Autonomous Robots 4, 387–403 (1997)

High Precision Formation Control of Mobile Robots Using Virtual Structures

M. ANTHONY LEWIS* AND KAR-HAN TAN*
The Commotion Laboratory, Computer Science Department, University of California, Los Angeles, CA 90095-1596
tlewis@uiuc.edu
tankh@uiuc.edu

Abstract. A key problem in cooperative robotics is the maintenance of a geometric configuration during movement. To address this problem, the concept of a **Virtual Structure** is introduced. Using this idea, a general control strategy is developed to force an ensemble of robots to behave as if they were particles embedded in a rigid structure. The method was instantiated and tested using both simulation and experimentation with a set of 3 differential drive mobile robots. Results are presented that demonstrate that this approach is capable of achieving high precision movement that is fault tolerant and exhibits graceful degradation of performance. In addition, this algorithm does not require leader selection as in other cooperative robotic strategies. Finally, the method is inherently highly flexible in the kinds of geometric formations that can be maintained.

Keywords: cooperative mobile robotics, formation control, virtual structures, moving in formation, autonomous robotics, collective behavior

1. Introduction

One of the most important and fundamental problems in the control of multiple mobile robots is *formation control*. While many interesting tasks and behaviors (Cao et al., 1995) can be demonstrated with multiple mobile robots that move freely without strict constraints on their relative positions, there are many important tasks that require the robots to coordinate their movements more closely.

Aircraft flying in a V-shaped formation, for example, could maximize fuel efficiency by taking advantage of aerodynamics similar to the way flocks of birds use formation flying to increase their efficiency (Lissaman and Shollenberger, 1970). In addition, if formation control is automated, multiple aircraft could be flown by one pilot at a time, reducing fatigue in long-distance flight (e.g., pilots in each aircraft would take turns controlling the formation). Alternately, a single piloted aircraft could control a large formation of semi-autonomous pilotless aircraft.

Another application is the control of spacecraft formation. Multiple small spacecraft may be used to form sensor arrays or to position sensors very precisely in fixed geometric relationships to each other.

Box pushing (and the related task of coach pushing) (Donald et al., 1995; Yamaguchi et al., 1995) is a well studied problem in cooperative robotics. As other authors have pointed out, multiple robots can take advantage of parallelism and enhance their capability (Johnson and Bay, 1995). In addition, it may be more cost effective to use a colony of identical robots to move boxes (or other loads) of varying size rather than to have a collection robots with equally varying sizes.

As illustrated in Fig. 1, moving the box continuously along a path requires that the robots maintain a fixed geometric relationship with the box as the robots and the box move and rotate in space. Loose adherence

*Present address: Beckman Institute, 405 N. Mathews Ave. University of Illinois Urbana-Champaign, Urbana, Illinois, 61801.

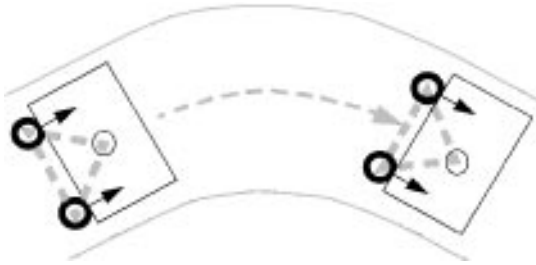

Figure 1. Mobile robots pushing a box have to maintain a fixed geometric relationship with the box to ensure continued effectiveness.

to the geometric constraints will result in uneven load distributions as robots slide along the box or lose contact with the box.

One way of approaching this problem is to imagine that the individual robots are particles embedded in a structure. The constraints normally imposed mechanically in a real structure are imposed by a control strategy. Since the robots behave as if they are embedded in a real structure, we call this a Virtual Structure (VS).

In the following section we formalize the definition of the VS. Next we give a general solution to the problem of creating control strategies for the robots such that they satisfy the geometric constraints of a VS while the VS itself is allowed to move. We then instantiate this technique and apply it to the case of a system of differential drive robots. We give simulation results as well as results using real robots (3 ISR Robotics R3 Robots). We then illustrate the flexibility of this paradigm by discussing how it can be applied to a wide range of applications.

2. Problem Statement

Prior to formalizing the problem of moving in formation, we define a Virtual Structure precisely. This abstraction will simplify subsequent reasoning about the problem.

2.1. Virtual Structure

A *rigid body* is composed of a system of point masses fixed by holonomic constraints such that: $|r_i - r_j| = d_{ij} = const$ where r_i, r_j are the positions of the particles (Arnold, 1989).

The particles can be thought of as being stationary with respect to a certain frame of reference that is also moving through space (a frame can be defined by any three non-colinear particles in the system). If the geometric relationships are not enforced by a physical system of constraints (i.e., the molecular forces holding a solid object together) but instead by a human-made control system, then we call the structure a Virtual Structure.

In a robotic system, the generation of a Virtual Structure is facilitated by the use of sensing, mobility and intelligent control.

Definition: *Virtual Structure (VS).* A virtual structure is a collection of elements, e.g., robots, which maintain a (semi-) rigid geometric relationship to each other and to a frame of reference.

2.2. Moving in Formation

A solution to the problem of moving in formation must generate robot movements that simultaneously satisfy two goals: (1) make progress in a given direction and (2) maintain a rigid geometric relationship among the mobile robots. It is easy to see that the second goal is the same problem as forming a virtual structure with mobile robots. The first goal is to move the virtual structure in a prescribed direction. As a rigid body will move if embedded in a force field (i.e., a gravitational field) a virtual structure can be made to move by embedding it in a virtual force field. This field can be generated by interacting with a human operator or through other automated means.

The geometry of the problem statement is illustrated in Fig. 2. Formalizing these ideas, we state the problem as follows:

Moving in Formation Problem Statement

Given n robots labelled $1, \ldots, n$ whose positions in the world coordinate frame are represented by vectors $r_1^W \ldots r_n^W$.

Impose a virtual structure that consists of n points, whose positions in its reference coordinate frame are represented by vectors $p_1^R \ldots p_n^R$.

Let I_R^W be a transformation that maps $p_1^R \ldots p_n^R$ to $p_1^W \ldots p_n^W$, the positions of the virtual structure points in the world coordinate frame.

Say the robots are moving in perfect formation if the robots are moving such that at each point in time there is a I_R^W such that $p_i^W = r_i^W$ for all robots.

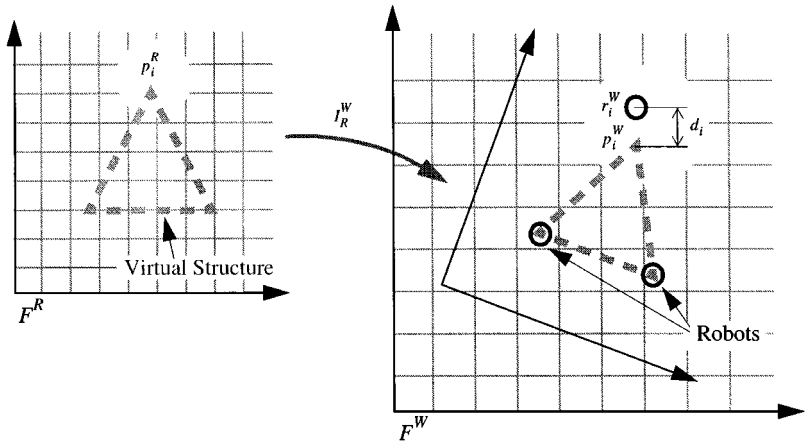

Figure 2. Geometry of the virtual structure definition in its reference coordinate frame mapped through the transformation I_R^W into the world coordinate frame. The vertex i shown on the top of the triangular virtual structure is mapped to position p_i^W. The corresponding robot, positioned at r_i^W, contributes an error of d_i. The two robots below are perfectly aligned with their virtual structure points. The transformation I_R^W does not minimize the system error: the virtual structure is so placed for clarity of presentation.

The problem is thus to design a control algorithm that is capable of making the n robots move in formation in a prescribed direction.

It can be seen in the definition that the virtual structure imposed represents the desired relative positions for robots in the formation. While the notion of moving in *perfect* formation is well defined, robots not in perfect formation may still be *close* to being in formation. The line between being *in* formation and *out* of formation can only be drawn based on the specific application.

