Decentralized Feedback Controllers for Multiagent Teams in Environments With Obstacles

Nora Ayanian, *Student Member, IEEE*, and Vijay Kumar, *Fellow, IEEE*

***Abstract*—We propose a method for synthesizing decentralized feedback controllers for a team of multiple heterogeneous agents navigating a known environment with obstacles. The controllers are designed to drive agents with limited team state information to goal sets while avoiding collisions and maintaining specified proximity constraints. The method, its successful application to nonholonomic agents in dynamic simulation and experimentation, and its limitations are presented in this paper.**

***Index Terms*—Decentralized feedback control, mobile robot motion planning, motion control, path planning for multiple mobile robot systems.**

I. Introduction

IN MANY scenarios, it is necessary to guide multiple vehicles to destinations or goal sets in an environment with obstacles while avoiding collisions and maintaining proximity constraints, either for communication or for sensing. It is important that controller synthesis for these scenarios not require expert knowledge so that they can be applied in many settings and for many problems.

In this paper, we consider the problem of synthesizing feedback controllers for a team of multiple heterogeneous agents navigating a known environment with obstacles. We want controllers that are guaranteed to drive agents with limited communication to goal sets while avoiding collisions and maintaining specified proximity constraints. We are particularly interested in guiding a heterogeneous team of aerial and ground vehicles in an urban environment to desired goal sets with constraints on relative configurations.[1]

A. Relevant Work

Problems of this nature have been discussed in the literature in many forms, including formations, abstractions to lower dimensionality, and gradient descent.

There has been extensive work on control of interagent constraints, which include formations and leader-follower control [2]–[6]. In general, however, these works do not guarantee that formation constraints are maintained in the presence of obstacles. Interagent repulsive forces can be used to avoid interrobot collisions [7], [8], but these methods do not guarantee convergence to a desired goal in the presence of obstacles.

For large groups of robots in environments with obstacles, abstractions are often used to lower the dimensionality of the problem at the expense of resolution [9]–[12]. Navigation and obstacle avoidance are at the abstraction level, decreasing computation significantly, but we lose control over the network topology, which can change as the group moves. In [9], robots can collide with each other. We can specify a particular configuration in [11], but the method is not entirely automatic.

Gradient descent has been discussed by Rimon and Koditschek to synthesize *navigation functions* [13], which are nonlinear feedback controllers that guarantee safety (obstacle avoidance) and global convergence. Much work has been done in gradient descent for mobile robots since then, including [14]–[18]. In [16]–[18], navigation functions are developed for multirobot problems. While this approach has the advantage of resulting in controllers with almost global convergence and smooth feedback, it is tedious for complex spaces, involves nonlinear equations, and requires hand tuning of parameters, which makes it impractical for a nonexpert user.

Another approach in gradient descent is to decompose the configuration space into obstacle-free cells and synthesize controllers in each cell to guarantee convergence from any starting configuration to the goal configuration. In [14], a Laplace heat equation is solved on a 2-D disk, which is then mapped to the convex polygons. Composing these mappings results in a smooth controller for a single robot on the 2-D workspace. This study is not applicable to multirobot problems without the addition of constraints, such as prohibiting more than one robot from occupying a room simultaneously.

We would like to design a controller with the safety and convergence properties of navigation functions that can be used in multirobot problems. To simplify the construction of a navigation-function-like controller, Lindemann and LaValle [19] directly construct a smooth field, which takes the place of the gradient of the navigation function. This enables the creation of navigation-function-like controllers in complex spaces of high dimensions by decomposition of the space into convex polytopes; however, it requires full knowledge of state information.

Along a similar vein to [19], we can directly construct vector fields to drive the system to the goal from any initial configuration. For piecewise affine systems, Habets and van Schuppen

Manuscript received October 26, 2009; revised May 18, 2010; accepted July 21, 2010. Date of publication August 30, 2010; date of current version September 27, 2010. This paper was recommended for publication by Associate Editor F. Lamiraux and Editor L. Parker upon evaluation of the reviewers' comments. This work was supported in part by the National Science Foundation under Grant CNS-0410514 and Grant IIS-0427313 and in part by the Army Research Officer Grant DAAD19-02-01-0383 and Grant W911NF-04-1-0148. The work of N. Ayanian was supported by the National Science Foundation.

The authors are with the Department of Mechanical Engineering and Applied Mechanics, University of Pennsylvania, Philadelphia, PA 19104 USA (e-mail: nayanian@grasp.upenn.edu; kumar@grasp.upenn.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TRO.2010.2062070

[1]This work was presented in part at the 2008 IEEE International Conference on Robotics and Automation [1].

[20] and Roszak and Broucke [21] have developed controllers which drive a system in a polytope to a specific exit facet ([21] is limited to simplices). Habets and Schuppen [20] use linear programming to synthesize controllers inside polytopes and provide guarantees on existence of controllers for fully actuated systems. In [21], which is an extension on [20], Roszak and Broucke solve a larger class of problems on simplices, but it is more difficult to implement since it requires nonlinear programming.

Our method is similar in spirit to the decomposition approach. The basic idea is to combine the information about individual Euclidean configuration spaces to construct the configuration space for all n agents and eliminate configurations that result in collisions or violate specified proximity constraints. We construct decentralized feedback controllers for point robots in cells to guarantee convergence to goal sets while satisfying all specified constraints. The controller is decentralized in a sense that controllers for agents that are not connected on the communication graph do not depend explicitly on each other's exact state information. Instead, each agent must know which simplex the system's state is in within the team configuration space. Building, planning, and synthesis of the controllers on the team configuration space are done offline and are centralized.

The controller is extended to nonholonomic robots by feedback linearization. As we will show, based on the constraints imposed (collision, connectivity, etc.), the special case where access to state information is not limited results in a *complete* method. In other words, the algorithm finds a feedback controller for the task if any path exists and fails to find a controller only if no such path exists [22]. The controller synthesis is based on the work of Habets and van Schuppen [20].

Our method can be used to satisfy any combination of constraints on pairs or groups of agents of the types:

1) collision avoidance;
2) connectivity on pairs of agents;
3) mutual exclusion constraints;
4) agent-specific goal configurations;
5) goal configurations that are not agent-specific.

The most important is collision avoidance. We can also specify *connectivity constraints* between designated pairs of agents that specify a maximum distance between these agents. We may wish to enforce a *mutual-exclusion constraint*: There can only be one agent in specified regions. We can also allow *logical combinations of goal configurations*. By this, we mean that if there exist n agents and $l \leq n$ goals in the configuration space, then we can require any l of the agents to achieve the goal positions or specify tasks for combinations of agents, e.g., either agent a_i or (exclusive) a_j must reach the goal set.

In contrast with previous papers on multivehicle coordination, the method is entirely automatic: The only user input is the agents' dynamics, final configuration, geometric information about the space and obstacles, and the task specification. Instead of planning a specific path for the agents to follow, we derive a globally convergent feedback controller. The basic approach is as follows. First, we combine the information about the n-agent configuration spaces and connectivity and collision constraints to generate the team configuration space, i.e., $\mathcal{C}_T$. Connectivity and collision constraints are static and cannot be changed once $\mathcal{C}_T$ is calculated. In this study, we chose to represent the space as a union of *polytopes*.

Definition 1.1: A *polytope* of dimension d is a bounded intersection of a finite set of half spaces in $\mathbb{R}^d$ or, alternatively, the set of solutions to the linear inequality $M\mathbf{x} \leq \mathbf{b}$, where $M \in \mathbb{R}^{d\times d}$, $\mathbf{x} \in \mathbb{R}^{d\times 1}$ is the vector of variables, and $\mathbf{b} \in \mathbb{R}^{d\times 1}$ is a vector of constants.

Next, we derive a discrete graph on the set of polytopes in which each edge represents an allowable path consistent with the task specifications. We then find a discrete path on this graph to the goal set and derive affine feedback controllers to drive any state in each polytope (except the goal polytope) to the next polytope on the path. Finally, we derive an appropriate feedback controller for the goal polytope to drive the system to its desired configuration.

The outline of the paper is as follows. We first introduce some basic definitions and formulate the problem in Section II. In Section III, we determine a discrete path through the space and describe in detail the controller synthesis. We present simulations and experiments in Section IV and discuss the complexity of our algorithm in Section V. We conclude the paper in Section VI.

