M. Brett McMickell
Bill Goodwine
Aerospace & Mechanical Engineering, University of Notre Dame
Notre Dame, IN 46556 USA
jgoodwin@nd.edu

Motion Planning for Nonlinear Symmetric Distributed Robotic Formations

Abstract

This paper develops a motion planning algorithm which exploits symmetries in distributed systems to reduce motion planning computation complexity. Symmetries allow for algebraic manipulations that are computationally costly, which normally must be carried out for each component in a distributed system, to be related among various symmetric components in a distributed system by a simple algebraic relationship. This leads to a large reduction in the complexity of the overall motion planning problem for a group of distributed mobile robotic agents. In particular, due to the manner in which a symmetric system is defined, the structure of the Chen–Fliess–Sussmann differential equations has a simple relationship among various symmetric components of a distributed system. Essentially, symmetries are defined in a manner which preserves the Lie algebraic structure of each component. In a system with distributed computational capability, the motion planning computations may be distributed throughout formation in such a way that the objectives of the formation are satisfied and collision avoidance is guaranteed. The algorithm maintains a rigid body formation at the beginning and end of the trajectory, as well as possibly specified intermediate points. Due to the generally nonholonomic nature of mobile robots, guaranteeing a rigid body formation during the intermediate motion is impossible. However, it is possible to bound the magnitude of the deviation from the rigid body formation at any point along the trajectory. Simulation and experimental results are provided to demonstrate the utility of the algorithm.

KEY WORDS—distributed robot systems, nonholonomic motion planning, path planning, multiple mobile robot systems

1. Introduction

This paper presents a distributed motion planning algorithm for formations consisting of a large number of robots. The motion planning algorithm developed is an extension of a well-known piecewise constant Lie-algebraic motion planning algorithm (Lafferriere and Sussman 1993). The main result in this paper is that the primary computational burden for motion planning for a system of agents which meet the required symmetry condition is essentially reduced to that for only one agent. In particular, the costly exercise (Kawski 2004) of computing the product of Lie exponential series and the subsequent algebraic manipulation of the resulting expression into the Chen–Fliess–Sussmann equations (as defined and outlined by (Murray et al. 1994) need only be done *once* for a group of robots that are appropriately related through a symmetry condition. While the method of Lafferriere and Sussman (1993) is far from optimal and only an approximate method for most systems, it is analytical, which makes possible the extension of the method in this paper.

If the appropriate symmetry conditions are satisfied, only *one* robot must complete the entire procedure to compute its trajectory using the Lie-algebraic method, which naturally makes the method especially amenable to distributed systems and distributed planning. Despite the superficial similarity that one robot has comparatively more computational burden, this method is not ‘leader-follower’ in the sense commonly used. The other robots do not depend upon the first robot successfully completing its maneuver or broadcasting any information during the maneuver. Also, *any* of the robots may be the one that is required to compute the entire trajectory, although the logical choice for which robot does so would be the robot with the most computational power. Thus, the main theoretical contribution of this paper is the fact that the nonlinear motion planning problem for a system of multiple symmetric robots *is effectively reduced to motion planning for one robot*.

The method that is extended by this paper, and therefore the extension of this method, is fundamentally open loop. However, if position sensing is possible, error correction is naturally incorporated into the algorithm as well as its extension. In particular, as outlined in Section 5, in order to guarantee collision avoidance, any large trajectory must be segmented into

The International Journal of Robotics Research
Vol. 26, No. 10, October 2007, pp. 1025–1041
DOI: 10.1177/0278364907082097

a sequence of smaller trajectories. Since the net motion will be implemented along these subtrajectories with a sequence of intermediate goal configurations, if position sensing is available, it can easily be incorporated into the plan for each of the subtrajectories.

Formation control of robotic systems has attracted a great deal of interest in the literature (Yamaguchi and Burdick 1998; Desai and Kumar 1999; Yamaguchi 1999; Egerstedt and Hu 2000; Chaimowicz et al. 2001; Leonard and Fiorelli 2001; Sugar et al. 2001; Yamaguchi et al. 2001; Olfati-Saber and Murray 2002; Fax and Murray 2004). However, many of these results consider only fully actuated robots (Yamaguchi 1999; Leonard and Fiorelli 2001; Yamaguchi et al. 2001) or are limited to a particular robot geometry (Yamaguchi and Burdick 1998). The goal of this work is to develop a general implementation of a formation control algorithm that may be applied to a large class of robotic systems, including nonholonomic robots. There has also been a great deal of research in the area of nonholonomic robot motion planning (Sussman 1991; Bates and Sniatycki 1993; Lafferriere and Sussman 1993; Murray and Sastry 1993; Teel et al. 1995; Laumond et al. 1998; Souères and Boissonat 1998; Chitta and Ostrowski 2002). For example, there have been motion planning algorithms developed using sinusoidal inputs (Murray and Sastry 1993; Teel et al. 1995) and piecewise-continuous inputs (Sussmann 1991; Laffierrere and Sussman 1993). However, there have been only a few efforts considering groups of nonholonomic robots working in a cooperative manner. The goal of our work is to develop methods for designing and implementing scalable motion planning algorithms for groups of nonholonomic robots.

This paper is structured as follows. In Section 2, the necessary background is outlined. Section 3 presents the theoretical development of the rigid body formation planning algorithm. A detailed example using a system of robots with nonholonomic kinematics such as that of a kinematic car is presented in Section 4. Collision avoidance is discussed in Section 5. Section 6 presents some experimental results and compares them with simulation results. Finally Section 7 presents conclusions and discusses possible avenues of future work.

2. Background

This section provides necessary background material from the authors' previous work (McMickell and Goodwine 2001, 2002, 2003a, 2003b; McMickell 2003) necessary to formulate our motion planning algorithm for symmetric nonlinear distributed systems. Only the main ideas as well as a computational means to determine a symmetry are presented here and the reader is referred to the references for a complete exposition. We also review the motion planning algorithm that is used by this work (Lafferriere and Sussman 1993). Examples of the computations necessary to implement the methods from this section are presented in subsequent sections along with simulation results.

2.1. Symmetric Nonlinear Systems

For most distributed robotic systems, the configuration manifold M is naturally partitioned into a set of n regular manifolds M_i, $i \in \{1, \dots, n\}$, such that M is the Cartesian product of the M_i, i.e. $M = \prod_{i=1}^{n} M_i$. In a system comprised of multiple robots, each M_i represents the configuration space for an individual robot in the formation. We further assume that associated with each M_i is a set of control inputs $\{u_i^1, u_i^2, \dots\}$, and that each input is associated with only one M_i. This is natural since actuators in a robot will typically only affect that robot and not other robots. Thus, we can define a *component*, denoted V_i, of the overall system to be the manifold M_i which is its configuration space and the associated control inputs, i.e. $V_i = (M_i, \{u_i^1, u_i^2, \dots\})$ and denote the ordered set of components $\mathbf{V} = \{V_i | i = 1, \dots, n\}$.

We will consider analytic driftless systems of the form

$$\begin{aligned} \Sigma : \qquad \dot{x} &= g_1^1(x)u_1^1 + g_1^2(x)u_1^2 + \cdots \\ &+ g_2^1(x)u_2^1 + g_2^2(x)u_2^2 + \cdots \\ &\vdots \\ &+ g_n^1(x)u_n^1 + g_n^2(x)u_n^2 + \cdots \qquad x \in M, \end{aligned} \tag{1}$$

where the g_i^j are smooth analytic vector fields on M. In Equation 1, the subscript on the g's and u's indicates the component to which the control input corresponds, and the superscript enumerates different inputs within that component.

Let σ denote a permutation on the set of n symbols $\{1, 2, \dots, n\}$. This naturally gives rise to an induced permutation of the ordered elements of M given by

$$\sigma_\#(M) = \prod_{i=1}^{n} M_{\sigma(i)}.$$

Since the Cartesian product is *ordered*, in general $M \neq \sigma_\#(M)$ unless σ is the identity or, in the case that is of interest in this paper, when the underlying manifolds M_i that are permuted by $\sigma_\#$ are either the same or diffeomorphically related. This latter case corresponds to when, in a system of multiple robots, the robots are the same or perhaps scaled versions of each other and the action of $\sigma_\#$ corresponds to interchanging the robots. Finally, vector fields on M are related to vector fields on $\sigma_\#(M)$ by the usual push forward of $\sigma_\#$ given by

$$(\sigma_\#)_* g_i^j = T\sigma_\# \circ g_i^j \circ \sigma_\#^{-1}.$$

In this case g_i^j is a vector field on M and $(\sigma_\#)_* g_i^j$ is a vector field on $\sigma_\#(M)$.

The fundamental idea motivating this work is that it is naturally the case in many distributed systems that many of the components of the system are identical or nearly identical and interact with other components in a (nearly) identical manner. In such a case it may be possible to *physically interchange* components without substantially altering the properties of the system. This will be represented by the fact that the equations of motion remain invariant under the action of $(\sigma_\#)_*$. In other words, the equations of motion are such that the system on M is the same as the system on $\sigma_\#(M)$, then the original system is *invariant* to interchanging components.

The means to test whether or not a system is symmetric in this sense is to determine whether or not some vector fields remain invariant under the action of $(\sigma_\#)_*$. In particular, two vector fields are defined to be *equivalent*, denoted by $g_i^j \sim g_r^j$ if there exists a $\sigma_\#$ such that

$$g_i^j = (\sigma_\#)_* g_r^j.$$

Note that this equality only makes sense in the case when $M = \sigma_\#(M)$ and that even checking this equality will generally not be possible when this is not the case.

Components V_i and V_r are said to be *symmetric* if for each j there exists a k and a $\sigma_\#$ such that

$$g_i^j = (\sigma_\#)_* g_r^k.$$

The reason for the different superscripts j and k is that the inputs on components i and r may not be ordered in the same way. In the usual case where they are, then the two components are symmetric when

$$g_i^j = (\sigma_\#)_* g_r^j,$$

for all the inputs u_i^j defined on components i and r.

This definition only encapsulates the notion of when two components are interchangeable. However, it is clear that a slightly weaker form of symmetry will probably still be useful. An example of this would be when two components are not exactly the same, but rather scaled versions of each other. In fact, as will be clear subsequently, all that is required is that they be diffeomorphically related. In light of this we will extend the definition of symmetric components to be that two components are symmetric if there exists an automorphism $f : M \to M$ such that

$$g_i^j = ((\sigma_\#)_* \circ f_*)\left(g_r^j\right).$$

For simplicity, in the subsequent development we will only include $\sigma_\#$ notationally in the computations and if an automorphism f is necessary for the system to be symmetric, it will be notationally absorbed into $\sigma_\#$.

An additional complication is that the vector fields for the system may depend on parameters other than the states, for example, the physical parameters (lengths, etc.) of a robot. It is desirable that our notion of symmetry encompasses the case where robots are not necessarily identical, but are related by a simple variation in physical parameters that does not alter the fundamental nature of the system. An example of this would be having two robots where one is simply twice the size of the other. We will denote the set of parameters corresponding to component i by p_i and extend the definition of symmetric vector fields to

$$g_i^j = \left((\sigma_\#)_* g_r^j\right)|_{p_i},$$

which denotes that the parameter values for component i are substituted for component r. We will limit our attention to systems with parameter values that do not change with time, and hence in all the computations that follow a simple substitution of parameter values is all that is necessary.

A subset $\mathbf{O_i} \subset \mathbf{V}$, called a *symmetry orbit*, is the set of components that have equivalent vector fields associated with them. Systems that satisfy the above assumptions are referred to as symmetric distributed systems.

2.2. Piecewise Constant Motion Planning

The motion planning algorithm developed in this paper is an extension of piecewise constant motion planning algorithm by (Lafferriere and Sussman 1993). A complete description of this motion planning algorithm is beyond the scope of this paper, so only an outline will be provided in this section. It is important to note that this method works exactly only for nilpotent systems. For systems that are not nilpotent, the method is approximate.

The basic idea of Lafferriere and Sussman (1993) is to decompose the desired trajectory into multiple subtrajectories along vector fields which, when evaluated at a point, form a basis for the tangent space of the configuration space. For underactuated systems, the basis will contain motion in a Lie bracket direction. Recall that a Lie bracket in coordinates is given by

$$[g^1, g^2] = \frac{\partial g^2}{\partial x} g^1 - \frac{\partial g^1}{\partial x} g^2.$$

Motion in a Lie bracket direction can be approximated using the following four segment flow,

$$\phi^t_{[g^1,g^2]}(x_0) = \phi^{\sqrt{t}}_{-g^2} \circ \phi^{\sqrt{t}}_{-g^1} \circ \phi^{\sqrt{t}}_{g^2} \circ \phi^{\sqrt{t}}_{g^1}(x_0), \tag{2}$$

where $\phi_g^t(x_0)$ represents the flow along the vector field g for time t starting at point x_0.

