2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

Automatic Control Synthesis for Swarm Robots from Formation and Location-based High-level Specifications

Ji Chen[1], Hanlin Wang[2], Michael Rubenstein[2], and Hadas Kress-Gazit[1]

***Abstract*— In this paper, we propose an abstraction that captures high-level formation and location-based swarm behaviors, and an automated control synthesis framework to generate correct-by-construction behaviors. Our abstraction includes symbols representing both possible formations and physical locations in the workspace. We allow users to write linear temporal logic (LTL) specifications over the symbols to specify high-level tasks for the swarm. To satisfy a specification, we automatically synthesize a centralized symbolic plan, and environment and swarm-size-dependent motion controllers that are guaranteed to implement the symbolic transitions. In addition, using integer programming (IP), we assign robots to different sub-swarms to execute the synthesized symbolic plan. Our framework gives insights into controlling a large fleet of autonomous robots to achieve complex tasks which require composition of behaviors at different locations and coordination among different groups of robots in a correct-by-construction way. We demonstrate the proposed framework in simulation with 16 UAVs and 8 ground vehicles, and on a physical platform with 20 ground robots, showcasing the generality of the approach and discussing the implications of controlling constrained physical hardware.**

I. Introduction

Swarm and multi-robot systems can perform various tasks such as surveillance [1], warehouse logistics [2], and object transport [3]. Such tasks might require robots to achieve certain formations for communication and collaboration, or even for entertainment [4].

Control laws that cause swarms to create a specific shape are referred to as formation control [5]. Many aspects of formation control have been studied, such as stability [6] and decentralized coordination [7]. However, work in formation control typically does not consider how to create control that ensures a formation happens in a specific location in the workspace, nor has past work enabled users to specify complex behaviors that include multiple different shapes, for different subswarms, in different locations, and automatically generate the appropriate control.

In this paper, we create abstractions for formations and locations of swarms, and utilize the abstractions to synthesize compositional and flexible behaviors in a provably-correct way from user specifications. The framework we describe could impact a variety of application domains from digital agriculture where a farmer may deploy a swarm of UAVs for monitoring and spraying behaviors, to entertainment, where a robotics-novice designer might create complex formations, to construction where a civil engineer may deploy swarms as scaffolding for different structures.

This work leverages our previous work [8], [9], where [8] focuses on synthesizing decentralized symbolic plans given high-level specifications, and [9] creates continuous control guaranteed to avoid collision, physically implement the symbolic plans, and mitigate deadlocks. Expanding on [8], [9] that only consider region-based specifications, in this work we create richer abstractions and specifications corresponding to both swarm formation and location. In addition, we design continuous controllers that are parametric in the specified formation and swarm size, and use a centralized assignment mechanism to ensure the execution of the symbolic plan satisfies the user-provided specification and avoids collisions.

We consider this work as addressing high-level behaviors for swarms even though our solution is centralized and may be considered as multi-robot. This is because our abstractions, specification formalism, and automatic control synthesis are agnostic to the exact number of robots which may change during execution, a characteristic of swarms.

Assumptions: In this work, we assume the environment and robot poses are known. Furthermore, we synthesize controls and provide guarantees for the robots assuming that they are holonomic; when controlling physical, differential drive robots, we increase the safety distance to mitigate the effects of the dynamic constraints, as described in Section VIII.

Paper contributions: This paper presents (i) a novel abstraction and grammar that allow a user to specify location and formation-based swarm behaviors, (ii) automated correct-by-construction control synthesis for the swarm robots that guarantees that the task will be satisfied, if feasible, and (iii) demonstrations on simulated and physical swarm platforms that showcase the generality of the approach.

II. Related Work

This paper lies at the intersection of formation control and automated control synthesis from high-level specifications.

Formation control has been studied extensively in swarm and multi-robot systems research [10]. Some work develops continuous control laws based on different robot sensing and interaction capabilities [10]: for example, a camera-based cooperative control framework for nonholonomic robots [11] and a position-based method with dynamic communication topology [12], while other works utilize graph theory to convert multi-robot formation control to a discrete path planning

[1]Ji Chen and Hadas Kress-Gazit are with Sibley School of Mechanical and Aerospace Engineering, Cornell University, Ithaca, NY 14850, USA jc3246@cornell.edu, hadaskg@cornell.edu

[2]Hanlin Wang and Michael Rubenstein are with the Department of Computer Science, Northwestern University, Evanston, IL 60201, USA hanlinwang2015@u.northwestern.edu, rubenstein@northwestern.edu

problem [13], such as robots forming lattice patterns [14] and switching formations on grids [15], [16]. Our work is different in that we tackle high-level behaviors that include both the sequencing of formations but also the automatic distributions of the robots into several sub-swarm formation, to achieve the desired high-level task in the continuous space.

Recently, compositions of swarm behaviors have been studied [17], [18]. The work in [17] focuses on continuous control for reaching certain spatial configurations and switching between different behaviors to guarantee effective information flow. The work in [18] focuses on selecting an optimal sequence of behaviors to achieve a certain task. In our paper, we focus on creating abstractions and synthesizing controls from user-defined formations and temporal logic specifications, which automates the mission planning and sub-swarm assignment process compared to [17], [18].

Control synthesis from high-level specifications has been applied to multi-robot systems and swarms [19]–[24]. For example, [19] deals with asynchronous motions of robots in high-level mission planning to guarantee collision and deadlock avoidance, and [20] composes motion primitives based on high-level specifications to maintain spatial configurations on a grid. In [21], the authors develop specifications for swarm robots and formally prove the system properties. Authors in [22] developed a framework to automatically generate optimal behaviors for heterogeneous multi-robot teams to achieve missions specified in temporal logic, and [23] describes a method to generate a sequence of multi-robot policies for task allocation and planning from LTL specifications. In [24], the authors present a mechanism for a team of robots to coordinate to fulfill tasks given in LTL under uncertainty. Our work also considers task allocation and coordination during task execution, but different from [22]–[24] where each robot has its own specification, we assign temporal goals to an arbitrary number of robots under formation-dependent constraints and synthesize controls to achieve a global task specified in LTL.