II. Problem Formulation

Let us consider a team of n kinematic agents $\mathcal{V}_A = \{a_i | i = 1, \ldots, n\}$, which, starting from some initial configuration, must reach some goal configuration while maintaining communication between specified agents and without colliding with each other or obstacles. The agent a_i has the configuration or state $\mathbf{x}_i = [x_i\ y_i\ z_i\ \ldots]^{\mathrm{T}} \in \mathbb{R}^{d_i}$ with the dynamics

$$\dot{\mathbf{x}}_i = \mathbf{u}_i, \qquad \mathbf{x}_i \in \mathbf{X}_i \subset \mathbb{R}^{d_i}, \qquad i = 1, \ldots, n. \qquad (1)$$

Definition 2.1: The *configuration space* $\mathcal{C}_i$ of an agent a_i is the set of all transformations of the agent. The *free space* $\mathcal{C}_i^{\text{free}}$ of a_i is the set of all transformations of a_i that do not intersect with obstacles in the configuration space.

Essentially, $\mathcal{C}_i^{\text{free}}$ is a representation of a_i's state space that contains only those states that do not intersect with obstacles. We assume $\mathcal{C}_i^{\text{free}}$ is tessellated into p_i polytopes with matching facets. As we will see later, this facilitates the controller-synthesis technique we use in this paper. A different decomposition, such as a semialgebraic decomposition, could also be used but must be used with a controller that is designed for that type of set. A convex polytopic decomposition is a simpler case computationally and can be used with the controller that we have chosen.

By matching facets, we mean that any hyperplane that supports two adjacent polytopes shares the same vertices in both polytopes, and any hyperplane can only support one facet of that polytope. This is illustrated in Fig. 1: The decomposition in Fig. 1(a) is an illegal decomposition, since facet (1,3) has an extra vertex from polytopes $\boldsymbol{B}$ and $\boldsymbol{C}$ on the right (i.e., what is one facet on $\boldsymbol{A}$ is two separate facets on $\boldsymbol{B}$ and $\boldsymbol{C}$). The decompositions in Fig. 1(b) and (c) are both legal since all facets match exactly on both sides. It is necessary to match facets to ensure that the state exits the polytope and predictably enters

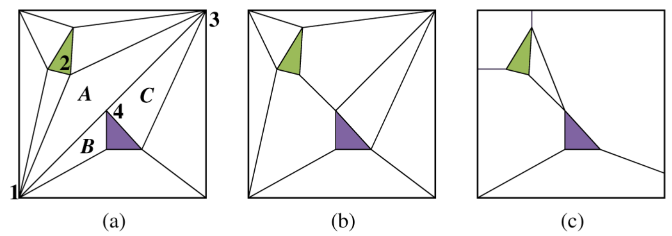


Fig. 1. Possible decompositions of a space. (a) Illegal decomposition [inconsistent number of facets on (1,3)]. (b) and (c) Legal decompositions.

the next polytope on the path. The controller that we use cannot restrict the state from exiting via a portion of the exit facet; the entire facet can be used for exit. Thus, if the desired path were to enter $\boldsymbol{B}$ from $\boldsymbol{A}$, then $\boldsymbol{A}$ must share an entire facet with $\boldsymbol{B}$ or the state could enter $\boldsymbol{C}$.

Definition 2.2: The *team configuration space* is the Cartesian product of the configuration spaces of each agent

$$\begin{aligned}\mathcal{C}_{\text{all}} &= \mathcal{C}_1^{\text{free}} \times \mathcal{C}_2^{\text{free}} \times \cdots \times \mathcal{C}_n^{\text{free}} \\ \mathbf{x} &= [\mathbf{x}_1^{\mathrm{T}}\ \mathbf{x}_2^{\mathrm{T}}\ \ldots\ \mathbf{x}_n^{\mathrm{T}}]^{\mathrm{T}} \in \mathcal{C}_{\text{all}}.\end{aligned} \tag{2}$$

Thus, the configuration of all of the n agents is described by a single point in $\mathcal{C}_{\text{all}}$. $\mathcal{C}_{\text{all}}$ has dimension $D \equiv \sum_{i=1}^{n} d_i$ and contains $\prod_{i=1}^{n} p_i$ polytopes.

In general, the team of agents can be heterogeneous; thus they might not share the same configuration space, so $d_i \neq d_j$. However, in this paper, we will only consider examples in which all configuration spaces are projected onto a plane. Thus, while the configuration spaces are different for different agents (e.g., aerial versus ground), the dimensions of the configuration spaces are identical. From this point, we will let $d \equiv d_1 = d_2 = \cdots = d_n$ so that $D = nd$.

The agents have a predetermined *connectivity graph* whose edges denote constraints on proximity that must be maintained for direct communication of state information between specified agents and a *collision graph* whose edges describe minimum-distance safety constraints that must be maintained.

Let us recall that a *graph* is a pair of sets $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, v_2, \ldots\}$ is the set of vertices or nodes, and $\mathcal{E} \subseteq [\mathcal{V}]^2$ is the set of edges on the graph. Pairs of vertices for which $(v_i, v_j) \in \mathcal{E}$ are called adjacent. A graph in which all pairs of vertices are adjacent is called a complete graph.

Definition 2.3: The *connectivity graph* on the set of agents is the static graph $G_N^{\rho} = (\mathcal{V}_A, \mathcal{E}_N)$, where $\mathcal{E}_N$ is the set of edges on the connectivity graph that describes pairs of agents that directly communicate state information, and ρ is a metric to determine interagent distances. We call pairs of agents that are adjacent on this graph *neighbors* or *neighboring agents*. To maintain connectivity, pairs $(a_i, a_j) \in \mathcal{E}_N$ must maintain a maximum distance $|\mathbf{x}_i - \mathbf{x}_j|_{\rho} \leq \delta_{\max}^{i,j}$. For nonneighboring agents $(a_i, a_j) \notin \mathcal{E}_N$, $\delta_{\max}^{i,j} = \infty$. The constraint can be written

$$\nu_{\rho}(\mathbf{x}_i, \mathbf{x}_j) = |\mathbf{x}_i - \mathbf{x}_j|_{\rho} - \delta_{\max}^{i,j} \leq 0 \ \ \forall (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{E}_N. \tag{3}$$

Examples of connectivity graphs for three or four agents are shown in Fig. 2.

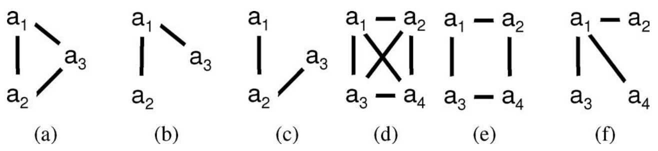


Fig. 2. Sample connectivity graphs.

The connectivity graph provides constraints on the physical proximity of agents so that communication links between pairs of agents $(a_i, a_j) \in \mathcal{E}_N$ can be maintained. However, this does not necessarily mean that agents cannot pass other agents' state information through the graph. For example, in Fig. 2(e), a_1 and a_4 are not connected by an edge; therefore, they cannot communicate directly. However, by passing a_1's state information through a_2 or a_3 (and *vice versa*), a_1 and a_4 would be capable of using each others' state information for feedback.

Definition 2.4: The *information graph* on the set of agents is the static graph $G_I = (\mathcal{V}_A, \mathcal{E}_I)$, where $\mathcal{E}_I$ is the set of all pairs of agents that can access each others' state information. State information can be shared through direct communication, if the pair is connected on G_N^{ρ}, or indirect communication, by passing state information through other agents.

Definition 2.5: The *collision graph* on the set of agents is a static graph $G_L^{\rho} = (\mathcal{V}_A, \mathcal{E}_L)$, where $\mathcal{E}_L$ is the set of all pairs of agents that cannot occupy the same coordinates simultaneously. Pairs $(a_i, a_j) \in \mathcal{E}_L$ must maintain a nonzero minimum distance $|\mathbf{x}_i - \mathbf{x}_j|_{\rho} \geq \delta_{\min}^{i,j}$. For pairs of nonadjacent agents (a_i, a_j), $\delta_{\min}^{i,j} = 0$. We write the constraint

$$\lambda_{\rho}(\mathbf{x}_i, \mathbf{x}_j) = |\mathbf{x}_i - \mathbf{x}_j|_{\rho} - \delta_{\min}^{i,j} \geq 0 \ \ \forall (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{E}_L. \tag{4}$$

For homogeneous agents that occupy the same space, this graph will be complete. However, for heterogeneous agents, this graph may not be complete. For example, if we consider a 2-D configuration space for all vehicles, an aerial vehicle cannot collide with ground vehicles or other aerial vehicles that fly at a different altitude.

The *proximity constraints* specified by the connectivity graph $G_N^{\rho} = (\mathcal{V}_A, \mathcal{E}_N)$ and the collision graph $G_L^{\rho} = (\mathcal{V}_A, \mathcal{E}_L)$ are realized using ρ as the infinity norm.