Given an underactuated control affine system of the form

$$\begin{aligned} \dot{x} &= g^1(x)u_1 + \cdots + g^m(x)u_m, \\ x &\in M, \quad \dim M = s > m, \end{aligned} \tag{3}$$

it is well known that the system is controllable if there exist Lie brackets such that

$$\dim\left(\{g^1, g^2, \ldots, g^n, [g^1, g^2], [g^1, g^3], \ldots, [g^1, [g^1, g^2]], \ldots\}\right) = s.$$

The method requires two basic steps. First, an *extended system* is constructed, which is of the form

$$\begin{aligned}\dot{x} &= g^1(x)v^1 + g^2(x)v^2 + \cdots + g^m(x)v^m \\ &+ g^{m+1}(x)v^{m+1} + \cdots + g^s(x)v^s, \end{aligned} \tag{4}$$

where the g^i for $i > m$ are Lie brackets among the vector fields in the original control system in Equation 3 such that the distribution $\Delta = \operatorname{span}(g^1, \ldots, g^s)$ is full rank. The v^i's are called *fictitious inputs* since they do not correspond to inputs in the real system for $i > m$. The reason for constructing the extended system is that motion planning is trivial for it because, given a desired trajectory, $\gamma(t)$,

$$\dot{\gamma}(t) = g^1(\gamma(t))v^1 + g^2(\gamma(t))v^2 + \cdots + g^s(\gamma(t))v^s$$

so

$$\begin{bmatrix} v^1 \\ v^2 \\ \vdots \\ v^s \end{bmatrix} = \begin{bmatrix} g^1(\gamma(t)) & g^2(\gamma(t)) & \cdots & g^s(\gamma(t)) \end{bmatrix}^{-1} \dot{\gamma}(t).$$

Note that since this matrix must be inverted as a function of desired trajectory, and hence a function of t, it must either be done analytically or numerically many times corresponding to many different points in time. One benefit from the analysis in this paper will be the need to compute this inverse only one time for symmetric components in a distributed system.

The second step in the method is to associate with each vector field g^i, $i = 1, \ldots, s$ an indeterminant B_i. In indeterminants, Lie brackets are of the form $[B_i, B_j] = B_iB_j - B_jB_i$. A main theoretical development in (Lafferriere and Sussman 1993) is to relate the solution of Equation 4 in M to solutions of the *formal differential extended system*

$$\dot{S}(t) = S(t)(B_1v^1 + \cdots + B_sv^s), \tag{5}$$

where $S(t)$ is the series representation of a given trajectory in the space of indeterminants. In particular, flows along vector fields in M, $\phi^t_{g^i}$ are related to exponentials in indeterminants, i.e. e^{B_it}. The main utility of using indeterminants is that it is possible to expand the exponential terms in the usual convenient manner,

$$e^{B_it} = 1 + B_it + \frac{1}{2}(B_it)^2 + \frac{1}{6}(B_it)^3 + \cdots, \tag{6}$$

which is not justified for vector fields since the square of a vector field is a differential operator but not another vector field.

It is the case that any smooth trajectory can be represented by the Chen–Fliess series

$$S_t(q) = e^{h_s(t)B_s}e^{h_{s-1}(t)B_{s-1}} \cdots e^{h_2(t)B_2}e^{h_1(t)B_1}, \tag{7}$$

where $h_1, \ldots, h_s$ are functions know as the (backward) *Philip Hall coordinates* and ultimately are related to the total time that the system must flow along each vector field to accomplish a given motion.

By differentiating Equation 7, computing expansions of each term in the form of Equation 6 up to a sufficiently high order, grouping terms into linear combinations that correspond to Lie brackets in the Lie algebra of indeterminants and equating it to Equation 5, we can solve for the $\dot{h}$'s in terms of the fictitious inputs, which results in a set of ordinary differential equations,

$$\dot{h} = Q(h)v, \quad h(0) = 0, \tag{8}$$

where $Q(h)$ is a coefficient matrix in terms of the Philip Hall coordinates and h and v are vectors composed of the individual components h_i and v^i, respectively.

It is emphasized that except for the simplest of systems, this step is burdensome since each term in Equation 7 must be expanded, then each expansion must be distributed over all the other expansions and then terms corresponding to a Lie brackets in indeterminants must be identified. It is also fundamentally an algebraic manipulation, so is not of the nature that may be easily automated numerically. In the case of a symmetric system of robots, repeating this step for each individual robot is what is avoided by making use of the symmetries in the system.

It is easier to determine the real inputs using forward rather than backward Philip Hall coordinates. The formal system corresponding to forward Philip Hall coordinates is

$$S_t(q) = e^{\tilde{h}_1(t)B_1}e^{\tilde{h}_2(t)B_2} \cdots e^{\tilde{h}_s(t)B_s}. \tag{9}$$

An algebraic transformation relates the forward Philip Hall coordinates to the backward Philip Hall coordinates. This transformation is calculated by equating Equations 7 and Equation 9 to obtain

$$e^{h_s(t)B_s} \cdots e^{h_2(t)B_2}e^{h_1(t)B_1}(q_0) = e^{\tilde{h}_1(t)B_1}e^{\tilde{h}_2(t)B_2} \cdots e^{\tilde{h}_s(t)B_s}(q_0),$$

expanding the exponentials and equating coefficients of the Philip Hall basis elements to get

$$\sum_{j=1}^{s} \tilde{p}_{j,k}(\tilde{h}) = \sum_{j=1}^{s} p_{j,k}(h),$$

where $\tilde{p}_{j,k}(\tilde{h})$ and $p_{j,k}(h)$are polynomial functions in terms of forward and backward Philip Hall coordinates, respectively.

3. Piecewise Motion Planning for Rigid Body Formations

The motion planning problem that is considered in this paper is as follows. Given m robots in a formation, determine an open

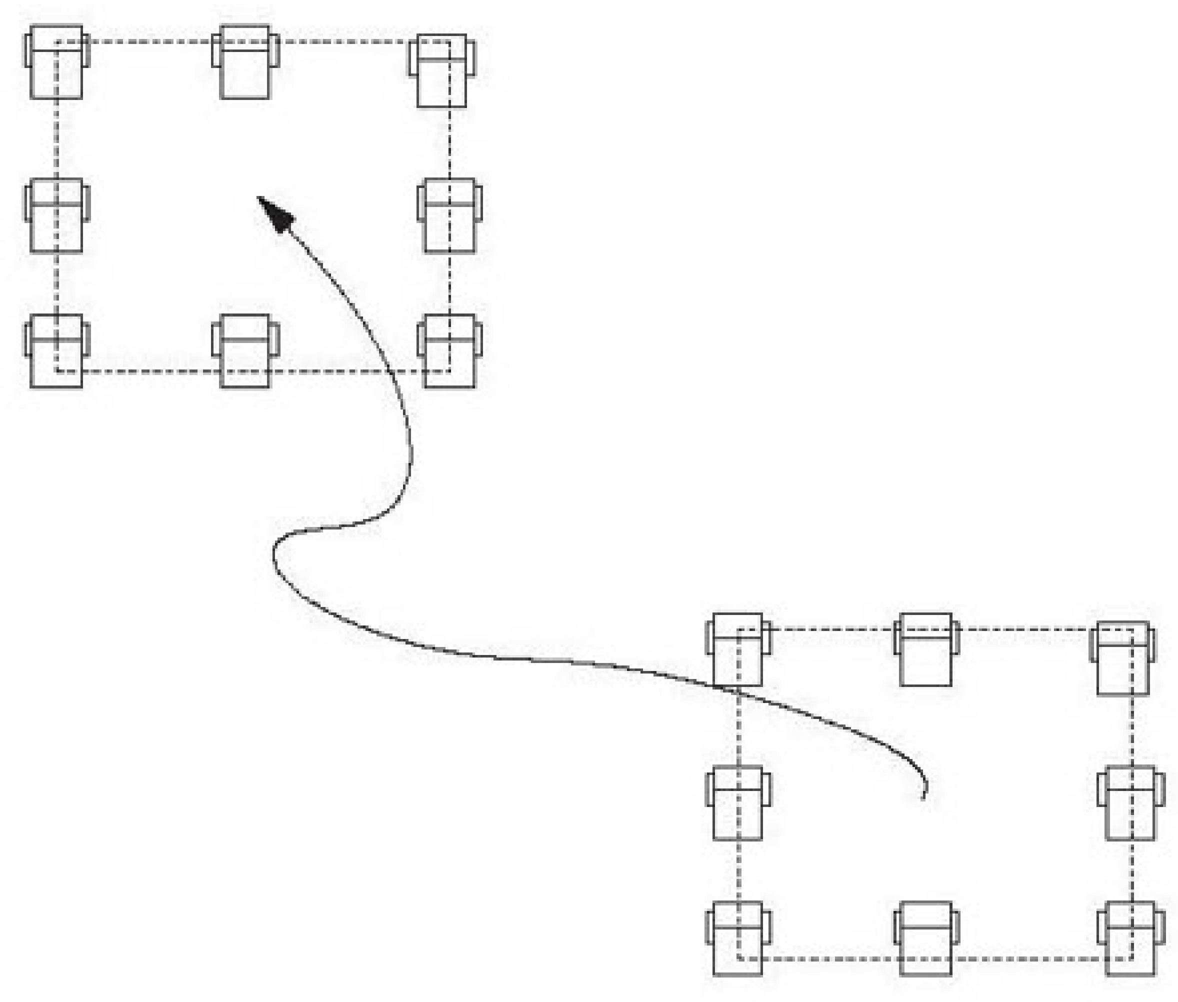
Fig. 1. Example of a rigid body of robots.

loop control $t \mapsto u(t)$ that steers the formation from any initial position and orientation to any final position and orientation while maintaining the relative positions of the robots in the formation. That is, the formation begins and ends its motion as a rigid body as shown in Figure 1. The motion planning of the entire formation is partitioned into motion planning of each robot individually. We then show that the motion plans between any two robots in a symmetry orbit are related by a symmetry operator.

For a nonlinear symmetric system of q robots in a rigid formation, motion planning may be accomplished as follows. Let each robot consist of n states, so that $\dim M_i = n$ and $\dim M = n \times q$. We consider robots where the dynamics of the ith robot is of the form

$$\Sigma_i: \quad \dot{x}_i = \sum_{j=1}^{m_i} g_i^j u_i^j. \tag{10}$$

Given a desired final position and orientation for the formation, we must determine the trajectory of the ith robot in order to apply the piecewise motion planning algorithm. The first step is to determine a rotation matrix

$$R(\omega t) \in \underbrace{SO(2) \oplus \cdots \oplus SO(2)}_{\text{q times}} = SO(2q)$$

such that $R(\omega T)$ produces the desired final orientation of the rigid formation. (This R is simply q copies of a 2×2 rotation in block diagonal form in a matrix representation). The frequency ω is simply the rotational velocity of the rigid formation. Next, choose a mapping $q(t) \in C^1$ connecting the initial center of the formation to a desired final center for a given $t \in [0, T]$. This can be done using any C^1 function, and often a straight line will suffice. Note that $q(t) \in \mathbb{R}^{2\times q}$. The trajectory of the rigid body is given by

$$p(t) = R(wt)P + q(t), \tag{11}$$

where $P \in \mathbb{R}^{n\times q}$ is a vector of the initial position of each of the robots in the formation relative to the center of the formation. The trajectory of the ith robot in the rigid body is given by

$$p_i(t) = \pi_i(R(\omega t)P + q(t)) = R_i(\omega t)P_i + q_i(t), \tag{12}$$

where $P_i \in \mathbb{R}^n$ is the initial position of robot i relative to the center of the formation and π_i is the projection that simply returns the components of $p(t)$ corresponding to the ith robot.

The rigid formation uniquely determines the position of each robot, but it does not constrain the robot's orientation within the formation. Let $r(t) \in C^1$ describe the changing orientation of the robots. The rigid body trajectory for the ith robot is,

$$\gamma_i(t) = A_i(wt)\hat{P}_i + \hat{q}_i(t) + \hat{r}_i(t),$$

where $\hat{P}_i$ is the initial state of the ith robot (position and orientation) with respect to the center of the rigid body, $A_i(wt)$ is an augmented rotation given by

$$A_i(wt) = \begin{bmatrix} R_i(wt) & 0 \\ 0 & 1 \end{bmatrix},$$

$\hat{q}_i(t)$ is an augmented trajectory given by

$$\hat{q}_i(t) = \begin{bmatrix} q_i(t) \\ 0 \end{bmatrix},$$

and $\hat{r}_i(t)$ is given by

$$\hat{r}_i(t) = \begin{bmatrix} 0 \\ r_i(t) \end{bmatrix}.$$

Taking the derivative of the trajectory, we find

$$\dot{\gamma}_i(t) = \dot{A}_i(wt)\hat{P}_i + \dot{\hat{q}}_i(t) + \dot{\hat{r}}_i(t).$$

The construction thus far only includes the components of the trajectory that represent the rigid body motion of the system in $SE(2)$ (Special Euclidean Group). If there are other components, such as internal configuration variables for each robot, they may be simply appended to $\gamma_i(t)$. This is subsequently illustrated in the kinematic car example.

