2011 IEEE/RSJ International Conference on
Intelligent Robots and Systems
September 25-30, 2011. San Francisco, CA, USA

Synthesis of Feedback Controllers for Multiple Aerial Robots with Geometric Constraints

Nora Ayanian Vinutha Kallem Vijay Kumar

Abstract— **We address the problem of developing feedback controllers for a group of robots with second-order dynamics in an obstacle-filled, D-dimensional environment. Our control algorithm takes into account communication constraints, obstacle avoidance, and inter-robot collision avoidance, by synthesizing a piecewise smooth vector field for safe navigation. First, the feasible free joint configuration space is tessellated into polytopes that account for the desired constraints. We search the graph of these polytopes to find a discrete path to the goal polytope. We then use a novel navigation function-based feedback controller that drives the system from one polytope to the next and eventually to the goal. The controller exploits the fact that two adjoining polytopes in the planned discrete path together form a star-shaped object that is obstacle free; this enables the design of navigation function-based controller for kinematic and dynamic fully actuated robots without spurious minima. We sequentially compose these controllers to drive the state to the goal. For a polygonal space, the algorithm we propose is complete. We present successful simulation results of the algorithm on a group of ground vehicles and quadrotors performing a cooperative navigation task in constrained environments.**

I. Introduction

Multiple robots are being deployed to carry out a single task or mission in search and rescue, surveillance and mapping operations. Often, these robots must maintain proximity constraints for communication needs, avoid obstacles in the environment, and avoid inter-robot collisions while navigating as a group to reach desired locations. For example, in urban search operations, groups of quadrotors can be deployed to locate objects of interest while avoiding obstacles. Central to such applications is the development of feedback controllers to automatically generate inputs to the robots in known cluttered environments while maintaining safety.

Khatib [1] introduced potential functions for robot navigation to account for obstacle avoidance. While this method is simple to implement, it results in spurious minima and does not guarantee convergence. Rimon and Koditschek [2] synthesize navigation functions, and develop gradient descent nonlinear feedback controllers that guarantee safety (obstacle avoidance) and global convergence. In [3]–[5], navigation functions are developed for multi-robot problems. While these result in global convergence, designing potential functions is tedious for complex spaces as it involves mapping the workspace into sphere-worlds [6].

N. Ayanian, V. Kallem and V. Kumar are with the GRASP Laboratory, University of Pennsylvania, Philadelphia, PA 19104, USA. {nayanian,vkallem,kumar}@grasp.upenn.edu

We gratefully acknowledge support from NSF grants IIS-0427313 and IIP-0742304, ARO Grant W911NF-05-1-0219, ONR Grants N00014-07-1-0829, N00014-08-1-0696, and N00014-09-1-1051 and Lockheed Martin.

N. Ayanian gratefully acknowledges support from the NSF.

Hinging on the idea of sequential composition of controllers [7], in [8]–[13] the free configuration space is decomposed into obstacle-free cells and local controllers are synthesized in these cells to drive the robots sequentially to the goal. In [8], a Laplace heat equation is solved on a 2D disk, which is then mapped to a convex polygon. These controllers are composed to obtain a globally convergent controller for a single robot in the 2D polygonal workspace. This work is not applicable to multirobot problems without additional constraints, such as prohibiting multiple robots from occupying a cell simultaneously.

Lindemann and LaValle [9] smoothly blend "face vector fields" using bump functions to generate an essentially global smooth vector field to navigate a fully actuated robot to the goal. This work has been applied to multi-robot formation problems using a kinematic model [14]. Although the vector field of [9] can be used for a second-order system, careful tuning of the face vector fields may be required to guarantee robots do not escape the allowable space. The vector field approach, unlike the method presented herein, does not have a known Lyapunov function, making the second-order lift, like the one given in (9), not possible; this lift facilitates practical implementations on highly dynamical systems such as quadrotors as we will show in Section V. Necessary and sufficient conditions for a linear affine system to exit through a facet of a convex polytope or simplex have been presented in [10], [11]. Polytope controllers have been developed for a three-state planar cart or unicycle in [12], [15].

There has been extensive work on controlling inter-agent constraints, including maintaining formations and leader-follower control [16]–[20]. In general, however, these works do not guarantee that formation constraints are maintained in the presence of obstacles. Inter-agent repulsive forces can be used to avoid inter-robot collisions [21], [22], but convergence is not guaranteed in the presence of obstacles.

In the present work, we consider the problem of navigating a group of kinematic (first-order) and dynamic (second-order) robots in an obstacle filled polygonal environment while maintaining desired constraints. The algorithm we present takes a control-theoretic approach to developing a feedback law that provably guarantees convergence of a multi-robot system to a desired goal configuration in environments with obstacles. Further, in contrast to kinematic and geometric motion planning techniques, which plan a specific trajectory, we plan a path in a discrete representation of the free configuration space. In other words, the low-level controllers need only drive the system through a chain of full-dimensional polytopes in the free workspace instead of on a specific trajectory, making the controller robust

978-1-61284-455-8/11/$26.00 ©2011 IEEE

to uncertainties due to the process model (as in the case of quadrotors) and measurement noise. This work extends our previous work [13], [14] in two distinct ways. First, we consider robots with second-order dynamics to which the methods of [10], [13], [14] cannot be applied. In our method, first we use navigation functions to show convergence, then lift the controller to second-order systems, which not only guarantees convergence to the desired location but also emulates the velocity profile of the first-order system, enabling robust implementation on dynamical systems such as quadrotors (Section V). Second, the techniques described here for modeling configuration spaces and synthesizing controllers are thousands of times faster (on the order of seconds vs. hours) than in [13], allowing real-time planning and control for teams of aerial robots.