Definition 2.6: The *infinity norm* $\|\mathbf{x}\|_{\infty}$ of a vector $\mathbf{x} = [x\ y\ \ldots]^{\mathrm{T}}$ in finite-dimensional coordinate space is defined as

$$\|\mathbf{x}\|_{\infty} = \max\{|x|, |y|, \ldots\}. \tag{5}$$

This is our metric of choice because it lends itself easily to convex decompositions. For pairs of agents $(a_i, a_j) \in \mathcal{E}_N \cap \mathcal{E}_L$, the intersection of these constraints corresponds to a square annulus in the relative space of two agents, as shown in Fig. 3(a), where the shaded region denotes illegal configurations. Pairs of agents $(a_i, a_j) \in \mathcal{E}_N - \mathcal{E}_L$ (neighbors without collision constraint) will have only the maximum distance constraint [see Fig. 3(b)]. Pairs of agents $(a_i, a_j) \in \mathcal{E}_L - \mathcal{E}_N$ (nonneighbors with collision constraint) will have infinite annuli [see Fig. 3(c)]. The annuli in Fig. 3(a) and (c) are decomposed into four convex polytopes with matching facets to satisfy our legal-decomposition criterion.

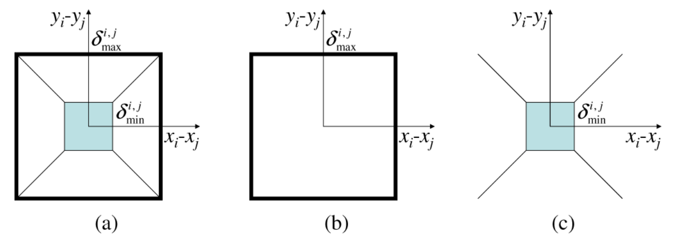


Fig. 3. Proximity constraints for pairs of 2-D agents. The shaded region indicates configurations that are not allowed. (a) Neighbors with collision constraint. (b) Neighbors with no collision constraint. (c) Nonneighbors with collision constraint.

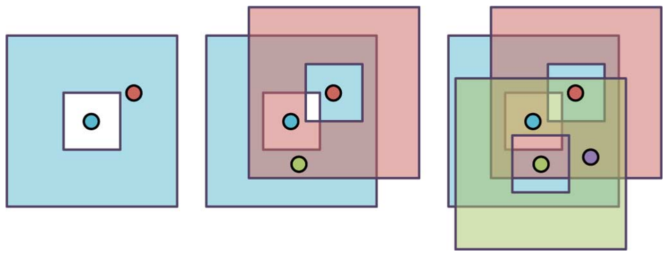


Fig. 4. Intersection of annuli for multiple agents.

For teams with complete collision and connectivity graphs, the proximity constraints can be viewed as a square annulus around every point in the physical space of an agent, which describes the possible locations of other agents. The safe region of a third agent, with the assumption of complete collision and connectivity graphs, would be a square annulus around the original agent intersected with the annulus of the second agent. This is shown in Fig. 4 for up to four agents with complete graphs.

Definition 2.7: The *mutual-exclusion graph* on the set of agents is the static graph $G_M = (\mathcal{V}_A, \mathcal{E}_M)$, where $\mathcal{E}_M$ is the set of all pairs of agents that cannot simultaneously occupy the same polytope in $\mathcal{C}_i^{\text{free}} = \mathcal{C}_j^{\text{free}}$ if they share the same decomposition. The constraint can be written as

$$\mu(\mathbf{x}_i, \mathbf{x}_j) = \begin{cases} 0, & (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{E}_M \\ 1, & \text{otherwise.} \end{cases} \tag{6}$$

Definition 2.8: The *task configuration space* $\mathcal{C}_T$ is the set

$$\mathcal{C}_T = \mathcal{C}_{\text{all}} \cap \mathcal{L}_\rho \cap \mathcal{M} \cap \mathcal{N}_\rho \tag{7}$$

where

$$\begin{aligned} \mathcal{L}_\rho &\equiv \{\mathbf{x} | \mathbf{x} \in \mathcal{C}_{\text{all}}, \lambda_\rho(\mathbf{x}_i, \mathbf{x}_j) \geq 0 \quad \forall (a_i, a_j) \in \mathcal{E}_L\} \\ \mathcal{M} &\equiv \{\mathbf{x} | \mathbf{x} \in \mathcal{C}_{\text{all}}, \mu(\mathbf{x}_i, \mathbf{x}_j) = 1 \quad \forall (a_i, a_j) \in \mathcal{E}_M\} \\ \mathcal{N}_\rho &\equiv \{\mathbf{x} | \mathbf{x} \in \mathcal{C}_{\text{all}}, \nu_\rho(\mathbf{x}_i, \mathbf{x}_j) \leq 0 \quad \forall (a_i, a_j) \in \mathcal{E}_N\}. \end{aligned} \tag{8}$$

The task configuration space is composed of polytopes in which the agents cannot collide, lose communication, or violate the mutual exclusion constraints. $\mathcal{C}_T$ is composed of polytopes since it is an intersection of bounded polytopes ($\mathcal{C}_{\text{all}}$) with unbounded polytopes (collision and connectivity constraints), with some polytopes excluded (mutual exclusion constraints). Since remaining in $\mathcal{C}_T$ ensures safety, we would like to automatically synthesize a controller that drives any state in $\mathcal{C}_T$ to the goal configuration without exiting $\mathcal{C}_T$. We seek an affine input function $\mathbf{u} = F\mathbf{x} + \mathbf{g}$, where $\mathbf{u} = [\mathbf{u}_1^{\mathrm{T}}\ \mathbf{u}_2^{\mathrm{T}}\ \ldots \mathbf{u}_n^{\mathrm{T}}]^{\mathrm{T}}$, that will ensure that the state is always in $\mathcal{C}_T$, and that at some time T_0, the state is arbitrarily close to the goal configuration.

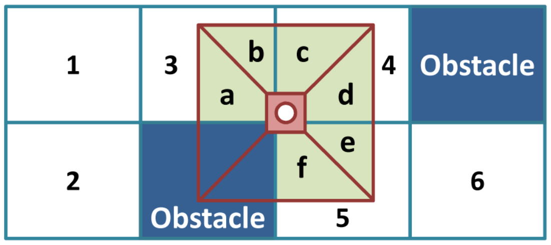


Fig. 5. Example of a combination goal requirement for two agents. Either agent must reach the goal in polygon 4. Proximity constraints require the other agent to be in one of the regions a–f.

Problem 2.9 (Continuous problem): For any initial state $\mathbf{x}_0$, let us consider the system (1) on $\mathbb{R}^D$, with goal configuration $\mathbf{x}^g \in \mathcal{C}_T \subset \mathbb{R}^D$ and metric ρ. Find a piecewise-affine state feedback policy $\mathbf{u} : [0, T_0] \to \mathbf{U}$ for any $\mathbf{x}_0 \in \mathcal{C}_T$ such that

1) $\forall t \in [0, T_0]$, $\mathbf{x} \in \mathcal{C}_{\text{all}}$ and $\mathbf{x}(T_0)$ arbitrarily close to $\mathbf{x}^g$;
2) $\dot{\mathbf{x}}_i = \mathbf{u}_i$;
3) $\mathbf{x}(t) \in \mathcal{L}_\rho \cap \mathcal{M} \cap \mathcal{N}_\rho, \forall t \in [0, T_0]$.

Note that in Problem 2.9, we can let $\mathbf{u}$ be any affine function of the state of the team. In other words, the team of robots must respect the desired connectivity, collision, and mutual exclusion graphs but without limitations on the state information available to the agents so that agents can share all state information through the communication graph via indirect communication.

Problem 2.10 (Continuous-problem, partial-state feedback): Let us consider Problem 2.9 with the additional constraint on the affine function $\mathbf{u} = F\mathbf{x} + \mathbf{g}$:

4) $\mathbf{u}_i$ does not depend explicitly on $\mathbf{x}_j$ if $(a_i, a_j) \notin \mathcal{E}_I$.

For situations when the goal positions are not specifically assigned to each agent, e.g., when only m of n agents are required to reach a goal location, we have a finite number of goal nodes, as opposed to a single goal node. Let $\mathcal{X}^g$ be the set of goal nodes. We can then replace $\mathbf{x}^g$ by the set $\mathcal{X}^g$ in Problems 2.9 and 2.10 to allow a set of goal configurations. Fig. 5 depicts an example where either one of two agents must reach the goal configuration in polygon 4. The proximity constraints limit the location of the other agent to the six polygons a–f. Thus, we have 12 possible goal nodes: six for each agent at the goal.

