Proceedings of the 2004 IEEE
International Conference on Robotics & Automation
New Orleans, LA • April 2004

A Configuration Space for Permutation-Invariant Multi-robot Formations

Stephen Kloder Sourabh Bhattacharya Seth Hutchinson
The Beckman Institute for Advanced Science and Technology
University of Illinois at Urbana-Champaign
Urbana, IL 61801
Email: {kloder, sbhattac, seth}@uiuc.edu

***Abstract*— In this paper we describe a new representation for a configuration space for formations of robots that translate in the plane. What makes this representation unique is that it is permutation-invariant, so the relabeling of robots does not affect the configuration. Earlier methods generally either pre-assign roles for each individual robot, or rely on local planning and behaviors to build emergent behaviors. Our method first plans the formation as a set, and only afterwards determines which robot takes which role.**

To build our representation of this formation space, we make use of a property of complex polynomials: they are unchanged by permutations of their roots. Thus we build a characteristic polynomial whose roots are the robot locations, and use its coefficients as a representation. Mappings between work spaces and formation spaces amount to building and solving polynomials.

In this paper we also perform basic path planning on this new representation, and show some practical and theoretical properties. We show that the paths generated are invariant – relative to their endpoints – with respect to linear coordinate transforms, and in most cases produce reasonable, if not linear, paths from start to finish.

I. Introduction

Multi-robot planning is an interesting and developing subfield of robot planning. Multiple coordinated robots have been used successfully in applications such as localization and exploration [1], surveillance and monitoring [2], search and rescue [3], satellite arrangement [4], and object manipulation and transportation [5],[6],[7].

We are particularly interested in one aspect of multi-robot planning: formations. The problem of robot formations is to move a group of robots into a predetermined shape, and then manipulate the shape accordingly. By treating a group of robots as a single formation, planning motion becomes planning on a single global entity, rather than the sum of individual robot plans. This is especially true for simple operations: it is much easier to rotate or scale a formation than to plan each robot's motion accordingly.

Previous approaches to formation planning have included planning individual trajctories for each robot in the formation [4],[8]; defining virtual structures in which subgroupwise constraints are imposed on the relative motions of robots [7],[9],[10],[11]; and using local beviours that give rise to emergent group behaviour [12],[13].

The first two methods, based on predefined roles and structures, are inflexible to changing roles and relationships among the multiple robots. Behavior-based methods can have the same restriction; the ones that are not based on such structures are significantly less powerful and can generally only create symmetrical or repetitive formations. This is adequate in the case of a single formation pattern plus general manipulations (scaling, rotation, etc.), or a limited formation set, but not in the general case, where formations can have any shape, any relationships, and any role assignments, all of which can change throughout the situation. We propose a new way of representing and planning robot formations that differs distinctly from the above. It will be designed to accommodate any formation, but without predefining roles and relationships.

Our representation will be a formation manifold comparable to configuration spaces. The space will be a manifold over unlabeled robot formations. The robots are unlabeled so the formations will be permutation-invariant, i.e. exchanging two robots does not change the formation. Previous methods required each robot to move only to its assigned role in the formation; our new representation allows the planner to find the optimal assignments for each case. While one could use a labeled robot formation space, plan for each possible formation permutation, and then select the optimal path; n robots have $n!$ formations, so such a method would not scale well. Once the new formation is reached, preexisting methods such as [14] can be applied to translate, rotate, or scale it.

This paper presents a representation for a particular domain: robots translating in the plane. We show that it is possible to build a representation for formations of arbitrary size and with the desired properties enumerated above. We also give a method for path planning in this configuration space that effectively plans in situations where there are no obstacles.

This paper is laid out as follows: section 2 describes the representation, section 3 describes the path planning algorithm as well as its properties, section 4 describes the results, section 5 describes collision properties, section 6 describes control issues, and section 7 describes the proposed additional work for this representation.