We present a smooth controller that drives a group of robots from one polytope to an adjoining polytope in D-dimensions for kinematic and dynamic systems, and a controller in the goal polytope in Section IV. We intersect the feasible free joint configuration space with the polygonal proximity constraints to generate the task configuration space in Section II. Using an informed graph search such as A*, we simultaneously generate a discrete representation of this space and find a path from the current polytope to the goal polytope in Section III; this section closely follows our formulations in [13], [14] and is presented here for completeness. Then we use our novel navigation function-based feedback controller (Section IV) to drive a group of robots from one polytope to the next and eventually to the goal. In Section V we present simulations of these algorithms on groups of dynamic ground robots and quadrotors (Ascending Technologies, Krailling, Germany) in urban environments.

II. Problem Formulation

Consider a group of n dynamic robotic agents $\mathcal{V}_A = \{a_i | i = 1, \ldots, n\}$ that must reach a goal configuration while maintaining constraints between agents and without colliding with each other or obstacles. The agent a_i has the state $\mathbf{x}_i = [x_i^1 \ x_i^2 \ \cdots \ x_i^{d_i}]^{\mathrm{T}} \in \mathbb{R}^{d_i}$ with the dynamics:

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

if the agent is kinematic or

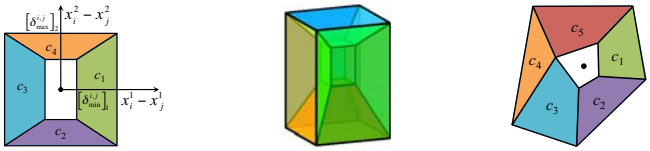
$$\ddot{\mathbf{x}}_i = \boldsymbol{\tau}_i, \ \mathbf{x}_i \in \mathbf{X}_i \subset \mathbb{R}^{d_i}, \ i = 1, \ldots, n, \tag{2}$$

if the agent is dynamic. Here we assume that the allowable inputs $\mathbf{u}_i, \boldsymbol{\tau}_i$ lie in $\mathbb{R}^{d_i}$ for simplicity of presentation; but if the allowable input space encloses a closed region around zero, the control law can be scaled appropriately.

The robots must maintain preset proximity constraints for collision avoidance and communication, which we associate with collision and connectivity graphs. We use halfspaces to define the constraints to generate a polytopic task configuration space. Recall that a *D-polytope* can be defined as the convex hull of finitely many vertices, or the intersection of N halfspaces, $P = \{\mathbf{x} \mid H\mathbf{x} \leq K\}$, where $H \in \mathbb{R}^{N \times D}, K \in \mathbb{R}^{N \times 1}$, where N is finite.

Definition 2.1: The *collision graph* on the set of agents is a static graph $G_L = (\mathcal{V}_A, \mathcal{E}_L)$ where $\mathcal{E}_L$ is the set of all pairs of agents which cannot occupy the same coordinates simultaneously. Pairs $(a_i, a_j) \in \mathcal{E}_L$ must maintain a nonzero minimum distance $|\mathbf{x}_i - \mathbf{x}_j| \geq \boldsymbol{\delta}_{\min}^{i,j}$, where $\boldsymbol{\delta}_{\min}^{i,j} \in \mathbb{R}^{\max(d_i, d_j)}$. This constraint can be written as

$$\lambda(\mathbf{x}_i, \mathbf{x}_j) = |\mathbf{x}_i - \mathbf{x}_j| - \boldsymbol{\delta}_{min}^{i,j} \geq \mathbf{0} \ \ \forall (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{E}_L.$$



(a) (b) (c)

Fig. 1. Example collision and connectivity constraints in the relative space of two agents in 2D and 3D. (a) 2D collision and connectivity constraints. (b) 3D collision and connectivity constraints. (c) Asymmetrical 2D constraints.

Definition 2.2: The *connectivity graph* on the set of agents is the static graph $G_N = (\mathcal{V}_A, \mathcal{E}_N)$ where $\mathcal{E}_N$ is the set of edges describing agent pairs $(a_i, a_j) \in \mathcal{E}_N$ that must maintain a maximum distance $|\mathbf{x}_i - \mathbf{x}_j| \leq \boldsymbol{\delta}_{\max}^{i,j}$, where $\boldsymbol{\delta}_{\max}^{i,j} \in \mathbb{R}^{\max(d_i, d_j)}$ to communicate state information. The constraint can be written as

$$\nu(\mathbf{x}_i, \mathbf{x}_j) = |\mathbf{x}_i - \mathbf{x}_j| - \boldsymbol{\delta}_{\max}^{i,j} \leq \mathbf{0} \ \ \forall (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{E}_N.$$

Figure 1 shows sample collision and connectivity constraints in the relative space of two agents in 2D and 3D. If no communication constraints are imposed, the regions will be infinite (i.e. $\boldsymbol{\delta}_{\max} = \infty$). The constraints need not be symmetric; any convex decomposition with matching facets is admissible. By matching facets we mean that any hyperplane supporting two adjacent polytopes shares the same vertices in both polytopes, and any hyperplane can only support one facet of that polytope. Although we will discuss constraints of the type $|\mathbf{x}_i - \mathbf{x}_j| \in [\boldsymbol{\delta}_{\min}^{i,j}, \boldsymbol{\delta}_{\max}^{i,j}]$, this framework can be used with any constraints defined as unions of non-overlapping polytopes with matching facets, such as those shown in 1(c), or excluding certain robots from or setting a maximum number of robots in a polygonal region.

