In *Proceedings, IEEE Conference on Robotics and Automation, (ICRA)*, Leuven, Belgium, May 1998.

Framed-Quadtree Path Planning for Mobile Robots Operating in Sparse Environments

Alex Yahja, Anthony Stentz, Sanjiv Singh, and Barry L. Brumitt

Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

Mobile robots operating in vast outdoor unstructured environments often only have incomplete maps and must deal with new objects found during traversal. Path planning in such sparsely occupied regions must be incremental to accommodate new information, and, must use efficient representations. In previous work we have developed an optimal method, D, to plan paths when the environment is not known ahead of time, but, rather is discovered as the robot moves around. To date, D* has been applied to a uniform grid representation for obstacles and free space. In this paper we propose the use of D* with framed quadtrees to improve the efficiency of planning paths in sparse environments. The new system has been tested in simulation as well on an autonomous jeep, equipped with local obstacle avoidance capabilities. We show how the use of framed quadtrees improves performance in terms of path length, computation speed, and memory requirements.*

1 Introduction

Path planning for a mobile is typically stated as getting from one place to another. The robot must successfully navigate around obstacles, reach its goal and do so efficiently. Outdoor environments pose special challenges over the structured world that is often found indoors. Not only must a robot avoid colliding with an obstacle such as a rock, it must also avoid falling into a pit or ravine and avoid travel on terrain that would cause it to tip over. Vast areas have their own associated issues. Such areas typically have large open areas where a robot might travel freely and are sparsely populated with obstacles. However, the range of obstacles that can interfere with the robot's passage is large— the robot must still avoid a rock as well as go around a mountain. Vast areas are unlikely to be mapped at high resolution a priori and hence the robot must explore as it goes, incorporating newly discovered information into its database. Hence, the solution must be *incremental* by necessity. Another challenge is dealing with a large amount of information and a complex model (our autonomous vehicle is a three degree of freedom, non-linear, non-holonomic system). Taken as a single problem, so much information must be processed to determine the robot's next action that it is not possible for the robot to perform at any reasonable rate. We deal with this issue by using a *layered* approach to navigation.

We have adopted the approach of decomposing navigation into two levels— *local* and *global*. The job of local planning is to avoid obstacles, reacting to sensory data as quickly as possible while driving towards a subgoal [4][5]. A more deliberative process, operating at a coarser resolution of information is used to decide how best select the subgoals such that the goal can be reached. This approach has been used successfully in the past in several systems at Carnegie Mellon [1][12]. In this paper we concentrate the discussion on global planning.

Approaches to path planning for mobile robots can be broadly classified into two categories— those that use exact representations of the world (e.g. [14]), and those that use a discretized representation (e.g. [3][6]). The main advantage of discretization is that the computational complexity of path planning can be controlled by adjusting the cell size. In contrast, the computational complexity of exact methods is a function of the number of obstacles and/or the number of obstacle facets, which we cannot normally control. Even with discretized worlds path planning can be computationally expensive and on-line performance is typically achieved by use of specialized computing hardware as in [3][6]. By comparison the proposed method requires general purpose computing only. This is made possible by precomputing an optimal path off-line given whatever a priori map is available, and then optimally modifying the path as new map information becomes available, on-line.

Methods that use uniform grid representations must allocate large amounts of memory for regions that may never be traversed, or contain any obstacles. Efficiency in map representation can be obtained by the use of quadtrees, but at a cost of optimality. Recently, a new data structure called a *framed quadtree* has been suggested as means to overcome some of the issues related to the use of quadtrees[2]. We have used this data structure to extend an existing path planner that has in the past used uniform (regular) grid cells to represent terrain. This path planner, D* [10][11] has been shown to be optimal in cases where the environment is incrementally discovered.

In this paper we discuss the advantages and implications of using framed quadtrees with D*. We present results of the path planner operating in simulated fractal worlds and results from implementation on an autonomous jeep.

