A Distributed and Optimal Motion Planning Approach for Multiple Mobile Robots

Yi Guo and Lynne E. Parker

Abstract— **We propose a distributed and optimal motion planning algorithm for multiple robots. The computationally expensive problem is decomposed into two modules – path planning and velocity planning. The D^* search method is applied in both modules, based on either geometric formulation or schedule formulation. Optimization is achieved at the individual robot level by defining cost functions to minimize, and also at the team level by a global measurement function reflecting performance indices of interest as a team. Contrary to our knowledge of previous results on multi-robot motion planning that either obtain optimal solutions through centralized and exhaustive computing, or achieve distributed implementations without considering any optimization issues, our approach combines these two features and explicitly optimizes performance functions through a distributed implementation. It is also one of the few that is capable of handling outdoor rough terrain environments and real time replanning. Simulations are shown on a Mars-like rough terrain using a 3D vehicle planner and control simulator. The algorithm was also implemented and successfully run on a group of Nomad 200 indoor robots.**

Keywords— **Multi robots, motion planning, performance index, cost function, coordination diagram.**

I. Introduction

Motion planning algorithms for single mobile robot systems have been intensively discussed for years (see e.g., the survey book [16]). In an environment that contains a set of stationary obstacles, path planning methods such as graph searching based on geometric configuration of the environment guarantee to return optimal paths (in the sense of a performance measure such as shortest distance) in polynomial time if one exists. However, motion planning in a dynamic environment with moving obstacles is inherently harder. Even for a simple case in two dimensions, the problem is NP-hard and is not solvable in polynomial time ([10], [13]). This fact makes multiple mobile robot motion planning a difficult problem. On the other hand, broad applications in cooperative mobile robotics call for practical and efficient motion planning strategies (see [24] and references therein). Simple reactive motion planning strategies cannot guarantee deadlock-free and convergence even in simple cases. A number of algorithms have been proposed towards finding collision-free and deadlock-free paths, but most of these algorithms work in limited domains, and are difficult to evaluate quantitatively due to the great variation of the premises and conditions in each approach ([1]). This paper makes contributions in proposing a practical motion planning approach with combined features of previous approaches. Contrary to our knowledge of previous results on multi-robot motion planning that either obtain optimal solutions through centralized and exhaustive computing, or achieve distributed implementations without considering any optimization issues, our approach combines these two features and explicitly optimizes performance functions through a distributed implementation. It is also capable of handling outdoor rough terrain environments and real time replanning, which have not been covered much in previous literatures, and are appealing to applications in areas such as surface mining and space exploration.

While a complete review of existing work on multi-robot motion planning is not possible in limited space, we mention a few projects to indicate where our approach stands. Motion planning in dynamic environments was originally addressed by adding the time dimension to the robot's configuration space. The approach in [8] discretizes the configuration-time space to a sequence of slices of the configuration space at successive time intervals, and represents the motions of moving obstacles using the set of slices embodying space-time. In [22], moving obstacles are represented as sheared cylinders, and a methodology was proposed to provide optimal tangent paths to the goal for a dynamic robot environment. Another approach to dynamic motion planning was proposed in [14] which decomposes the problem into smaller subproblems: *path planning* and *velocity planning*. The complex trajectory planning problem is then transformed to planning a velocity profile for a fixed path obtained from an earlier static path planning step. The decomposition of path and velocity planning provides a solution through the complexity barrier caused by the additional time

The authors are with the Center for Engineering Science Advanced Research, Computer Science and Mathematics Division, Oak Ridge National Laboratory, Oak Ridge, TN 37831. Email: guoy@ornl.gov, parkerle@ornl.gov .

This article has been authored by contractors of the U. S. Government under Contract No. DE-AC05-00OR22725 with UT-Battelle, LLC. The U. S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U. S. Government purposes. Research sponsored by the Engineering Research Program of the Office of Basic Energy Sciences, U. S. Department of Energy.

dimension, and also provides an opening that has applications in robotics where robots move along fixed paths. A number of studies have been made based on path-velocity decomposition. In [18], a similar idea was used to plan motions of two robots. The authors of [11] take uncertainty of the moving obstacles into account while using the same principle for path planning. The approach in [21] uses the concept of traversability vectors to analyze the spatial relationship between the robot and moving obstacles, and develops a search algorithm to coordinate the robot motion. Path coordination schedules, which are another form of velocity planning, are studied in [3], [19], [20]. In this paper, we follow the principle of path-velocity decomposition design, and furthermore achieve a decentralized implementation and performance optimization.

The existing multi-robot motion planning algorithms are often categorized as centralized or decentralized ([1]), according to the information handling structure among robots. Examples of centralized planning work are [8], [9], [25], [29], among which [9], [29] assign priorities to robots in advance. In decentralized planning, each robot plans individually for itself by means of collecting information from other robots and environmental information around the robot. Decentralized planning work includes [2], [6], [15], where [15] applies traffic rules and is suitable for the route network; [2] uses dynamic priority assignment and negotiation to solve the coordination and conflict problem; and [6] randomly chooses one robot to stop and inserts random time delays to resolve the potential collision. Our algorithm is a decentralized approach; moreover, certain performance optimization is introduced to autonomously coordinate trajectory conflicts.

