planning module constitute the mission planning element. As shown in Figure 2, this element resides on one of the vehicles. This was mostly for convenience with respect to our hardware environment. In an alternative system, the planning element should reside on a separate, off-vehicle, along with the user interface, in way similar to the UGV Demo II OCU[15].

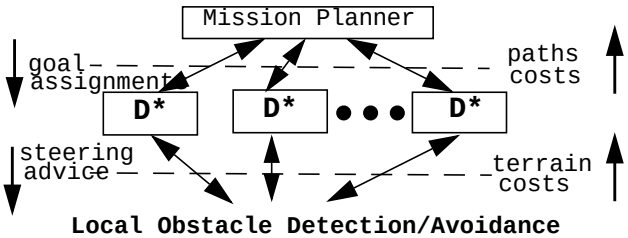


Figure 4: Conceptual view of GRAMMPS.

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.

The mission descriptions used in the tests included a mix of three types of directives: drive a given vehicle to a specific goal or a set of goals in a specific order; drive each vehicle to a specific set of goals using any goal ordering; and drive to a set of goals using any goal ordering and any allocation of goals to vehicles. Result of the analysis of the data recorded during those experiments, e.g., performance and failure modes, and implementation details, e.g., parameters used during the actual missions, are described in Section 5.

For reasons of space, we can discuss only one mission in detail. 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 and by any vehicle as decided by the run-time system. In order to facilitate the description of the mission, each goal is designated by a name (always in uppercase in this paper.) In this example, 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.

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.

A short paragraph is included after each snapshot to explain the details of the corresponding mission segment. 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. The obstacle and vehicle coordinates shown in the displays are computed by mapping directly GPS coordinates to screen coordinates.

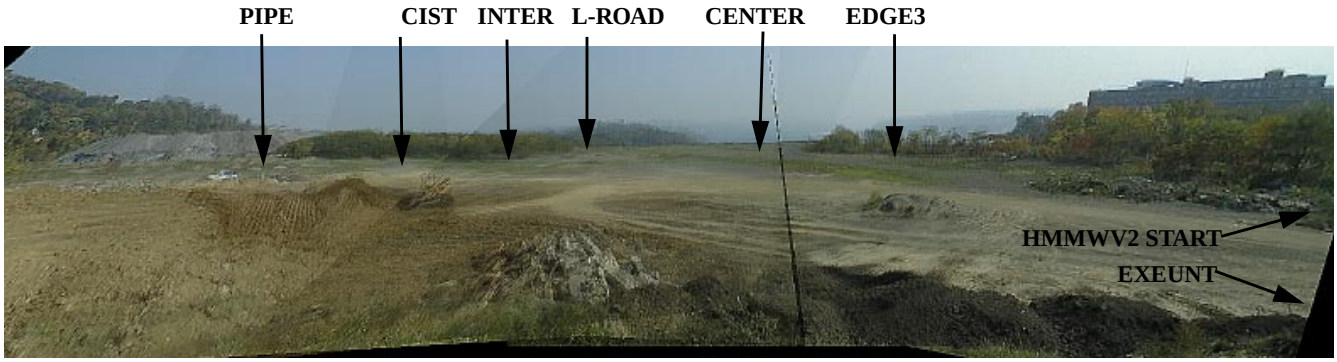
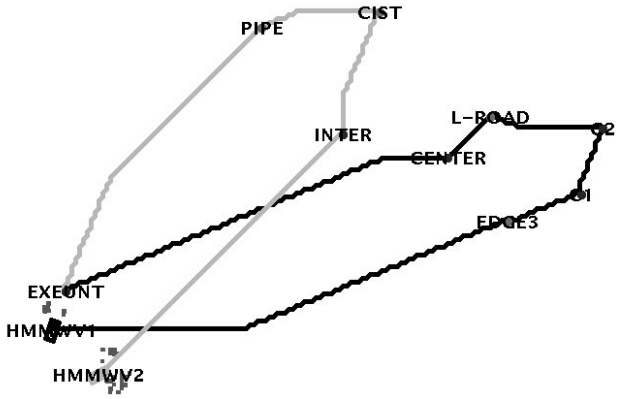
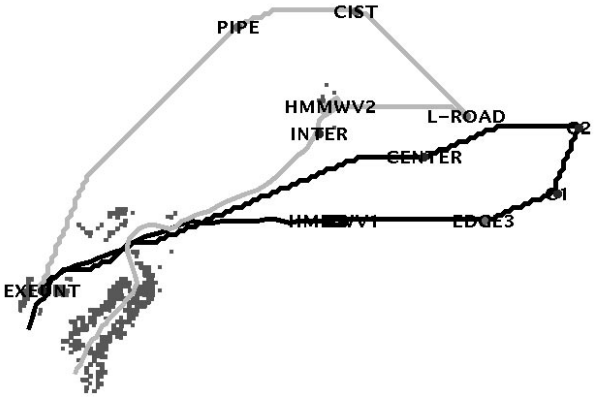


