2010 IEEE International Conference on Robotics and Automation
Anchorage Convention District
May 3-8, 2010, Anchorage, Alaska, USA

Abstractions and Controllers for Groups of Robots in Environments with Obstacles

Nora Ayanian and Vijay Kumar

Abstract— **We address the problem of controlling a formation of robots in a cluttered environment. Instead of explicitly controlling the relative positions between the robots and the environment, we construct a lower-dimensional abstraction of the group that establishes a boundary for the group. We then synthesize feedback controllers that allow the abstracted group to navigate a two-dimensional environment to a desired goal position, while automatically adapting the shape of the boundary as well as the position and orientation of the group to avoid collisions between the virtual boundary and the environment. In contrast to previous approaches, we address the planning and control problems concurrently and are naturally able to establish bounds on the positions of the robots through the abstraction. The complexity of the method is *independent* of the number of robots which promises scalability to large teams.**

I. INTRODUCTION

There are many applications in which groups of robots are required to navigate complex environments while maintaining desired proximity constraints either for sensing or communication, while avoiding collisions. In automated warehouse systems, for example, it may be necessary to perform a wide range of tasks ranging from automated storage and retrieval to machine tending with one or more robots [1]. In search and rescue applications, robots may need to navigate city streets or inside buildings in formations [2], [3]. In security or defense applications robots may need to maintain coverage or persistent surveillance in urban environments [4]. In applications involving manipulation tasks, robots might need to work together to manipulate an object while transporting it through the environment. Robots must keep in close proximity of each other and of the object, in order to prevent the object from escaping [5]. All these tasks generally occur in complex environments where having robots determine individual controllers to get to a goal location could result in added expense, loss of communication, and may result in deadlock. In these situations it makes sense to address the problem of group navigation separately from the problem of interactions between robot group members.

We address the problem of navigating a large group of robots through an obstacle-filled environment in a scalable fashion. We take a hierarchical approach to this problem to reduce complexity, by using an *abstraction* of the group of robots to reduce the dimensionality of the group navigation problem. The construction of an abstraction establishes a boundary for the group, allowing us to reformulate the group navigation problem as a problem of planning and controlling the shape, position and orientation of this boundary while avoiding collisions of this boundary with the environment.

Many authors have addressed the problem of multi-robot control in complex environments. Approaches to group navigation problems include formations, leader-follower schemes, abstractions, potential field methods, and flocking.

By restricting relative positions of robots, formations and leader-follower schemes reduce the complexity of group navigation problems [6]–[17]. Much of this work does not guarantee that formation constraints are maintained in the presence of obstacles [6]–[14].

The idea of constructing abstractions to reduce the dimensionality of the problem is not new. In [18]–[20], the central idea is to develop a surjection from the high-dimensional, multi-robot state space to a low-dimensional abstract space which characterizes the statistics of the group as well as the position and orientation of the group. Simple controllers and estimators can be designed to estimate the statistics and the pose, while driving these variables to desired values. However, these methods do not establish bounds on the positions of the robots. Neither can enforce constraints on network topology or formation which can change as the group moves. In [18], safety is not guaranteed: robots can collide and escape from the abstraction. Some limitations of [18] are addressed in [19], which still does not enable us to specify formations in the sense of exact shape and topology. A particular formation can be specified in [20], but the number of moments which must be supplied to specify a particular formation increases with the number of robots, and the method is not entirely automated.

Flocking or schooling strategies enable control of large groups of robots with relatively little computation [21]. The entire group's velocity is stabilized to a single velocity by each agent adjusting their velocity according to its neighbors. However, in the presence of obstacles, no guarantees are made. In [22], large groups of robots are stabilized to shapes. Again, the presence of obstacles in the workspace could cause local minima or deadlock.

With small group sizes, one can provide some guarantees while taking advantage of some reduction in complexity. In [23], proofs are provided for creating and maintaining formations, but collision avoidance is guaranteed only in most cases, requiring careful parameter choices. In [24], undesirable local minima can occur if sufficient virtual leaders are not added. In [25] a method is proposed for creating

We gratefully acknowledge support from NSF grant no. IIS-0427313, ARO grant no. W911NF-05-1-0219, ONR grants no. N00014-07-1-0829 and N00014-08-1-0696, and ARL grant no. W911NF-08-2-0004.

N. Ayanian and V. Kumar are with the GRASP Laboratory, Department of Mechanical Engineering and Applied Mechanics, University of Pennsylvania, Philadelphia, PA 19104, USA {nayanian,kumar}@grasp.upenn.edu

N. Ayanian gratefully acknowledges support from the NSF.

a formation and maintaining it during motion. However, no guarantees are made in the presence of obstacles, and formations are unlabelled. For small groups, [17], [26] provide guarantees of safety and maintaining desired constraints, however, these become intractable as the group grows.

