Temporal Logic Planning and Control of Robotic Swarms by Hierarchical Abstractions

Marius Kloetzer, *Student Member, IEEE*, and Calin Belta, *Member, IEEE*

***Abstract*—We develop a hierarchical framework for planning and control of arbitrarily large groups (swarms) of fully actuated robots with polyhedral velocity bounds moving in polygonal environments with polygonal obstacles. At the first level of hierarchy, we aggregate the high-dimensional control system of the swarm into a small-dimensional control system capturing its essential features. These features describe the position of the swarm in the world and its size. At the second level, we reduce the problem of controlling the essential features of the swarm to a model-checking problem. In the obtained hierarchical framework, high-level specifications given in natural language, such as linear temporal logic formulas over linear predicates in the essential features, are automatically mapped to provably correct robot control laws. For the particular case of an abstraction based on centroid and variance, we show that swarm cohesion, interrobot collision avoidance, and environment containment can also be specified and automatically guaranteed in our framework. The obtained communication architecture is centralized.**

***Index Terms*—Control, model checking, motion planning, robotic swarms, temporal logic.**

I. Introduction

As a result of recent technological advances, it is now possible to build teams of hundreds of small and inexpensive ground, air, and underwater robots. Such *swarms of autonomous agents* provide increased robustness to individual failures, the possibility to cover wide regions, and improved computational power through parallelism. However, planning and controlling such large teams of agents with limited communication and computation capabilities is a hard problem that receives a lot of attention from various communities. Even though, in some cases, it was observed or proved that local interaction rules in distributed natural or engineered multiagent systems produce global behavior, the fundamental questions still remain to be answered. What are the essential features of a large group? How do we specify its behavior? How can we generate control laws for each agent so that a desired group behavior is achieved?

The starting point for this paper is the observation that tasks for large groups evolving in complex environments are "qualitatively" specified. This notion has a dual meaning. First, a swarm is naturally described in terms of a small set of "features," such as shape, size, and position of the region in the plane or space occupied by the robots, while the exact position or trajectory of each robot is not of interest. Second, the accomplishment of a swarming mission usually does not require exact values for swarm features, but rather their inclusion in certain sets. For example, in the planar case, if the robots are constrained to stay inside an ellipse, there is a whole set of values for the pose and semi-axes of the ellipse which guarantees that the swarm will not collide with an obstacle of given geometry. Moreover, specifications for mobile robots are often temporal, even though time is not necessarily captured explicitly. For example, a swarm might be required to reach a certain position and shape eventually, or maintain a size smaller than a specified value until a final desired value is achieved. Collision avoidance among robots, obstacle avoidance, and cohesion are required always. In a surveillance mission, a certain area should be visited "infinitely often."

Motivated by the above ideas, in this paper, we present a computational method for planning and control of robotic swarms based on *abstractions*. Our framework is hierarchical. At the first level, we construct a *continuous abstraction* by extracting a small set of features of interest of the swarm. Even though the treatment in this paper is quite general, the focus is on a 3-D abstraction consisting of the mean and variance of the positions of the team, which lead to a description of the swarm position and size. At the second level of hierarchy, we map arbitrary linear temporal logic LTL_{-X}[1] formulas over linear predicates in the abstract variables to a control strategy in the abstract space, which is eventually projected back to the individual robots. We show that for this particular abstraction, and under the assumption that the environment and the obstacles are polygonal, containment in the environment, swarm cohesion, and interrobot and obstacle collision avoidance translate naturally to LTL_{-X} formulas over linear predicates in the continuous abstraction space. We also show that the semantics of LTL_{-X} formulas over linear predicates in the abstract space is rich enough to capture temporal specifications such as the examples at the end of the previous paragraph. Our framework therefore allows for a rich spectrum of swarm specifications.

One way of reducing the dimension of the control problem for a large number of robots is to constrain them to a rigid virtual structure [1]. In this case, the problem is reduced to a left invariant control system in SE(2) or SE(3). Most of the recent works on stabilization and control of virtual structures model formations using *formation graphs* [2]–[4]. Virtual structures unnecessarily constrain the problem, making this approach

Manuscript received March 18, 2006; revised July 28, 2006. This paper was recommended for publication by Associate Editor J. Wen and Editor L. Parker upon evaluation of the reviewers' comments. This work was supported in part by the National Science Foundation under Grant CAREER 0447721 and under Grant 0410514. This paper was presented in part at the IEEE International Conference on Robotics and Automation (ICRA), Orlando, FL, 2006.

The authors are with the Center for Information and Systems Engineering, Boston University, Boston, MA 02446 USA (e-mail: kmarius@bu.edu; cbelta@bu.edu).

Color versions of Figs. 2–4 are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TRO.2006.889492

[1] LTL_{-X} is a propositional linear temporal logic that does not allow for the "next" temporal operator. Our motivation for this choice and more details are included in Section II.

inappropriate for tasks such as obstacle avoidance, passing of narrow tunnels, etc. The rigidity assumption is relaxed in an attempt to produce optimal trajectories in a geometrical framework in [5], however, the amount of computation becomes prohibitively large. The notions of invariance with respect to world frames and permutations of robots are properly approached in shape theory in the well-known "n-body problem," where it is agreed that the coordinates divide into internal (shape) and orientational coordinates. However, in some applications, it is not clear how to construct shape coordinates explicitly, unless they are local, or the problem is restricted to three or four bodies [6]. Some of these ideas were applied to robotics [7], [8]. The problem of constructing an abstract description of the swarm with a product structure of a group and a reduced shape space is approached in [9].

The continuous abstraction defined in this paper is inspired from [9]. One of the contributions of this paper is defining and proving its consistency. This paper also relates to results on reducing the dimension of control systems, such as the ones reported in [10]–[12]. However, as opposed to [10], where the approach is time-abstract, our notion of consistency captures time explicitly. The problem of explicitly computing a timed trajectory for a control system from a trajectory of a lower dimensional (abstract) control system is considered in [11]. While focusing on simpler control systems, we generate trajectories for whole equivalence classes produced by the abstraction map, rather than one particular trajectory. From this point of view, our aggregation produces a bisimulation quotient, and relates to using foliations for constructing quotients as in [12] and [13].

The discrete abstraction of this paper is an application of results from [14], and it also relates to [15]. Recent works advocating the use of discrete abstractions and temporal logic in mobile robotics include [16]–[19]. One of the main contributions of this paper is to show that a large class of robotic swarm specifications translate naturally to linear temporal logic (LTL), and a fully automated framework for generation of robot control laws can be constructed. Moreover, as far as we know, this paper is the first example of a combination of continuous and discrete abstractions for groups of robots.

II. Preliminaries

In this section, we first review the syntax and semantics of the linear temporal logic LTL_{-X}. The reader is referred to [20] for more details. We then define the semantics of this logic over continuous curves. A more formal treatment can be found in our previous work [14].

A. Linear Temporal Logic LTL_{-X}

Let $\Pi = \{\pi_1, \pi_2, \ldots, \pi_K\}$ be a finite set of atomic propositions.

Definition 1: [Syntax of LTL_{-X} formulas] A linear temporal logic LTL_{-X} formula over Π is recursively defined as follows:

- every atomic proposition π_i, $i = 1, \ldots, K$ is a formula; and
- if ϕ_1 and ϕ_2 are formulas, then $\phi_1 \vee \phi_2$, $\neg\phi_1$, $\phi_1 \mathcal{U} \phi_2$ are also formulas.

The semantics of LTL_{-X} formulas are given over ω, words in the power set 2^{Π} (infinite sequences of sets of propositions) of the form $w = w(1)w(2)w(3)\ldots$, where $w(i) \in 2^{\Pi}$, $i \geq 1$.

Definition 2: [Semantics of LTL_{-X} formulas] The satisfaction of formula ϕ at position $i \in \mathbb{N}$ of word w, denoted by $w(i) \vDash \phi$, is defined recursively as follows:

- $w(i) \vDash \pi$ if $\pi \in w(i)$;
- $w(i) \vDash \neg\phi$ if $w(i) \nvDash \phi$;
- $w(i) \vDash \phi_1 \vee \phi_2$ if $w(i) \vDash \phi_1$ or $w(i) \vDash \phi_2$;
- $w(i) \vDash \phi_1 \mathcal{U} \phi_2$ if there exists a $j \geq i$ such that $w(j) \vDash \phi_2$ and for all $i \leq k < j$, we have $w(k) \vDash \phi_1$.