III. Feedback Controllers on $\mathcal{C}_T$

In this section, we develop feedback controllers to solve Problems 2.9 and 2.10. In other words, we ensure that the agents are always inside the team configuration space $\mathcal{C}_T$ and that they reach the goal configuration. There are two stages in this process. First, we pursue a discrete representation of C_T and find paths in this discrete representation. Next, we translate these paths into feedback controllers.

The key step in the first stage is to define an adjacency graph on the set of polytopes.

Definition 3.1: The *polytope graph* $G_P = (\mathcal{V}_P, \mathcal{E}_P)$ on the polytopes in $\mathcal{C}_T$ is the pair of sets $\mathcal{V}_P = \{P^1, P^2, \ldots\}$, where P^k is the kth polytope, and $\mathcal{E}_P = \{e_P^{k,m} | P^k$ is adjacent to P^m, $\forall k, m\}$ is the set of all pairs of polytopes that share a (matching) facet.

Problem 3.2 (Discrete polytope path): For all nodes $P^k \in \mathcal{V}_P$, let us find a path on the polytope graph G_P to the goal node P^g.

Because of the constraints on sharing state information, we must triangulate the polytopes P^k into simplices s_q^k. The reason for this will become clear in the discussion of the controller that we will use. We now define an adjacency graph on the set of simplices in each polytope.

Definition 3.3: The kth *simplex graph* $G_S^k = (\mathcal{V}_S^k, \mathcal{E}_S^k)$ on the simplices s_q^k in polytope P^k is the pair of sets $\mathcal{V}_S^k = \{s_1^k, s_2^k, \ldots\}$, where s_q^k is the qth simplex $s_q^k \subset P^k$, and $\mathcal{E}_S^k = \{e_S^{k(q,r)} | s_q^k$ is adjacent to s_r^k, $\forall q, r\}$ is the set of all pairs of simplices that share a facet. We use the simplex graph to determine a discrete path from each simplex in a polytope's triangulation to the simplices on its exit facet.

Since there are multiple paths to the goal, we can use the usual notion of distance on a graph to find the shortest paths in the polytope graph using an algorithm like Dijkstra's algorithm. We can associate with each edge $e_P^{k,m} \in \mathcal{E}_P$ between adjacent polytopes P^k and P^m a cost, which we minimize. From our view point, the least cost path minimizes the number of polytopes that are visited; therefore, $|e_P^{k,m}| = 1, \forall e_P^{k,m} \in \mathcal{E}_P$. This in turn minimizes the number of transitions between polytopes. Similarly, we find the shortest path in the same sense on the simplex graphs to any of the simplices on the exit facet. In the goal polytope P^g, we find the shortest path from any simplex to the goal simplex s^g.

Once the paths on the polytope and simplex graphs are identified, we want to be able to synthesize decentralized feedback controllers to solve Problem 2.10. The synthesis procedure is similar, in spirit, to those discussed in [15], [20], [21], and [23] but is closest to the one developed by Habets and van Schuppen [20] to determine a centralized affine state feedback that satisfies a set of inequalities on a polytope. In their paper, they derive controllers that drive an affine system from any initial condition in a polytope through a desired exit facet in the polytope while guaranteeing the system does not leave the polytope. We slightly modify this algorithm to design a decentralized affine controller within each simplex.

We now consider the subproblem of steering states in a simplex to a specified exit facet without limiting state information.

Problem 3.4 (Continuous subproblem): Let us consider the system (1) on simplex $s_q^k \in P^k \in \mathcal{C}_T$, where s_q^k is the qth simplex in P^k, the kth polytope on a path to the goal, and $x^g \notin s_q^k$. Let s_r^k be the next simplex on the path. Let $\mathcal{F}_q$ be the facet shared by s_q^k and s_r^k with normal vector $\mathbf{n}_q$ pointing out of s_q^k. For any initial state $\mathbf{x}_0 \in s_q^k$, we have to find a time instant $T \geq 0$ and an input function $\mathbf{u} : [0, T] \to \mathbf{U}$ (where $\mathbf{u}$ is realized by the application of a continuous feedback law $\mathbf{u}(t) = F\mathbf{x} + \mathbf{g}$, $F \in \mathbb{R}^{D\times D}$, $\mathbf{g} \in \mathbb{R}^{D\times 1}$) such that

1) $\forall t \in [0, T] : \mathbf{x}(t) \in s_q^k$;
2) $\mathbf{x}(T) \in \mathcal{F}_q$, and T is the smallest time instant in the interval $[0, \infty)$ for which the state reaches facet $\mathcal{F}_q$;
3) $\mathbf{n}_q^{\mathrm{T}} \dot{\mathbf{x}}(T) > 0$, i.e., the velocity vector $\dot{\mathbf{x}}(T)$ at $\mathbf{x}(T) \in \mathcal{F}_q$ has a positive component in the direction of $\mathbf{n}_q$.

Note that Problem 3.4 naturally implies that all agents have access to the full state since the information graph is complete. We now consider the subproblem of steering states in a simplex to a specified exit facet with limited state information.

Problem 3.5 (Continuous-subproblem, limited feedback): Let us consider Problem 3.4 with the additional constraint.

4) Matrix F, which is composed of matrices $F^{ij} \in \mathbb{R}^{d\times d}, i, j = 1, \ldots, n, i \neq j$, is such that $F^{ij} = \mathbf{0}$ if $(a_i, a_j) \notin \mathcal{E}_I$.

In the goal simplex s^g, we solve the equation

$$\dot{\mathbf{x}}|_{\mathbf{x}=\mathbf{x}^g} = F\mathbf{x}^g + \mathbf{g} = \mathbf{0}. \tag{9}$$

Let us assume (without loss of generality) that the exit facet for each simplex has index 1 and outward normal $\mathbf{n}_1$ and contains vertices $\{\mathbf{v}_2, \ldots, \mathbf{v}_{D+1}\}$. Problem 3.4 is solved on the simplex by the linear program

$$\begin{aligned} \min \quad & \mathbf{0}^{\mathrm{T}} [\mathbf{u}_1^{\mathrm{T}}\ \mathbf{u}_2^{\mathrm{T}}\ \ldots\ \mathbf{u}_{D+1}^{\mathrm{T}}]^{\mathrm{T}} \\ s.t. \quad & \mathbf{n}_1^{\mathrm{T}} \mathbf{u}|_{\mathbf{v}_c} > 0, \qquad c \in \{2, \ldots, D+1\} \\ & \mathbf{n}_b^{\mathrm{T}} \mathbf{u}|_{\mathbf{v}_1} \leq 0, \qquad b \in \{2, \ldots, D+1\} \\ & \mathbf{n}_b^{\mathrm{T}} \mathbf{u}|_{\mathbf{v}_c} \leq 0, \qquad b, c \in \{2, \ldots, D+1\}, \qquad b \neq c \end{aligned} \tag{10}$$

where $\{\mathbf{u}|_{\mathbf{v}_1}, \ldots, \mathbf{u}|_{\mathbf{v}_{D+1}}\}$ are the inputs evaluated at the vertices. The linear program (10) generates the inputs at the vertices of each simplex, which must be interpolated within the simplex (for more detail see [20]). Every point in a simplex is described by a unique convex combination of the vertices. The same convex combination of the inputs at the vertices is used to evaluate the controller within each simplex and determines the calculation of F and $\mathbf{g}$. The feedback matrix F and vector $\mathbf{g}$ must be solved in each simplex, and thus are constant in each simplex.

Theorem 3.6: Problem 2.9 has a solution if and only if Problem 3.2 has a solution and the inputs are not constrained.

Proof: This proof is a straightforward extension of the results in [20]. ■

Theorem 3.7: Problem 3.2 has a solution to any goal from any initial configuration if and only if G_P is connected.

Proof: $\mathcal{C}_T$ contains every allowable configuration $\mathbf{x}$ in our polytopic world model. G_P contains all the information about the connectivity of $\mathcal{C}_T$. Thus, if there is a solution to Problem 3.2, there must exist a path from any node in G_P to the goal node(s). Conversely, if there is no path on G_P between any two nodes, there is no solution to Problem 3.2. ■

Theorem 3.8: Problem 2.10 has a solution if Problem 3.2 has a solution, and there exists a corresponding solution to Problem 3.5 for each simplex $s_q^k \in \mathcal{C}_T$, as well as a solution to (9) for the goal simplex s^g.