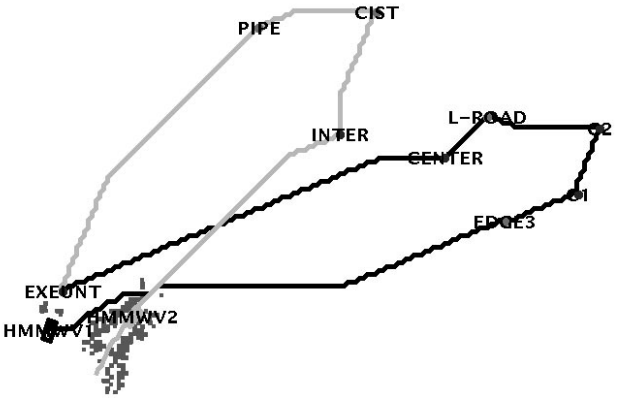
Figure 5: Actual position of the goals used in the mission described below; total distance travelled from starting point to final goal is approximately 500 meters



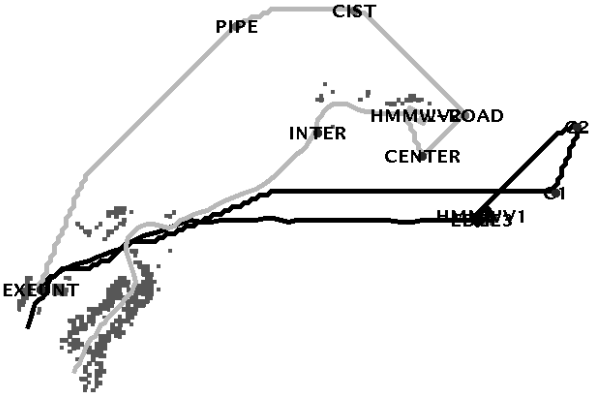
Initial configuration: Obstacles are known only in the immediate neighborhood of the vehicles at their starting positions. HMMWV2 plans its route from the starting location to the first goal, INTER, which is imposed by the mission plan, through two intermediate goals, ending at the final goal, EXEUNT. 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.



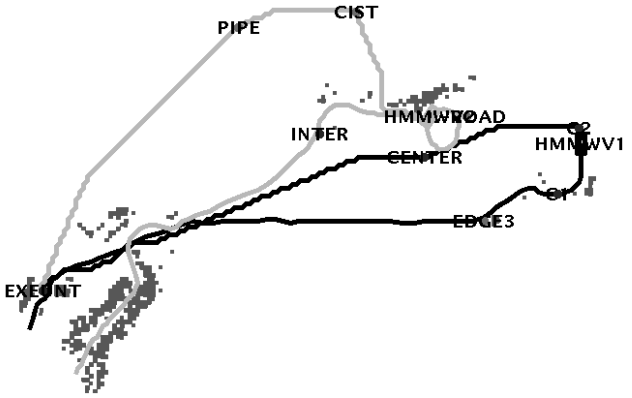
Change in goal allocation: Shortly after HMMWV2 reaches goal INTER, GRAMMPS decides that goal L-ROAD should be switched to HMMWV2's plan. This event occurs because newly detected obstacles just past INTER forced HMMWV2 to move closer to L-ROAD than had been initially planned. As the goal switching event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



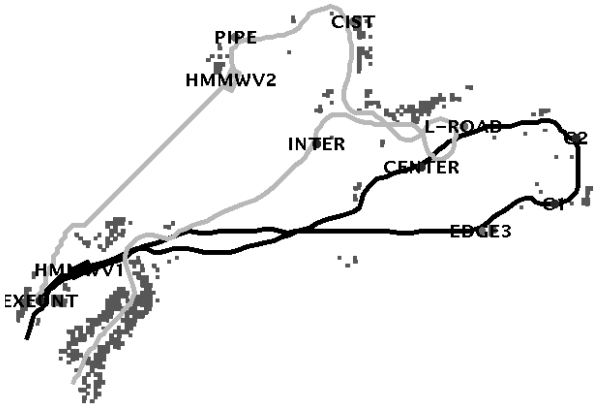
Mission start: HMMWV2 starts driving first, modifying its route based on the obstacle map accumulated since the start of the mission. At the same time, although it is not moving, HMMWV1 is updating its planned route based on the map built by HMMWV2. Specifically, the path to EDGE3 has now changed based on the new obstacles found by the laser system on HMMWV2.



