Distributed Progressive Formation Control with One-Way Communication for Multi-Agent Systems

Jorge Peña Queralta[2], Li Qingqing[1,2], Tuan Nguyen Gia[2], Hannu Tenhunen[3], Z. Zou[1] and Tomi Westerlund[2]
[1] School of Information Science and Technology, Fudan University, China
[2] Department of Future Technologies, University of Turku, Finland
[3] Department of Electronics, KTH Royal Institute of Technology
Emails: {jopequ, tunggi, tovewe}@utu.fi, {qingqingli16, zhuo}@fudan.edu.cn, hannu@kth.se

***Abstract*—The cooperation of multiple robots towards a common goal requires a certain spatial distribution, or formation configuration, of the agents in order to succeed. Centralized controllers that have information about the absolute or relative positions of all agents, or distributed approaches using communication to share system-wide information between agents, are able to calculate optimal individual paths. However, this reserves important computational resources as the number of agents in the system increases. We address the problem of distributed formation control with minimal communication and minimal computational power required. The algorithm introduced in this paper progressively generates a directed path graph to uniquely assign formation positions to all agents. The benefits of the proposed algorithm compared to previous include the need for one-way communication only, low computational complexity, ability to converge without any a priori assignments under certain geometric conditions and need for limited sensing information only. The algorithm can be deployed to computationally constrained devices, enabling its deployment in robots with simpler hardware architectures. The ability to converge and distributively assign positions, or roles, with only one-way communication makes this algorithm robust during deployment, at which time all agents are equivalent and anonymous. Moreover, we account for limited communication and sensing range, and agents only need to have information about other agents in their vicinity in order to make decisions. Communication is simple and allows for scalability without an impact on performance or convergence latency, with a linear dependence on the number of agents. Agents only need to broadcast their status to other neighboring agents and do not reply any message. Finally, this algorithm enables almost-arbitrary configurations. The main limitation for the choice of formation configuration is that all pairs of points forming an edge in the polygonal line defining the boundary of the convex hull must be within sensing range.**

***Index Terms*—Multi-Agent Systems; Formation Control; Distributed Control; Progressive Formation Control; Distributed Role Assignment; Multi-Agent Cooperation;**

I. Introduction

The robotics field often finds its inspiration in nature [1]. This has had a significant impact on fields such as swarm robotics [2], [3], or multi-agent and multi-robot systems [4], [5]. With part of the robotics community aiming at developing robots with capabilities that match or even outperform those of nature [6], others explore the possibilities of creating complex behaviour from relatively simple robots [7], [8]. The coordination of multiple robots can originate in robust robotic systems with the ability of complex behaviour [9]. We explore one of the basic elements of multi-robot cooperation and collaboration: formation control or pattern configuration [10].

Multi-robot and multi-agent systems have been increasingly studied by the research community over the past few decades [11]–[14]. Formation control algorithms are one of the key aspects for collaborative operation in multi-agent systems [15], [16]. Cooperation of multiple agents to perform a predefined task often involves the definition of a specific spatial distribution of the agents [17]. Hence the interest in formation control algorithms for multi-robot and multi-agent systems as the basis towards advanced collaboration.

A. Background and Related Works

Formation control algorithms can be broadly classified as a function of the variables that agents actively control when converging to the desired configuration [14]. These variables are often an absolute position in a global reference frame [18], [19]; a relative position, or displacement, with respect to other agents and measured in a local reference frame [20], [21]; or the distance or bearing to neighboring agents [22]–[25]. Equivalently to the case of distance-based formation control, other approaches propose the use of inter-agent bearing instead [25]. Alternatively, both distance and bearing have been used for non-holonomic formation control problems [26].

A formation control problem typically has two differentiated parts: (i) the assignment of positions or roles; and (ii) the definition of control laws that ensures convergence towards the assigned objectives for all units in the system [27], [28]. The second step can also be considered as path planning in a multi-robot scenario. In order for a group of agents to converge to a predefined configuration, positions in the formation have to be first assigned to individual agents via a bijective correspondence. This can be done a priori, in the case that agents are indexed and preassigned to a certain position, or decided after deploying agents to their initial positions [29]–[34]. This step is not necessary if all positions in the formation are equivalent from the point of view of the controlled variable, as is the case of flocking [35]. In this paper, we propose an algorithm that allows agents to autonomously self-assign their objective position after deployment, and use the assignment of local neighbors to perform autonomous path planning and converge to the objective configuration. We have focused on a solution that requires less a priori information given to agents because of the effect this has on scalability and deployment flexibility. In terms of scalability, the proposed algorithm requires a time complexity that is linear with respect to the number of agents to complete the position assignment of all agents. Regarding flexibility, problems can arise with a large number of agents converging towards a given formation configuration. If positions have been assigned a priori, an unfavourable initial distribution where agents are far from their objective position might significantly increase the time needed for convergence and total travelled distance in the system.

Centralized algorithms that use relative or global positioning of all agents to calculate the individual optimal paths have been traditionally used [14]. However, distributed approaches have gained increasing popularity over the past decades, enabling operation in more diverse environments. We propose a distributed formation control algorithm that requires no a priori position assignments and minimal one-way communication between agents. Positions are assigned autonomously by the agents in a progressively through a directed acyclic graph.