III. Preliminaries

A. Linear Temporal Logic

Linear temporal logic (LTL) [25] is a formal language consisting of propositions and logical and temporal operators. Let AP be a set of atomic propositions. The syntax of LTL is defined as follows:

$$\varphi ::= \pi \,|\, \neg\varphi \,|\, \varphi \vee \varphi \,|\, \bigcirc \varphi \,|\, \varphi \mathcal{U} \varphi$$

where $\pi \in AP$ is a proposition, $\neg$ is negation, $\vee$ is disjunction, $\bigcirc$ is next and $\mathcal{U}$ is until. Other logical operators such as conjunction ($\wedge$), implication ($\Rightarrow$) and temporal operators such as *always* ($\Box$) and *eventually* ($\Diamond$) can be derived from these basic operators. An LTL formula over propositions AP is interpreted over infinite words $w \in (2^{AP})^{\omega}$. The language of an LTL formula φ, denoted by $\mathcal{L}(\varphi)$, is the set of infinite words that satisfy φ, i.e., $\mathcal{L}(\varphi) = \{w \in (2^{AP})^{\omega} \mid w \models \varphi\}$. Intuitively, $\Box\pi$ means proposition π has to be true at all times, $\Diamond\pi$ means proposition π will be true at some point, and $\bigcirc\pi$ means that proposition π has to be true in the next step. We refer the readers to [25] for more details about LTL.

B. Abstractions and Specifications for Swarms

In [8], [9], [26] we propose a region based abstraction for swarm behaviors. We partition a 2D workspace into a set of regions $\mathbf{R} = \{r_1, ..., r_m\}$, and use a region graph $\mathbf{G_R} = (\mathbf{R}, \mathbf{E})$ to represent the connectivity of the regions, where $\mathbf{E} \subseteq \mathbf{R} \times \mathbf{R}$ represents possible transitions between regions, i.e., $(r_i, r_j) \in \mathbf{E}$ indicates robots can move directly from region r_i to r_j without going through any other region. We define $\pi_r \in AP$ as an atomic proposition which is true *iff* at least one robot is in region $r \in \mathbf{R}$. By using these propositions, a user can write location-based specifications in the following format:

$$\varphi = \phi^i \wedge \bigwedge_j \phi^s_j \wedge \bigwedge_k \phi^g_k, \tag{1}$$

where ϕ^i is a Boolean formula over AP representing the initial condition, $\phi^s_j = \Box\varphi^s_j$ are *safety* constraints where φ^s_j is a Boolean formula over $AP \bigcup \bigcirc AP$ representing conditions that must always hold, and $\phi^g_k = \Box\Diamond\varphi^g_k$ are system *livenesses* where φ^g_k is a Boolean formula over AP representing goals that must eventually be reached. For example, a formula $\varphi = r_1 \wedge \Box\neg r_2 \wedge \Box\Diamond r_m$ specifies that the swarm initially starts at region r_1, should repeatably visit r_m, and never enter r_2. Formula (1) has similar structure to *generalized reactivity (1)* (GR(1)) [27] which is a fragment of LTL that has polynomial time complexity in synthesis. However, different from GR(1), formula (1) does not contain environment propositions as the environment is assumed static and known.

C. Control Synthesis for Swarms

In our previous work [8], [9], [26], we proposed a control synthesis framework to satisfy specifications of swarm navigation tasks. We first apply LTL synthesis to obtain a symbolic plan given the high-level specifications. The symbolic plan $\mathcal{S} = \{q_0 q_1 q_2 ...\}$ can be represented as a sequence of states q_i such that each state q_i is labeled with a set of propositions $L(q_i) \subseteq AP$ that are true in state q_i. Then, we partition the symbolic plan into decentralized plans and execute the plans on different sub-swarms. By using control barrier functions (CBF) [28], we generate continuous controls that drive robots to reach the goal in collision-free paths. The synthesis framework guarantees that the swarm executes the task while avoiding collisions and satisfies the specifications, when possible, although there may be instances of deadlock which we mitigate.

IV. Problem Formulation and Approach

This paper solves the following problem: Given a workspace that is partitioned into a set of regions, a set of 2D/3D shapes, a robot swarm and formation (shapes) and location (regions) based high-level tasks, we automatically synthesize controls for all individual robots such that the robots correctly and safely satisfy the task. We assume the

robots have full state information, i.e., their pose is known, and that they are velocity controlled.

To solve this problem we first create abstractions for formation and location-based robot behaviors, and then allow users to specify tasks over those abstractions using the LTL formulas in Eq. (1). We automatically synthesize a symbolic plan by leveraging the work in [8], and we use integer programming (IP) to assign robots to sub-swarms for achieving the symbolic plan. We automatically create continuous control for individual robots that implements the symbolic plan while guaranteeing collision avoidance.

V. High-level Specifications and Symbolic Synthesis

In this section we define the abstractions over which we create propositions that we use for the task specifications, and describe the specifications and the symbolic synthesis.

A. Abstractions: symbols and their physical grounding

Our abstraction contains three types of symbols; *location*, *target*, and *formation*, that allow a user to capture formation and location-based swarm behaviors.

1) Location symbols: As done in [9], [26], we partition the continuous workspace into regions and create a symbol for each region $\mathbf{R} = \{r_1, ..., r_m\}$. These symbols are grounded to the physical space and when used as propositions (Section V-B), the proposition are true only if there is at least one robot in the corresponding region. The physical space and regions can be either 2D or 3D, as shown in the following.

2) Target symbol: We define the symbol T that represents existence of a target in a region. In section V-B we use this symbol in conjunction with formation symbols to create propositions whose truth values depend on whether the formation surrounds the target.

3) Formation symbols: We define formation symbols as a set $\mathbf{F} = \{C, F_1, ..., F_k\}$ where C represents coverage (robots distribute in a region according to a given coverage metric as described in Section VI-C), and $F_i (i \in \{1, ..., k\})$ is a shape formation.