Optimal motion planning is studied in [3], [4], [7], [17], [26] and references therein. In [3], [4], [7], [26], time-optimal trajectory planning algorithms are proposed for two robotic manipulators to avoid collisions. Usually, the physical models of robotic arms are used, which makes the results not directly extendable to mobile robots. The research in [17] considers multiple robots with independent goals and performance measures, and proposes algorithms optimizing a scalarizing function which is a weighted-average of individual performance functions. However, the weighted-average of individual performance does not always ensure efficiency of all robots as a team. A global performance function should be introduced in this case.

We design a decentralized motion planning algorithm for multiple mobile robots. First, each robot plans its own path independently. Then, a coordination diagram, which represents an N-dimensional (N is the number of robots) space mapped from the 3D workspace using path length as the parameter, is constructed based on collision checks among all robot paths. A search algorithm is then executed on the coordination diagram, and a velocity profile which minimizes a global performance function is chosen. D^* serves as the local search method. Due to the capability of 3D environments and real-time replanning of D^*, our decentralized motion planning algorithm can be used in partially known environments where online planning is required. In summary, our approach has the following combined features that are contributions over previous results:

- decentralized motion planning;
- capable of outdoor environment and real time replanning;
- a global performance measurement is defined and optimized.

It should be noted that the scope of the current paper is different from earlier works such as [5] using D^* search, where a mission planner layer is constructed to assign goals to each robot dynamically, and explicit methods for avoiding robot-robot collisions are not used.

The rest of the paper is organized as follows. In Section II, the multiple-robot motion planning problem is defined, and premises and assumptions are stated. Then in Section III the main algorithm is described. Implementation examples are given in Section IV, where 3D simulations in an outdoor terrain environment along with experiments using a group of Nomad indoor robots are described. Finally the paper is concluded with brief remarks in Section V.

II. Premises and Problem Statement

We focus on the multi-robot motion planning problem. Other issues such as the task planner are beyond the scope of this paper. Premises and assumptions of our study are stated as follows:

Assumptions:

1. Each robot has an assigned goal, and each robot knows its start and goal positions;

2. Robots operate in either indoor or outdoor environments, and have a pre-defined map. Specifically, in the indoor environment, the map defines the static polygonal obstacles in the environment; in the outdoor environment, the map defines terrain elevation and traversability based on a grid representation of the terrain;

3. Robots' onboard sensors detect the discrepancy between the pre-defined map and the environment, and revise the map online;

4. Robots are equipped with communication devices so that they can broadcast messages to others, and the communication is reliable;

5. A robot's motion control layer tracks pre-assigned trajectories within a small margin of error;

6. Robots move at constant fixed speeds;
7. Robots can switch instantaneously between a fixed speed and halting.

Assumptions 1 to 3 are standard assumptions for indoor/outdoor motion planning. Assumption 4 is often true as most robots have Ethernet based LANs. Though robot dynamics should be taken into account at some level, we decouple the general plan-and-control problem into two modules – motion planning (which is the topic of this paper) and trajectory control; therefore, Assumption 5 is required. Assumption 6 is a usual assumption (e.g., [6]). Assumption 7 is also a typical one used in multiple-robot motion planning literature, for example, [8], [14], [17].

The problem under concern is defined as follows:

Multiple-robot motion planning problem:

Based on Assumptions 1 to 7, find a sequence of traverse states for each robot $\mathcal{A}_i, (i = 1, \ldots, N)$ enrouting from its start position S_i to its goal position G_i, without collisions with static obstacles and each other, while minimizing the following global performance index:

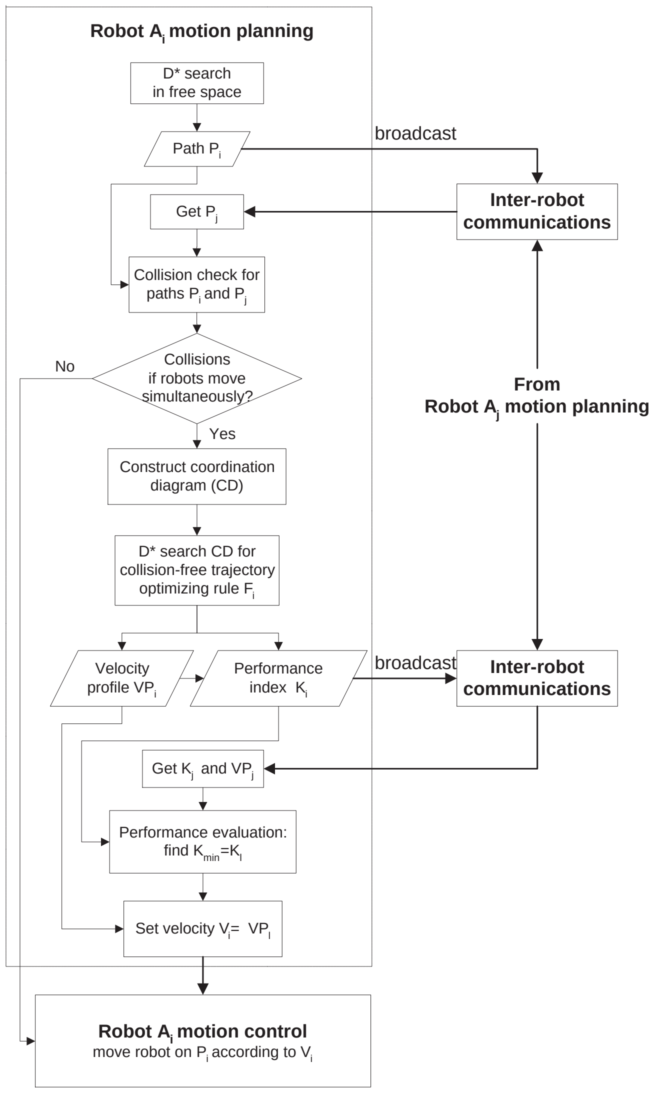
$$\Gamma = \gamma_1 \max(T_1, T_2, \ldots, T_N) + \gamma_2 \sum_{i=1}^{N} I_i \qquad (1)$$