Definition 2.3: The *configuration space* $\mathcal{C}_i$ of an agent a_i is the set of all transformations of the agent. The *free space* $\mathcal{C}_i^{free}$ of a_i is the set of all transformations of a_i which do not intersect with obstacles in the configuration space.

We assume $\mathcal{C}_i^{free}$ is tessellated into p_i polytopes with matching facets.

Definition 2.4: The *team configuration space* is the Cartesian product of the configuration spaces of each agent,

$$\mathcal{C}_1^{free} \times \mathcal{C}_2^{free} \times \cdots \times \mathcal{C}_n^{free} \equiv \mathcal{C}_{all} \ni \mathbf{x} \equiv [\mathbf{x}_1^{\mathrm{T}} \ \mathbf{x}_2^{\mathrm{T}} \ \cdots \ \mathbf{x}_n^{\mathrm{T}}]^{\mathrm{T}}.$$

Thus the configuration of all n agents is described by a single point $\mathbf{x} \in \mathcal{C}_{all} \in \mathbb{R}^D$, $D \equiv \sum_{i=1}^n d_i$.

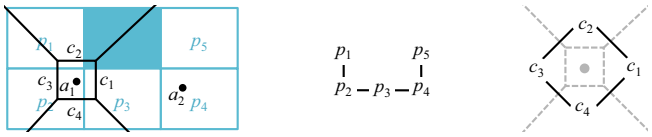
We can rewrite all of our constraints with respect to $\mathcal{C}_{all}$:

$$\begin{aligned} \mathcal{L} &\equiv \{\mathbf{x} | \mathbf{x} \in \mathcal{C}_{all}, \lambda(\mathbf{x}_i, \mathbf{x}_j) \geq 0 \ \forall (a_i, a_j) \in \mathcal{E}_L\}, \\ \mathcal{N} &\equiv \{\mathbf{x} | \mathbf{x} \in \mathcal{C}_{all}, \nu(\mathbf{x}_i, \mathbf{x}_j) \leq 0 \ \forall (a_i, a_j) \in \mathcal{E}_N\}. \end{aligned} \tag{3}$$

Problem 2.5: Consider a group of n robots with dynamics (1). For an initial state, find a piecewise smooth state feedback policy $\mathbf{u}(\mathbf{x})$ that drives the system to the goal configuration $\mathbf{x}^g$ such that:

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

Problem 2.6: Consider the above multirobot navigation problem with robot dynamics given by $\ddot{\mathbf{x}} = \boldsymbol{\tau}$.



(a) (b) (c)

Fig. 2. Discrete system pose example. (a) Two robots in free configuration space with proximity constraints. Discrete pose is $[p_2\, p_4\, c_1]$. (b) Adjacency graph on workspace. (c) Adjacency graph on agents' proximity constraints.

III. Task Configuration Space

To solve Problems 2.5, 2.6, we remove from $\mathcal{C}_{all}$ points that violate the constraints, and synthesize controllers on the resulting space to drive the system to the goal configuration.

Definition 3.1: The *task configuration space* $\mathcal{C}_T$ is the set

$$\mathcal{C}_T = \mathcal{C}_{all} \cap \mathcal{L} \cap \mathcal{N}. \tag{4}$$

$\mathcal{C}_T$ is a space composed of polytopes, in which robots cannot collide with obstacles or each other or lose communication.

We build a discrete representation of $\mathcal{C}_T$, on which we plan a path to the goal using an A* algorithm [23], then synthesize controllers on pairs of adjacent polytopes to reach the goal.

Definition 3.2: The *polytope graph* $G_P = (\mathcal{V}_P, \mathcal{E}_P)$ on the polytopes in $\mathcal{C}_T$ is the pair of sets $\mathcal{V}_P = \{P^0, P^1, P^2 \dots\}$, where P^m is the m-th polytope, and $\mathcal{E}_P = \{e_P^{m,k} | P^m \text{ is adjacent to } P^k,\ \forall m, k\}$, the set of all pairs of polytopes which share a (matching) facet.

Without loss of generality, let the state initially be in P^0.

Problem 3.3 (Discrete Path): For the node $P^0 \in \mathcal{V}_P$, find a path on the polytope graph G_P to the goal node P^N.

Since $\mathcal{C}_T$ is large and computing it may be prohibitively complex, we use an A* algorithm, using a heuristic defined by the adjacency graphs in $\mathcal{C}_i^{free}$ and the graph on the agents' proximity constraints, to solve Problem 3.3. The heuristic is identical to that in [14] but also includes the workspace cell of each robot. As an example, we use the two robots with the environment and constraints in Fig. 2(a). Figure 2(b) shows the corresponding adjacency graph on $\mathcal{C}_i^{free}$, and Fig. 2(c) the graph on the agents' proximity constraints. We generate a discrete system pose by concatenating the location (cell number) of each robot in its free space with the relative location of each pair of robots. Each polytope P^m in $\mathcal{C}_T$ corresponds to a unique discrete pose $\mathcal{F}_d^m$ of the system.