We allow two types of shape formation in this paper: i) forming a perimeter of a 2D polygon or ii) surface/curve coverage of a 3D shape. For 2D shapes, a formation symbol F_i represents a polygon (a list of vertices) $s \in \mathbf{S}$ where

$$\mathbf{S} = \{[(x_1, y_1), ..., (x_p, y_p)] | x_i, y_i \in \mathbb{R}, p \geq 3\}.$$

For example, we can use $s_{sqr} = [(0,0), (1,0), (1,1), (0,1)]$ and $s_{tri} = [(1,0), (0,\sqrt{3}), (-1,0)]$ to represent a square and an equilateral triangle respectively. Note that the coordinates are only used to determine the shape rather than the actual formation size which is scaled based on the region.

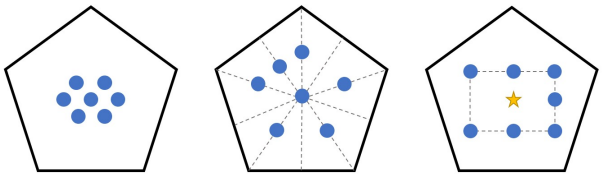
For 3D shapes, the formation symbol F_i corresponds to a parametric function $\mathbf{p} : \mathbb{R} \rightarrow \mathbb{R}^3$ representing a 3D shape/curve. For example, $\mathbf{p}(t) = [rcos(t), rsin(t), ct]$ $(0 \leq t \leq 6\pi)$ represents a helix with a radius of r and height of $6c\pi$. In this paper, we use the set of formation symbols $\mathbf{F} = \{C, sqr, tri, dia, hex, oct, pen, helix, sph\}$, where the symbols represent coverage, a square, a triangle, a diamond, a hexagon, an octagon, a five-point star, a helix (3D), and a sphere (3D) respectively.

B. Proposition Creation

Given the symbols defined in section V-A, we define the set of atomic proposition over which the specifications will be created, as $\mathbf{Prop} = (\mathbf{F} \cup \{\emptyset\}) \times \mathbf{R} \times \{T, \emptyset\}$, where $\times$ is the Cartesian product. We use the notation $F_j\_r_i\_T$ for the tuple (F_j, r_i, T), C_r_i for both $(C, r_i, \emptyset)$ and (C, r_i, T) (same semantics), $F_j\_r_i$ for $(F_j, R, \emptyset)$, and r_i for both $(\emptyset, r_i, \emptyset)$ and $(\emptyset, r_i, T)$ (same semantics), where $F_j \in \mathbf{F}$ and $r_i \in \mathbf{R}$. The semantics of the propositions are defined as:

- $r_i = true$ indicates that at least one robot is in region r_i, regardless of formation.
- $C\_r_i = true$ indicates that at least one robot spreads out to cover the region represented by r_i. We note that "at least one robot" is the semantic definition due to r_i, but in practice a sub-swarm will cover the whole region.
- $F_j\_r_i\_T = true$ indicates that robots form the shape F_j around the target in region r_i with the target inside the shape, and $F_j\_r_i = true$ indicates that robots form the shape F_j in region r_i disregarding the location of the target, even if one exists.

Note that the proposition $F_j\_r_i\_T = true$ requires that there exists a target in region r_i, which might not be the case. Thus, we relax the semantics by assigning $F_j\_r_i\_T = true$ when robots form shape F_j anywhere in r_i if there is no target in region r_i. Figure 1 depicts possible physical representations of different atomic propositions.



(a) A swarm satisfying r_i. (b) A swarm satisfying C_r_i (c) A swarm satisfying $rect\_r_i\_T$

Fig. 1: Proposition satisfying formations for seven robots in a pentagonal region. Blue dots represent locations of robots. The yellow star represents the target in the region.

C. Specifications

Given a set of propositions, $\mathbf{Prop}$, users can write specification of the form (1). Given such a formula, we automatically reduce the number of propositions for synthesis and add formulas representing the swarm motion constraints, as described below.

1) Minimizing the set of propositions: Given the set of propositions that appear in the user's specifications, $\mathbf{Prop}_{spec}$, we create the set $\mathbf{Prop}' = \mathbf{Prop}_{spec} \bigcup \mathbf{R} \subseteq \mathbf{Prop}$, that will be used in the synthesis process. Calculating $\mathbf{Prop}'$ can reduce the number of propositions, which makes the synthesis faster as synthesis time grows exponentially with the number of propositions in the worst case [8].

2) Encoding motion constraints: In addition to the user specification, we automatically create formulas that represent motion constraints due to the topology and the formations.

For each region $r \in \mathbf{R}$, we first define sets of outgoing and incoming neighbors of r as $\mathbf{out}(r) = \{r' \in \mathbf{R} | (r, r') \in \mathbf{E}\}$ and $\mathbf{in}(r) = \{r' \in \mathbf{R} | (r', r) \in \mathbf{E}\}$, respectively. Then, we add for each r two *safety* formulas to the specifications: $\phi_{G1} = \square(r \rightarrow \bigcirc r \vee \bigvee_{r' \in \mathbf{out}(r)} \bigcirc r')$ and $\phi_{G2} = \square(\bigwedge_{r' \in \mathbf{in}(r)} \neg r' \wedge \neg r \rightarrow \bigcirc \neg r)$. The formula ϕ_{G1} indicates that robots in region r can only move to the outgoing neighbors of r or stay in r at the next step, and ϕ_{G2} indicates that robots cannot be in r at the next step if currently there is no robot located in r or any of its incoming neighbors.

To ensure formations in specific regions, we specify a *safety* formula $\phi_p = \bigwedge_{F\_r\_T \in \mathbf{Prop}', F \in \mathbf{F}} \square(F\_r\_T \rightarrow (r \wedge \bigcirc r)) \wedge \bigwedge_{F\_r \in \mathbf{Prop}', F \in \mathbf{F}} \square(F\_r \rightarrow (r \wedge \bigcirc r))$ which encodes that if a sub-swarm achieves a formation in region r, there exist at least one robot in region r currently and at the next step. Intuitively, the safety formula connects formation and location to ensure that transition between formations is reachable in physical space given the environment topology.

