Path Planning for Permutation-Invariant Multirobot Formations

Stephen Kloder, *Student Member, IEEE*, and Seth Hutchinson, *Senior Member, IEEE*

***Abstract*—In many multirobot applications, the specific assignment of goal configurations to robots is less important than the overall behavior of the robot formation. In such cases, it is convenient to define a permutation-invariant multirobot formation as a set of robot configurations, without assigning specific configurations to specific robots. For the case of robots that translate in the plane, we can represent such a formation by the coefficients of a complex polynomial whose roots represent the robot configurations. Since these coefficients are invariant with respect to permutation of the roots of the polynomial, they provide an effective representation for permutation-invariant formations. In this paper, we extend this idea to build a full representation of a permutation-invariant formation space. We describe the properties of the representation, and show how it can be used to construct collision-free paths for permutation-invariant formations.**

***Index Terms*—Configuration spaces, mobile robots, multirobot systems, path planning.**

I. Introduction

In this paper, we address the problem of planning collision-free paths for permutation-invariant multirobot formations. Such formations have the property that there is no specific *a priori* assignment of robots to individual tasks. Applications of permutation-invariant multirobot formations include tasks such as localization and exploration [1], surveillance and monitoring [2], search and rescue [3], and object manipulation and transportation [4]–[6].

In this paper we describe a new representation for the configuration space of permutation-invariant multirobot formations, for the specific case of formations of robots that translate in the plane. We refer to this space simply as the *formation space*. In this paper, we are concerned not with the maintaining of formations or applying transformations over them, but with creating and changing them. We represent a specific formation by the coefficients of a complex polynomial whose roots correspond to the unassigned configurations for the robots in the formation. These coefficients are invariant with respect to permutation of the roots, and thus the representation of a formation is invariant with respect to the assignment of specific configurations to specific robots.

Manuscript received December 14, 2004. This paper was recommended for publication by Associate Editor D. Fox and Editor K. Lynch upon evaluation of the reviewers' comments. This work was supported in part by the National Science Foundation under Award CCR-0085917 and IIS-0083275. This paper was presented in part at the IEEE International Conference on Robotics and Automation, New Orleans, LA, April 2004, and in part at the IEEE International Conference on Robotics and Automation, Barcelona, Spain, April 2005.

The authors are with the Beckman Institute of Advanced Science and Technology, University of Illinois, Urbana, IL 61801 USA (e-mail: kloder@uiuc.edu; seth@uiuc.edu).

Digital Object Identifier 10.1109/TRO.2006.878952

We use our formation space to plan the collision-free paths for a collection of robots from their specified initial configurations to a set of goal configurations—the goal configurations are not preassigned to specific robots. Collision detection is performed efficiently by exploiting the invariance of paths in the formation space with respect to translation, rotation and scaling operations. Paths are constructed using a simple probabilistic roadmap (PRM) planner whose local planning algorithm is merely a straight-line planner in the formation space. The path returned by our algorithm is such that each robot terminates its motion in one of the configurations from the set of goal configurations. Thus, each robot determines its eventual goal by traveling there, not by being assigned it.

Existing methods of formation planning are distinguished by their level of centralization. In fully centralized methods [7]–[15], a single planner decides the paths of all robots. In these methods, generally each robot has a predefined role, starting position, and goal position. Less centralized are emergent behavior methods [6], [16]–[24], which plan motions by specifying the relationships between the robots. The robots move toward the goal while maintaining these relationships. Most emergent behavior methods also prespecify individual robot roles, but some, like [25], use only role-independent relationships. Fully decentralized methods are mostly coverage methods [26]–[41]. In fully decentralized methods, robots do not have individual identities or assignments. Rather, the robots decide their motions based on their current locations, and the locations of whatever other robots are nearby. Generally, there is no specific goal configuration; instead robots move until certain local constraints are met.

Our method uses a single representation for the entire formation. This resembles centralized methods, like [7]–[9], that use an ordered list of robot configurations as a formation configuration. This has the advantage that any formation can be represented precisely, so it can be used as a goal formation. Other centralized and emergent behaviors have this advantage: [11]–[15] use separate paths for each robot, and [6], [17]–[21], [23], and [24] use relative positions of the robots to define formations. This is generally not true for less centralized methods; in general, these methods define their goal formations implicitly through constraints. For example, [22], [25], and [33] use a potential field to determine if the formation is at a goal. Similarly, [30]–[32] use a Voronoi diagram to test for a goal formation. This is a potential limitation; describing formations implicitly limits the specific types of formations that can be represented, e.g., to symmetric patterns.

Our method resembles decentralized methods in that each robot moves without knowing its eventual goal. Each robot uses its own location and the locations of other robots to determine

how to move. For example, in [30] and [32], motions are determined based on the Voronoi regions for the robots. Since Voronoi diagrams are unlabeled, this means the robots will move without knowing their individual roles. There are also emergent behavior methods with this property; [25] generates a potential field in which all robots have equal attractive force. This property is generally not present in the more centralized methods described above; they either order the robots [7]–[9], [13], assign a goal to each robot [11], [12], [14], [15], or assign relationships between the robots [6], [17]–[21], [23], [24]. It is evident that two significant properties of our method, explicit formation representation and permutation invariance, are generally mutually exclusive in previous work.

This paper includes and extends work we have published previously. [42] introduces the representation and its properties, and [43] adds obstacle collision detection and roadmap planning. This paper collects the previous work while adding a new path-traversal method, and robot-to-robot collision detection.

The remainder of the paper is organized as follows. In Section II, we describe more precisely our new representation of the formation space for permutation-invariant multirobot formations. In Section III, we propose a simple local planner for such formations, and investigate some basic invariance properties of the plans it creates. In Section IV, we describe how these invariance properties can be used to efficiently perform collision detection. In Section V, we combine the local planner and collision detection methods with a PRM planner to create collision-free paths for permutation-invariant formations. We discuss potential future work in Section VI, and give conclusions in Section VII. Various proofs and derivations are given in the appendices.

II. Representation

The configuration space of a labeled multirobot formation is often represented by an ordered list of configurations, one for each robot. If there are n robots, each with configuration space X, then the configuration space for the entire formation would be X^n. However, we are building a configuration space for *unlabeled* formations, which are unaffected by exchanging two robots. We call this a *formation space*, and write $FS[X]^n = X^n/S_n$, where S_n is the symmetric group of permutations of n elements. In this quotient space, $(x_1, x_2, \ldots, x_n)$ and $(x'_1, x'_2, \ldots, x'_n)$ are identified with each other if and only if (iff) they are permutations of each other. We shall sometimes refer to $FS[X]^n$ simply as FS^n if X is known. If X^n is a manifold, then FS^n is also a manifold, as its neighborhoods are topologically equivalent to the neighborhoods in X^n.

One way to investigate properties of FS^n is to derive a homeomorphism $f : FS[X]^n \to M$, where M is a manifold with known properties. This can alternatively be viewed as $f : X^n \to M$ where f is not a bijection, but is continuous, onto, invertible, and permutation-invariant, i.e., $f(x_1, x_2, \ldots, x_n) = f(x'_1, x'_2, \ldots, x'_n)$ iff $(x'_1, x'_2, \ldots, x'_n)$ is a permutation of $(x_1, x_2, \ldots, x_n)$. Now we determine a suitable f and FS^n for translating robots in two dimensions.

We represent a single configuration not as $(x, y) \in \mathbb{R}^2$, but as $z = x + iy \in \mathbb{C}$. We can use the complex plane $\mathbb{C}$ since $\mathbb{R}^2 \cong \mathbb{C}$, and we choose to do so to take advantage of the larger set of available operations on complex numbers, as well as the properties of complex polynomials. This will be clarified below. Therefore, set $X = \mathbb{C}$, and represent the robots' *workspace* $\mathcal{W}$, the space in which the individual robots move, by the complex plane $\mathbb{C}$. We now define a representation for $FS[\mathbb{C}]^n$ for any $n \in \mathbb{N}$. The representation we will define is not completely closed form, but it is well defined and permutation-invariant.

Given a set of values $z_1, z_2, \ldots, z_n \in \mathbb{C}$ representing the locations of the n robots, define polynomial $P(\lambda) = (\lambda - z_1)(\lambda - z_2)\ldots(\lambda - z_n)$. Since complex numbers form a field, this polynomial is unchanged by permuting the z-values. Therefore, it is a suitable representation for permutation-invariant point sets in the plane. Using $[1, n]$ to represent $\{1, 2, \ldots, n-1, n\}$, we write

$$\begin{aligned} P(\lambda) &= (\lambda - z_1)(\lambda - z_2)\ldots(\lambda - z_n) \\ &= \lambda^n + a_1\lambda^{n-1} + \cdots + a_{n-1}\lambda + a_n \\ &= \lambda^n + \sum_{j=1}^{n} a_j \lambda^{n-j} \end{aligned} \tag{1}$$

in which the coefficients a_k are given by

$$a_k = (-1)^k \sum_{\substack{S \subseteq [1,n] \\ |S| = k}} \prod_{j \in S} z_j. \tag{2}$$

The complex coefficients $a_1, \ldots, a_n$ can be used to define a permutation-invariant formation, or simply a *formation*. For any set of configurations $Z = \{z_1, \ldots, z_n\}$, (1) defines a unique formation $\mathbf{a} = (a_1, \ldots, a_n)$. Throughout this paper, we will refer to sets of robot configurations, like Z, as *configuration sets*, while the n-entry vectors like $\mathbf{a}$ that specify these multirobot configurations will be referred to as *formations*.

Equation (1) defines the mapping $f : \mathbb{C}^n/S_n \to \mathbb{C}^n$ to be $Z \longmapsto \mathbf{a}$. Since the roots of a polynomial vary continuously as a function of the coefficients, f^{-1}, which is $\mathbf{a} \longmapsto Z$, is a continuous mapping from a formation to a configuration set. f is a homeomorphism because of what is already known about complex polynomials.

1) Every complex polynomial of degree n has exactly n roots (counting multiple roots multiple times). This is the *Fundamental Theorem of Algebra*. See [44], [45], etc. for proofs.
2) Every polynomial has a unique factorization. This is a direct consequence of (1).
3) The mapping f from polynomial coefficients to roots is continuous in both directions. The fact that f is continuous is straightforward. The fact that f^{-1} is continuous is proven in [46].

Therefore, $FS[\mathbb{C}]^n$ is homeomorphic to $\mathbb{C}^n$. We thus define our formation space as $FS[\mathbb{C}]^n = \mathbb{C}^n$, the set of all possible n-tuples of complex polynomial coefficients.

Although f^{-1} is well defined, it has no known closed form for $n > 4$. Therefore, application of this mapping will require numerical methods.

III. A Simple Local Planner

Now that we have a suitable representation for unlabeled formations, we can do motion planning for them. To plan motion with the aid of a configuration space, one would normally: 1) map the initial and goal configurations to the configuration space; 2) determine a path from start to goal in this configuration space; and 3) map this path back into the workspace. In our case, step 3 is not straightforward, as there is no way to directly translate a trajectory in formation space to a corresponding trajectory in work space. Therefore, we use methods like those described in Section III-B to guide the robots in the workspace through discrete segments of the formation space path.

This section deals with the simplest formation space planning algorithm: the straight-line planner. Section III-A describes a simple, straight-line planner, Section III-B shows how the paths are traversed, Section III-C describes some path properties, and Section III-D shows some other results and examples.

A. The Straight-Line Planner

The plans we construct comprise a sequence of straight-line segments in the formation space. To follow a straight line from formation $(a_1, \ldots, a_n)$ to formation $(b_1, \ldots, b_n)$, follow the path

$$\ell(t) = (1-t)(a_1, \ldots, a_n) + t(b_1, \ldots, b_n)$$

where $t \in [0, 1]$. Therefore, at time t, the robots will be at the roots of the polynomial

$$\lambda^n + \sum_{k=1}^{n} \left[(1-t)a_k + tb_k\right] \lambda^{n-k} = 0. \tag{3}$$

These roots always lie on a specific polynomial in x and y. See Appendix I.