In contrast to the papers above, we address the planning and control problems concurrently and are naturally able to establish bounds on the positions of the robots through the abstraction. This allows us to guarantee safety — the controllers are designed so that the virtual boundary associated with the abstractions does not pass through obstacles. The inter-robot constraints and specifications are satisfied by robot controllers that are designed to satisfy these constraints and specifications without exiting the boundary of the abstraction. In this way, the complexity of the problem of synthesizing controllers is *independent* of the number of robots which promises scalability to large groups.

II. PROBLEM FORMULATION

Consider a group $\mathcal{G}$, of N kinematic agents $V_R = \{r_j | j = 1, \ldots, N\}$. A group consists of a small number of robots which can communicate with each other at high bandwidth, enabling centralization (adopted from [27]). The group must navigate an obstacle-filled environment to a specified task location. Each agent has the state $x_j \in \mathbb{R}^2$ with dynamics:

$$\dot{x}_j = U_j, \; x_j \in X_j \subset \mathbb{R}^2, \; j = 1, \ldots, N. \tag{1}$$

We assume that within a group, communication occurs very rapidly, so that control can be centralized over the group.

An abstraction of the group of robots reduces the computational complexity of the problem. Our abstraction defines a virtual adaptive *boundary* for the group of robots, while the controllers at the robot level ensure the robots stay within the boundary. The abstraction boundary is determined by automatically synthesized controllers, and the robots react to the changing size and shape of the abstraction boundary. Thus, a group of robots can navigate a space knowing only the boundary and the local state of the group, decoupling planning and control of the agents from the physical space. The dimension of the abstraction is independent of the number of robots N. The specific abstraction we use is discussed in detail in Section III.

Figure 1 is a graphical representation of the hierarchichal structure of our approach. At the bottom level, individual robots execute the continuous controllers that are designed to satisfy specifications and desired formation properties. At the middle level, individual robots interact with each other to maintain constraints (this level may not be necessary for all problems). At the top level, the abstracted group navigates the space while maintaining constraints designed to ensure there is enough room for the robots. We assume that there are no obstacles within the group boundary. However, if an obstacle appears in the boundary, [28] provides a framework for dividing and reconfiguring the group in another location.

The input to each agent in the global reference frame is

$$U_j = R(u_A^\theta) \; u_j + u_A^{x,y}, \tag{2}$$

where $u_A^{x,y}$ is the translational and u_A^θ the angular component of the abstraction input, and $R(\cdot)$ is a rotation matrix, and u_j is the individual input of robot r_j.

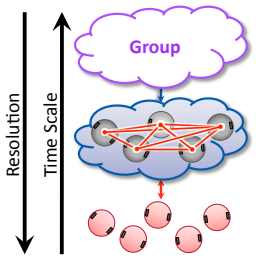


Fig. 1: Hierarchical structure. At the bottom level, robots implement the continuous controller. At the top level, the abstracted group navigates the space.

Our hierarchical control system enables a multiple time scale approach, as shown in Fig. 1, to eliminate the possibility of local minima which may occur when summing two controllers. Group dynamics (motion within a group) are assumed to evolve on a much faster time-scale than abstraction dynamics (overall motion of an entire group); sufficient time scale separation between group and abstraction dynamics ensures the two controllers can be designed independently.

Since the abstraction is independent of the robots' configuration, synthesis of and planning in the abstraction workspace occurs in advance. At least one robot must have knowledge of the evolution of the abstraction over time. This information, as well as individual robot state information, propagates through the group rapidly via explicit communication.

III. GEOMETRIC ABSTRACTION

The abstraction enables the group of robots to navigate the space in an obstacle-free environment limited by the abstraction boundary. Group navigation is handled by a high-level controller, which treats the abstraction as a single robot. This controller is then summed to the individual controllers as in (2). We use a rectangle for the boundary, however, this is only one example of the possible choices for abstractions.

Definition 3.1: The *group abstraction* A is a triple

$$A = (x_A, \theta, s) \in SE(2) \times \mathbb{R}^2, \tag{3}$$

where x_A is the center of the group abstraction, θ is the angular orientation, and s is a shape vector representing the boundary and size of the abstraction which encloses the group of robots. We assign the abstraction dynamics

$$\dot{x}_A = u_A, \tag{4}$$

where u_A can be considered a *virtual input*.

In this work we choose a rectangle as the abstraction boundary since a rectangle can be described by two parameters, width s_w and height s_h, so that the shape vector is the pair $s = (s_w, s_h)$. Although we choose a rectangle, it is important to note that the rectangle can be replaced by any convex polytopic shape, provided the shape vector contains enough information to describe a unique boundary.