where $T_1, T_2 \ldots, T_N$ are the times spent for each robot to reach its goal, I_i is the idle time for robot i, and γ_1, γ_2 are positive weighting constants.

By *traverse states*, we mean the pair of geometric paths (which is the shape of the curve in the robot's configuration space) and velocity profiles. By minimizing the global performance index (1), the solution provides a minimum time criterion, i.e., minimizing a weighted sum of the most expensive time to reach its goal and total idling time of all robots.

III. Multi-Robot Motion Planning Algorithm

The main flow chart of the algorithm is shown in Figure 1. First, each robot plans its own path independently using D^*. The path is broadcast to all other robots, so every robot knows all path information. Under our approach, the paths that are planned for each robot are fixed, i.e., the following steps will not alter the (x, y) sequences of the paths. Instead, we define velocity profiles so that, while robots follow their paths, they insert delays as required to avoid collisions. Once the paths are planned, the collision check is then executed. If the collision is time-space collision, that is, two or more robots reach the same point at the same time, an N-dimensional coordination diagram (CD) is constructed with collision regions marked as obstacles in the diagram. D^* searches for a free trajectory in the coordination diagram. The trajectory is then interpreted into a velocity profile for each robot, and the performance index of the current trajectory solution is calculated. Since the searching in CD is distributed across the robots, each search can take a different cost function to minimize based upon differences in priorities between robots at intersections. Then the performance index and velocity profile are broadcast to all other robots. An evaluation is done to get a minimum value of the performance index, and the corresponding velocity profile is chosen.



Robot A_i motion planning
D* search in free space
Path P_i
broadcast
Inter-robot communications
Get P_j
Collision check for paths P_i and P_j
Collisions if robots move simultaneously?
No
Yes
From Robot A_j motion planning
Construct coordination diagram (CD)
D* search CD for collision-free trajectory optimizing rule F_i
Velocity profile VP_i
Performance index K_i
broadcast
Inter-robot communications
Get K_j and VP_j
Performance evaluation: find $K_{min}=K_i$
Set velocity $V_i=$ VP_i
Robot A_i motion control
move robot on P_i according to V_i

Fig. 1. Flow chart of the multi-robot motion planning algorithm.

The following subsections discuss these modules in more detail.

A. D^* Search in Free Space

Assume robot $\mathcal{A}_i$ moves on a workspace $\mathcal{W} \subset \Re^3$, and the prohibited space is $\mathcal{W}_{prohibit} \subset \Re^3$. The prohibited space defines regions in which the robot should not travel due to mission constraints (e.g., minefields) or undetectable navigation challenges (e.g., quicksand). We define a safety margin by a positive constant ε, and $\mathcal{W}^{\varepsilon}_{prohibit}$ is the space that is within ε distance of $\mathcal{W}_{prohibit}$. Denote the valid search space to be $\mathcal{W}^{\varepsilon}_{free} = \mathcal{W} \setminus \mathcal{W}^{\varepsilon}_{prohibit}$. The path searching task is to find a sequence of geometric points from the start to the goal in $\mathcal{W}^{\varepsilon}_{free}$ according to the existing map defined on the regular grid.

The D^* search algorithm, which is a dynamic version of A^*, was proposed in [27], [28]. It produces an optimal path from the start position to the goal in the sense of minimizing a pre-defined cost function. It has the capability of rapid replanning, and has been used in real time planning in partially known environments with challenging terrains. As in A^*, its efficiency is highly determined by the chosen cost function. The cost function in an indoor environment can be simple. However, it can be very complicated in a 3D outdoor environment, due to the complexity of the terrain and traversal capability of the vehicle. The cost function we use for D^* path planning is the following:

$$f_{pp} \quad = \quad \rho + \alpha_1 d + \alpha_2 s + \alpha_3 t \qquad (2)$$

where ρ is a large value if there is any obstacle penetrated by the path, and 0 otherwise; d is the geometric distance; s is the slope of the terrain; t is the penalty for turning; and $\alpha_1, \alpha_2, \alpha_3$ are positive weighting factors, where $\|(\alpha_1, \alpha_2, \alpha_3)\| = 1$. Such a cost function guarantees that D^* returns an optimal path that avoids static obstacles, and is the shortest, flattest, smoothest possible path if one exists. Note that the last two terms are only valid in outdoor and rough terrain environments.

The output of this module is a path P_i for robot $\mathcal{A}_i$, which consists of a sequence of geometric points the robot is to pass through, in the resolution of the grid of the environmental map. Path P_i is then broadcast to other robots.