3. General Solution

Virtual Structures can be used to solve the problem of moving in formation. The solution is based on the idea that when a virtual force field is exerted on a VS, the individual robots in the VS will move in the direction of the force. The position and orientation of the virtual structure must then be determined in turn by the mobile robots' relative positions. *Therefore the robots move to stay in the VS and the VS moves to fit the robots' current positions*.

This recursive description outlines the essence of our solution: *bi-directional control* (see Fig. 3). The mobile robots can be controlled by applying a virtual force field to the virtual structure, but the position of the virtual structure is ultimately determined by the position of the mobile robots. This behavior can be simulated by iterating the following algorithm, which is also illustrated in Fig. 4:

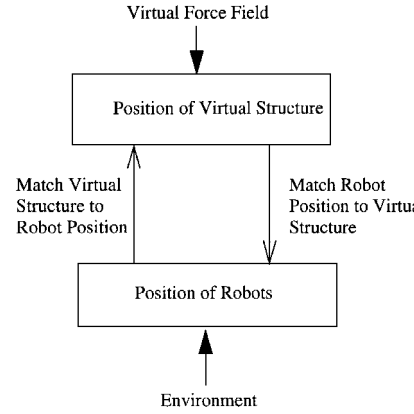

Figure 3. Bidirection flow of control. At the top level, a high level planner or human input generates a force field that influences the position of the VS. The virtual structure in turn influences the position of the robots. From the top down: The positions of the robots are influenced by the environment (obstacles may influence the movement of the robots for example). The robots positions in turn influences the position of the virtual structure.

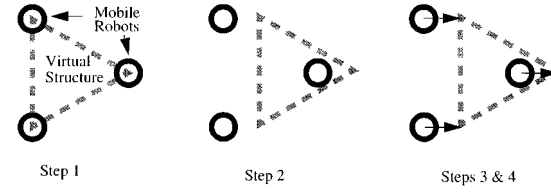

Figure 4. Steps in the virtual structure control algorithm for moving in formation.

Moving in Formation Algorithm

(1) Align the VS with the current robot positions.
(2) Move the VS by Δx and $\Delta\theta$.
(3) Compute individual robot trajectories to move robot to the desired VS point.
(4) Adjust wheel velocities to follow the desired trajectories.
(5) Goto step 1.

In step 1, the virtual structure is aligned with the robots. An external force then pushes the virtual structure in a desired direction in step 2. In steps 3 and 4 the mobile robots, in trying to maintain the virtual structure, update their trajectories so that they are moving like points in a rigid structure. The individual steps will be explained in more detail in the following section.

3.1. Fitting the Virtual Structure

A virtual structure is made up of an arbitrary (but non zero) number of points which may include a passive element (i.e., a box being pushed). For our purpose we let the number of elements be equal to the number of robots we are interested in controlling. In order to align the virtual structure to the robots, we define a fixed one-to-one mapping between the points on the virtual structure and each robot. This mapping from robot to virtual structure is fixed and is determined when the robotic system is initialized. As described in the definition, the alignment is performed by minimizing the error between the actual positions of the robots and their corresponding virtual structure points.

An objective function can be created that measures the fit of the virtual structure to the robots. The objective function is the sum of errors for each robot from its assigned point in the virtual structure. The objective function maps the parameters of the transform function I_R^W onto the real number line ($f : \Re^m \to \Re$) and is of the form:

$$f(X) = \sum_{i=1}^{N} d\left(r_i^W, I_R^W(X) \cdot p_i^R\right) \qquad (1)$$

where N is the number of robots, $d(\ )$ is a distance or norm which measures the cost of being away from the assigned VS point. Recall that r_i^W is the position of robot i in the world frame and p_i^R is the assigned position of the robot i in the frame of the VS. Also, $X \in \Re^m$. If, for example, the virtual structure moves on a plane, the $m = 3$; x, y translation and θ rotation.

The function I_R^W represented by the homogeneous transform as:

$$I_R^W(R) = \begin{bmatrix} R & P \\ 0 & 1 \end{bmatrix} \qquad (2)$$

where $R \in SO(3)$ and $P \in \Re^3$ in general. R and P are parameterized by the element of $X = [X_1, \ldots, X_6]$. Thus X is the set of variables to be optimized. Our goal is to find X^* such that $f(X^*) \leq f(X)$ for all X.

The significance of this is that the search space for transformations can be generated by the Cartesian product of the space of rotations and translations.

After obtaining X^*, we have determined the 'best' position of the virtual structure. The next steps will make the robots move in formation.

3.2. Moving the Virtual Structure

This involves simply adding a displacement to the translation and rotation specified in the I_R^W transformation. The manner and magnitude by which to displace the virtual structure depends on both the given task and the capability of the robots in the system, since the robots will attempt to move towards the displaced virtual structure points. If the displaced points are within the capability of the mobile robots then there will be no errors in the next iteration. However displacing the virtual structure beyond the capability of the robots causes errors (Fig. 5). We shall see in a later section a real example of taking robot limitations into account.

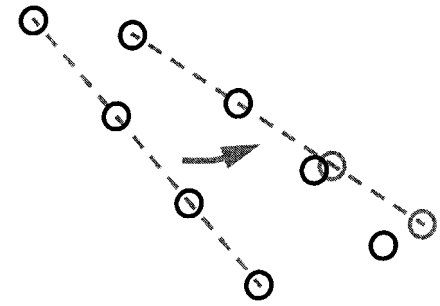

Figure 5. A straight line virtual structure is moved. The lower two robots were given destinations beyond their capabilities causing an increase in the virtual structure fitting error. Note that the distance between the displaced virtual structure points (gray) and the final robot positions (black) do not represent the virtual structure fitting error in the next round of iteration, but rather is a measure of how much the virtual structure was over-displaced.

3.3. *Moving the Robots*

The next step in the execution of the algorithm is the computation of robot velocities to move the robots onto the displaced virtual structure before the next iteration. Note that it is not sufficient that the robots move along paths that intercept the corresponding displaced virtual structure points. Moving too fast causes the robots to move *past* the desired position. The robots have to be *on* the points the next time the virtual structure is aligned. Thus the computation of the velocities requires the use of the cycle time of execution of the algorithm. The rest of the computation requires information about the robots themselves. Robots with different modes of locomotion would require different constraint equations. For example, aircraft flying in formation would compute their velocities differently from legged robots moving in formation. Below we will see an example of velocity computation for a specific type of robot.

3.4. *Communication Requirements*

Each robot performs an identical optimization computation and computes its next based on two pieces of information: (1) The position of each robot in the formation and (2) a force field command from a higher level planner (which could perhaps be distributed as well) or human input. The communications requirements are a function of four factors: (1) N, the number of robots in the formation, (2) K, the bits needed to encode the position of a single robot (3) T, the update period of the robot control cycle.

If we assume that each robot can determine its positions in a world frame, and that each robot broadcasts its position in each control cycle, then the required bandwidth in units of bits per second is:

$$BW = K \cdot N/T \tag{3}$$

4. A Problem Instance and Solution

We now turn to a specific problem instance to illustrate the capability of the algorithm. We have presented a general solution for the problem of moving in formation. It is a general algorithm that can be adapted for different kinds of virtual structures and different kinds of robots. In particular, the steps of moving the virtual structure and moving the robots have to be adapted for each setup. Aligning the virtual structure is done in the same way in most solutions, although the search space can be constrained and the method of search for the transformation can be adapted so that task mechanics can be exploited to reduced the computation required. For example, when a number of aircraft are moving in formation the virtual structure is capable of moving with six degrees of freedom, whereas virtual structures formed by intelligent cars on a straight highway are capable of moving with only two, or sometimes one degree of freedom. In the this section, a specific problem is instantiated and the general solution is adapted to solve it. Simulations and experiments with real robots were subsequently carried out to investigate the behavior of robots controlled with the virtual structure algorithm.

In order to demonstrate the solution with real robots, we choose a problem instance that can be solved by a collection of IS Robotic's R3 robots. These are two-wheel differential drive mobile robots capable of moving only on the ground. Virtual Structures formed with these robots are therefore always planar, and only move with three degrees of freedom on the ground: two degrees in translation parallel to the ground and one degree in rotation about an axis perpendicular to the ground. In simulations and experiments, the virtual structure formed with the robots is given a destination position and orientation. The Virtual Structure should then make progress towards the destination while keeping the fitting error as low as possible.