We treat the abstraction as a single robot which can change shape and orientation while navigating the space. *Shape constraints* limit the size and shape of the abstraction in order to ensure we have enough room for the number of robots in the group. We set bounds on s_w and s_h, as well as bounds on perimeter, $(s_w + s_h)$. We can write these constraints:

$$H_s s \leq K_s. \tag{5}$$

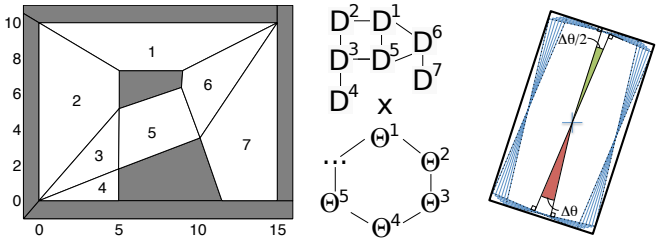


Fig. 2: (a) A decomposition of the workspace into districts (Obstacles are shaded) (b) We take the Cartesian Product of each district D^m with each θ-slice Θ^k. (c) Our choice for overestimating the abstraction.

Although perimeter bounds are a less intuitive choice than area, perimeter is a linear constraint, and area is not. The reason for linear constraints will become clearer in Section IV.

IV. ABSTRACT CONFIGURATION SPACE

We would like to drive the abstraction through the physical workspace using a controller that can be synthesized automatically. To that end, we build a polytopic configuration space based on the parameters which define the abstraction.

Definition 4.1: The *configuration space* $\mathcal{C}$ of the abstraction A is the set of all transformations, including rotations, of A. The *free space* $\mathcal{C}^{free}$ of A is the set of all transformations, including rotations, of A which do not intersect with obstacles in the configuration space.

To simplify building and planning on $\mathcal{C}^{free}$, we take a hierarchical approach to constructing it. First, slice the angular component of the configuration space into θ-slices, $\Theta^k = [(k-1)\Delta\theta, k\Delta\theta], k = \{1,\dots,q\}$, so that $q\Delta\theta = 2\pi$, and tesselate the $\mathbb{R}^2$ workspace into districts $D^m, m \in \{1,\dots,d\}$, (Fig. 2a). Each district is described by constraints:

$$H_D^m x_A \leq K_D^m,\ m \in \{1,\dots,d\}. \tag{6}$$

Representing the free space exactly is not possible with a finite number of polytopes, as the free space includes curves due to rotations of the abstraction. Therefore, we seek to underestimate $\mathcal{C}^{free}$ with $\mathcal{C}_A^{free}$, such that $\mathcal{C}_A^{free} \subset \mathcal{C}^{free}$. We do this by overestimating the abstraction with A_{Θ^k}:

$$A_{\Theta^k} \supset \{A(x_A,\theta,s) | A \in \mathcal{C}_A^{free}, \theta \in [(k-1)\Delta\theta, k\Delta\theta], \\ k = 1,\dots,q-1\}.$$

A_{Θ^k} must be a union of convex polygons. The vertices of A_{Θ^k} must be linear functions of x_A and s, and the outward normals must not be an explicit function of s. This is necessary for computing the applicability conditions for types A and B contact [29].

We choose to overestimate the abstraction with a rectangular superset of the abstractions through an angular interval $\Delta\theta$. Figure 2c shows the abstraction that we choose, which is a rectangle rotated to angle $\Delta\theta/2$ (smallest rectangular superset of the abstractions). The overestimate is shown for small $\Delta\theta$. With large $\Delta\theta$ this overestimate becomes excessive, so another overestimate should be chosen.

Workspace obstacles $O^l, l \in \{1,\dots,o\}$ must be represented as a finite unions of convex polygons (note that the boundary of the workspace is considered an obstacle). Obstacles in the workspace map to C-obstacles in the configuration space for each θ-slice Θ^k. The C-obstacle for each θ-slice is computed using the specific overestimate for that θ-slice.

To construct C-obstacles, we follow an algorithm originally proposed by Lózano-Perez [30] and discussed by Latombe [29]. To do so, we must project the vertices and outward normals of the boundary into polyhedral space, thus the outward normals cannot depend on the shape vector. Here we follow Latombe's notation closely.

Let the vertices of the overestimated abstraction be represented $a_j^k(x_A)$, where $j = 1,\dots,4$ for θ-slice Θ^k, and the outward normal for the facet between $a_j^k(x_A)$ and $a_{j+1}^k(x_A)$ as $\overrightarrow{v}_j^{A,k}(x_A)$. Similarly, let the vertices of obstacle O^l be represented b_j^l, with the outward normal for the facet between b_j^l and b_{j+1}^l as $\overrightarrow{v}_j^{B^l}$.