A word w satisfies an LTL_{-X} formula ϕ, written as $w \vDash \phi$, if $w(1) \vDash \phi$.

The symbols $\neg$ and $\vee$ stand for negation and disjunction. The Boolean constants $\top$ and $\bot$ are defined as $\top = \pi \vee \neg\pi$ and $\bot = \neg\top$. The other Boolean connectors $\wedge$ (conjunction), $\Rightarrow$ (implication), and $\Leftrightarrow$ (equivalence) are defined from $\neg$ and $\vee$ in the usual way. The *temporal operator* $\mathcal{U}$ is called the *until* operator. Formula $\phi_1 \mathcal{U} \phi_2$ intuitively means that (over a word) ϕ_2 will eventually become true and ϕ_1 is true until this happens. Two useful additional temporal operators, "eventually" and "always," can be defined as $\Diamond\phi = \top\mathcal{U}\phi$ and $\Box\phi = \phi\mathcal{U}\bot$, respectively. Formula $\Diamond\phi$ means that ϕ becomes eventually true, whereas $\Box\phi$ indicates that ϕ is true at all positions of w. More expressiveness can be achieved by combining the temporal operators. Examples include $\Box\Diamond\phi$ (ϕ is true infinitely often) and $\Diamond\Box\phi$ (ϕ becomes eventually true and stays true forever).

B. LTL_{-X} Semantics Over Continuous Curves

Let us now assume that the propositions π_i in Π are strict linear inequalities in $\mathbb{R}^n$, $n \geq 2$, i.e., Π is given by

$$\Pi = \{\pi_i | \pi_i : c_i^T x + d_i < 0, \quad i = 1, \ldots, K\} \tag{1}$$

where $c_i \in \mathbb{R}^n$ and $d_i \in \mathbb{R}$. Let $a : \mathbb{R}_+ \to \mathbb{R}^n$ be a (possibly nonsmooth) continuous curve in $\mathbb{R}^n$ (a is allowed to have self-intersections). We also assume that $c_i^T \bar{a} + d_i \neq 0$ and $c_i^T a(0) + d_i \neq 0$ for all $i = 1, \ldots, K$, where $\bar{a} = \lim_{t\to\infty} a(t)$ (if it exists). The semantics of an LTL_{-X} formula in Π over a continuous curve a follows naturally from the above definitions. A *word* generated by a is a sequence $w_a = w(1)w(2)w(3)\ldots$, $w(i) \in 2^{\Pi}$, $i \geq 1$ obeying the following rules: 1) $w(1)$ is the set of all atomic propositions satisfied by $a(0)$; 2) a symbol $w(i) \neq w(i-1)$, $i \geq 2$ is added to w_a if there exist t_1, t_2, $0 \leq t_1 < t_2$ so that $a(t_1)$ satisfies all the propositions in $w(i-1)$, $a(t_2)$ satisfies all the propositions in $w(i)$, and $a(t)$ satisfies only propositions from $w(i-1) \cup w(i)$, for all $t_1 \leq t \leq t_2$; 3) an infinite number of symbols $w(i)$, $i \geq 1$ is added to w_a if the region represented by $w(i)$ is a "sink" for trajectory a, in the sense that $\exists \tau > 0$ such that all and only propositions in $w(i)$ are satisfied by $a(t)$, $\forall t \geq \tau$. Finally, a trajectory $a : \mathbb{R}_+ \to \mathbb{R}^n$ satisfies ϕ, written as $a \vDash \phi$ if and only if $w_a \vDash \phi$ (as defined above).

Intuitively, a word produced by trajectory a is an enumeration of the sets of propositions from Π satisfied by $a(t)$ while time evolves. Some illustrative examples are given in Fig. 1, where π_i, $i = 1, \ldots, 6$ are open half-spaces, and $\Pi_1 = \{\pi_1, \pi_3\}$,

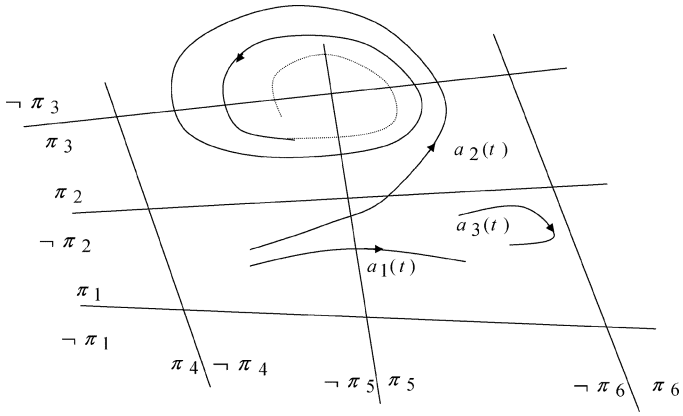

Fig. 1. Illustrative examples of continuous trajectories in $\mathbb{R}^2$.

$\Pi_2 = \{\pi_1, \pi_3, \pi_5\}, \Pi_3 = \{\pi_1, \pi_2, \pi_3, \pi_5\}, \Pi_4 = \{\pi_1, \pi_2, \pi_5\}$, $\Pi_5 = \{\pi_1, \pi_2\}$, $\Pi_6 = \{\pi_1, \pi_2, \pi_3\}$. Continuous trajectory a_1 starts from the region where the predicates in Π_1 are true and converges to a point in the region where the predicates in Π_2 are true. By the above definition, the word w_{a_1} is $\Pi_1\Pi_2\Pi_2\ldots$. Trajectory a_2 starts from Π_1 and loops infinitely, as shown in the figure. The corresponding word w_{a_2} will be $\Pi_1\Pi_2\Pi_3\Pi_4\Pi_5\Pi_6\Pi_3\Pi_4\Pi_5\Pi_6\ldots$. For trajectory a_3 originating in Π_2 and converging inside Π_2, the word w_{a_3} is $\Pi_2\Pi_2\Pi_2\ldots$.

LTL, the most used propositional linear temporal logic [20], is richer than LTL_{-X} in the sense that it allows for an additional temporal operator, called "next." The increased expressivity of *LTL* is manifested only over words with a finite number of successive repetitions of a symbol. Our choice of LTL_{-X} over *LTL* is motivated by the fact that a word corresponding to a continuous trajectory will never have a finite number of successive repetitions of a symbol.

III. Problem Formulation and Approach

Consider a set of N identical planar fully actuated pointlike robots described by

$$\dot{r}_i = u_i, \quad r_i \in P, \quad u_i \in U, \quad i = 1, \ldots, N \tag{2}$$

where $r_i \in \mathbb{R}^2$ is the position vector of robot i in a world frame $\{F\}$ and u_i is its velocity, which can be directly controlled. $U \subseteq \mathbb{R}^2$ is a polyhedral set capturing the control constraints, and P is a polygonal environment. Assume that P contains a set of polygonal obstacles O_j, $j = 1, \ldots, o$. When necessary, and as it will become clear from the context, we also use P and O_j to denote the propositional logic formulas describing the polygonal environment and the obstacles, respectively (they consist of conjunctions and disjunctions over linear inequalities in $\mathbb{R}^2$).