4.1. *Naive Solution*

The initial solution is simple: just push the virtual structure with some constant translational and rotational velocity. The robots would just try to head towards the virtual structure points. The robots would need to compute velocities for their left and right wheels given their current positions and the desired positions. The equations used are based on the following reasoning: there are two components of motion for the differential drive robots, translation in a direction perpendicular to the wheel axle and rotation. In the pure translation case the two wheels would get equal velocities. In the pure rotation case the wheels would get opposite velocities. A simple method to move a robot to a location is to rotate the robot until it is heading in the direction of the desired position, and then translate towards the desired position.

Generalizing, we can blend the two components to generate a smooth trajectory. The robot would initially

turn towards the desired position. When the angle between its heading and the desired heading comes within a threshold it starts to acquire a small translational component. By making the threshold smaller than π radians, the robot would be moving closer to the destination, though it may not be heading directly towards it. Now we decrease the rotational component as the robot corrects its heading to face the destination, and vary the translational component proportionally to the distance between the robot and its destination. This generates a smooth path, and is completely described by the following equations:

$$v_l = \theta_{\text{err}} \cdot K + \text{MIN}(d_{\max}, dist) \cdot (H_c \cdot H_d) \quad (4)$$

$$v_r = -\theta_{\text{err}} \cdot K + \text{MIN}(d_{\max}, dist) \cdot (H_c \cdot H_d) \quad (5)$$

where *dist* is the distance between the robot and its desired position, $d_{\max}$, K are real constants, and

$$\theta_{\text{err}} = \text{atan}\left(\frac{\|H_c \times H_d\|}{H_c \cdot H_d}\right) \quad (6)$$

is the angle between the current robot heading vector H_c and the desired robot heading vector H_d (in the direction of the desired position). '$\cdot$' is the scalar product and '$\times$' is the vector product.

By limiting the virtual structure displacement in each iteration it is possible to make the virtual structure move with small errors.

In initial simulation and real-robot studies, we found that while pure translation and pure rotations can be smoothly executed, the virtual structure cannot smoothly go from translation to rotation, and vice versa without incurring large errors. This is due to the nonholonomic constraints of differential drive robots.

When the virtual structure is translating, the robots are all facing the same direction, but when it needs to rotate, the robots have to adjust their heading to be tangential to the circular path followed by the virtual structure points. Also, the fact that the robots do not always try to be on the spot causes errors as well.

In the next section we introduce an improvement to the naive solution.

4.2. *Developing the Objective Function: Using the Concept of Reachability*

In the second version of the solution, the capability of the differential drive robot is taken into account. Specifically, we take into account the region that the robot is capable of reaching in the next instant given limits on the acceleration of its actuators. This defines the *reachability region*. Intuitively, a differential drive robot is not capable of moving sideways instantaneously, but is capable of motion forward or backwards. Thus if the virtual structure can be displaced in such a way that it takes into account that fact, it will result in lower error and smoother robot motion.

In order to model the motion of the robots we derived two operators: one operator takes the desired position for the robot and computes the wheel velocities needed, the other takes the wheel velocities and predicts where the robot will be during the next round of alignment. The second operator is used in computing the appropriate virtual structure displacement, while the first is used to command the robots after displacing the virtual structure. Since the algorithm is a control loop, for each iteration of the algorithm only one command is sent to each robot. This means that the robots will be moving with uniform velocities within one time-step. For differential drive robots, this implies that the robots are always moving along circular arcs (a straight line can be thought of as being a circular arc with infinite radius). Using geometry illustrated in Fig. 6, we can write the relationship between the turning radius R, the angle of turn θ and the translation $(\Delta x, \Delta y)$ as the following equations:

$$\Delta x = R \sin\theta \quad (7)$$

$$\Delta y = R(1 - \cos\theta) \quad (8)$$

Using these two equations we can express the turning radius R and turn angle θ in terms of the translational displacements:

$$\theta = \text{asin}\left(\frac{1 - \left(\frac{\Delta y}{\Delta x}\right)^2}{1 + \left(\frac{\Delta y}{\Delta x}\right)^2}\right) \quad (9)$$

$$R = \frac{\Delta x}{\sin\theta} \quad (10)$$

Given the length of a time-step ΔT, we arrive at two

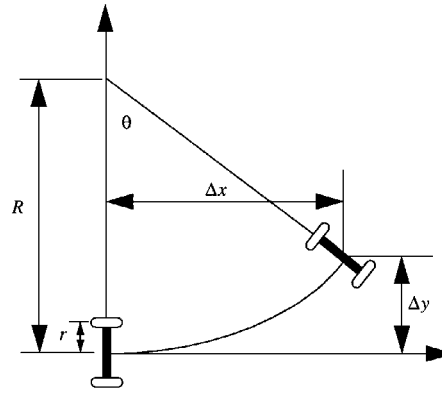

Figure 6. Derivation of wheel equations.

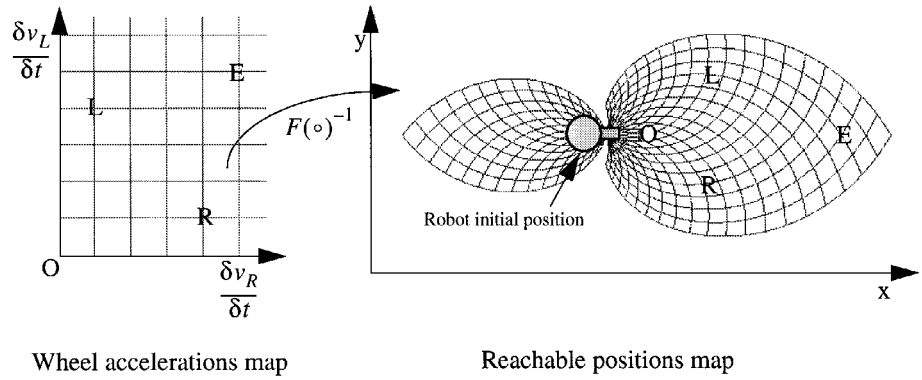

Figure 7. Chart of the function $F^{-1}(\circ)$. Relationship between acceleration applied to the left and right wheels of a differential drive robot and the positions reached by the robot. The wheel accelerations map shows a two-dimensional grid of the accelerations applied to the wheels, with point *O* representing no acceleration, point E representing large and equal accelerations, and points L and R representing unequal accelerations, the larger applied to the left and right wheels respectively. The reachable positions map shows the corresponding displacements. The robot has an initial velocity along the *x*-axis. The positions reached by applying the accelerations represented by the points O, E, L and R are shown in the reachable positions map. The size of the map is determined by the period ΔT of the control cycle. The map is shown, in proportion to the robot, for a very large value of ΔT. In practice, the map would be a fraction of the size illustrated.

velocities

$$v = (R \pm r)\frac{\theta}{\Delta T} \tag{11}$$

which are then easily assigned to the left and right wheels depending on whether the desired destination is on the 'left' or 'right' side of the robot. When $\theta = 0$, the robot moves along a straight line, and the wheel velocities are simply equal to $\Delta x/\Delta T$. Note that the equations will produce arbitrarily large velocities when destinations are far away or when time slices are small. In real robots there is always a finite upperbound on the wheel speed. To limit the velocities, when the velocities are too high, scale down the velocities by a factor as follows:

$$v_{\text{limit}} = \frac{v_{\max}}{\text{MAX}(|v_l|, |v_r|)}v \tag{12}$$

which ensures that the magnitude of the velocities do not exceed $v_{\max}$.