Proof: If there is a solution to Problem 3.2, there exists a path from any node in G_P to the goal node(s). On each polytope P^k that does not contain the goal, there exists a path in G_S^k to the exit facet of P^k defined by the solution to Problem 3.2 (triangulation of a polytope results in a connected graph). In addition, there trivially exists a path on the simplex graph in P^g to the goal simplex s^g. Paths on the simplex graphs define the exit facet for each simplex in $\mathcal{C}_T$. If there exists a solution to Problem 3.5

for the *particular exit facet for each of these simplices*, and a solution for (9) in s^g, then Problem 2.10 has a solution. ∎

The constraint (4) in Problem 3.5 can be formulated as a supplementary equality constraint on the linear program that was used to solve for the inputs at the vertices of each simplex. Without requirement (4), F and $\mathbf{g}$ are calculated after solving the linear program by using the equation

$$\begin{bmatrix} F^{\mathrm{T}} \\ \hline \mathbf{g}^{\mathrm{T}} \end{bmatrix} = WU \tag{11}$$

where

$$W \equiv \begin{bmatrix} \mathbf{v}_1{}^{\mathrm{T}} & 1 \\ \vdots & \vdots \\ \mathbf{v}_{D+1}{}^{\mathrm{T}} & 1 \end{bmatrix}^{-1} \equiv \begin{bmatrix} W_1{}^{\mathrm{T}} \\ \vdots \\ W_{D+1}{}^{\mathrm{T}} \end{bmatrix}$$

$$U \equiv \begin{bmatrix} \mathbf{u}|_{\mathbf{v}_1}{}^{\mathrm{T}} \\ \vdots \\ \mathbf{u}|_{\mathbf{v}_{D+1}}{}^{\mathrm{T}} \end{bmatrix} \equiv [U_1 \ \cdots \ U_{D+1}].$$

However, since we know certain entries of F must be zero, we restrict solutions of U accordingly. To more easily impose these constraints on F, we solve for U at the simplex level. (If we fix the triangulation, it is possible to solve for U at the polytope level by calculating the constraints on U at the simplex level but solving F at the polytope level; however, we do not explore this idea further since we do not object to nondifferentiable controls at the facets.) Then, we can write

$$F^{ij} = \begin{bmatrix} W_{2j-1}{}^{\mathrm{T}} U_{2i-1} & W_{2j-1}{}^{\mathrm{T}} U_{2i} \\ W_{2j}{}^{\mathrm{T}} U_{2i-1} & W_{2j}{}^{\mathrm{T}} U_{2i} \end{bmatrix}.$$

The linear program to solve Problem 3.5 is as follows:

$$\begin{aligned} \min \quad & \mathbf{0}^{\mathrm{T}} [\mathbf{u}_1^{\mathrm{T}} \ \mathbf{u}_2^{\mathrm{T}} \ \ldots \ \mathbf{u}_{D+1}^{\mathrm{T}}]^{\mathrm{T}} \\ s.t. \quad & \mathbf{n}_1^{\mathrm{T}} \mathbf{u}|_{\mathbf{v}_c} > 0, \qquad c \in \{2, \ldots, D+1\} \\ & \mathbf{n}_b^{\mathrm{T}} \mathbf{u}|_{\mathbf{v}_1} \le 0, \qquad b \in \{2, \ldots, D+1\} \\ & \mathbf{n}_b^{\mathrm{T}} \mathbf{u}|_{\mathbf{v}_c} \le 0, \qquad b, c \in \{2, \ldots, D+1\}, \ b \neq c \\ & F^{ij} = \mathbf{0}, \qquad i, j \in \{1, \ldots, n\}, (a_i, a_j) \notin \mathcal{E}_I, \ i \neq j. \end{aligned} \tag{12}$$

The equality constraint in (12) results directly from Problem 3.5 constraint (4). The linear program (12) has $2D(D+1)$ constraints: $D(D+1)$ inequality and $D(D+1)$ equality constraints. Since we have $2D(D+1)$ unknowns ($2D$ entries of F, D entries of $\mathbf{g}$, and $\mathbb{R}^D$ inputs at each of the $D+1$ vertices), we cannot guarantee that a solution to Problem 3.5 will be found. However, if in some simplex a solution is not found for a particular exit facet, it is possible to "reroute" the path on the simplex graph so that all states that enter that simplex exit through a different exit facet. Furthermore, it is possible to build G_P and all G_S^k based on the existence of controllers that drive the system to particular exit facets.

In summary, the algorithm for controller synthesis or the solution to Problem 2.9 (respectively, 2.10) involves the following four steps:

Algorithm 3.9:

1) Construct task configuration space $\mathcal{C}_T$ (Definition 2.8).
2) Find paths on polytope and simplex graphs G_P, G_S.
3) Solve Problem 3.4 (respectively, 3.5) on all simplices except s^g.
4) Solve Equation 9 in s^g.

Using vector fields instead of potential or navigation functions for a controller has several advantages. The vector field approach, which results in piecewise linear or piecewise affine feedback, is a computational approach instead of an analytical approach, and therefore, it can be used in any environment, while navigation functions are only formulated for star-shaped sets [13]. The vector field method can be automated, since all values can be determined with a linear program. In contrast, at least one parameter must be manually chosen for the navigation-function method: The more complex the space, the more parameters must be chosen [13]. While the navigation function method does not scale, the vector field method scales exponentially (as described in Section V).

IV. Simulations and Experiments

In this section, we solve several multiagent coordinated control problems to illustrate the application of the technique. Preprocessing of the controller is done in MATLAB using the multiparametric toolbox for polytope computations [24]. Three-dimensional dynamic simulation of the robots is done using GAZEBO, which is a part of the PLAYER/STAGE/GAZEBO project [25]. GAZEBO is an open source multirobot simulator, which is designed to accurately simulate a small population of robots with high fidelity. We use a MATLAB API [26] that interacts directly with GAZEBO and PLAYER to provide real-time control in simulations and experiments, respectively.

The robot we use is the SCARAB robot, as shown in Fig. 6(a), which is a nonholonomic platform. Although the controller is designed for robots with the dynamics of (1), we use feedback linearization to provide inputs to the robots in (v, ω) form [27]. Fig. 6(c) shows the effect of feedback linearization on the minimum distance requirement. In the differential drive SCARABs, we track the reference point P_i, which is offset from P_o by a feedback linearization distance f. The entire robot lies within a circle centered at P_i of radius $R = f + r$, where r is the original radius of the robot. We translate the linear velocity commands output by the controller to linear and angular velocity by inverting the equations (for $f > 0$)

$$\begin{bmatrix} \dot{x}_i \\ \dot{y}_i \end{bmatrix} = \begin{bmatrix} \cos\theta_i & -f\sin\theta_i \\ \sin\theta_i & f\cos\theta_i \end{bmatrix} \begin{bmatrix} v \\ w \end{bmatrix}. \tag{13}$$

Since the radius enclosing the robot at the reference point is much larger than the radius of the robot itself, we must increase the size of the obstacles by the new radius R to prevent collisions, as well as decrease our maximum and increase our minimum proximity constraints.

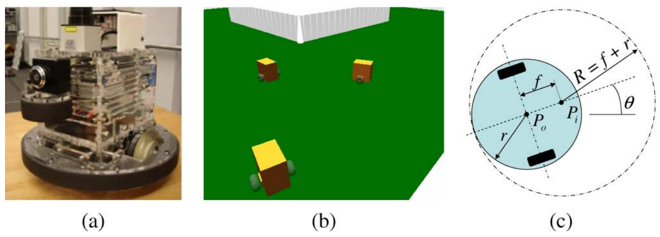


Fig. 6. SCARAB robot.

A. Three Agents Negotiate Passage Through a Corridor

Fig. 7(a) and (b) shows a simulation in which the agents successfully traverse a corridor too narrow to allow the agents to traverse it in anything other than a single-file formation, which makes the proximity constraints difficult to preserve. Note that no formation is specified. Only the collision graph (complete), connectivity, and information graphs [both as shown in Fig. 2(c)] are specified ($\delta_{\min} = 0.7$ ft and $\delta_{\max} = 1.6$ ft).

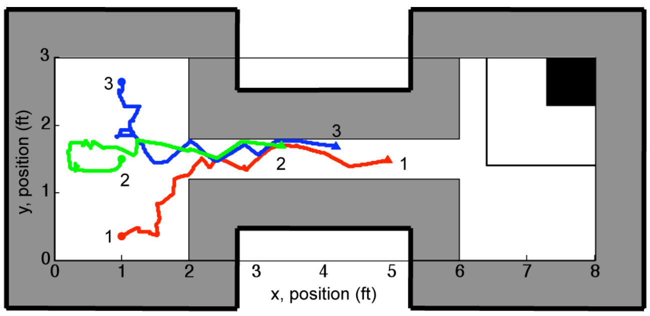
B. MATLAB Simulations Through a Complex and Real Space

Fig. 8 shows a simulation of a real-world problem, where vehicles with realistic ranges of connectivity [150 m in Fig. 8(b) and 250 m in Fig. 8(c)] navigate through an urban environment to their respective destinations while keeping within the specified range. Fig. 8(b) has two ground vehicles with complete connectivity, collision, and information graphs. Fig. 8(c) shows two ground vehicles and one unmanned aerial vehicle (UAV) with a very large connectivity range (greater than 1000 m.).