We collect all the robot states in $r = [r_1^T, \ldots, r_N^T]^T \in \mathbb{R}^{2N}$ (referred to as the configuration of the "swarm") and the robot controls in $u = [u_1^T, \ldots, u_N^T]^T \in \mathbb{R}^{2N}$. To recover the individual states and controls, we define the canonical projection $pr_i(r) = r_i$, $pr_i(u) = u_i$, $i = 1, \ldots, N$. Equation (2) can therefore be written as

$$\dot{r} = u, \quad pr_i(r) \in P, \quad pr_i(u) \in U, \quad i = 1, \ldots, N. \tag{3}$$

Swarming tasks are specified in high-level language in terms of a small set of properties to be satisfied by the swarm. Examples of such properties include *containment* of motion inside the environment P, *avoidance of obstacles* O_j, $j = 1, \ldots, o$, *cohesion* (i.e., all pairwise distances smaller than a maximum predefined value), and *interrobot collision avoidance* (i.e., all pairwise distances larger than a minimum predefined value). In addition to these, the motion tasks are usually given in terms of temporal and logical specifications over a small set $a \in \mathbb{R}^n$, $n \ll N$ of essential features, while the exact values of r are not of interest. The essential features usually include information about the position, orientation, size, and shape of the region in the plane spanned by the swarm. For example, assume that $a = (\mu, s) \in \mathbb{R}^3$, where $\mu \in \mathbb{R}^2$ gives the centroid of a swarm, and $s \in \mathbb{R}$ is its size (e.g., area). If it is desired that the swarm converges to a configuration in which its centroid belongs to a polygon $P^d \subset P$ and with a size smaller than s^d, this can be written more formally as "eventually always ($\mu \in P^d$ and $s < s^d$)," with the obvious interpretation that it will eventually happen that $\mu \in P^d$ and $s < s^d$, and this will remain true for all future times. If during the convergence to the final desired configuration it is necessary that the swarm visits a position $\bar{\mu}$ with a size $\bar{s}$, then the specification becomes "eventually (($\mu = \bar{\mu}$ and $s = \bar{s}$) and (eventually always ($\mu \in P^d$ and $s < s^d$)))." If, in addition, it is required that the size s is smaller than $\bar{s}$ for all times before $\bar{s}$ is reached, the specification changes to "$s < \bar{s}$ until (($\mu = \bar{\mu}$ and $s = \bar{s}$) and (eventually always ($\mu \in P^d$ and $s < s^d$)))."

The starting point for this work is the observation that such specifications translate naturally to LTL_{-X} formulas over linear predicates interpreted over trajectories of essential features a (as defined in Section II). For example, the last specification in the above paragraph corresponds to the formula $(s < \bar{s})\mathcal{U}(\mu = \bar{\mu} \wedge s = \bar{s}) \wedge (\Diamond\Box(\mu \in P^d \wedge s < s^d))$. In this paper, we consider the following problem.

1) Problem 1: Identify a set of features a describing the region spanned by the swarm, and construct robot control strategies $u_i \in U$, $i = 1, \ldots, N$ so that:

1) containment, obstacle avoidance, interrobot collision avoidance, and cohesion are achieved; and
2) arbitrary LTL_{-X} formulas over arbitrary linear predicates in a are satisfied by all produced trajectories $a(t)$, $t \geq 0$.

To provide a solution to *Problem 1*, we propose a *hierarchical abstraction* approach (see Fig. 2 for a graphical illustration). In the first level of abstraction, called *continuous abstraction*, we extract the essential features of the swarm by building a smooth surjective map

$$h : \mathbb{R}^{2N} \to \mathbb{R}^n, \quad h(r) = a \tag{4}$$

where h is called the (continuous) *abstraction*, or *aggregation*, or *quotient* map, and a is denoted as the *abstract state* of the swarm. In addition to providing a description of the swarm position, size, and shape, we will require h to perform a *correct aggregation* of the large-dimensional state space $\mathbb{R}^{2N}$ of the swarm. As we define in Section IV, a correct aggregation has

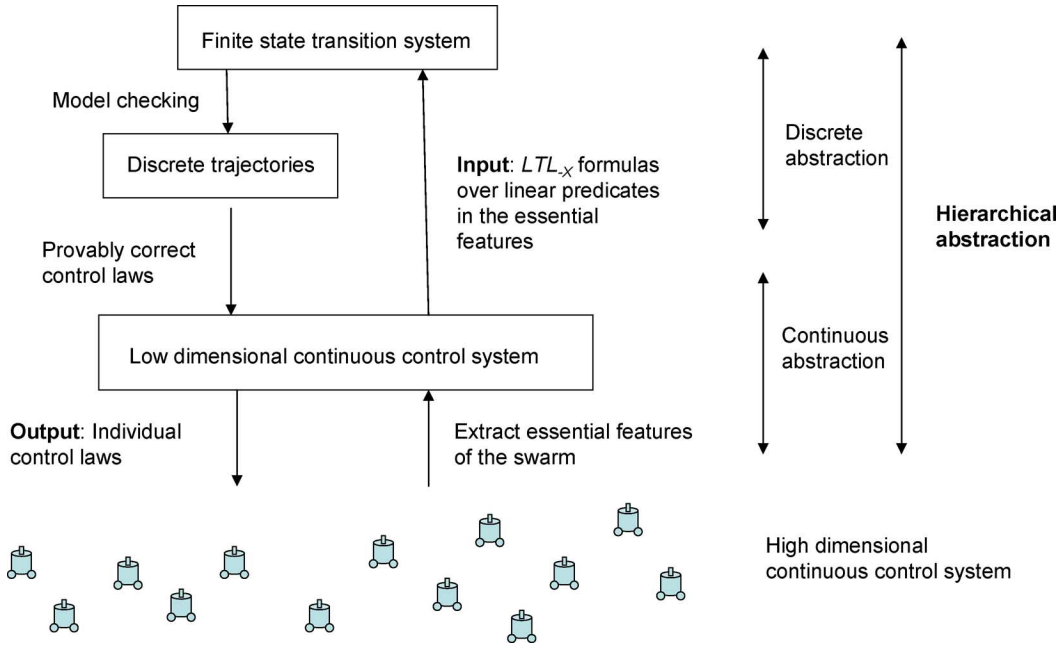

Fig. 2. Hierarchical abstraction architecture for planning and control of robotic swarms. High-level specifications given in human-like language such as temporal logic formulas are used to construct a discrete and finite description of the problem. Tools resembling model checking are involved to find a solution for this problem, which is then implemented as a hybrid control strategy for the continuous abstraction. Individual robot control laws are then generated through projection.

three requirements. First, the flow (3) and the quotient map (4) have to be *consistent*, which intuitively means that swarm configurations that are equivalent (indistinguishable) with respect to h are treated in an equivalent way by the flow (3). Second, a correct aggregation should allow for any motion on the abstract space $\mathbb{R}^n$, which we call the *actuation* property. Third, we do not allow the swarm to spend energy in motions which are not "seen" in the abstract space $\mathbb{R}^n$ (*detectability*).

If the state is correctly aggregated, then any trajectory $a(t) \in \mathbb{R}^n$ can be produced by the swarm. In the second level of abstraction, called *discrete abstraction*, we employ the method from [14] to generate control strategies in $\mathbb{R}^n$ so that arbitrary LTL_{-X} formulas over arbitrary linear predicates in $\mathbb{R}^n$ are satisfied by the abstract trajectories $a(t)$. A description of this method is given in Section V.

A particular abstraction is presented in Section VI. It is based on a 3-D continuous abstraction $a = (\mu, \sigma)$, where μ and σ are the centroid and variance of the robot positions, respectively. We show that the containment, obstacle avoidance, cohesion, and interrobot collision avoidance conditions reduce to LTL_{-X} formulas over linear predicates in a, therefore providing a common framework for both requirements 1) and 2) of *Problem 1*. Moreover, the polyhedral control bounds U are satisfied by imposing corresponding bounds for the velocity of the continuous abstraction. Finally, we show that for all robots, the feedback controller u_i depends only on the state r_i of the robot and the small-dimensional state of the continuous abstraction a.

IV. Continuous Abstraction

Let $u \in T\mathbb{R}^{2N}$ and $w \in T\mathbb{R}^n$ be two vector fields giving the full dynamics of the swarm

$$\dot{r} = u(r) \tag{5}$$

and its abstract dynamics

$$\dot{a} = w(a) \tag{6}$$

respectively, where $a = h(r)$. Let $dh(r) : T_r\mathbb{R}^{2N} \to T_{h(r)}\mathbb{R}^n$ denote the differential (tangent) map of h at point r. If $h = (h_1, \ldots, h_n)$, then $dh(r)$ is a $n \times 2N$ real matrix whose rows are dh_i, $i = 1, \ldots, n$.[2]

Definition 3 (h-Related Vector Fields [21]): The vector fields $u \in T\mathbb{R}^{2N}$ and $w \in T\mathbb{R}^n$ are called h-related [h is the smooth surjection from (4)] if

$$w(h(r)) = dh(r)u(r) \quad \forall r \in \mathbb{R}^{2N} \tag{7}$$

and the following matching condition is satisfied:

$$dh(r)u(r) = dh(r')u(r') \quad \forall r, r' \text{ with } h(r) = h(r'). \tag{8}$$

The h–relation is an extension of the more used notion of push–forward, which is only defined when h is a diffeomorphism [21].