In Fig. 2(a), $\mathcal{F}_d = [p_2\, p_4\, c_1]$, where p_2 corresponds to the cell of a_1, p_4 to the cell of a_2, and c_1 to the location of a_2 with respect to a_1. Using the discrete pose and the adjacency graphs of the free spaces of the agents, we generate adjacent cells. For example, since p_2 is adjacent to p_1 and p_3, $[p_2 p_4 c_1]$ is adjacent to $[p_1\ p_4\ c_1]$, $[p_3\ p_4\ c_1]$. Similarly, since p_4 is adjacent to p_3 and p_5, $[p_2\, p_4\, c_1]$ is adjacent to $[p_2\, p_3\, c_1]$ and $[p_2\ p_5\ c_1]$. Finally, c_1 is adjacent to c_2 and c_4, generating potential adjacent polytopes corresponding to $[p_2\, p_4\, c_2]$ and $[p_2\, p_4\, c_4]$. However, in Fig. 2(a) it is clear these cells do not exist. When we expand $[p_2\ p_4\ c_1]$ in the A* algorithm, we check to ensure existence before adding it to the open set. Thus, finding a path on the polytope graph is equivalent to finding a path on the discrete poses.

A heuristic is admissible for A* if does not overestimate actual cost [23]. We assign a cost of 1 to each transition. We can also penalize polytopes corresponding to undesired configurations or size; this decreases search performance, but still results in an optimal discrete path.

Proposition 3.4: Problem 3.3 has a solution from P^0 to P^N if and only these nodes are connected on G_P.

Proof: $\mathcal{C}_T$ contains every allowable configuration in our polytopic world model. G_P contains all the information about connectivity of $\mathcal{C}_T$. Thus, if there is a solution to Problem 3.3, there must exist a path from P^0 to P^N on G_P. Conversely, if a path does not exist between the two nodes on G_P, there is no solution to Problem 3.3. ■

IV. Automatic Controller Synthesis

Without loss of generality, let the discrete path be given by $\{P^0, P^1, \cdots, P^N\}$, where P^N is the goal polytope. For $m \in \{0, 1, \dots, N-1\}$, P^m is in the discrete plan with successor cell P^{m+1} such that P^m and P^{m+1} share a facet. Hence the feedback controller in P^m should prepare the system for the controller in P^{m+1} to reach the goal cell. In the goal cell, P^N, the controller should drive the system to the goal configuration. Below we develop smooth controllers for these situations based on navigation functions on star-worlds [2].

A. Intermediate local cell controller

Definition 4.1: [2], [24] Let $\mathcal{Q}$ be an n-dimensional compact, simply connected manifold with boundary and let $\mathbf{q}_g \in \mathcal{Q}$ be a unique point. A function $\varphi : \mathcal{Q} \mapsto [0, 1]$ is a *navigation function* if it is twice differentiable on $\mathcal{Q}$, achieves a unique minimum of 0 at $\mathbf{q}_g \in \mathcal{Q}$, is uniformly maximal (i.e. evaluates to 1) on the boundary, and is Morse.

A feedback controller can be generated from a navigation function to drive a kinematic system ($\dot{\mathbf{x}} = \mathbf{u}$) to the goal, $\mathbf{q}^g$ while staying inside the manifold $\mathcal{Q}$ at all times and is given by $\mathbf{u} = -\nabla\varphi(\mathbf{x})$ [2].

We first consider the case of kinematic agents, where the equation of motion each robot is (1). Let the system currently lie in polytope P^1 and the next polytope is P^2. The group dynamics is given by concatenating the agents' dynamics

$$\dot{\mathbf{x}} = \mathbf{u},\ \mathbf{x} \in \mathcal{C}_T \subset \mathbb{R}^D. \tag{5}$$

Corollary 4.2: There exists a navigation function based controller that drives the system in (5) from a polytope P^1 to its adjoining polytope P^2.

Proof: The adjacent polytopes, P^1 and P^2 share a common facet. The union of these polytopes is a star-shaped object[1] whose center can be any point on the interior of the common facet. Let this star be denoted S^{12}. For example, if in Figure 3(a) the red polygon on the left is P^1 and blue polygon P^2, the union is of these is the star S^{12}.

Now, pick as local goal position, $\mathbf{x}^l \in P^2 \subset S^{12}$. Following the proof by Rimon and Koditschek [2], there exists a navigation function, φ, with $\mathbf{x}^l$ as the unique minimal point. As in navigation function literature, the controller

$$\mathbf{u} = -\nabla\varphi(\mathbf{x}) \tag{6}$$

will drive the system given by (5) to the local goal position $\mathbf{x}^l$. This means that the system must cross the common facet between P^1 and P^2, implying that this controller will asymptotically drive the system from P^1 to adjacent P^2. ■

[1]Star shaped sets are closed sets that consists of a "center point" from where any ray crosses the boundary of the set once and only once. They are topologically equivalent to spheres of the same dimension. Common examples of such sets include convex polygons, star polygons, and spheres.