B. Collision Check and Coordination Diagram

After robot $\mathcal{A}_i$ obtains its own path P_i and all other paths $P_j, (j = 1, 2, \ldots, N, j \neq i)$, it executes a collision check procedure, which returns all collision regions. Since the configuration space is on a regular grid representation, the collision region is represented by sets of (x, y) pairs at which path intersections occur. Note that the collision region should be enlarged by the radius of the robot plus a certain safe margin.

Similar to the procedure used in [14], [17], we map each path in $\mathcal{W}^{\varepsilon}_{free}$ to a one-dimensional trajectory based on path length. That is, each path P_i can be seen as a continuous mapping $[0, l] \to \mathcal{W}^{\varepsilon}_{free}$, where l is the path length. Without loss of generality, assume that the path P_i is followed at constant speed. Let $\mathcal{S}_i = [0, l]$ denotes the set of points that place the robot along the path P_i. We combine all these mappings into an N-dimensional coordination space. That is, the path coordination space is defined as $\mathcal{S} = \mathcal{S}_1 \times \mathcal{S}_2 \times \ldots \times \mathcal{S}_N$, and the *coordination diagram* is an N-dimensional diagram representing the path coordination space. By an inverse mapping, a point in the N-dimensional coordination diagram determines the position along the path of each robot in the team.

C. Search in Coordination Diagram

To search in the coordination diagram (CD), first the collision regions in $\mathcal{W}^{\varepsilon}_{free}$ are mapped into the path coordination space as static obstacles. Since the path coordination space is parameterized by the non-decreasing path length, the preferred movement in CD should be non-decreasing. So the search objective is to find a non-decreasing curve that connects the lower left corner of the diagram $(0, 0, \ldots, 0)$ to the top right corner $(l_1, l_2, \ldots, l_N)$ avoiding penetration into the static obstacles. We call such a free curve a *trajectory*.

The computational expense is reduced by the non-decreasing constraint of the search. At each grid point, $2^N - 1$ action combinations are considered. Although the complexity is exponential in the number of robots, the algorithm is efficient for a fixed N.

It is clear that a short path in CD corresponds to a fast trajectory in $\mathcal{W}$. To resolve conflicts based on delays within fixed paths, it is necessary to define priorities among robots that indicate which robot should stop at a certain point. The cost function chosen on each robot can be slightly different, based on which robot has priority at intersections. That is, on robot $\mathcal{A}_i$, schemes are evaluated by giving $\mathcal{A}_i$ top priority and assigning other robots to give way to robot $\mathcal{A}_i$ at the intersections, which is achieved by putting a penalty function in the cost function. The cost function for D^* velocity planning is chosen to be:

$$f_{vp} \quad = \quad \varrho + \beta_1 d + \beta_2 t_{idle} + \beta_3 p \qquad (3)$$

where ϱ is a large value if there are any collision regions penetrated by the trajectory, and 0 otherwise; d is the N-dimensional Euclidean distance; t_{idle} is the total idle time for all robots; p is the penalty if robot $\mathcal{A}_i$ has to give way to others; and $\beta_1, \beta_2, \beta_3$ are positive weighting factors, where $\|(\beta_1, \beta_2, \beta_3)\| = 1$.

An example search result in a 3-dimensional CD is shown in Figure 2, where the shaded areas denote intersection regions in $\mathcal{S}$.

The returned trajectory in CD is then interpreted as a robot velocity profile, which is denoted as VP_i. Note that VP_i contains the set of velocity profiles for all N robots. The time for each robot to reach its goal and the idle time during the transit can then be calculated according to its velocity profile within VP_i. The performance index K_i can be obtained according to the definition of Γ in (1).

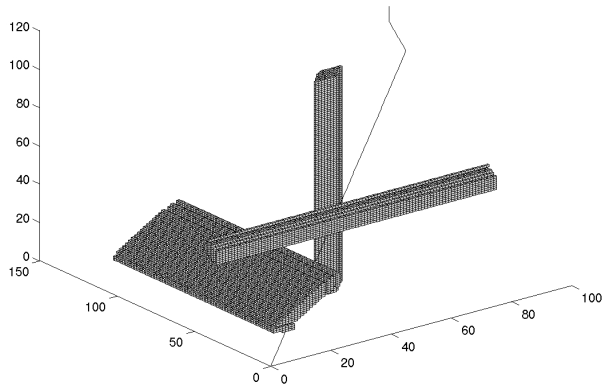


Fig. 2. A search result in a coordination diagram.

D. Selecting an Optimal Solution

Once the velocity profile $VP_i(i = 1, \dots, N)$ and performance index $K_i(i = 1, \dots, N)$ are obtained on each robot, they are broadcast among all robots. An evaluation is then done to compare which K_i is the smallest. If $K_l(1 \leq l \leq N)$ is the smallest, then VP_l is selected, and each robot is assigned the velocity profile within the set of VP_l. The robots then execute their paths according to the derived velocity profiles.