Definition 4 (Correct Aggregation): The map (4) and the vector field (5) define a correct aggregation of the swarm if the following three properties are satisfied.

1) Consistency: $h(r(t)) = h(r'(t))\ \forall t \geq 0$, for all trajectories $r(t)$ and $r'(t)$ of (5) with $h(r(0)) = h(r'(0))$.
2) Actuation: The linear map $dh(r) : T_r\mathbb{R}^{2N} \to T_{h(r)}\mathbb{R}^n$ is surjective for all $r \in \mathbb{R}^{2N}$.
3) Detectability: $u(r) = 0$ if and only if $w(h(r)) = 0$ for all $r \in \mathbb{R}^{2N}$.

In other words, consistency means that swarm configurations which are equivalent with respect to the quotient produced by h

[2]For simplicity, and since we deal with Euclidean spaces only, throughout this paper, we will assume that vector fields, maps, and tangent maps are written in coordinates and they are global.

remain equivalent for all times under the flow (5) [or, the equivalence classes of h are invariant with respect to the flow (5)]. This condition is necessary and sufficient to reduce system (5) to system (6), if the specifications for the trajectories of (5) are given in terms of $a = h(r)$, rather than explicitly in terms of r. Actuation guarantees that any velocity $w(a)$ (and therefore, any motion) in the abstract space $\mathbb{R}^n$ can be achieved by the swarm. The detectability condition 3) guarantees that the swarm does not spend energy in "uninteresting" motions. Indeed, $u(r) \neq 0$ and $w(h(r)) = 0$ would correspond to a motion of the swarm resulting in no change in the abstract state $a \in \mathbb{R}^n$, which captures the features of interest of the swarm.

Proposition 1: Given a vector field (5) and a map (4), the consistency condition 1) from *Definition 4* is equivalent with the matching condition (8).

Proof: For necessity, assume that the matching condition (8) is satisfied. Let $r(t)$ and $r'(t)$ be two trajectories with $h(r(0)) = h(r'(0))$. The Lie derivative of $h(r(t)) - h(r'(t))$ along the vector field of (5) is given by

$$L_u(h(r(t))-h(r'(t)))=dh(r(t))u(r(t)) - dh(r'(t))u(r'(t)). \tag{9}$$

By evaluating (9) at time $t = 0$, we obtain

$$L_u(h(r(t)) - h(r'(t)))|_{t=0} = dh(r(0))u(r(0)) - dh(r'(0))u(r'(0)) = 0 \tag{10}$$

because $h(r(0)) = h(r'(0))$ and the matching condition is satisfied by hypothesis. We conclude that $h(r(t)) = h(r'(t))$ $\forall t \geq 0$, i.e., the trajectories $r(t)$ and $r'(t)$ remain equivalent for all times.

For sufficiency, assume consistency is satisfied, and let $\bar{r}$, $\bar{r}'$ be two equivalent states, i.e., $h(\bar{r}) = h(\bar{r}')$. Let $r(t)$ and $r'(t)$ be two trajectories such that $r(0) = \bar{r}$ and $r'(0) = \bar{r}'$. Since $h(r(t)) = h(r'(t))\ \forall t \geq 0$, the Lie derivative (9) is zero for all times, which implies that it is zero at time 0, which translates to $dh(\bar{r})u(\bar{r}) = dh(\bar{r}')u(\bar{r}')$. The matching condition is therefore satisfied, and the proposition is proved. ■

The actuation condition 2) is equivalent to requiring that h be a submersion. In other words, $h_i, i = 1, \ldots, n$ are functionally independent, or equivalently, dh_i are linearly independent for all r, which again is equivalent to dh is full-row rank for all r. Indeed, it is well known that a linear map is surjective if and only if it is full-row rank.

The submersion h determines an orthogonal decomposition of $T_r\mathbb{R}^{2N}$ in $\mathcal{N}(dh(r))$ (of dimension $2N - n$) and $\mathcal{R}(dh^T(r))$ (of dimension n), where $\mathcal{N}$ and $\mathcal{R}$ denote the null space and range of a matrix, respectively. With this observation, the detectability condition 3) is equivalent to restricting $u(r) \in \mathcal{R}(dh^T(r))$.[3] We can now collect all these results in the following theorem.

Theorem 1: [Correct aggregation] The smooth surjection (4) and the vector field (5) define a correct aggregation of the swarm if and only if:

1) the matching condition (8) is satisfied;
2) h is a submersion; and
3) $u(r) \in \mathcal{R}(dh^T(r))$.

If the conditions of *Theorem 1* are satisfied, then arbitrary "abstract" vector fields $w(a)$ ($a = h(r)$) in $\mathbb{R}^n$ can be produced by "swarm" vector fields $u(r)$ using (7), which is well defined since the aggregation is consistent. A particular solution of this equation is the minimum (Euclidean) norm solution

$$u(r) = dh^T(r)(dh(r)dh^T(r))^{-1}w(a) \quad \forall r \in h^{-1}(a) \tag{11}$$

where $h^{-1}(a)$ is the equivalence class of a, or explicitly $h^{-1}(a) = \{r \in \mathbb{R}^{2N} | h(r) = a\}$. Note that $dh(r)dh^T(r)^{-1}$ is invertible for all r since h is a submersion. It is obvious that $u(r)$ given by (11) satisfies condition 3) of *Theorem 1*. It also satisfies condition 1), since $dh(r)u(r) = w(a)$ for all $r \in h^{-1}(a)$. This result is summarized in the following *Corollary* of *Theorem 1*.

Corollary 1: If $w(a)$ is an arbitrary vector field in $\mathbb{R}^n$ and h is a submersion, then $u(r)$ from (11) and h define a correct aggregation of the swarm.

Finally, note that if, in addition to being linearly independent, dh_i, $i = 1, \ldots, n$ are orthogonal (in Euclidean metric), then (11) assumes a particularly simple form

$$u(r) = \sum_{i=1}^{n} \frac{w_i(a)}{dh_i(r)dh_i^T(r)} dh_i^T(r). \tag{12}$$

Remark 1: There is an interesting connection between consistency [*Definition 4*, 1)] and bisimilar quotients of continuous systems: the quotient system produced by the equivalence classes determined by h in $\mathbb{R}^{2N}$ is a bisimilarity quotient if and only if the aggregation is consistent. The interested reader is referred to [22] for definitions of bisimilarity relations for transition systems, and to [12] for extending such ideas to affine control systems. The consistency condition introduced here is related to continuous abstractions for affine control systems as defined in [10]–[12]. Our matching condition is related to the geometrical condition for bisimilarity of two control systems from [12], where drift is allowed. Note that a consistency condition for a (second-order) dynamical system can, in principle, be derived by reducing the system to first order using state space augmentation. However, the matching condition may be more difficult to satisfy, and it will be the subject of future work.

V. Discrete Abstraction

Once the large-dimensional state of the swarm is correctly aggregated by properly choosing the aggregation map and the robot control laws, we have the freedom to assign arbitrary vector fields (6) in the abstract space $\mathbb{R}^n$. To provide a solution to *Problem 1*, the produced trajectories should satisfy arbitrary LTL_{-X} formulas over linear predicates in $\mathbb{R}^n$. To this goal, we use the computational framework for control of linear systems from LTL_{-X} specifications over linear predicates from [14]. In this section, we very briefly outline this procedure.

Let ϕ denote an arbitrary LTL_{-X} formula over linear predicates in $\mathbb{R}^n$, and let Π [(1)] be the set of all linear predicates appearing in ϕ. The framework described in [14] consists of two main steps. In the first, a finite-state transition system is constructed. This construction starts with a proposition preserving

[3]There is a slight abuse of notation in this equation. dh_i are differential forms and their span is a co-distribution. However, when written in coordinates, they can be treated as vector fields, when their span is a distribution.

the partition of $\mathbb{R}^n$ into polytopes determined by feasible combinations of linear predicates from Π. The states of the transition system are the equivalence classes produced by the partition. Its transitions are determined by adjacency of polytopes and existence of affine feedback controllers making such polytopes invariant, or driving all states in a polytope to an adjacent polytope through a common facet. The second step consists of producing runs of the transition system that satisfy formula ϕ. This is, in essence, a model checking problem. A Büchi automaton [23] is first generated from formula ϕ, and then the product between the transition system and the Büchi automaton is constructed. All runs of the transition system obtained as projections of runs of the product automaton satisfy ϕ.