2 Map Representation

Apart from the fact that discretization of space allows for control over the complexity of path planning, it also provides a flexible representation for obstacles and cost maps, and eases implementation. One method of cell decomposition is to tessellate space into equal sized cells each of which is connected to its neighbors with four or eight arcs. This method, however, has two drawbacks: resulting paths can be suboptimal and memory requirements high. Quadtrees address the latter problem, while framed quadtrees address both problems, especially in sparse environments.

2.1 Regular Grids

Regular grids represent space inefficiently. Natural terrains are usually sparsely populated and are often not completely known in advance. In the absence of map information unknown environments are encoded sparsely during initial exploration and many areas remain sparsely populated even during execution. Many equally sized cells are needed to encode these empty areas making search expensive since more cells are processed than actually needed. Moreover, regular grids allow only eight angles for direction, resulting in abrupt changes in path direction and an inability, in some cases, to generate a straight path through empty areas (Fig. 1a). It is possible to smooth such jagged paths, but there is no guarantee that the smoothed path will converge to the truly optimal path.

2.2 Quadtrees

One way to reduce memory requirements is to use a quadtree instead of a regular grid. A quadtree [8][9] is based on the successive subdivision of region into four equally sized quadrants. A region is recursively subdivided until either a subregion free of obstacles is found or the smallest grid cell is reached. Quadtrees allow efficient partitioning of the environment since single cells can be used to encode large empty regions. However, paths generated by quadtrees are suboptimal because they are constrained to segments between the centers of the cells. Fig. 1b shows an example path generated using a quadtree.

2.3 Framed Quadtrees

To remedy the above problem, we have used a modified data structure in which cells of the highest resolution are added around the perimeter of each quadtree region. This augmented representation is called a *framed quadtree*. A path generated using this representation is shown in Fig. 1c. The small grey rectangles around the cells are the border cells of each quadrant. This representation permits many angles of direction, instead of just eight angles as in the case of regular grids. A path can be constructed between two border cells lying far away from each other. Most importantly, the paths generated more closely approximate optimal paths.

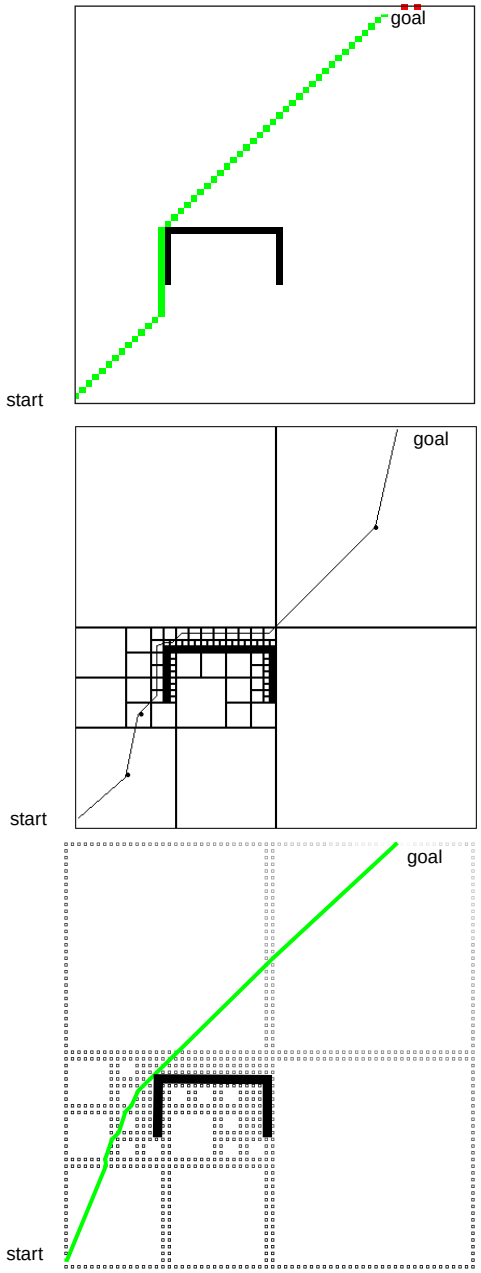

Fig. 1 An example of a path generated using (a) regular grid representation, (b) quadtree, (c) framed-quadtree. The black cul-de-sac is an obstacle.