B. Traversing Paths

As described earlier, it is impossible to describe the corresponding paths in the workspace in a closed form. One can see what these paths look like by plotting the roots of the polynomial in (3) for various values of $t \in [0, 1]$. However, when following these paths, each robot needs to know which individual path to follow. To determine this, we calculate each robot's velocity at each point.

First, we write t as a function of z. Solving (3) for t, we obtain

$$z^n + \sum_{k=1}^{n} \left[(1-t)a_k + tb_k\right] z^{n-k} = 0 \tag{4}$$

$$z^n + \sum_{k=1}^{n} a_k z^{n-k} + t \sum_{k=1}^{n} (b_k - a_k) z^{n-k} = 0 \tag{5}$$

$$t = \tau(z) = \frac{z^n + \sum_{k=1}^{n} a_k z^{n-k}}{\sum_{k=1}^{n} (a_k - b_k) z^{n-k}}. \tag{6}$$

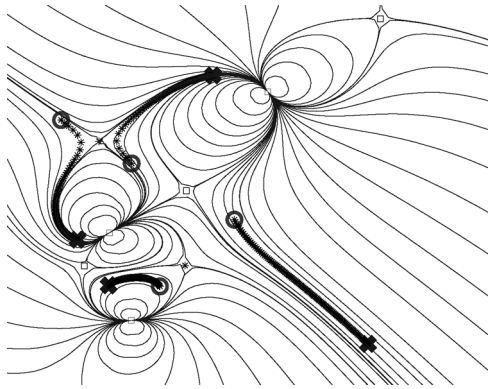


Fig. 1. Flow curves for vector field for dz/dt.

Therefore

$$\frac{dt}{dz} = \frac{g(z)}{\left(\sum_{k=1}^{n} (a_k - b_k) z^{n-k}\right)^2} \tag{7}$$

where

$$\begin{aligned} g(z) = &\left(nz^{n-1} + \sum_{k=1}^{n-1} (n-k) a_k z^{n-k-1}\right) \\ &\times \sum_{k=1}^{n} (a_k - b_k) z^{n-k} - \left(z^n + \sum_{k=1}^{n} a_k z^{n-k}\right) \\ &\times \sum_{k=1}^{n-1} (n-k)(a_k - b_k) z^{n-k-1}. \end{aligned} \tag{8}$$

$g(z)$ can be written as a polynomial in z; see Appendix II. From this, we derive

$$\frac{dz}{dt} = \frac{1}{dt/dz} = \frac{\left(\sum_{k=1}^{n} (a_k - b_k) z^{n-k}\right)^2}{g(z)}. \tag{9}$$

Removing t from the equation allows each robot to move while only considering its own location, without trying to synchronize with the other robots. Since the magnitude of this velocity can range from impractically slow (on shorter paths) to impossibly fast (on longer paths), we pick a speed appropriate to the robots and move in the direction of $(dz/dt)/|dz/dt|$. Fig. 1 shows the vector field of dz/dt.

Since the robot is not following the smooth path precisely (since numerical methods are used to solve for positions along the path), it may deviate from the course. From Fig. 1, it is evident that following the vector field will not necessarily bring a deviated robot back to its desired path. Therefore, a second velocity is added to rectify this. If $\tau(z) \in \mathbb{R}$, then z is exactly on the path. Therefore, $\Im(\tau(z))$, the imaginary component of $\tau(z)$, gives an estimate of how far off the path z is. This can be approximately corrected by changing z by

$$(t - \Re(t)) \cdot dz/dt = -i \cdot \Im(\tau(z)) \cdot dz/dt$$

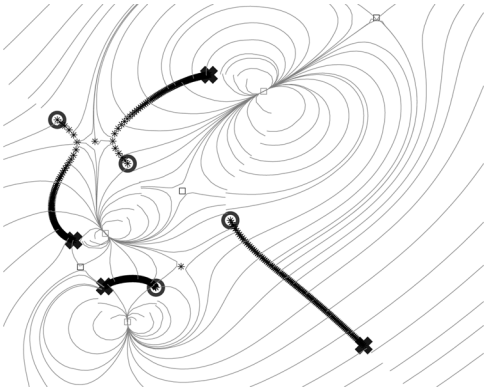


Fig. 2. Flow curves for vector field for $-i * \Im(\tau(z))dz/dt$.

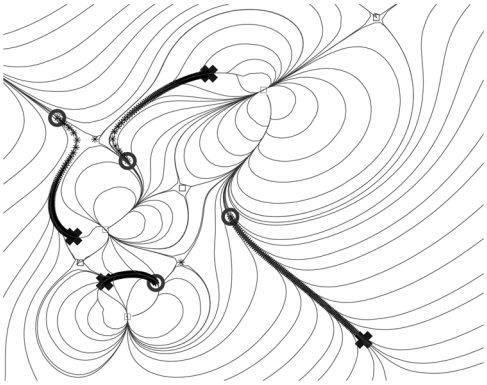


Fig. 3. Flow curves for combined vector field.

where $\Re(t)$ is the real component of t. The resulting vector field is shown in Fig. 2. The result of combining the two velocities is shown in Fig. 3. Notice that all points near the paths converge to the appropriate paths.

Therefore, at every step, each robot checks its current position, calculates its velocity, moves in that direction for a short time, and then recalculates. Since this involves evaluating a degree $(2n-1)$ polynomial and a degree $(n-1)$ polynomial, this can be done in $O(n)$ time. Using this method, robots can reach their individual goals separately, without considering each other or knowing which goal they are approaching.

There are two possible situations where this method does not work normally.

1) *Zero derivative*: This happens when the numerator of (9) is zero. This situation occurs when z is a root of the polynomials corresponding to both $\mathbf{a}$ and $\mathbf{b}$. Therefore, z is a point in both start and goal, and does not need to move. This is not a problem.

   Zero-derivative points appear on the vector fields as dipoles. In general, robots are rarely near these dipoles, as the vector field directs them away from them.

2) *Infinite derivative*: This happens when $g(z) = 0$. This only happens to robots when two robots collide. See Section IV-B for details and proof.

   Infinite derivative points appear on the vector fields as saddle points. Saddle points off the robot paths may cause problems if they are too close, as the derivatives around them can become unpredictable. In our experiments, this has always happened when two robot paths are close to each other. Saddle points can be avoided by checking in advance the distance from each saddle point to the robot paths, and having each robot move by a distance significantly less than that distance before recalculating its direction.

The distance from a saddle point z_s to the robot paths from $\mathbf{a}$ to $\mathbf{b}$ can be approximated by calculating

$$\min_j |z_s - z_j|$$

for

$$z_j \in f^{-1}\left((1-t)\mathbf{a} + t\mathbf{b}\right)$$

where

$$t = \Re\left(\tau(z_s)\right).$$

C. Properties

Due to its linear nature, as well as the nature of FS^n, the straight-line planning method has some nice algebraic properties. The upshot of these is that paths produced by straight-line planning, relative to their endpoints, are invariant to linear coordinate transformations. In the lemmas that follow, we will frequently refer to the straight-line path in FS^n that connects two configurations $\mathbf{a} = (a_1, \ldots, a_n)$ and $\mathbf{b} = (b_1, \ldots, b_n)$. We denote the path by $L(\mathbf{a}, \mathbf{b}, t)$, where

$$L(\mathbf{a}, \mathbf{b}, t) = (1-t)\mathbf{a} + t\mathbf{b} \tag{10}$$

in which $t \in [0, 1]$ parameterizes the path.

1) Translation: Define the translational operator

$$T_c(\{z_1, \ldots, z_n\}) = \{z_1 + c, z_2 + c, \ldots, z_n + c\}. \tag{11}$$

T_c translates every robot by $c \in \mathbb{C}$, which can be any translation in the plane. From T_c, define the lifted translational operator $\widetilde{T}_c = fT_cf^{-1}$. $\widetilde{T}_c$ shows the effect of T_c (which is applied to the roots) on the coefficients of the polynomial, i.e., for any configuration set $Z \subset \mathbb{C}$, if $\mathbf{a} = f(Z)$, then $\widetilde{T}_c(\mathbf{a}) = f(T_c(Z))$.

Theorem 1: $\widetilde{T}_c$ commutes with L, i.e.,

$$L\left(\widetilde{T}_c(\mathbf{a}), \widetilde{T}_c(\mathbf{b}), t\right) = \widetilde{T}_c\left(L(\mathbf{a}, \mathbf{b}, t)\right). \tag{12}$$

This means that if the initial and goal configurations both translate by the same amount, the resulting path will translate accordingly. See Fig. 4 for an example of this.

Proof: See Appendix III. ■

An important consequence of this is that the location of the origin of the coordinate frame is irrelevant when doing straight-line planning.

2) Scaling and Rotation: In the complex plane, scaling and rotation are actually the same operation: multiplication. Every $c \in \mathbb{C}$ can be written as $c = re^{i\theta}$ where $r, \theta \in \mathbb{R}$. Multiplying

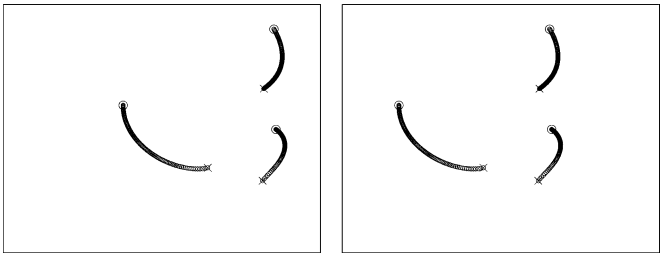


Fig. 4. Before and after translation.

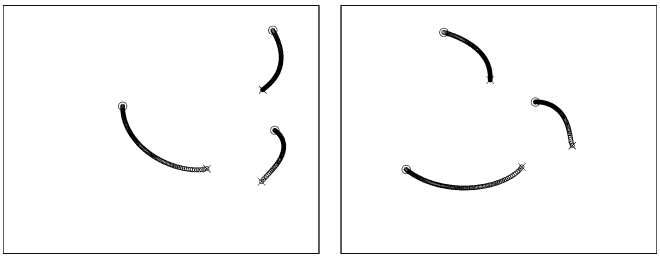


Fig. 5. Before and after rotation.

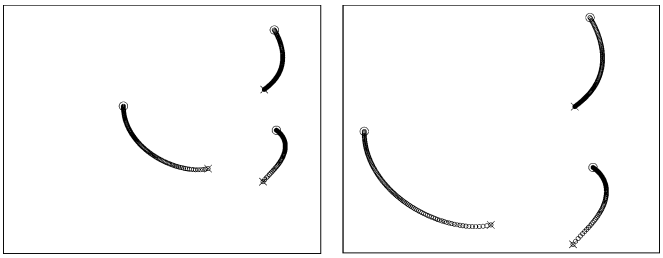


Fig. 6. Before and after scaling.

by c scales by r and rotates by θ about the origin. Define the scale/rotate operator

$$SR_c(\{z_1,\ldots,z_n\}) = \{cz_1, cz_2, \ldots, cz_n\}. \tag{13}$$

SR_c applies the scale and rotation operations corresponding to c to all robots. From SR_c, define the lifted scale/rotate operator $\widetilde{SR_c} = fSR_cf^{-1}$, analogous to $\widetilde{T}_c$.

Theorem 2: $\widetilde{SR_c}$ commutes with L, i.e.,

$$L\left(\widetilde{SR_c}(\mathbf{a}), \widetilde{SR_c}(\mathbf{b}), t\right) = \widetilde{SR_c}\left(L(\mathbf{a},\mathbf{b},t)\right). \tag{14}$$

This means that if the initial and goal configurations both scale and rotate by the same amount, the resulting path will scale and rotate accordingly. See Figs. 5 and 6 for examples.

