Proceedings of the 2003 IEEE
International Conference on Robotics & Automation
Taipei, Taiwan, September 14-19, 2003

Planning and Control of UGV Formations in a Dynamic Environment: A Practical Framework with Experiments

Yongxing Hao[*], Benjamin Laxton[†], Sunil K. Agrawal[‡], Edward Lee[§], Eric Benson[¶]
Mechanical Systems Laboratory, Department of Mechanical Engineering
University of Delaware, Newark, DE 19716, USA

February 14, 2003

Abstract

This paper provides a practical framework for planning and control of formations of multiple unmanned ground vehicles to traverse between goal points in a dynamic environment. This framework allows online planning of the formation paths using a Dijkstra's search algorithm based on the current sensor data. The formation is allowed to dynamically change in order to avoid obstacles in the environment while minimizing a cost function aimed at obtaining collision-free and deadlock-free paths. Based on a feasible path for a virtual leader of the group, the trajectory planner satisfies the kinematic constraints of the individual vehicles while accounting for inter-vehicle collision and path constraints. A Lyapunov based controller is designed to keep the vehicles on their planned trajectories. Illustrative simulations of groups of unmanned ground vehicles and their laboratory implementation with three unmanned ground vehicles are presented.

Keywords: Unmanned Vehicles, Formation Control, Dynamic Environment, Dijkstra's Algorithm.

1 Introduction

It is expected that groups of unmanned ground and air vehicles will play a major role in future civilian and military applications [1],[2],[3]. Algorithms for their coordination and control must account for the dynamic nature of the environment in which they will operate [4]. With this capability, robots can be used in applications including search and rescue, exploration, surveillance, and scientific data gathering. In this paper, we propose a practical framework for online planning and control of multiple mobile robots moving in groups in a dynamically changing environment. The group must maintain some predetermined geometric shape while moving and is allowed to change formation as necessary to negotiate through the environment. As illustrated in Fig. 1, the framework consists of two main parts: (i) online planning of the formation and its transitions during the path to the goal; (ii) trajectory planning of the individual robots and their tracking control.

[*]Ph.D. Student, Mechanical Engineering, hao@me.udel.edu
[†]Research Student, Computer Science, blaxton@udel.edu
[‡]Professor of Mechanical Engineering, Corresponding author, agrawal@me.udel.edu
[§]Research Student, Computer Engineering, edlee@udel.edu
[¶]Assistant Professor of Bioresources Engineering, ebenson@udel.edu

The following assumptions are made in this study: (i) Robots have on-board sensors which can detect surrounding objects within a range with a small margin of error; (ii) The odometry data from the robot has small errors over short distances; (iii) A global camera can update at a slower speed relative to local sensors; (iv) Robots can reliably communicate with each other and can share the environment information; (v) Ground robots are driven by heading and rotation velocity which have upper and lower bounds; (vi) The environment is dynamic with slowly moving obstacles. Based on these assumptions, the following features must be embedded in trajectory planning and control: (i) Robots can change formations recursively as needed to avoid obstacles; (ii) Robots' ability must be accounted for during trajectory planning; (iii) A tracking controller must be used to keep the error bounded. Based on these needs, the outline of this paper is as follows: In Section 2 we present the flow chart and the strategy for formation change during motion. Map building of the environment and path selection using Dijkstra's search strategy is discussed in Section 3. Trajectory generation and tracking controllers are presented in Section 4. Illustrative experiments of groups of unmanned ground vehicles in formations are shown in Section 5.

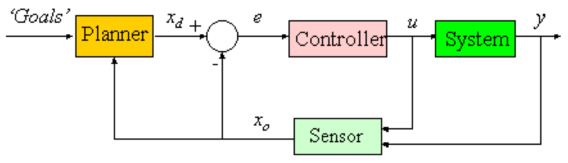

Figure 1: A block diagram of control of a group of unmanned vehicles with a planner and feedback controller.