The algorithms in [14] return a set of initial states in the form of a union of polytopes in $\mathbb{R}^n$ and a feedback control strategy for (6) induced by the runs found using model checking. All trajectories $a(t)$ of the closed-loop system satisfy ϕ as defined in Section II-B. The feedback controllers w can have different values at different times at the same state a. The produced trajectories are, in general, non-smooth, can have self-intersections, and are continuous in time. Arbitrary polyhedral control constraints $W \subset \mathbb{R}^n$ can be accommodated.

Remark 2: While addressing the problem of planning and control of a large group of fully actuated planar robots, the hierarchical abstraction framework proposed in this paper is quite general. It can be used for any process with directly controllable velocity [such as (5)] and specifications given in terms of temporal logic formulas over linear predicates in some output variables [such as (4)], provided that the state of the initial system is correctly aggregated with respect to the output map (*Theorem 1*). Moreover, since the continuous and discrete abstraction procedures are completely independent, any type of discrete abstraction can be built on top of the continuous abstraction presented in this paper. For example, instead of linear predicates resulting in the construction of affine feedback controllers [14] for the continuous abstraction, one might use rectangular predicates (i.e., $a_i < c_i$), which will result in the construction of multiaffine controllers [24].

VI. Hierarchical Abstraction Based on Mean and Variance

In this section, we focus on a particular abstraction map $h : \mathbb{R}^{2N} \to \mathbb{R}^3$ [$n = 3$ in (4)], given by

$$h(r) = a, \;\; a = (\mu, \sigma), \;\; \mu = \frac{1}{N}\sum_{i=1}^{N} r_i$$
$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(r_i - \mu)^T(r_i - \mu)}. \tag{13}$$

It is easy to see that h is smooth everywhere in $\mathbb{R}^{2N}$ except for the set $r_1 = r_2 = \cdots = r_N$, which corresponds to $\sigma = 0$ (all the robots coincide). In what follows, we exclude this degenerate case. We will show that this choice of an abstraction provides a useful and *fully automatic* solution to *Problem 1*. Specifically, we will show that the containment, obstacle avoidance, inter-robot collision avoidance, and cohesion requirements translate to LTL_{-X} formulas over linear predicates in a, which leads to a unifying computational framework for both requirements 1) and 2) of *Problem 1*.

We first need to design robot control laws to make sure that the state of the swarm is correctly aggregated as described in *Definition 4*. The differential of h is given by

$$dh(r) = \frac{1}{N}\begin{bmatrix} I_2 & \cdots & I_2 \\ \frac{(r_1-\mu)^T}{\sigma} & \cdots & \frac{(r_n-\mu)^T}{\sigma} \end{bmatrix} \tag{14}$$

where I_n denotes the identity matrix of size n. From (14), it can be seen that h is a submersion (under the initial assumption that $\sigma \neq 0$). Moreover, dh_i, $i = 1, 2, 3$ are mutually orthogonal and $dhdh^T = (1/N)I_3$. Using (11) and by canonical projection, we obtain the control laws for each robot in the form

$$u_i(r_i, a) = \begin{bmatrix} I_2 & \frac{r_i-\mu}{\sigma} \end{bmatrix} w(a), \quad i = 1, \ldots, N \tag{15}$$

where $w(a)$ is an arbitrary vector field in the abstract space $\mathbb{R}^3$. According to *Corollary 1*, the control laws (15) and the abstraction map (13) define a correct aggregation of the swarm.

Note that the control u_i of robot i depends on its own state r_i and on the abstract state a. Even though this corresponds to a centralized communication architecture, it is easily implementable in a scenario where a swarm of ground vehicles is deployed together with a central agent, such as an unmanned aerial vehicle (UAV), or a blimp. At every time instant, the UAV determines (for example, using a camera and GPS) or receives the robot positions, calculates a, and disseminates it to all the robots (this is a small bandwidth signal). Each robot then computes its own control using its own state and (15). Note also that the overall control architecture is robust with respect to individual failures, in the sense that if a robot fails, only the central agent has to simply decrease by one the number of robots in the swarm.

A careful examination of (15) shows that all position vectors $r_i, i = 1, \ldots, N$ undergo the same affine transformation parameterized by the abstract variables μ and σ. Equation (15) can in fact be integrated, leading to

$$r_i(t) - \mu(t) = \frac{\sigma(t)}{\sigma(0)}(r_i(0) - \mu(0)) \tag{16}$$

$i = 1, \ldots, N$. Therefore, in a frame centered at the centroid μ and parallel with the world frame $\{F\}$, all position vectors scale by the same value given by σ. By combining (15) and (16), we also obtain

$$u_i(r_i(0), a(0), a) = \begin{bmatrix} I_2 & \frac{r_i(0)-\mu(0)}{\sigma(0)} \end{bmatrix} w(a) \tag{17}$$

$i = 1, \ldots, N$, where we emphasize that the control law u_i depends on the initial state of the robot, the initial value of the abstract state, and the current value of the abstract state (as opposed to the equivalent form in (15), where the dependence was on the current state of the robot and of the abstract state).

Remark 3 (Invariance of the Continuous Abstraction): As we emphasized in [9], it is important that a swarm abstraction map is invariant to permutations of robots and choice of world frames. The invariance to robot permutations can be easily seen

from (13), where the abstraction map is defined as a "democratic" sum over all the members of the swarm. The invariance with respect to the world frame translates to left invariance of h with respect to the diagonal action of SE(2) on $\mathbb{R}^{2N}$, which can be proved as in [9], where a slightly different abstraction map is considered.

A. Description of the Region Spanned by the Swarm

Let $V \subseteq \{1, \ldots, N\}$ denote the set of indices of robots which are at the vertices of the convex hull of the swarm. Since the controls (15) determine an affine transformation, then V does not change in time. At time 0, we convert from the vertex to the hyperplane representation of the convex hull of the swarm

$$\begin{aligned}\mathcal{P}_0 &= \operatorname{conv}\{r_i(0), i \in V\} \\ &= \{x \in \mathbb{R}^2 | a_i^T(x - \mu(0)) + b_i \leq 0, \quad i \in V\} \end{aligned} \tag{18}$$

where $a_i \in \mathbb{R}^2$ are the unit outer normals to the facets of the polytope, and $-b_i$ are the distances from $\mu(0)$ to the facets, $i \in V$ (note that $b_i \leq 0$). Since (15) corresponds to a particular affine transformation consisting of translating and scaling by the same factor, the convex hull of the swarm at time t will be described by

$$\mathcal{P} = \{x \in \mathbb{R}^2 | a_i^T(x - \mu(t)) + \frac{\sigma(t)}{\sigma(0)} b_i \leq 0, \quad i \in V\}. \tag{19}$$

It is important to note that description (19) of the region spanned by the swarm at time t is a conjunction of linear inequalities in the abstract variables $a(t) = (\mu(t), \sigma(t))$. The coefficients a_i, b_i, $\sigma(0)$, and the set V are all determined at time 0 and constant during the motion.