Proof: See Appendix III. ■

Scaling and rotation operations can be composed with translation operations to produce operations that scale and rotate about any point. Consequently, the orientation of the coordinate frame is irrelevant, as is the unit size, when doing straight-line planning.

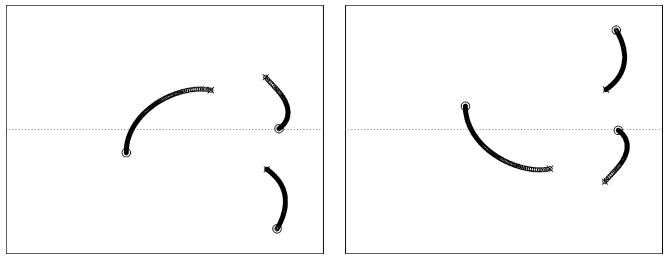


Fig. 7. Before and after reflection.

3) Reflection: Reflection about the real axis is performed by conjugating all robot coordinates. Therefore, define the reflection operator

$$F(\{z_1,\ldots,z_n\}) = \{\overline{z}_1,\ldots,\overline{z}_n\}$$

where $\overline{z}$ is the complex conjugate of z. F reflects all robots across the real axis. From F, define the lifted reflection operator $\widetilde{F} = fFf^{-1}$, analogous to $\widetilde{T}_c$ and $\widetilde{SR_c}$.

Theorem 3: $\widetilde{F}$ commutes with L, i.e.,

$$L\left(\widetilde{F}(\mathbf{a}), \widetilde{F}(\mathbf{b}), t\right) = \widetilde{F}\left(L(\mathbf{a},\mathbf{b},t)\right). \tag{15}$$

This means that if the initial and goal configurations are both reflected about the real axis, the resulting path will also be reflected about the real axis. See Fig. 7 for an example.

Proof: See Appendix III. ■

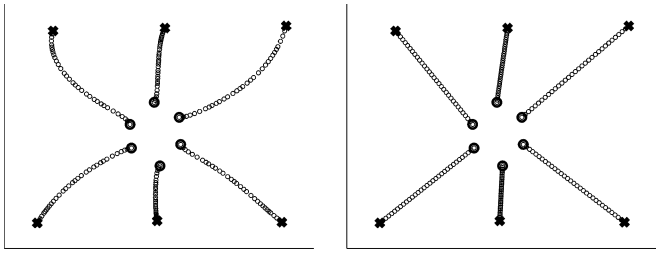
Composing reflection about the horizontal axis with rotation and translation produces all reflections, so reflecting both start and goal produces a path reflected the same way. Therefore, left-hand/right-hand coordinate frame orientation is irrelevant to straight-line planning.

D. Results

We have run our straight-line planning algorithm on a variety of start and goal configurations for varying formation sizes. The resulting paths are generally simple and straightforward. The paths never intersect themselves or other paths, with the exception of certain degenerate cases, described here.

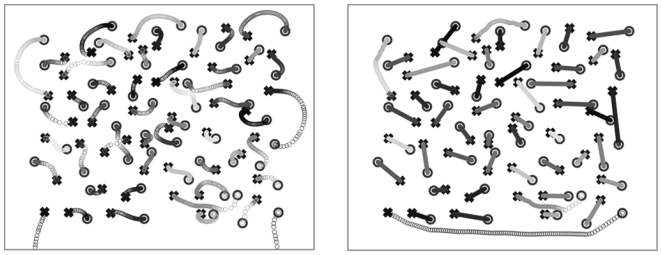
Figs. 8(a), 9(a), and 10(a) show some typical results. In all figures, robots move from X's to O's. Often the resulting paths are nice, simple, and almost direct. Fig. 8(a) shows an example of this. But often the resulting paths are convoluted and roundabout. Fig. 9(a) shows many robots following convoluted paths, and some moving directly away from their eventual goals. Fig. 10(a) shows a more extreme example of this, one where the start and goal formations each consist of robots that are close to each other, but the start and goal are far apart. In this case, some robots move straight to a goal, but many robots take unusually roundabout routes, moving far away from the other robots before moving in toward the goal.

We run a simple smoothing algorithm on these paths. All paths are iteratively contracted by the method below, which averages neighboring points while ensuring that paths never intersect.



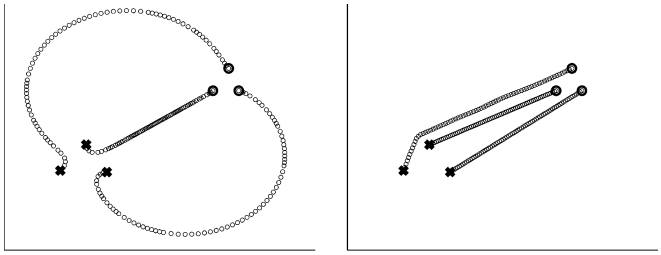
(a) (b)

Fig. 8. Typical plan for six robots. (a) Before smoothing. (b) After smoothing.



(a) (b)

Fig. 9. Typical plan for 50 robots. (a) Before smoothing. (b) After smoothing.



(a) (b)

Fig. 10. Plan for three robots that requires extraneous motion by two of the robots. (a) Before smoothing. (b) After smoothing.

Each path can be written as a sequence of points. These are the points at which the robot recalculated its direction. Let s be the maximum distance a robot is permitted to travel before recalculating its direction. Let w be the minimum permissible distance between robots. Usually, this is the width of the robot. At each iteration, each path is replaced by a new path, such that we have the following.

1) The first and last points remain the same.
2) Any point whose two surrounding points (one before, one after) are too close together (i.e., their distance is less than s) is removed.
3) Any point that is too close to another robot's path (i.e., within distance w) is replaced with the closest point far enough away (at distance w) from the path. For example, if z is the point being examined, and z_o is the closest point on another robot's path, and $|z-z_o| < w$, then z is replaced by

$$z_o + w\frac{z - z_o}{|z - z_o|}.$$

4) All other points are replaced with the mean of the four surrounding points (two before, two after).

This process is repeated until the path cannot be improved on. Usually, this is less than 100 iterations. While smoothed paths no longer follow the paths described mathematically in the earlier sections, the paths are more efficient, and each robot can precalculate its path and do its own smoothing separately. Figs. 8(b), 9(b), and 10(b) show smoothing results.

Although some paths appear to collide in these figures, they are just very close to each other. The only way we have found a collision is by contriving the initial and goal configurations specifically to generate a collision. Section IV-B discusses this further.

IV. Collisions

In order for a formation-planning system to be useful, it must be able to recognize and avoid collisions. In this section, we address two types of collisions: robot–obstacle collisions, and robot–robot collisions.

A. Robot–Obstacle Collisions

To define robot–obstacle collisions, we first define the *swept volume* of a path $V(\mathbf{a}, \mathbf{b})$ to be the locus of points in the workspace traversed by the robots while following the formation space path $L(\mathbf{a}, \mathbf{b}, t)$ from (10)

$$V(\mathbf{a}, \mathbf{b}) = \left\{ z \in \mathbb{C} | z \in f^{-1}\left((1-t)\mathbf{a} + t\mathbf{b}\right), t \in [0,1] \right\}$$

i.e., $V(\mathbf{a}, \mathbf{b})$ is the set of all roots of $L(\mathbf{a}, \mathbf{b}, t)$ for all values of $t \in [0,1]$. Given polynomial configurations $\mathbf{a}$ and $\mathbf{b}$, and an obstacle region $\mathcal{O} \subset \mathcal{W}$, the straight-line path in the formation space from formation $\mathbf{a}$ to formation $\mathbf{b}$ generates a collision iff $V(\mathbf{a}, \mathbf{b}) \cap \mathcal{O} \neq \emptyset$.

In this section, we will incrementally build up the methods to check for collisions. We will check point obstacles, use point obstacles to check horizontal-line obstacles, use horizontal-line obstacles to check line-segment obstacles, and use line-segment obstacles to check polygonal obstacles. Although point obstacles are degenerate and horizontal-line obstacles are uncommon, they are necessary as building blocks toward the intended goal of segments and polygons.

1) Point Obstacles: Consider a point obstacle at location z_c. For a given formation $\mathbf{a}$, a collision occurs iff

$$z_c^n + \sum_{j=1}^{n} a_j z_c^{n-j} = 0$$

i.e., z_c is a root of (1). To determine if $V(\mathbf{a}, \mathbf{b})$ collides with z_c, recall $\tau(z)$ from (6) from Section III-B

$$\tau(z) = \frac{z^n + \sum_{k=1}^{n} a_k z^{n-k}}{\sum_{k=1}^{n} (a_k - b_k) z^{n-k}}.$$

$$\omega_m = \begin{cases} s_m - q_m & m = 1 \\ s_m - q_m + \sum_{j=1}^{m-1} p_{m-j}(s_j - q_j) + q_{m-j}(p_j - r_j) & 2 \le m \le n \\ \sum_{j=m-n}^{n} p_{m-j}(s_j - q_j) + q_{m-j}(p_j - r_j) & n+1 \le m \le 2n \end{cases}$$

Fig. 11. Formulae for coefficients of $\Omega(x)$.

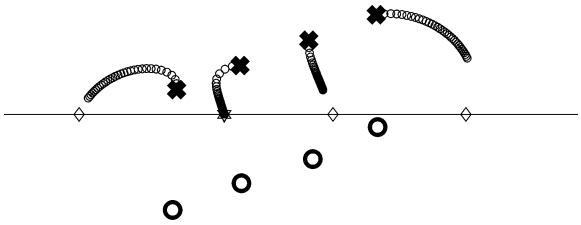


Fig. 12. Collision with real axis.

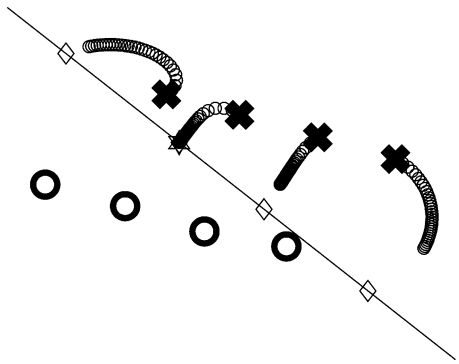


Fig. 13. Collision with segment.

Since $\tau(z)$ gives the value for t necessary for $L(\mathbf{a}, \mathbf{b}, t)$ to contain z, it needs to be real and between 0 and 1 in order for z to be on the path. Therefore, $V(\mathbf{a}, \mathbf{b})$ collides with z_c iff $\tau(z_c) \in \mathbb{R}$ and $0 \le \tau(z_c) \le 1$.

Notice that $\tau(z)$ is unique. This means that given any point in $\mathbb{C}$, robots moving in a coefficient-space straight-line path will cross that point at most once. Therefore, single robot paths will not cross themselves, and different robots' paths will only intersect at interrobot collisions (when they reach the same point at the same time; they will not reach the same point at different times). Interrobot collisions are dealt with below, in Section IV-B.

2) Horizontal Segment Obstacles: We showed in Section III-C that if the start and goal are translated and rotated, the paths transform accordingly. Since any edge can be rotated and translated onto the real axis, we can rotate and translate $\mathbf{a}$ and $\mathbf{b}$ the same way. If $V(\mathbf{a}, \mathbf{b})$ collides with an edge, then the transformed path will collide with a subset of the real axis. This means we can test collision with any line segment if we can test for collision with the real axis.