II. Representation

The configuration space of a labelled multirobot formation is generally an ordered list of configurations, one for each robot. However, we are building a configuration space for *unlabelled* formations, where exchanging two robots does not change the configuration. We call this a *formation space*, and write $FS[X]^n = X^n/S_n$, where n is the number of robots, X is the configuration space of one robot, and 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 iff they are permutations of each other. We shall sometimes refer to $FS[X]^n$ as FS^n for short if X is known. If X^n is a manifold, then clearly FS^n is also a manifold, as its neighborhoods are topologically equivalent to the neighborhoods in X^n. However, what kind of manifold is it? And can it be represented in a continuous manner? Simply sorting the individual robot configurations by a well-defined full ordering would uniquely represent every possible configuration, but would do so with discontinuities at points where robots switch places in the ordering.

Therefore what is needed is a homeomorphism $f : FS[X]^n \to M$, where M is some manifold with known properties. This can alternatively be viewed as $f : X^n \to M$ where f is not a bijection, but is 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 $x_1, x_2, \ldots, x_n$).

Below is a representation of the formation space for robots in the plane. We can represent a location in the plane as $(x, y) \in \mathbb{R}^2$. We can also use the complex plane $\mathbb{C}$ to represent these locations, since $\mathbb{R}^2 \cong \mathbb{C}$. Any point $(x, y) \in \mathbb{R}^2$ can be represented as $z = x + iy \in \mathbb{C}$. In this paper, we use $\mathbb{C}$ so that we can 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 define $FS[\mathbb{C}]^n$ for any $n \in \mathbb{N}$. The representation we will define is not completely closed form, but it is well defined and fits the above criteria.

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.

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

$$a_1 = -(z_1 + z_2 + \cdots + z_n) = -\sum_{j=1}^{n} z_j, \tag{2}$$

$$a_2 = \sum_{j=1}^{n}\sum_{k=1}^{j-1} z_j z_k \quad , \cdots, \tag{3}$$

$$a_n = (-1)^n \prod_{j=1}^{n} z_j \tag{4}$$

$$a_k = (-1)^k \sum_{\substack{S\subseteq[1,n] \\ |S|=k}} \prod_{j\in S} z_j \tag{5}$$

$(a_1, \ldots, a_n) \in \mathbb{C}^n$ constitutes a formation space for $(z_1, \ldots, z_n) \in FS^n$. $f(z_1, \ldots, z_n) = (a_1, \ldots, a_n)$ defined above, and $f^{-1}(a_1, \ldots, a_n)$ is defined as the roots of $\lambda^n + \sum_{j=1}^{n} a_j \lambda^{n-j}$. f is a homeomorphism because of what is already known about complex polynomials: A polynomial of degree n has exactly n roots (counting multiple roots multiple times), every polynomial has a unique factorization, and the mapping from polynomial coefficients to roots is continuous. Therefore $f : FS[\mathbb{C}]^n \to \mathbb{C}^n$ is a homeomorphism, so $FS[\mathbb{C}]^n \cong \mathbb{C}^n$. 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. Planning

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. But in this 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 it must be broken down into discrete steps, which can individually mapped back into the workspace. These discrete workspace configurations can then be reconnected to produce a path in the workspace.

This section deals with the simplest formation space planning algorithm: the straight line planner.

A. The Basic Planning Algorithm

The basic method is to follow a straight line from start to finish in formation space, and follow it linearly. Therefore to plan from $(a_1, \ldots, a_n)$ to $(a'_1, \ldots, a'_n)$, follow the path $p(t) = (1-t)(a_1, \ldots, a_n) + t(a'_1, \ldots, a'_n)$. Since, as stated above, it is impossible to determine the trajectory directly, we need to discretize the path, and then connect the resulting configirations. We take a sampling of the interval [0,1] by taking N values: $0 = t_1 < t_2 < \cdots < t_N = 1$. Calculating $f^{-1}(p(t_i))$ for each t_i produces the locations of the robots at each time. To produce a path, we connect $f^{-1}(p(t_i))$ to $f^{-1}(p(t_{i+1}))$ for each t_i.