IV. Implementation Examples

A. A 3D Simulation in Outdoor Environment

The proposed algorithm has been implemented in a 3D vehicle planner and control simulation environment. The path planning (D^* search in workspace) on a 160×160 grid took 1 second on a Sun UltraSPARC 60 workstation. Figure 3 shows the planned paths for three robot vehicles on a 3D Mars-like terrain. Based on these paths, if the three robots move simultaneously, collisions will occur. Therefore velocity planning is necessary to resolve potential collisions. The velocity planning (D^* search in coordination diagram) on the $117 \times 95 \times 99$ grid took about 4 minutes. No consideration was given to reduce computation time in the software implementation. We choose constants $\gamma_1 = \gamma_2 = 1$ in (1). The performance indices returned from the three robots are $119dt, 120dt$, and $119dt$ respectively, where dt is the unit time that the robot moves a unit distance. Correspondingly, the velocity profiles show that the first solution is to insert three unit time delays for robot 2 at the beginning of its movement, the second solution is to insert four unit time delays for robot 3 at the beginning of its movement, and the third solution is identical to the first. Remember that the differences in schedules are caused by assigning different priorities to robots. Since the first index is the smallest, the corresponding set of velocity profiles are chosen for each robot, as shown in Figure 4. It should be noted that although more complicated velocity profiles (with many stop-move schedules in the middle of the velocity profile) can be generated by the described algorithm, from a practical concern, based on the same or a comparable performance index value, it is preferred to have delays at the beginning of the velocity profiles, or to be consolidated, instead of requiring a lot of move-stop-move procedures during the robot movement. This can be achieved by smoothing zig-zag paths in the searching algorithm.

Experimental preparation is underway to implement this algorithm on a group of ATRV-mini all-terrain mobile robots. An experimental scenario with two ATRV-mini robots is shown in Figure 5.

B. Nomad 200 in an Indoor Environment

The proposed algorithm has also been implemented and tested on a group of Nomad 200 indoor robots, both in simulations and on physical robots. The simulation results are shown in Figure 6, in which the left map window shows robot start positions and goal positions, and the right three windows show the planned paths for each robot independently. The path planning (D^* search in workspace) on a 90×90 grid took 0.5 seconds on a Sun UltraSPARC 60 workstation; and the velocity planning (D^* search in coordination diagram) on the $93 \times 99 \times 112$ grid took about 2 minutes. No consideration was given to reduce computation time in the software implementation. The velocity profile is shown in Figure 7. The result shows that by inserting six unit time delays for robot 2 and two unit time delays for robot 3 at the beginning of their movements, the robot team achieves a minimum sum of the most expensive time (to reach robot goals) and total idling time (of all robots). Here the idling time does not include the time after the robot reaches its goal, as it can be assigned to other tasks as long as it reaches its goal.

In the physical robot experiments, encoders were used for localization. We take consideration of the

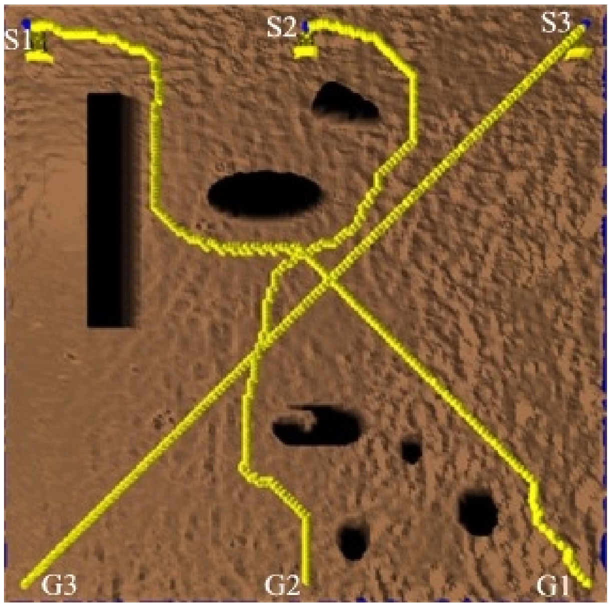


Fig. 3. 3D simulation results in an outdoor environment with a Mars-like terrain. Black areas are untraversable terrain. S1,S2,S3 denote the start positions of each robot respectively, and G1,G2,G3 denote the goal positions.

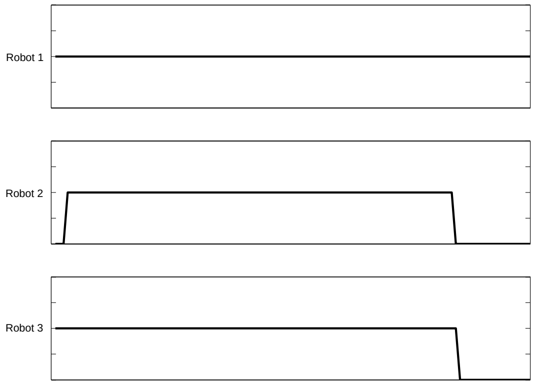


Fig. 4. Velocity profile of 3D simulations.