Considering one of the symmetric components, say V_i, we can form an extended system consisting of s linearly independent vector fields,

$$\dot{x} = g_i^1 v_i^1 + \cdots + g_i^m v_i^m + g_i^{m+1} v_i^{m+1} \cdots + g_i^s v_i^s,$$

where the v_i's are fictitious inputs and the g_i's are selected such that $\Delta_i = \text{span}\left(\pi_i g_i^1, \pi_i g_i^2, \ldots, \pi_i g_i^s\right)$ is full rank, where π_i is the projection onto the ith component (i.e. it just picks off the components from the very large vector fields that correspond to the states in component i). In order to determine the fictitious inputs, define an ordered matrix C_i composed of vector fields that are linearly independent for all $\gamma_i(t)$,

$$C_i\left(\gamma_i(t)\right) = [\pi_i g_i^1\left(\gamma_i(t)\right), \pi_i g_i^2\left(\gamma_i(t)\right), \ldots, \pi_i g_i^s\left(\gamma_i(t)\right)].$$

Recall that if component V_i is underactuated, some of the g's will be Lie brackets between vector fields in the original system. The vector fields in C_i were chosen so they have full rank over the entire trajectory. Therefore, C is invertible and fictitious inputs for robot i are given by

$$v_i = C_i^{-1}\left(\gamma_i(t)\right)\dot{\gamma}_i(t),$$

where $v_i = [v_i^1, \ldots, v_i^s]^T$. The backward Philip Hall coordinates are determined by solving the Chen–Fliess–Sussmann equation,

$$\dot{h}_i = Q_i(h_i)v_i, \tag{13}$$

with the initial condition $h(0) = 0$. The main result of this paper is that the backward Philip Hall coordinates for the ith robot can be extended to other robots in its symmetry orbit using the following theorem.

Theorem 3.1 *Let Σ be a robotic system containing q robots in the form of Equation 10. If Σ is a symmetric distributed system, then the Philip Hall coordinates of any two robots, i and r, in the same symmetry orbit of the system with a desired trajectory given by Equation 12, are related by*

$$\begin{aligned}\dot{h}_r &= Q_r(h_r)v_r \\ &= Q_i(h_r)C_i^{-1}\left((\sigma_\#)^{-1}\gamma_r(t)\right)\dot{\gamma}_r(t),\end{aligned} \tag{14}$$

where $\gamma_r(t) = \pi_r\left(R(\omega t)P_r + q_i(t)\right)$ and $\sigma(V_r) = V_i$. If components r and i have different parameter values, then they must be substituted into C_i^{-1} and hence in that case

$$\dot{h}_r = Q_i(h_r)C_i^{-1}\left((\sigma_\#)^{-1}\gamma_r(t)\right)\Big|_{p_r}\dot{\gamma}_r(t).$$

Before presenting the proof, observe that while the trajectory of the rth robot must be used, the Lie algebraic structure of the ith robot may be used for computing the trajectory of the rth robot. In particular, this is represented by the Q_i and C_i terms. The C_i term may be used because the extended systems can be related through the symmetry operator $\sigma_\#$ and the Q_i term may be used since the form of the Chen–Fliess–Sussmann equations only depends upon the algebraic structure of the Lie algebra of indeterminants, which will be the same for symmetric components. The computational savings come about through both terms. Being able to use Q_i eliminates the burden of constructing the Chen–Fliess–Sussmann equations for the rth component, and using C_i^{-1} eliminates the need to invert C_r, which would have to be done analytically. The additional computations needed are related to $\sigma_\#$, but these are easy to compute since they are simply permutations.

Proof. Assume that robot i and robot r are in the same symmetry orbit, that is, there exists an induced permutation $\sigma_\sharp$ such that

$$\begin{aligned}g_r^j &= (\sigma_\sharp)_* g_i^j \\ &= T\sigma_\# \circ g_i^j \circ (\sigma_\#)^{-1}\end{aligned}$$

for all g_i^j and g_r^j.

The proof relies on two facts that result from the symmetry. First, it is a basic result from differential geometry that the push forward is natural with respect to Lie brackets, i.e.

$$(\sigma_\sharp)_*[f, g] = [(\sigma_\sharp)_* f, (\sigma_\sharp)_* g].$$

Since $\Delta_i = \text{span}\left(g_i^1, g_i^2, \ldots, g_i^s\right)$ is full rank, Δ_r may be constructed as

$$\begin{aligned}\Delta_r &= \text{span}\left((\sigma_\#)_* g_i^1, (\sigma_\#)_* g_i^2, \ldots, (\sigma_\#)_* g_i^s\right) \\ &= \text{span}\left(g_r^1, g_r^2, \ldots, g_r^s\right).\end{aligned}$$

Since $\sigma_\#$ is a diffeomorphism and Δ_i is full rank, then Δ_r must be full rank and hence

$$\dot{x}_r = g_r^1 v_i^1 + \cdots + g_r^{m_i} v_i^{m_i} + g_r^{m_i+1} v_i^{m_i+1} \cdots + g_r^s v_i^s \tag{15}$$

$$\begin{aligned}&= (\sigma_\sharp)_* g_i^1 v_i^1 + \cdots + (\sigma_\sharp)_* g_i^{m_r} v_i^{m_r} \\ &+ (\sigma_\sharp)_* g_i^{m_r+1} v_i^{m_r+1} + \cdots + (\sigma_\sharp)_* g_i^s v_i^s\end{aligned} \tag{16}$$

is an extended system for the rth component where $m_r = m_i$. Hence, C_r is given by

$$\begin{aligned}C_r\left(\gamma_r(t)\right) &= \left[\pi_r g_r^1\left(\gamma_r(t)\right) \quad \pi_r g_r^2\left(\gamma_r(t)\right) \right. \\ &\qquad \left. \cdots \quad \pi_r g_r^s\left(\gamma_r(t)\right)\right] \\ &= \left[\pi_r\left((\sigma_\#)g_i^1\right)\left(\gamma_r(t)\right) \quad \pi_r\left((\sigma_\#)g_i^2\right)\left(\gamma_r(t)\right)\right. \\ &\qquad \left. \cdots \quad \pi_r\left((\sigma_\#)g_i^s\right)\left(\gamma_r(t)\right)\right] \\ &= \left[\pi_r\left(T\sigma_\# g_i^1(\sigma_\#)^{-1}\right)\right. \\ &\qquad \left. \pi_r\left(T\sigma_\# g_i^2(\sigma_\#)^{-1}\right)\left(\gamma_r(t)\right) \quad \cdots\right].\end{aligned}$$

To simplify the notation somewhat, note that the *components* of the matrix are given by the $T\sigma_\# g_i^j$ terms and that each component is a function of $(\sigma_\#)^{-1}\left(\gamma_r(t)\right)$. Hence, this may be rewritten as

$$\begin{aligned}C_r\left(\gamma_r(t)\right) &= \left[\pi_r T\sigma_\# g_i^1 \quad \pi_r T\sigma_\# g_i^2 \quad \cdots \quad \pi_r T\sigma_\# g_i^s\right] \\ &\times \left((\sigma_\#)^{-1}\gamma_r(t)\right) \\ &= \pi_r(T\sigma_\#)\left[g_i^1 \quad g_i^2 \quad \cdots \quad g_i^s\right]\left((\sigma_\#)^{-1}\gamma_r(t)\right) \\ &= \pi_i\left[g_i^1 \quad g_i^2 \quad \cdots \quad g_i^s\right]\left((\sigma_\#)^{-1}\gamma_r(t)\right) \\ &= C_i\left((\sigma_\#)^{-1}\gamma_r(t)\right).\end{aligned}$$

Furthermore and finally, we have

$$C_r^{-1}\left(\gamma_r(t)\right) = C_i^{-1}\left((\sigma_\#)^{-1}\gamma_r(t)\right)$$

and hence

$$\begin{aligned}\dot v_r &= C_r^{-1}\left(\gamma_r(t)\right)\dot\gamma_r(t)\\ &= C_i^{-1}\left((\sigma_\#)^{-1}\gamma_r(t)\right)\dot\gamma_r(t).\end{aligned}$$

Since it follows from the construction of the Chen–Fliess–Sussmann equations that the form of the differential equations in indeterminants only depends upon the dimension of the configuration space and the number of inputs, $Q_r = Q_i$ and the form of the differential equations for the backward Philip Hall coordinates for component r is

$$\begin{aligned}\dot h_r &= Q_r(h_r)v_r\\ &= Q_i(h_r)C_i^{-1}\left((\sigma_\#)^{-1}\gamma_r(t)\right)\dot\gamma_r(t). \quad \blacksquare\end{aligned}$$

4. Symmetry Analysis for a Fleet of ‘Kinematic Car’ Mobile Robots

This section presents, in detail, the implementation of the result of the previous section to a fleet of mobile robots. The model used is called the kinematic car and is a simple model of wheeled vehicle with steering wheels. This model has been thoroughly studied and Murray et al. (1994) provide a good review. For a group of such robots, the equations of motion for the ith robot are given by

$$\begin{bmatrix}\dot x_i\\ \dot y_i\\ \dot\theta_i\\ \dot\phi_i\end{bmatrix} = \begin{bmatrix}\cos\theta_i\\ \sin\theta_i\\ \frac{1}{l_i}\tan\phi_i\\ 0\end{bmatrix}u_i^1 + \begin{bmatrix}0\\0\\0\\1\end{bmatrix}u_i^2,$$

where u_i^1 is the forward velocity of the drive wheels, u_i^2 is the angular velocity of the steering wheels and l_i is the wheelbase of the robot, i.e. the distance between the front and rear wheels and x_i, y_i and θ_i represent the planar position and orientation of the robot and ϕ_i is the angle of the steering wheels. More compactly, this is represented by

$$\dot x_i + g_i^1u_i^1 + g_i^2u_i^2,$$

where g_i^1 and g_i^2 are the vector fields in the previous equation. This model is controllable with

$$g_i^3 = [g_i^1, g_i^2] = \begin{bmatrix}0\\0\\-\frac{1}{l_i\cos^2\phi_i}\\0\end{bmatrix}$$

and

$$g_i^4 = [g_i^1, g_i^3] = [g_i^1,[g_i^1,g_i^2]] = \begin{bmatrix}-\frac{\sin\theta_i}{l_i\cos^2\phi_i}\\ \frac{\cos\theta_i}{l_i\cos^2\phi_i}\\ 0\\ 0\end{bmatrix}.$$

For the ith robot, the matrix C_i is given by

$$C_i\left(x_i, y_i, \theta_i, \phi_i\right) = \begin{bmatrix}\cos\theta_i & 0 & 0 & -\frac{\sin\theta_i}{l_i\cos^2\phi_i}\\ \sin\theta_i & 0 & 0 & \frac{\cos\theta_i}{l_i\cos^2\phi_i}\\ \frac{\tan\phi_i}{l_i} & 0 & -\frac{1}{l_i\cos^2\phi_i} & 0\\ 0 & 1 & 0 & 0\end{bmatrix},$$

and its inverse by

$$C_i^{-1} = \begin{bmatrix}\cos\theta_i & \sin\theta_i & 0 & 0\\ 0 & 0 & 0 & 1\\ \cos\phi_i\sin\phi_i\cos\theta_i & \cos\phi_i\sin\phi_i\sin\theta_i & -l_i\cos^2\phi_i & 0\\ -l_i\cos^2\phi_i\sin\theta_i & l_i\cos^2\phi_i\cos\theta_i & 0 & 0\end{bmatrix}.$$

Note that as long as $l_i \neq 0$, C_i is defined as is its inverse.