The C-obstacles are computed by calculating for each θ-slice Θ^k the applicability conditions for type A contact (we drop the superscripts k and l for clarity),

$$\begin{aligned} \texttt{APPL}_{i,j}^A(x_A) \quad = \quad & \left[\overrightarrow{v}_i^A(x_A)\cdot(b_{j-1}-b_j) \geq 0\right] \\ & \bigwedge \left[\overrightarrow{v}_i^A(x_A)\cdot(b_{j+1}-b_j) \geq 0\right]. \end{aligned} \tag{7}$$

If $\texttt{APPL}_{\texttt{i,j}}^{\texttt{A}}(\texttt{x}_\texttt{A})$ holds, then add to C-obstacle the constraint

$$f_{i,j}^A(x_A) \equiv \overrightarrow{v}_i^A(x_A)\cdot(b_j - a_i(x_A)) \leq 0. \tag{8}$$

Similarly, we calculate

$$\begin{aligned} \texttt{APPL}_{i,j}^B \quad = \quad & \left[(a_{i-1}(x_A)-a_i(x_A))\cdot\overrightarrow{v}_j^B \geq 0\right] \\ & \bigwedge \left[(a_{i+1}(x_A)-a_i(x_A))\cdot\overrightarrow{v}_j^B \geq 0\right]. \end{aligned} \tag{9}$$

If $\texttt{APPL}_{\texttt{i,j}}^{\texttt{B}}(\texttt{x}_\texttt{A})$ holds, then add to C-obstacle the constraint

$$f_{i,j}^B(x_A) \equiv \overrightarrow{v}_j^B\cdot(a_i(x_A)-b_j) \leq 0. \tag{10}$$

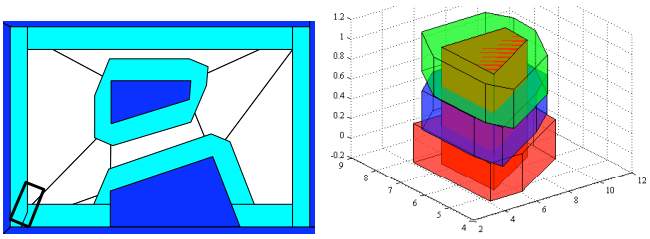
For more details on the applicability condition and the constraints $f_{i,j}^A(x_A)$ and $f_{i,j}^B(x_A)$, refer to [29].

This generates obstacles in $\mathbb{R}^4 \times \mathbb{N}$ for each θ-slice, which we can extrude through the interval Θ_k to get $O^{l,k} \in SE(2)\times\mathbb{R}^2$. We can represent $O^{l,k}$

$$H_O^{l,k} x_A \leq K_O^{l,k}, k \in \{1,\dots,q\}. \tag{11}$$

As depicted in Fig. 3b, the C-obstacles will generally not coincide at the θ-slice interfaces. Since our controller requires facets to match on adjacent polytopes, we reconcile this by considering each interval Θ_k with corresponding $\mathbb{R}^4$ obstacles, and any intersecting districts. In each district, before removing obstacles, we extend all obstacle hyperplane in the intersection of the obstacle and the district for the adjacent θ-slices. For example, in the 3rd θ-slice in district 7, we intersect the polytopes with the extended hyperplanes for the 2nd, 3rd, and 4th θ-slices. This results in a union of polytopes in 4-dimensions for each district and θ-slice.

Finally, the resulting polytopes in each district are extruded into their θ-slices to construct *θ-districts*: $D_m^{\Theta_k}, m \in \{1,\dots,d\}, k \in \{1,\dots,q\}$. In each θ-district we remove the polytopes which intersect with obstacles. Since the supporting hyperplanes of the intersections of the 4-d districts with every obstacle in every θ-slice were extended into the district before extending into θ-space, the polytopes within each θ-district have matching facets. Furthermore, any pair of polytopes across a θ-slice interface which are adjacent



(a) (b)

Fig. 3: Configuration space obstacles for θ-slices. (a) C-obstacles generated for a rectangular abstraction with fixed length, width, and angle (shown in the bottom left corner). (b) C-obstacles for θ-slices do not coincide on the interface.

will have matching vertices, ensuring that we cannot drive the abstraction into prohibited regions.

Thus, the overestimated free space is $\mathcal{C}_A^{free}$:

$$C_A^{free} = \bigcup_{k=1}^{q} \bigcup_{m=1}^{d} C_A^{free,k,m}, \tag{12}$$

$$C_A^{free,k,m} = \bigcap_{l=1}^{o} \left(D_m^{\Theta_k} \cap O^{l,k} \right).$$

$\mathcal{C}_A^{free}$ is a union of polytopes in which the abstraction cannot collide with any obstacles or the boundary of the space.

Problem 4.2: Consider the system (4) and goal group abstraction state x_A^g. Find an input function $u_A : [0, T_0] \to \mathcal{U}_A$ for some initial group abstraction state $x_A^0 \in C_A^{free,k^0,m^0} \subset \mathcal{C}_A^{free}$ such that

1) for all time $t \in [0, T_0]$, $x_A \in \mathcal{C}_A^{free}$ and $x_A(T_0)$ arbitrarily close to x_A^g,
2) $\dot{x}_A = u_A$.

A. Feedback controllers on $\mathcal{C}_A^{free}$

We now develop feedback controllers to solve Problem 4.2 and drive the abstraction to the goal location. A key step is to define a hierarchical discrete representation of the free space. Then we translate these paths into feedback controllers.