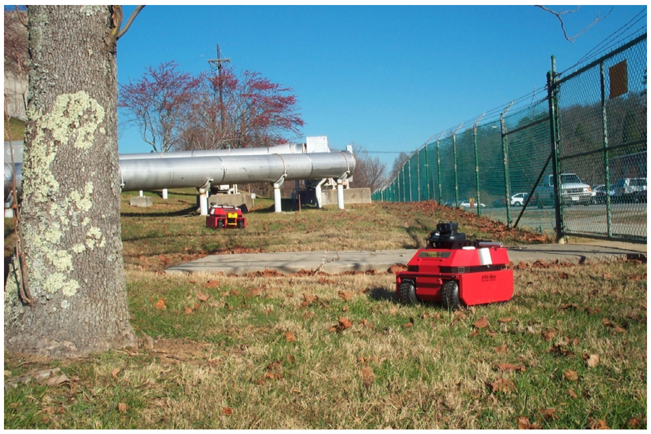
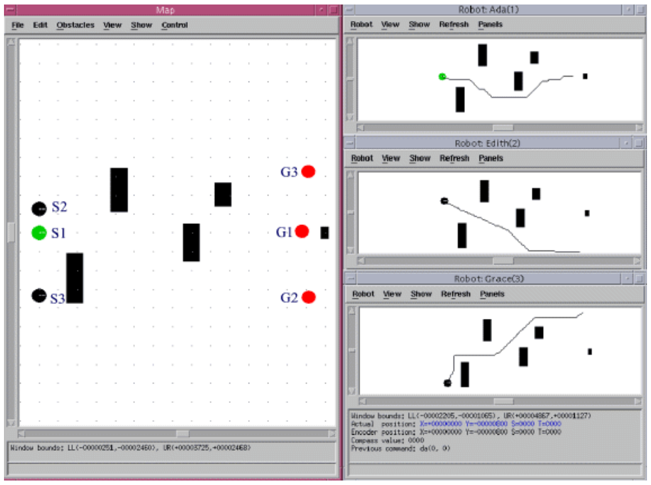


Fig. 5. ATRV-Mini all-terrain mobile robots in outdoor natural environment.


Map
File Edit Obstacles View Show Control
S2
S1
S3
G3
G1
G2
Robot: Ada(1)
Robot View Show Refresh Panels
Robot: Edith(2)
Robot View Show Refresh Panels
Robot: Grace(3)
Robot View Show Refresh Panels

```
Window bounds: LL(-00000251,-00002460), UR(+00003725,+00002468)
```

```
Window bounds: LL(-00002205,-00001065), UR(+00004867,+00001127)
Actual  position: X=+0000000 Y=-0000800 S=0000 T=0000
Encoder position: X=+0000000 Y=-0000800 S=0000 T=0000
Compass value: 0000
Previous command: da(0, 0)
```

Fig. 6. Simulation results on Nomad 200 indoor robots. Here, S1,S2,S3 denote the start positions of each robot respectively, and G1,G2,G3 denote the goal positions.

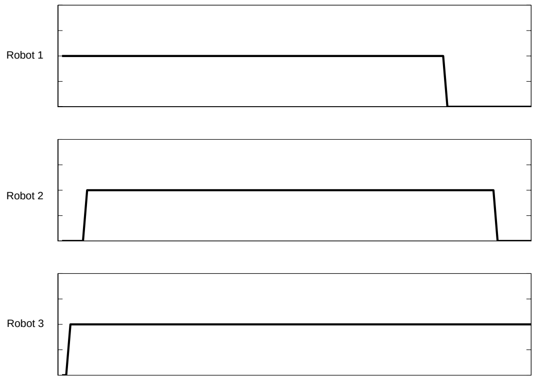


Fig. 7. Velocity profile of Nomad 200 simulations.

localization error into the path planning design by enlarging the safe margin in the D^* search in workspace. Also, the motion of the robot includes other uncertainties. For example, it does not take a unit time for the robots to track a unit distance, and the robot does not switch instantaneously between moving and stopping. We take this into consideration by putting a small amount of uncertainty margin into the D^* search in coordination diagram.

Figure 8 shows the pre-defined map of the environment, and Figure 9 shows the robots (named Edith, Alexandra and Ada) at their start positions in the 9.14×7.31 square meters experimental area. Figure 10 shows the robots in motion and avoiding each other when traveling on their assigned paths. The move-stop schedule for each robot is shown in Figure 11, and the encoder trajectories at the end of the experiment are shown in Figure 12. The robots successfully moved to their goals while avoiding collisions. This experiment demonstrates that our algorithm is robust with respect to certain degrees of localization and motion uncertainties.

V. Conclusions

We have described a decentralized and optimal motion planning algorithm for multiple robots. The computationally expensive problem is decomposed into two modules for path planning and velocity planning. D^* searching is applied, and the proposed algorithm is capable of operating in partially known and outdoor environments with rough terrain. A global performance measurement is introduced to minimize a weighted sum of the most expensive time to reach the goals and all idle time. Individual performance optimization is achieved by choosing an appropriate cost function to minimize for the D^* search algorithm. The paper provides a general motion planning framework; different performance metrics can be chosen depending on the requirements of particular applications. The proposed algorithm was implemented both in simulations and on physical Nomad robots. Simulations were performed on a Mars-like rough terrain using a 3D vehicle planner and control simulator. Experiments were done on a group of Nomad 200 indoor robots, and satisfactory performances were observed. Future work includes an implementation of the algorithm on a group of ATRV-mini all-terrain mobile robots.

References

[1] T. Arai and J. Ota. Motion planning of multiple mobile robots. In *Proceedings of the 1992 IEEE/RSJ International Conference on Intelligent robots and systems*, pages 1761–1768, July 1992.

[2] K. Azarm and G. Schmidt. Conflict-free motion of multiple mobile robots based on decentralized motion planning and negotiation. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 3526–3533, 1997.