Different distributed approaches exist depending on the communication within the agents, and the a priori information given such as predefined position assignments. If no assignments are made, when no communication is allowed, only formation configurations where all positions have an equivalent definition in terms of their neighbors is possible. This is the case of flocks, where distance between neighbors is constant, or regular polygons, where the angle between the positions of the two nearest neighbors is also constant [35], [36]. When agents can achieve consensus through communication, then arbitrary formation shapes are possible. This can be done either by using system-wide communication to achieve consensus or via local interactions only. In the former case, auction mechanisms have been used to perform task allocation, which is equivalent to assign positions in a formation [37], [38]. Regarding the latter scenario, a very interesting approach has been proposed by Pinciroli *et al.*, in which positions are assigned progressively via local interactions between neighboring agents [29], [30]. The authors propose a solution that is easily scalable, robust, and specifically suited for the natural scenario where agents are deployed progressively and not at once. Moreover, only minimal communication is required at the time of joining the formation, and a given agent only needs to exchange information with two agents that are already part of the objective configuration. Compared to their approach, we propose a method that reduces further the amount of necessary communication. The proposed algorithm only requires one-way communication by broadcasting status information. However, this simplification limits the number of possible configurations. We prove that if the set of initial positions of deployed agents is a *convexly layered set* with a constraint below the agents' sensing range, then agents converge to the objective formation. This mainly requires that agents can be assigned to the vertices of a series of convex polygonal layers, such that they follow an inclusion relation, and all pairs of agents forming a polygon edge are within sensing range.

The progressive position assignment inherently introduces latency in the system when compared to algorithms that do not require communication between agents [32], [34]. However, it also allows us to ensure the existence of consensus during the position assignment and enables the convergence to almost arbitrary configurations. Distributed formation control algorithms that require, to some extent, communication between agents and measuring the relative positioning of neighboring agents often rely on situated communication [33], [39]. This type of communication refers to data exchanges using wireless technology and devices capable of estimating the relative position of a message sender. Two-way situated communication enables mutual localization of agents. Other approaches have been proposed for very-large-scale systems where individual positions are not assigned but instead a certain number of agents must be located in a particular area or volume in space. This idea, introduced by Bandyopadhyay *et al.*, uses probability distributions to define the formation configuration [31]. Even though agents do not communicate to achieve consensus regarding their objective positions, they need to be aware of the global spatial distribution of the swarm.

Our work has been partially inspired from previous works from Pinciroli *et al.* [29], [30]. Nevertheless, the differences are significant. First, we propose a method that requires only one-way communication. Second, we also assume that agents share a common orientation reference. This, which can be implemented via inexpensive magnetometers or other kind of inertial sensors or digital compasses, enables us to define configurations with a predefined orientation. This has an important impact when agents should move towards a given direction after or while converging towards the objective configuration. Furthermore, our algorithm does not require of a predefined pair of identified agents which are necessary for assigning the rest of roles in Pinciroli's work. Instead, we propose a method that enables autonomous self-assignment of all roles or positions in the objective pattern.

B. Contribution and Organization

The main contributions of this paper are the following: (i) the definition of an algorithm that enables distributed and autonomous position assignment in multi-robot systems with one-way communication; (ii) the introduction of a control law that ensures convergence and utilizes the information acquired during the position assignment; and (iii) the simulation and analysis of multiple scenarios and pattern configurations showing that our proposed approach is scalable and. A more realistic simulation has been implemented and analyzed using the Robot Operating System (ROS), the Gazebo simulator and the RotorS module for dynamics modelling in another paper [33]. These tools enable a more advanced modelling of unmanned aerial vehicle (UAV) dynamics and simulation and have allowed us to demonstrate the efficiency and applicability of our algorithm. Nonetheless, in this paper we focus on the theoretical foundations of our algorithm and on analyzing its performance and scalability for different objective patterns and initial distributions of an increasing number of agents.

The rest of this paper is organized as follows. Section II introduces the formation control problem and notation used within the paper. Also, a methodology for labeling the positions in a formation via the creation of a directed path graph is introduced. In Section III, we present an algorithm to progressively assign the positions through the creation of an analogous path graph when agents are deployed. Section IV, we propose a basic control law that ensures convergence, with simulation results illustrated in Section V. Section VI concludes the paper and outlines future work directions.

II. Problem Formulation

In this paper, we use the following notation. We use $[N] = \{k \in \mathbb{Z}^+ : k \leq N\}$ to denote the set of the first N positive integers. Given a set of vectors $x_1, \dots, x_N \in \mathbb{R}^n$, $\mathbf{x} = [x_1^T, \dots, x_N^T] \in \mathbb{R}^{nN}$ denotes the stacking of the vectors. Given a set of points in $\mathbb{R}^n$, the convex envelope, hull or closure is the smallest convex set containing all points. We denote by $Conv(\mathbf{q})$ the convex hull of $\mathbf{q}$ and by $\delta Conv(\mathbf{q})$ its boundary. The algorithms presented in this paper are formulated for two-dimensional formation control, and therefore we implicitly assume $n = 2$ and all points belong to $\mathbb{R}^2$.

Consider a planar formation configuration defined as a set of points, represented by a set $\mathbf{q} = [q_1, \dots, q_N] \in \mathbb{R}^{2N}$. Given a set of N agents with positions $\mathbf{p}(t) = [p_1(t), \dots, p_N(t)] \in \mathbb{R}^{2N}$, we address the problem of achieving a spatial distribution equivalent to $\mathbf{q}$ with respect to a translation.

Problem 1 (Formation Objective)**.** *Given an objective point set* $\boldsymbol{q}$ *and a set of agents represented by their positions* $\boldsymbol{p}(t)$*, we consider that the formation has been achieved at a time* $t = t'$ *if a permutation* $\sigma : [N] \to [N]$ *exists such that*

$$\begin{gathered} \|p_i(t') - p_0(t') + q_{\sigma(i)} - q_{\sigma(0)}\| < \varepsilon \\ \|\dot{p}_i(t')\| < \delta \end{gathered} \tag{1}$$

for predefined constants $\varepsilon, \delta > 0$ *that represent the maximum error allowed for positions and speed. We assume that agents are able to measure the position of any other agent in line of sight up to a predefined sensing distance* δ_s*.*

In order to solve Problem 1, we provide a methodology for uniquely assigning a position in the formation to each agent through the definition of a directed path graph. One-way communication is used to progressively assign positions in the formation. The definition of the path graph requires the following two conditions, where the sensing range of agents is taken into account.