B. Containment and Obstacle Avoidance

Recall from Section III that the polygonal environment and the obstacles are described by propositional logic formulas P and O_j, $j = 1, \ldots, o$ over linear predicates in the plane. Using description (19) of the area spanned by the swarm, containment and obstacle avoidance is guaranteed if the following first-order formula is true:

$$\forall x : \mathcal{P} \Rightarrow \left(P \bigwedge_{j=1}^{o} \neg O_j \right). \tag{20}$$

Since $\mathcal{P}$ is linear in x and a, and P, O_j are linear in x, (20) is a formula in the logic of the reals with addition and comparison. Informally, the formulas of this first-order logic consist of linear inequalities with rational coefficients connected by logical and quantification operators. A basic computational feature of this logic is that any formula is equivalent to a quantifier-free formula, which can be effectively computed [25]. Let P_{co} denote a quantifier-free formula equivalent to (20), which is, of course, linear in the free variables μ and σ. Since containment in the environment and obstacle avoidance is desired for all times during a task, this leads to the following LTL_{-X} formula over linear predicates in μ and σ:

$$\text{containment and obstacle avoidance:} \quad \Box P_{co}. \tag{21}$$

C. Cohesion and Interrobot Collision Avoidance

Since all pairwise distances scale by the same factor $\sigma(t)/\sigma(0)$ under the affine transformation (15), the initial maximum and minimum pairwise distances remain maximum and minimum at any time. This leads to simple conditions for guaranteeing maximum and minimum distances between robots for all times, which we call cohesion and interrobot collision avoidance, respectively. Let $d_{\min}$ and $d_{\max}$ denote specified minimum and maximum pairwise distances. Let $d_{\min}^0 \geq d_{\min}$ and $d_{\max}^0 \leq d_{\max}$ denote the minimum and maximum pairwise distances at the initial time $t = 0$. Then the predicate

$$P_d : \frac{d_{\min}\sigma(0)}{d_{\min}^0} \leq \sigma(t) \leq \frac{d_{\max}\sigma(0)}{d_{\max}^0} \tag{22}$$

guarantees that at time t, all pairwise distances are between $d_{\min}$ and $d_{\max}$. The specification that cohesion and interrobot collision avoidance are required for all times becomes an LTL_{-X} formula over two linear predicates in the abstract variable σ

$$\text{cohesion and interrobot collision avoidance:} \quad \Box P_d. \tag{23}$$

Remark 4: Due to technical reasons that go beyond the scope of this paper, the LTL_{-X} control algorithm from [14] is restricted to formulas over strict inequalities, as in (1). Therefore, with the price of adding a bit of conservatism, we assume that the inequalities from P_{co} and P_d are strict. We also restrict the additional specifications [*Problem 1*, 2)] to be given in terms of LTL_{-X} formulas over strict linear inequalities in μ and σ. From an application point of view, this assumption makes sense, since it is unreasonable to assume that a sensor could detect equality constraints.

D. Robot Control Bounds

We now map the robot control constraints to constraints for the control of the abstract state. In other words, for an arbitrary polyhedral set $U \subset \mathbb{R}^2$, we construct a polyhedral set $W \subset \mathbb{R}^3$ with the property that $w \in W$ guarantees $u_i \in U, i = 1, \ldots, N$, where w is the velocity in the abstract space and u_i is the control of robot i, which are related by the linear map (17).

Assume U is given in the hyperplane representation

$$U = \{u \in \mathbb{R}^2 | f_k^T u + g_k < 0, \ k \in C\} \tag{24}$$

where $f_k \in \mathbb{R}^2$, $g_k \in \mathbb{R}$, and C is some index set. Let us also denote by $A_i \in \mathbb{R}^{2\times 3}, i = 1, \ldots, N$ the matrix from (17). Then it is easy to see that $u_i \in U$ if and only if $w \in W_i$, where

$$W_i = \{w \in \mathbb{R}^3 | f_k^T A_i w + g_k < 0, \ k \in C\} + \mathcal{N}(A_i) \tag{25}$$

and $\mathcal{N}$ denotes the null space of a matrix. Since the swarm undergoes an affine transformation, there exist constants $\lambda_j^i \geq 0$, $j \in V$ with $\sum_{j\in V} \lambda_j^i = 1$, so that $r_i(t) = \sum_{j\in V} \lambda_j^i r_j$, for all $i = 1, \ldots, N$. From this, we conclude that $u_i(t) = \sum_{j\in V} \lambda_j^i u_j$, $i = 1, \ldots, N$. Therefore, $u_i(t) \in U$ for all $i = 1, \ldots, N$ if and only if $u_j \in U$ for all $j \in V$. In other words, the polyhedral control bounds U are guaranteed for all members of the swarms

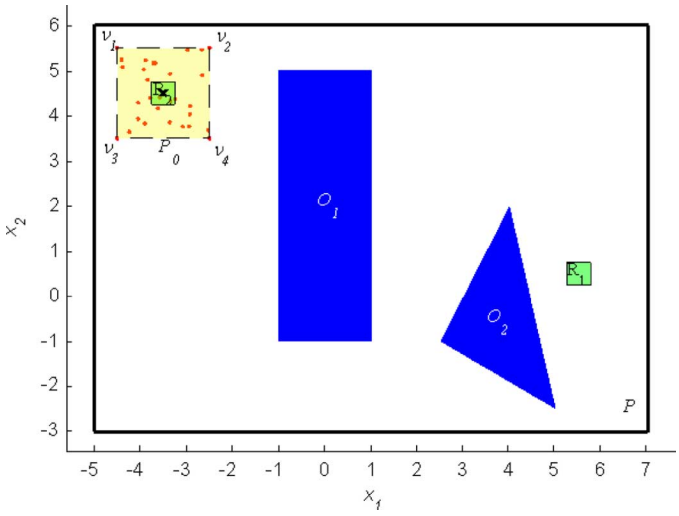

Fig. 3. Initial deployment of a swarm consisting of 30 robots in a rectangular environment P with two obstacles O_1 and O_2. The regions R_1 and R_2 are used in the task specification.

if and only if they are satisfied by the robots at the vertices. This leads to

$$u_i \in U \quad \forall i = 1, \ldots, N \Leftrightarrow w \in W = \bigcap_{j \in V} W_j. \tag{26}$$

Note that the sets W_i from (25) are not polyhedral, since they are products between polyhedral sets and vector spaces $\mathcal{N}(A_i)$. However, it is easy to see that the dimension of all $\mathcal{N}(A_i)$ is 1, and two $\mathcal{N}(A_i)$ and $\mathcal{N}(A_j)$ coincide if and only if $r_i(0) = r_j(0)$, which is excluded since we assume that the robots do not overlap at time 0. We conclude that W from (26) is polyhedral, and the computational framework for the construction of the control strategy $w(a)$ as outlined in Section V can be applied.

E. Case Study

Consider a swarm consisting of $N = 30$ robots moving in a rectangular environment P with two obstacles O_1 and O_2, as shown in Fig. 3. The initial configuration of the swarm is described by mean $\mu(0) = [-3.5,\ 4.5]^T$ and variance $\sigma(0) = 0.903$. The convex hull of the swarm is initially the square of center $\mu(0)$ and side 2 shown in the top left corner of Fig. 3. The cohesion requirement is given in terms of a maximum pairwise distance $d_{\max} = 3.5$, while the interrobot collision avoidance imposes $d_{\min} = 0.01$. The control bounds for robots are captured by the set $U = [-2, 2] \times [-2, 2]$. The corresponding constraint set W as in (26) is an octahedron in $\mathbb{R}^3$ with vertices $(0, 0, -2)$, $(-2, -2, 0)$, $(-2, 2, 0)$, $(2, 2, 0)$, $(2, -2, 0)$, $(0, 0, 2)$. Let R_1 and R_2 be two square regions, as shown in Fig. 3.

Consider the following swarming task given in natural language: *Always respect containment, obstacle avoidance, cohesion, and interrobot collision avoidance. In addition, the centroid μ must eventually visit region R_1. Until then, the minimum pairwise distance must be greater than 0.03. After R_1 is visited, the swarm must reach such a configuration that its centroid is in region R_2 and the spanned area is greater than the initial one, and remain in this configuration forever.* This task translates to the following LTL_{-X} formula over linear predicates in μ and σ:

$$\phi = (P_{co} \wedge P_d) \wedge \{(\sigma > 0.54)\mathcal{U}[(\mu \in R_1) \wedge \Diamond((\mu \in R_2) \wedge (\sigma > \sigma(0)))]\} \tag{27}$$

where P_{co} and P_d were defined in Sections VI-B and VI-C, respectively, and $\sigma > 0.54$ corresponds to pairwise distance greater than 0.03. After eliminating the quantifier from (20), P_{co} consists of 27 occurences of 19 different linear predicates in μ and σ.

By running the algorithms from [14], we conclude that there exists a trajectory in the abstract space $\mathbb{R}^3$ satisfying the formula from the initial values of μ and σ, i.e., the task can be accomplished by the swarm.

The trace of the spanning polytope $\mathcal{P}$ is given in Fig. 4, from which, by close examination, it can be seen that the specified task was accomplished. The robot control constraints are also satisfied during the motion, as it can be seen from Fig. 4(b), where the controls (velocities) of the robots at the vertices of $\mathcal{P}$ are plotted versus time.