2 Framework for Planning and Control

A flow chart for the planning and control of the group is shown in Fig. 2. First, a robot explores the environment and produces a map representing the free space in the environment at that time. Using a Dijkstra's search algorithm, a reference path for the virtual leader of the group is identified. A set of way-points is selected for the virtual leader and the trajectory generator produces continuous time trajectory for the virtual leader. Then according to the real robots' positions in the formation with respect to the virtual leader, the reference trajectory for each real robot is generated. Next, each robot tracks its own trajectory. Robots obtain position feedback through odometry readings and image data. The image data is more reliable but is updated at a much slower rate than the odometry readings. Hence, during image data update intervals, robots update their position using odometry. Thus the errors in the odometry readings get reset periodically. If the environment doesn't change, the robots track their computed trajectories. If the environment changes, a new path and trajectory is generated and is tracked by the controller.

2.1 Formation change

Since one of the goals of the planner is to keep the vehicles in a given formation, from the planning point of view, one may treat the entire formation as a point and enlarge the obstacles in the environment to accommodate the formation passing through it. This procedure is conservative but can potentially produce a feasible path of the formation through the environment. However, a situation may occur in which the formation must change in order to negotiate through the environment. The following steps are used to trigger formation change: (i) Compute the path for the formation using Dijkstra's search algorithm, (ii) If no feasible path is found, reduce the width while keeping the formation width at least as large as a single robot, (iii) If a feasible solution is found, adjust the path, compute the formation change points and vehicle trajectories.

For illustration, we consider a formation which has the form of a binary tree. The units in the formation are required to travel such that the formation changes from a *tree* to a *line* and then subsequently reassembles itself back into a tree. A practical scenario that motivates this example is when the vehicles cross a bridge which has a narrow width. The steps of the algorithm are illustrated schematically in Fig. 3. In the phase where the tree transforms into a line, we start from the bottom of the tree and recursively move up to the leader. In each step, through appropriate placement of way-points, we transform the three nodes of a triangle into a line. The way-points placement can be scheduled and tested offline to make sure no collision occurs between robots. It can be modularized and plugged in online whenever necessary. An example of this scenario on a path with two narrow openings is shown in Fig. 4. The full path is the result of piecing all five sub-paths together.

3 Environment Exploration and Dijkstra's Search

In our system the environment map was represented as a rectangle with an overlaid grid-cell structure where each cell represented one point of resolution. By using the rectangular structure we were able to automate the entire environment exploration process. The key steps in this process are (i) Map Creation and Updating; (ii) Path generation.

3.1 Creating and Updating Maps

Our system can be used in situations involving fully known, partially known, or unknown environments. In the first two