To connect $(z_1, \ldots, z_n) = f^{-1}(p(t_i))$ to $(z'_1, \ldots, z'_n) = f^{-1}(p(t_{i+1}))$, we need to determine which z_i moved to which z'_j. Define $closest(z_i) = \arg\min_{z'_j} \left|z_i - z'_j\right|^2$. Calculate $closest(z_i)$ for each z_i. If every value of $closest$ is unique, then connect each z_i to $closest(z_i)$ with a straight-line path.

However, If $\exists i \neq j$ such that $closest(z_i) = closest(z_j)$, then there is an ambiguity. In this case select a new $t'_i \in (t_i, t_{i+1})$, (usually $t'_i = \frac{t_i + t_{i+1}}{2}$) and use the above method to connect $f^{-1}(p(t_i))$ to $f^{-1}(p(t_{i+1}))$ and $f^{-1}(p(t_i))$

to $f^{-1}(p(t_{i+1}))$. This is a recursive procedure; it may be necessary to set $t_i'' \in (t_i, t_i')$ or $t_i''' \in (t_i', t_{i+1})$, etc. many times until there are no ambiguities. If $|z_i - closest(z_i)|^2$ is too large (above a preset threshold), this can also be considered an ambiguity and require a new t_i'.

This algorithm produces a set of distinct paths out of unlabeled point sets, and allows for variable-density sampling to reflect the variable nature of the resulting path.

B. 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 that connects two configurations $\mathbf{a} = (a_1, \ldots, a_n)$ and $\mathbf{a}' = (a_1', \ldots, a_n')$. We denote the path by $L(\mathbf{a}, \mathbf{a}')$, where

$$L(\mathbf{a}, \mathbf{a}') = (1-t)(a_1, \ldots, a_n) + t(a_1', \ldots, a_n') \quad (6)$$

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

1) Translation: Define the translational operator

$$T_c(z_1, \ldots, z_n) = (z_1 + c, z_2 + c, \ldots, z_n + c). \quad (7)$$

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 aplied to the roots) on the coefficients of the polynomial, i.e. if $\mathbf{a} = f(\mathbf{z})$, then $\widetilde{T}_c(\mathbf{a}) = f(T_c(\mathbf{z}))$.

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

$$L\left(\widetilde{T}_c(\mathbf{a}), \widetilde{T}_c(\mathbf{a}')\right) = \widetilde{T}_c(L(\mathbf{a}, \mathbf{a}')). \quad (8)$$

This means that if the initial and goal configurations both translate by the same amount, the resulting path will translate accordingly.

Proof: First, set $\mathbf{a} = f(\mathbf{z})$ and determine $\mathbf{a}' = \widetilde{T}_c(\mathbf{a})$.

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

$$= \sum_{\substack{T \subseteq [1,n] \\ |T| \leq k}} \left(c^{k-|T|} \prod_{j \in T} z_j\right) \cdot |\{S \subseteq [1,n] \mid T \subseteq S, |S| = k\}|$$

$$= \sum_{\substack{T \subseteq [1,n] \\ |T| \leq k}} \binom{n-|T|}{k-|T|} c^{k-|T|} \prod_{j \in T} z_j$$

$$= \sum_{j=1}^{k} c^{k-j} \binom{n-j}{k-j} \sum_{\substack{T \subseteq [1,n] \\ |T|=j}} \prod_{m \in T} z_m = \sum_{j=1}^{k} c^{k-j} (-1)^j a_j \binom{n-j}{k-j}.$$

Therefore $a_k' = \sum_{j=0}^{k} a_j \binom{n-j}{k-j} (-c)^{k-j}$, where $a_0 = 1$.

Now, set:

$$(\alpha_1 \ldots \alpha_n) = L\left(\widetilde{T}_c(\mathbf{a}), \widetilde{T}_c(\mathbf{a}')\right) \quad (9)$$

$$(\beta_1, \ldots, \beta_n) = \widetilde{T}_c(L(\mathbf{a}, \mathbf{a}')). \quad (10)$$

Therefore,

$$\begin{aligned} \alpha_k &= (1-t)a_k' + tb_k' && (11) \\ &= (1-t)\sum_{j=0}^{k} a_j \binom{n-j}{k-j} (-c)^{k-j} \\ &\quad + t\sum_{j=0}^{k} b_j \binom{n-j}{k-j} (-c)^{k-j} && (12) \\ &= \sum_{j=0}^{k} \binom{n-j}{k-j} (-c)^{k-j} ((1-t)a_j + tb_j) && (13) \\ &= \beta_k && (14) \end{aligned}$$