Each polytope in $\mathcal{C}_A^{free}$ is associated with a θ-district $D_m^{\Theta_k}$. We define a two adjacency graphs, one on $\mathcal{C}_A^{free}$ at the θ-district level, and one on each θ-district at the polytope level.

Definition 4.3: The *upper-adjacency graph* on $\mathcal{C}_A^{free}$ is the triple $G_U = (V_U, E_U, C_U)$, where $V_U = \{[1\ 1], [1\ 2], \ldots, [q\ d]\}$, E_U is the set of all pairs $(D_m^{\Theta_k}, D_{m'}^{\Theta'_k})$ of θ-districts which share an interface, and C_U is the cost associated with each edge in E_U.

Definition 4.4: The *lower-adjacency graph* on each $D_m^{\Theta_k}$ is the triple $G_L^{k,m} = (V_L^{k,m}, E_L^{k,m}, C_L^{k,m})$, where $V_L^{k,m} = \{p_1^{k,m}, \ldots, p_{n^{k,m}}^{k,m}\}$, where $p_i^{k,m}$ is the i-th polytope in $D_m^{\Theta_k}$, $E_L^{k,m}$ is the set of all pairs of polytopes which share a facet, and $C_L^{k,m}$ is the cost associated with each edge in $E_L^{k,m}$. We set the cost $C_L^{k,m} = 1\ \forall k, m$ to minimize transitions within each $\mathcal{C}_A^{free,k,m}$. In Sec. VI we show results for solving the same problem with different high-level cost functions.

Problem 4.5: For the initial group abstraction state x_A^0, find a path on G_U and corresponding paths through θ-districts to the goal group abstraction state x_A^g.

Since there are multiple paths to the goal, we use an algorithm such as Dijkstra to determine the lowest cost path to the goal. We first find a path on the upper-adjacency graph to the goal θ-district . Then we determine a path from every polytope in $D_m^{\Theta_k}$ to $D_m^{\prime\Theta'_k}$ on the lower-adjacency graph.

Theorem 4.6 (Necessary and Sufficient condition): Problem 4.2 has a solution iff Problem 4.5 has a solution.

Proof: $\mathcal{C}_A^{free}$ contains every allowable configuration x_A in our polytopic world model. G_U and G_L together contain all the information about the connectivity of $\mathcal{C}_A^{free}$. Thus, if there is a solution to Problem 4.2, there must exist a path from the start node in $G_L^{k^0,m^0}$ to the goal node in $G_L^{k^g,m^g}$. Conversely, if there is no path on the graph between the start node and the goal node, there is no solution to Problem 4.2.

We would like to automatically synthesize feedback controllers on each polytope on the path to solve Problem 4.2. There are a few options for controllers. Both [31] and [32] derive controllers that drive a system from any initial condition in a polytope through a desired exit facet while guaranteeing the system does not leave the polytope. [31] requires solving a linear program on each polytope to determine the inputs at each vertex, then triangulating the polytope to deterministically apply the controller within the polytope. Though this is a straightforward approach, smoothness is difficult to impose. We choose to use the controller in [32]. Here, a vector field is assigned to each facet of the polytope, inward for invariant facets and outward for the exit facet. On the Generalized Voronoi Diagram (GVD) a vector field is assigned pointing out of the exit facet. A bump function allows for smooth blending between the fields inside the cells of the GVD. To ensure smoothness between polytopes, we choose the facet vector fields perpendicular to the facets.

Corollary 4.7 (Completeness): Problem 4.5, and therefore Problem 4.2, has a solution if the start and goal nodes on the polytope graph G_U are connected.

V. LOWER LEVEL CONTROL

In this section we discuss a few local robot controllers, which keep the robots within the boundary as it traverses the space and deforms. Since the boundary restricts the allowable space for the robots, the robots' configuration space is local and decoupled from the cluttered physical workspace. The input to each robot in the local reference frame of the abstraction is

$$u_j = u_j \left(x_1^l, x_2^l, \ldots, x_N^l, s \right), \tag{13}$$

where x_i^l is the position of robot r_i in the local reference frame (fixed to and rotating with the center of the group abstraction) and s is the shape vector.

A. Robot controllers

In this paper we use a navigation function approach to demonstrate the use of a controller in a decentralized way while guaranteeing that the robots stay within the abstraction boundary. To solve the local navigation problem, we can use any controller reactive to changing workspace boundaries. For example, we demonstrate a flexible formation strategy in Sec. VI-A. A Voronoi coverage type controller [33]–[35] would be useful if the group was being used for sensing or

surveillance of different spaces (demonstrated in Sec. VI-B. If maintaining a specific shape is required [22] can be used if parametrized. For stricter formations, [15], [24], [28] provide a more structured organization of robots.

It is critical that the group dynamics evolve on a sufficiently faster time scale than abstraction dynamics to guarantee safety. Summing controllers without sufficient time scale separation could result in local minima.