B. Local controllers for robots with second-order dynamics

Now consider the case of dynamic agents (2). As before, let the system currently lie in polytope P^1, with P^2 next. The group dynamics is now given by

$$\ddot{\mathbf{x}} = \boldsymbol{\tau},\ \mathbf{x} \in \mathcal{C}_T \subset \mathbb{R}^D. \tag{7}$$

Given the construction of S^{12} and local goal position $\mathbf{x}^l \in P^2 \subset S^{12}$ as in the kinematic case, the same navigation function is valid for the second-order case. The control law

$$\boldsymbol{\tau} = -\nabla\varphi(\mathbf{x}) - \Gamma\dot{\mathbf{x}}, \tag{8}$$

where $\Gamma \in \mathbb{R}^{D\times D}$ is a positive definite matrix, drives the system to the goal $\mathbf{x}^l$ while remaining inside S^{12} at all times. Convergence of this controller can be shown by using $V(\mathbf{x}, \dot{\mathbf{x}}) = \varphi(\mathbf{x}) + \frac{1}{2}\dot{\mathbf{x}}^{\mathrm{T}}\dot{\mathbf{x}}$ as a Lyapunov function.

The above controller drives the second-order system to the goal, but the velocity profile will be different from the first-order system, $\dot{\mathbf{x}} = -\nabla\varphi(\mathbf{x})$. If a similar velocity profile as the first-order case is desired for the second-order case, the following controller can be used,

$$\boldsymbol{\tau} = -\Gamma\left(\dot{\mathbf{x}} + \nabla\varphi(\mathbf{x})\right) - \nabla\varphi(\mathbf{x}) - \frac{\partial^2\varphi(\mathbf{x})}{\partial\mathbf{x}^2}\,\dot{\mathbf{x}}, \tag{9}$$

where $\Gamma \in \mathbb{R}^{D\times D}$ is a positive definite matrix. Convergence of this controller can be shown by using $V(\mathbf{x}, \dot{\mathbf{x}}) = \varphi(\mathbf{x}) + \frac{1}{2}\left(\dot{\mathbf{x}} + \nabla\varphi(\mathbf{x})\right)^{\mathrm{T}}\left(\dot{\mathbf{x}} + \nabla\varphi(\mathbf{x})\right)$ as a Lyapunov function [25]. Such a choice is desired in highly dynamic systems such as quadrotors, as we demonstrate in Section V-C.

C. Goal-cell controller

Given the polygonal workspace, polygonal obstacles and proximity constraints described above, the controllers in IV-A can be used to drive the robot to the cell containing the goal state. Once the goal cell is reached, the controllers described above can be used with the star as the goal polytope itself and the local goal position being chosen as the desired goal configuration of the entire robotic agent system.

D. Construction of navigation functions

The efficient construction of navigation functions in large dimensions is key to using these local controllers. We propose one method which we have successfully implemented in a variety of simulations described in Section V. Consider a polytope with r facets with goal $\mathbf{x}^l$ inside. Let each of the facets be given by $l_i \equiv k_i - \mathbf{h}_i\mathbf{x} = 0$ for $i = 1, 2, \ldots, r$ where $\mathbf{h}_i\mathbf{x} - k_i$ is the i^{th} row of the halfspace representation of the polytope, so that $l_i > 0$ inside the polytope. The boundary of the polytope is given by $l_1 l_2 \cdots l_r = 0$. Note that for small $\epsilon > 0$, the set $\beta = l_1 l_2 \cdots l_r - \epsilon > 0$ is contained inside the polytope and better approximates the polytope as $\epsilon \to 0$. This allows us to choose the below navigation function

$$\varphi = ||\mathbf{x} - \mathbf{x}^l||^2 / \left(||\mathbf{x} - \mathbf{x}^l||^{2\mu} + \beta\right)^{\frac{1}{\mu}},\quad \mu \in \mathbb{R}^{++} \tag{10}$$

to drive the system to $\mathbf{x}^l$ while staying inside the polytope.

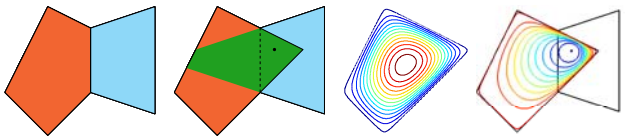
For the purpose of local controllers in Section IV-A, we need to construct navigation functions on the union of two adjoining convex polytopes, which is a star and need not be convex. However, since the polytopes have a matching facet at the interface between them, we can construct a convex polytope extension whose intersection with one of the polytopes is exactly that polytope. Consider adjacent polytopes P^m and P^{m+1} that share a matching facet. In other words, there exists a hyperplane that supports both polytopes, and at the interface, P^m and P^{m+1} share the exact same vertices. Let the halfspace representation of P^m and P^{m+1} be $H^m\mathbf{x} \le K^m$ and $H^{m+1}\mathbf{x} \le K^{m+1}$, respectively. Without loss of generality, assume the shared hyperplane is the first row in both H^m, K^m and H^{m+1}, K^{m+1}, so that $\mathbf{h}_1^m\mathbf{x} \le k_1^m$ and $\mathbf{h}_1^{m+1}\mathbf{x} \le k_1^{m+1}$. Then $\mathbf{h}_1^m = -\mathbf{h}_1^{m+1}$ and $k_1^m = -k_1^{m+1}$. Let H^m, K^m contain r^m rows, and H^{m+1}, K^{m+1} contain s^{m+1} rows. Construct the transitional polytope

$$T^m = \{\mathbf{x} \quad | \quad \mathbf{h}_r^m\mathbf{x} \le k_r^m, r = 2, \ldots, r^m, \\ \mathbf{h}_s^{m+1}\mathbf{x} \le k_s^{m+1}, s = 2, \ldots, s^{m+1}\}.$$