Example: Given the workspace in Fig. 4 and the formation set $\mathbf{F}$ in section V-A, the task is to: 1) repeatedly and visit r_1 and form a triangle in r_2 at the same time; 2) repeatedly form diamonds in r_1 and r_2 around the targets, and cover r_3 at the same time; and 3) never enter r_4. Using the symbols, the user can write the LTL formulas as:

- $\phi_{u1} = \square\lozenge(r_1 \wedge tri\_r_2)$
- $\phi_{u2} = \square\lozenge(dia\_r_1\_T \wedge dia\_r_2\_T \wedge C\_r_3)$
- $\phi_{u3} = \square\neg r_4$

Based on the task, $\mathbf{Prop}' = \{r_0, r_1, r_2, r_3, r_4, tri\_r_2, dia\_r_1\_T, dia\_r_2\_T, C\_r_3\}$. The added motion constraints are $\phi_p = \bigwedge_{j=1,\ldots,4} \phi_{pj}$:

- $\phi_{p1} = \square(tri\_r_2 \rightarrow (r_2 \wedge \bigcirc r_2))$
- $\phi_{p2} = \square(dia\_r_1\_T \rightarrow (r_1 \wedge \bigcirc r_1))$
- $\phi_{p3} = \square(dia\_r_2\_T \rightarrow (r_2 \wedge \bigcirc r_2))$
- $\phi_{p4} = \square(C\_r_3 \rightarrow (r_3 \wedge \bigcirc r_3))$.

The full specifications is

$$\Phi = \bigwedge_{i=1,2,3} \phi_{ui} \wedge \bigwedge_{i=1,2} \phi_{Gi} \wedge \phi_p.$$

Expressiveness: Given the abstractions, we can specify tasks that: 1) require robots to visit or avoid regions, and 2) require robots to form shapes in certain regions (at most one shape in each region), potentially around a target, or cover regions. Our abstractions currently do not allow multiple formations in one region or a formation that is created across multiple regions, however, we note that the choice or workspace partition and formation shapes are up to the user.

D. Synthesis

Given a specification, we use *Slugs* [29] to generate the symbolic plan $\mathcal{S}$ and we leverage the work in [8] to verify that the swarm will exhibit correct continuous behavior when continuously implementing the symbolic plan. For the example in Section V-C, synthesizing the symbolic plan took $0.655s$ and the plan included 6 states.

VI. Continuous Control Synthesis and Execution

In this section we first briefly explain the continuous execution from a given a synthesized symbolic plan. Then, we formalize the integer programming (IP) based approach to dynamically assign robots to sub-swarms. Finally, we define the continuous control synthesis for each individual robot.

A. Continuous Execution

Given a synthesized symbolic plan $\mathcal{S}$ from Section III-C, we describe the continuous execution of a transition from state q_i to q_{i+1}:

First, we obtain the desired formations (including region visit and coverage) and the goal regions from $L(q_{i+1})$. Then, we calculate the minimum and maximum number of robots needed to achieve each desired formation, and use an IP based approach (Section VI-B) to assign robots to sub-swarms. Finally we apply the continuous control (Section VI-C) with control barrier functions [28] to each sub-swarm to achieve the desired formations with no collisions.

B. Sub-swarm Assignment

For correct continuous implementation of the symbolic plan, we must assign robots to sub-swarms based on the required symbolic transitions. Our method is based on linear assignment [30], which deals with optimally assigning n items (robots) to n places (goals). The main difference with respect to [30] is that we add constraints to the IP formulation to ensure the correct execution of the symbolic plan and enforces the topological constraints of the space, i.e. robots can only move to their neighboring regions or stay in the same region.

For N robots in a workspace partitioned into n regions, we define an assignment matrix $M \in \mathbb{R}^{N \times n}$ as

$$[M]_{ij} = \begin{cases} 1 \text{ if robot } i \text{ is assigned to region } j \\ 0 \text{ otherwise.} \end{cases}$$

For each symbolic transition (q_i, q_{i+1}), and for each region appearing in $L(q_{i+1})$, we define $N^{min}_{L(q_{i+1}),j}$ and $N^{max}_{L(q_{i+1}),j}$ as the minimum and maximum number of robots required to be in r_j in state q_{i+1} for the desired formation. In the following we will use N^{min}_j and N^{max}_j to denote $N^{min}_{L(q_{i+1}),j}$ and $N^{max}_{L(q_{i+1}),j}$ for simplicity. Furthermore, we define C_{ij} as the cost of robot i moving into region j. We can obtain an optimal assignment strategy by solving the following IP:

$$\begin{aligned} M = \underset{[M]_{ij} \in \{0,1\}}{\operatorname{argmin}} \quad & \sum_{i=1}^{N} \sum_{j=1}^{n} M_{ij} C_{ij} \\ s.t. \quad & \sum_{j=1}^{n} M_{ij} = 1 \quad \forall 1 \le i \le N \\ & \sum_{i=1}^{N} M_{ij} \ge N^{min}_j \quad \forall 1 \le j \le n \\ & \sum_{i=1}^{N} M_{ij} \le N^{max}_j \quad \forall 1 \le j \le n \\ & M_{ij} = 0 \quad \text{if } (r_{k_i}, r_j) \notin \mathbf{E}, \end{aligned} \tag{2}$$

where r_{k_i} is the region where robot i is currently located. We use the estimated travel distance for robot i to enter region j as the cost C_{ij}, thus we are optimizing the sum of the distances the robot travel. The cost function can be easily changed to capture other metrics such as minimum time.

The resulting assignment from (2) ensures that robots only move to their neighboring regions and the sub-swarm size fits the corresponding behaviors as long as N_j^{min} and N_j^{max} are well defined.

Given D_s, the inter-robot safety distance, we automatically determine N_j^{min} and N_j^{max} as follows:

Visit and Coverage. For 2D region visit and coverage, we set $N_j^{min} = 1$. For computing N_j^{max}, we first estimate the maximum number as $N_{j\_e}^{max} = \frac{A(r_j)}{\beta A_R}$, where $A(r_j)$ is the area of r_j, A_R is the area covered by a single robot, and $\beta > 1$ is a parameter. $N_{j\_e}^{max}$ is an initial guess of N_j^{max} based on the ratio of the region area and the robot dimensions, and β can be defined based on the safety distance between robots. We verify the result $N_{j\_e}^{max}$ by finding the goals given $N_{j\_e}^{max}$ and r_j using the method in Section VI-C, and let $N_j^{max} = N_{j\_e}^{max}$ if there exists a solution, otherwise we replace $N_{j\_e}^{max}$ by $N_{j\_e}^{max} - 1$ and repeat the process until a solution for the goals is found. For 3D regions, we choose the central plane of the region to calculate N_j^{min} and N_j^{max} as the controller we use for region visit and coverage is the same as in 2D.