The drawback of using framed quadtrees is that they can require more memory than regular grids in uniformly, highly cluttered environments because of the overhead involved in the book-keeping.

3 Incremental Planning

Unstructured outdoor environments are often not only sparse but also typically only partial maps are available. If complete and accurate maps were available, it would be sufficient to use A* [7] once to search the map and produce a path. The robot could simply follow this path during its traverse. Furthermore, errors in control and perception often introduce erroneous and changing information. Ideally, the robot should gather new information about the environment, and efficiently replan new paths based on this new information. In these partially known environments, a traverse can be achieved by incrementally incorporating information as it becomes available.

The idea is to produce a path based on all available information and replan from the current position to the goal every time new information is discovered. This is called "Best Information Planning". This approach [13] is intuitively satisfying and has been shown to produce lower-cost traverses on average than other selected algorithms for unknown and partially-known environments. Furthermore, Best Information Planning is able to make use of prior information to reduce the traversal cost.

Obviously, we can just use A* to replan a new path every time it is needed, but this approach is computationally expensive. Our approach is to use the incremental planner, D* [10][11], that allows replanning to occur in realtime. Incremental replanning makes it possible to greatly reduce computational cost, as it only updates the path locally, when possible, to obtain the globally optimal path. D* produces the same results as planning from scratch with A* for each new piece of information, but for large environments it is hundreds of times faster.

4 Simulation Results

We have run simulations to compare the performance of D* when used with framed quadtrees as opposed to regular grids in incrementally discovered environments. The simulation environment is a binary 256 x 256 cell world with obstacles (each 1 x 1 cell) distributed by a fractal terrain generator.

Two sets of simulations were done. In the first set, the environment is completely known in advance, that is, a perfect a priori map is assumed. In the second set none of the obstacle cells are known— all obstacles must be discovered by vehicle sensors. In the simulated world, the vehicle is able to detect obstacles within a radius of 4 cells. For each of these sets, we varied the density of the fractal obstacles and for each of the ten fractal densities, we created 100 simulated worlds. Hence each data point shown in the graphs below is the mean value obtained from 100 runs. Fig. 2 shows traverses generated by the use of a regular grid and a framed quadtree in a world that is completely unknown to the planner when the vehicle starts.

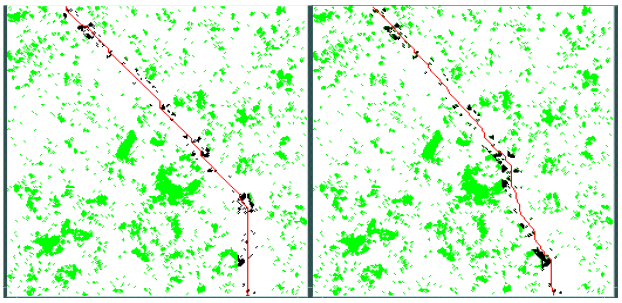

Fig. 2 Traverses generated in a fractal world using regular grids (left) and frame quadtrees (right). The lighter cells represent occupied areas that are unknown in advance. The dark cells represent the obstacles that are discovered by the vehicle's sensors. This world has a fractal gain of 12.

The more the world is cluttered, the less advantage is provided by the use of framed quadtrees. For instance, Fig. 3 shows a traverse generated in a densely populated fractal world. Although the traverses generated are more natural, the difference in the traverse length over a regular grid is small.

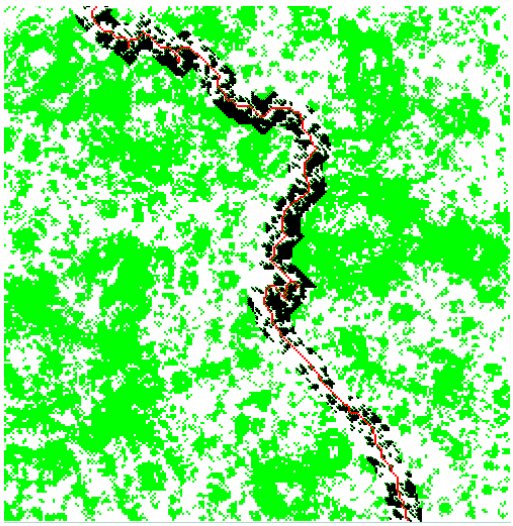

Fig. 3 Traverse in a dense environment (fractal gain=25) using a framed-quadtree, starting with no knowledge of the obstacles.