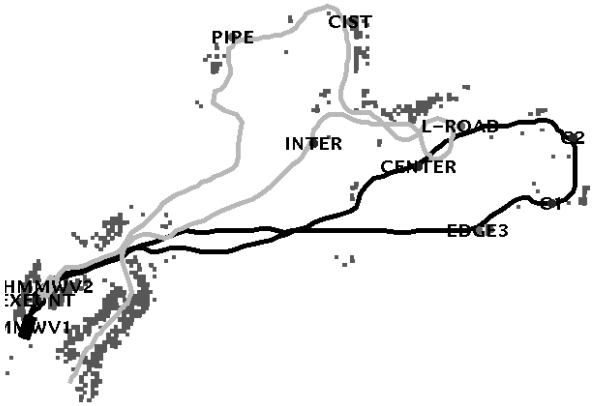
Change in goal ordering: As HMMWV2 is deflected from its path on its way to L-ROAD, the planner re-allocates goal CENTER to HMMWV2. As a result of this event, the ordering of goals for HMMWV1 is also changed; the plan now calls for C2 to be visited before C1.



Correction: The planner has reverted its previous decision and restored the original goal allocation. At the same time, the ordering of goals for HMMWV1 has also been restored, as expected.



Driving back: Although it occupies a very cluttered area, HMMWV2 executes a path that leads it out of the area around L-ROAD (the edge of a forest.) This path is executed through the steering arbiter by combining the global optimal path to CIST and the local steering commands for obstacle avoidance. Because the speed arbiter reduced its speed in this area, HMMWV2 has just reached its last goal, PIPE, while HMMWV1 as is completing the mission. The route uses all the information accumulated since the beginning of the mission. For example, the path from PIPE to EXEUNT computed by D* uses the portion of the obstacle map constructed at the very beginning of the mission.



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. For example, the map constructed between L-ROAD, INTER and CIST would prevent future mission from wasting time trying to cross directly from L-ROAD to CIST as HMMWV2's initial plan recommended. Instead, using this map, a planner would now generate the correct behavior, that is, the vehical would follow the tree line between L-ROAD and CIST.

4 Implementation Details

We describe below the details of the implementation of the system for the experiments previously described. 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 Communication

Communication bandwidth between the vehicles is always a serious concern with multi-agent systems because of limited bandwidths of radio systems. In these experiments, a FreeWave radio modem system was used. This system has a peak bandwidth of 110 Kbaud.

The communication rate between the vehicles was recorded and analyzed. The vehicles primarily exchange vote distributions and obstacles map updates. Vote distributions are updated at a maximum rate of 2Hz. The data rate for map updates can be measured in obstacle cells transmitted per second. The peak rate measured during the experiments was 80 cells per second. This rate was observed in the most cluttered areas of the test site. In those situations, the transmitted lists of obstacle cells are segmented into 50-cell packets.

The aggregate volume of transmission was always below 4800 baud. No compression algorithm was used. Straightforward approaches to data compression can be designed to reduce the data rate by a factor of 4. In particular, the encoding of cell coordinates and votes as floating-point numbers can be easily replaced by 1-byte encoding.

This analysis shows that this system design is suitable for use with low-bandwidth communication hardware.

4.2 Sensing

A video rate stereo machine with a three-camera system is used on HMMWV1 for range sensing. The machine generates 100x256 range images at up to 15Hz. A new imaging laser range finder, used on HMMWV2, is capable of acquiring range data up to 50m in a 360^o by 30^o swath [7][8]. Because of mechanical limitations, the rotational speed of the scanner is set to a maximum of 2400Hz. As a result, although a new laser design used in this sensor is capable

of measuring points at 500kHz, the acquisition rate was limited to two images per second for 60x1000 images. More recent, faster, versions of the scanner are described in [7].