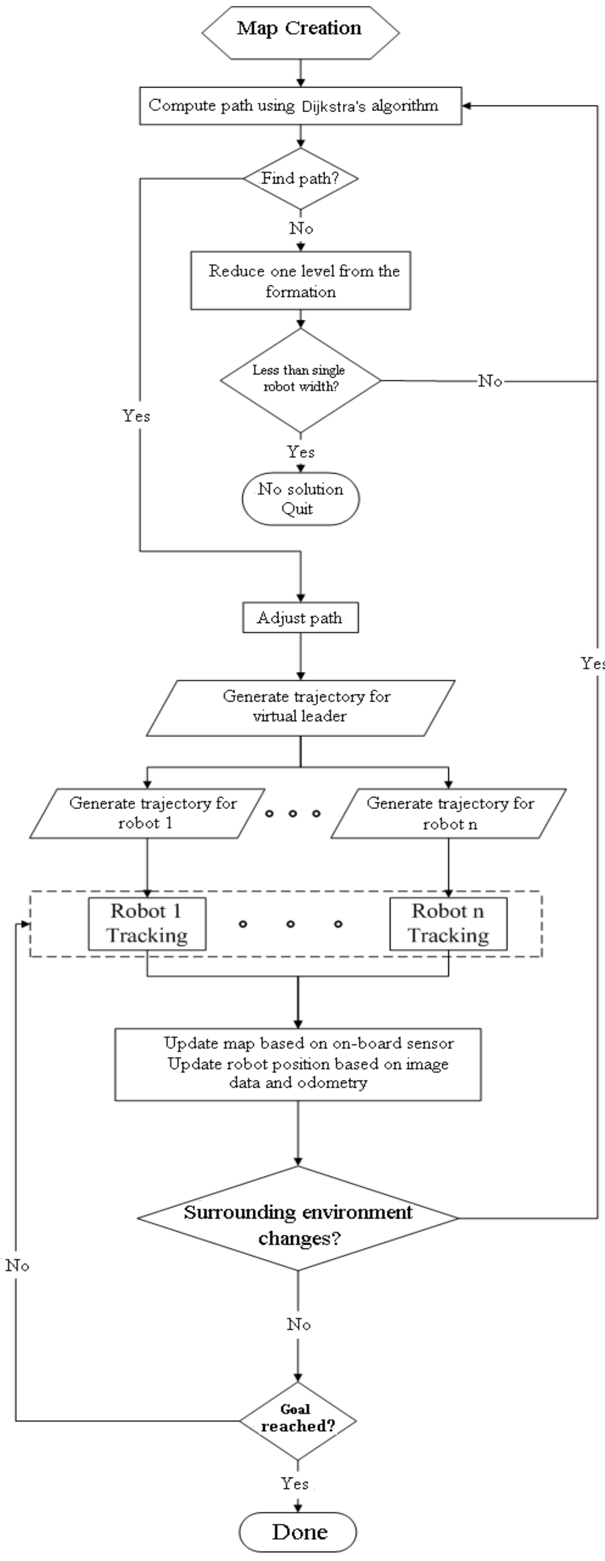
Map Creation
Compute path using Dijkstra's algorithm
Find path?
No
Yes
Reduce one level from the formation
Less than single robot width?
No
Yes
No solution Quit
Adjust path
Yes
Generate trajectory for virtual leader
Generate trajectory for robot 1
Generate trajectory for robot n
Robot 1 Tracking
Robot n Tracking
Update map based on on-board sensor
Update robot position based on image data and odometry
Surrounding environment changes?
No
No
Goal reached?
Yes
Done

Figure 2: Flow chart of formation control.

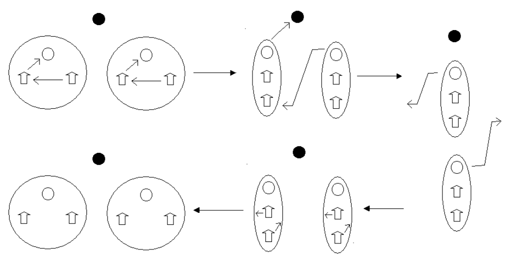

Figure 3: A schematic illustration of a formation change from a tree to a line to a tree.

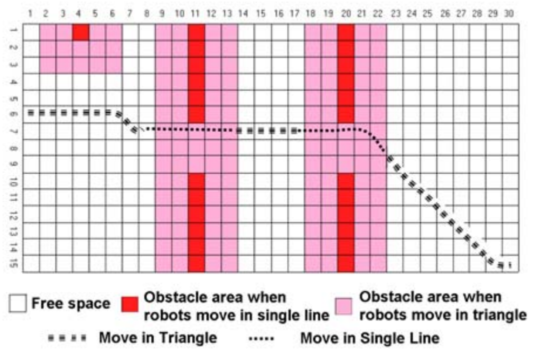
Free space
Obstacle area when robots move in single line
Obstacle area when robots move in triangle
Move in Triangle
Move in Single Line

Figure 4: Formation change to accommodate narrow passes.

cases information already known about the environment can be loaded into the system prior to running the robots. In the case of an unknown environment, we have implemented a system that creates an environment map from scratch that can be updated as new information is accumulated as illustrated in Fig. 5. First, the dimensions of the environment's bounding rectangle and the desired resolution are used to create an empty grid representation of the environment. Next, a robot is set out as a scout to autonomously create the initial map of the environment. This is achieved by using a full-coverage algorithm that finds points of unexplored space in the environment. The initial map creation has been completed when a desired percentage of the environment has been covered.