Below we compare simulation results using three criteria: traverse length, memory usage, and execution time.

4.1 Traverse Length

Traverse length is measured in cell units. Horizontal and vertical traverses between smallest cells count as 1 unit, while diagonal traverses count as $\sqrt{2}$ units. Traverses through a large empty area (that is, across a large framed-quadtree cell) count as the distance between the centers of the starting and ending border cells. Fig. 4 shows the comparison. The fractal gain controls the span and the density of fractal area. The larger the fractal gain, the wider and denser the fractal area. For simplicity, we will use fractal gain and fractal density to denote the same notion.

As expected, traverses in known worlds are shorter than in unknown worlds because in the latter case, newly sensed obstacles force replanning of the path. Traverses generated using framed quadtrees are shorter than those generated when regular grids are used primarily because the path is not forced to travel on diagonals. This effect is particularly noticeable when the environment is sparse or unknown. Correspondingly, the difference in traverse length is smaller as the fractal environments become denser.

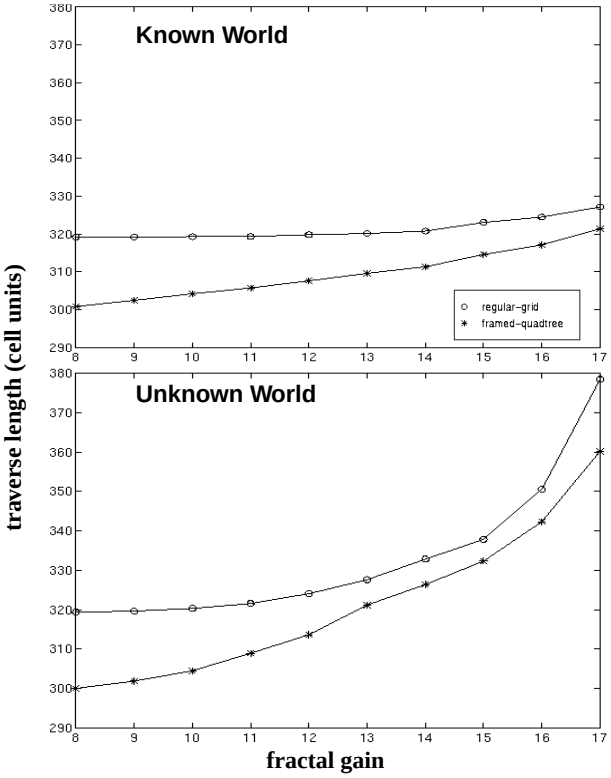

Fig. 4 Traverse length comparison in the fully-known and the completely unknown worlds as a function of fractal gain.

4.2 Memory Usage

Memory usage is the maximum memory (in bytes) used by the program during a run (Fig. 5). For known worlds, as the fractal gain increases, framed quadtrees use more memory to encode the environment than regular grids. As the result, beyond a certain fractal density, framed quadtrees require more memory. For unknown worlds, framed quadtrees use less memory in the range of densities with which we have experimented. Comparing known and unknown worlds, framed quadtrees use much less memory when the world is unknown because the world is assumed to be empty initially, allowing framed quadtrees to use large cells.

Our experimental results show that for unknown worlds, framed quadtrees reduces memory usage by over 40%. For sparse known worlds below fractal gain of 10, framed-quadtrees reduces memory usage by over 30%.