The Eqs. (9)–(11) are a forward mapping $F : \Delta X \rightarrow V$. The inverse operator $F^{-1} : \Delta V \rightarrow X$ takes velocities and finds the resulting position. Using the fact that the wheels are moving along circular arcs we can derive R and θ as follows:

$$\theta = \frac{(vl - vr)}{-2r}\Delta T \tag{13}$$

$$R = \frac{(vl + vr)}{2\theta}\Delta T \tag{14}$$

From these we can compute the translation parallel (Δx) and perpendicular (Δy) to the robot's axis using Eqs. (7) and (8) given above. Since the inverse operator is an analytical result, it does not require numerical integration and is inexpensive in terms of computation. Therefore it lends itself to real-time simulation and control. In order to visualize the implications of the equations, Fig. 7 was plotted. The diagram clearly shows that within a limited amount of time a differential drive robot can reach further along its axis. This confirms the intuition that it is 'harder' for differential drive robots to move 'sideways' instantaneously. This result, when applied to our problem implies that given a robot configuration, some ways of moving the virtual structure are better than others. This is illustrated in Fig. 8. The non-holonomic constraints on

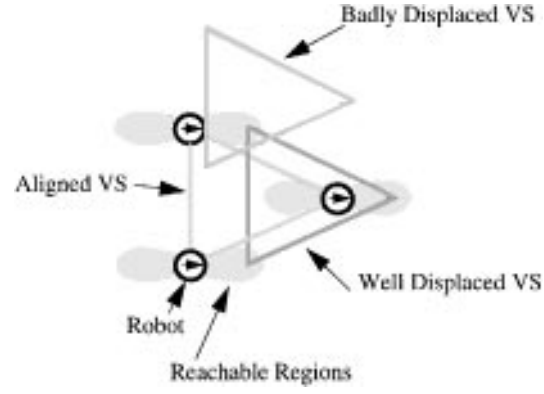

Figure 8. The virtual structure should be moved such that the points are reachable by the mobile robots.

each differential drive robot is visualized by the shaded regions in front and behind each individual robot. The virtual structure should thus be moved in a way that attempts to place the displaced virtual structure points inside the reachable region of each robot. However, it is not always possible to place the virtual structure that way. Sometimes a displaced virtual structure may not be entirely within the reach of all the robots, but incurs a small error in terms of deviation from the reachable regions. With this notion, we can quantify the "goodness" of a displaced virtual structure with respect to the current mobile robot configuration.

Let D_R^W be a transformation that maps the virtual structure reference coordinate frame into the world coordinate frame. The problem of displacing the VS can then be solved by finding a D_R^W that minimizes the cost function. The derivation is described below.

4.3. VS Displacement

The main idea is illustrated in Fig. 9. When a robot is given a destination in each time step, it computes the appropriate wheel velocities to reach the destination within the time given. However due to limitations in the robot's capacity (which can be encoded as $v_{\max}$, the velocity limit), the robot may not be able to reach the destination. In this case the resulting distance between the actual point reached and the desired destination is a measure of how well the virtual structure was displaced. Formalizing, the cost function can be defined as

$$E\left(D_R^W\right) = \sum_i e_i^2 \tag{15}$$

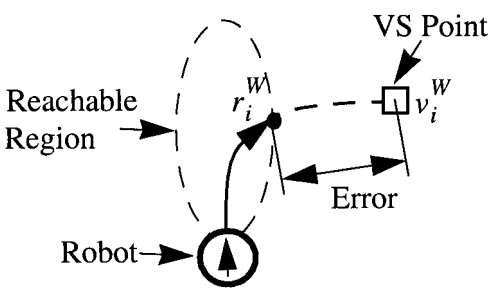

Figure 9. If a robot is given an unreachable destination, velocity clamping will cause it to reach a point on the reachable region boundary. The distance from the desired destination and the actual destination reached is a measure of error.

where

$$e_i^2 = \left(v_i^W - r_i^W\right)^T \left(v_i^W - r_i^W\right) \tag{16}$$

and v_i^W is the position of the displaced Virtual Structure point i, r_i^W is the position of the robot i after moving with the possibly clamped velocity.

Therefore, computation of the cost function e_i for each robot i involves

Computation of Cost of Virtual Structure Fitting

1. Computing the velocities necessary to reach v_i^W within a time step.
2. If the velocities required are beyond the robot's performance envelope, limit the velocities.
3. Compute p_i^W, the position reached using the (possibly limited) velocities.
4. Compute e_i, the distance between p_i^W, the point reached and v_i^W, the desired destination.

It should be clear that when there is no velocity limiting, e_i will be zero, since the robot is capable of reaching the desired destination. The cost function then increases as the desired destination point moves further from the robot's reachable region (Fig. 10(a)).

Recall that the second goal of moving in formation is to make progress in a prescribed direction. A bias can thus be added to direct the motion of the virtual structure. The result is shown in Fig. 10(b). The preceding two figures seem to suggest a nice simple cost surface. Indeed when the robots are oriented properly, the cost surface is quite simple. However, when the robots are not in the right configuration for the desired formation motion, the cost surface can quickly become complex. Figure 10(c) shows the cost function of a translating virtual structure with two robots of different orientations. In simulation the cost function generates smooth trajectories for the robots in the virtual structure, at each step moving the virtual structure in a way that takes advantage of the configuration of the robots.

The remaining part of the solution for our problem instance is to examine how to carry out the minimization of the two cost functions, one for fitting the virtual structure and one for moving the virtual structure. Davidon-Fletcher-Powell method for multidimensional minimization (Press et al., 1995), does not require the derivatives of the cost functions. This feature makes it useful for research and experimental

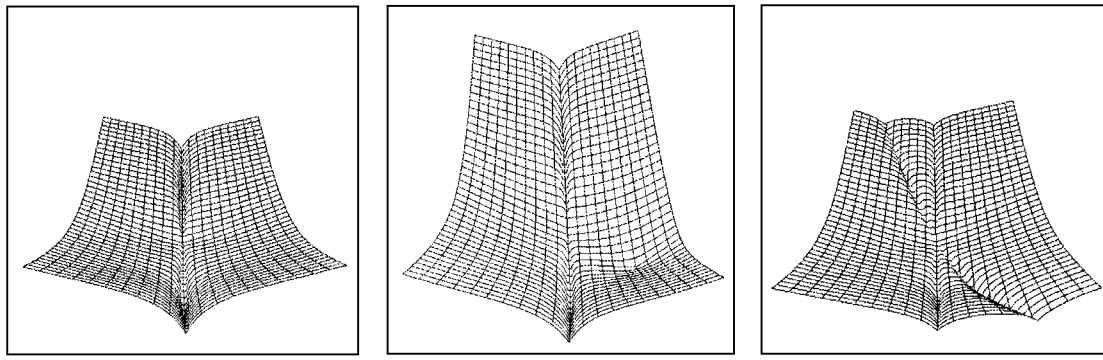

Figure 10. Plots of virtual structure displacement cost functions for: (a) one robot, (b) one robot with bias towards destination added and (c) two robots in different orientations.

work. In a real-world application, the minimization method most suitable for the problem instance would be chosen.

Next we present results of both simulation and experiments with real robots. We use simulation results primarily to illustrate the capability and flexibility of the algorithm. Virtual structure fitting errors are collected in experiments with real robots and proves that the solution proposed is capable of high-precision.

5. Simulation Experiments

The control algorithm was implemented on a Silicon Graphics Incorporated Indigo II High Impact workstation, which provides very fast three-dimensional graphical renderings of the simulated workspace. An accurate model of the R3 robot was used and the kinematic equations derived earlier were used to simulated the behavior of the robots when various wheel velocities were assigned. A screen shot is shown in Fig. 11 with a photograph of the real robots to illustrate the realism of the simulation.

Below we illustrate the behavior of the algorithm when the virtual structure is given commands to translate and rotate. A 'docking' simulation shows how the algorithm can potentially be applied to the coordinated motion control of multiple mobile robots. The fault-tolerance property of the algorithm will also be shown.

5.1. Translations and Rotations

The algorithm generates very smooth trajectories for virtual structure translation. Figure 12 shows a typical

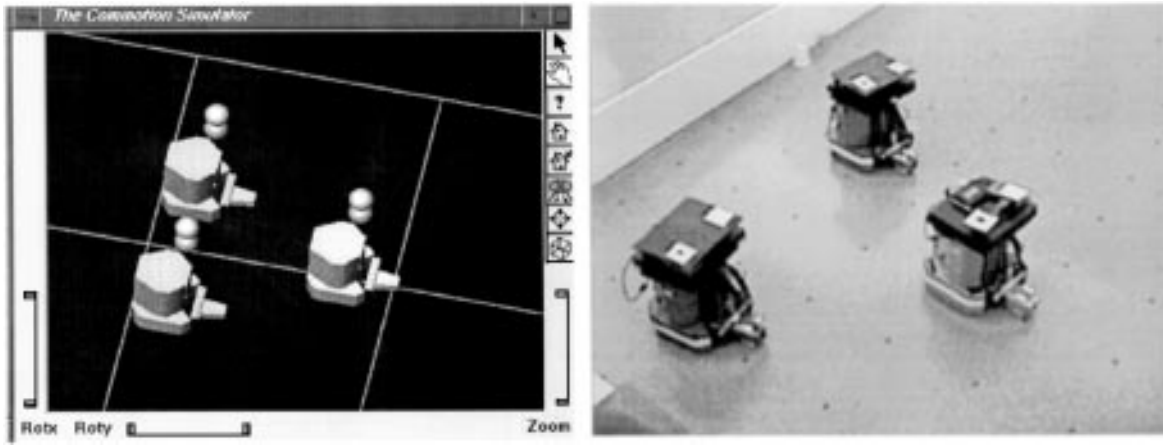

Figure 11. On the left is a screen shot of the graphical simulation used. The two spheres above the robots represent the position of the robot (lower sphere) and the position of the virtual structure (upper sphere). On the right is a photograph of the real robots with the added LINUX-based laptop computers and tracking features.