In order to accommodate dynamic environments and for robots to move about unknown terrain it is necessary for the map to be updated. The update frequency governs how fast the robots can react to changes in their environment - in our case it was around 20 hertz. The robot's pose as well as its sensor readings was used to extract data about free and obstructed space within the environment. The value of a grid-cell is incremented when the sensors indicate an obstacle or decremented if the space is unobstructed. In this way, the map can represent a constantly changing environment with varying levels of terrain cost. In order to simplify the implementation, the obstacles represented on the map are made larger by some buffer radius to account for a single robot or formation of robots - allowing the robot(s) to be treated as a point navigating through the free space.

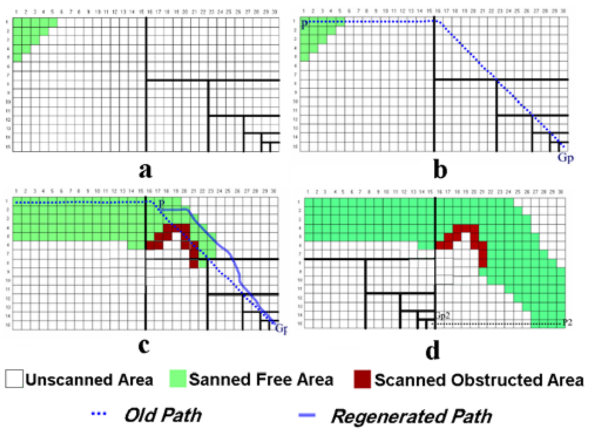

Figure 5: Steps in map creation.

3.2 Path Generation

This step is aimed at finding a path between two points, if it exists. Given a map, shortest paths can easily be found using one of the standard graph search algorithms, such as Dijkstra's or Floyd and Warshal's shortest path Algorithm, A*, or dynamic programming [5]. In the implementation, a version of Dijkstra's search algorithm is used which is modified to suit the needs.

This step returns a path with the lowest cost based on a heuristic given by

$$J = \sum_{i=0}^{n} [(x_g - x_i)^2 + (y_g - y_i)^2] + c_i, \qquad (1)$$

where the first term denotes the distance from the ith node to the goal, and c_i is a positive terrain cost for the ith node in the path.

Combining this algorithm and the map-updating algorithm described above, the system can find deadlock and collision free optimal paths through dynamic environments.

4 Trajectory Generation and Tracking Controller

4.1 Trajectory Generation

The Dijkstra's search algorithm generates a set of way-points for the virtual leader. The trajectory generator uses these way-points to produce a smooth trajectory of the virtual leader in the formation. The individual robots in the group have predetermined geometric relationships with respect to the leader and each other. These relationships define the reference trajectory for each robot based on the reference trajectory of the virtual leader.

For a given set of n way-points for a mobile robot, illustrated in Fig. 6, a smooth trajectory can be found to pass through them exactly or approximately [6]. There are many solutions of this problem based on limits on velocity and acceleration. For example, the way-points can be connected

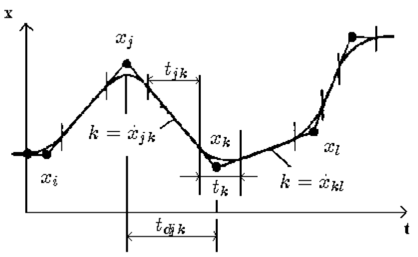

Figure 6: Linear interpolation with polynomial transition.

by linear interpolation with polynomial transition around the way-points. Given the way-point coordinates x_j, x_k, x_l, which are the x coordinates of j, k, l and $|\ddot{x}_k|$, which is the absolute value of acceleration for k, the parameters of motion according to Fig. 6 are given by

$$\begin{aligned} \dot{x}_{jk} &= \frac{x_k - x_j}{t_{djk}} \\ \ddot{x}_k &= sgn(\dot{x}_{kl} - \dot{x}_{jk})|\ddot{x}_k| \\ t_k &= \frac{\dot{x}_{kl} - \dot{x}_{jk}}{\ddot{x}_k} \\ t_{jk} &= t_{djk} - \frac{1}{2}t_j - \frac{1}{2}t_k \end{aligned} \quad (2)$$

The parameters in (2) can be adjusted to meet the kinematic and control input constraints. In general, trajectories with linear interpolation and polynomial transition do not pass through the way-points but pass near them. If a robot is required to pass through certain way-points with non-zero velocity, virtual way-points can be added on to the two sides of the desired way-point. Sometimes, when the environment changes, the new way-points may require sharp turns and the robot may not be able to execute the motion. In such situations, buffer way-points can be introduced as illustrated in Fig. 7. Fig. 8 shows the trajectories of seven vehicles in a maneuver from a tree to triple lines to tree given virtual leader's way-points and formation change points.