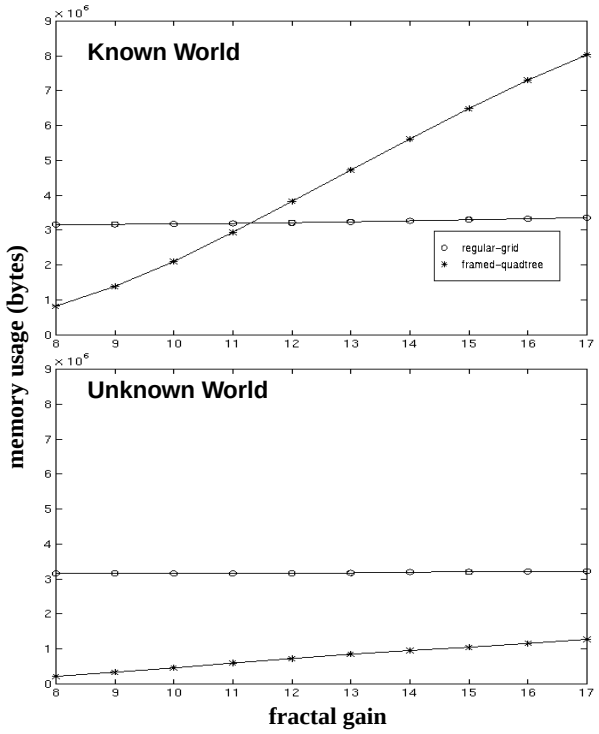

Fig. 5 Memory usage in the fully known and the completely unknown worlds.

4.3 Execution Time

We have compared the execution time required by D* when a framed quadtree is used versus when a regular grid is used. Results are summarized in Fig. 6. Total time is measured in seconds and is the sum of off-line time and on-line time. Off-line time measures the time needed to create the map data structure and to produce the initial path before the vehicle starts moving. Hence in the completely known world, the full path can be planned right away. When the world is unknown, the vehicle finds obstacles as it moves and the total time required includes the off-line time as well as the time necessary to incorporate new information.

In the fully known environment, use of a framed quadtree reduces the execution time below a certain fractal density over the use of regular-grids. For a completely unknown environment, the total time is consistently lower when framed quadtrees are used but the on-line time is higher due to increased overhead in maintaining a complex data structure. For completely known environments, regular-grid D* propagates costs to fewer cells as the fractal density increases, resulting in reduced off-line time, while framed-quadtree D* must create more subcells, resulting in larger off-line time.

For an unknown environment, the off-line time represents a time to propagate D* values in an empty initial environment. This explains why the off-line times for both repre-

sentations are almost constant with respect to the fractal density. However, the use of framed quadtrees significantly reduces the off-line time over regular grids, since very few cells are needed the represent the assumed free space. Some of this savings is lost during the on-line phase, as the algorithm builds the quadtree, but in general the total time is less since the framed quadtree is only partially constructed.

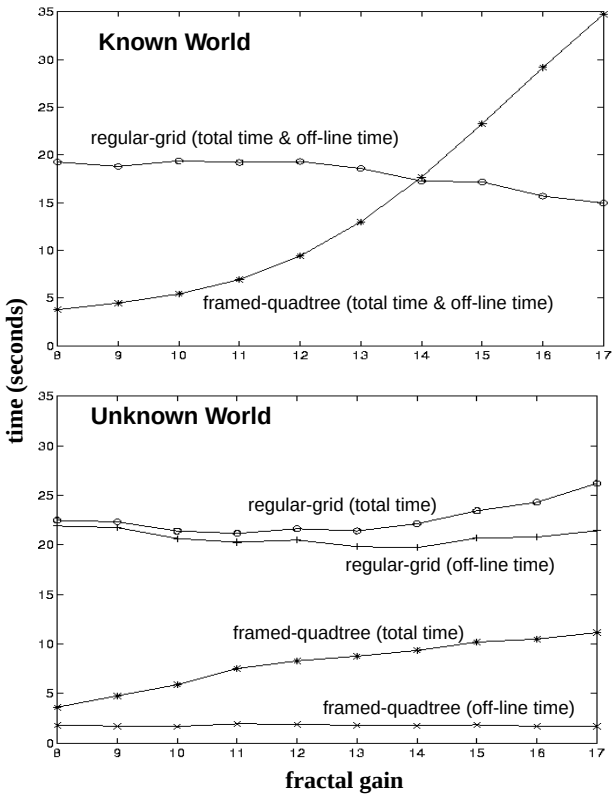

Fig. 6 Time comparison in the fully known and the completely unknown worlds.