Now, for the ith robot, the fictitious inputs corresponding to a desired trajectory, $(x_i(t), y_i(t), \theta_i(t), \phi_i(t)) = \gamma_i(t)$ are given by

$$v_i(t) = C_i^{-1}\left(\gamma_i(t)\right)\dot\gamma_i(t)$$

and, skipping over a substantial amount of algebraic effort, the Chen–Fliess–Sussmann equations for the backward Philip Hall coordinates are given by

$$\begin{aligned}\dot h_i^1 &= v_i^1\\ \dot h_i^2 &= v_i^2\\ \dot h_i^3 &= h_i^1v_i^2 + v_i^3\\ \dot h_i^4 &= \frac{1}{2}h_i^1h_i^1v_i^2 + h_i^1v_i^3 + v_i^4,\end{aligned}$$

which may be written in vector form as

$$\begin{aligned}\begin{bmatrix}\dot h_i^1\\ \dot h_i^2\\ \dot h_i^3\\ \dot h_i^4\end{bmatrix} &= \begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & h_i^1 & 1 & 0\\ 0 & \frac{1}{2}h_i^1h_i^1 & h_i^1 & 1\end{bmatrix}\begin{bmatrix}v_i^1\\ v_i^2\\ v_i^3\\ v_i^4\end{bmatrix}\\ &= Q_i(h_i)v_i. \qquad (17)\end{aligned}$$

Solving these differential equations gives the backward Philip Hall coordinates. The conversion to forward coordinates, again skipping a substantial amount of algebra, is given by

$$\begin{aligned} \tilde{h}_i^1 &= h_i^1 \\ \tilde{h}_i^2 &= h_i^2 \\ \tilde{h}_i^3 &= h_i^3 - h_i^1 h_i^2 \\ \tilde{h}_i^4 &= h_i^4 - h_i^1 h_i^3. \end{aligned}$$

The forward Philip Hall coordinates are almost the 'final answer' in that they provide the amount of time the system must flow along g_i^1, g_i^2, g_i^3 and g_i^4 in a sequential manner. Since g_i^1 and g_i^2 are vector fields in the original system, this is accomplished simply by having $u_i^1 = 1$ for an amount of time equal to $\tilde{h}_i^1$ and $u_i^2 = 1$ for an amount of time equal to $\tilde{h}_i^2$. Since g_i^3 is a bracket, the flow along it is accomplished by a sequence of four flows of the form

$$\{u_i^1, u_i^2, -u_i^1, -u_i^2\}, \tag{18}$$

where each input is 'on' for a time equal to $\sqrt{\tilde{h}_i^3}$. This notation means that

- $u_i^1 = 1$ and $u_i^2 = 0$ for $t \in [0, \tilde{h}_i^3]$, followed by
- $u_i^2 = 1$ and $u_i^1 = 0$ for $t \in [\tilde{h}_i^3, 2\tilde{h}_i^3]$, followed by
- $u_i^1 = -1$ and $u_i^2 = 0$ for $t \in [2\tilde{h}_i^3, 3\tilde{h}_i^3]$ followed by
- $u_i^1 = 0$ and $u_i^2 = -1$ for $t \in [3\tilde{h}_i^3 4\tilde{h}_i^3]$,

which results in a flow along g_i^3 for a time equal to $\tilde{h}_i^3$ *to leading order.*

The sequence of inputs that results in a flow along g_i^4 and also compensates for the 3rd order error resulting from the approximation for g_i^3 is given by a sequence of 20 inputs, the first 10 of which are

$$\{u_i^1, u_i^1, u_i^2, -u_i^1, -u_i^2, -u_i^1, u_i^2, u_i^1, -u_i^2, -u_i^1\}, \tag{19}$$

where each input is used sequentially and for a time equal to

$$\tilde{h}_i^4 - \frac{1}{2}\left(\tilde{h}_i^3\right)^{\frac{3}{2}},$$

followed by the sequence of 10 more inputs

$$\{u_i^2, u_i^1, u_i^2, -u_i^1, -u_i^2, -u_i^2, u_i^2, u_i^1, -u_i^2, -u_i^1\}, \tag{20}$$

where each input is used sequentially and for a time equal to

$$\frac{1}{2}\left(\tilde{h}_i^3\right)^{\frac{3}{2}}.$$

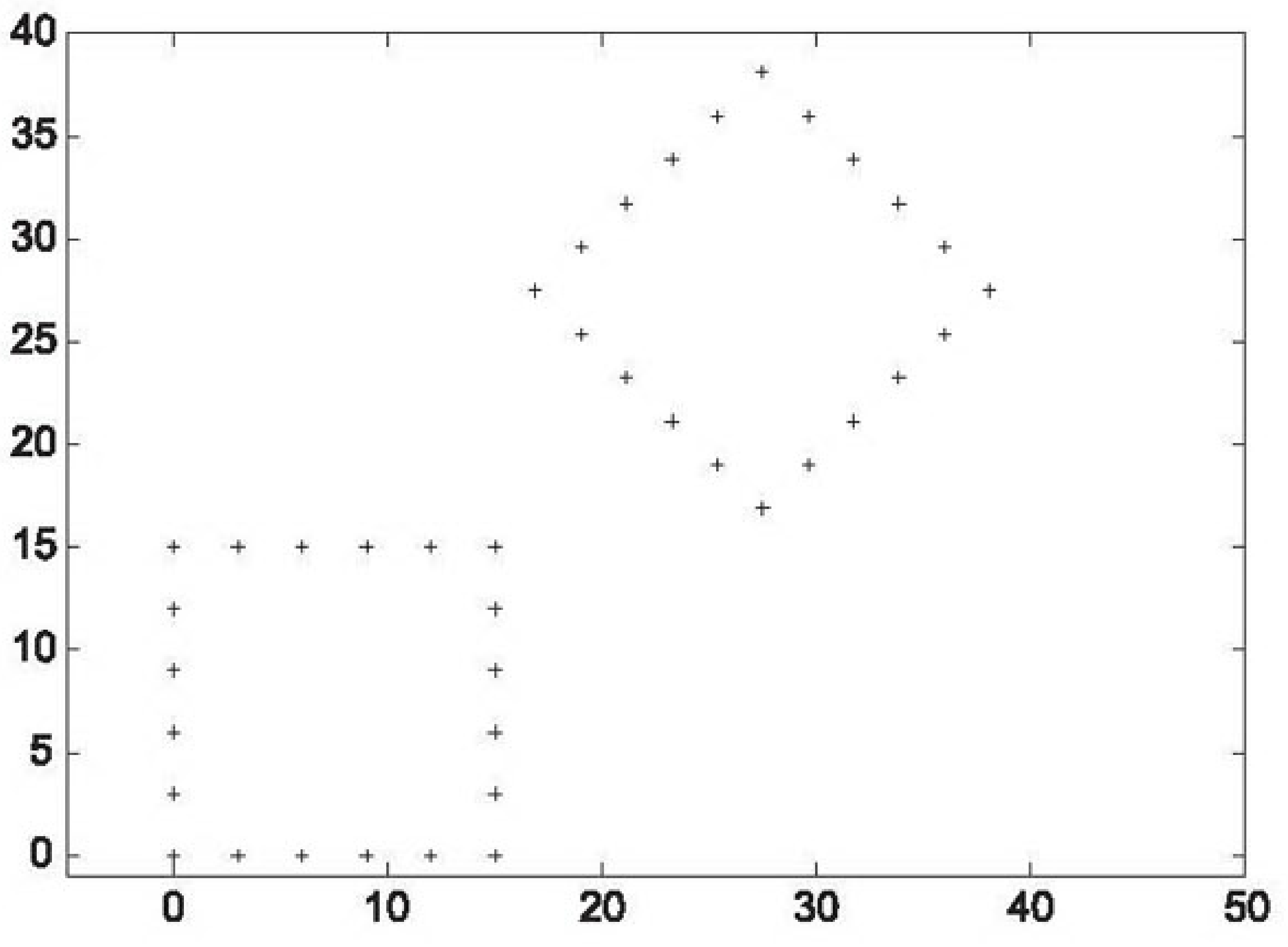

Fig. 2. Initial (lower left) and final (upper right) configurations for the fleet of kinematic cars.

The derivation of this relatively complicated sequence on inputs can be found in Lafferriere and Sussman (1993).

Having computed the sequence of inputs necessary for the ith robot, we will now make use of the symmetry properties of the system to reduce the amount of work necessary to compute the trajectories of the other robots. In particular, the work necessary to invert C_i is avoided for symmetric robots as well as all the algebraic manipulation to determine the differential equations for the backward Philip Hall coordinates and the algebraic effort to determine transformation to the forward Philip Hall coordinates. The sequence of inputs necessary to generate the motion are saved for other robots related to the first through the symmetry condition.

As an example, consider a formation of 20 robots of this type in the formation illustrated in Figure 2. The overall motion of the formation is a displacement in (x, y) by an amount $(20, 20)$ and a net rotation of $\frac{\pi}{4}$. We will assume that each robot starts with an orientation of $\theta_i = 0$ and at the terminal formation each robot has an orientation of $\theta_i = \frac{\pi}{2}$. Hence, while the formation rotates by $\frac{\pi}{4}$, each robot rotates an additional $\frac{\pi}{4}$ relative to the formation. To add further complexity to the problem to illustrate the utility of the method, we do not assume that the parameter l is the same for each robot. In particular, let $l_i = 0.5$ for robots 1 through 6 and 11 through 20 and let $l_i = 0.25$ for robots 7 through 10. This corresponds to the robots on the right side of the initial square formation having a different wheelbase than the rest of the robots.

The configuration space is comprised of the Cartesian product

$$M = \prod_{i=1}^{20} M_i = M_1 \times M_2 \times \cdots \times M_{20},$$

where $M_i = \mathbb{R}^2 \times S \times S$ is parameterized by $(x_i, y_i, \theta_i, \phi_i)$.

The equations of motion for the entire system are

$$\begin{bmatrix} x_1 \\ y_1 \\ \theta_1 \\ \phi_1 \\ x_2 \\ y_2 \\ \theta_2 \\ \phi_2 \\ \vdots \\ x_{20} \\ y_{20} \\ \theta_{20} \\ \phi_{20} \end{bmatrix} = \begin{bmatrix} \cos\theta_1 \\ \sin\theta_1 \\ \frac{1}{l_1}\tan\phi_1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} u_1^1 + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} u_1^2 + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \cos\theta_2 \\ \sin\theta_2 \\ \frac{1}{l_2}\tan\phi_i \\ 0 \\ \vdots \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} u_2^1 + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ \vdots \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} u_2^2 + \cdots + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ \cos\theta_{20} \\ \sin\theta_{20} \\ \frac{1}{l_{20}}\tan\phi_{20} \\ 0 \end{bmatrix} u_{20}^1 + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} u_{20}^2.$$

To consider the nature in which this system is symmetric, consider the permutation

$$\sigma \begin{pmatrix} 1 & 2 & \cdots & 20 \\ 2 & 1 & \cdots & 20 \end{pmatrix},$$

which interchanges 1 and 2 and leaves the rest of the elements fixed. The corresponding induced permutation on M is

$$\begin{aligned} &\sigma_\# \left(x_1, y_1, \theta_1, \phi_1, x_2, y_2, \theta_2, \phi_2, \ldots, x_{20}, y_{20}, \theta_{20}, \phi_{20}\right) \\ &\quad = \left(x_2, y_2, \theta_2, \phi_2, x_1, y_1, \theta_1, \phi_1, \ldots, x_{20}, y_{20}, \theta_{20}, \phi_{20}\right), \end{aligned}$$

which corresponds to interchanging components 1 and 2. To compute the pushforward, note that

$$T\sigma_\# = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \cdots & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ \vdots & & & & & & & \vdots & \ddots & \vdots & & & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 1 \end{bmatrix}$$

and hence,

$$\begin{aligned} &(\sigma_\#)_* \, g_1^1 \left(x_1, \ldots, \phi_{20}\right) \\ &= T\sigma_\# g_1^1 \left(x_2, y_2, \theta_2, \phi_2, x_1, y_1, \theta_1, \phi_1, \ldots, \phi_{20}\right) \\ &= T\sigma_\# \begin{bmatrix} \cos\theta_2 \\ \sin\theta_2 \\ \frac{1}{l_1}\tan\phi_2 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \cos\theta_2 \\ \sin\theta_2 \\ \frac{1}{l_1}\tan\phi_2 \\ 0 \\ \vdots \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \end{aligned}$$

which is equal to $g_2^1(x_1,\ y_1,\ \theta_1,\ \phi_1,\ x_2,\ y_2,\ \theta_2,\ \phi_2,\ \ldots,\ x_{20},\ y_{20},\ \theta_{20}\phi_{20})$ as long as $l_1 = l_2$. If $l_1 \neq l_2$, then by substituting the parameters for component 1 we have

$$\left((\sigma_\#)_* \, g_1^1 \left(x_1, \ldots, \phi_{20}\right)\right)\big|_{l_2} = g_2^1 \left(x_1, \ldots, \phi_{20}\right).$$

Hence, components 1 and 2 are symmetric. Clearly, all of the components are similarly symmetric under all possible permutations, and hence the symmetry orbit is the entire system.

Since the overall motion of the formation is large and the Lie-algebraic method is only approximate for systems that are not nilpotent, as is the case for the kinematic car, we must segment the overall motion into smaller trajectories. The next section presents a derivation of a bound for the net error and deviation from the nominal trajectory for these systems, but for now we will somewhat arbitrarily segment the trajectory into 50 subtrajectories. At the end of the execution of the motion plan for each segment, a new nominal trajectory is computed to the next goal point. This is necessary first to avoid the accumulation of errors due to the approximate nature of the method. Additionally, it provides an element of realism in that real systems will have substantial open loop drift from the desired motion and incorporating periodic updates on the actual position, while perhaps not continuously available, is straightforward to implement in such a periodic manner using this method.