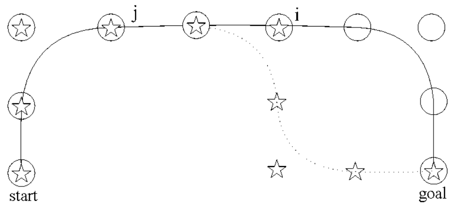

Figure 7: Smooth Turning with buffer way-points.

4.2 Tracking Controller

In this section, we assume that the robot reference trajectories have already been computed using the procedure in Section 4.1. Now, corrective strategies are required to keep the vehicles on these trajectories. In our experiment, the mobile robot is driven by a simple differential drive, with two coaxial powered wheels and a passive supporting castor wheel. The motion for robot i is governed by the following equations:

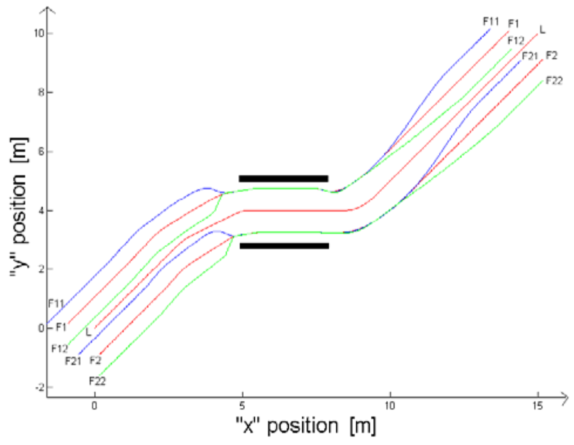

Figure 8: The trajectories for the example with seven vehicles that change formation from a tree to triple lines to tree.

$$\dot{x}_i = u_{1i} cos\theta_i \quad (3)$$

$$\dot{y}_i = u_{1i} sin\theta_i \quad (4)$$

$$\dot{\theta}_i = u_{2i} \quad (5)$$

Here (x_i, y_i) denotes the position of the center of the axle with respect to the inertial frame and θ_i denotes the orientation of the vehicle in the inertial frame. The inputs to the controller are u_{1i} and u_{2i} which are the forward driving speed and angular speed of the robot. The tracking controller is chosen based on Lyapunov theory. The reference trajectories for the ith robot are denoted by $x_{ri}(t)$, $y_{ri}(t)$, $\theta_{ri}(t)$. The errors between the actual and the reference trajectory are defined by

$$\begin{aligned} x_{ei}(t) &= x_{ri}(t) - x_i(t) \\ y_{ei}(t) &= y_{ri}(t) - y_i(t) \\ \theta_{ei}(t) &= \theta_{di}(t) - \theta_i(t) \end{aligned} \quad (6)$$

where θ_{di} is the desired angle given by the robot's position relative to the corresponding point in the reference trajectory in the inertial frame.

$$\theta_{di} = tan^{-1}\frac{y_{ei}(t)}{x_{ei}(t)} \quad (7)$$

On differentiating (6), we get

$$\begin{aligned} \dot{x}_{ei}(t) &= \dot{x}_{ri}(t) - u_{1i}cos\theta_i \\ \dot{y}_{ei}(t) &= \dot{y}_{ri}(t) - u_{1i}sin\theta_i \\ \dot{\theta}_{ei}(t) &= \dot{\theta}_{di}(t) - u_{2i} \end{aligned} \quad (8)$$

On selecting a Lyapunov function given by

$$V = \frac{1}{2}{x_{ei}}^2 + \frac{1}{2}{y_{ei}}^2 + \frac{1}{2}{\theta_{ei}}^2, \quad (9)$$