Therefore (8) holds. ■

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 $c = m \cdot d$, where m is an integer and d has absolute value 1 (set $m = |c|$, and $d = c/m$). Multiplying by m is a scale operation, and multiplying by d is a rotation. Therefore multiplying by c does both. Define the scale/rotate operator

$$SR_c(z_1, \ldots, z_n) = (z_1c, z_2c, \ldots, z_nc). \quad (15)$$

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

Lemma 1: $\widetilde{SR}_c$ commutes with L, i.e.

$$L\left(\widetilde{SR}_c(\mathbf{a}), \widetilde{SR}_c(\mathbf{a}')\right) = \widetilde{SR}_c(L(\mathbf{a}, \mathbf{a}')). \quad (16)$$

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.

Proof: First, set $\mathbf{a} = f(\mathbf{z})$ and determine $\mathbf{a}' = \widetilde{SR}_c(\mathbf{a})$.

$$\begin{aligned} a_k' &= (-1)^k \sum_{\substack{S \subseteq [1,n] \\ |S|=k}} \prod_{j \in S} z_j c && (17) \\ &= (-1)^k \sum_{\substack{S \subseteq [1,n] \\ |S|=k}} c^{|S|} \prod_{j \in S} z_j && (18) \\ &= (-1)^k c^k \sum_{\substack{S \subseteq [1,n] \\ |S|=k}} \prod_{j \in S} z_j && (19) \\ &= a_k c^k && (20) \end{aligned}$$

Now, set:

$$\begin{aligned}(\alpha_1 \ldots \alpha_n) &= L\left(\widetilde{SR}_c(\mathbf{a}), \widetilde{SR}_c(\mathbf{z}')\right) & (21)\\ (\beta_1, \ldots, \beta_n) &= \widetilde{SR}_c(L(\mathbf{a}, \mathbf{a}')). & (22)\end{aligned}$$

Therefore,

$$\begin{aligned}\alpha_k &= (1-t)\, a'_k + t b'_k & (23)\\ &= (1-t)\, a_k c^k + t b_k c^k & (24)\\ &= ((1-t)\, a_k + t b_k)\, c^k = \beta_k & (25)\end{aligned}$$

Therefore (16) holds. ∎

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 origin of the coordinate frame is irrelevant, as is the unit size, when doing straight-line planning.

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

$$F(z_1, \ldots, z_n) = (\overline{z_1}, \ldots, \overline{z_n}), \qquad (26)$$

where $\overline{z}$ is the complex conjugate of z. F reflects all robots across the x axis. From F define the lifted reflection operator $\widetilde{F} = fFf^{-1}$, with the same reasoning as for $\widetilde{T}_c$ and $\widetilde{SR}_c$ above.

Lemma 1: $\widetilde{F}$ commutes with L, i.e.

$$L\left(\widetilde{F}(\mathbf{a}), \widetilde{F}(\mathbf{a}')\right) = \widetilde{F}(L(\mathbf{a}, \mathbf{a}')). \qquad (27)$$

This means that if the initial and goal configurations are both reflected about the x-axis, the resulting path will also be reflected about the x-axis.

Proof: First, set $\mathbf{a} = \underline{f(\mathbf{z})}$ 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, $\widetilde{F}(a_1, \ldots, a_n) = (\overline{a_1}, \ldots, \overline{a_n})$. Now, set:

$$\begin{aligned}(\alpha_1 \ldots \alpha_n) &= L\left(\widetilde{F}(\mathbf{a}), \widetilde{F}(\mathbf{a}'\prime)\right) & (28)\\ (\beta_1, \ldots, \beta_n) &= \widetilde{F}(L(\mathbf{a}, \mathbf{a}')). & (29)\end{aligned}$$

Therefore,

$$\alpha_k = (1-t)\,\overline{a_k} + t\overline{b_k} = \overline{(1-t)\, a_k + t b_k} = \beta_k. \qquad (30)$$

Therefore (27) holds. ∎

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.