We will compute the inputs necessary for robot 1 and then use the symmetry conditions to determine them for the other robots. In particular, for robot 1 we have

$$\begin{aligned} A_1(\omega t) &= \begin{bmatrix} \cos\frac{n\pi t}{200} & \sin\frac{n\pi t}{200} \\ \cos\frac{n\pi t}{200} & \sin\frac{n\pi t}{200} \end{bmatrix} \\ \hat{P} &= \begin{bmatrix} -7.5 \\ -7.5 \end{bmatrix} \quad \hat{q}(t) = \begin{bmatrix} \frac{2nt}{5} \\ \frac{2nt}{5} \end{bmatrix} \\ \hat{r}(t) &= \begin{bmatrix} 0 \\ 0 \\ \frac{n\pi t}{100} \end{bmatrix}, \end{aligned}$$

where n is the step number and $n \in \{1, \dots, 50\}$. Parameterizing a trajectory connecting the starting point at $t = 0$ to the final point for this segment at $t = 1$ with a straight line gives

$$\gamma_1(t) = \begin{bmatrix} x_1(t) \\ y_1(t) \\ \theta_1(t) \\ \phi_1(t) \end{bmatrix} = \begin{bmatrix} 0.55t \\ 0.337868t \\ 0.031415t \\ 0 \end{bmatrix},$$

assuming that the desired starting and ending configuration for the steering wheel is 0. Finally, the fictitious inputs are given by

$$\begin{aligned} v_1 &= C^{-1}\left(\gamma_1(t)\right)\dot{\gamma}_1(t) \\ &= \begin{bmatrix} \cos(0.031415t) & \sin(0.031415t) & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -l_1 & 0 \\ -l_1\sin 0.031415t & l_1\cos 0.031415t & 0 & 0 \end{bmatrix} \\ &\times \begin{bmatrix} 0.55 \\ 0.337868 \\ 0.031415 \\ 0 \end{bmatrix}. \end{aligned}$$

Substituting into $h_1 = Q_1(h_1)v_1$, where Q_1 is calculated from Equation 17, we have the backward coordinates

$$\begin{aligned} h_1^1 &= 0.55t \\ h_1^2 &= 0 \\ h_1^3 &= -0.0157079t \\ h_1^4 &= 0.1689339t - 0.0043196t^2 \end{aligned}$$

and hence the forward coordinates

$$\begin{aligned} \tilde{h}_1^1 &= 0.55t \\ \tilde{h}_1^2 &= 0 \\ \tilde{h}_1^3 &= -0.0157079t \\ \tilde{h}_1^4 &= 0.1689339t - 0.0043196t^2. \end{aligned}$$

Thus, to move to the desired end point at $t = 1$ of the first segment of this trajectory, the first robot must flow along g_1^1 for a time of 0.55, along g_1^2 for a time of 0, flow along g_1^3, which requires a four input sequence to approximate for a time of -0.0157079 (the negative time is easily accommodated by the method) and finally g_1^4 (requiring 20 input sequences) for a time of 0.164614.

For the second robot, we employ Theorem 3.1. First we must compute $\gamma_2(t)$, which is

$$\gamma_1(t) = \begin{bmatrix} x_2(t) \\ y_2(t) \\ \theta_2(t) \\ \phi_2(t) \end{bmatrix} = \begin{bmatrix} 3(1-t) + 3.53243t \\ 0.380294t \\ 0.031415t \\ 0 \end{bmatrix}.$$

Hence,

$$\dot{\gamma}_2 = \begin{bmatrix} 0.5324 \\ 0.380294 \\ 0.031415 \\ 0 \end{bmatrix}.$$

We want to compute

$$\dot{h}_2 = Q_1(h_2)C_1^{-1}\left((\sigma_\#)^{-1}\gamma_2(t)\right)\Big|_{l_2}\dot{\gamma}_2(t).$$

At this point we have $\gamma_2(t)$, $\dot{\gamma}_2(t)$ and $Q_1(h)$. The $\left((\sigma_\#)^{-1}\gamma_2(t)\right)$ term simply implies the trajectory is to be substituted for $\gamma_1(t)$ into C_1^{-1}. Hence we have

$$\begin{bmatrix} \dot{h}_2^1 \\ \dot{h}_2^2 \\ \dot{h}_2^3 \\ \dot{h}_2^4 \end{bmatrix}_i = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & h_i^1 & 1 & 0 \\ 0 & \frac{1}{2}h_i^1h_i^1 & h_i^1 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta_2 & \sin\theta_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ \cos\phi_2\sin\phi_2\cos\theta_2 & \cos\phi_2\sin\phi_2\sin\theta_2 & -l_2\cos^2\phi_2 & 0 \\ -l_2\cos^2\phi_2\sin\theta_2 & l_2\cos^2\phi_2\cos\theta_2 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0.5324 \\ 0.380294 \\ 0.031415 \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & h_i^1 & 1 & 0 \\ 0 & \frac{1}{2}h_i^1h_i^1 & h_i^1 & 1 \end{bmatrix} \begin{bmatrix} \cos(0.031415t) & \sin(0.031415t) & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -l_2 & 0 \\ -l_2\sin(0.031415t) & l_2\cos(0.031415t) & 0 & 0 \end{bmatrix} \begin{bmatrix} 0.5324 \\ 0.380294 \\ 0.031415 \\ 0 \end{bmatrix}.$$

It is emphasized that making this simple substitution into C_1^{-1} and also using the same form for $Q_i(h)$ for each symmetric component is the manifestation of the main computational savings from the symmetry analysis.

Solving these and converting to forward Philip Hall coordinates finally gives

$$\tilde{h}_1^1 = 0.532426t$$
$$\tilde{h}_1^2 = 0$$
$$\tilde{h}_1^3 = -0.0157079t$$
$$\tilde{h}_1^4 = 0.190147t - 0.00418167t^2.$$

It is worth noting that these values are close to, but not exactly the same as, those for robot 1, which is to be expected since they are adjacent robots in the formation. Evaluating these at $t = 1$ gives the times to flow along each vector field g_2^1, g_2^2, g_2^3 and g_2^4. The same sequence approximations may be used as were done in Equations 18, 19 and 20.

Repeating this exercise for all 20 robots and for each segment of the desired trajectory gives the final motion, which is illustrated in Figure 3.

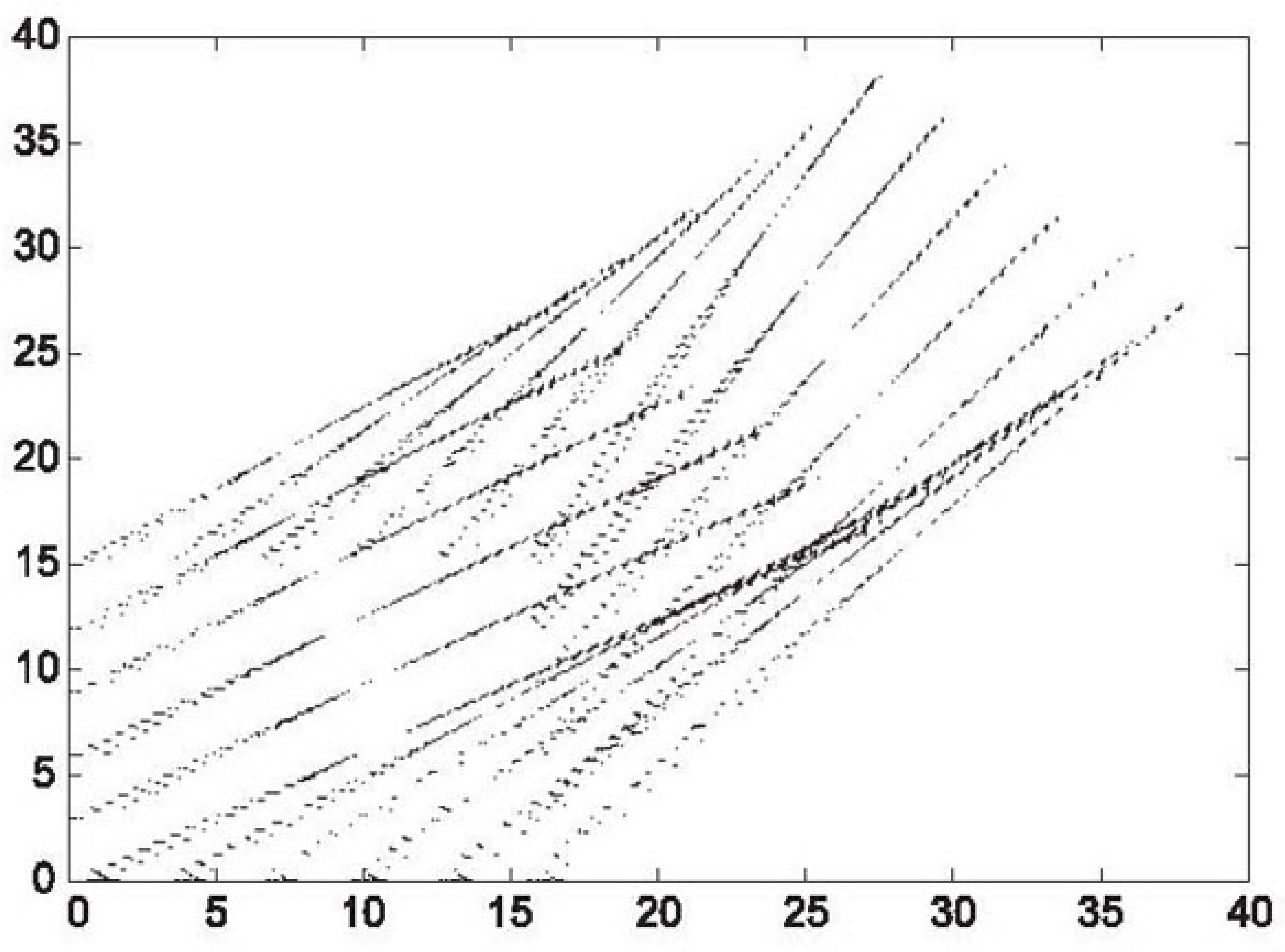

Fig. 3. Overall motion for a system of 20 kinematic car robots.

5. Collision Avoidance

Since the motion of the robots is not generally along the nominal trajectory, it is necessary to bound the deviation from the nominal trajectory in order to guarantee collision avoidance. Our approach will be to show that the deviation from the nominal trajectory is bounded by a term that is a function of the length of the nominal trajectory as well as the order of the system. Hence, it will be possible to guarantee collision avoidance if the nominal trajectories do not collide, i.e. intersect at a given time, by segmenting the overall trajectory into smaller subtrajectories that can be sequentially followed.

The overall approach is to decompose the complete trajectory into subtrajectories that are small enough to ensure there is no collision in the system. Since we are considering small motions, we will consider the system locally in $\mathbb{R}^n$. For the trajectory γ, given in Equation 12 for $t \in [0, T]$, let $\mathcal{R}_i = \min_{t\in[0,T]} \|\gamma_i(t) - \gamma_j(t)\|$, such that $i \neq j$, i.e. the closest any robot gets to robot i while following the trajectory. Also, let $\Delta_i = \|\gamma_i(T) - \gamma_i(0)\|$. Consider a linear trajectory $\Gamma_i(t) = \gamma_i(0) + t(\gamma_i(T) - \gamma_i(0))$ connecting the initial position to the final position. Recall, the fictitious inputs are calculated by solving $\dot{\gamma}_i(t) = [g_1(\gamma_i(t)), \dots, g_s(\gamma_i(t))]v$. Applying this to the linear trajectory $\Gamma_i(t)$ we find

$$\|\dot{\Gamma}_i\| = \|\gamma_i(T) - \gamma_i(0)\| < \|[g_i^1(\gamma_i(t)), \dots, g_i^s(\gamma_i(t))]\| \|v_i\|.$$

From this equation, we find that the fictitious inputs are bounded by a constant i_i, i.e. $\|v_i\| < \zeta_i\|\dot{\Gamma}_i\| = \zeta_i\Delta_i$. By construction of the real inputs from the fictitious inputs, $\|u\| < \zeta_i\Delta^{1/k}$ where k is the order of the highest Lie bracket needed to make $\tilde{C}$ full rank. Let $x_{i,\max} = \max_{t\in[0,T]} \|x_i(t) - \gamma_i(0)\|$ denote the flow that is maximally distant from the starting point. Note that this is not necessarily $\gamma_i(T)$. Pick a ball $\mathcal{B}_i$ of radius $\mathcal{R}_i$ centered at the initial point. Let η_i be the maximum norm of all the first order vector fields for all points in the ball $\mathcal{B}_i$. The distance $\|x_{i,\max} - \gamma_i(0)\|$ is necessarily bounded by the sum of the norms of each individual flow associated with one real control input u_i^{lj}. That is,

$$\|x_{i,\max} - \gamma(0)\| \leq \sum_l \sum_j \| \int_0^1 g_i^l u_i^{lj} dt \|.$$

We know that $\|u_i^l\| \leq \eta_i \Delta_i^{1/k}$ and $\|g_i^l(x)\| \leq \zeta_i$ for all $x \in \mathcal{B}_i$. Therefore

$$\|x_{i,\max} - \gamma_i(0)\| \leq \sum_l \sum_j \eta_i \zeta_i \Delta^{1/k}$$

and since $\Delta_i = \|\gamma_i(T) - \gamma_i(0)\|$, by choosing the desired final point close enough to the starting point, the robots will not collide. Because Δ_i is raised to the power of $1/k$, if k is large, then Δ_i may be exceedingly small. This approach is very conservative and it is best to identify the appropriate step length experimentally.