We obtain

$$\begin{aligned} \dot{V} &= x_{ei}\dot{x}_{ei} + y_{ei}\dot{y}_{ei} + \theta_{ei}\dot{\theta_{ei}} \\ &= x_{ei}\dot{x}_{ri} + y_{ei}\dot{y}_{ri} - (x_{ei}cos\theta_i + y_{ei}sin\theta_i)u_{1i} \\ &+\theta_{ei}(\frac{x_{ei}(\dot{y}_{ri} - u_{1i}sin\theta_i) - y_{ei}(\dot{x}_{ri} - u_{1i}cos\theta_i)}{{x_{ei}}^2 + {y_{ei}}^2} \\ &-u_{2i}) \end{aligned} \quad (10)$$

The tracking controller can be chosen to satisfy the following conditions to meet the negative definiteness condition on Lyapunov function

$$u_{1i} = \frac{x_{ei}\dot{x}_{ri} + y_{ei}\dot{y}_{ri}}{x_{ei}cos\theta_i + y_{ei}sin\theta_i} + k_{1i}(x_{ei}cos\theta_i + y_{ei}sin\theta_i)$$
$$u_{2i} = \frac{x_{ei}(\dot{y}_{ri} - u_{1i}sin\theta_i) - y_{ei}(\dot{x}_{ri} - u_{1i}cos\theta_i)}{{x_{ei}}^2 + {y_{ei}}^2} + k_{2i}\theta_{ei} \qquad (11)$$

where k_{1i} and k_{2i} are positive constants. There are two singularities: the robot is in the ideal position when ${x_{ei}}^2 + {y_{ei}}^2 = 0$ or the robot's heading direction is perpendicular to the desired angle θ_{di} when $x_{ei}cos\theta_i + y_{ei}sin\theta_i = 0$. In implementation, one can use (12) when ${x_{ei}}^2 + {y_{ei}}^2 < \epsilon$ or $|x_{ei}cos\theta_i + y_{ei}sin\theta_i| < \epsilon$, where ϵ is a small positive number.

$$u_{1i} = 0$$
$$u_{2i} = k_i(\theta_{ri} - \theta_i) \qquad (12)$$

where k_i is a positive number.

5 Laboratory Experiments

A physical implementation of the strategy for formation planning and control was performed to test the practicality of the concept. The setup consists of three differential drive Magellan Pro mobile robots. A schematic of the experimental setup is shown in Fig. 9. Eqs.(3), (4), and (5) provide the dynamic model for each robot. Each robot has an on-board PC consisting of an EBX motherboard and a Pentium III processor. The robot operates under Linux operating system and its software integrates sensor and communication data. The robots communicate through wireless Ethernet capable of transmitting data up to 3Mb per second. The distributed computation is implemented using CORBA.

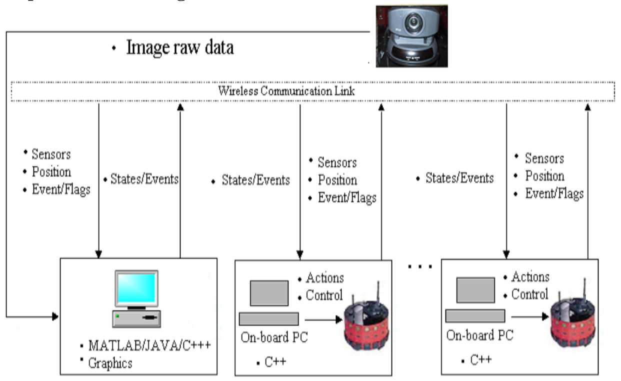

Figure 9: experimental setup

A Sony CCD camera is mounted 2.6 meters above the experiment floor. A C program accesses the streaming data coming into the frame grabber from the camera and stores the data in a 320x240 image file. The mean error between the original coordinates and coordinates extracted from the image is 2.08% for the X-axis, 9.41% for the Y-axis, and 0.32% for heading direction. Colored disks of known shape and size are located on each robot to enhance robot identification and heading direction. Once the image processing is completed, the robot odometry and heading is calibrated according to the vision data. There is an array of 16 sonar sensors and 16 IR sensors on the robot which are also used for local identification of the environment. Translational and rotational velocity controllers are used to reposition each robot. MATLAB/C++/JAVA are used as the computational engine for decision making, control and graphical display. The purpose of the experiments is to show that these algorithms work in dynamic environments. A block-diagram of the computational procedure is shown in Fig. 10. We ran experiments on two different configurations of free/obstructed space within the environment.