[3] Z. Bien and J. Lee. A minimum-time trajectory planning method for two robots. *IEEE Transactions on Robotics and Automation*, 8:414–418, 1992.

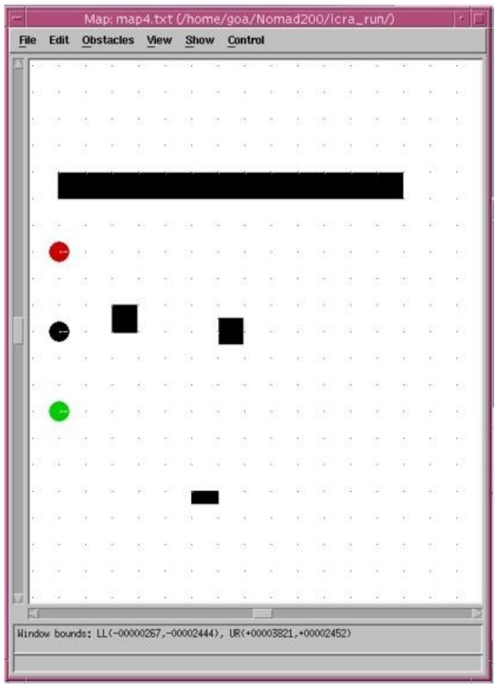
[4] J. E. Bobrow. Optimal robot path planning using the minimum-time criterion. *IEEE Transactions on Robotics and Automation*, 4(4):443–450, 1988.

[5] B. L. Brumitt and A. Stentz. Dynamic mission planning for multiple mobile robots. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 2396–1401, 1996.

[6] S. Carpin and E. Pagello. A distributed algorithm for multi-robot motion planning. In *Proceedings of the Fourth European Conference on Advanced Mobile Robotics*, Sept. 2001. to appear.

[7] C. Chang, M. J. Chung, and B. H. Lee. Collision avoidance of two general robot manipulators by minimum delay time. *IEEE Transactions on Robotics and Automation*, 24(3):517–522, 1994.

[8] M. Erdmann and T. Lozano-Perez. On multiple moving



```
Window bounds: LL(-00000267,-00002444), UR(+00003821,+00002452)
```

Fig. 8. Pre-defined experimental map and robots at start positions.

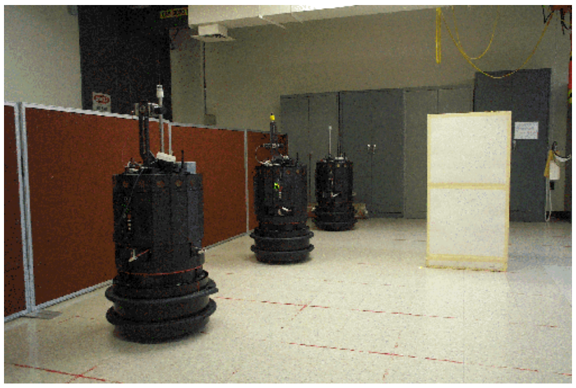


Fig. 9. Nomads are at their start positions.

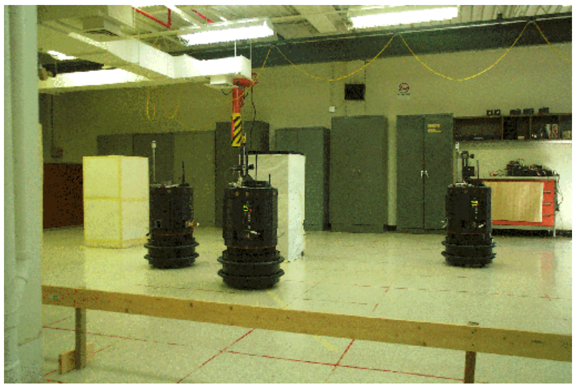


Fig. 10. Nomads are in motion on their planned paths.

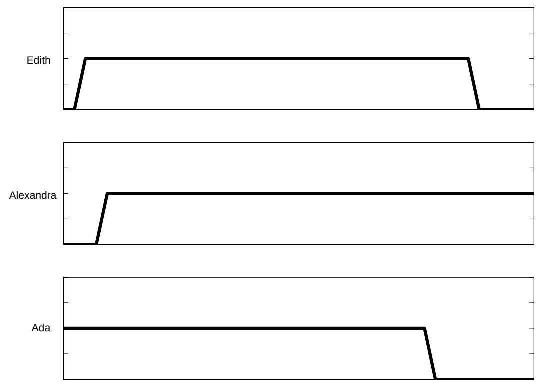
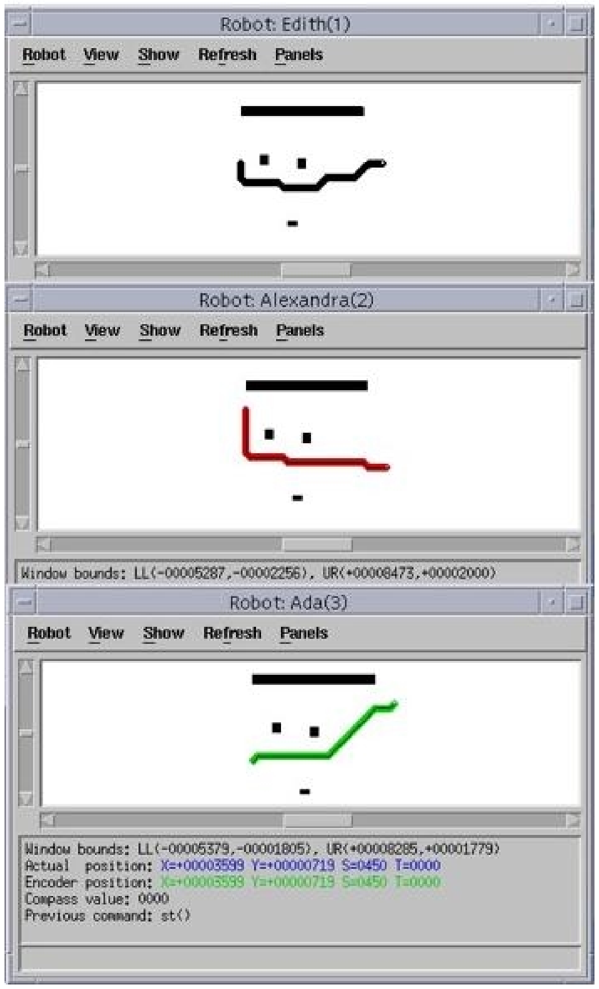