Assumption 1. *The set of points can be divided in a set of* L *convex polygons, or layers,* $\{\boldsymbol{l}_1, \dots, \boldsymbol{l}_L\}$ *such that the polygonal areas they delimit* $\{A_{l_1}, \dots, A_{l_L}\}$*, defined as compact sets in* $\mathbb{R}^2$*, follow a strict inclusion relation* $A_{l_1} \supset A_{l_2} \supset \cdots \supset A_{l_L}$*. Any two consecutive points in a given layer are separated by a distance smaller than the agents' maximum sensing distance,* $\|l_{ki} - l_{k(i+1 \mod L_k)}\| < \delta_s$*. We denote by* L_k *the number of points if the* k*-th layer, and define* $\boldsymbol{l}_k = \{l_{k1}, \dots, l_{kL_k}\}$ *in an order such that any pair of consecutive points* $(l_{ki}, l_{k(i+1 \mod L_k)})$ *defines an edge of the convex polygon.*

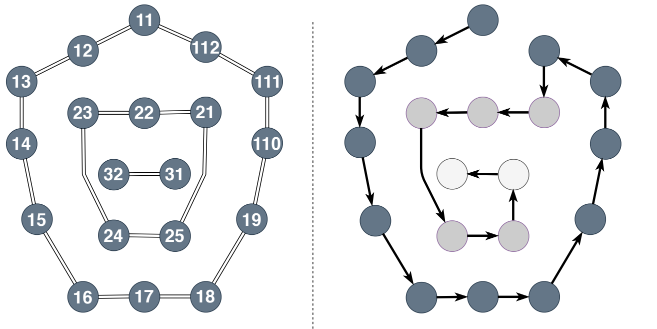


Fig. 1. Representation of convex layers in a 2D point set and SDPG generation. Non straight connections between nodes are merely illustrative.

Assumption 2. *For each point* l_{ki}*,* $1 \leq k < L$*, there exists* $1 \leq j \leq L_{k+1}$ *such that* $\|l_{ki} - l_{(k+1)j}\| < \delta_s$ *and* $\|l_{ki} - l_{(k+1)(j+1 \mod L_{k+1})}\| < \delta_s$*. Equivalently, we assume that for each point* l_{ki}*,* $1 < k \leq L$*, there exists* $1 \leq j \leq L_{k-1}$ *such that* $\|l_{ki} - l_{(k-1)j}\| < \delta_s$ *and* $\|l_{ki} - l_{(k-1)(j+1 \mod L_{k-1})}\| < \delta_s$*. This essentially implies that, for each point, there are at least two points in the previous and next layer within sensing range, unless the point is in the first or last layer. If there is a single point in the innermost layer, then at least three other points must be within sensing range. Intermediate layers cannot have one or two points only, as any three points form a triangle and all triangles are convex polygons.*

The parameter $\delta_s > 0$ is a lower limit of the agents' sensing range. This value is different in each application scenario and should be chosen lower than the real range to ensure agents are able to sense their neighbors consistently through time.

The first layer $\mathbf{l}_1$ is the boundary of the convex hull or convex envelope of the point set $\mathbf{l}$. Therefore, we can easily calculate the points in any layer using the following relations:

$$\mathbf{l}_1 = \delta Conv(\mathbf{q}), \;\; \mathbf{l}_k = \delta Conv\left(\mathbf{q} \setminus \bigcup_{1 \leq k' < k} \mathbf{l}_{k'}\right) \; \forall 1 < k \leq L \tag{2}$$

The convex hull of a finite point set in $\mathbb{R}^2$, or $\mathbb{R}^3$, can be calculated in $\mathcal{O}(n \log h)$ time with Chan's algorithm, where h is the output size, i.e., the number of points defining the convex hull [40].

Definition 1 (Convexly layered set)**.** *Given a set of points in the plane, represented by a stacked vector* $\boldsymbol{q} = [q_1^T, \dots, q_N^T] \in \mathbb{R}^{2N}$*, and a distance* $\delta_s \in \mathbb{R}$*,* $\delta_s > 0$*, we define the pair* $(\boldsymbol{q}, \delta_s)$ *as a convexly layered set with constraint* δ_s *if Assumption 1 and Assumption 2 hold.*

We should note than given any set $\mathbf{q}$, there always exists a value δ_s large enough such that $(\mathbf{q}, \delta_s)$ is a convexly layered set with constraint δ_s. This can be easily proven from the definition of $\mathbf{l}_k$ in Eq. 2, as the layers can be calculated first and then the minimum sensing range is obtained from the conditions in Definition 1. Figure 1 (a) shows three convex layers for a set of 19 points. The first digit of each node's identifier references the layer number $\{1, 2, 3\}$ and the rest of digits are unique for each layer, with layer sizes $\{12, 5, 2\}$. This identifiers have been merely chosen for illustration purposes; in a real application, a sequence of increasing natural numbers can be used as agents are given a priori information about the objective configuration. This information can then include the number of agents in each layer.

We can now define a directed path graph on the point layer that uniquely assigns identifiers to each point progressively. The directed path graph is generated as follows. First, a node is chosen as the graph root. Any edge node can be chosen at this point. A node is an edge node if it is a point $q_i = (q_{i_x}, q_{i_y})$ that belongs to the convex hull and there exist constants $m, n \geq 0$, defining a line $f(x) = mx+n$, and constants $s_1, s_2 \in \{-1, 1\}$, such that the edge node q_i belongs to the line, all other points in the set belong to only one of the two half-planes defined by the line, and other points that belong to the line belong to only one of the two half-lines in which q_i divides the line. Mathematically, this means that

$$\begin{gathered} q_{i_y} = f(q_{i_x}) \\ q_{j_y} \leq s_1 f(q_{j_x}) \ \forall j \in [N],\, j \neq i \\ \forall j \neq i \mid q_{j_y} = f(q_{j_x}) \Longrightarrow q_{j_x} < s_2\, q_{i_x} \end{gathered} \tag{3}$$

Second, a clockwise or counterclockwise *assignment direction* is chosen. This direction is used in all layers to define the order in which identifiers are assigned to points in the set. Finally, starting at the graph root, the next node in the graph is one of the two points that share an edge with the root in the outer layer, chosen accordingly with the assignment direction. The assignment continues iteratively in the same direction through the outer layer until all points in the layer have been assigned an identifier. When the last point in the first layer has been identified, the assignment continues to inner layers by choosing the closest point in the next layer and repeating the same process as in the first layer. This process continues until all points in the set have been assigned a position.