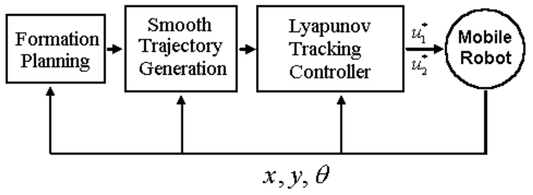

Figure 10: formation planning and tracking control.

In the first experiment, an obstructed area was created with cardboard boxes in the middle of the workspace, initially out of view from the robots' on-board sensors. The map provided to the robots at the beginning of the run indicated that it was possible for them to move to the goal in a triangular formation. However, as the robots moved, their on-board sensors detected the obstacles and their formation had to be changed to a line. The experiment plots are shown in Fig. 11.

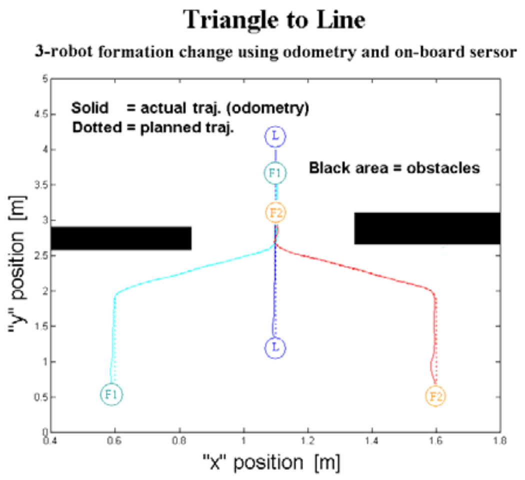

Figure 11: Triangle to a Line.

In the second experiment, the obstacles were moved closer to the start position of the robots. In this experiment, the robots detected the obstacles early on and went to a line in the beginning of the trajectory. After moving past the obstacles, the robots detected that the environment was free of obstacles, so they could again move to a triangle. Plots for this experiment are shown in Fig. 12.

Fig. 13 shows the robots' actual trajectory while using the global camera to correct for odometry errors. In this figure, there are 'jumps' in the robot position. These 'jumps' are not reflected in actual robot movement, but instead indicate times at which the robots' position was updated using the camera. Given the more precise location, the tracking controller could allow the robots to smoothly converge back to the correct trajectory. The videos for the two experiments are online at *http://mechsys4.me.udel.edu*. The snap shots taken from the second experiment are shown in Fig. 14. From the results of these two experiments, it is clear that the algorithms described in this paper can be applied to a dynamic environment. In addition, it is feasible to implement them for real-time responsive behavior for currently available hardware.

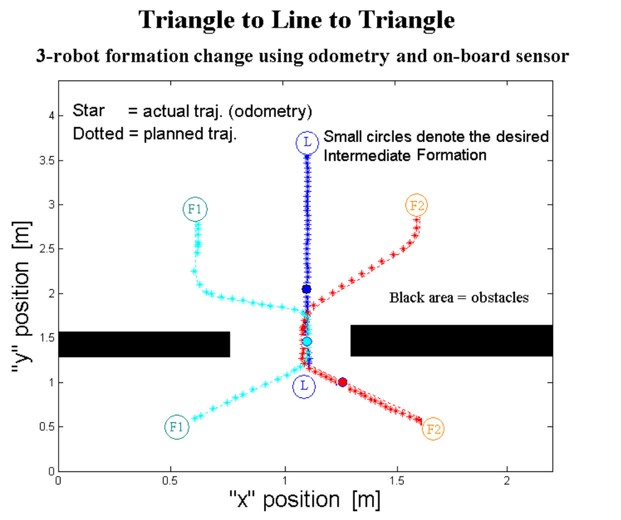

Figure 12: Triangle to Line to Triangle.