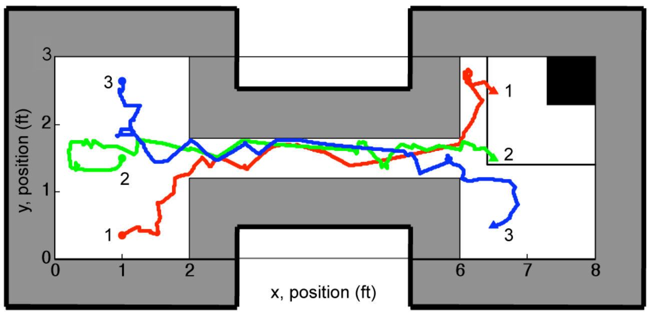
C. Three Agents in MATLAB and GAZEBO

In a multiply connected space, agents have more than one route to the goal. Based on initial conditions and distance constraints, the agents will choose different paths. A simple multiply connected space is shown in Fig. 9(a) and (c). In these simulations, the agents share the same configuration space, collision graph (complete), start and goal configurations, and proximity constraints ($\delta_{\min} = 0.7$ m, and $\delta_{\max} = 2.5$ m). In the centralized case, the connectivity and information graphs are complete; in the decentralized case [see Fig. 9(c) and (d)], they are not complete [both as shown in Fig. 2(c), with agent 1 red, agent 2 green, and agent 3 blue]. The distance between the solid lines and the black dotted lines at the starting point is because of the feedback linearization distances (centralized: 0.3 m and decentralized: 0.4 m). The agents take different routes to the goal since the feedback-linearization points for the two cases at $t = 0$ are in different polytopes.

Fig. 9(b) and (d) shows distance between agents in each case. The first row corresponds to agents 1 and 2, the second to agents 2 and 3, and the third to agents 1 and 3. The left column shows the Euclidean distance between pairs of agents; center and right columns show distance in the x-direction and y-direction, respectively. The plots show that proximity constraints are maintained. In the center plot in Fig. 9(d), the red line that depicts interrobot distance dips below the dashed blue line, which indicates that the robots' distance has exceeded the limit; however,



(a)



(b)

Fig. 7. Three agents through a narrow corridor. The black boxes on the upper right show proximity constraints, i.e., $\delta_{\min} = 0.7$ ft, $\delta_{\max} = 1.6$ ft. The thick black outline denotes the physical workspace, with the gray areas representing the areas of the configuration space, which are within $\delta_{\min}$ of the workspace boundaries. (a) Intermediate state. (b) Final state.

the distance between the feedback linearization points does not exceed the limit. Since sufficient security margin was built into the interrobot distances, the robots do not collide. This shows the importance of building in sufficient security margin of at least the robot radius plus the feedback linearization distance, as shown in Fig. 6(c).

D. Experiments With Two Agents

For experiments, we have used two SCARAB robots in the environment pictured in Figs. 10 and 11. Overhead cameras, which are accurate within 3 cm, were used for robot tracking. Obstacles were padded [see Fig. 10(a)] according to the feedback linearization distance of 18 cm.

Fig. 10(a) and (b) shows the results of one experiment. Fig. 10(b) shows distance between the two robots and the distance between the feedback linearization points ($\delta_{\min} = 0.5$ m and $\delta_{\max} = 3$ m). Fig. 11 shows sequential still frames of the experiment. This experiment shows that the controller can successfully be applied to real robot navigation problems.

V. COMPUTATIONAL COMPLEXITY

In this section, we discuss the computational complexity of our method. Worst-case complexity is mostly determined by the number of polytopes in $\mathcal{C}_T$, which scales exponentially with n. However, actual numbers observed are much lower.

$\mathcal{C}_{\text{all}} \in \mathbb{R}^D$ contains $\prod_{i=1}^{n} p_i$ polytopes, where p_i is the number of polytopes in $\mathcal{C}_i^{\text{free}}$. For each pair of agents with collision

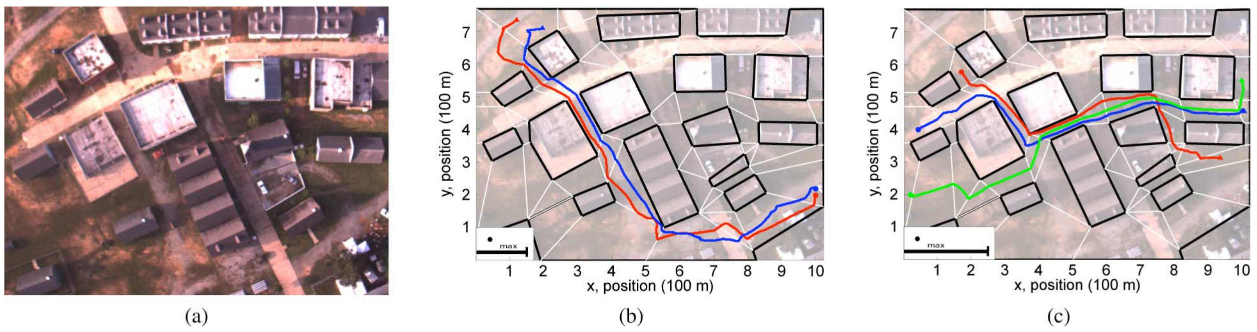


Fig. 8. Team of unmanned vehicles navigates an urban environment. (a) Aerial view of the urban environment used in the simulation. (b) Results with a team of two ground agents. (c) Results with two ground agents and one UAV. In (b) and (c), buildings are enclosed by black polygons. The bar on the bottom left shows $\delta_{\max}$ for each simulation. In (c), one agent is a UAV, which has a long range of connectivity.

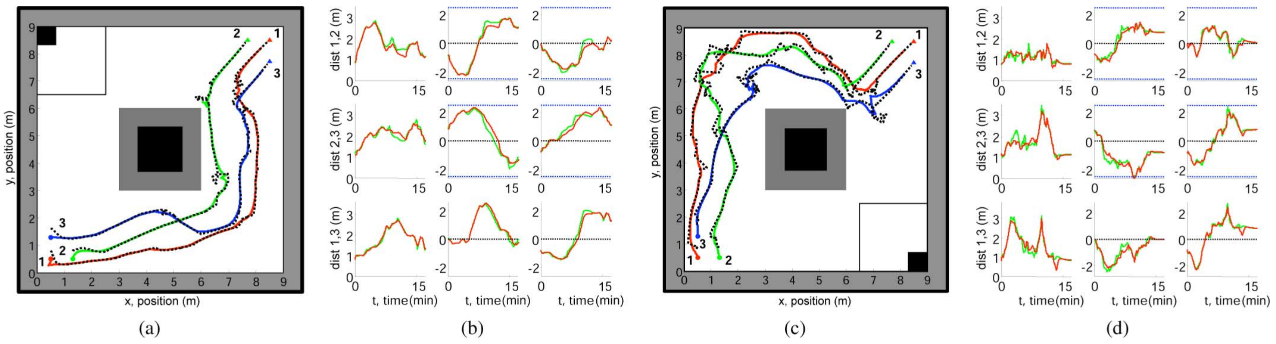


Fig. 9. GAZEBO simulations in a multiply connected space with identical proximity constraints [agent pair (1,3) not connected] with centralized and decentralized controllers. The thick black outline and black box in the center denotes the physical workspace, with the gray areas representing the areas of the configuration space that are within $\delta_{\min}$ of the workspace boundaries. The black boxes in the corners denote proximity constraints $\delta_{\min}$ and $\delta_{\max}$. (a) Results of centralized simulation. (b) Distances between agents in centralized simulation. (c) Results of decentralized simulation. (d) Distances between agents in decentralized simulation. In panels (a) and (c), solid lines show the position of the robot, and dotted lines show the position of the feedback linearization point. In panels (b) and (d), red depicts robot position, green depicts feedback linearization point position, and dotted blue marks the maximum allowable distance. In the center plot of panel (d), the robot exceeds the maximum since limits are on the feedback linearization point and not on robot position.