B. Choosing shape constraints

Choosing shape constraints (5) is critical to ensure that there is enough space in the abstraction to maintain the desired formation. Knowing the shape of the abstraction, we can determine the minimum size of the abstraction so that it is large enough to contain the number of robots in the group. Without considering a formation, the minimum size of the abstraction depends on the number of robots in the group, N (and a minimum distance collision constraint $\delta_{\min}$, if desired). Indeed it is possible that two formations with the same number of robots require different shape constraints. For general guidelines on requirements for abstraction size and shape constraints, the interested reader is referred to [28].

VI. SIMULATIONS

In this section we demonstrate two lower level controllers in the exact same space. In both simulations, $\Delta\theta = 20°$, the start location is in district D^3 at $A = (x_A, \theta, s) = \left([2.5\ 2.5]^{\mathrm{T}}, 0, [3.3\ 3.8]^{\mathrm{T}}\right)$, and the goal location is in district D^7, at $A{=}(x_A, \theta, s){=}\left([13\ 4]^{\mathrm{T}}, 0, [2.5\ 4]^{\mathrm{T}}\right)$ (units in meters).

A. Flexible Formation Simulation

Here we use a flexible formation: each robot is assigned part of a uniform 3×3 grid in the boundary, parametrized by s_w and s_h. To prevent inter-robot collision, we enforce $2.1\text{m} \leq \{s_w, s_h\}$ and $4.2\text{m} \leq s_w + s_h \leq 8\text{m}$, so that the distance between robots would ideally be a minimum of 0.7m. (Maintaining this inter-robot distance is not guaranteed using this robot controller. However, using other controllers, such as [26], it is possible to guarantee inter-robot distances.) The navigation functions, coupled with the faster time scale at the robot level, ensure the robots do not escape the boundary.

B. Voronoi Coverage Simulation

Here we use navigation functions to drive robots to the centroid of their Voronoi regions. With this type of controller, the bearing of the robots relative to each other will change as the abstraction boundary changes. The constraints on this simulation are $1\text{m} \leq \{s_w, s_h\}$, $4\text{m} \leq s_w + s_h \leq 8\text{m}$. In this simulation, the cost of changing angular intervals is doubled, so that the lowest cost path travels through more rooms.

VII. COMPLEXITY

The complexity of our method is independent of the number of robots. It is dominated by the number of polytopes in $\mathcal{C}_A^{free}$. We extend the hyperplanes which support the intersections of districts with obstacles. The boundary of 2-d C-obstacles is made of at most $\mathcal{O}(n_A, n_B)$ edges where n_A is the number of vertices in the abstraction, and n_B the number of vertices in the obstacle. We construct C-obstacles in $\mathbb{R}^4$, since we consider changing sizes of the abstraction. Since each hyperplane may divide the existing polytopes into two, the maximum number of polytopes in one θ-slice of $\mathcal{C}_A^{free}$ is $\mathcal{O}(2^{n_A n_B} q)$ where q is the number of θ-slices.

VIII. CONCLUDING REMARKS

The central contribution of this paper is to provide a method to construct abstractions for a group of robots which allows automatic synthesis of feedback controllers for navigating an environment with obstacles. We defined abstractions to be virtual boundaries that bound the positions of the robots in the group. We designed feedback controllers that allow the virtual boundary associated with the abstractions to change its position and shape while avoiding the obstacles in the environment. We then designed robot controllers that satisfied inter-robot constraints and specifications while guaranteeing that the robots do not exit the boundary of the abstraction. This two-level hierarchy allows us to manage the complexity of synthesizing controllers for large groups.

While this paper addressed the problems of planning and control for a single group with homogeneous robots, it is also possible to consider a team with multiple groups with merging and splitting operations to change the composition of the group. Further, the robots in each group could be heterogenous with unique capabilities and identities. Some of these ideas are presented in a recently published paper [28]. Our future work is to extend the methods in the present paper to incorporate ideas from [28], and to consider experimental validation of the basic ideas in this paper.

REFERENCES

[1] R. D'Andrea, "Towards a ten thousand mobile robot warehouse," in *Proc. Int. Symp. Robot. Research*, Luzern, Switzerland, August 2009.

[2] J. Carlson and R. Murphy, "How ugvs physically fail in the field," *IEEE Trans. Robot.*, vol. 21, no. 3, pp. 423–437, June 2005.

[3] R. Murphy, J. Kravitz, S. Stover, and R. Shoureshi, "Mobile robots in mine rescue and recovery," *IEEE Robot. Autom. Magazine*, vol. 16, no. 2, pp. 91–103, June 2009.

[4] M. Saptharishi, C. Spence Oliver, C. Diehl, K. Bhat, J. Dolan, A. Trebi-Ollennu, and P. Khosla, "Distributed surveillance and reconnaissance using multiple autonomous ATVs: Cyberscout," *IEEE Trans. Robot. Autom.*, vol. 18, no. 5, pp. 826–836, Oct 2002.

[5] J. Fink, M. A. Hsieh, and V. Kumar, "Multi-robot manipulation via caging in environments with obstacles," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2008, pp. 1471–1476.