IV. Results

Simulating the straight line path planning method has produced some interesting results. The algorithm has been run on a variety of arbitrary start and goal configurations for a varying number of formation sizes. The resulting paths are generally simple and straightforward. In many cases they are close to optimal. The paths never intersect themselves or other paths, with the exception of certain degenerate cases, described below. The results also scale well: planning 50 robots does not take significantly longer than planning 2 robots.

The diagrams on the next page show some typical results. Often the resulting paths are nice, simple, and almost direct. Fig. 1 and 2 show two examples of this. But often the resulitng paths are convoluted and roundabout. Fig. 3 shows many robots following convoluted paths, and some moving directly away from their eventual goals. Fig. 4 shows a more extreme example of this is: one where the start and goal configurations are each very close together, but the ends are very far from each other. In this case, some robots move straight to a goal, but many robots take unusually roundabout routes, moving as far away from the other robots before moving in toward the goal. In cases like these it may be better to translate the goal to a configuration very close to the start, plan to this new goal, then translate the formation to the original goal.

Although some paths appear to collide in these figures, they are just very close to each other. The only way we've been able to generate a collision is by contriving the initial and goal configurations specifically to generate a collision.

V. Characterizing Collisions

Although we have not been able to generate collisions by picking arbitrary initial and goal configurations, there are ways to explicitly generate 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 that all collisions conform to.

Define a *generalized collision path* to be a path that contains a collision 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 point that corresponds to a collision. 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.

First define a *collision point* $\mathbf{a} = (a_1, \ldots, a_n) \in \mathbb{C}^n$ such that $\lambda^n + \sum_{j=1}^{n} a_j \lambda^{n-j}$ has at least one multiple root, which corresponds to a collision. Any collision paths involving this a are straight lines in the formation space through a, and can be written as $L(\mathbf{a} - \mathbf{d}, \mathbf{a} + k\mathbf{d})$, where L is from (6), $\mathbf{d} = (d_1, \ldots, d_n) \in \mathbb{C}^n$, and $k \in \mathbb{R}^+$. Therefore every nontrivial collision path is uniquely defined by a single $(\mathbf{a}, \mathbf{d}, k) \in Col \times \mathbb{C}^n \times \mathbb{R}^+$, where Col is the set of all collision points.

These specific cases are the only configurations that generate collisions. Our experiments have shown that even configurations that approximate these don't generate collisions. It is highly unlikely that a random selection of start and goal configurations will lead to a collision path using the straight line method.

One interesting example of this is the *alternating regular polygon*. As its name suggests, the alternating regular polygon is a regular polygon that alternates between start and goal