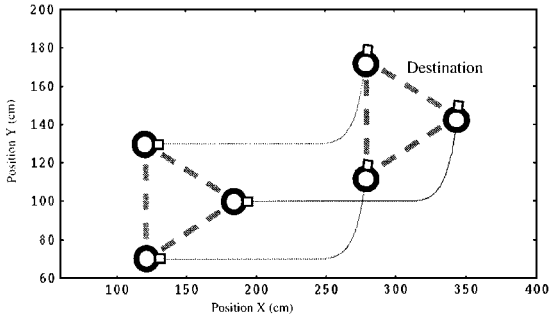

Figure 12. Trajectory followed by robots when virtual structure translates.

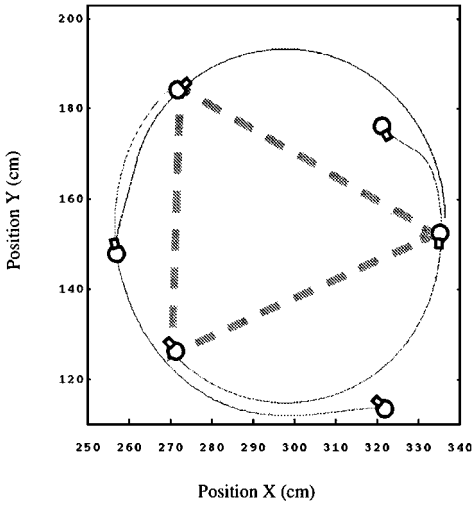

Figure 13. Trajectories of robots when the virtual structure rotates.

trajectory in which the virtual structure starts by moving in the direction in which all the robots were facing, and then gradually moving in the direction of the prescribed destination.

The Virtual Structure can also be easily commanded to rotate about a centroid. The equations similarly reorient the robots gradually as their capabilities permit and go into full-swing rotation when the robots are oriented for rotation, as illustrated in Fig. 13.

5.2. *Robot Failure*

One of the significant behaviors of the control algorithm is that when a robot fails, the virtual structure will not disintegrate. In a leader follower paradigm, a straggler would be left behind. To simulate a robot failure, we stop one of the robots while a command is given to move the virtual structure. The other robots

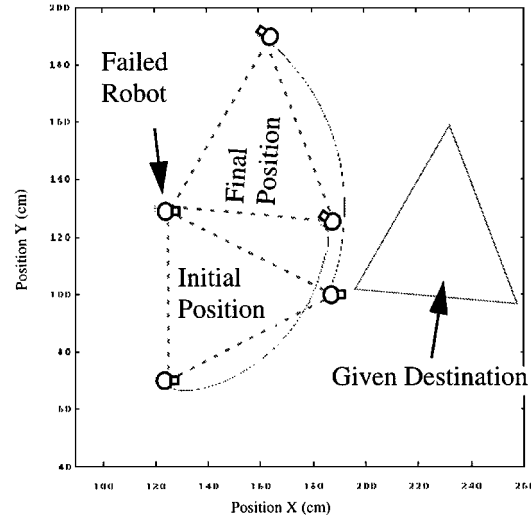

Figure 14. During a robot failure, the other robots adjust their paths to maintain formation.

have no information of the failure at all. If this were a leader-follower formation, the leader would have just moved off, leaving behind the failed robot. The formation disintegrates as a result. With the virtual structure algorithm, the robots can be seen in Fig. 14 to maintain the formation even when one of the robot fails. In fact the virtual structure as a whole rotates itself to be in the destination orientation specified, although the formation cannot move towards the given destination position without re-configuring. In real applications the robot failure can subsequently be detected since the virtual structure is not behaving as instructed, and higher-level decision processes can change the virtual structure as needed.

5.3. *Docking*

An interesting experiment to illustrate the flexibility of the virtual structure algorithm is that of 'docking'. In this experiment, two robots were initially in a virtual structure. The two robots then approach a third robot which is not in the formation and may possibly be moving. The three robots should then fall into the formation of a new three-robot virtual structure, and then move off in formation. There are two ways in which this can be done. The first method is to start with a virtual structure with three points and then slowly deform the virtual structure into the final desired formation. The second way is to simply tell the three robots to use as the new virtual structure the final desired formation.

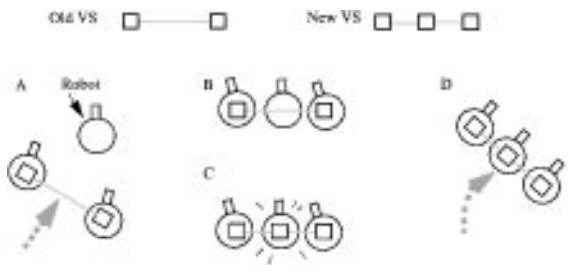

Figure 15. Steps in a docking operation: (A) the old VS approaches the third robot, (B) the old VS aligns itself with the third robot, (C) the new VS is imposed, (D) the new three-robot VS moves off. The configurations of the old and new VSs are shown on top.

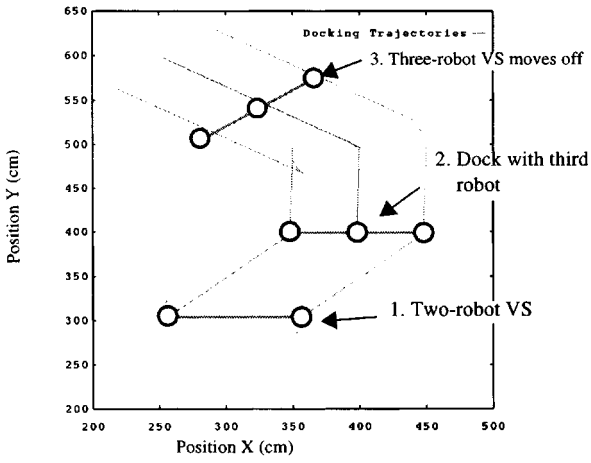

Figure 16. Trajectories during docking.

It is this method (shown in Fig. 15) that we demonstrate in this paper, and we present two simulations of the method. In the first simulation (Fig. 16), the two robots approach the third robot and position themselves such that the third robot is in its correct position in the new formation. The new virtual structure is then imposed, locking the three robots in formation. The new three-robot formation then moves off, rotates and heads in a new direction. The second simulation (Fig. 17) shows a more realistic scenario in which the new virtual structure is activated before the three robots are in position for the new formation. It can be seen that the robots converge to the new formation, and subsequently is able to move off and rotate as a new virtual structure. The only difference in this approach is that the virtual structure fitting error is initially high, and declines to a negligible level as the robots move *into* formation, as shown in Fig. 18.

6. Experiments with Real Robots

The control algorithm was tested in the mobile robotic testbed named R3Net (Zhen et al., 1996). The R3s were augmented with Linux-based computers, as shown in Fig. 19. This in turn enables the use of the AT&T WaveLAN wireless network for high-speed communications at 2 Mbits/s. For typical mobile robotic systems with less than ten robots exchanging control information with this setup eliminates communication bandwidth considerations. Video cameras mounted over the mobile robot workspace capture the positions of robots.