Shape formation. For 2D polygon forming, we let $N_j^{min} = p$ where p is the number of vertices of the specified shape as defined in Section V-A. We enforce robots to at least occupy all the vertices of a given shape. For the maximum number, we make $N_j^{max} = \sum[l_i/D_s]$ where l_i is the length of each edge and $[x] = max\{m \in \mathbb{Z} | m \leq x\}$. The intuition is to ensure that the distances between goal points are at least D_s.

The edge length l_i is determined based on the formation $\mathbf{S}$ defined in V-A and the actual region size once the user has created the formation symbols. We assume $min(l_i) > D_s$ to avoid unsafe occupancy at the vertices. For 3D curve/surface forming, we have the user specify N_j^{min} and N_j^{max} as there are no vertices that robots have to occupy, and we verify that minimum distance between generated goal points is larger than the safety distance given N_j^{max}.

Optimality and feasibility: The robot assignment happens once for each transition in the symbolic plan $\mathcal{S}$, thus the assignment optimizes one transition; the assignment is not necessarily optimal for the whole task. In addition, when the number of robots cannot satisfy the constraints in Eq. 2, the execution will terminate as the task cannot be correctly executed even though the high-level specifications are realizable. In [8] we discussed an IP-based method to calculate ahead of time the minimum number of robots for each region in each state of the symbolic plan, needed to guarantee the execution of location-based tasks. In the future we will apply the method to formation-based tasks to provide N_j^{min} and N_j^{max} as the assignment constraints.

Computation complexity: Integer programming is generally NP-complete [31]. However, when the constraints and objective functions are linear, the IP becomes an *integer linear programming* and it can be solved in polynomial time. In the IP formulation (2), all the equality constraints can be converted to a combination of two inequalities constraints, and all the constraints as well as the objective function are linear. Thus, the assignment can be calculated in polynomial time with respect to the number of robots. Our formulation can easily optimizes other metrics, such as minimum time, at the expense of longer computation times.

C. Continuous Control

Given the symbolic plan and the assignment of robots to sub-swarms, we present the approach to automatically synthesizing continuous controls that ensure robots to achieve the specified behavior while avoiding collisions. Control barrier functions (CBF) with nonlinear dynamics constraints are presented in [32], [33] for collision avoidance. However, in this paper we use a simple version of CBF [9], [28] for multi-robot systems assuming holonomic robots, and mitigate the effect of the robots' dynamics by increasing the safety distance D_s in the physical demonstrations.

For all symbolic transitions, we divide the control synthesis into two sub-problems: 1) determine goal points for robots in the sub-swarms, and 2) create controls that drive robots to the goals with no collision.

Region visit and coverage. To determine the goal points in any region r_j for the visit behavior, we first pick as a starting point the region centroid, and then increase the number of points around the centroid in a hexagonal lattice pattern (Fig. 1a). The growing distance is $d = \beta D_s$, where $\beta > 1$, to ensure collision-avoidance. We choose a hexagonal lattice as this pattern guarantees the same distance between any two neighbor points as shown in Fig. 1a. For region coverage, we first find line segments that connect the centroid with the vertices or the edge midpoints ($2p$ in total where p is the number of vertices), and divide these line segments evenly by N such that $(N-2)p + 1 \leq N_R \leq (N-1)p + 1$ where N_R is the number of robots. Then we choose the centroid and $N_R - 1$ points at the equidistant points on the line segments from the edges to the centroid. We check the distances d between points. If there exists a distance $d < D_s$, we replace N by $N+1$, thus creating more points to choose from, and repeat the checking process. If N becomes large enough such that distances between neighboring equidistant points are all less than D_s, the algorithm returns no solution. We note that here we assume convex regions; we can easily substitute any other coverage algorithm that would also work for non-convex regions.

To solve sub-problem 2), we run a linear assignment [30] to distribute goal points to individual robots, then design a PID controller for each single robot, and finally apply the CBF [28] to ensure that robots reach their goals with no collisions. CBFs are used to modify control inputs (velocities) such that robots are guaranteed to move with no collisions and reach their goals if there is no deadlock. For 3D regions, we choose the vertically central 2D plane to execute region visit and coverage motions although we could also choose a 3D coverage metric.

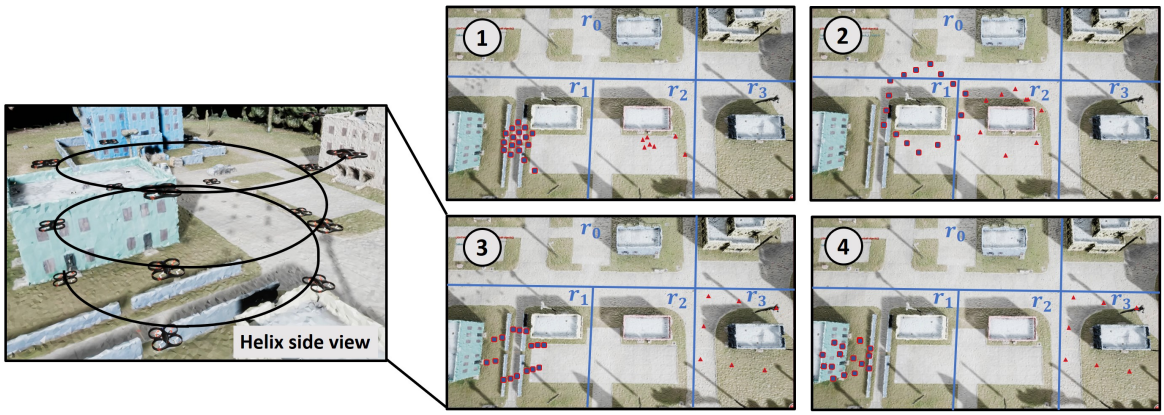


Fig. 2: Screenshots from a demonstration of 16 UAVs and 8 rovers performing a high-level formation and location-based task in the AirSim simulation. Four target buildings are located in regions r_i ($i \in \{0, ..., 3\}$) respectively. The rounded squares and triangles represent the UAV and the rover locations respectively. The numbers 1-4 show the time sequence of the screenshots. A side view is provided for better visualization of a 3D helix.