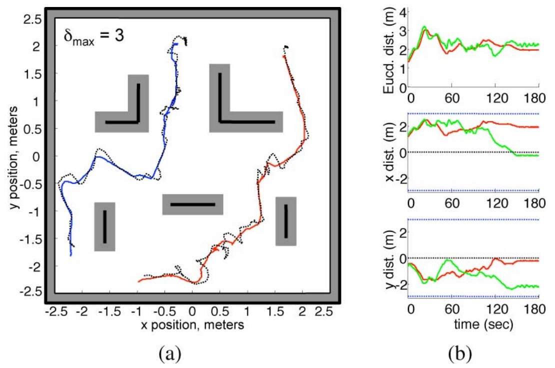


Fig. 10. Experiment with two SCARAB robots in a multiply connected space. (a) Experimental results. Solid lines show the position of the robot, and dotted lines show position of the feedback linearization point. The thick black outline of the workspace and the thick black lines within the workspace denote the physical workspace, with the gray areas representing the areas of the configuration space that are within $\delta_{\min} = 0.5$ m of the workspace boundaries and obstacles ($\delta_{\max} = 3$ m). (b) Distances between robots. Red depicts robot position, green depicts the feedback linearization point, dotted blue marks the maximum allowable distance.

constraints, we have one annulus with four regions, which results in a maximum of $4^{n(n-1)/2}$ proximity regions intersected with $\mathcal{C}_{\text{all}}$. Thus, the maximum number of polytopes in $\mathcal{C}_T$ is

$$P_{\max} = 4^{n(n-1)/2} \prod_{i=1}^{n} p_i. \quad (14)$$

This is a worst-case scenario, as the proximity constraints are dependent ($x_1 < x_2,\ x_2 < x_3 \implies x_1 < x_3$). Additionally, a portion of these polytopes will violate proximity constraints. If the physical space is large and two agents are neighbors, then they cannot be at opposite ends of the space. Fig. 12 depicts a situation of this type for two neighboring agents. If one agent is in region 1, the other cannot be in region 4 since no point in region 4 is within the constraints, and *vice versa* (similarly for regions 3 and 6). Any product of polytopes in neighboring agents' configuration spaces that are beyond the maximum connectivity limit will be eliminated when intersecting $\mathcal{C}_{\text{all}}$ with the proximity constraints (8) to create $\mathcal{C}_T$.

Table I specifies the complexity of every step in the process, where LP(u, w) represents the complexity of a linear program in u dimensions and w constraints, h_k is the number of inequalities used to describe the polytope P^k after proximity constraints are added, $|V_k|$ (respectively, $|F_k|$) is the number of vertices (respectively, facets) in polytope P^k, and $|S_k|$ is the number of simplices in the Delaunay triangulation of P^k.

The computational expense of the process depends largely on the methods used for Cartesian products and intersections, as well as the number of agents, connectivity, and complexity of the space. The larger the number of agents, the higher the connectivity, and the more complex the space, the more computation

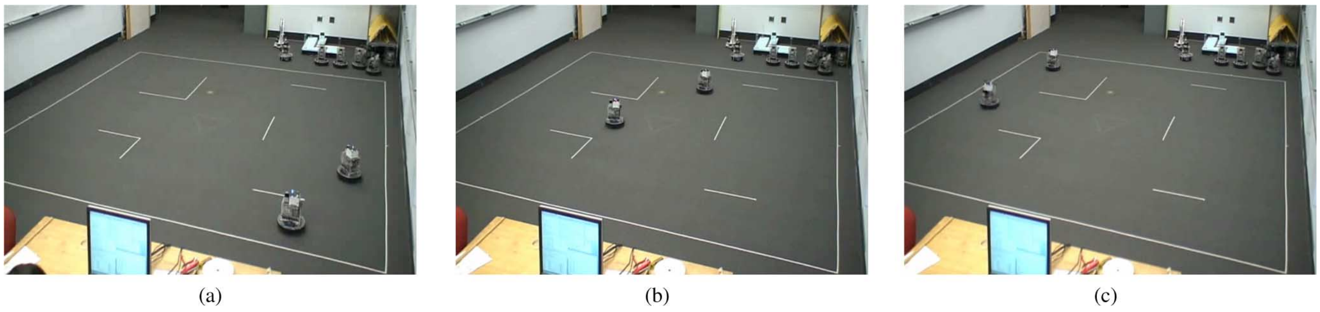


Fig. 11. Sequential still frames of the experiment on two SCARAB robots. (a) Start configuration. (b) Intermediate configuration. (c) Goal configuration.

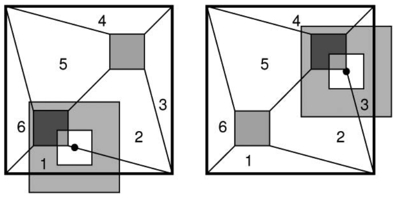


Fig. 12. Distant pairs of polytopes.

TABLE I
COMPLEXITY OF EACH STEP

Task	**Complexity**	**Ref**
Construct $\mathcal{C}_T$ from $\{\mathcal{C}_i^{free}\}_1^n$	$\mathcal{O}(P_{\max} \cdot LP(h_k+1, d))$	[28]
Plan on polytopes	$\mathcal{O}(\lvert\mathcal{E}_A\rvert + P_{\max}\log P_{\max})$	[29]
Solve inequalities^a	$\mathcal{O}(LP(\lvert V_k\rvert\lvert F_k\rvert - d, d^2))$	[30]
Triangulation^a	$\mathcal{O}(\lvert\mathcal{V}_k\rvert^{d/2})$	[28]
Plan in goal polytope P^g	$\mathcal{O}(\lvert E_S\rvert + \lvert S_k\rvert \log \lvert S_k\rvert)$	[29]
Solve inequalities^b in P^g	$\mathcal{O}(LP(d^2+d+1, d^2))$	[30]
Solve F, g on polytopes^b	$\mathcal{O}\left(\frac{2}{3}(d+1)^3 + 2d(d+1)^2\right)$	[30]

^{a}per polytope, ^{b}per simplex

TABLE II
CRITICAL VALUES IN OUR SIMULATIONS

	$P_{\max}$	$P = \lvert\mathcal{C}_T\rvert$	$\sum_{i=1}^P V_i/P$	$\sum_{i=1}^P F_i/P$
Fig. 8b	12,996	2572	18	9
Fig. 9a	4096	464	92	15

will be required to generate $\mathcal{C}_T$. Additionally, one must consider the controller that is used on the resulting polytopes.

Table II presents critical values from our simulations. Although $P_{\max}$ scales exponentially, the actual number of polytopes in $\mathcal{C}_T$ is much lower. The table also presents the average number of vertices and facets per polytope. Several methods can be used to decrease the computation time required.

A. Distant Pairs of Polytopes

Given a connectivity graph, it is possible to rule out some combinations of polytopes before taking the Cartesian product. An example was shown in Fig. 12. This does not decrease the number of polytopes in $\mathcal{C}_T$; however, it decreases the computation time it takes to generate $\mathcal{C}_T$.

B. Cartesian Products of Free Spaces

Because of the size of $P_{\max}$, constructing $\mathcal{C}_T$ is the most time-consuming portion of preprocessing. If we are considering homogeneous agents that share the exact same workspace, then $\mathcal{C}_i^{\text{free}} = \mathcal{C}_j^{\text{free}}\ \forall i, j$. Therefore, when constructing $\mathcal{C}_{\text{all}}$, we simply take the Cartesian product of $\mathcal{C}_i^{\text{free}}$ with itself. There is potential to reduce the complexity of constructing $\mathcal{C}_T$ by taking this fact into consideration. By clever bookkeeping, we can take advantage of this to reduce the computation required to calculate $\mathcal{C}_{\text{all}}$ and build G_P.

C. Lazy Evaluation

If an initial configuration is given, and the agents cannot initiate in another polytope, find a controller only in the polytopes that are on the path from the initial to final polytope. Although this will not reduce the number of calculations to find $\mathcal{C}_T$, it will *significantly* decrease the number of polytopes for which controllers must be found.

More generally, given a limited number of polytopes that contain all possible initial configurations of the agents, solve only in those polytopes through which the state will pass.

VI. CONCLUDING REMARKS

We have presented a method for synthesizing feedback controllers for a team of multiple heterogeneous agents to navigate a known environment with obstacles and its application to navigation in urban environments. The feedback controllers obtained by this method provide guarantees of convergence to a goal in any known polygonal environment. In MATLAB simulations, we showed the application of the algorithm to agents that navigate a narrow corridor as well as an urban environment. In experiments and 3-D dynamic GAZEBO simulations, the controllers, although not specifically designed for nonholonomic robots, successfully drive agents with limited system state information to goal sets while avoiding collisions and maintaining specified proximity constraints. Additionally, we have shown in experiments that the controllers can be successfully applied to real robot navigation problems.