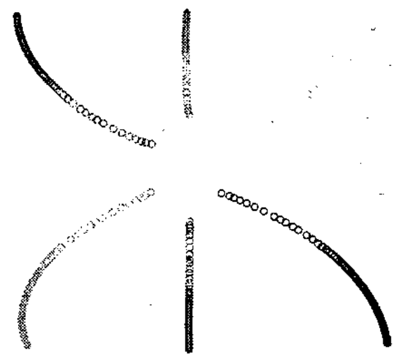
Fig. 1. 6 robots, "nice" paths

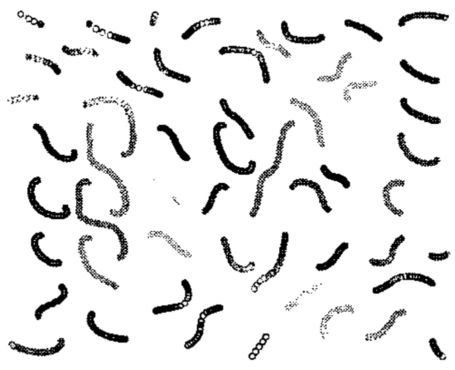
Fig. 3. 50 robots, convoluted paths

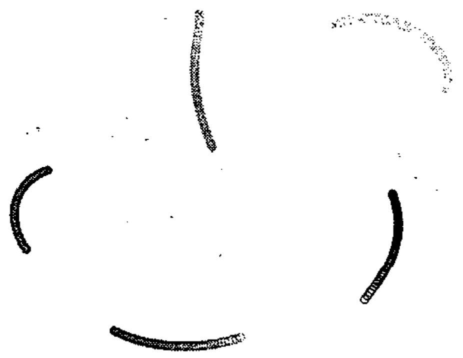
Fig. 2. 5 robots, "nice" paths

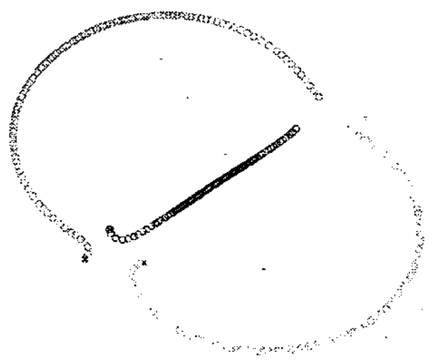
Fig. 4. 3 robots, roundabout paths

positions. The robots start on the odd positions and finish in the even ones. An alternating regular polygon centered at the origin with radius 1 and one point on the positive x axis can be generated by setting $a = (0, 0, \ldots, 0)$, $d = (0, \ldots, 0, -1)$, and $k = 1$. Applying the $\widetilde{T}$ and $\widetilde{SR}$ operators produces an alternating regular polygon in any location, size, or orientation. Allowing k to have a value other than one produces a *generalized alternating regular polygon*, where the start and goal polygons can be different sizes.

VI. Coefficient Control

In a more general sense, we can control the robot movements by controlling the coefficients of the polynomial. This leads to the notion of "coefficient control". Let us analyze how the roots of the polynomials change as we give small changes in the value of the coefficients.

Taking partial derivatives on both the sides of Equation 1 from section II with respect to a_k and applying chain rule to the R.H.S:

$$\frac{\partial P(\lambda)}{\partial a_k}|_{\lambda=z_i} = \frac{\partial P(\lambda)}{\partial \lambda}|_{\lambda=z_i}\frac{\partial z_i}{\partial a_k} \tag{31}$$

$$\frac{\partial z_i}{\partial a_k} = \frac{\frac{\partial P(\lambda)}{\partial a_k}|_{\lambda=z_i}}{\frac{\partial P(\lambda)}{\partial \lambda}|_{\lambda=z_i}} \tag{32}$$

$$= \frac{\lambda^{n-k}|_{\lambda=z_i}}{\Pi_{j=1,j\neq i}^{n}(z_i - z_j)} \tag{33}$$

$$= \frac{z_i^{n-k}|_{\lambda=z_i}}{\Pi_{j=1,j\neq i}^{n}(z_i - z_j)} \tag{34}$$

Hence

$$\Delta z_i = \sum_{k=1}^{n}\frac{\partial z_i}{\partial a_k}\Delta a_k \tag{35}$$

$$\Delta z_i = \frac{\sum_{k=1}^{n} z_i^{n-k}\Delta a_k}{\Pi_{j=1,j\neq i}^{n}(z_i - z_j)} \tag{36}$$

From (36) we can observe some points given below

- For a given robot if the surrounding robots are quite nearby, a small change in the coefficient leads to a large change in the robot position.
- Due to the unequal weighting of the coefficients in the equation, the same change in different coefficients will lead to different changes in the position of the robot. Hence if z_i is large in magnitude, then the change in coefficients with smaller subscripts largely control the change in the position of the roots and hence the robots.
- An interesting event occurs if the robots collide. Then the denominator of (36) becomes zero. Hence the change in robot position seems to be independent of the amount

of change applied to the coefficients, except those coefficients that remain unchanged. Hence there is a notion of loss of control in this case.