Theorem 4: For any $\mathbf{p}, \mathbf{q}, \mathbf{r}, \mathbf{s} \in \mathbb{R}^n$, $V(\mathbf{p}+\mathbf{q}i, \mathbf{r}+\mathbf{s}i)$ collides with the real axis iff there exists an $x \in \mathbb{R}$ such that x is a root of the polynomial $\Omega(x) = \sum_{m=1}^{2n} \omega_m x^{2n-m}$, where ω_m is given in Fig. 11, and $\tau(x) \in [0, 1]$.

Proof: See Appendix IV ∎

This method can also be used to test for collisions with a segment of the real axis, e.g., a line segment on the real axis from $x_{\min}$ to $x_{\max}$ (where $x_{\min} < x_{\max}$). This segment can be checked for collision by rejecting any $x_j \notin [x_{\min}, x_{\max}]$ without calculating $\tau(x_j)$.

Fig. 12 shows an example of collisions with the real axis. The robots are moving from the X's toward the O's, and the diamonds mark the points of collision with the real axis, as returned by the method described here. The robot paths are given up to the earliest collision, i.e., the x_j among the roots of $\Omega(x)$ with minimum $\tau(x_j)$.

3) Arbitrary Segment Obstacles: Now that we can test for collision with segments on the real axis, we can test for collisions with any segment from $u \in \mathbb{C}$ to $v \in \mathbb{C}$. To do so, we make use of the transformations in Section III-C. Any line segment can be rotated, scaled, and translated to the real axis via T_{-u} and $SR_{\overline{v}-\overline{u}}$

$$u' = SR_{\overline{v}-\overline{u}}(T_{-u}(u)) = 0$$
$$v' = SR_{\overline{v}-\overline{u}}(T_{-u}(v)) = |v-u|^2.$$

We then take the transformational operators we applied to u and v, and apply the corresponding lifted transformation to the polynomials

$$\mathbf{a}' = \widetilde{SR}_{\overline{v}-\overline{u}}\left(\widetilde{T}_{-u}(\mathbf{a})\right)$$
$$\mathbf{b}' = \widetilde{SR}_{\overline{v}-\overline{u}}\left(\widetilde{T}_{-u}(\mathbf{b})\right).$$

If $V(\mathbf{a}, \mathbf{b})$ collides with uv, then $V(\mathbf{a}', \mathbf{b}')$ will also collide with the real axis between u' and v'. We can, therefore, apply the method in Section IV-A.2 to $\mathbf{a}'$, $\mathbf{b}'$, u', and v' to determine if the original path collides with uv.

Fig. 13 shows an example of collisions with arbitrary segments. The robots and edge in this example can be rotated and scaled to match Fig. 12 exactly, and the resulting collision information transforms accordingly. Therefore, we can use the collision information from Fig. 12 to find the collisions in Fig. 13.

4) Polygonal Obstacle Collisions: We can use the edge-collision checker to check collision with any polygonal obstacle region by checking for collision with each edge, as a robot cannot enter the polygon without crossing an edge. We stop with polygons, because nonpolygonal regions can be approximated by polygons.

B. Robot-to-Robot Collisions

Although we have not been able to generate collisions between robots by picking arbitrary initial and goal configurations, there are ways to explicitly generate such collisions. Note that

in the current representation, robots have zero size, so a collision only happens when two robots coincide. Here we show a specific form to which all collisions conform.

1) Collision Points: In order to define collision paths, we need to first define collision points. Define $\mathrm{Col} \subset FS^n$ to be the set of points that correspond to polynomials with multiple roots. We call these *collision points* or *collision formations*. We determine these properties about this set.

1) Col has a lower dimensionality than FS^n.
2) Col is path-connected.
3) Col is closed, and has no interior.
4) Col is nowhere dense in FS^n.

Proofs of these properties can be found in Appendix V.

2) Collision Paths: Define a *generalized collision path* to be a path that contains a collision between at least two robots somewhere on its interior. Since our current planning method uses straight lines in $\mathbb{C}^n$, a generalized collision path must be a line through a collision point. Using this idea, we show that every generalized collision path can be generated by a single method. Note that we are not concerned with *trivial collision paths*, paths with start or goal collisions, but no collisions in between.

Let $\mathbf{a} \in \mathrm{Col}$ be an arbitrary collision point. Any collision paths involving this $\mathbf{a}$ are straight lines in the formation space through $\mathbf{a}$, and therefore can be written as $L(\mathbf{a}-\mathbf{d}, \mathbf{a}+k\mathbf{d}, t)$, where L is from (10), $\mathbf{d} = (d_1, \ldots, d_n) \in \mathbb{C}^n$, and $k \in \mathbb{R}^+$. The collision along this path will be at time $t = 1/(k+1)$. Therefore, every nontrivial collision path is uniquely defined by a single $(\mathbf{a}, \mathbf{d}, k) \in \mathrm{Col} \times \mathbb{C}^n \times \mathbb{R}^+$.

We can test for collision paths by looking for the path's collision points. We do this by noting that a polynomial $p(z)$ has a multiple root at z iff $p(z) = p'(z) = 0$. Therefore, if z is a collision point for $L(\mathbf{a}, \mathbf{b}, t)$, then not only must (4) hold, but also its derivative

$$nz^{n-1} + \sum_{k=1}^{n-1} [(1-t)a_k + tb_k](n-k)z^{n-k-1} = 0.$$

Solving both equations for t yields

$$t = \frac{z^n + \sum_{k=1}^{n} a_k z^{n-k}}{\sum_{k=1}^{n} (a_k - b_k) z^{n-k}} = \frac{nz^{n-1} + \sum_{k=1}^{n-1} (n-k) a_k z^{n-k-1}}{\sum_{k=1}^{n-1} (n-k)(a_k - b_k) z^{n-k-1}}.$$

Therefore

$$0 = \frac{nz^{n-1} + \sum_{k=1}^{n-1} (n-k) a_k z^{n-k-1}}{\sum_{k=1}^{n-1} (n-k)(a_k - b_k) z^{n-k-1}} - \frac{z^n + \sum_{k=1}^{n} a_k z^{n-k}}{\sum_{k=1}^{n} (a_k - b_k) z^{n-k}}$$

$$= \frac{g(z)}{\left(\sum_{k=1}^{n-1} (n-k)(a_k - b_k) z^{n-k-1}\right) \sum_{k=1}^{n} (a_k - b_k) z^{n-k}}$$

where $g(z)$ is from (8). Therefore, we can check for collisions by examining the roots of $g(z)$. If any of them satisfy $\tau(z) \in [0, 1]$, then the path has a collision.

Fig. 14. Paths with collisions.

Fig. 15. Destination slightly altered; no collisions.

This method is for testing collisions between point robots. Using the method described in Section III-B, one can also check for near-misses, which would be collisions between robots of positive width. If the distance from a saddle point to the robot path is less than half the width of the robot, the robots will collide. In all of our experiments, near-misses have been near roots of $g(z)$.

These specific cases are the only formations that generate collisions. Since Col is nowhere dense, and the set of collision paths has a lower dimensionality than the set of formation paths, we conclude that the set of collision paths is nowhere dense. Therefore, the probability of a randomly selected path containing a collision is zero. In our experiments, we have been unable to generate a collision except through the methods described earlier in this section.

To show how unlikely collisions are in practice, consider Figs. 14 and 15. Fig. 14 shows a path with four robots and two collisions. In the figure, there are two instances of a light gray path touching a dark gray path (these paths do not cross). At both collision points, the robot paths just touch, and then diverge. The start and goal formations have been selected specifically to generate those two collisions. In Fig. 15, the goal locations in Fig. 14 have been adjusted by small random amounts (approximately 2% of the distance to the collision points). The resulting paths do not collide; they are not even close to colliding. The typical randomly selected or human-selected start and goal formations will not produce collisions.

V. Roadmap Planning

The straight-line planner, combined with the obstacle-collision checker, form an effective local planner. We use it to apply simple PRM [47] methods to permutation-invariant formations. To generate each node in the roadmap, we generated n uniform

random samples $z_1, \ldots, z_n$ from the collision-free workspace, and constructed a node at $\mathbf{a} = f(\{z_1, \ldots, z_n\})$. Therefore, m samples in the formation space require mn samples in the workspace. The roadmap nodes corresponding to formations $\mathbf{a}$ and $\mathbf{b}$ were connected with a roadmap edge iff $V(\mathbf{a}, \mathbf{b})$ did not collide with any edges, and there were no near-misses as described in Section IV-B. The start and goal formation were selected by the user, and their nodes were added to the roadmap afterwards. To determine routes, we used A* with a metric of

$$d(\mathbf{a}, \mathbf{b}) = \left(\sum_{j=1}^{n} \left| a'_j - b'_j \right|^{\frac{n}{j}} \right)^{\frac{1}{n}} \tag{16}$$

where

$$\mathbf{a}' = \widetilde{T}_{c'}(\mathbf{a})$$
$$\mathbf{b}' = \widetilde{T}_{c'}(\mathbf{b})$$
$$c' = a_1/n.$$

This metric is designed so that for any $c \in \mathbb{C}$

$$d\left(\widetilde{T}_c(\mathbf{a}), \widetilde{T}_c(\mathbf{b})\right) = d(\mathbf{a}, \mathbf{b})$$
$$d\left(\widetilde{SR}_c(\mathbf{a}), \widetilde{SR}_c(\mathbf{a})\right) = |c| \cdot d(\mathbf{a}, \mathbf{b}).$$

This means the metric is invariant to translation and rotation.

In our experiments, finding the roots of a degree-n polynomial of the type our method uses (bounded, single roots only) has generally taken $O(n^2)$ running time. For each roadmap edge, checking for obstacle collisions requires solving a $(2n - 1)$-degree polynomial for each obstacle edge. This requires $O(n^2)$ running time per obstacle edge. Checking for robot-to-robot collisions requires solving a $(2n - 2)$-degree polynomial, and then for each root, determining the distance to each robot path. Since that requires solving $2n-2$ polynomials of degree n, checking for robot-to-robot collisions requires $O(n^3)$ running time. Therefore, if there are n robots and e obstacle edges, each roadmap edge requires $O(n^3 + en^2)$ time to check. If there are m roadmap nodes, the entire roadmap takes $O(m^2n^2(n + e))$ time to generate.

We implemented this planning method in MATLAB on a 2.0 GHz CPU. Figs. 16–19 show the results of planning for three robots in an environment that simulates an office with cubicles. This environment, and the environments used in later examples, are inspired by examples from [14]. The PRM used 200 nodes, and was generated in 2 min. The individual figures show the steps used to generate paths.

1) *Generate paths*. For every edge $(\mathbf{a_j}, \mathbf{a_{j+1}})$ in the route, follow the procedure described in Section III-B to move the robots from $\mathbf{a_j}$ to $\mathbf{a_{j+1}}$. See Fig. 16.
2) *Remove cusps and loops*. For each path, any time a robot returns to a point it has already been to (or very close to), remove the steps in between. This has the effect of removing places where the robot's path crosses or retraces itself. See Fig. 17.
3) *Exchange path segments*. Every time two paths cross, have each crossing robot continue along the opposite robot's path. Now the robots' paths do not cross, but just touch each other. See Fig. 18.
4) *Smooth paths*. Apply the smoothing algorithm described in Section III-D, augmented to prevent obstacle collisions. See Fig. 19.

This process of generating the final path from the roadmap took 3 min.

Figs. 20–23 show an example with seven robots in a more restrictive environment of rooms and corridors. The PRM used 500 nodes and was generated in 1 h. The final paths were generated in 5 min. Fig. 24 shows the final plan for 10 robots in a room with 12 obstacles. Here, the roadmap had 2000 nodes, which took 28 h to generate. The final paths were generated in 8 min.