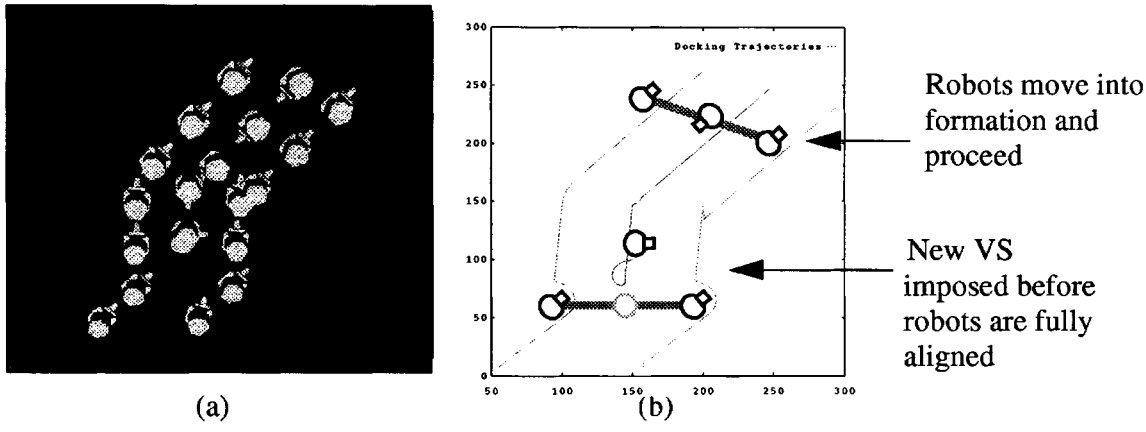

Figure 17. Docking in which the new virtual structure is imposed before the robots are completely aligned. Shown are: (a) graphical rendering of robots in selected time steps, (b) plot of trajectories followed.

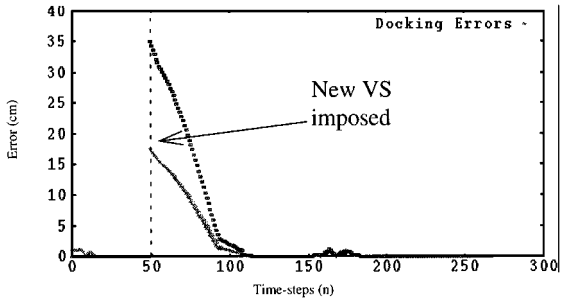

Figure 18. Errors during the second docking simulation. Imposing the new virtual structure before robots are fully aligned causes transient errors which drops off as robots move into formation.

A vision-based tracking system, developed in our lab, which we call VGPS (Vision-based Global Positioning System), segments the colored spots mounted on the mobile robots, and computes the position and orientation of each robot. This information is then read by the control algorithm running on a computation server networked to the tracking server. Wheel velocity commands for each individual robot are then sent to the robots through the WavePoint wireless gateway and executed on the robots. This complex system enables the development of mobile robotic algorithms and also supports the W3R3 system (Zhen et al., 1996; URL) which provides robotics researchers world-wide access to mobile robots in the Commotion Lab. The software used for controlling the real robots is identical to that used for simulation, except that the wheel velocities computed are now sent to the robots via R3Net. While the software is the same, experiments with real robots have the constraint that the space within which the robots can move is limited by the range of the tracking cameras.

6.1. Translation and Rotation

In order to measure the performance of the algorithm for the basic tasks of translating and rotating robot formations, we give commands to repeat the tasks ten times. In the case of translation, the commands make the formation move to-and-fro between two points. The data collected from a representative run is shown in Fig. 20. It can be seen that the mean error is below 1 cm. A similar experiment is conducted to investigate rotations, reorienting the virtual structure repeatedly. Results are shown in Fig. 21. The mean errors were generally larger than those incurred during translation, though still on the order of 1 cm.

6.2. Docking

The docking experiment illustrated in Fig. 22 demonstrates that the algorithm works in the real world as well. Two robots in formation are aligned with a third robot, forming a new virtual structure, which then moves off as a formation. The error behavior is exactly as predicted in the simulation studies.

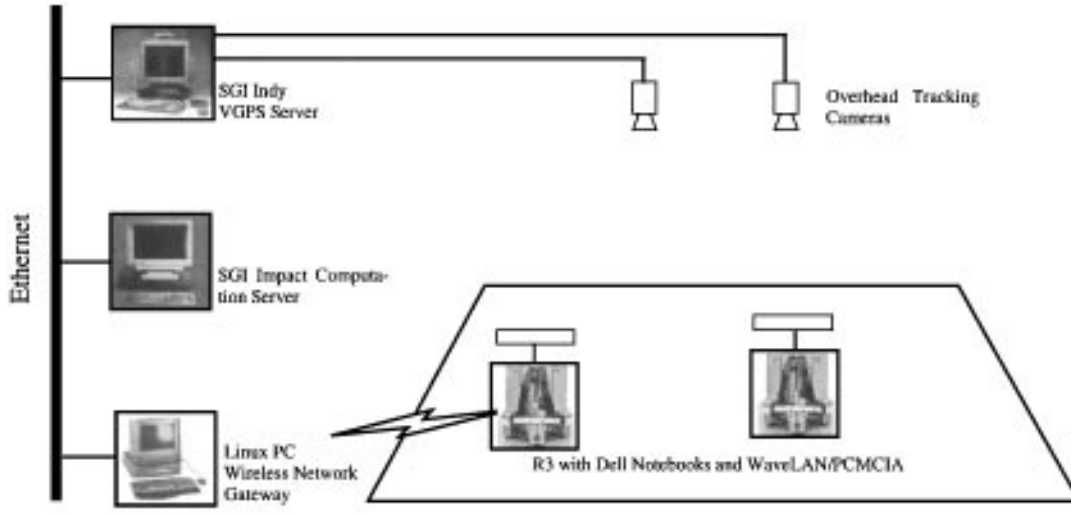

Figure 19. R3Net hardware overview.

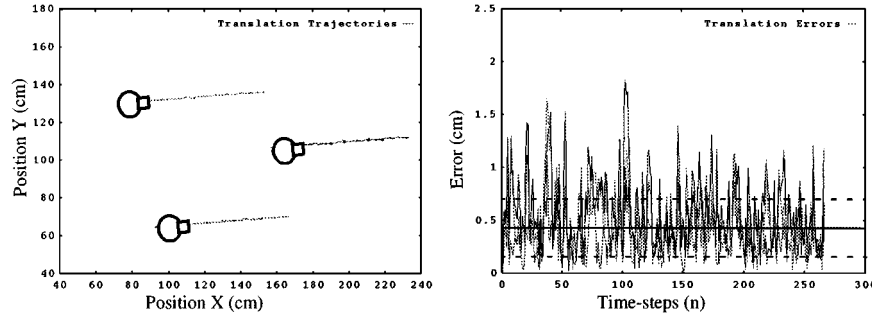

Figure 20. Translation trajectories and errors (mean = 0.4739 cm, standard deviation = 0.2941 cm).

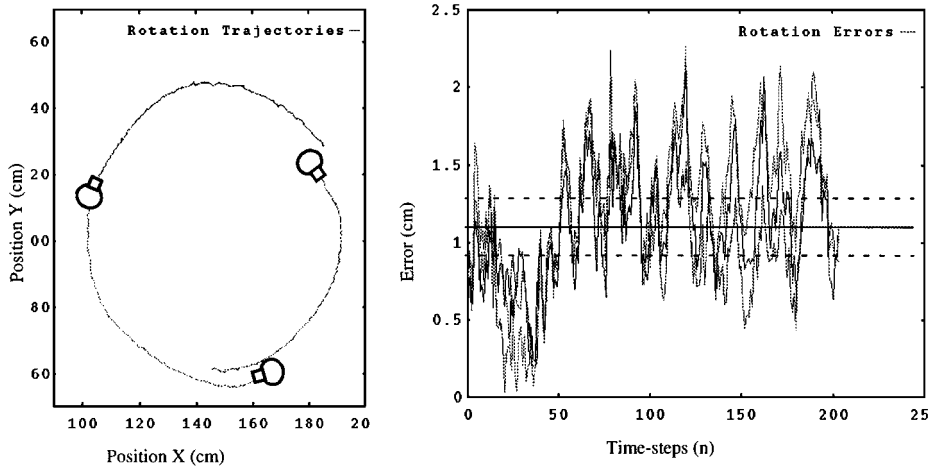

Figure 21. Rotation trajectories and errors (mean = 1.1499 cm, standard deviation = 0.2941 cm).