6. Experimental Implementation

We implemented this motion planning algorithm on a system of four MICAbot mobile robots. A detailed description of this experimental platform can be found in McMickell et al. (2003). We first present a sketch of the model used and associated simulation results. Then we present the result of the algorithm implemented on the actual robots.

6.1. Symmetry Analysis for Hilare-type Mobile Robots

The kinematics of the MICAbots are the same as that of Hilare-type mobile robots. For a formation of four Hilare-type robots, the kinematics of the ith robot are described by Sastry (1999),

$$\begin{bmatrix} \dot{x}_i \\ \dot{y}_i \\ \dot{\theta}_i \end{bmatrix} = \begin{bmatrix} \cos\theta_i \\ \sin\theta_i \\ 0 \end{bmatrix} u_i^1 + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u_i^2, \qquad (21)$$

where u_1 is the linear velocity input and u_2 is the angular velocity input. In the model as well as experiment, it is assumed that all robots are identically parameterized. This model is particularly useful since there exists a transformation that renders the system nilpotent, in which case the method of Lafferriere and Sussman (1993) is exact.

In particular, using inputs,

$$\begin{aligned} u_i^1 &= \frac{1}{\cos(\theta_i)} w_i^1 \qquad (22) \\ u_i^2 &= \cos^2(\theta_i) w_i^2, \end{aligned}$$

the system becomes

$$\begin{bmatrix} \dot{x}_i \\ \dot{y}_i \\ \dot{\theta}_i \end{bmatrix} = \begin{bmatrix} 1 \\ \tan\theta_i \\ 0 \end{bmatrix} w_i^1 + \begin{bmatrix} 0 \\ 0 \\ \cos^2\theta_i \end{bmatrix} w_i^2,$$

which is nilpotent of order 2. For this system, the extended system is given by

$$\begin{bmatrix} \dot{x}_i \\ \dot{y}_i \\ \dot{\theta}_i \end{bmatrix} = \begin{bmatrix} 1 \\ \tan\theta_i \\ 0 \end{bmatrix} v_i^1 + \begin{bmatrix} 0 \\ 0 \\ \cos^2\theta_i \end{bmatrix} v_i^2 + \begin{bmatrix} 0 \\ \cos^4\theta_i \\ 0 \end{bmatrix} v_i^3.$$

Hence,

$$C_i\left(\gamma_i(t)\right) = \begin{bmatrix} 1 & 0 & 0 \\ \tan\theta_i & 0 & \cos^4\theta_i \\ 0 & \cos^2\theta_i & 0 \end{bmatrix}.$$

Differentiating the Chen–Fliess series, equating it to the formal extended system and substituting for the fictitious inputs results in the ordinary differential equation,

$$\begin{bmatrix} \dot{h}_i^1 \\ \dot{h}_i^2 \\ \dot{h}_i^3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & h_i^1 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 \\ \tan\theta_i & 0 & \cos^4\theta_i \\ 0 & \cos^2\theta_i & 0 \end{bmatrix}^{-1} \dot{\gamma}_i(t).$$

Due to the identical and simple kinematics of each robot, using the result from Theorem 3.1 is simply a matter of computing the nominal trajectory for each robot and substituting it for $\dot{\gamma}_i(t)$ and in the components of C_i^{-1}.

We assume that the robots are initially in a square formation centered about the origin a distance of unity apart. The robots are to follow a linear path $q(t) = [t, t, 0]^T$ for a time $t \in [0, 1]$ with the orientation of the square rotating by an angle π. The step size for this example was computed experimentally. It was determined that four steps would be necessary to ensure collision-free movement. The Philip Hall coordinates for robot 1 were computed. They were then used by the other robots to compute their Philip Hall coordinates.

Figure 4 displays a simulation of the four robots. Figure 4(a) displays the robots' initial and final positions shown as 'o' and 'x', respectively. Figures 4(b)–(e) display the motion during each of four subtrajectories necessary to move to

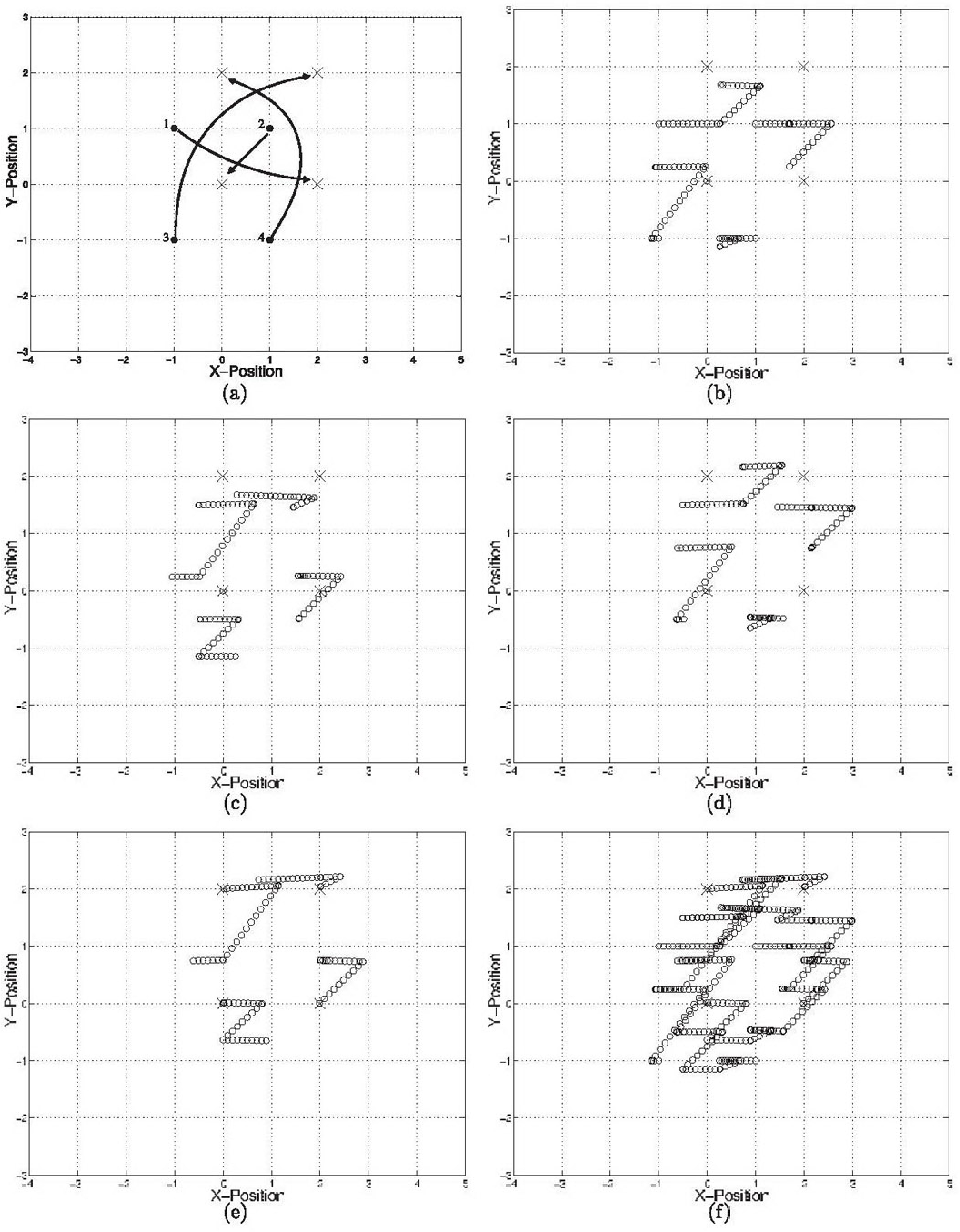

Fig. 4. (a) Initial and final configurations shown as ‘o’ and ‘x’, respectively; (b)–(e) display the motion plan in four steps to avoid collisions; (f) displays the combined motion plan of all four steps.

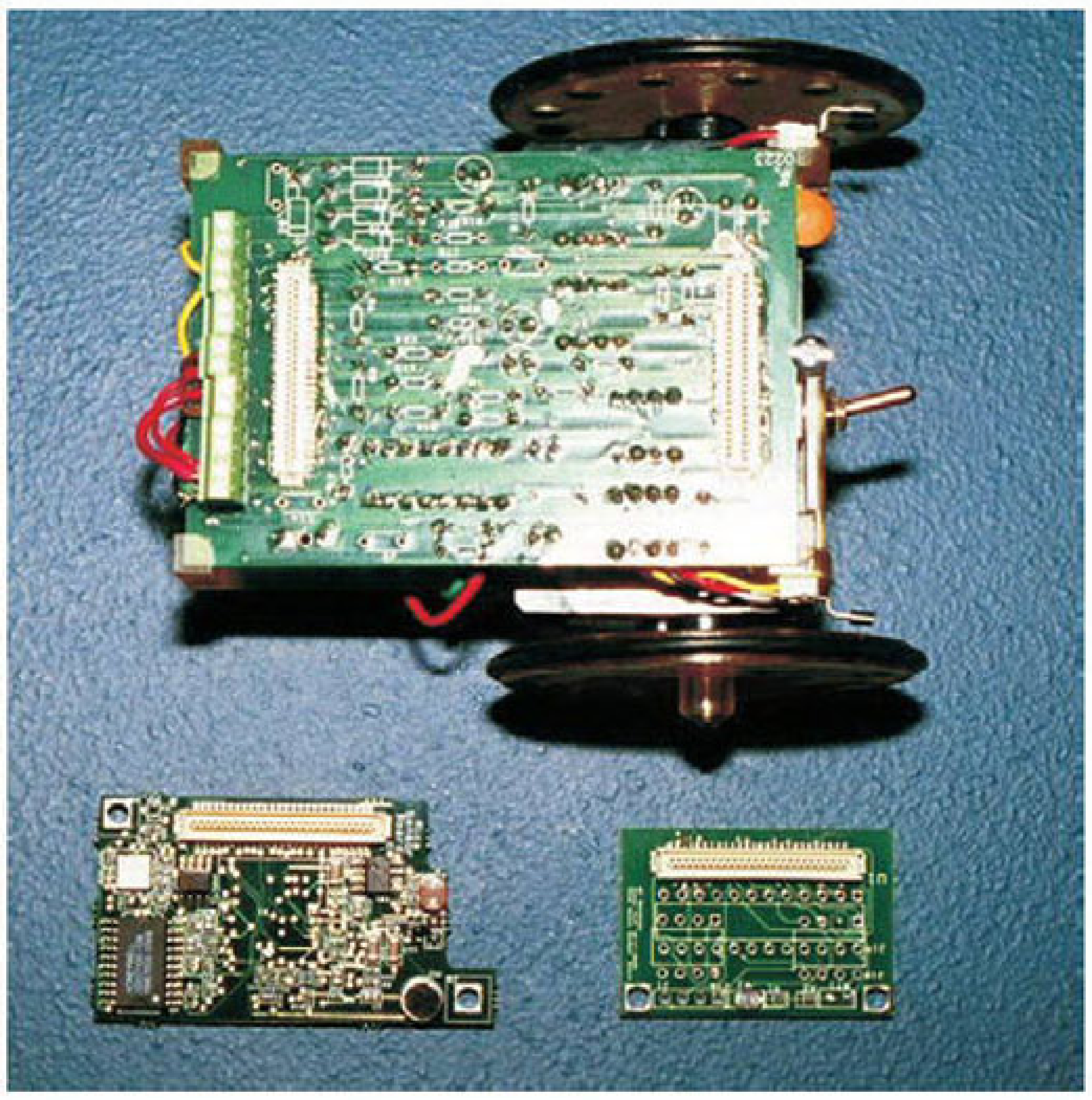

Fig. 5. The MICAbot interface board with sensor boards.