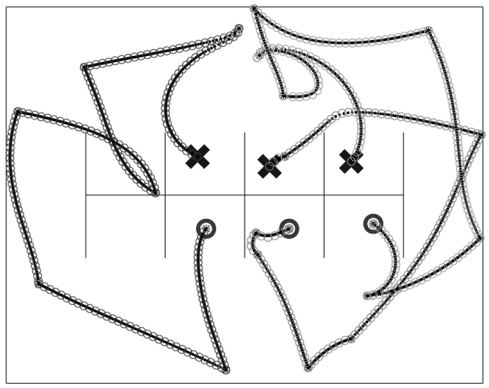


Fig. 16. Generate paths.

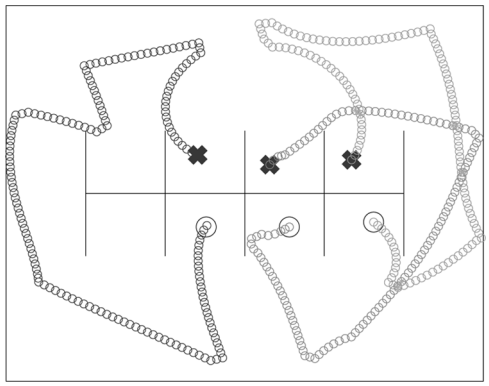


Fig. 17. Remove cusps and loops.

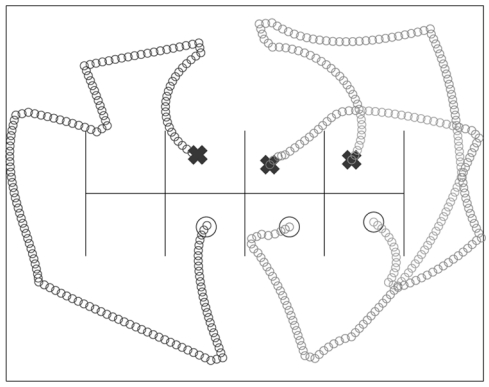


Fig. 18. Exchange path segments.

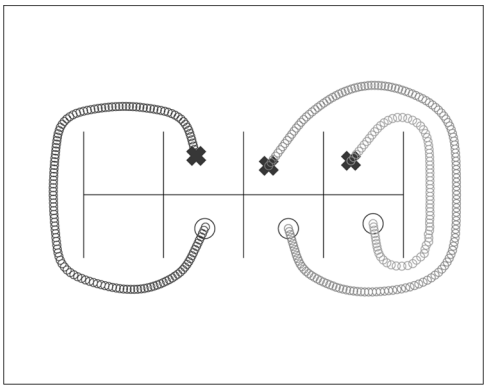


Fig. 19. Smooth paths.

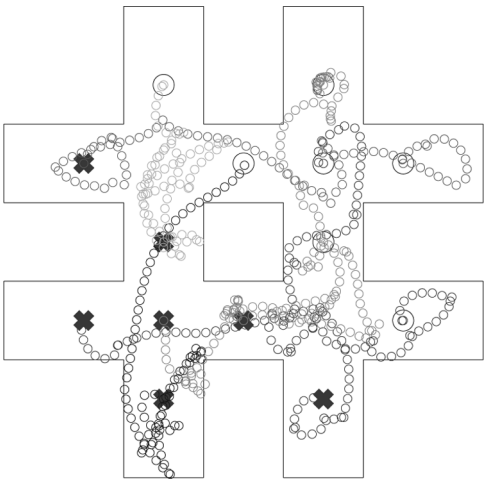


Fig. 20. Generate paths.

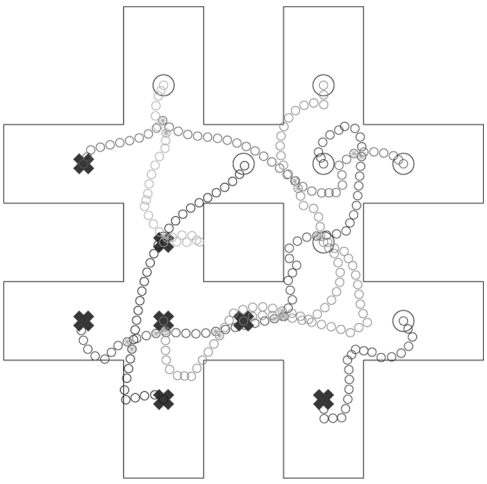


Fig. 21. Remove cusps and loops.

The PRM used here is simple and not guaranteed to succeed. In our experiments, paths could be found for most start/goal pairs in most environments once enough nodes were generated for the roadmap.

VI. Future Work

Our method applies only for the case of robots translating in the plane. This is because this method relies on the properties of

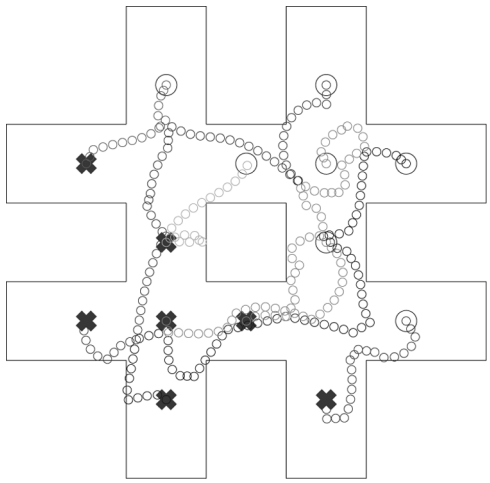


Fig. 22. Exchange path segments.

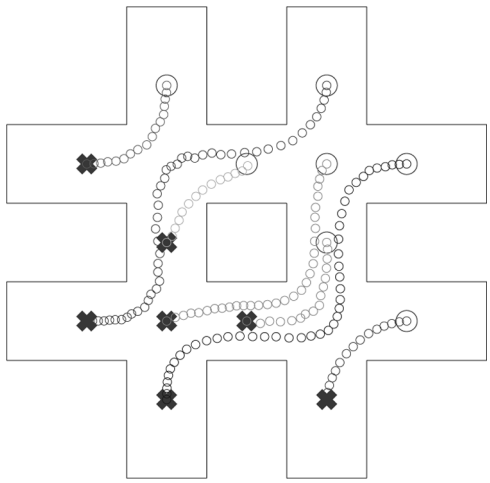


Fig. 23. Smooth paths.

2-D complex numbers. It might seem that our approach could be extended to higher dimensional configuration spaces by exploiting higher dimensional complex numbers, such as quaternions. Unfortunately, quaternions cannot be used the same way to build formations, as quaternions do not commute. Furthermore, a quaternion polynomial does not necessarily have a finite set of solutions. For example, $\lambda^2 + 1 = 0$ has infinite solution set $\{ai + bj + ck | a^2 + b^2 + c^2 = 1\}$. Therefore, a new method entirely would be needed to plan permutation-invariant formations for higher dimensions.

The system is designed for point robots. Circular robots can be dealt with by expanding the obstacles by the radius of the robots, and checking for near-misses with a distance of the diameter of the robots (See Section IV-B.1). However, the latter method is not proven; it has only been confirmed experimentally. Furthermore, robots with noncircular shapes (e.g., polygons) would need to be handled differently.

The robots in this method are holonomic. Since it is possible to calculate second derivative and curvature at any point, it may be feasible to do more detailed analysis over any path, and determine whether differential drive or car-like robots can traverse

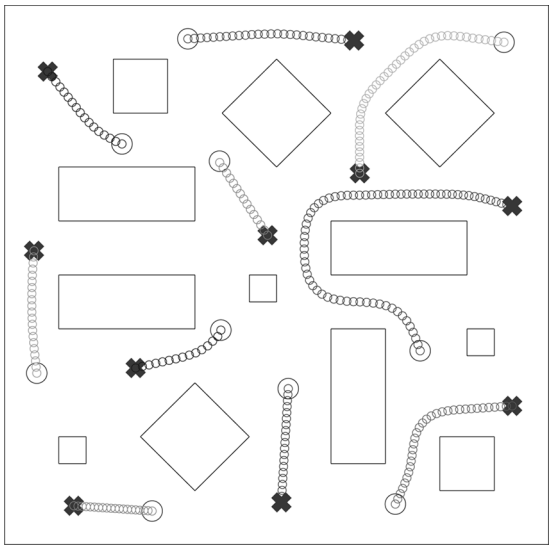


Fig. 24. 10 robots, 12 obstacles.

the paths. Similarly, calculating other properties may reveal information applicable to nonholonomic control systems.

There is work in system stability that is relevant to our work. Kharitonov [48] studied the stability robustness of interval polynomials, which are families of polynomials defined by prespecifying a list of intervals. A polynomial is contained in an interval polynomial if each coefficient is contained in the corresponding interval polynomial's interval. Kharitonov's law gives a way of determining if all polynomials in the family have roots with negative real components. While [48] tests structurally hypercubic regions of the coefficient space, [49] tests general polytopes, and [50] tests ellipsoidal regions. Also, [51] and [52] show how to test if an interval polynomial has any roots in a specific sector of the left half of the complex plane. Using the transformational operators defined in Section III-C, one can turn any obstacle region into a stability problem. Therefore, it may be possible to find large regions in the formation space that are completely obstacle-free, and then plan motion within and between these regions.

VII. Conclusion

In this paper, we have built a continuous configuration-space representation for formations of unlabeled robots that translate in the plane. We have shown that the formation space is homeomorphic to the configuration space for labeled formations, and that the representation is both invertible and permutation-invariant. We have built a local planner in this formation space, and shown some of its properties. We have demonstrated obstacle collisions for this local planner, and used it to build a general formation planner.

Appendix I
$V(\mathbf{a},\mathbf{b})$ is a Polynomial

In this appendix, we show that the paths generated by a straight-line path in formation space form part of a polynomial in x and y. We saw in Section IV-A-1 that $z \in V(\mathbf{a},\mathbf{b})$ iff

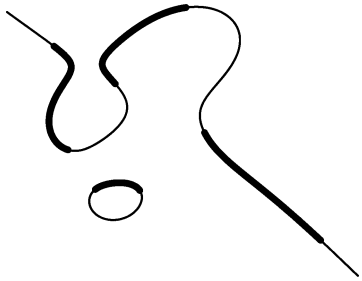


Fig. 25. The robot paths are a subset of the polynomial.

$\tau(z) \in [0,1]$. A necessary condition of this is $Im(\tau(z)) = 0$. We rewrite $\tau(z)$ as

$$\begin{aligned}\tau(z) &= \frac{z^n + \sum_{k=1}^{n} a_k z^{n-k}}{\sum_{k=1}^{n}(a_k - b_k)z^{n-k}} \cdot \frac{\overline{\left(\sum_{k=1}^{n}(a_k - b_k)z^{n-k}\right)}}{\overline{\left(\sum_{k=1}^{n}(a_k - b_k)z^{n-k}\right)}} \\ &= \frac{\left(z^n + \sum_{k=1}^{n} a_k z^{n-k}\right)\left(\sum_{k=1}^{n}(\overline{a}_k - \overline{b}_k)\overline{z}^{n-k}\right)}{\left|\sum_{k=1}^{n}(a_k - b_k)z^{n-k}\right|^2} \\ &= \frac{z^n \sum_{j=1}^{n}(\overline{a}_j - \overline{b}_j)\overline{z}^{n-j}}{\left|\sum_{k=1}^{n}(a_k - b_k)z^{n-k}\right|^2} + \frac{\sum_{j=1}^{n}\sum_{k=1}^{n} a_k(\overline{a}_j - \overline{b}_j)z^{n-k}\overline{z}^{n-j}}{\left|\sum_{k=1}^{n}(a_k - b_k)z^{n-k}\right|^2}.\end{aligned}$$

Since the denominator is real, $\tau(z)$ is real iff the imaginary component of the numerator is 0. Also note

$$2\sum_{j=1}^{n}\sum_{k=1}^{n} a_k\overline{a}_j z^{n-k}\overline{z}^{n-j} = \left|\sum_{j=1}^{n} a_j z^{n-j}\right|^2 - \sum_{j=1}^{n}|a_j z^{n-j}|^2.$$

Therefore

$$\Im\left(\sum_{j=1}^{n}\sum_{k=1}^{n} a_k\overline{a}_j z^{n-k}\overline{z}^{n-j}\right) = 0.$$

Therefore, in order for z to be on the path

$$\Im\left(z^n\sum_{j=1}^{n}(\overline{a}_j - \overline{b}_j)\overline{z}^{n-j} - \sum_{j=1}^{n}\sum_{k=1}^{n} a_k\overline{b}_j z^{n-k}\overline{z}^{n-j}\right) = 0$$

must hold. Substituting $z = x + yi$ and $\overline{z} = x - yi$ into this equation and setting the imaginary component to zero produces a polynomial of degree $2n-1$ in x and y. Clearly, the points along the robots' paths lie on this polynomial.