5 Test Results on Autonomous Vehicle

We have performed several tests on an automated military jeep (Fig. 7). Our vehicle uses a vertical-baseline stereo system to generate range images. The resulting images are processed by the SMARTY local navigator [4], which handles local obstacle detection and avoidance. This obstacle map is fed to a global navigator running a path planning algorithm, such as framed-quadtree D*. Both the local and global navigators submit steering advice to an arbiter, which selects a steering command each time interval and passes it to the controller [12]. Fig. 8 shows the system modules and data flow.

The first set of tests shows that use of framed quadtrees remedies one drawback of regular grids— the inability to drive a straight diagonal traverse, in certain cases, through an empty area (Fig. 9). As can be seen, use of a framed

Fig. 7 The autonomous vehicle (HMMWV) that used for our experiments.The vehicle is equipped with stereo vision, inertial guidance and GPS positioning.

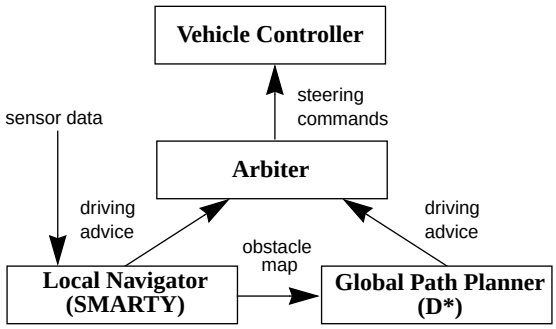

Fig. 8 Data flow in the implemented system.

quadtree results in a straight diagonal traverse, while use of a regular grid does not.

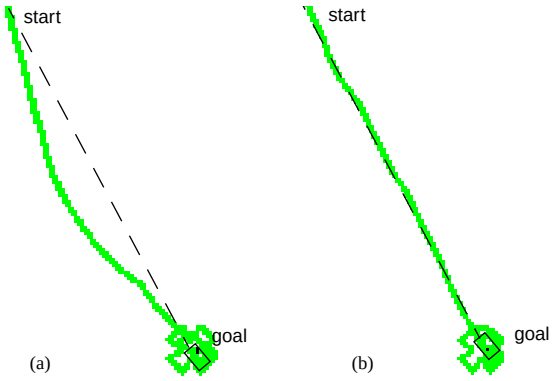

Fig. 9 Traverse of a free area using (a) regular grid (b) a framed quadtree

Fig. 10 shows a successful traverse of the vehicle that covered 200 meters in 6 minutes. During this traverse, the vehicle detected and avoided 80 obstacles.

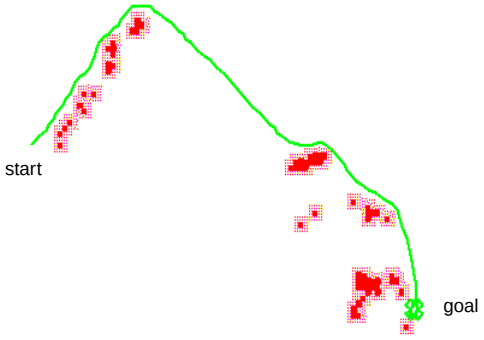

Fig. 10 Successful long traverse of the vehicle using framed-quadtree D* through a terrain with obstacles to the goal. The dark rectangles are obstacles detected and avoided during the traverse. The shaded areas surrounding the dark obstacles are potentially dangerous zones.

Fig. 11 shows a close-up of the data structure produced after the above run. As expected, a large part of the environment that is not explored is represented by a small number of cells.