Definition 2 (SDPG from a convexly layered set)**.** *Given a convexly layered set ($\boldsymbol{q}, \delta_s$), we define a Spiral Directed Path Graph (SDPG) following the next steps:*

1. Choose an edge node as the graph root, and constants m, n, s_1, s_2 that uniquely define the point within the set $\boldsymbol{q}$. The constants m, s_1, s_2 will be used by agents to self decide whether they are the root node after deployment. The value n is not necessary because we consider any configuration equivalent with respect to a translation. Therefore, any line from the set of parallel lines defined by m can be used to uniquely identify the root, together with constants s_1, s_2.

2. Choose an assignment direction, clockwise or counter-clockwise.

3. Identify the nodes following the assignment direction, starting from the root node, and following the previous indications when all nodes in a layer have been identified.

A path graph is a tree where only two nodes have degree one, and all other nodes have degree two. Because the SDPG is a directed graph, the root and terminal nodes have degree one. All nodes except the root have a parent node, and all nodes except the terminal have a child node.

To solve Problem 1, we first assign a unique identifier to the positions in a given formation configuration by generating an SDPG. Then agents perform a progressive self-assignment of positions until an equivalent SDPG is generated from the moment they are deployed. This analog process is defined in Section III. At this point, agents actively control their position with respect to the nodes they are connected with in the SDPG according to the displacement defined in the objective formation configuration. The desired pattern is achieved when all agents error are below a predefined threshold. However, agents are not aware of the global error due to the lack of communication, and local errors are used instead to estimate the global system state.

Figure 1 (b) shows the SDPG generated from a given set of points as the desired formation configuration. The edge node is given by constants $m = 0, n = q_{1_y}$, defining a line $f(x) = q_{1_y}$, where q_1 is the point with identifier *11*. The half-plane where all other points lay is defined by $s_1 = 1$. The value s_2 is not relevant in this case because there is no other point in the line, i.e., $q_{j_y} \neq q_{1_y} \ \forall j \neq 1$. However, the origin choice might affect the graph generation in the agent set when assigning positions in the corresponding SDPG. Therefore, even if not significant, a value for $s_2 \in \{\pm 1\}$ must be chosen. The assignment direction is counter-clockwise in this case. We use the term *spiral* to refer to the directed path graph because of the decreasing distance to the center of mass when considering different layers, even though this does not necessarily happen within a certain layer.

III. Progressive Position Assignment Algorithm

In this section, we describe a position assignment algorithm that only requires one-way, minimal communication between agents. One-way communication is used to enable the assignment of positions in a random spatial distribution of agents.

Suppose a set of N agents with positions represented by $\mathbf{p}(t) = [p_1^T(t), \ldots, p_N^T(t)] \in \mathbb{R}^{2N}$ and maximum sensing range δ_s is given. An objective formation configuration is defined by a point set $\mathbf{q} = [q_1^T, \ldots, q_N^T] \in \mathbb{R}^{2N}$. We assume that the set $\mathbf{p}(0)$, together with the agent sensing range δ_s, is a convexly layered set $\{\mathbf{p}(0), \delta_s\}$.

During the position assignment process, agents can have one of three different states: **(1)** *Assigned*: agents that know their position and identifier; **(2)** *Known layer:* agents with unassigned position but that are aware of their layer; and **(3)** *Unknown layer*: agents with unknown layer.

All agents in the outer layer are in the *known layer* state immediately after deployment, and all other agents in the *unknown layer* state. Agents continuously transmit their state through a broadcast signal. Situated communication can be used to both transmit the signal and measure the position of other agents [39]. The broadcasted signal is used by neighboring agents to track the position of the transmitting agent while, at the same time, make decisions with respect to their objective position. If an agent is in the *assigned* state, then the broadcast signal includes information about its objective position. We assume that if agent i is able to receive agent j's signal at a certain time, then agent j is able to receive agent's i signal at the same time.

The progressive position assignment (PPA) then proceeds as follows: **(1)** With the same constants m, s_1, s_2 used in the definition of the SDPG from the objective configuration, each agent in position $p_i = (p_{i_x}, p_{i_y})$ checks if the three conditions in Equation 3 hold for $n = p_{i_y} - p_{i_x} m$. with $f(x) = mx + n$. If an agent i fulfills all three conditions, with $f(x) = m(x - p_{i_x}) + p_{i_y}$, then the agent is the root node and it changes its state to *assigned*. **(2)** The root agent starts broadcasting signal. Then, there are two agents next to it in the outer layer which self-decide whether they are or not the next node in the graph based on the assignment direction. In the case of a positive decision, then the next node starts broadcasting signal as well. After all nodes in the outer layer have a position assigned, then nodes in the next layer can automatically change their state to *known layer*. **(3)** Finally, each node starts to actively control its position immediately after it is in the *assigned state*.

Theorem 1 (Uniqueness of the PPA). *Let $\boldsymbol{p} \in \mathbb{R}^{2N}$ represent the spatial distribution of a set of N agents with sensing range $\delta_s > 0$, and capable of one-way communication by continuously broadcasting a signal over time. If $\{\boldsymbol{p}, \delta_s\}$ is a convexly layered set, then the position assignment process defined by the previous three steps uniquely assigns a position to each agent.*

Proof. Positions are self-assigned in an autonomous way by agents, based on the positions and states of neighboring agents. Therefore, in order to prove the theorem, we first need to demonstrate that a single agent will self-assign itself the role of graph root, and that at every step the appropriate agent, and only that agent, will self-assign the next node in the SDPG.