This does not mean that all points in the polynomial will be in the robot paths, as there are values of z where $\tau(z) < 0$ or $\tau(z) > 1$. In Fig. 25, the thick curves trace the robot's paths, while the thin curves show the remainder of the polynomial's graph.

APPENDIX II
SIMPLIFYING g

Define

$$N = z^n + \sum_{k=1}^{n} a_k z^{n-k}$$
$$D = \sum_{k=1}^{n} (a_k - b_k) z^{n-k}.$$

We can now write (6) as

$$t = \frac{N}{D}.$$

(7) then becomes

$$\frac{dt}{dz} = \frac{g(z)}{\left(\sum_{k=1}^{n}(a_k - b_k)z^{n-k}\right)^2} = \frac{N'D - D'N}{D^2}.$$

Now

$$N'D = \sum_{j=1}^{n} n(a_j - b_j) z^{2n-j-1} + \sum_{j=1}^{n}\sum_{k=1}^{n}(n-j)a_j(a_k - b_k)z^{2n-j-k-1}$$
$$D'N = \sum_{j=1}^{n}(n-j)(a_j - b_j)z^{2n-j-1} + \sum_{j=1}^{n}\sum_{k=1}^{n}(n-k)a_j(a_k - b_k)z^{2n-j-k-1}.$$

Therefore

$$g(z) = N'D - D'N = \sum_{j=1}^{n} j(a_j - b_j)z^{2n-j-1} + \sum_{j=1}^{n}\sum_{k=1}^{n}(k-j)a_j(a_k - b_k)z^{2n-j-k-1}.$$

Setting $m = j + k$ and rearranging terms yields

$$g(z) = \sum_{j=1}^{n} j(a_j - b_j)z^{2n-j-1} + \sum_{m=2}^{2n-1}\sum_{j=1}^{n}(m-2j)a_j(a_{m-j} - b_{m-j})z^{2n-m-1}. \quad (17)$$

The terms in the double sum are only valid if $1 \le m - j \le n$, i.e., $m - n \le j \le m - 1$, while still satisfying $1 \le j \le n$. Also, if $j = k = n$, then the term inside the double sum is 0. Therefore, we write (17) as

$$g(z) = \sum_{j=1}^{n} j(a_j - b_j)z^{2n-j-1} + \sum_{m=2}^{n}\sum_{j=1}^{m-1}(m-2j)a_j(a_{m-j} - b_{m-j})z^{2n-m-1} + \sum_{m=n+1}^{2n-1}\sum_{j=m-n}^{n}(m-2j)a_j(a_{m-j} - b_{m-j})z^{2n-m-1}.$$

Therefore, $g(z)$ can be written as the polynomial $\sum_{m=1}^{2n-1} g_m x^{2n-1-m}$, where

$$g_m = \begin{cases} m(a_m - b_m), & m = 1 \\ m(a_m - b_m) + \sum_{j=1}^{m-1}(m-2j)a_j(a_{m-j} - b_{m-j}), & 2 \le m \le n \\ \sum_{j=m-n}^{n}(m-2j)a_j(a_{m-j} - b_{m-j}), & n+1 \le m < 2n. \end{cases}$$

APPENDIX III
PROOFS OF LEMMAS IN SECTION III-C

Theorem 1: $\widetilde{T}_c$ commutes with L, i.e.,

$$L\left(\widetilde{T}_c(\mathbf{a}), \widetilde{T}_c(\mathbf{b}), t\right) = \widetilde{T}_c\left(L(\mathbf{a}, \mathbf{b}, t)\right). \quad (18)$$

Proof: First, set $\mathbf{a} = f(Z)$ and determine $\mathbf{a}' = \widetilde{T}_c(\mathbf{a})$. We combine (2) and (11) to get

$$a'_k = (-1)^k \sum_{\substack{S \subseteq [1,n] \\ |S| = k}} \prod_{j \in S}(z_j + c).$$

Since the expansion of $\prod_{j \in S}(z_j + c)$ consists solely of terms of the form $z_{j_1} z_{j_2} \ldots z_{j_\ell} c^{n-\ell}$, with one term for each possible $0 \le \ell \le k$ and $\{j_1, j_2, \ldots, j_\ell\} \subseteq S$, we can write

$$\prod_{j \in S}(z_j + c) = \sum_{T \subseteq S} c^{k-|T|} \prod_{j \in T} z_j.$$

Therefore

$$a'_k = (-1)^k \sum_{\substack{S \subseteq [1,n] \\ |S| = k}} \sum_{T \subseteq S} c^{k-|T|} \prod_{j \in T} z_j.$$

Reordering the sums produces

$$a'_k = (-1)^k \sum_{\substack{T \subseteq [1,n] \\ |T| \le k}} \sum_{\substack{S \supseteq T \\ |S| = k \\ S \subseteq [1,n]}} c^{k-|T|} \prod_{j \in T} z_j.$$

The terms of the second sum are independent of S, and each term appears once for each S of size k that contains T. Therefore

$$a'_k = (-1)^k \sum_{\substack{T\subseteq[1,n]\\ |T|\le k}} \left(c^{k-|T|} \prod_{j\in T} z_j \right) |\mathcal{S}_k(T)|$$

where

$$\mathcal{S}_k(T) = \{S \subseteq [1,n] | T \subseteq S, |S| = k\}$$

is the collection of sets of size k that contain T.

Since each set in $\mathcal{S}_k(T)$ contains all of T plus $k-|T|$ elements not in T, we have

$$|\mathcal{S}_k(T)| = \binom{n-|T|}{k-|T|}$$

and therefore

$$a'_k = (-1)^k \sum_{\substack{T\subseteq[1,n]\\ |T|\le k}} \binom{n-|T|}{k-|T|} c^{k-|T|} \prod_{j\in T} z_j.$$

Now, we set $j = |T|$ and separate the cases where $|T| = 1$, $|T| = 2$, etc.

$$a'_k = (-1)^k \sum_{j=0}^{k} c^{k-j} \binom{n-j}{k-j} \sum_{\substack{T\subseteq[1,n]\\ |T|=j}} \prod_{m\in T} z_m.$$

Applying (2) yields

$$\begin{aligned} a'_k &= (-1)^k \sum_{j=0}^{k} c^{k-j} (-1)^j a_j \binom{n-j}{k-j} \\ &= \sum_{j=0}^{k} a_j \binom{n-j}{k-j} (-c)^{k-j}. \end{aligned} \tag{19}$$

Note that $a_0 = 1$, in accordance with (2).

Now, set

$$\begin{aligned} (\alpha_1 \dots \alpha_n) &= L\left(\widetilde{T}_c(\mathbf{a}), \widetilde{T}_c(\mathbf{b}), t\right) \\ (\beta_1, \dots, \beta_n) &= \widetilde{T}_c\left(L(\mathbf{a}, \mathbf{b}, t)\right). \end{aligned}$$

Combining the above with (19) and (10) yields

$$\begin{aligned} \alpha_k &= (1-t)a'_k + tb'_k \\ &= (1-t) \sum_{j=0}^{k} a_j \binom{n-j}{k-j} (-c)^{k-j} \\ &\quad + t \sum_{j=0}^{k} b_j \binom{n-j}{k-j} (-c)^{k-j} \\ &= \sum_{j=0}^{k} \binom{n-j}{k-j} (-c)^{k-j} \left((1-t)a_j + tb_j\right) \\ &= \widetilde{T}_c\left((1-t)a_j + tb_j\right) \\ &= \beta_k. \end{aligned}$$

Therefore, (18) holds. ∎

Theorem 2: $\widetilde{SR_c}$ commutes with L, i.e.,

$$L\left(\widetilde{SR_c}(\mathbf{a}), \widetilde{SR_c}(\mathbf{b}), t\right) = \widetilde{SR_c}\left(L(\mathbf{a}, \mathbf{b}, t)\right). \tag{20}$$

Proof: First, set $\mathbf{a} = f(Z)$ and determine $\mathbf{a}' = \widetilde{SR_c}(\mathbf{a})$. We combine (2) and (13) to get

$$\begin{aligned} a'_k &= (-1)^k \sum_{\substack{S\subseteq[1,n]\\ |S|=k}} \prod_{j\in S} z_j c \\ &= (-1)^k \sum_{\substack{S\subseteq[1,n]\\ |S|=k}} c^{|S|} \prod_{j\in S} z_j \\ &= (-1)^k c^k \sum_{\substack{S\subseteq[1,n]\\ |S|=k}} \prod_{j\in S} z_j \\ &= a_k c^k. \end{aligned}$$

Now, set

$$\begin{aligned} (\alpha_1 \dots \alpha_n) &= L\left(\widetilde{SR_c}(\mathbf{a}), \widetilde{SR_c}(\mathbf{b}), t\right) \\ (\beta_1, \dots, \beta_n) &= \widetilde{SR_c}\left(L(\mathbf{a}, \mathbf{b}, t)\right). \end{aligned}$$

Therefore

$$\begin{aligned} \alpha_k &= (1-t)a'_k + tb'_k \\ &= (1-t)a_k c^k + tb_k c^k \\ &= \left((1-t)a_k + tb_k\right) c^k = \beta_k. \end{aligned}$$

Therefore, (20) holds. ∎

Theorem 3: $\widetilde{F}$ commutes with L, i.e.,

$$L\left(\widetilde{F}(\mathbf{a}), \widetilde{F}(\mathbf{b}), t\right) = \widetilde{F}\left(L(\mathbf{a}, \mathbf{b}, t)\right). \tag{21}$$

Proof: First, set $\mathbf{a} = f(Z)$ and determine $\mathbf{a}' = \widetilde{F}(\mathbf{a})$. Since $\overline{a} \cdot \overline{b} = \overline{a \cdot b}$, and $\overline{a} + \overline{b} = \overline{a+b}$ for all complex a and b, and f is only composed of additions and multiplications, $\widetilde{F}(a_1, \dots, a_n) = (\overline{a}_1, \dots, \overline{a}_n)$. Now, set

$$\begin{aligned} (\alpha_1 \dots \alpha_n) &= L\left(\widetilde{F}(\mathbf{a}), \widetilde{F}(\mathbf{b}), t\right) \\ (\beta_1, \dots, \beta_n) &= \widetilde{F}\left(L(\mathbf{a}, \mathbf{b}, t)\right). \end{aligned}$$

Therefore

$$\alpha_k = (1-t)\overline{a}_k + t\overline{b}_k = \overline{(1-t)a_k + tb_k} = \beta_k.$$

Therefore, (21) holds. ∎

APPENDIX IV
PROOF OF COLLISION-DETECTION METHOD

Theorem 4: For any $\mathbf{p}, \mathbf{q}, \mathbf{r}, \mathbf{s} \in \mathbb{R}^n$, $V(\mathbf{p}+\mathbf{q}i, \mathbf{r}+\mathbf{s}i)$ collides with the real axis iff there exists an $x \in \mathbb{R}$ such that x is a root of the polynomial $\Omega(x) = \sum_{m=1}^{2n} \omega_m x^{2n-m}$, where ω_m is given in Fig. 11, and $\tau(x) \in [0,1]$.