The two sensors have very different characteristics. In particular, the stereo system is faster but has a narrow field of view (approximately 50^o; this was extended somewhat by using a pan control algorithm) while the laser range finder can sense all the way around the vehicle. In practice, the two sensors are complementary; stereo detects obstacles further ahead, while the laser range finder provides data in the periphery of the vehicle.

The use of very different sensors illustrates also the flexibility of the architecture. Both sensors used the obstacle detection and avoidance system described in [9][11], each with a different sensor model. Thanks to the use of the behavior-based approach, the change in sensor is limited to the obstacle detection module, and the rest of the system remains identical on both vehicles,

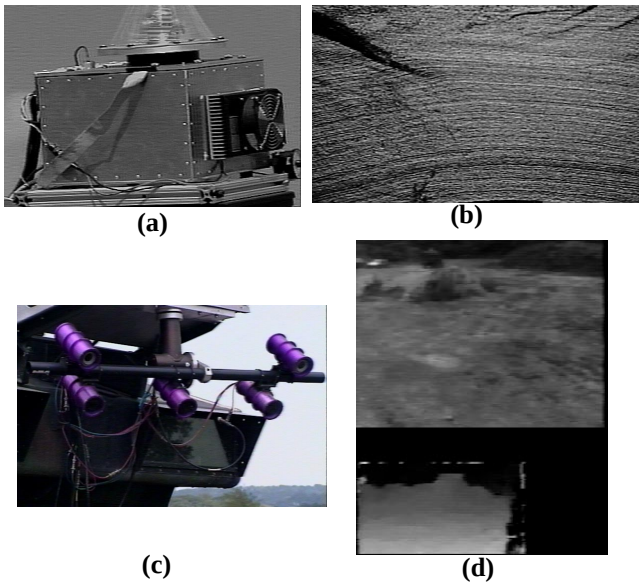


Figure 6: (a) Laser range finder; (b) 3-D display of data points from the laser range finder; (c) setereo cameras; (d) intensity image and corresponding disparity image from the stereo machine.

4.3 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.

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 [20] 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 [12][2]; 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.

4.4 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 7: 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 based on the complexity allowed for fast driving across the obstacle-free areas of the map, e.g., in the segment between PIPE and EXEUNT in the earlier exam-

ple, and to drive at reduced speed near obstacle regions such as the tree line near L-ROAD.

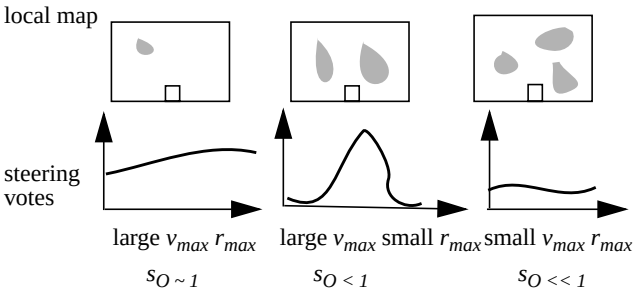


Figure 7: Speed adaptation in three cases: obstacle-free, narrow corridor, and cluttered environment; for each case, the local map is shown at the top and the corresponding steering vote distribution is shown at the bottom.

4.5 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 [11]. 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 8.

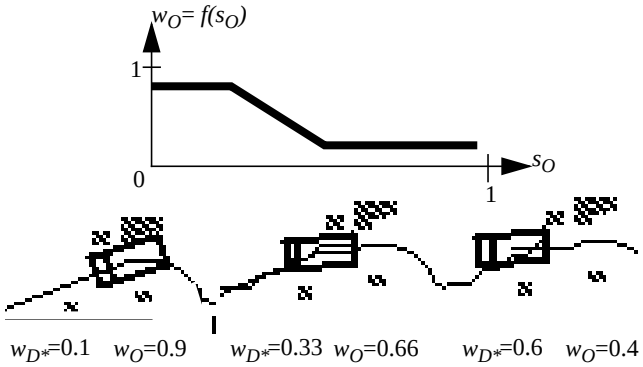


Figure 8: Adaptation of arbiter weight as function of the density of obstacles: (top) weight adaptation function; (bottom) example sequence in which w_{D*} increases as the vehicle drives away from the obstacles.