Suppose there are two agents such that, immediately after deployment, claim that they are the root agent. Let p_i, p_j be the two positions of such agents. Without any loss of generality, we can assume $j > i$. With constants m, s_1, each of the points defines a half-plane. If both half-planes are the same, then p_i and p_j lay on the same line. If agents i and j are within sensing range, then s_2 has different value for each of them and this is a contradiction. If they are not able to sense each other, then there exists another agent in the outer layer with position p_k, $i < k < j$, such that it belongs to the half-plane defined by the p_i and p_j. If it belongs to the boundary, then the same problem arises with the sign of s_2 (and new points can be equally generated if k is not within sensing range of both i and j). If it belong to the interior of the half-plane, then the segment $\overline{p_i p_j}$ is outside of the convex outer layer, and this contradicts the definition of convex polygon and breaks Asumption 1.

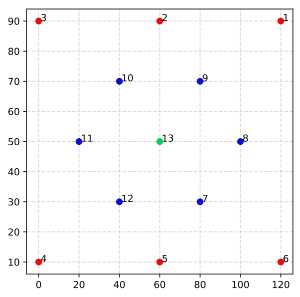
If the half-planes are not the same, we can assume without any loss of generality, and based on the value of s_2, that the half-plane generated by p_i is contained within the half-plane generated by p_j. This necessarily means that they do not sense each other. Let k be now the nearest edge to i in the outer layer, such that it is between i and j. Then p_k necessarily belongs to the half-plane defined by p_i based on Eq. 3. However, then $\overline{p_i p_j}$, which lies completely outside of that same half-plane, is not within the outer layer and a contradiction arises.

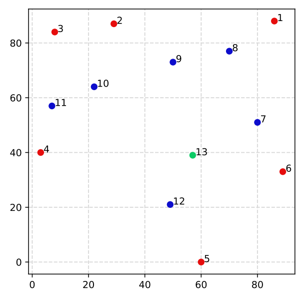
The above implies that all agents in the external layer can be uniquely identified following the SDPG generation, without two agents claiming to have the same objective position. Now we just need to prove that a single agent in the next layer will claim to have the appropriate objective position. The problem can be reduced to prove that a single agent will claim to be the closest to the last agent identified in the outer layer. Let p_i be the position of the last agent assigned to the outer layer, and assume that two agents in position p_j, p_k claim to be the closest. Both p_j and p_k know that they are in the next layer in the same manner than initially all agents in the outer layer are aware of that, since all these have been already identified. If p_j and p_k are within sensing range, then they can both masure the other's distance to p_i, and only one will claim to be the closest. In case of equivalent distance, the decision is made based on the assignment direction. Therefore, for agents j and k to make the claim, they cannot be within sensing range of each other. Based on Assumptions 1 and 2, this means there is at least one other agent h in the same layer in between. However, because h is farther from i, it is outside the triangular area formed by agents i, j, k. This implies that the segment $\overline{p_j p_k}$ lays outside any convex poligonal area with vertices including p_j, p_k, p_h, that also leaves p_i outside. This contradicts the definition of convex polygon, and we started assuming that p_j, p_k, p_h. $\square$

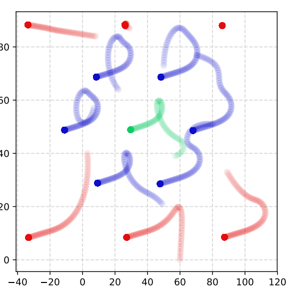
As previously noted, all agents except the root have a parent node, and the root can be uniquely identified. This allows us to propose a control law for each agent based on a basic leader-follower formation, where the parent is the leader and the agent itself is the follower. We assume that the parent and child nodes of any agent, which are within sensing range at time $t = 0$, stay within sensing range at any time $t' > 0$. If a new objective configuration is required, then agents loose their assignments and the process is restarted.

IV. Control Input

We propose a control law that enables agents to converge to the objective configuration while avoiding inter-agent collisions. A natural solution is for agents to actively control the displacement with respect to their parent node in the path graph. However, this has the disadvantage of increasing the system error with small drifts in the displacement of each parent-child pair. We propose a methodology for reducing the system error after individual agent errors are below a certain threshold, but its analysis is not wihin the scope of this paper. The main contribution of this paper is on the progressive assignment algorithm that uniquely generates a SDPG given an spatial distribution of agents, and not on the control input that enables convergence. Multiple leader-follower formation control solutions exist in the literature and


(a) Objective formation F1


(b) Initial positions


(c) Agent paths

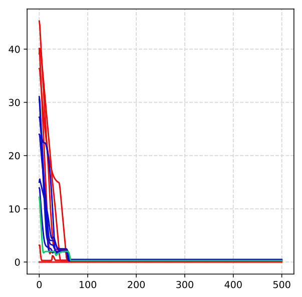
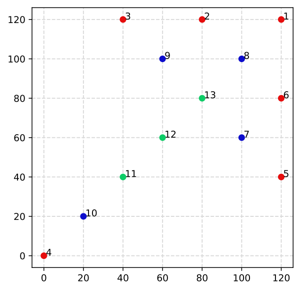
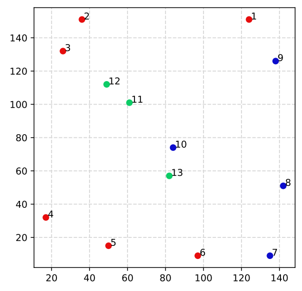
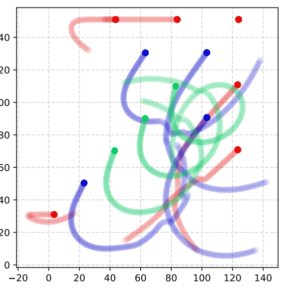

(d) Agent errors

Fig. 2. Illustration of 13 agents in random initial positions converging towards the formation configuration F1 defined in (a).