Proof: Set $\mathbf{a} = \mathbf{p} + \mathbf{q}i$ and $\mathbf{b} = \mathbf{r} + \mathbf{s}i$. We use the results from Section IV-A.1 to test for collision with the real axis. With point obstacles, z_c was fixed, so we tested z_c in (6) and determined whether t was real and in [0,1]. Here, however, z_c can be any point on the edge, so we search for a z_c that makes t real and in [0,1]. Substituting $x \in \mathbb{R}$ for z_c in (6), we obtain

$$t = \tau(x) = \frac{x^n + \sum_{k=1}^{n} a_k x^{n-k}}{\sum_{k=1}^{n} (a_k - b_k) x^{n-k}}. \tag{22}$$

If there is some x such that $\tau(x)$ is real and between 0 and 1, then $V(\mathbf{a}, \mathbf{b})$ intersects the real axis at x.

To make this determination, first find the set of values for x such that $\tau(x)$ is real. Since (22) is a ratio of two complex numbers, we make the denominator real, and focus on the numerator. We transform (22) as

$$\begin{aligned}
\tau(x) &= \frac{x^n + \sum_{k=1}^{n} a_k x^{n-k}}{\sum_{k=1}^{n} (a_k - b_k) x^{n-k}} \cdot \frac{\overline{\left(\sum_{k=1}^{n} (a_k - b_k) x^{n-k}\right)}}{\overline{\left(\sum_{k=1}^{n} (a_k - b_k) x^{n-k}\right)}} \\
&= \frac{\left(x^n + \sum_{k=1}^{n} a_k x^{n-k}\right)\left(\sum_{k=1}^{n} (\overline{a}_k - \overline{b}_k) x^{n-k}\right)}{\left|\sum_{k=1}^{n} (a_k - b_k) x^{n-k}\right|^2} \\
&= \frac{\sum_{j=1}^{n} (\overline{a}_j - \overline{b}_j) x^{2n-j}}{\left|\sum_{k=1}^{n} (a_k - b_k) x^{n-k}\right|^2} \\
&\quad + \frac{\sum_{j=1}^{n}\sum_{k=1}^{n} a_k (\overline{a}_j - \overline{b}_j) x^{2n-j-k}}{\left|\sum_{k=1}^{n} (a_k - b_k) x^{n-k}\right|^2}.
\end{aligned} \tag{23}$$

Since the denominator is real, $\tau(x)$ is real iff the imaginary component of the numerator is 0. We rearrange terms on the right-hand side (RHS) of (23) by substituting $m = j + k$ and removing k

$$\sum_{j=1}^{n}\sum_{k=1}^{n} a_k (\overline{a}_j - \overline{b}_j) x^{2n-j-k} = \sum_{m=2}^{2n} x^{2n-m} \sum_{j=1}^{n} a_{m-j} (\overline{a}_j - \overline{b}_j). \tag{24}$$

However, the terms on the RHS are only valid if $1 \le m - j \le n$, i.e., $m - n \le j \le m - 1$, while still satisfying $1 \le j \le n$. Therefore, we write (24) as

$$\begin{aligned}
&\sum_{j=1}^{n}\sum_{k=1}^{n} a_k (\overline{a}_j - \overline{b}_j) x^{2n-j-k} \\
&\qquad = \sum_{m=2}^{2n} x^{2n-m} \sum_{j=\max(1,m-n)}^{\min(n,m-1)} a_{m-j} (\overline{a}_j - \overline{b}_j).
\end{aligned}$$

Using this, we can rewrite the numerator of (23) as a polynomial $\sum_{m=1}^{2n} w_m x^{2n-m}$, where

$$w_m = \begin{cases} \overline{a}_m - \overline{b}_m, & m = 1 \\ \overline{a}_m - \overline{b}_m + \sum_{j=1}^{m-1} a_{m-j} (\overline{a}_j - \overline{b}_j), & 2 \le m \le n \\ \sum_{j=m-n}^{n} a_{m-j} (\overline{a}_j - \overline{b}_j), & n+1 \le m \le 2n. \end{cases} \tag{25}$$

To determine $\Im(w_m)$, we use the fact that $\mathbf{a} = \mathbf{p} + \mathbf{q}i$, $\mathbf{b} = \mathbf{r} + \mathbf{s}i$, and $\mathbf{p}, \mathbf{q}, \mathbf{r}, \mathbf{s} \in \mathbb{R}^n$. Therefore

$$\overline{a}_j - \overline{b}_j = (p_j - r_j) - i(s_j - q_j) \tag{26}$$

$$\begin{aligned}
a_j(\overline{a}_k - \overline{b}_k) &= (p_j + iq_j)\left((p_k - r_k) - i(s_k - q_k)\right) \\
&= \left(p_j(p_k - r_k) + q_j(q_k - s_k)\right) \\
&\quad + i\left(p_j(s_k - q_k) + q_j(p_k - r_k)\right).
\end{aligned} \tag{27}$$

We substitute (26) and (27) into (25) to construct $\omega_m = \Im(w_m)$, producing the coefficients in Fig. 11.

The real zeroes of the degree $2n-1$ real polynomial $\Omega(x) = \sum_{m=1}^{2n} \omega_m x^{2n-m}$ are the values for x such that $\tau(x) \in \mathbb{R}$. If $0 < \tau(x_j) < 1$, then $V(\mathbf{a}, \mathbf{b})$ intersects the real axis at x_j. ■

APPENDIX V
PROOFS OF PROPERTIES OF Col

Proposition 1: Col has a lower dimensionality than FS^n.

Proof: Each formation can be represented as a point $\mathbf{a} \in \mathbb{C}$. But collision formations define polynomials with multiple roots. Every formation with a multiple root at the origin can be defined by $\mathbf{a} \in \mathbb{C}$ such that $a_{n-1} = a_n = 0$. Therefore, every collision formation can be described by $\widetilde{T}_c(a_1, \ldots, a_{n-2}, 0, 0)$ for any $a_1, \ldots, a_{n-2}, c \in \mathbb{C}$, where $\widetilde{T}_c$ is the translation operator defined in Section III-C.1. This description is not unique: a formation with two collisions has two descriptions, one for each collision location. Therefore, while each general formation is described with n complex parameters, each collision formation is described with at most $n-1$ nonzero complex parameters. ■

Proposition 2: Col is path-connected.

Proof: Given a formation $\mathbf{a}$ with a collision at c_a, and a formation $\mathbf{b}$ with a collision at c_b, there exists a path from $\mathbf{a}$ to $\mathbf{b}$ such that every point along the path corresponds to a collision formation. Here is a way to construct such a path.

1) Move all the robots at c_a to c_b simultaneously along the same path. They should remain incident to each other at all times while moving along this path.
2) Move all remaining robots in $\mathbf{a}$ to appropriate locations in $\mathbf{b}$. While moving the other robots, the robots at c_b must stay together.
3) If a different number of robots collide at c_a than c_b, then move the appropriate number of robots to or from c_b, until the current formation is $\mathbf{b}$.

Therefore, any two points in Col can be connected in Col, which means Col is path-connected. ■

Proposition 3: Col is closed, and has no interior.

Proof: To show that Col is closed, it suffices to note that each collision-free formation has a collision-free neighborhood. Therefore, $FS^n - \text{Col}$ is open, so Col is closed.

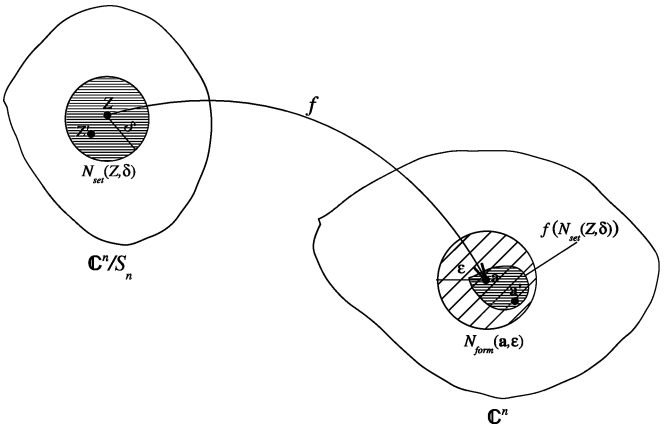


Fig. 26. Neighborhoods of formations and configuration sets.

We define neighborhoods in the formation space to be

$$N_{\text{form}}(\mathbf{a},\epsilon)=\left\{(a'_1,\ldots,a'_n)\mid \forall j\in\{1,\ldots,n\},\left|a_j-a'_j\right|<\epsilon\right\}.$$

Similarly, we define neighborhoods in the space of configurations sets to be

$$N_{\text{set}}(Z,\epsilon)=\{\{z'_1,\ldots,z'_n\}\mid\exists\pi\in S_n \text{ s.t. } \forall j\in\{1,\ldots,n\},\left|z_j-z'_{\pi(j)}\right|<\epsilon\}$$

where S_n is the set of permutations of $\{1,\ldots,n\}$. These are the configuration sets generated by perturbing the points in Z by a distance of at most ϵ.

Consider a collision formation $\mathbf{a}$ and any $\epsilon>0$. Set $Z=f^{-1}(\mathbf{a})$. Since f is continuous, there is a $\delta>0$, defined relative to f, $\mathbf{a}$, and ϵ, such that $f(N_{\text{set}}(Z,\delta))\subseteq N_{\text{form}}(\mathbf{a},\epsilon)$. But any collision configuration set Z can be perturbed by less than δ to produce a noncollision configuration set. Therefore, there exists a collision-free configuration set $Z'\in N_{\text{set}}(Z,\delta)$. Therefore, $\mathbf{a}'=f(Z)$ is a collision-free formation, and $\mathbf{a}'\in N_{\text{form}}(\mathbf{a},\epsilon)$. Therefore, every neighborhood of $\mathbf{a}$ contains a collision-free formation. See Fig. 26.

Therefore, there are no collision formations with neighborhoods contained in Col, i.e., interior points. Therefore, all points in Col are on its boundary. ■

Proposition 4: Col is nowhere dense in FS^n.

Proof: This follows directly from *Propositions 1* and *3*. Since Col is closed, it is its own closure. Since Col has no interior, its closure also has no interior. Since Col has a lower dimensionality than FS^n, it is a strict subset of FS^n. Therefore, it is nowhere dense. ■

ACKNOWLEDGMENT

The authors are grateful to R. Murrieta and S. Bhattacharya for discussions during the early stages of the work, and to the anonymous reviewers, whose comments have significantly improved the content of this paper.

REFERENCES

[1] D. Fox, W. Burgard, H. Kruppa, and S. Thrun, "A probabilistic approach to collaborative multi-robot localization," *Auton. Robots*, vol. 8, no. 3, pp. 325–344, Jun. 2000.

[2] J. Feddema and D. Schoenwald, "Decentralized control of cooperative robotic vehicles," *IEEE Trans. Robot. Autom.*, vol. 18, no. 5, pp. 852–864, Oct. 2002.

[3] J. S. Jennings, G. Whelan, and W. F. Evans, "Cooperative search and rescue with a team of mobile robots," in *Proc. IEEE Int. Conf. Adv. Robot.*, Jul. 7–9, 1997, pp. 193–200.

[4] D. Rus, B. Donald, and J. Jennings, "Moving furniture with teams of autonomous robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Pittsburgh, PA, Aug. 1995, pp. 235–242.

[5] M. Mataric, M. Nilsson, and K. Simsarian, "Cooperative multi-robot box pushing," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Pittsburgh, PA, Aug. 1995, pp. 556–561.