4.6 Position Estimation

Accurate position estimation is critical since, at all times, the vehicles need to know their positions with respect to each other and with respect to the map maintained by Intercom at all times. In exercising the system in real environments, it rapidly becomes clear that solutions that rely on single position sensors are not viable: GPS satellites may be occluded by terrain features; INS systems drift over time; and slippage makes encoder integration useless over any substantial amount of travel.

The solution chosen here is a tuned Kalman Filter that integrates data from a Novatel RT-20 differential GPS, the heading rate gyro from a Lear Astronautics MIAG INS system, and encoder readings. At low speeds, the filter integrates the readings from the heading rate gyro for heading estimation and GPS readings for position estimation; at higher speeds, the GPS readings are also used for estimating heading. Zero velocity updates are used whenever the vehicle is stopped, as measured by the encoders, in order to eliminate drift. This approach allows for seamless transition between different modes of estimation at different speeds.

5 Limitations and Failure Modes

As we indicated earlier, the two-vehicle navigation system was able to carry out complex missions in difficult, natural terrain. Although the system provides a solid base for the development of future autonomous systems, a number of limitations remain as can be seen in occasional failures of the system.

An important limitation has to do with the use of range sensors for obstacle detection to the exclusion of other types of sensors. The main problem here is that range sensors cannot discriminate between the changes in range due to vegetation and those due to actual obstacles. As a result, the vehicles are constantly avoiding small patches of vegetation which should not be reported as obstacles. In densely vegetated areas, the situation can degrade to the point that the system stops the vehicle and the mission fails.

This limitation is not an issue with the fundamental design of the system, but it is an issue of considerable importance for applying this work in any practical environment. The solution is to augment the sensor suite with a non-range sensor that can reliably discriminate vege-

tation. For example, we are currently working with a multispectral sensor which will permit real-time classification of vegetation. Details on the sensor and the algorithms can be found in [10].

A practical limitation of the system as presented here is that the vehicles do not have the ability to drive in reverse. More importantly, the arbiter and other parts of the system do not support reverse paths. As a result, a vehicle cannot escape if it is too far inside a cul-de-sac, even though it can recognize this occurance. Such a situation is unavailable and leads to failure of the mission. Consider the case of a long, narrow corridor; assuming the corridor is long enough, the sensors cannot see the end until the vehicle is well inside the corridor and cannot turn around. Support for reverse driving must be added to the system in order to avoid this failure mode.

One drawback of an asynchronous, distributed system is the latency in communicating information between the components. If the maximum commanded vehicle speed is low, the latency and asynchronous nature of the system do not cause any significant problems. If fact, in the experiments described above, the system runs on a non-real time Unix OS. As vehicle speed increases, latency and unknown delays due to the OS become critical. Therefore, moving toward higher speeds will require porting the navigation system to a real-time operating system.

The last common failure mode is loss of position estimation, typically due to loss of GPS visibility. In fact, this was the most common failure mode in early experiments with the system. As we indicated earlier, the areas of the test site where GPS drop-outs were more frequent were known, thus allowing us to avoid this failure mode in subsequent missions. More generally, such systems will be truly widely applicable only when reliable positioning systems, both internal and external, are built.

6 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.

In addition to the implementation and demonstration of the multi-vehicle/multi-goal capability, substantial enhancements were introduced in the various components of the system, including: adaptable weight in command arbitration; smooth estimation of vote vectors in D*; and use of two new sensors for obstacle detection.

The navigation system described here provides a solid basis for the development of reliable systems for complex missions. The experiments have helped identify the major limitations that need to be addressed for the next generation system in order to progress closer to fieldable systems.

References.

[1] M Alami, R, et al. “A General Framework for Multi-Robot Cooperation and its Implementation on a Set of Three Hilare Robots.” Experimental Robotics IV, 1995, pp. 26-39.

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

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

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

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

[6] 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.

[7] Froelich, C., J. Hancock, R. Sullivan, D. Langer. “High-performance Imaging Laser Radar.” Submitted to ICRA’98.

[8] 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.

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

[10]D. Huber, L. Denes, M. Hebert, M. Gottlieb, B. Kaminsky. A Spectro-polarimetric Imager for Intelligent Transportation Systems. In *Proc. SPIE Conference on Transportation Sensors. Pittsburgh*. October 1997.

[11]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.

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

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

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

[15]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.

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

[17]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.

[18]J.K. Rosenblatt, C.E. Thorpe. Combining Multiple Goals in a Behavior-Based Architecture. *Proceedings of IROS 1995*. 1995.

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

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