(a) Objective formation F2


(b) Initial positions


(c) Agent paths

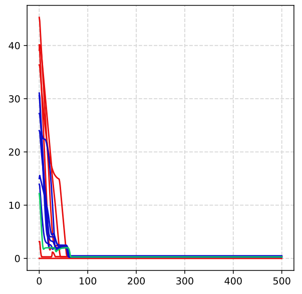

(d) F2 errors

Fig. 3. Illustration of 13 agents in random initial positions converging towards the formation configuration F1 defined in (a).

can be easily adapted to work with our proposed progressive position assignment algorithm [14], [26], [41], [42].

Let $\mathbf{q}$ be a set of N points representing the objective formation, and $\mathbf{p}(t)$ the position of N agents actively trying to converge to a spatial configuration equivalent to $\mathbf{q}$ with respect to a translation $\vec{d_p}(t)$. Without any loss of generality, we assume $\vec{d_p} = q_0 - p_0(t)$ and agents are indexed such that $p_i(t) \xrightarrow[t\to\infty]{} q_i + \vec{d_p}$. Let $p_j^i(t)$ be the position of agent j measured by agent i in its local reference frame, $p_j^i(t) = p_j(t) - p_i(t)$. Then, agent i's objective is fulfilled at time t', from the point of view of a leader-follower formation, if $p_{i-1}^i(t') = q_{i-1} - q_i$, for $i > 1$. Let $\mathcal{N}_i$ be the set of agents that agent i is able to sense. Necessarily, this set contains at least the parent node, $p_{i-1} \in \mathcal{N}_i \; \forall i > 0$.

In order to test the feasibility of this method, we propose a simple control law based on a single integrator model, $\dot{p}_i = u_i$, where u_i is the control input for agent i. For agent $i = 0$, a trivial solution is to have $u_i = 0 \, forall t > 0$. For all other agents, a simple follower equation can be written generically as $u_i = \varphi(p_{i-1}^i(t) - q_{i-1} + q_i)$ such that is ensures asymptotical convergence towards the objective displacement. This function should minimize a cost function such as

$$J_i(t) = \gamma_1 \|p_{i-1}^i(t) - q_{i-1} + q_i\|^2 + \gamma_2 \sum_{j<i,\, j\in\mathcal{N}_i} \|p_j^i(t) - q_j + q_i\|^2 + \gamma_3 \|\dot{p}_i\|^2 \tag{4}$$

In this paper, we use $\gamma_2 = 0$. However, we introduce this term as a proposal to reduce the overall system error when the parent-child displacement is already within a predefined limit. We introduce the constraint $j < i$ because the smaller the index, the smaller the agent position error due to less accumulated drift. Moreover, agents with smaller index converge faster as they are closer to the static root node from the point of view of the directed path graph. Therefore, agents can take as reference any other agents that they can sense, and that precede them in the SDPG.

For collision avoidance, we use a potential such as [43]:

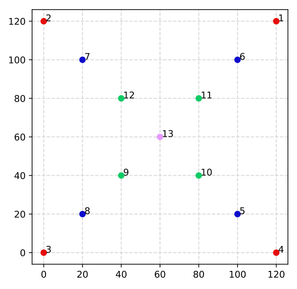
$$V_{ij}(t) = \begin{cases} \left(\min\left\{0, \dfrac{\|p_j^i(t)\|^2 - R^2}{\|p_j^i(t)\|^2 - r^2}\right\}\right)^2 & \|p_j^i(t)\| > r \\ \infty & \|p_j^i(t)\| < r \end{cases} \tag{5}$$

where R, r represent the *warning* and *danger* distance, respectively. These constants are defined in a way such that an agent actively tries to avoid another agent when the distance that separates them is smaller than the warning distance, and it must never be below or equals to the danger distance.

During the simulations we assume that, for any given agent other than the root, its parent agent is always within sensing range. However, this is unrealistic as even line of sight could be lost when another agent passes by through both. To solve this, since parent agents are also able to measure the position of their child agent, we propose to reduce the parent speed closer to 0 as the distance between parent and child increases towards the sensing range or a predetermined limit.

V. Experimental Results

In order to test the feasibility and effectiveness of the proposed algorithm, we have run a set of simulations to analyze


(a) Objective formation F3

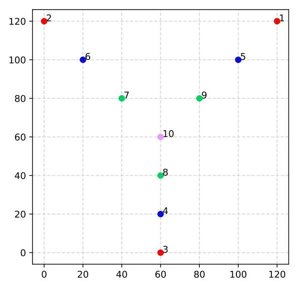

(b) Objective formation F4

Fig. 4. Objective distributions F3 anf F4 utlized in the experimental simulations.

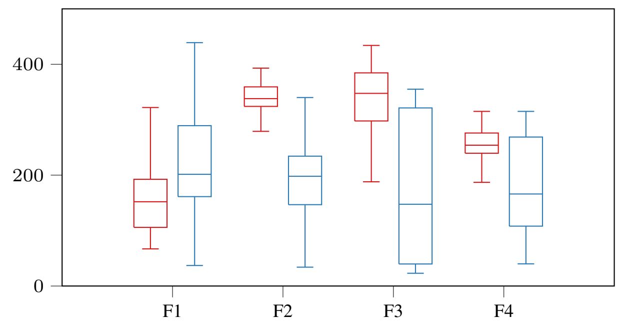
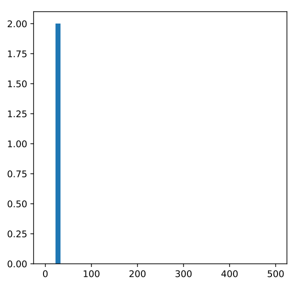

(c)

Fig. 5. Boxplot illustrating the number of iterations (vertical axis) needed to converge to different objective patterns. In each simulation, the initial distribution of agents is either random (red) or has been generated adding noise to the objective distribution (blue). In the horizontal axis, each of the objective formation configurations introduced in Figures 2, 3 and 4.