Shape formation. Given the sub-swarm size N, we solve sub-problem 1) for a 2D polygon by making p robots occupy the shape vertices, and $N - p$ distribute evenly on the edges. For sub-problem 2), we design a two-step process for obtaining continuous controls, which first drives robots to surround the centroid of the desired formation or the target, depending on the required behavior, and then assign them to different goals to complete the desired shape. In the surrounding step, we compute the velocities based on robot poses with the objective of evenly distributing robots around the polygon centroid or target. After completion of the surrounding step, we assign goals to robots and apply a PID controller with the CBF to make the robots form the shape while avoiding collisions. Once all sub-swarms finish forming the shapes, they proceed to the next state in the symbolic plan. When forming 3D shapes, we use the parametric function to generate a list of goal points given N, the number of robots, to solve sub-problem 1). For sub-problem 2), instead of the two-step control process for 2D polygons, we assign robots to goal points and drive them towards their goals to achieve the 3D shape.

Deadlocks:. In multi-robot systems, deadlocks are situations where robots cannot make progress towards their goals. In [9] we discussed different deadlock scenarios and provided perturbation and roadmap-based method to mitigate deadlocks. In general, in environments with obstacles, we cannot guarantee deadlock avoidance; however, in practice, such techniques often result in deadlocks being resolved.

VII. SIMULATIONS

We first demonstrate the control framework on 16 UAVs and 8 ground vehicles in AirSim [34]. The simulation environment is a space with four target buildings. We partition the 3D workspace into r_{i_j} ($i \in \{0, 1, 2, 3\}, j \in \{0, 1, 2\}$) such that the target buildings are in regions r_{i_0}, $i \in \{0, ..., 3\}$. The index j represents the height of the regions: $j = 0$ represents the ground-plane 2D regions, $j = 1$ represents 3D regions at 10 to 20 meters above ground, and $j = 2$ represents 20 to 30 meters above ground.

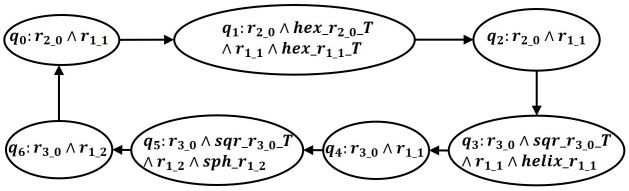


Fig. 3: Symbolic plan for the example of Section VII

A. Specifications

Initially, the UAVs and the rovers are located in regions r_{1_1} and r_{2_0} respectively. We require the swarm to accomplish three goals: repeatedly form hexagons around the targets in r_{1_1} and r_{2_0}, repeatedly form a square around the target in r_{3_0} and a helix in r_{1_1}, and repeatedly form a square around the target in r_{3_0} and a sphere in r_{1_2}. Note that we assume the UAVs only fly in the air and rovers only move on the ground, which is encoded in the region graph. Formally, the specifications are expressed in LTL and include:

- $\square\lozenge(hex\_r_{2\_0}\_T \wedge hex\_r_{1\_1}\_T \wedge \bigwedge_{k \neq 2\_0, 1\_1} \neg r_k)$,
- $\square\lozenge(sqr\_r_{3\_0}\_T \wedge helix\_r_{1\_1} \wedge \bigwedge_{k \neq 3\_0, 1\_1} \neg r_k)$,
- $\square\lozenge(sqr\_r_{3\_0}\_T \wedge sph\_r_{1\_2} \wedge \bigwedge_{k \neq 3\_0, 1\_2} \neg r_k)$.

The negation of propositions in the formulas indicates that no robots should be in those regions, thus robots should only appear in the regions that have the specified formations. These specifications simulate tasks where UAVs and ground vehicles collaboratively guard some target buildings.

B. Results

From the given specifications in Section VII-A, we use the approach in Section V to synthesize a symbolic plan shown in Fig. 3, where states q_i ($i \in \{0, ..., 6\}$) are labeled with propositions that are true in that state.

We execute the synthesized symbolic plan on 16 UAVs and 8 rovers using the approaches in Section VI. The result shows that the swarm achieves the high-level task correctly and safely (Fig. 2). For executing two cycles of the symbolic plan, the average computation time for sub-swarm assignment is $0.08s$, and the average time for computing the velocity commands during execution is $0.12s$. The time for synthesizing the symbolic plan in this demonstration is $0.97s$. The simulation was run on a desktop machine with an Intel Core i7-7700 CPU@3.6GHz and 16GB RAM.

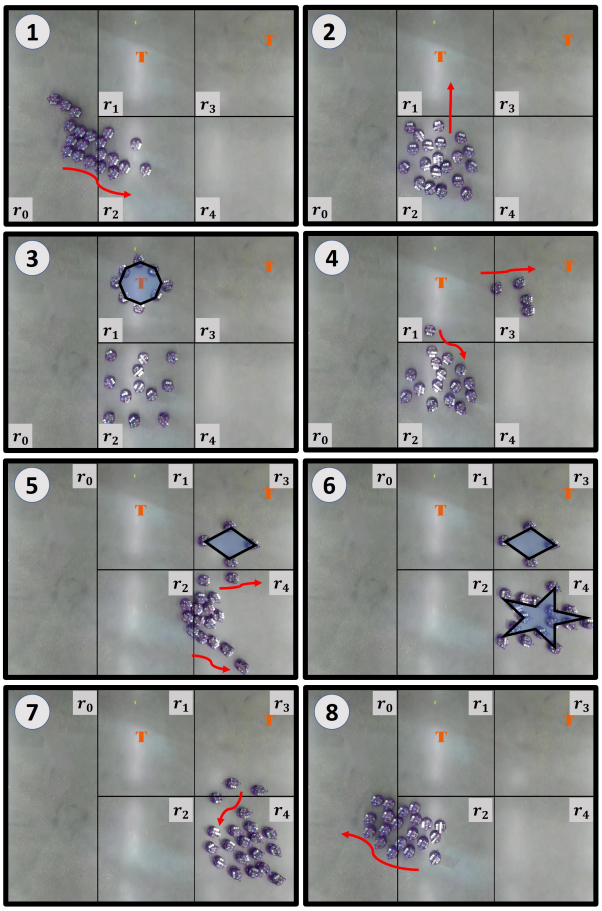


Fig. 4: A demonstration of 20 coachbots performing a high-level task. Two targets (labeled with "T") are located in r_1 and r_3. The arrows indicate the motion of nearby robots.