Note that T^m is a convex polytope such that $T^m \subset (P^m \cup P^{m+1})$ and $(P^{m+1} \cap T^m) \setminus \partial(P^{m+1} \cap T^m) \neq \emptyset$. Now construct the polytope M^m, which is a convex extension of polytope P^m into polytope P^{m+1}, by

$$M^m \equiv P^m \cup T^m. \tag{11}$$

We synthesize a navigation function on M^m, and by placing the local goal at $\mathbf{x}^l$ in the interior of $P^{m+1} \cap T^m$, we drive the system to the next polytope (the centroid or the center of the Chebyshev ball of the polytope $P^{m+1} \cap T^m$ are good choices). It is important to ensure that the local goal $\mathbf{x}^l$ is within the boundary of the approximation of the next polytope.



(a) (b) (c) (d)

Fig. 3. Constructing convex extensions of polytopes. (a) Adjacent polytopes P^m (orange, left) and P^{m+1} (blue, right). (b) Transitional polytope T^m (green), and local goal $\mathbf{x}^l$(black dot). The convex extension M^m of P^m is the union of the orange and green polytopes. (c) Smooth approximation of boundary of M^m. (d) Contour plot of navigation function (10) of M^m.

An example of constructing the extended polytope M^m, as well as the navigation function is shown in Fig. 3. This construction makes the regions of attractions of the individual controllers of the sequential composition overlap; this is an advantage over other non-overlapping sequential composition methods used in multi-robot problems as the velocity of the system will decrease as it approaches the goal. In dynamic systems, this lowers the risk of crossing the interface at high speed, and overshooting the next cell. These benefits are most strongly exhibited in microprocessor systems, where control is not truly continuous, and a slight lag in control input can result in violation of constraints.

E. Complete algorithm for multi-robot navigation

Algorithm 4.3: Construction of piecewise smooth controllers for kinematic and dynamic systems.

1) Find a path on G_P and construct $\mathcal{C}_T$ (Definition 3.1) simultaneously using heuristic-based graph search.
2) Construct polytopes M^m as in (11) for each polytope on the path except the goal polytope.
3) Synthesize navigation functions (10) in each polytope on the path.

Theorem 4.4: Algorithm 4.3 is complete based on the constraints (3).

Proof: For each pair of adjacent polytopes on the path, an extended polytope can be found, as in (11). Since $T^m \cap P^{m+1} \neq \emptyset$, there exists a point in P^{m+1} which can be chosen as the local goal $\mathbf{x}^l$. Since $\mathbf{x}^l \in (T^m - \partial T^m)$, $\exists\, \epsilon > 0$ such that $\mathbf{x}^l$ is guaranteed to be within the boundary of the approximation of both polygons. By definition, since there are no obstacles in the polytope M^m, a navigation function can be constructed without local minima which drives the state to the next polytope. In the goal polytope, a navigation function can be constructed without local minima which drives the state to the goal configuration $\mathbf{x}^g$. Therefore, if a path exists, we are guaranteed to find a controller to drive the system to the goal configuration. By Theorem 3.4, we are guaranteed to find a path if one exists given the constraints (3). Therefore, Algorithm 4.3 is complete. ■

V. Simulation Results

In this section we present several illustrative examples of the successful application of the controller to groups of unmanned ground vehicles (UGVs), unmanned aerial vehicles (UAVs) and quadrotors. Simulations are conducted in MATLAB, using second-order models for all robots. These simulations are included on the supplemental video. For UGVs and quadrotors, we use the model

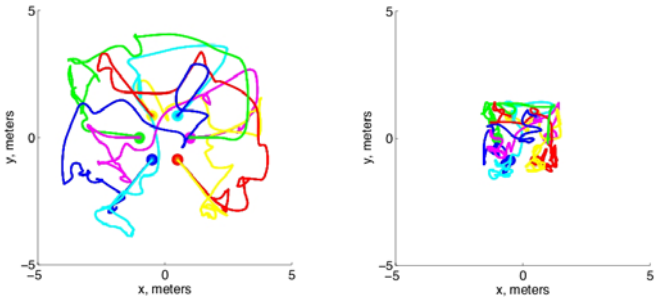
$$\ddot{\mathbf{x}}_i = \boldsymbol{\tau}_i,\ \mathbf{x}_i \in \mathbb{R}^{d_i}, \tag{12}$$

where $d_i = 2$ for UGVs and $d_i = 3$ for quadrotors. Since a quadrotor is a 12-state dynamical system (three each of positions, linear velocities, orientation, and angular velocities) with four inputs (rotor velocities) [26], using the full model would be prohibitively complex for multiple quadrotor navigation. Therefore, the control for quadrotors is designed for (12), an abstraction of the full 12-state dynamical model. The abstraction is obtained by linearization of the full 12-state model around the nominal hover state, generating the control input τ of (8) or (9), as is desired. The input τ is then converted into motor thrusts and moments following [26]. Because the abstraction is not an exact model of the full system, it is important for the control algorithm to be robust to modeling errors. As can be observed in Sections V-B, and V-C, our method is robust to these errors due to its cell-based nature as well as the second-order lift using Lyapunov functions. To achieve faster convergence, once inside the next polytope, we switch the controller after ensuring ground robots were sufficiently close to the local goal and quadrotor velocity was below a specified threshold.