Hence the path-planning problem in multirobot systems with coefficient control can be modelled as a matrix differential equation(MDE) given by

$$\dot{R}(t) = T(R(t))\dot{A}(t) \tag{37}$$

where,

$R(t) = [z_1(t), z_2(t), ...z_n(t)]^T$ is the $n \times 1$ vector of robot positions, $T(R(t))$ is a non-linear transformation on $R(t)$ given by the $n \times n$ matrix $T(k,j) = \frac{z_k^{n-j}(t)}{\Pi_{m=1,m\neq k}^{n}(z_k(t)-z_m(t))}$ and $A(t) = [a_1(t), a_2(t), ...a_n(t)]^T$ is the $n \times 1$ vector of coefficients. The boundary conditions of the matrix differential equation are given by the initial $(R(0))$ and final $(R(1))$ robot positions.

The problem of optimization of distances in the workspace can be addressed using the MDE. For instance if it is required to minimize the L^2 norm of the distances then the problem can be addressed as the minimization of $\int_0^1 [\dot{R}(t)^*]^T \dot{R}(t)dt$. The terms inside the integral can be related to the coefficient control terms(a_k) by the MDE.

In the path planning method we noted that sometimes connecting configurations in two timesteps is ambiguous, and requires additional timesteps. By using the coefficient control we can determine what is the maximum change in the coefficients allowed in a time step so that the disambiguity condition is satisfied.

VII. Conclusion and Further Work

In this work we have built a continuous configuration space representation for formations of unlabeled robots that translate in the plane. We have shown that the space is homeomorphic to the configuration space for labeled formations, and that the representation is both invertible and permutation invariant. We have built a method that plans in this configuration space. We have shown that it has convenient algebraic properties, namely that it is invariant to linear coordinate transforms.

But there is much that remains to be done, as this is a very specific subset of the problem of unlabeled formations. The paths produced are reasonable, but not optimal. We would like to generate more efficient paths, and remove problems with extreme cases. The current formations are very ideal: they represent holonomic robots with zero size and no obstacles. We would like to place constraints on these robots, starting with physical obstacles and nonzero robot sizes, and see how this representation accommodates them. The current system is for centralized planning with certain knowledge; we would like to see whether uncertainty and decentralized planning can be reasonably accommodated. Lastly, the formations are only for a single type of robot. Surely there are other comparable methods for representing formation space X^n/S_n for any configuration space X.

Acknowledgments

The authors would like to thank Robert Ghrist for his help with some of the pure mathematics concepts.

References

[1] D. Fox, W. Burgard, H. Kruppa, and S. Thrun, "A probabilistic approach to collaborative multi-robot localization," *Autonomous Robots*, vol. 8, no. 3, pp. 325–344, June 2000. [Online]. Available: citeseer.nj.nec.com/fox00probabilistic.html

[2] J. Feddema and D. Schoenwald, "Decentralized control of cooperative robotic vehicles," *IEEE Trans. Robot. Automat.*, 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 ICAR Int. Conf. Advanced Robotics, 1997*, July7-9 1997, pp. 193–200.

[4] W. Kang, N. Xi, and A. Sparks, "Formation control of autonomous agents in 3d workspace," in *Proc. IEEE ICRA '00. Int. Conf. Robotics and Automation 2000*, vol. 2, San Francisco, CA, Apr. 2000, pp. 1755–1760.

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

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

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

[8] P. Ogren, M. Egerstedt, and X. Hu, "A general algorithm for robot formations using local sensing and minimal communication," *IEEE Trans. Robot. Automat.*, vol. 18, no. 5, pp. 847–851, Oct. 2002.

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

[10] P. Tabuada, G. Pappas, and P. Lima, "Feasible formations of multi-agent systems," Arlington, VA, pp. 56–61, June 2001. [Online]. Available: citeseer.nj.nec.com/tabuada01feasible.html

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

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

[13] P. Ogren, E. Fiorelli, and N. Leonard, "Formations with a mission: Stable coordination of vehicle group maneuvers," 2002. [Online]. Available: citeseer.nj.nec.com/ogren02formations.html

[14] J. Lawton, B. Young, and R. Beard, "A decentralized approach to elementary formation manuevers," in *Proc. IEEE ICRA '00. Int. Conf. Robotics and Automation 2000*, vol. 3, San Francisco, CA, Apr. 2000, pp. 2728–2733.