Fig. 11. Move-stop schedule for Nomads.


Robot: Edith(1)
Robot View Show Refresh Panels
Robot: Alexandra(2)
Robot View Show Refresh Panels
Robot: Ada(3)
Robot View Show Refresh Panels

```
Window bounds: LL(-00005287,-00002256), UR(+00008473,+00002000)
```

```
Window bounds: LL(-00005379,-00001805), UR(+00008285,+00001779)
Actual  position: X=+00003599 Y=+00000719 S=0450 T=0000
Encoder position: X=+00003599 Y=+00000719 S=0450 T=0000
Compass value: 0000
Previous command: st()
```

Fig. 12. Encoder records of trajectories on the physical robots.

objects. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 1419–1424, 1986.

[9] C. Ferrari, E. Pagello, J. Ota, and T. Arai. Multirobot motion coordination in space and time. *Robotics and Autonomous Systems*, 25:219–229, 1998.

[10] K. Fujimura. *Motion Planning in Dynamic Environment.* Computer Science Workbench. Springer-Velag, Tokyo, 1991.

[11] N. C. Griswold and J. Eem. Control for mobile robots in the presence of moving objects. *IEEE Transactions on Robotics and Automation*, 6(2):263–268, 1990.

[12] Y. Guo, Z. P. Jiang, and D. J. Hill. Decentralized robust disturbance attenuation for a class of large-scale nonlinear systems. *Systems & Control Letters*, 37:71–85, 1999.

[13] J. E. Hopcroft, J. T. Schwartz, and M Sharir. On the complexity of motion planning for multiple independent objects; PSPACE-Hardness of the "Warehouseman's Problem". *The International Journal of Robotics Research*, 3(4):76–88, 1984.

[14] K. Kant and S. W. Zucker. Toward efficient trajectory planning: the path-velocity decomposition. *The International Journal of Robotics Research*, 5(3):72–89, 1986.

[15] S. Kato, S. Nishiyama, and J. Takeno. Coordinating mobile robots by applying traffic rules. In *Proceedings of the 1992 IEEE/RSJ International Conference on Intelligent robots and systems*, pages 1535–1541, July 1992.

[16] J-C. Latombe. *Robot Motion Planning.* Kluwer Academic Publishers, 1991.

[17] S. M. LaValle and S. A. Hutchinson. Optimal motion planning for multiple robots having independent goals. *IEEE Transactions on Robotics and Automation*, 14:912–925, 1998.

[18] B. H. Lee and C. S. Lee. Collision-free motion planning of two robots. *IEEE Transactions on Systems, Man, and Cybernetics*, 17(1):21–32, 1987.

[19] S. Leroy, J. P. Laumond, and T. Simeon. Multiple path coordination for mobile robots: a geometric algorithm. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 1999.

[20] P. A. O'Donnell and T. Lozano-Perez. Deadlock-free and collision-free coordination of two robot manipulators. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 484–489, 1989.

[21] T. J. Pan and R. C. Luo. Motion panning for mobile robots in a dynamic environment. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 578–583, 1990.

[22] L. E. Parker. A robot navigation algorithm for moving obstacles. Master's thesis, The University of Tennessee, Knoxville, 1988.

[23] L. E. Parker. ALLIANCE: An architecture for fault tolerant multi-robot cooperation. *IEEE Transactions on Robotics and Automation*, 14:220–240, 1998.

[24] L. E. Parker, G. Bekey, and J. Barhen (Eds.). *Distributed Autonomous Robotic Systems 4.* Springer-Verlag, 2000.

[25] D. Parsons and J. Canny. A motion planner for multiple mobile robots. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 8–13, 1990.

[26] Z. Shiller and H. H. Lu. Robust computation of path constrained time optimal motions. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 144–149, 1990.

[27] A. Stentz. Optimal and efficient path planning for partially-known environments. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 3310–3317, 1994.

[28] A. Stentz. The focussed D* algorithm for real-time replanning. In *Proceedings of the International Joint Conference on Artificial Intelligence*, August 1995.

[29] C. W. Warren. Multiple robot path coordination using artificial potential fields. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 500–505, 1990.