Proximity constraints for all simulations are $\delta_{\min} = 0.5$m in the x, y plane, while on the z axis, the minimum clearance above is 0.5m and below is 1m, to compensate for air flow between quadrotors. We ran the control loop at 100Hz. All simulations were done on a MacBook Pro, with 2.53 GHz Intel Core 2 Duo processor, 4GB 1067 MHz RAM, running MATLAB R2009b. Polytope computations were done using the MultiParametric Toolbox [27].

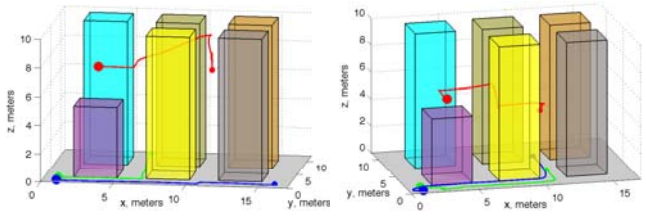
A. Six robots in a circle

Figure 4 shows six ground robots switch to positions across from their current position while avoiding collisions.



(a) (b)

Fig. 4. Six ground robots (2D) in a circle switch to opposite positions. (a) No communication constraints enforced. (b) Maximum distance of 3 m communication constraints enforced.



(a) (b)

Fig. 5. A quadrotor (red) and UGV (green, blue) simulation in an urban environment. The robots are deployed from the left front corner. (a) The robots are deployed to different intersections without communication constraints. (b) The robots navigate to the same intersection while maintaining a maximum horizontal distance of 6.5m between all pairs of robots.

The simulation in Fig. 4(a) is not communication constrained, while that in Fig. 4(b) has a maximum communication distance of 3m for each pair of robots. Robots start at the large dot and end at the smaller dot. Here, we favor large polytopes in the A* search with the transition cost

$$1 + L_{\max}/\rho L_{\min} \tag{13}$$

for each polytope, where $L_{\max}$ and $L_{\min}$ are the largest and smallest lengths, respectively, of the smallest hyperrectangle containing the next polytope, and ρ is the radius of the largest ball inscribed in the next polytope. Note that while this cost penalizes small "skinny" polytopes, it does not remove any polytopes from $\mathcal{C}_T$. For the heuristic we use the cost of 1 per transition, which is admissible since each transition has at least a cost of 1, increasing precomputation time (146s and 194s for the simulations without and with communications constraints, respectively) because of heuristic accuracy. A more accurate heuristic would lead to faster computations.

B. Two ground robots and one quadrotor

Figure 5 shows two examples of the application of our controller to simulations on groups of cooperating heterogeneous robots in an urban environment, with and without communication constraints. Precomputation time was about 11s for both simulations. In Fig. 5(a) the robots deploy to different intersections without communication constraints. Figure 5(b) correspond to a simulation with communication constraints enforced. Robots must maintain a maximum distance of 6.5m in x, y, and z. The trajectory of the 2 UGVs are in green and blue and that of the quadrotor is in red.

C. Three quadrotors through a window

Figure 6 shows the application of our controller to a simulation on three cooperating quadrotors that must switch to

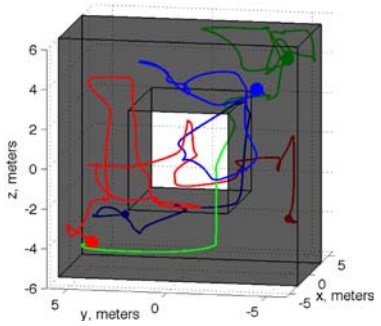


Fig. 6. Three quadrotors must switch to opposite sides of a window. Each quadrotor is represented by a different color in green, blue and red.

opposite sides of a window. We enforce collision constraints, however no communication constraints are considered in this simulation. Because quadrotors are highly dynamic and the process model has uncertainties, we again favor large polytopes in the A* search using the cost (13), increasing the precomputation time to 50s.

VI. Discussion

We have presented a novel navigation function-based feedback controller to drive a group of robots with second-order dynamics in a constrained environment to a goal. We decompose the workspace into obstacle-free polytopes and generate local smooth feedback laws that drive a single robot or a team of robots from one cell to an adjoining cell. These local controllers are then sequenced using discrete graph search methods like A* or incremental D* to reach the goal. Our method is complete: it is guaranteed to find a solution if one exists. The complexity of the algorithm depends on the number of robots and the complexity of the free space. By using the heuristic derived in Section III for discrete planning we significantly reduce the computational burden.

We have successfully applied our algorithm to teams of second-order ground robots and quadrotors. Since the controller is free of local minima, tuning is not required but improves performance in discrete implementations of control loops. The tessellation of the workspace as well as the inter-robot constraints influence the trajectory of the system through the configuration space. Since the state transitions through polytope facets, the path may appear unintuitive. This can be improved by choosing the goal point while taking into account group objectives instead of choosing the Chebyshev center.

We plan to conduct experiments on heterogeneous groups of UAV and UGV platforms, such as quadrotors and SCARABs, working cooperatively. We are currently pursuing extensions of this work so as to directly apply a similar controller framework to nonholonomic systems and groups of nonholonomic systems while guaranteeing correctness and completeness of the control algorithm.

VII. Acknowledgments

The authors thank Profs. Daniel Koditschek and Noah Cowan for their valuable insights on construction of navigation functions on manifolds with corners and Daniel Mellinger for sharing the model for quadrotors.