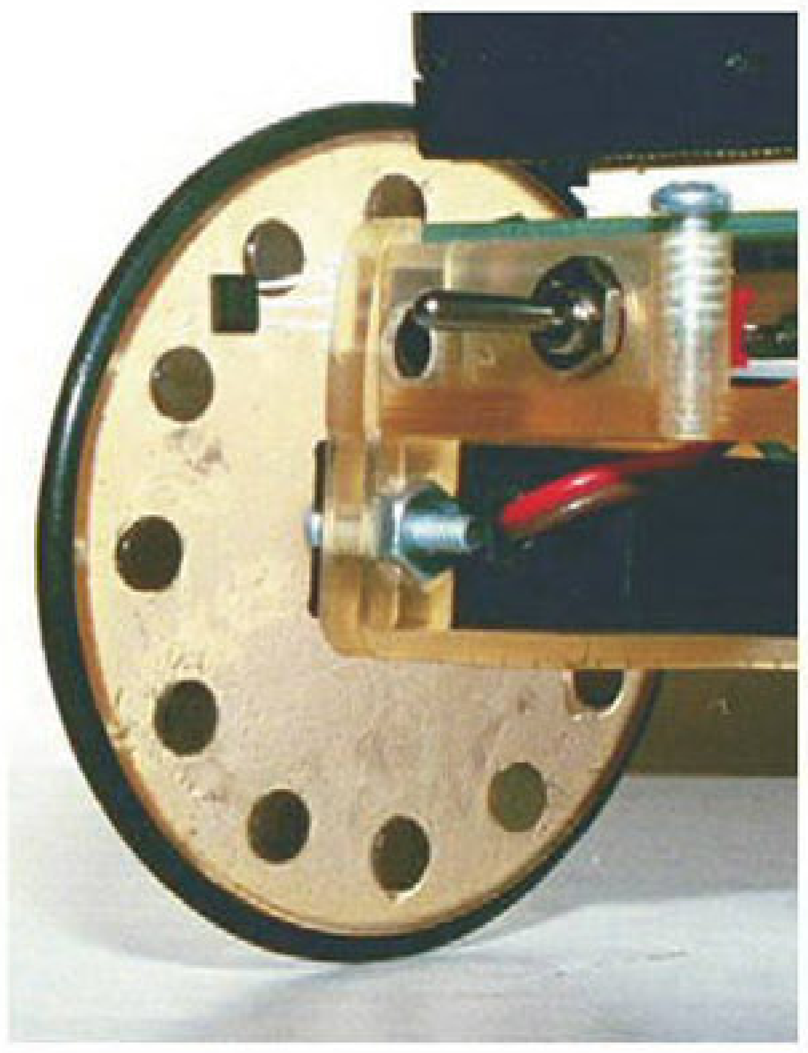

Fig. 6. Magnets and Hall-effect sensor used for odometry.

the final position without a collision. The final position and overall motion of the robots is shown in Figure 4(f).

One of the MICAbots used for the experimental implementation is illustrated in Figures 5 and 6. The body of the MICAbot is one single unit and supports the interface board, battery pack and motors. The body is constructed out of a lightweight polymer using a stereolithography. The wheels of the MICAbot are also made from the same material. The MICAbot is a two-wheeled robot, where each wheel is 3.3 cm in diameter. Each wheel has 12 evenly spaced magnets embedded in its interior face (see Figure 6). MICAbots are actuated using two modified submicro servo-motors that provide 25 oz-inches of torque. These modified servo-motors function as direct drive DC gear-headed motors that drive the MICAbot at a maximum velocity of 30 cm/s. Each motor is controlled by a pulse-width-modulated (PWM) signal, which allows us to control the angular direction and velocity of each wheel.

The MICAbot uses the MICA platform for its central processing and communication. Its central processor is an ATMEGA103L running at 4 MHz. This microcontroller has 128 kB of memory and 4 kB of RAM (Hill and Culler 2002). We use two PWM channels provided by the microcontroller, PWM1A and PWM1B, to control the DC motors. The PWM frequency is 4 kHz with 2048 steps between the maximum positive and negative voltages. Additionally, it has an AT90LS2343 flash-based microcontroller which can be used for wireless reprogramming.

An operating system has been developed for this platform called TinyOS (version 0.6) (Hill et al. 2000; Hill and Culler 2002; Mainwaring et al. 2002). The TinyOS operating system is micro-threaded and designed specifically for embed-

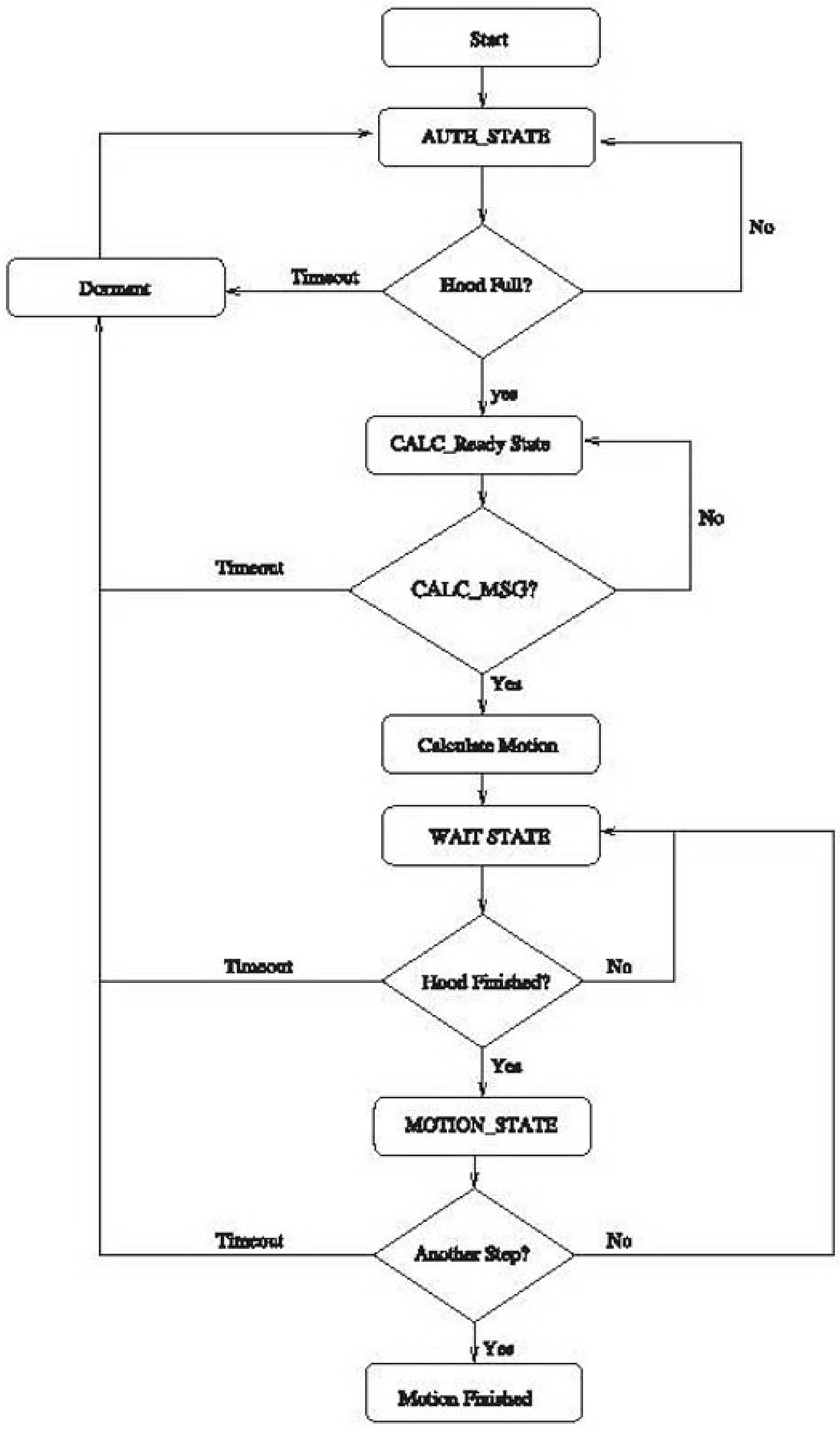

Fig. 7. Flowchart describing the motion planning program.

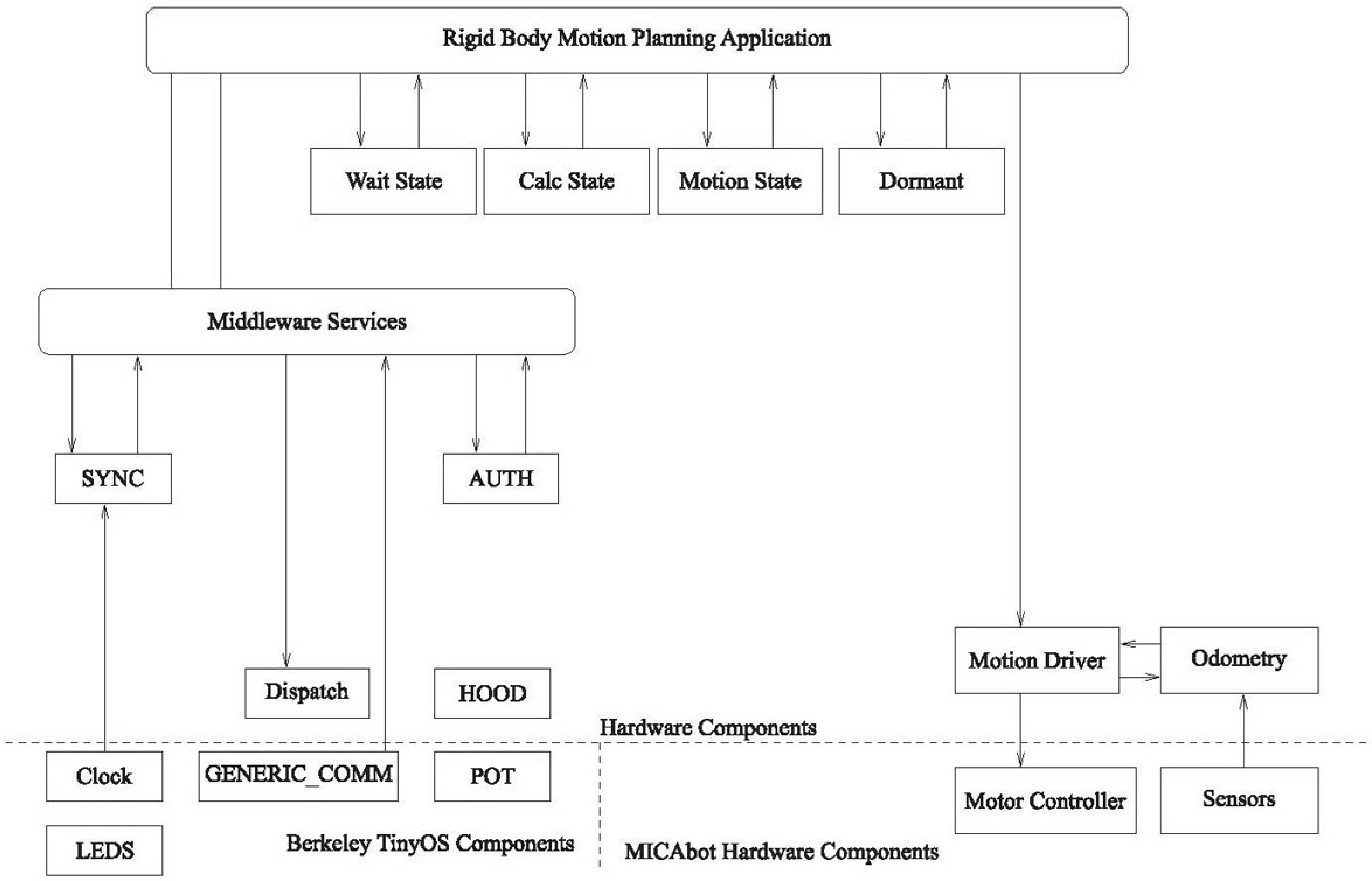

Fig. 8. Schematic showing the structure of the motion planning program.

ded processors. It provides an object-oriented approach to embedded programming and reusable encapsulated software for controlling the low-level hardware of the system. TinyOS has the essential attributes needed for real-time embedded control, making it ideal for the MICAbots.

Communication is accomplished via an RF Monolithic TR1000 transceiver at rates of up to 115 kB (Hill et al. 2000; Hill and Culler 2002). The MICA platform also has an external UART and SPI port. A DS2401 silicon serial number provides each MICA platform with a unique identification number (Hill and Culler 2002).

The radio board was designed to enhance the networking capabilities of the existing MICAbot. An ATMEGA103L microprocessor is used as the main processor on the radio board, which is the same processor as the main processor on the MICA. A separate radio transmitter and receiver are used to send and receive packets. The transmitter module is a TXM-900-HP-II by Linux Technologies and the receiver module is a RXM-900-HP-II. They operate using frequency modulation in the 902-928 MHz band and can establish a radio link for different channels or frequencies for added flexibility. Data received and transmitted through the radio board is handled by the UART and a time interrupt. Advantages of using the UART are that it synchronizes itself to incoming data for each byte transmitted and it includes a start and stop bit for switching of the data transmission in order to keep the receiver circuits well biased. Furthermore, all these operations, including error detection, are handled by hardware freeing the processor for other tasks.