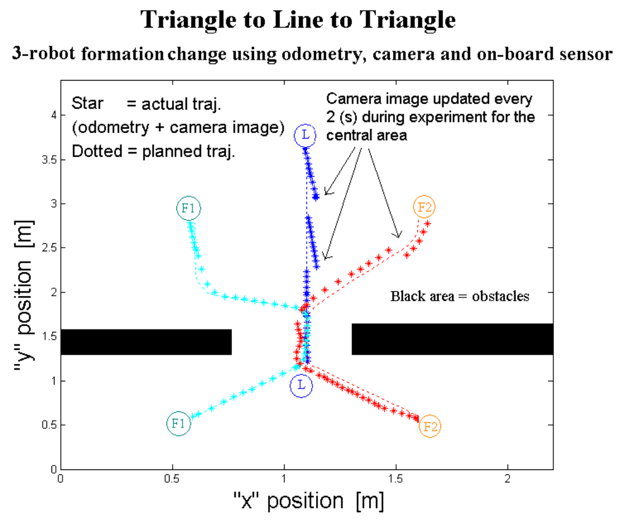

Figure 13: Triangle to Line to Triangle using camera.

6 Conclusions

This paper provides a practical framework for planning and control of formations of multiple unmanned ground vehicles to traverse between goal points in a dynamic environment. This framework allows online planning of the formation paths using a Dijkstra's search algorithm based on the current sensor data. The formation is allowed to dynamically change in order to avoid obstacles in the environment. By planning the trajectory of a virtual leader, we reduce the computational complexity of the planning problem significantly. The trajectory re-planning is realized online and the global convergence is guaranteed by Dijkstra's search algorithm. The tracking controller is developed based on Lyapunov theory. Illustrative experiments of groups of unmanned ground vehicles show promise of this approach.

Acknowledgments: We acknowledge the research support of NSF Award # IIS-9912447 and NIST MEL Award # 60NANB2D0137.

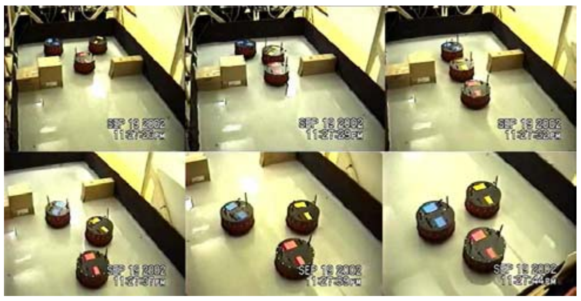

Figure 14: experiment snap shots.

References

[1] Balch, T., Arkin, R.C. "Behavior-based Formation Control for Multi-robot Teams", *IEEE Transactions on Robotics and Automation*, Vol. 14, pp 926-939, Dec. 1998.

[2] Fredslund, J., Mataric, M.J. "Robot Formations Using Only Local Sensing and Control", *IEEE International Symposium on Computational Intelligence in Robotics and Automation*, pp 308-313, Alberta, Canada, 2001.

[3] Guo, Y., Parker, L.E. "A Distributed and Optimal Motion Planning Approach for Multiple Mobile Robots", *Proceedings of IEEE International Conference on Robotics and Automation*, Washington, DC, 2002.

[4] Pledgie, S. T., Hao, Y., Ferreira, A. M., Agrawal, S. K., Murphey, R., "Groups of Unmanned Vehicles: Differential Flatness, Trajectory Planning, and Tracking Control", *Proceedings of IEEE International Conference on Robotics and Automation*, Washington,DC, 2002.

[5] Kortenkamp, D., Bonasso, R.P., Murphy R. *Artificial Intelligence and Mobile Robots*, AAAI Press/The MIT Press, 1998.

[6] Xiong, Y., *Fundamental Principal of Robotics*, Huazhong Sci. and Tech. Press, P.R.China, 1996. ISBN: 7-5609-1305-9.