F. Implementation and Computational Issues

We developed a program for planning and control of robotic swarms in Matlab. Through a graphical interface, the package takes as input the polygonal environment P, the obstacles O_j, $j = 1, \ldots, o$, the control constraint set U, the initial positions of the robots, and an LTL_{-X} formula ϕ over linear predicates in the mean and variance of the swarm. The program tests the feasibility of the task, computes a control strategy, and displays the produced motion.

From a computational point of view, five main steps are involved.

1) Quantifier elimination for calculation of formula P_{co} from Section VI-B.
2) Generation of the transition system from Section V.
3) Derivation of a Büchi automaton from the LTL_{-X} formula.
4) Calculation of the product automaton and generation of runs.
5) Calculation of abstract controllers and generation of individual robot controllers.

For 1), we used Redlog [25]. Steps 2) and 5) involve polyhedral set operations and triangulations for which we used CDD [26]. For 3), we used LTL2BA [23]. Step 4) involved optimal path generation on graphs for which we used the well-known Dijkstra algorithm. However, the use of all these is transparent to the user, who interacts with the Matlab interface only.

From a complexity point of view, first note that the number N of robots in the swarm has almost no effect on the total amount of computation. Indeed, the number of robots is only involved in the number of terms in the sums defining the abstract variables [(13)]. However, the method we propose in this paper is computationally expensive. The determining factors are (in this order) the quantifier elimination procedure (necessary for construction of formula P_{co} [(20)] and the discrete abstraction procedure (which combines the construction of the transition system and of the Büchi automaton, and finding the runs on the product automaton). The time required for quantifier elimination in the

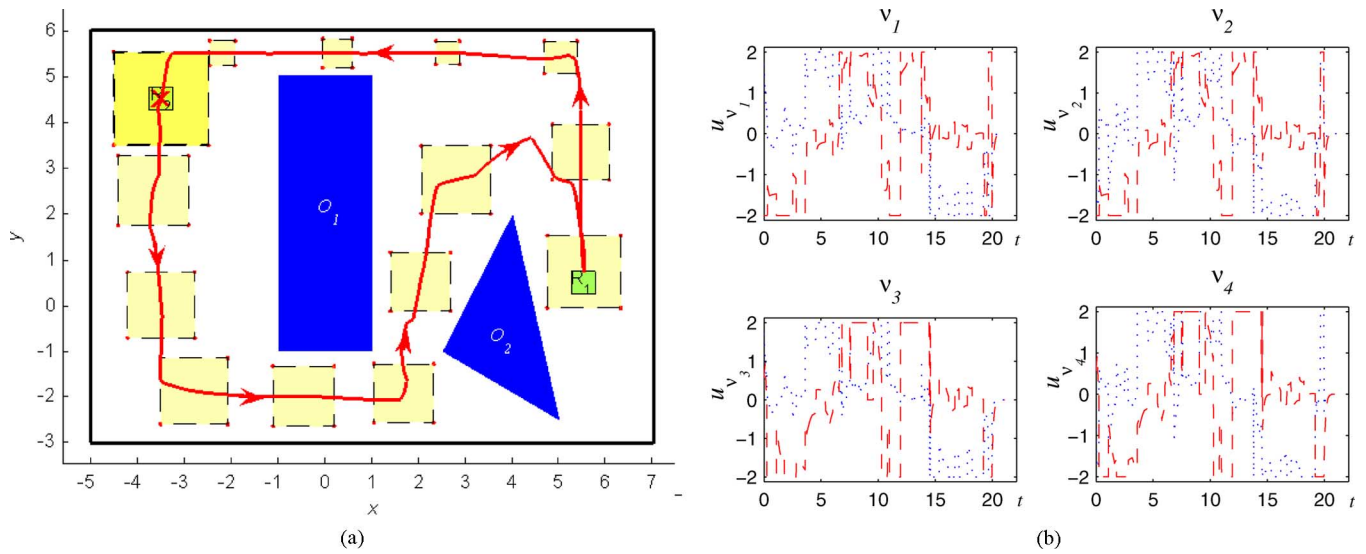

Fig. 4. Simulation results. (a) Trace of the spanning polytope $\mathcal{P}$ (light squares/yellow) and trajectory of centroid μ (dark arrow line/red). (b) Controls u_{ν_i} of the robots ν_i, $i = 1, 2, 3, 4$ at the vertices of $\mathcal{P}$. Dotted line shows shows the time evolution of the velocity in the x direction, while dashed line corresponds to velocity in the y direction.

linear theory of the real closed fields is polynomial in the length of the formula, exponential in the length of the quantified variables, and doubly exponential in the number of quantified blocks [27].[4] In our particular case [see (20)], only the length of the formula can vary, and this is determined by the geometry of the environment (P), obstacles (O_j, $j = 1, \ldots, o$), and initial distribution of the swarm ($\mathcal{P}_0$). The number of states of both the transition system and the Büchi automaton scale exponentially with the number of linear predicates in formula ϕ [14], which is the sum of the predicates in the quantifier-free formula (21) with the number of predicates in the temporal logic specification of the task (in practice, in LTL model checking, this limit was never attained). In conclusion, the determining complexity factors are the geometry of the environment, obstacles, and initial distribution of the swarm, followed by the number of linear predicates in swarm centroid and variance giving the temporal logic specification of the task. Also, from our previous work [14], we know that the size n of the continuous abstraction does not affect the computation significantly, which is encouraging for those interested in extending this work to richer continuous abstractions, such as the ones proposed in [9].

For exemplification, we give here some numerical values obtained by running the case study in Section VI-F on a Pentium 4 (2.66 GHz, 1 GB RAM) machine, with Windows XP and Matlab 7. The transition system from Section V had 744 states, and it was constructed in about 60 s. The accepted run [corresponding to initial condition $(\mu(0), \sigma(0))$] and the corresponding vector fields (6) and (17) were generated in about 12 s.

G. Conservativeness and Limitations of the Approach

Our solution to *Problem 1* is obviously conservative, in the sense that our algorithm might not produce a solution even though a motion of the swarm satisfying the specification might exist. There are three main sources of conservativeness in our approach.

First, in order to guarantee that the swarm does not collide with an obstacle, we impose that the whole convex hull of the swarm has empty intersection with the obstacle. One can imagine that there might exist motions of the swarm where an obstacle enters the convex hull without hitting any of the robots. Second, we restrict our attention to a very small set of essential features, which only allow for translation and scaling of the convex hull. For example, if rotation was allowed, motions would have been possible where the spanning polytope would rotate to avoid obstacles, rather than just unnecessarily shrinking. Third, our approach to the control of the essential features of the swarm based on discrete abstractions is conservative. Indeed, in the general framework for control of linear systems from temporal logic specifications developed in [14], we only focus on determining sets of initial states from which a solution exists in the form of unions of full-dimensional polytopes. However, while introducing more conservativeness into the problem, this approach has the advantage of conferring robustness of the solution with respect to exact knowledge of the initial or current state.

While providing a fully automated solution for planning and control of an arbitrarily large swarm from specifications given in human-like language, our approach has three main limitations. First, in its current form, it assumes that the environment is static and known. Any change in the boundaries of the environment or in the obstacles should be followed by a full recomputation. This being said, for a static and known environment, our framework is robust with respect to small errors in knowledge about the environment. This results from the fact that the discrete abstraction procedure for control of the continuous abstraction from temporal logic specifications over linear predicates (developed in our previous work [14]) is robust to small changes in these predicates.

[4]This is significantly better than the time required for quantifier elimination in the full theory of the real closed fields (i.e., the terms are polynomials in the variables with rational coefficients), which is, in general, doubly exponential in the number of variables and polynomial in the number of predicates.

Second, for the particular example that we consider, the communication architecture is centralized. However, as already mentioned at the beginning of Section VI, it can be implemented using a leader agent (such a UAV flying above the swarm) that has to broadcast a signal of small bandwidth. Moreover, a simple decentralized version of this framework can be constructed: assume that each agent uses exactly the same control law, but can only compute the mean and variance of its neighbors. Preliminary numerical simulations show that if the individual neighborhoods overlap enough, then the behavior of the swarm is exactly as the one obtained in the centralized case.