(a) Collision avoidance in F1

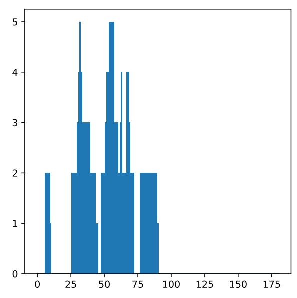
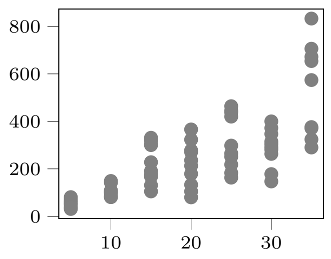

(a) Collision avoidance in F2

Fig. 6. Collision avoidance triggers


(a) Random initial distributions

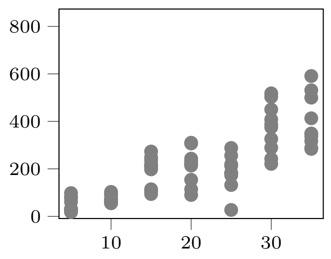

(b) Noise in initial distribution

Fig. 7. Number of iterations converging to a random configuration

its performance under favorable and unfavorable conditions. Figures 2 and 3 show two example configurations, F1 and F2, with 13 agents each. Both configurations can be divided in a set of three convex layers. In both cases, random initial distributions have been used. These two examples serve as an initial illustration of the feasibility of our proposed algorithm. Figure 6 shows the number of agents that have at least one neighboring agent closer than the warning distance.

Common to all simulation results presented in this paper are the following parameters. When distributing agents randomly, a minimum distance of 15 is kept between agents. The inter-agent warning distance to avoid collisions is set to 8, and the danger distance to 4. The maximum speed of agents is 1. The assignment direction is counterclockwise, and the root node is chosen with constants $m = 0, s_1 = 1, s_2 = 1$. In order to avoid that agents lose sight of their parent node, we have also successfully tested a modification of the collision avoidance scheme in which roles are assigned a priority based on their distance to the root node in the graph. When the collision avoidance potential is activated, only the agent with lower priority actively avoids the collision. Moreover, if a parent node notices that its follower is getting to a distance near the maximum sensing range, it can reduce its speed until the distance is reduced.

In Figures 5 and 7, we analyze how the objective configuration and the initial distribution of agents affect the performance of the algorithm in terms of time to convergence. Figure 5 (c) summarizes the number of iterations needed to converge to the objective formations F1, F2, F3 and F4 over 800 simulations, 100 for each configuration and type of initial distribution. In red is shown the results of simulations where the initial distribution of agents is random over an area similar to the objective one. Formation F1 clearly requires less time. This is due to the sparse distribution of agents in space. Formations F2, F3 and F4 show similar complexity, with F4 requiring fewer iterations presumably due to the lower number of agents. The box graphs in blue show the number of iterations needed to converge in the case in which the initial positions of agents are calculated by adding a random drift to the objective positions. In this case, the space sparsity of the formation does not play such a significant role, as the initial distribution is similar. Therefore, very similar complexity is shown by the different formations.

In Figure 7, we show the result of running 280 simulations for 2 different types of initial distributions and 7 different number of agents. The objective configuration is a random distribution of N agents over an area with side length $L_{grid} = 50\sqrt{2}\sqrt{N/5}$. In Figure 7 (a), the initial distribution of agents is also random over an area of the same size. However, in (b), agents are individually deployed nearby the objective distribution, in areas of side length 30. This resembles the idea of adding noise to the objective system. We can see that the complexity of the system rapidly grows in the case of a random distribution, while the number of iterations increases slower when the shapes are more similar. In a real scenario, a person deploying agents in an objective scenario probably has an idea of the final spatial distribution that agents will converge to. Therefore, it is natural to expect that the original distribution is not totally random and is in some way correlated to the objective distribution.

VI. Conclusion and Future Work

We have presented a distributed progressive formation control algorithm that enables a wide range of formation configurations. Any formation configuration is possible if the sensing range enables the definition of a convexly layered set. Compared to a previous progressive formation control algorithm proposed by Pinciroli *et al.*, our approach requires only one-way communication, and all agents are equivalent and anonymous upon deployment. Therefore, the proposed methodology does not require any agent to have a preassigned role or objective position. Furthermore, we assume that agents share a common orientation reference. This enables convergence to a formation configuration with respect to a translation, keeping the desired orientation. Finally, we propose a control law based on a leader-follower scheme that requires only the same information utilized during the role assignment process. The algorithm is lightweight and can be implemented in resource constrained devices.

Future work will include an implementation of the proposed algorithm in terrestrial and aerial vehicles, introducing the dynamics of different agents and adapting control inputs accordingly. The algorithms introduced in this paper will be also extended to 3D formation control.

Acknowledgment

This work has been supported by NFSC grant No. 61876039, and the Shanghai Platform for Neuromorphic and AI Chip (NeuHeilium).