VIII. Physical Demonstrations

In this section we demonstrate our approach on physical robots, using 20 *Coachbot V2.0* robots [35] (customized differential-drive ground robots) in an environment partitioned into five regions. We use the formation symbols $\mathbf{F}$ (Section V-A) for creating propositions and specifications.

A. Specifications

The swarm is initially in r_0. We require the swarm to achieve three goals: repeatedly form an octagon around the target in r_1 and cover the space of r_2 at the same time while avoiding all other regions, repeatedly form a diamond in r_3 and a five-point star in r_4 at the same time while avoiding all other regions, and repeatedly gather in r_0. We add a safety specification that prevents the swarm from being in r_1 and r_4 simultaneously at any time. The specifications is formally expressed in LTL (part of the full formula is shown here):

- $\square\lozenge(oct\_r_1\_T \wedge C\_r_2 \wedge \neg r_0 \wedge \neg r_3 \wedge \neg r_4)$
- $\square\lozenge(dia\_r_3 \wedge pen\_r_4 \wedge \neg r_0 \wedge \neg r_1 \wedge \neg r_2)$
- $\square\lozenge(r_0 \wedge \neg r_1 \wedge \neg r_2 \wedge \neg r_3 \wedge \neg r_4)$
- $\square\neg(r_1 \wedge r_4)$.

B. Results

We synthesize a symbolic plan from the given specifications in Section VIII-A and show the execution in Fig. 4. The robots start from r_0, first form an octagon in r_1 and cover r_2, then form a diamond and a five-point star in r_3 and r_4 respectively, and go back to gather in r_0, as shown in Fig. 4. All the objectives and constraints are satisfied using the symbolic and continuous control synthesis. Note that the robots gather in r_2 and r_3 before entering r_4 because robots are not allowed in r_1 and r_4 at the same time and the unsafe transitions between r_1, r_2 and r_3, r_4 are ruled out during the synthesis process.

C. Discussion

Controlling a swarm of physical robots correctly and safely is challenging. We address the problems that we have encountered during the implementation on physical robots and discuss the possible solutions that might improve the physical swarm ability to perform high-level tasks.

In contrast to the simulation, where the behavior of the swarm satisfied the desired behaviors and no collisions occurred, during the physical demonstrations, collisions did happen when too many robots were in a narrow space, e.g., when forming the five-point star in r_4 in Fig. 4. This is due to the mismatch between our assumptions of full state information and idealized kinematic robots and the reality of the physical platform where we have 1) imperfect localization, 2) nonlinear robot dynamics, and 3) a tight workspace. Specifically, we assume a honolomic kinematic model for each robot to design collision-free control, and then map the robot velocity in the global frame to the wheel speeds for coachbots. However, the mapping is not fully accurate as the differential-drive robots cannot move sideways, which might violate some constraints in the CBF when robots get too close in a tight environment. Usually increasing the safety distance can improve the robustness for collision avoidance. In this work, we set the safety distance $D_s = 1.2D_r$ where D_r is the robot diameter, for the purpose of fitting 20 robots in one region to achieve the synthesized symbolic plan. Such tight safety margins, together with the nonlinear dynamics and sensing error, lead to some collisions in the physical demonstration. Our future work is to consider robot dynamics in the CBF formulation thus guaranteeing collision avoidance [32], [33], [36].

IX. Conclusion and Future Work

In this paper we propose a novel abstraction for robot swarms that allows a user to specify tasks regarding the

swarm's formations and locations. We describe a framework for automatically creating control for the swarm robots such that the swarm is guaranteed to achieve its tasks while avoiding collisions with the environment and between the robots. We demonstrate the approach both in simulation and on physical robots.

In future work we will robustify the approach with respect to constrained physical systems. Specifically, we will create continuous controls by adding robot dynamics to the CBF constraints. In addition, we will study the amount of information that individual robots need, and design communication and synchronization frameworks for swarms to achieve such high-level tasks in a distributed manner to increase the scalability of the approach.

ACKNOWLEDGMENT

This work is supported by the NSF IIS-1830471.

REFERENCES

[1] M. Saska, V. Vonásek, J. Chudoba, J. Thomas, G. Loianno, and V. Kumar, "Swarm distribution and deployment for cooperative surveillance by micro-aerial vehicles," *Journal of Intelligent & Robotic Systems*, vol. 84, no. 1-4, pp. 469–492, 2016.

[2] A. Farinelli, E. Zanotto, E. Pagello, *et al.*, "Advanced approaches for multi-robot coordination in logistic scenarios," *Robotics and Autonomous Systems*, vol. 90, pp. 34–44, 2017.

[3] J. Chen, M. Gauci, W. Li, A. Kolling, and R. Groß, "Occlusion-based cooperative transport with a swarm of miniature mobile robots," *IEEE Transactions on Robotics*, vol. 31, no. 2, pp. 307–321, 2015.

[4] J. Alonso-Mora, A. Breitenmoser, M. Rufli, R. Siegwart, and P. Beardsley, "Image and animation display with multiple mobile robots," *The International Journal of Robotics Research*, vol. 31, no. 6, pp. 753–773, 2012.

[5] L. Peng, F. Guan, L. Perneel, H. Fayyad-Kazan, and M. Timmerman, "Decentralized multi-robot formation control with communication delay and asynchronous clock," *Journal of Intelligent & Robotic Systems*, vol. 89, no. 3-4, pp. 465–484, 2018.