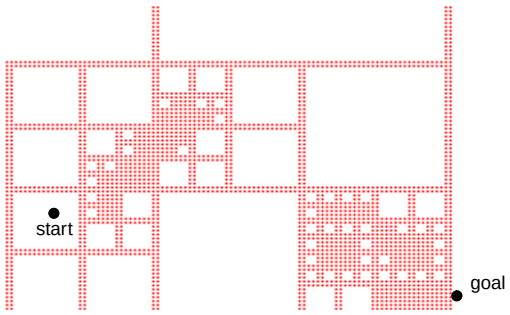

Fig. 11 A close up of the data structure produced from the execution of the path in Fig. 10.

6 Conclusions

We have implemented a method for global path planning suited to autonomous vehicles. Our method combines the D* algorithm, which allows dynamic path replanning in real time, and a framed quadtree data structure, which allows efficient spatial representation.

The results for sparse and unknown worlds are encouraging, giving us shorter traverse lengths when framed quadtrees are used as opposed to regular grids especially when the environment is sparse or when the map has to be built incrementally. Future work will extend the representation to continuously varying worlds as opposed to the binary worlds considered to date. Additionally, we will show the utility of this method in partially known environments, that is where maps exist but at a coarser scale.

Acknowledgments

The authors would like to thank Dr. Martial Hebert and Dr. Bruce Digney for their help with vehicle field tests, and, Daniel Huber and Stewart Moorehead for their helpful comments. This research was sponsored in part by DARPA, under contract DAAE07-96-C-X075 “Technology Enhancements for Unmanned Ground Vehicles” and by the Defense Research Establishment Suffield, Canada, under contract W7702-6-R577/001 “Performance Improvements for Autonomous Cross-Country Navigation”.

References

[1] Brumitt, B.L., Stentz, A., “Dynamic Mission Planning for Multiple Mobile Robots,” Proceedings of the IEEE International Conference on Robotics and Automation, May 1996.

[2] Chen, Szczerba, Uhran, “Planning Conditional Shortest Paths through an Unknown Environment: A Framed-Quadtree Approach,” Proceedings of the IEEE International Conference on Robotics and Automation, May 1995.

[3] Connolly and Grupen, “The Application of Harmonic Functions to Robotics,” Journal of Robotic Systems, 10(7):931-946.

[4] Hebert, Martial H., “SMARTY: Point-Based Range Processing for Autonomous Driving,” *Intelligent Unmanned Ground Vehicle*, Martial H. Hebert, Charles Thorpe, and Anthony Stentz, editors, Kluwer Academic Publishers, 1997.

[5] Kelly, A, “An Intelligent Predictive Control Approach to the High Speed Cross Country Autonomous Navigation Problem,” Ph.D Thesis, 1995, Carnegie Mellon University, Pittsburgh, PA 15213.

[6] Lengyel, J. and Reichert, M. and Donald, B. R. and Greenberg, D. P., “Real Time Robot Motion Planning Using Rasterizing Computer Graphics Hardware,” In Proc. SIGGRAPH. 1990.

[7] Russell, S., Norvig, P., *Artificial Intelligence: A Modern Approach*, Prentice Hall, 1995.

[8] Samet, H., “Neighbor Finding Techniques for Images Represented by Quadtrees,” Computer Graphics and Image Processing 18, 37-57, 1982.

[9] Samet, H., “An Overview of Quadtrees, Octrees, and Related Hierarchical Data Structures,” NATO ASI Series, Vol. F40, 1988.

[10] Stentz, A., “The Focussed D* Algorithm for Real-Time Replanning,” Proceedings of the International Joint Conference on Artificial Intelligence, August 1995.

[11] Stentz, A., “Optimal and Efficient Path Planning for Partially-Known Environments,” Proceedings of the IEEE International Conference on Robotics and Automation, May 1994.

[12] Stentz A., Hebert, M., “A Complete Navigation System for Goal Acquisition in Unknown Environments,” Autonomous Robots, 2(2), 1995.

[13] Stentz, A., “Best Information Planning for Unknown, Uncertain, and Changing Domains,” AAAI-97 Workshop on On-line-Search.

[14] Whitcomb, L. L. and Koditschek, D. E. “Automatic Assembly Planning and Control via Potential Functions,” In Proc. IEEE/RSJ International Workshop on Intelligent Robots and Systems. 1991.