References

[1] G. A. Bekey. *Autonomous robots: from biological inspiration to implementation and control*. MIT press, 2005.
[2] E. Sahin. Swarm robotics: From sources of inspiration to domains of application. In *Int. workshop on swarm robotics*. Springer, 2004.
[3] Y. Tan, *et al.* Research advance in swarm robotics. *Defence Technology*, 9(1):18–39, 2013.
[4] C. W. Reynolds. *Flocks, herds and schools: A distributed behavioral model*, volume 21. ACM, 1987.
[5] R. Olfati-Saber. Flocking for multi-agent dynamic systems: Algorithms and theory. Technical report, CALIFORNIA INST OF TECH PASADENA CONTROL AND DYNAMICAL SYSTEMS, 2004.
[6] M. Hutter *et al.* Anymal-a highly mobile and dynamic quadrupedal robot. In *IEEE/RSJ IROS*. IEEE, 2016.
[7] G. Beni. From swarm intelligence to swarm robotics. In *International Workshop on Swarm Robotics*, pages 1–9. Springer, 2004.
[8] M. Brambilla *et al.* Swarm robotics: a review from the swarm engineering perspective. *Swarm Intelligence*, 7(1):1–41, 2013.
[9] J. Kennedy. Swarm intelligence. In *Handbook of nature-inspired and innovative computing*, pages 187–219. Springer, 2006.
[10] M. Egerstedt *et al.* Formation constrained multi-agent control. *IEEE Transactions on Robotics and Automation*, 17(6):947–951, Dec 2001.
[11] Y. Q. Chen and Z Wang. Formation control: a review and a new consideration. In *IEEE/RSJ IROS*, 2005.
[12] J. M . Hendrickx*et al.* Directed graphs for the analysis of rigidity and persistence in autonomous agent systems. *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal*, 2007.
[13] B. D. O. Anderson *et al.* Rigid graph control architectures for autonomous formations. *IEEE Control Systems Magazine*, 2008.
[14] K. K. Oh *et al.* A survey of multi-agent formation control. *Automatica*, 53, 10 2014.
[15] Reza Olfati-Saber, J Alex Fax, and Richard M Murray. Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE*, 95(1):215–233, 2007.
[16] Wei Ren and Yongcan Cao. *Distributed coordination of multi-agent networks: emergent problems, models, and issues*. Springer Science & Business Media, 2010.
[17] T. Balch *et al.* Social potentials for scalable multi-robot formations. In *ICRA*. IEEE, 2000.
[18] W. Ren and E. Atkins. Distributed multi-vehicle coordinated control via local information exchange. *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal*, 17(10-11):1002–1033, 2007.
[19] T. Van den Broek *et al.* Formation control of unicycle mobile robots: a virtual structure approach. In *48h IEEE CDC*. IEEE, 2009.
[20] Guanghui Wen *et al.* Distributed consensus of multi-agent systems with general linear node dynamics and intermittent communications. *International Journal of Robust and Nonlinear Control*, 2014.
[21] S Coogan and M. Arcak. Scaling the size of a formation using relative position feedback. *Automatica*, 48(10):2677–2685, 2012.
[22] Myoung-Chul Park *et al.* Control of undirected four-agent formations in 3-dimensional space. In *52nd IEEE CDC*. IEEE, 2013.
[23] R. Olfati-Saber and R. M. Murray. Distributed cooperative control of multiple vehicle formations using structural potential functions. *IFAC Proceedings Volumes*, 35, 2002. 15th IFAC World Congress.
[24] K. K. Oh *et al.* Formation control of mobile agents based on inter-agent distance dynamics. *Automatica*, 47(10), 2011.
[25] N. Shiell *et al.* A bearing-only pattern formation algorithm for swarm robotics. In *Swarm Intelligence*. Springer International Publishing, 2016.
[26] S. A. Barogh and H. Werner. Cascaded formation control using angle and distance between agents with orientation control (part 1 and part 2). In *UKACC 11th International Conference on Control*, Aug 2016.
[27] N. Michael *et al.* Distributed multi-robot task assignment and formation control. In *IEEE ICRA*. IEEE, 2008.
[28] M. Ji *et al.* Role-assignment in multi-agent coordination. *nternation-alJournal of Assistive Robotics and Mechatronics*, 7(1):32–40, 2006.
[29] C. Pinciroli *et al.* Decentralized progressive shape formation with robot swarms. In *DARS*, pages 433–445. Springer, 2018.
[30] Guannan Li *et al.* Decentralized progressive shape formation with robot swarms. *Autonomous Robots*, Oct 2018.
[31] S. Bandyopadhyay *et al.* Probabilistic and distributed control of a large-scale swarm of autonomous agents. *IEEE Transactions on Robotics*, 33(5):1103–1123, Oct 2017.
[32] J. Peña Queralta *et al.* Communication-free and index-free distributed formation control algorithm for multi-robot systems. *Procedia Computer Science*, 2019.
[33] C. McCord *et al.* Distributed Progressive Formation Control for Multi-Agent Systems: 2D and 3D deployment of UAVs in ROS/Gazebo with RotorS. In *European Conference on Mobile Robots (ECMR)*, 2019.
[34] J. Peña Queralta *et al.* Towards designing index-free formation control. Master's thesis, University of Turku, Finland, and Fudan University, China, 2018.
[35] G. Vásárhelyi *et al.* Outdoor flocking and formation flight with autonomous aerial robots. In *IEEE/RSJ IROS*, 2014.
[36] S. L. Smith *et al.* Stabilizing a multi-agent system to an equilateral polygon formation. In *17th MTNS*, 2006.
[37] M. Hoeing *et al.* Auction-based multi-robot task allocation in comstar. In *Proceedings of the 6th AAMAS*, pages 280:1–280:8, 2007.
[38] D. Morgan *et al.* Swarm assignment and trajectory optimization using variable-swarm, distributed auction assignment and sequential convex programming. *The International Journal of Robotics Research*, 35(10):1261–1285, 2016.
[39] Kasper Støy. Using situated communication in distributed autonomous mobile robotics. In *Proceedings of the Seventh Scandinavian Conference on Artificial Intelligence*, SCAI '01, pages 44–52. IOS Press, 2001.
[40] T. M. Chan. Optimal output-sensitive convex hull algorithms in two and three dimensions. *Discrete & Computational Geometry*, 1996.
[41] J. Ghommam *et al.* Cascade design for formation control of non-holonomic systems in chained form. *Journal of the Franklin Institute*, 348(6):973 – 998, 2011.
[42] L. Consolini *et al.* Leader–follower formation control of nonholonomic mobile robots with input constraints. *Automatica*, 44, 2008.
[43] S. Ahmadi Barogh *et al.* Formation control of non-holonomic agents with collision avoidance. In *American Control Conference*, 2015.