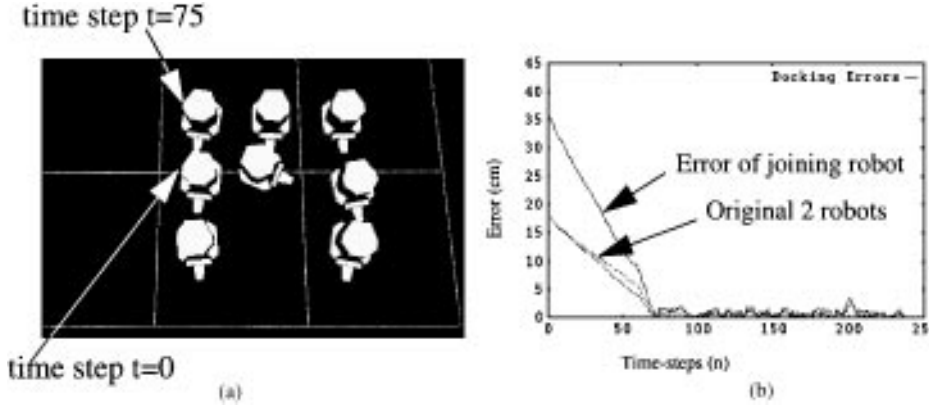

Figure 22. Docking experiment: (a) graphical rendering of the robots during the experiment and (b) measured error with time $t = 0$ corresponds to the moment of docking the virtual structure.

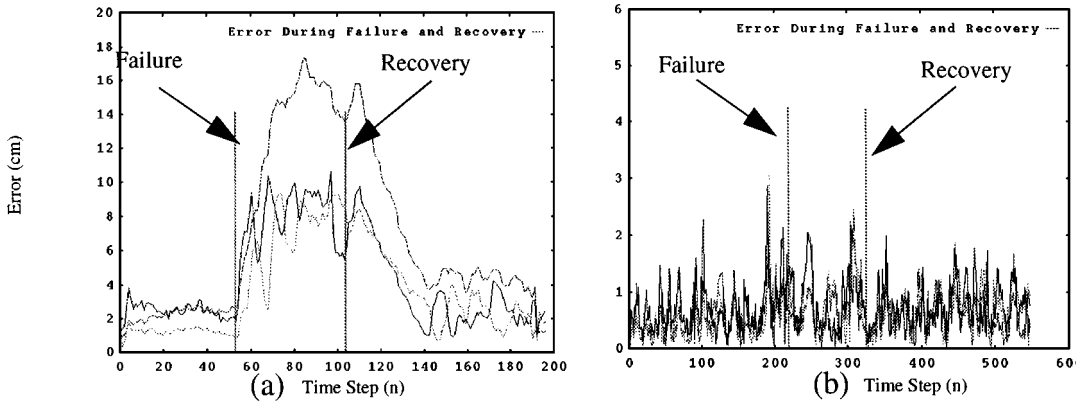

Figure 23. Error during failure: (a) formation is moving at high speed, (b) formation is moving slowly.

6.3. Robot Failure

Testing the behavior of the formation under failure is easier in the real world than in simulation: just turn off the power on one of the robots during a formation movement experiment. As predicted in simulation, the formation halts when a robot fails. The error behavior is shown in Fig. 23, which reveals a trade-off between the stabilizing error level and the speed at which the formation is moving. The harder the formation is pushed, the higher the error required to stop the formation.

6.4. Communication Requirements

The bandwidth requirements can be estimated by Eq. (3). In this instance $N = 3$, $T = 1$ seconds and $K = 3$ words or $K = 96$ bits (32 bits per word). Thus the required information transmitted is $BW = (3.96 \text{ bits})/(0.1 \text{ sec}) = (2.9 \text{ Kbits})/\text{sec}$. The bandwidth of the system we used is about 2 Mbits/sec.

6.5. The Issue of Maintaining Orientation

In the above work, the robots' position was controlled but the orientation was not explicitly under control of the VS algorithm. In the original formulation of the VS problem, the robots are treated as point objects, hence it is not surprising that orientation was not controlled.

7. Related Work

Most closely related is the work by Balch and Arkin (1995) on motor schema-based formation control. Their motivation is to control a collection of four robotic military vehicles moving in four formations defined in the US Army doctrine. A set of reactive behaviors called motor schemas were incorporated. A few possible control structures were proposed, including leader following, neighbor referencing, and unit-center referencing in which the mobile robots attempt to maintain relative position to the centroid of the formation obtained by averaging the positions of all robots. As the work was targeted towards military operations controlled by human drivers the leader following control was favored, since this reduces the sensing requirements of each robots—each robot only needs to track the leader and the leader does not have to worry about maintaining formation with the rest of the robots. However the simulations presented did reveal the property that the unit-center referencing method performs best in terms of fault tolerance and formation maintenance. Also, the motor schema paradigm is especially well suited to the incorporation of reactive collision avoidance.

Wang proposed control strategies that can employ a mixture of single leader following and multiple neighbor tracking (Wang, 1991). Other formation control algorithms use the leader following control architecture. Dudek and his colleagues described a method for leader following and investigated the cases in which there was only implicit communication, explicit one-way communication and two-way communication (Dudek et al., 1995). Testing was done with a pair of wheeled mobile robots. Similarly, Brock and associates described another formation maintenance algorithm which employs the leader following method (Brock et al., 1992) tested in simulation. Pachter and

his group presented a control strategy for an aircraft to follow another in a leader/wingman "diamond" formation (Pachter et al., 1994). Finally, Sheikholeslam describes the Platoon Problem in which vehicles follow a lead vehicle in order to maximize utilization of roads (Sheikholeslam and Desoer, 1992).

Another class of problem that appears different but is closely related is that of box pushing and load transportation (Hashimoto et al., 1995). The link is that in order to ensure continual effectiveness in pushing a box, mobile robots have to comply with certain geometric constraints that are similar to the problem of moving in formation. Also, most work in box pushing attempts to reduce hardware requirements by employing local sensing and implicit communication, which are recurrent themes in formation control. Donald and colleagues (—, 1995; Rus et al., 1995) took this to the limit and demonstrated the manipulation of furniture with mobile robots without explicit communication, global control and planning. Although the most important aspect of their work is on the theory of information invariants, which attempts to formalize the complexity of robotic tasks, it is interesting that the final protocols derived do not use any form of explicit communications. This is accomplished by exploiting the properties of the specific tasks of pushing a piece of furniture in a straight line or reorienting it in place. By sensing the relative position of the furniture and acting accordingly, the mobile robots can be thought of as communicating through the furniture: since the furniture itself is the only thing that matters in the state of the system, the robots can sense everything they need to know about the system by sensing the piece of furniture. The other way to look at this is that the robots are using the load, which is a *global entity*, for motion control.

Many researchers have also investigated the problem of moving *into* formation. As with the aforementioned works in cooperative robotics, the emphasis is on reduced sensing and asynchronous operation. Sugihara and Suzuki (1990) described a set of distributed, asynchronous protocols for the formation of a circle, line, open polygon and filled polygon. The circle algorithm is perhaps the most intriguing, for it does not always lead to the formation of a circle. This is due to the ambiguous distributed definition of a circle that they used: the furthest neighbor of a point forming a circle must be at a distance equal to the diameter of the desired circle. The property is unfortunately also satisfied by a shape known as Reuleaux's triangle. This illustrates the point that limiting sensing and communication ultimately limits the range of tasks the cooperative robotic system can perform, and can make task representations unnecessarily complex. Implicit in the distributed algorithm implementing this definition is the requirement for each robot to be able to sense the position of all other robots. This illustrates the point that while the decision protocols may be distributed, the load of accomplishing the given task can be shifted to sensing. More recently the formation controllability of mobile robot groups was proved (Yamaguchi and Arai, 1994). In this formulation, a formation is represented by a strongly connected graph, with mobile robots on each node maintaining a prescribed distance from neighbors. Mobile robots only need to sense neighbors on the graph, and in time will converge to a stable pattern. Intermediate configurations are naturally not guaranteed to be in formation.

From the above survey, it is clear that to achieve high-precision formation control the leader following method should not be used. Also, explicit communication protocols are not necessary if some implicit communications exploiting the task mechanics can be used. We have seen in a previous section a solution for formation control that does not use leader following, and thus achieving a high level of fault tolerance. Each robot can execute the same control algorithm asynchronously and maintain the global formation. Implicit communication is also achieved through a global entity. However in our solution the global entity does not have to physically exist—it is a *Virtual Structure* (Tan and Lewis, 1996).