One limitation of this algorithm is that the complexity is exponential in the number of agents. However, we have briefly discussed several methods of decreasing complexity. We can reduce complexity while constructing the task configuration space by considering the distance between polygons in the free space (see Section V-A), by taking advantage of homogeneous agents with identical free spaces (see Section V-B), and by using lazy evaluation for controller synthesis (see Section V-C). Furthermore, although $P_{\max}$ is exponential in the number of agents n, the proximity constraint dependency combined with connectivity constraints significantly decrease the number of polytopes in the space, as shown in Table II.

Although our algorithm does not require exact knowledge of the position of nonneighbor robots, all robots must have knowledge of the current simplex the state is located in to determine which controller to apply. This may seem infeasible; however, some limited information about position can be passed through neighbors in the connectivity graph. This would require state information to flow across the robot network but not at the bandwidth required for a complete information graph. Since exact position need not be known, any latency in the network will likely not result in a safety violation.

ACKNOWLEDGMENT

The authors would like to thank N. Michael, for use of his MATLAB API and for his assistance with dynamic simulations and experiments.

REFERENCES

[1] N. Ayanian and V. Kumar, "Decentralized feedback controllers for multiagent teams in environments with obstacles," in *Proc. IEEE Int. Conf. Robot. Autom.*, Pasadena, CA, May 2008, pp. 1936–1941.
[2] M. Egerstedt and X. Hu, "Formation constrained multi-agent control," *IEEE Trans. Robot. Autom.*, vol. 17, no. 6, pp. 947–951, Dec. 2001.
[3] R. Olfati-Saber and R. Murray, "Distributed cooperative control of multiple vehicle formations using structural potential functions," presented at the IFAC World Congr., Barcelona, Spain, Jul. 2002.
[4] J. Desai, J. Ostrowski, and V. Kumar, "Modeling and control of formations of nonholonomic mobile robots," *IEEE Trans. Robot. Autom.*, vol. 17, no. 6, pp. 905–908, Dec. 2001.
[5] N. Cowan, O. Shakerina, R. Vidal, and S. Sastry, "Vision-based follow-the-leader," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2003, vol. 2, pp. 1796–1801.
[6] O. Orqueda and R. Fierro, "Robust vision-based nonlinear formation control," in *Proc. Amer. Control Conf.*, Jun. 2006, pp. 1422–1427.
[7] M. A. Hsieh, S. G. Loizou, and V. Kumar, "Stabilization of multiple robots on stable orbits via local sensing," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2007, pp. 2312–2317.
[8] V. Gazi and K. Passino, "Stability analysis of social foraging swarms," *IEEE Trans. Syst., Man, Cybern.*, vol. 34, no. 1, pp. 539–557, Feb. 2004.
[9] C. Belta and V. Kumar, "Abstraction and control for groups of robots," *IEEE Trans. Robot.*, vol. 20, no. 5, pp. 865–875, Oct. 2004.
[10] N. Michael and V. Kumar, "Controlling shapes of ensembles of robots of finite size with nonholonomic constraints," in *Proc. Robot.: Sci. Syst.*, Jun. 2008, pp. 157–162.
[11] P. Yang, R. Freeman, and K. Lynch, "Multi-agent coordination by decentralized estimation and control," *IEEE Trans. Automat. Control*, vol. 53, no. 11, pp. 2480–2496, Dec. 2008.
[12] N. Ayanian and V. Kumar, "Abstractions and controllers for groups of robots in environments with obstacles," in *Proc. IEEE Conf. Robot. Autom.*, Anchorage, AK, May 2010, pp. 3537–3542.
[13] E. Rimon and D. Koditschek, "Exact robot navigation using artificial potential functions," *IEEE Trans. Robot. Autom.*, vol. 8, no. 5, pp. 501–518, Oct. 1992.
[14] D. Conner, A. Rizzi, and H. Choset, "Composition of local potential functions for global robot control and navigation," in *Proc. IEEE Int. Conf. Intell. Robots Syst.*, Las Vegas, NV, Oct. 2003, pp. 3546–3551.
[15] S. Lindemann and S. Lavalle, "Computing smooth feedback plans over cylindrical algebraic decompositions," in *Robotics: Science and Systems*, Cambridge, MA: MIT Press, Aug. 2006.
[16] S. Loizou and K. Kyriakopoulos, "Closed loop navigation for multiple non-holonomic vehicles," in *Proc. IEEE Int. Conf. Robot. Autom.*, Sep. 2003, vol. 3, pp. 4240–4245.
[17] D. Dimarogonas, M. Zavlanos, S. Loizou, and K. Kyriakopoulos, "Decentralized motion control of multiple holonomic agents under input constraints," in *Proc. IEEE Conf. Decis. Control*, Dec. 2003, vol. 4, pp. 3390–3395.
[18] S. Loizou, D. Dimarogonas, and K. Kyriakopoulos, "Decentralized feedback stabilization of multiple nonholonomic agents," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2004, vol. 3, pp. 3012–3017.
[19] S. R. Lindemann and S. M. LaValle, "Smoothly blending vector fields for global robot navigation," in *Proc. IEEE Conf. Decis. Control*, Seville, Spain, 2005, pp. 3553–3559.
[20] L. Habets and J. van Schuppen, "A control problem for affine dynamical systems on a full-dimensional polytope," *Automatica*, vol. 40, no. 1, no. 21–35, Jan. 2004.
[21] B. Roszak and M. E. Broucke, "Necessary and sufficient conditions for reachability on a simplex," *Automatica*, vol. 42, no. 11, pp. 1913–1918, Nov. 2006.
[22] S. M. Lavalle, *Planning Algorithms*. Cambridge, U.K.: Cambridge University Press, 2006.
[23] D. Conner, H. Choset, and A. Rizzi, "Provably correct navigation control for non-holonomic convex-bodied systems operating in cluttered environments," in *Robotics: Science and Systems*. Cambridge, MA: MIT Press, Aug. 2006.
[24] M. Kvasnica, P. Grieder, and M. Baoti. (2004). Multi-parametric toolbox (mpt) [Online]. Available: http://control.ee.ethz.ch/mpt/
[25] B. Gerkey, R. T. Vaughan, and A. Howard, "The player/stage project: Tools for multi-robot and distributed sensor systems," presented at the Int. Conf. Advanced Robotics, Coimbra, Portugal, Jun. 2003.
[26] N. Michael, personal communication, 2007, Univ. Penna.
[27] H. Khalil, *Nonlinear Systems*, 3rd ed. Upper Saddle River, NJ: Prentice-Hall, 2002.
[28] K. Fukuda. (2000). Frequently asked questions in polyhedral computation. [Online]. Available: http://www.ifor.math.ethz.ch/fukuda/polyfaq/polyfaq.html
[29] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed. Cambridge, MA/New York: MIT Press/McGraw-Hill, 2001.
[30] S. Boyd and L. Vandeberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

Nora Ayanian (S'07) received the B.Sc. degree in mechanical engineering from Drexel University, Philadelphia, PA, in 2005, and the M.S.E. degree from the University of Pennsylvania, Philadelphia, PA, in 2008, where she is currently working toward the Ph.D. degree in mechanical engineering and applied mechanics.

From 2008 to 2009, she was a Coinstructor with the University of Pennsylvania Summer Academy in Advanced Science and Technology Robotics Program. Her current research interests include multi-robot control in complex environments.

Ms. Ayanian is a member of the American Society of Mechanical Engineers and a National Science Foundation Fellow. She was the recipient of the 2008 IEEE International Conference on Robotics and Automation Best Student Paper Award.

Vijay Kumar (M'87–SM'02–F'05) received the M.S. and Ph.D. degrees in mechanical engineering from The Ohio State University, Columbus, in 1985 and 1987, respectively.

Since 1987, he has been with the Faculty of the Department of Mechanical Engineering and Applied Mechanics with a secondary appointment with the Department of Computer and Information Science, University of Pennsylvania, Philadelphia, where he is currently the UPS Foundation Professor and the Deputy Dean for education with the School of Engineering and Applied Science. From 2000 to 2004, he was the Deputy Dean with the School of Engineering and Applied Science.

Prof. Kumar has served on the editorial boards of the *Journal of the Franklin Institute*, the American Society of Mechanical Engineers (ASME) *Journal of Mechanical Design*, and the ASME *Journal of Mechanisms and Robotics*. He is a Fellow of the ASME. He has served on the editorial boards of the IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION and the IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING. He was the recipient of the 1991 National Science Foundation Presidential Young Investigator Award, the Lindback Award for Distinguished Teaching, the 1997 Freudenstein Award for significant accomplishments in mechanisms and robotics, and the 2004 IEEE International Conference on Robotics and Automation Kawamori Best Paper Award. He is also a Distinguished Lecturer with the IEEE Robotics and Automation Society and an Elected Member of the Robotics and Automation Society Administrative Committee.