[6] A. Das, R. Fierro, V. Kumar, J. Ostrowski, J. Spletzer, and C. Taylor, "A vision-based formation control framework," *IEEE Trans. Robot. Autom.*, vol. 18, no. 5, pp. 813–825, Oct. 2002.

[7] J. Latombe, *Robot Motion Planning*. Norwell, MA: Kluwer, 1991.

[8] J. Schwartz and M. Sharir, "On the piano movers' problem: III. Coordinating the motion of several independent bodies: The special case of circular bodies moving amidst polygonal barriers," *Robot. Res.*, vol. 2, no. 3, pp. 46–75, 1983.

[9] P. Svestka and M. Overmars, "Coordinated path planning for multiple robots," *Robot. Auton. Syst.*, vol. 23, no. 4, pp. 125–152, 1998.

[10] G. Sanchez and J. Latombe, "Using a PRM planner to compare centralized and decoupled planning for multi-robot systems," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2002, vol. 2, pp. 2112–2119.

[11] Y. Guo and L. E. Parker, "A distributed and optimal motion planning approach for multiple mobile robots," in *Proc. IEEE Int. Conf. Robot. Autom.*, Washington, DC, May 2002, pp. 2612–2619.

[12] T. Simeon, S. Leroy, and J.-P. Laumond, "Path coordination for multiple mobile robots: A resolution complete algorithm," *IEEE Trans. Robot. Autom.*, vol. 18, no. 1, pp. 42–49, Feb. 2002.

[13] S. M. LaValle and S. A. Hutchinson, "Optimal motion planning for multiple robots having independent goals," *IEEE Trans. Robot. Autom.* , vol. 14, no. 6, pp. 912–925, Dec. 1998.

[14] M. Bennewitz, W. Burgard, and S. Thrun, "Finding and optimizing solvable priority schemes for decoupled path planning techniques for teams of mobile robots," *Robot. Auton. Syst.*, vol. 41, no. 2, pp. 89–99, 2002.

[15] J. Peng and S. Akella, "Coordinating multiple robots with kinodynamic constraints along specified paths," *Int. J. Robot. Res.*, vol. 24, pp. 295–310, Apr. 2005.

[16] R. Arkin, *Behavior-Based Robotics*. Cambridge, MA: MIT Press, 1998.

[17] P. Tabuada, G. Pappas, and P. Lima, "Feasible formations of multi-agent systems," in *Proc. Amer. Control Conf.*, Arlington, VA, Jun. 2001, pp. 56–61.

[18] K. H. Tan and M. A. Lewis, "Virtual structures for high precision cooperative mobile robot control," *Auton. Robots*, vol. 4, pp. 387–403, Oct. 1997.

[19] D. Bendersky and J. Santos, "Robot formations as an emergent collective task using target-following behavior," in *Proc. 4th Argentine Symp. Artif. Intell.*, Santa Fe, Argentina, Sep. 2002, CD-ROM.

[20] T. Balch and R. C. Arkin, "Behavior-based formation control for multirobot teams," *IEEE Trans. Robot. Autom.*, vol. 14, no. 6, pp. 926–939, Dec. 1998.

[21] J. Fredslund and M. Mataric, "A general algorithm for robot formations using local sensing and minimal communication," *IEEE Trans. Robot. Autom.*, vol. 18, no. 5, pp. 837–846, Oct. 2002.

[22] V. Gazi, "Swarm aggregations using artificial potentials and sliding-mode control regions," *IEEE Trans. Robot.*, vol. 21, no. 6, pp. 1208–1214, Dec. 2005.

[23] H. G. Tanner, G. J. Pappas, and V. Kumar, "Leader-to-formation stability," *IEEE Trans. Robot. Autom.*, vol. 20, no. 3, pp. 443–455, Jun. 2004.

[24] P. Tabuada, G. J. Pappas, and P. Lima, "Motion feasibility of multiagent formations," *IEEE Trans. Robot.*, vol. 21, no. 3, pp. 387–392, Jun. 2005.

[25] P. Ogren, E. Fiorelli, and N. Leonard, "Formations with a mission: Stable coordination of vehicle group maneuvers," in *Proc. 15th Int. Symp. Math. Theory Netw. Syst.*, Notre Dame, IN, Aug. 2002, CD-ROM.

[26] D. W. Gage, "Command control for many-robot systems," in *Proc. 19th Annu. AUVS Tech. Symp.*, Hunstville, AL, Jun. 1992, pp. 22–24.

[27] S. Poduri and G. Sukhatme, "Constrained coverage for mobile sensor networks," in *Proc. IEEE Int. Conf. Robot. Autom.*, New Orleans, LA, Apr. 2004, vol. 1, pp. 165–171.

[28] A. Howard, M. Mataric, and G. Sukhatme, "An incremental deployment algorithm for mobile robot teams," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2002, vol. 3, pp. 2849–2854.

[29] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *Proc. IEEE Int. Symp. Comput. Intell. Robot. Autom.*, Jul. 1997, pp. 146–151.
[30] Q. Du, V. Faber, and M. Gunzburger, "Centroidal Voronoi tessellations: Applications and algorithms," *SIAM Rev.*, vol. 41, no. 4, pp. 637–676, 1999.
[31] A. Okabe and A. Suzuki, "Locational optimization problems solved through Voronoi diagrams," *Euro. J. Oper. Res.*, vol. 1998, pp. 445–456, 1997.
[32] J. Cortes, S. Martinez, and T. K. T. F. Bullo, "Coverage control for mobile sensing networks," *IEEE Trans. Robot. Autom.*, vol. 20, no. 2, pp. 243–255, Apr. 2004.
[33] D. Popa, H. Stephanou, C. Helm, and A. Sanderson, "Robotic deployment of sensor networks using potential fields," in *Proc. IEEE Int. Conf. Robot. Autom.*, New Orleans, LA, Apr. 2004, vol. 1, pp. 642–647.
[34] T. Clouqueur, V. Phipatanasuphorn, P. Ramanathan, and K. K. Saluja, "Sensor deployment strategy for target detection," in *Proc. 1st ACM Int. Workshop Wireless Sensor Netw. Appl.*, Sep. 2002, pp. 42–48.
[35] E. Acar, Y. Zhang, H. Choset, M. Schervish, A. Costa, R. Melamud, D. Lean, and A. Graveline, "Path planning for robotic demining and development of a test platform," in *Proc. Int. Conf. Field Service Robot.*, 2001, pp. 161–168.
[36] G. Schmidt and C. Hofner, "An advanced planning and navigation approach for autonomous cleaning robot operations," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 1998, vol. 2, pp. 1230–1235.
[37] D. Kurabayashi, J. Ota, T. Arai, and E. Yoshida, "Cooperative sweeping by multiple mobile robots," in *Proc. IEEE Int. Conf. Robot. Autom.*, Apr. 1996, vol. 2, pp. 1744–1749.
[38] I. Rekleitis, G. Dudek, and E. Milios, "Multi-robot collaboration for robust exploration," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2000, vol. 4, pp. 3164–3169.
[39] I. Rekleitis, V. Lee-Shue, A. P. New, and H. Choset, "Limited communication, multi-robot team based coverage," in *Proc. IEEE Int. Conf. Robot. Autom.*, New Orleans, LA, Apr. 2004, vol. 4, pp. 3462–3468.
[40] I. Wagner, M. Lindenbaum, and A. Bruckstein, "Distributed covering by ant-robots using evaporating traces," *IEEE Trans. Robot. Autom.*, vol. 15, no. 5, pp. 918–933, Oct. 1999.
[41] W. Burgard, M. Moors, C. Stachniss, and F. E. Schneider, "Coordinated multi-robot exploration," *IEEE Trans. Robot.*, vol. 21, no. 3, pp. 376–386, Jun. 2005.
[42] S. Kloder, S. Bhattacharya, and S. Hutchinson, "A configuration space for permutation-invariant multi-robot formations," in *Proc. IEEE Int. Conf. Robot. Autom.*, New Orleans, LA, Apr. 2004, vol. 3, pp. 2746–2751.
[43] S. Kloder and S. Hutchinson, "Path planning for permutation-invariant multi-robot formations," in *Proc. IEEE Int. Conf. Robot. Autom.*, Barcelona, Spain, Apr. 2005, CD-ROM.
[44] C. F. Gauss, "Beitrage zur theorie der algebraischen gleichungen," *Abh. Ges. Wiss. Gottingen* , vol. 4 (1850), *Ges. Werke* vol. 3, pp. 73–102.
[45] T. Motzkin and A. Ostrowski, "Uber den fundamentalsalz der algabra," *Sitzb. Preuss. Akad. Wiss. Phys. Math. Klasse*, pp. 1–4, 1933.
[46] M. Marden, *The Geometry of the Zeros of a Polynomial in a Complex Variable*. New York: Amer. Math. Soc., 1949.
[47] L. Kavraki, P. Svestka, J. C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robot. Autom.*, vol. 12, no. 4, pp. 556–580, Aug. 1996.
[48] V. Kharitonov, "Asymptotic stability of an equilibrium position of a family of systems of linear differential equations," *Differen. Equat.*, vol. 14, pp. 1483–1485, 1979.
[49] A. C. Bartlett, C. V. Hollot, and L. Huang, "Root location of an entire polytope of polynomials: It suffices to check the edges," *Math. Contr., Signals Syst.*, vol. 1, pp. 61–71, 1988.
[50] J. Kogan, "Robust Hurwitz l^p stability of polynomials with complex coefficients," *IEEE Trans. Autom. Control*, vol. 38, no. 8, pp. 1304–1308, Aug. 1993.
[51] A. Katbab and E. Jury, "A note on two methods related to stability robustness of polynomials in a sector (relative stability)," *IEEE Trans. Autom. Control*, vol. 38, no. 2, pp. 380–383, Feb. 1993.
[52] M. Bozorg and E. M. Nebot, "Comments on 'A note on two methods related to stability robustness of polynomials in a sector (relative stability)'," *IEEE Trans. Autom. Control*, vol. 42, no. 5, pp. 712–714, May 1997.

Stephen Kloder (S'05) received the B.S. degree in computer science in 2001 from the Georgia Institute of Technology, Atlanta, and the M.S. degree in 2004 from the University of Illinois at Urbana-Champaign (UIUC), where he is currently working toward the Ph.D. degree.

His interests include multirobot planning and applications. He has done work on the building of formations, and is currently working on multirobot coverage problems.

Seth Hutchinson (SM'00) received the Ph.D. degree from Purdue University, West Lafayette, IN, in 1988.

In 1990, he joined the faculty of the University of Illinois at Urbana-Champaign, where he is currently a Professor with the Department of Electrical and Computer Engineering, the Coordinated Science Laboratory, and the Beckman Institute for Advanced Science and Technology. He has published more than 100 papers on the topics of robotics and computer vision, and is coauthor of *Principles of Robot Motion: Theory, Algorithms, and Implementations* (Cambridge, MA: MIT Press), and *Robot Modeling and Control* (New York: Wiley).

Dr. Hutchinson serves on the editorial boards of the *International Journal of Robotics Research* and the *Journal of Intelligent Service Robotics*. He served as Associate Editor and then Senior Editor for the IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION, now the IEEE TRANSACTIONS ON ROBOTICS, from 1997 to 2005. In 1996, he was a Guest Editor for a Special Section of the TRANSACTIONS devoted to the topic of visual servo control, and in 1994 he was Co-Chair of an IEEE Workshop on Visual Servoing. In 1996 and 1998, he coauthored papers that were finalists for the King-Sun Fu Memorial Best Transactions Paper Award. He was Co-Chair of the IEEE Robotics and Automation Society Technical Committee on Computer and Robot Vision from 1992 to 1996, and has served on the program committees for more than 50 conferences related to robotics and computer vision.