8. Discussion

In the preceding sections we have presented a general solution for solving the problem of moving in formation using the virtual structure control algorithm. In an instance of the problem, illustrated through simulations and experiments, the following properties of the control algorithm were demonstrated:

Bidirectional Control. From the control algorithm, it is apparent that control does not flow exclusively from the virtual structure to the robots. Neither do the robots alone dictate the behavior of the system. There is an implicit bi-directional flow of control in the algorithm, as illustrated in Fig. 24.

High-Precision. It is shown by implementation on real robots that the control algorithm can achieve a high

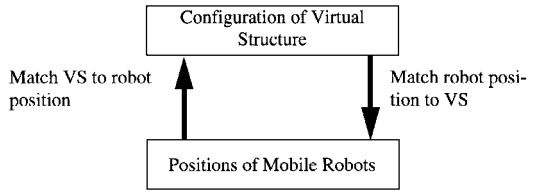

Figure 24. Bidirectional flow of control.

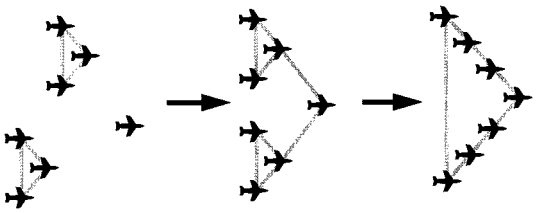

Figure 25. Virtual structures can be dynamically re-configured to add or drop robots, merge formations and reposition robots.

level of precision, even with robots not designed for high-precision applications.

Dynamic Re-Configuration. Robots can be added, dropped and repositioned on-the-fly, even when the formation is in motion. Entire virtual structures can be merged and split in the same way, as illustrated in Fig. 25. Through the operation of the control algorithm, robots eventually converge into a formation and maintain the virtual structure with high precision.

Inherently Fault Tolerant. When one or more robots in the system fail, it is important that the rest of the mobile robots do not let the faulty robots fall behind before some higher-level process can detect the failure and decide on the next action to take. It has been shown in both simulation and real experiments that the mobile robots collectively maintain adherence to the virtual structure in the case of a robot failure.

No Leader Election Required. Unlike many cooperative robotics control strategies, this solution does not rely on the existence of some leader among the robots. This increases the fault tolerance and avoids leader election, which is a hard problem in distributed algorithms.

Immediately Applicable to Different Kinds of Virtual Structures. As the algorithm does not rely on particular properties of geometric primitives such as circles or straight lines, the same method can be applied to virtual structures of various shapes with no modification.

Can be Implemented in a Distributed Fashion. Although, for development and data collection, we have implemented the algorithm on a centralized server, the fact that there is no task differentiation among robots in the algorithm implies that it can be duplicated and executed on each of the mobile robots with no increase in communications from a centralized implementation. Distributing the execution in turn adds one more layer of fault tolerance.

No Explicit Functional Decomposition. An interesting feature of the solution is that there are no complex protocols for communications or decision making. All the above features are implicit properties and emergent behaviors from a set of control equations.

Can be Used to Implement Leader Following as Well. By designating a particular robot as a leader and letting the virtual structure position be directly determined by the position of the leader, the algorithm is reduced to leader following.

9. Conclusion

The problem of moving in formation is stated using the concept of virtual rigid structures. A novel, extensible and effective method for high-precision formation control is proposed. The solution, also inspired by virtual structures, has a bidirectional control architecture which gives it a fault-tolerant quality: when robots fail, the entire system adjusts to maintain formation so that the failed robots are not left behind. This allows time for higher-level processes to reconfigure the virtual structure. Although the control algorithm has only been tested with robots capable of moving on a plane, there is no inherent limitation to its application to formation control in three dimensional space.

Acknowledgments

The authors would like to thank Romeo Elias for help in early experimentation. In addition, special thanks to Jian Zhen and other members of the commotion lab for R3Net and communication support.

This research was supported by NSF Grant CDA-9303148.

References

Arnold, V.I. 1989. *Mathematical Methods of Classical Mechanics*, 2nd edition, Springer Verlag: New York.

Balch, T. and Arkin, R.C. 1995. Motor schema-based formation control for multiagent robot teams. In *Proceedings of the First International Conference on Multiagent Systems*, pp. 19–16.

Brock, L., Montana, D.J., and Ceranowicz, A.Z. 1992. Coordination and control of multiple autonomous vehicles. In *Proceedings 1992 IEEE International Conference on Robotics and Automation*, pp. 2725–2730.

Cao, T., Fukunaga, A., Kahng, A.B., and Meng, F. 1995. Cooperative mobile robotics: Antecedents and directions. In *Proceedings 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 226–234.

Donald, B.R. On information invariants in robotics, *Artificial Intelligence*, 72(1–2):217–304.

Donald, B.R., Jennings, J., and Rus, D. 1995. Moving furniture with teams of autonomous robots. In *Proceedings 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 235–242.

Dudek, G., Jenkin, M., Milios, E., and Wilkes, D. 1995. Experiments in sensing and communication for robot convoy navigation. In *Proceedings 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vol. 2, pp. 268–273.

Hashimoto, M., Oba, F., and Eguchi, T. 1995. Dynamic control approach for motion coordination of multiple wheeled mobile robots transporting a single object. In *Proceedings 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1944–1951.

Johnson, P.J. and Bay, J.S. 1995. Distributed control of simulated autonomous mobile robot collectives in payload transportation, *Autonomous Robots*, 2(1):43–63.

Lissaman, P.B.S. and Shollenberger, C.A. 1970. Formation flight in birds, *Science*, 168:1003–1005.

Pachter, M., D'Azzo, J.J., and Dargan, J.L. 1994. Automatic formation flight control. *Journal of Guidance, Control, and Dynamics*, 17(6):1380–1383.

Press, W.H., Teukolsky, S.A., Vetterling, W.T., and Flannery, B.P. 1995. *Numerical Recipes in C: The Art of Scientific Computing*, Second edition, Cambridge University Press: New York.

Rus, D., Donald, B., and Jennings, J. 1995. Moving furniture with teams of autonomous robots. In *Proceedings 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 235–248.

Sheikholeslam, S. and Desoer, C.A. 1992. Control of interconnected nonlinear dynamical systems: The platoon problem. *IEEE Transactions on Automatic Control*, 37(6):806–810.

Sugihara, K. and Suzuki, I. 1990. Distributed motion coordination of multiple mobile robots. In *Proceedings 5th IEEE International Symposium on Intelligent Control*, pp. 138–143.

Tan, K.-H. and Lewis, M.A. 1996. Virtual structures for high-precision cooperative mobile robotic control. *1996 IEEE/RSJ International Conference on Intelligent Robots and Systems*.

Wang, P.K.C. 1991. Navigation strategies for multiple autonomous mobile robots moving in formation. *Journal of Robotic Systems*, 8(2):177–195.

Yamaguchi, H. and Arai, T. 1994. Distributed and autonomous control method for generating shape of multiple mobile robot group. In *Proceedings 1994 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 800–807.

Yamaguchi, Y., Tan, J.K., Ishikawa, S., and Kato, K. 1995. On the development of a cooperative work strategy by multiple robots. In *Proceedings 34th SICE Annual Conference 1995*, pp. 1453–1456.

Zhen, J.L., Lewis, M.A., and Tan, K.-H. 1996. Towards universal access to robotic resources. *1996 IEEE/RSJ International Conference on Intelligent Robots and Systems*.

URL: HTTP: //muster.cs.ucla.edu/w3r3.

M. Anthony Lewis is a visiting Assistant Professor at the Beckman Institute, University of Illinois. He received a B.S. in Cybernetics from the University of California, Los Angeles, and a M.S. and Ph.D. in Electrical Engineering from the University of Sourthern California in 1994 and 1996 respectively. His research areas include biologically inspired legged locomotion, visuomotor coordination and multi-robot systems. Prior to joining the University of Illinois, he was director of the U.C.L.A. Commotion Robotics Laboratory. Dr. Lewis has also worked at the California Institute of Technology's Jet Propulsion Laboratory in the area of anthropomorphic telemanipulators and at Hughes Aircraft in the area of flexible snake-like manipulators.

Kar-Han Tan was born in Singapore in 1970. He received the B.Sc. degree from the National University of Singapore in 1994, and the M.S. degree from the University of California, Los Angeles in 1996, both in computer science.

Formerly with the U.C.L.A. Commotion Lab, he is currently with the Artificial Intelligence Group, University of Illinois, Urbana-Champaign. His research interests include cooperative mobile robotics, realtime computer vision and computer graphics.