[6] V. Kumar, N. Leonard, and A. S. Morse, Eds., *Cooperative Control*, ser. Lecture Notes in Control and Information Sciences, vol. 309. Berlin Heidelberg New York: Springer-Verlag, 2004, http://www.cis.upenn.edu/ kumar/wcc/index.html.

[7] 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, October 2003.

[8] J. Desai, J. Ostrowski, and V. Kumar, "Modeling and control of formations of nonholonomic mobile robots," *IEEE Trans. Robot. Autom.*, vol. 17, no. 6, Dec. 2001.

[9] H. Tanner, A. Jadbabaie, and G. J. Pappas, "Coordination of multiple autonomous vehicles," in *Proc. of IEEE Mediterranean Conf. Control and Automation*, Rhodes, Greece, June 2003.

[10] C. Belta and V. Kumar, "Towards abstraction and control for large groups of robots," in *Control Problems in Robotics, Springer Tracts in Advanced Robotics*. Berlin: Springer-Verlag, 2002.

[11] M. Ji, A. Muhammad, and M. Egerstedt, "Leader-based multi-agent coordination: controllability and optimal control," in *Proc. Amer. Control Conf.*, Minneapolis, Minnesota, June 2006.

[12] M. C. D. Gennaro and A. Jadbabaie, "Formation control for a cooperative multi-agent system using decentralized navigation functions," in *Proc. Amer. Control Conf.*, Minneapolis, Minnesota, June 2006.

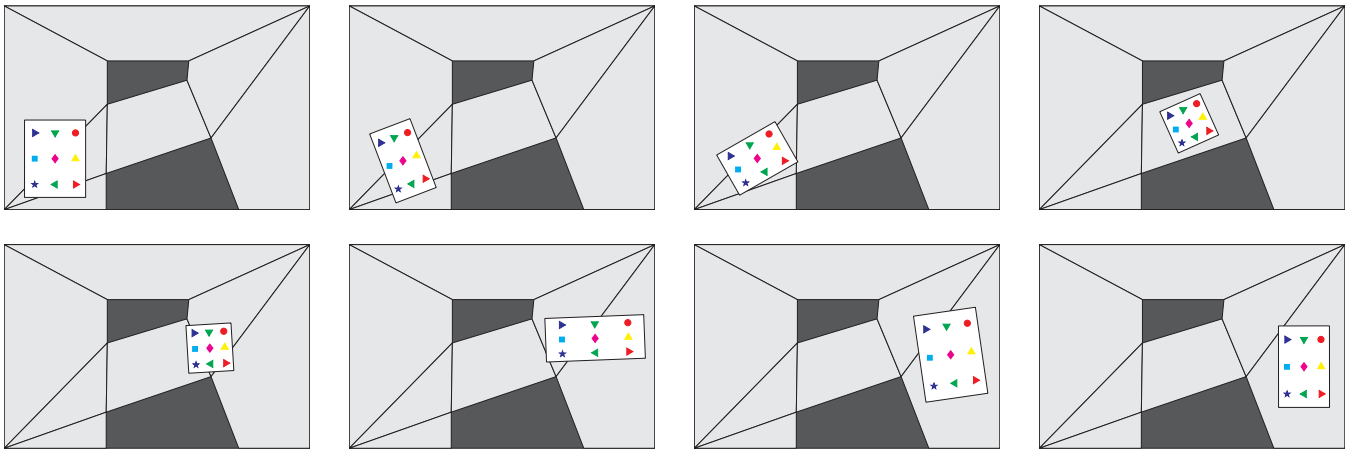


Fig. 4: A simulation of 9 robots navigating a complex space while maintaining a flexible formation inside the abstraction.