[6] D. V. Dimarogonas and K. H. Johansson, "On the stability of distance-based formation control," in *2008 47th IEEE Conference on Decision and Control*. IEEE, 2008, pp. 1200–1205.

[7] P. Abichandani, K. Levin, and D. Bucci, "Decentralized formation coordination of multiple quadcopters under communication constraints," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 3326–3332.

[8] S. Moarref and H. Kress-Gazit, "Automated synthesis of decentralized controllers for robot swarms from high-level temporal logic specifications," *Autonomous Robots*, pp. 1–16, 2019.

[9] J. Chen, S. Moarref, and H. Kress-Gazit, "Verifiable control of robotic swarm from high-level specifications," in *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2018, pp. 568–576.

[10] K.-K. Oh, M.-C. Park, and H.-S. Ahn, "A survey of multi-agent formation control," *Automatica*, vol. 53, pp. 424–440, 2015.

[11] A. K. Das, R. Fierro, V. Kumar, J. P. Ostrowski, J. Spletzer, and C. J. Taylor, "A vision-based formation control framework," *IEEE transactions on robotics and automation*, vol. 18, no. 5, pp. 813–825, 2002.

[12] W. Ren, R. W. Beard, and E. M. Atkins, "Information consensus in multivehicle cooperative control," *IEEE Control systems magazine*, vol. 27, no. 2, pp. 71–82, 2007.

[13] Y. Q. Chen and Z. Wang, "Formation control: a review and a new consideration," in *2005 IEEE/RSJ International conference on intelligent robots and systems*. IEEE, 2005, pp. 3181–3186.

[14] Y. Song and J. M. O'Kane, "Forming repeating patterns of mobile robots: A provably correct decentralized algorithm," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 5737–5744.

[15] W. Hönig, T. S. Kumar, H. Ma, S. Koenig, and N. Ayanian, "Formation change for robot groups in occluded environments," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 4836–4842.

[16] J. A. Preiss, W. Hönig, N. Ayanian, and G. S. Sukhatme, "Downwash-aware trajectory planning for large quadrotor teams," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 250–257.

[17] P. Pierpaoli, A. Li, M. Srinivasan, X. Cai, S. Coogan, and M. Egerstedt, "A sequential composition framework for coordinating multi-robot behaviors," *arXiv preprint arXiv:1907.07718*, 2019.

[18] P. Pierpaoli, T. Doan, J. Romberg, and M. Egerstedt, "A reinforcement learning framework for sequencing multi-robot behaviors," *arXiv preprint arXiv:1909.05731*, 2019.

[19] V. Raman and H. Kress-Gazit, "Synthesis for multi-robot controllers with interleaved motion," in *2014 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2014, pp. 4316–4321.

[20] I. Saha, R. Ramaithitima, V. Kumar, G. J. Pappas, and S. A. Seshia, "Automated composition of motion primitives for multi-robot systems from safe ltl specifications," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 1525–1532.

[21] A. F. Winfield, J. Sa, M.-C. Fernández-Gago, C. Dixon, and M. Fisher, "On formal specification of emergent behaviours in swarm robotic systems," *International journal of advanced robotic systems*, vol. 2, no. 4, p. 39, 2005.

[22] P. Schillinger, M. Bürger, and D. V. Dimarogonas, "Simultaneous task allocation and planning for temporal logic goals in heterogeneous multi-robot systems," *The international journal of robotics research*, vol. 37, no. 7, pp. 818–838, 2018.

[23] F. Faruq, D. Parker, B. Laccrda, and N. Hawes, "Simultaneous task allocation and planning under uncertainty," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 3559–3564.

[24] P. Schillinger, M. Bürger, and D. V. Dimarogonas, "Auctioning over probabilistic options for temporal logic-based multi-robot cooperation under uncertainty," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 7330–7337.

[25] E. A. Emerson, "Temporal and modal logic," in *Formal Models and Semantics*. Elsevier, 1990, pp. 995–1072.

[26] S. Moarref and H. Kress-Gazit, "Decentralized control of robotic swarms from high-level temporal logic specifications," in *2017 international symposium on multi-robot and multi-agent systems (MRS)*. IEEE, 2017, pp. 17–23.

[27] R. Bloem, B. Jobstmann, N. Piterman, A. Pnueli, and Y. Saar, "Synthesis of reactive (1) designs," *Journal of Computer and System Sciences*, vol. 78, no. 3, pp. 911–938, 2012.

[28] L. Wang, A. D. Ames, and M. Egerstedt, "Safety barrier certificates for collisions-free multirobot systems," *IEEE Transactions on Robotics*, vol. 33, no. 3, pp. 661–674, 2017.

[29] R. Ehlers and V. Raman, "Slugs: Extensible gr (1) synthesis," in *International Conference on Computer Aided Verification*. Springer, 2016, pp. 333–339.

[30] R. E. Burkard and E. Cela, "Linear assignment problems and extensions," in *Handbook of combinatorial optimization*. Springer, 1999, pp. 75–149.

[31] C. H. Papadimitriou, "On the complexity of integer programming," *Journal of the ACM (JACM)*, vol. 28, no. 4, pp. 765–768, 1981.

[32] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *2019 18th European Control Conference (ECC)*. IEEE, 2019, pp. 3420–3431.

[33] G. Yang, B. Vang, Z. Serlin, C. Belta, and R. Tron, "Sampling-based motion planning via control barrier functions," in *Proceedings of the 2019 3rd International Conference on Automation, Control and Robots*, 2019, pp. 22–29.

[34] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," in *Field and Service Robotics*, 2017. [Online]. Available: https://arxiv.org/abs/1705.05065

[35] H. Wang and M. Rubenstein, "Walk, stop, count, and swap: Decentralized multi-agent path finding with theoretical guarantees," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1119–1126, 2020.

[36] Y. Emam, P. Glotfelter, and M. Egerstedt, "Robust barrier functions for a fully autonomous, remotely accessible swarm-robotics testbed," *arXiv preprint arXiv:1909.02966*, 2019.