Third, throughout this paper, we assume that the robots are fully actuated point masses. However, to accommodate robots of nonnegligible sizes, one can enlarge the obstacles and shrink the environment boundaries [28]. To accommodate more complex dynamics, one can use one of the following two approaches. First, one can add another level in the hierarchy. For example, using input–output regulation, controlling a unicycle can be reduced to controlling the velocity of an off-axis reference point [29]. Second, one might try to capture robot underactuation constraints by properly constructing the continuous abstraction, as suggested in [9].

VII. Conclusion

We proposed a fully automated framework for deployment of arbitrarily large swarms of fully actuated robots. Our approach is hierarchical. In the first level of the hierarchy, we aggregate the large-dimensional state space of the swarm into a small-dimensional continuous abstract space which captures essential features of the swarm. In the second level, we control the continuous abstraction so that specifications given in linear temporal logic over linear predicates in the essential features are satisfied. Individual robot control laws are generated by projection. For planar robots with polyhedral control constraints moving in polygonal environments with polygonal obstacles, and a 3-D continuous abstraction consisting of mean and variance, we show that a large class of specifications are captured. Directions of future work include the development of larger and more expressive continuous abstractions, modeling of communication requirements, and development of provable robot control laws based on local information only.

References

[1] T. Eren, P. N. Belhumeur, and A. S. Morse, "Closing ranks in vehicle formations based rigidity," in *Proc. IEEE Conf. Decision Control*, Las Vegas, NV, Dec. 2002, vol. 3, pp. 2959–2964.

[2] R. Olfati-Saber and R. M. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Trans. Autom. Control*, vol. 49, no. 9, pp. 1520–1533, Sep. 2004.

[3] M. Egerstedt and X. Hu, "Formation constrained multi-agent control," in *Proc. IEEE Conf. Robot. Autom.*, Seoul, Korea, May 2001, vol. 4, pp. 3961–3966.

[4] P. Ogren, E. Fiorelli, and N. E. Leonard, "Cooperative control of mobile sensor networks: Adaptive gradient climbing in a distributed environment," *IEEE Trans. Autom. Control*, vol. 49, no. 8, pp. 1292–1302, Aug. 2004.

[5] C. Belta and V. Kumar, "Optimal motion generation for groups of robots: A geometric approach," *J. Mech. Des.*, vol. 126, pp. 63–70, 2004.

[6] R. G. Littlejohn and M. Reinsch, "Internal or shape coordinates in the n-body problem," *Phys. Rev. A*, vol. 52, no. 3, pp. 2035–2051, 1995.

[7] F. Zhang, M. Goldgeier, and P. S. Krishnaprasad, "Control of small formations using shape coordinates," in *Proc. IEEE ICRA*, Taipei, Taiwan, 2003, vol. 2, pp. 2510–2515.

[8] E. Justh and P. Krishnaprasad, "Steering laws and continuum models for planar formations," in *Proc. IEEE Conf. Decision Control*, 2003, pp. 3609–3614.

[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] G. J. Pappas and S. Simic, "Consistent abstractions of affine control systems," *IEEE Trans. Autom. Control*, vol. 47, no. 5, pp. 745–756, May 2002.

[11] P. Tabuada and G. J. Pappas, "Hierarchical trajectory generation for a class of nonlinear systems," *Automatica*, vol. 41, no. 4, pp. 701–708, 2005.

[12] ——, "Bisimilar control affine systems," *Syst. Control Lett.*, vol. 52, no. 1, pp. 49–58, 2004.

[13] M. Broucke, "A geometric approach to bisimulation and verification of hybrid systems," in *Hybrid Systems: Computation and Control*, F. W. Varager and J. H. van Schuppen, Eds. New York: Springer-Verlag, 1999, vol. 1569, Lecture Notes in Computer Science, pp. 61–75.

[14] M. Kloetzer and C. Belta, J. Hespanha and A. Tiwari, Eds., "A fully automated framework for control of linear systems from LTL specifications," in *Proc. Hybrid Syst.: Comput. Control: 9th Int. Workshop*, Berlin, Heidelberg, 2006, vol. 3927, Lecture Notes in Computer Science, pp. 333–347.

[15] P. Tabuada and G. Pappas, "Model checking LTL over controllable linear systems is decidable," in *Hybrid Systems: Computation and Control*, O. Maler and A. Pnueli, Eds. New York: Springer-Verlag, 2003, vol. 2623, Lecture Notes in Computer Science, pp. 498–513.

[16] C. Belta, V. Isler, and G. J. Pappas, "Discrete abstractions for robot planning and control in polygonal environments," *IEEE Trans. Robot.*, vol. 21, no. 5, pp. 864–874, Oct. 2005.

[17] S. G. Loizou and K. J. Kyriakopoulos, "Automatic synthesis of multiagent motion tasks based on LTL specifications," in *Proc. 43rd IEEE Conf. Decision Control*, Dec. 2004, vol. 1, pp. 153–158.

[18] M. M. Quottrup, T. Bak, and R. Izadi-Zamanabadi, "Multi-robot motion planning: A timed automata approach," in *Proc. IEEE Int. Conf. Robot. Autom.*, New Orleans, LA, Apr. 2004, pp. 4417–4422.

[19] G. E. Fainekos, H. Kress-Gazit, and G. J. Pappas, "Temporal logic motion planning for mobile robots," in *Proc. IEEE Int. Conf. Robot. Autom.*, Apr. 2005, pp. 2020–2025.

[20] E. A. Emerson, "Temporal and modal logic," in *Handbook of Theoretical Computer Science: Formal Models and Semantics*, J. van Leeuwen, Ed. Amsterdam, The Netherlands: North-Holland/MIT Press, 1990, vol. B, pp. 995–1072.

[21] M. Spivak, *A Comprehensive Introduction to Differential Geometry*. Berkeley, CA: Publish or Perish, 1979.

[22] R. Milner, *Communication and Concurrency*. Englewood Cliffs, NJ: Prentice-Hall, 1989.

[23] P. Gastin and D. Oddoux, H. C. G. Berry and A. Finkel, Eds., "Fast LTL to Büchi automata translation," in *Proc. 13th Conf. Comput. Aided Verification*, 2001, vol. 2102, LNCS, pp. 53–65.

[24] M. Kloetzer, L. Habets, and C. Belta, "Control of rectangular multiaffine hybrid systems," in *Proc. 45th IEEE Conf. Decision Control*, San Diego, CA, 2006, pp. 2619–2624.

[25] V. Weispfenning, "A new approach to quantifier elimination for real algebra," Universität Passau, Germany, Tech. Rep. MIP-9305, 1993.

[26] K. Fukuda, "cdd/cdd+ package," [Online]. Available: http://www.cs.mcgill.ca/~fukuda/soft/cdd_home/cdd.html

[27] V. Weispfenning, "The complexity of linear problems in fields," *J. Symbolic Comput.*, vol. 5, no. 1-2, pp. 3–27, Feb./Apr. 1988.

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

[29] 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.

Marius Kloetzer (S'05) received the B.S. and M.Sc. degrees in computer science from the Technical University of Iasi, Iasi, Romania. He is currently working toward the Ph.D. degree in systems engineering at Boston University, Boston, MA.

His research interests include robot motion planning and verification and control of hybrid systems.

Calin Belta (S'00–M'03) received the B.S. and M.Sc. degrees in control and computer science from the Technical University of Iasi, Iasi, Romania, the M.Sc. degree in electrical engineering from Louisiana State University, Baton Rouge, and the M.Sc. and Ph.D. degrees in mechanical engineering from the University of Pennsylvania, Philadelphia.

He is currently an Assistant Professor in the Department of Manufacturing Engineering, Boston University, Boston, MA. His research interests include verification and control of hybrid systems, robot motion planning and control, gene and metabolic networks .

Dr. Belta received a National Science Foundation CAREER award in 2005, a Fulbright study award in 1997, and was the Valedictorian of his class in 1995. He received the Best Paper Award at the International Conference on Systems Biology in 2004, and was a finalist for the ASME Design Engineering Technical Conference Best Paper Award in 2002, and for the Anton Philips Best Student Paper Award at the IEEE International Conference on Robotics and Automation in 2001.