References

[1] O. Khatib, "Commande dynamique dans l'espace operationnel des robots manipulateurs en presence d'obstacles," Ph.D. dissertation, L'Ecole Nationale Superieure de l'Aeronautique et de l'Espace, 1980.
[2] E. Rimon and D. E. Koditschek, "Exact robot navigation using artificial potential fields," *IEEE Trans. Robot. Autom.*, vol. 8, no. 5, pp. 501–518, 1992.
[3] S. Loizou and K. Kyriakopoulos, "Closed loop navigation for multiple non-holonomic vehicles," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2003, pp. 4240–4245.
[4] S. Loizou, D. Dimarogonas, and K. Kyriakopoulos, "Decentralized feedback stabilization of multiple nonholonomic agents," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2004, pp. 3012–3017.
[5] D. Dimarogonas, M. Zavlanos, S. Loizou, and K. Kyriakopoulos, "Decentralized motion control of multiple holonomic agents under input constraints," in *Proc. Conf. Dec. Contr.*, 2003, pp. 3390–3395.
[6] E. Rimon and D. E. Koditschek, "The construction of analytic diffeomorphisms for exact robot navigation on star worlds," *Trans. American Mathematical Society*, vol. 327, no. 1, pp. 71–115, 1991.
[7] R. R. Burridge, A. A. Rizzi, and D. E. Koditschek, "Sequential composition of dynamically dexterous robot behaviors," *Int. J. Robot. Res.*, vol. 18, no. 6, pp. 534–555, 1999.
[8] D. C. Conner, H. Choset, and A. Rizzi, "Flow-through policies for hybrid controller synthesis applied to fully actuated systems," *IEEE Trans. Robot.*, vol. 25, pp. 136–146, 2009.
[9] S. R. Lindemann and S. M. LaValle, "Smoothly blending vector fields for global robot navigation," in *Proc. Conf. Decision and Control*, 2005, pp. 3353–3559.
[10] L. C. G. J. M. Habets and J. H. van Schuppen, "A control problem for affine dynamical systems on a full-dimensional polytope," *Automatica*, vol. 40, no. 1, pp. 21–35, 2004.
[11] B. Roszak and M. Broucke, "Necessary and sufficient conditions for reachability on a simplex," *Automatica*, vol. 42, no. 11, pp. 1913–1918, 2006.
[12] V. Kallem, A. T. Komoroski, and V. Kumar, "Sequential composition for navigating a nonholonomic cart in the presence of obstacles," *IEEE Trans. Robot.*, 2011, in press.
[13] N. Ayanian and V. Kumar, "Decentralized feedback controllers for multi-agent teams in environments with obstacles," *IEEE Trans. Robot.*, vol. 26, pp. 878–887, October 2010.
[14] N. Ayanian, V. Kumar, and D. Koditschek, "Synthesis of controllers to create, maintain, and reconfigure robot formations with communication constraints," in *Proc. Int. Symp. Robot. Research*, Aug. 2009.
[15] S. R. Lindemann, I. I. Hussein, and S. M. LaValle, "Real time feedback control for nonholonomic mobile robots with obstacles," in *Proc. Conf. Decision and Control*, 2006, pp. 2406–2411.
[16] M. Egerstedt and X. Hu, "Formation constrained multi-agent control," *IEEE Trans. Rob. Autom.*, vol. 17, no. 6, pp. 947–951, Dec 2001.
[17] R. Olfati-Saber and R. Murray, "Distributed cooperative control of multiple vehicle formations using structural potential functions," in *Proc. of IFAC World Congress*, Barcelona, July 2002.
[18] J. Desai, J. Ostrowski, and V. Kumar, "Modeling and control of formations of nonholonomic mobile robots," *IEEE Trans. Rob. Autom.*, vol. 17, no. 6, pp. 905–908, Dec 2001.
[19] N. J. Cowan, O. Shakernia, R. Vidal, and S. Sastry, "Vision-based formation control," in *Intelligent Robots and Systems (IROS)*. Las Vegas, NV: IEEE/RSJ, Oct. 2003.
[20] O. Orqueda and R. Fierro, "Robust vision-based nonlinear formation control," in *Proc. American Control Conf.*, June 2006.
[21] M. A. Hsieh, V. Kumar, and L. Chaimowicz, "Decentralized controllers for shape generation with robotic swarms," *Robotica*, vol. 26, no. 5, pp. 691–701, September 2008.
[22] V. Gazi and K. Passino, "Stability analysis of social foraging swarms," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 34, no. 1, Feb. 2004.
[23] P. Hart, N. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100 –107, july 1968.
[24] N. J. Cowan, "Navigation functions on cross product spaces," *IEEE Trans. Autom. Control*, vol. 52, no. 7, pp. 1297–1302, 2007.
[25] D. Koditschek, "Adaptive techniques for mechanical systems." in *Fifth Yale Workshop on Applications of Adaptive Systems Theory*, May 1987, pp. 259–265.
[26] N. Michael, D. Mellinger, Q. Lindsey, and V. Kumar, "The grasp multiple micro-uav testbed," *IEEE Robot. Autom. Mag.*, vol. 17, no. 3, pp. 56 –65, sep. 2010.
[27] M. Kvasnica, P. Grieder, and M. Baotić, "Multi-Parametric Toolbox (MPT)," 2004. [Online]. Available: http://control.ee.ethz.ch/~mpt/