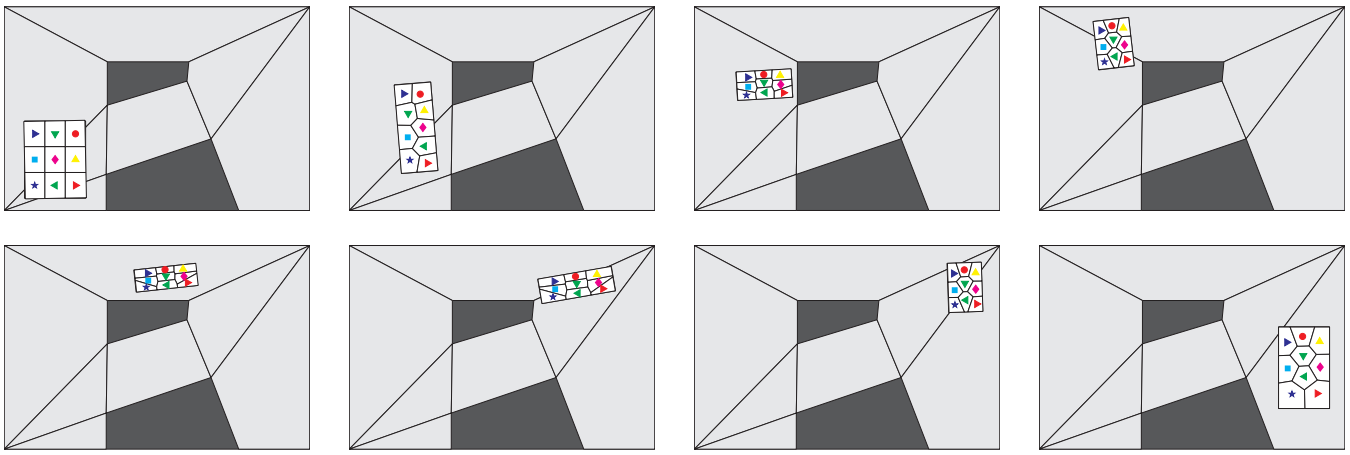


Fig. 5: Simulation of 9 robots navigating a complex space while using Voronoi region coverage methods inside the abstraction.

[13] S. Zelinski, T. Koo, and S. Sastry, "Optimization-based formation reconfiguration planning for autonomous vehicles," in *Proc. IEEE Conf. Robot. Autom.*, Taiwan, September 2003.

[14] O. Orqueda and R. Fierro, "Robust vision-based nonlinear formation control," in *Proc. American Control Conf.*, June 2006.

[15] M. Egerstedt and X. Hu, "Formation constrained multi-agent control," *IEEE Trans. Rob. Autom.*, vol. 17, no. 6, pp. 947–951, Dec 2001.

[16] 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.

[17] 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.

[18] C. Belta and V. Kumar, "Abstraction and control for groups of robots," *IEEE Trans. Rob.*, vol. 20, no. 5, pp. 865–875, Oct. 2004.

[19] N. Michael and V. Kumar, "Controlling shapes of ensembles of robots of finite size with nonholonomic constraints," in *In Proc. Robotics: Science and Systems*, June 2008, pp. 157–162.

[20] 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.

[21] H. G. Tanner, A. Jadbabaie, and G. J. Pappas, "Flocking in fixed and switching networks," *IEEE Trans. Automat. Contr.*, vol. 52, no. 5, pp. 863–868, May 2007.

[22] 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.

[23] P. Ogren, M. Egerstedt, and X. Hu, "A control lyapunov function approach to multi-agent coordination," in *Proc. IEEE Conf. Dec. Control*, vol. 2, 2001, pp. 1150–1155 vol.2.

[24] N. Leonard and E. Fiorelli, "Virtual leaders, artificial potentials and coordinated control of groups," in *Proc. of IEEE Conf. on Decision and Control*, vol. 3, 2001, pp. 2968–2973.

[25] B. Smith, J. Wang, and M. Egerstedt, "Persistent formation control of multi-robot networks," in *Proceedings of the IEEE Conference on Decision and Control*, Dec. 2008, pp. 471–476.

[26] N. Ayanian and V. Kumar, "Decentralized feedback controllers for multi-agent teams in environments with obstacles," in *Proc. IEEE Int. Conf, Robot. Autom.*, Pasadena, CA, May 2008, pp. 1936–1941.

[27] C. Anderson and N. R. Franks, "Teams in animal societies," *Behav Ecol*, vol. 12, no. 5, pp. 534–540, 2001.

[28] N. Ayanian, V. Kumar, and D. Koditschek, "Synthesis of controllers to create, maintain, and reconfigure robot formations with communication constraints," in *Proc. Int. Symposium Robot. Research*, Luzern, Switzerland, August 2009.

[29] J.-C. Latombe, *Robot Motion Planning*. Kluwer Academic, 1991.

[30] T. Lozano-Perez, "Spatial planning: A configuration space approach," *IEEE Trans. Comput.*, vol. 32, no. 2, pp. 108–120, 1983.

[31] L. Habets and J. van Schuppen, "A control problem for affine dynamical systems on a full-dimensional polytope," *Automatica*, vol. 40, no. 1, January 2004.

[32] S. Lindemann and S. Lavalle, "Computing smooth feedback plans over cylindrical algebraic decompositions," in *Robotics: Science and Systems*. Philadelphia, Pennsylvania: MIT Press, August 2006.

[33] J. Cortes, S. Martinez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," *IEEE Trans. Robot. Autom.*, vol. 20, no. 2, pp. 243–255, April 2004.

[34] L. C. A. Pimenta, M. Schwager, Q. Lindsey, V. Kumar, D. Rus, R. C. Mesquita, and G. A. S. Pereira, "Simultaneous coverage and tracking (scat) of moving targets with robot networks," in *Proc. Int. Workshop Algorithmic Found. Robot.*, Guanajuato, Mexico, December 2008.

[35] M. Schwager, D. Rus, and J.-J. E. Slotine, "Decentralized, adaptive control for coverage with networked robots," *The International Journal of Robotics Research*, vol. 28, no. 3, pp. 357–375, 2009.