The primary software structure used for motion planning and communication between the MICAbots is a finite state machine. Each state is constructed with one initiating command and two exit events, a timeout event and a complete event. Timeouts are to prevent deadlock and usually reset the program. State events are sent to the finite state machine and are used to determine which states are active. Figure 7 presents the flowchart describing the program. Figure 8 displays a schematic of the structure of the program. The program's structure can be viewed as a hierarchy of subprograms. On the lowest level are programs directly controlling the hardware. The next level are timers, the motion driver and the odometry controller which interact directly with the low-level hardware components. On the highest level is the motion planning application.

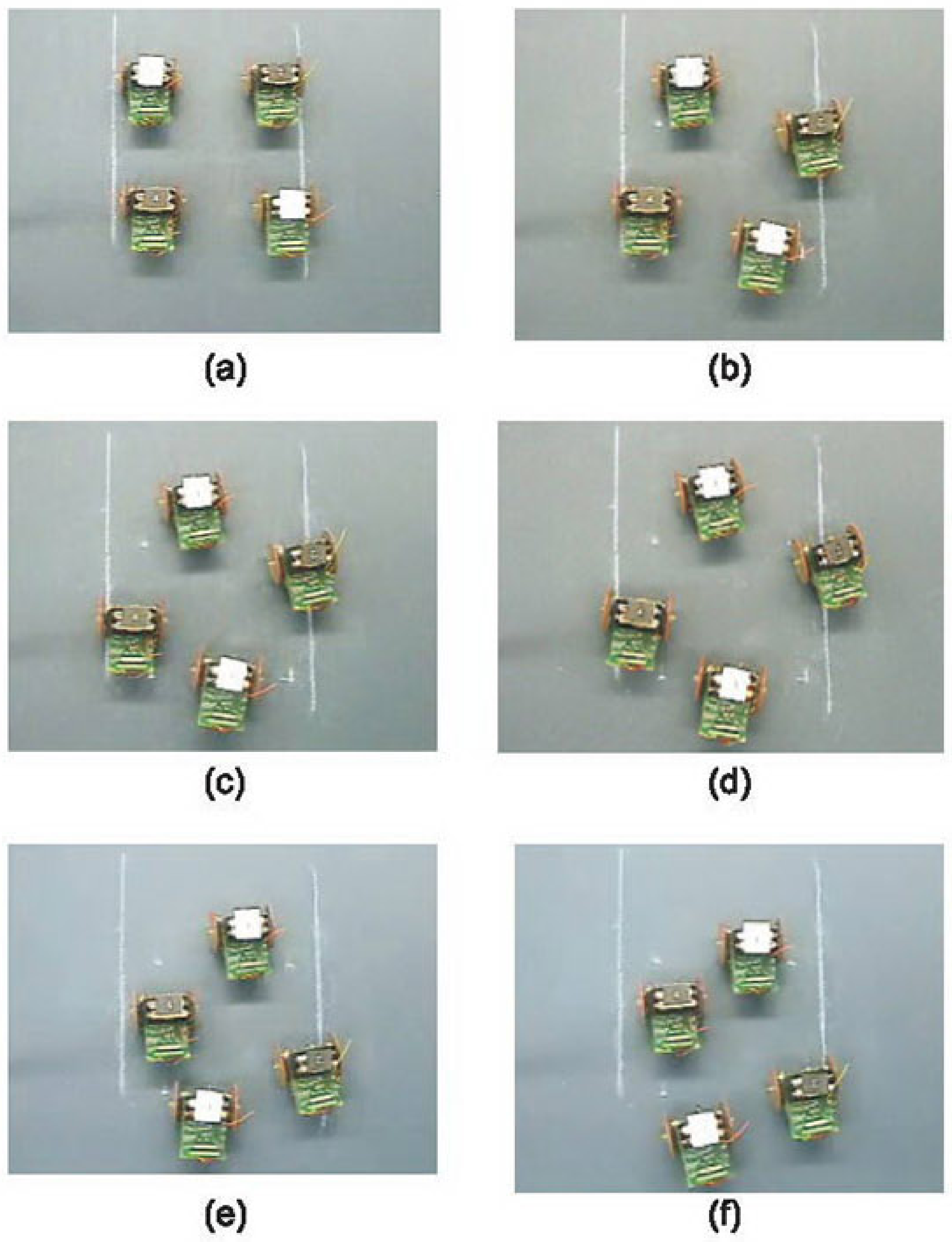

Fig. 9. (a) The starting formation of four MICAbots; (b)–(f) the formation after 2, 4, 6, 10 and 15 steps, respectively.

The implementation of the piecewise constant motion planning algorithm was tested on formations consisting of 2, 3 and 4 robots. Figures 9(a)–(f) display a rotation of a tight square formation by 90 degrees with no translation. MICAbots are placed in each corner of the square approximately 7.5″ apart. The four MICAbots were programmed with their initial position relative to the center of the formation in a global coordinate frame. This example demonstrates that the motion planning algorithm works; however, there is significant drift as the number of steps increases. This drift can mainly be attributed to the low resolution of our position sensor (the widely spaced Hall-effect sensors on the wheels). We would expect significant improvement by using high-resolution encoders.

7. Conclusions and Future Work

In this paper, we have presented a distributed motion planning algorithm for rigid body formations consisting of a large number of robots. The motion planning algorithm developed here is an extension of the piecewise constant motion planning algorithm by Lafferriere and Sussman (1993) that exploited symmetries in the system to reduce the computational burden for motion planning. In particular, for a symmetric system, the most computationally intense part of the motion planning algorithm must be executed only once for a set of symmetric robots.

The formation control algorithm was simulated and experimentally verified. A simulation of a group of mobile robots was used to demonstrate the utility of this algorithm. A novel centimeter-scale robot designed for research in large-scale distributed robotics and mobile *ad hoc* sensor networks was presented. MICAbots are inexpensive, and provide enough flexibility for a wide range of experimental goals. The MICAbot has an additional radio board, which increases its networking capabilities and provides additional I/O ports. These robots were used to experimentally test the motion planning algorithm. Experimental results show that the formation control algorithm can be used to control the formation of a mobile wireless network.

Future plans include using position and orientation states in the shared memory model in order to create a better method for collision avoidance. Future work on theoretical issues include investigating the optimal control problem for symmetric distributed systems.

Acknowledgments

The partial support of the National Science Foundation grants ISS99-84107-CAREER, CCR01-13131 and the Center for Applied Mathematics at the University of Notre Dame is gratefully acknowledged. The many helpful comments from two anonymous reviewers are appreciated.

References

Bates, L. and Sniatycki, J. (1993). Nonholonomic Reduction. *Reports on Mathematical Physics*, **32**: 99–115.

Chaimowicz, L., Sugar, T., Kumar, V. and Campos, M. F. M. (2001). An Architecture for Tightly Coupled Multi-Robot Cooperation. *Proceedings of the 2001 IEEE International Conference on Robotics and Automation*, **3**: 2992–2997.

Chitta, S. and Ostrowski, J. P. (2002). Motion Planning for Heterogeneous Modular Mobile Systems. *Proceedings of the 2002 IEEE International Conference On Robotics and Automation*, **4**: 4077–4082.

Desai, J. P. and Kumar, V. (1999). Motion planning for cooperating mobile manipulators. *Journal of Robotic Systems*, **10**: 557–579.

Egerstedt, M. and Hu, X. (2000). Coordinated Trajectory Following for Mobile Manipulation. *Proceedings of the 2000 IEEE International Conference on Robotics and Automation*, **4**: 3479–3484.

Fax, J. A. and Murrray, R. M. (2004). Information Flow and Cooperative Control of Vehicle Formations. *IEEE Transactions on Automatic Control*, **49**(49): 1465–1476.

Hill, J. and Culler, D. (2002). MICA: A wireless Platform for Deeply Embedded Networks. *IEEE Micro*, **22**(6): 12–24.

Hill, J., Szewczyk, R., Woo, A., Hollar, S., Culler, D. and Pister, K. (2000). System architecture directions for networked sensors. http://www.cs.berkeley.edu/~culler/cs252-s02/papers/MICA_ARCH.pdf.

Kawski, M. (2004). Bases for Lie algebras and a continuous CBH formula. In *Unsolved Problems in Mathematical Systems and Control Theory*. V. Blondel and A. Megretski (eds). Princeton University Press, 97–102.

Lafferriere, G. and Sussmann, H. J. (1993). A Differential Geometric Approach to Motion Planning. In *Nonholonomic Motion Planning*. X. Li and J. F. Canny (eds). Kluwer, 235–270.

Laumond, J.-P., Sekhavat, S. and Lamiraux, F. (1998). Guideline in Nonholonomic Motion Planning for Mobile Robots. In *Robot Motion Planning and control*. J.-P. Laumond (ed). Springer-Verlag.

Leonard, N. E. and Fiorelli, E. (2001). Virtual Leaders, artificial potentials, and coordinated control of groups. *Proceedings of the 2001 IEEE Conference on Decision and Control*, **3**: 2968–2973.

Mainwaring, A., Culler, D., Polastre, J., Szewczyk., R. and Anderson, J. (2002). Wireless sensor networks for habitat monitoring. *Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications*. 88–97.

McMickell, M. B. (2003). *Reduction and Control of Nonlinear Symmetric Distributed Robotic Systems*. Ph.D. thesis, University of Notre Dame.

McMickell, M. B. and Goodwine, B. (2001). Reduction and Nonlinear Controllability of Symmetric Distributed Systems. *Proceedings of the 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems*, **3**: 1232–1237.

McMickell, M. B. and Goodwine, B. (2002). Reduction and Nonlinear Controllability of Symmetric Distributed Robotic Systems with Drift. *Poceedings of the 2002 IEEE International Conference on Robotics and Automation*, **4**: 3454–3460.

McMickell, M. B. and Goodwine, B. (2003a). Reduced Order Motion Planning for Nonlinear Symmetric Distributed Robotic Systems. *Proceedings of the 2003 IEEE International Conference on Robotics and Automation*, **3**: 4228–4233.

McMickell, M. B. and Goodwine, B. (2003b). Reduction and nonlinear controllability of symmetric distributed systems. *International Journal of Control*, **76**(18): 1809–1822.

McMickell, M. B., Goodwine, B. and Montestruque, L. A. (2003). MICAbot: A Robotic Platform for Large-Scale Distributed Robotics. *Proceedings of the 2003 IEEE International Conference on Robotics and Automation*, **2**, 1600–1605.

Murray, R. M. and Sastry, S. (1993). Nonholonomic Motion Planning: Steering Using Sinusoids. *IEEE Transactions on Automatic Control*, **38**(5): 700–716.

Murray, R. M., Li, Z. and Sastry, S. S. (1994). *A Mathematical Introduction to Robotic Manipulation*. CRC Press, Inc.

Olfati-Saber, R. and Murray, R. M. (2002). Graph rigidity and distributed formation stabilization of multi-vehicle systems. *Proceedings of the 2002 IEEE Conference on Decision and Control*, **3**: 2965–2971.

Sastry, S. (1999). *Nonlinear Systems: Analysis, Stability, and Control*. Springer.

Souères, P. and Boissonat, J. D. (1998). Optimal Trajectories for Nonholonomic Mobile Robots. In *Robot Motion Planning and Control*, J. P. Laumond (ed). *Lecture Notes in Control and Information Sciences*, **229**, Springer-Verlag.

Sugar, T., Desai, J. P., Kumar, V. and Ostrowski, J. P. (2001). Coordination of Multiple Mobile Manipulators. *Proceedings of the 2001 IEEE International Conference on Robotics and Automation*, **3**: 3022–3027.

Sussmann, H. J. (1991). Two New Methods for Motion Planning for Controllable Systems without Drift. *European Control Conference*, 1501–1506.

Teel, A. R., Murray, R. M. and Walsh, G. C. (1995). Nonholonomic Control Systems: from steering to stabilization with sinusoids. *International Journal of Control*, **62**(4): 849–870.

Yamaguchi, H. (1999). A cooperative hunting behavior by mobile robot troops. *International Journal of Robotics Research*, **18**(9): 931–940.

Yamaguchi, H. and Burdick, J. (1998). Time-Varying Feedback Control for Nonholonomic Mobile Robots Forming Group Formations. *Proceedings of the 1998 IEEE Conference on Decision and Control*, 4156–4163.

Yamaguchi, H., Arai, T. and Beni, G. (2001). A distributed control scheme for multiple robotic vehicles to make group formations. *Robotics and Autonomous Systems*, **36**: 125–147.