Decentralized Progressive Shape Formation with Robot Swarms

Guannan Li · David St-Onge · Carlo Pinciroli · Andrea Gasparri · Emanuele Garone · Giovanni Beltrame

Abstract We address the problem of progressively deploying a set of robots to a formation defined as a point cloud, in a decentralized manner. To achieve this, we present an algorithm that transforms a given point cloud into an acyclic directed graph. This graph is used by the control law to allow a swarm of robots to progressively form the target shape based only on local decisions. This means that free robots (i.e., not yet part of the formation) find their location based on the perceived location of the robots already in the formation. We prove that for a 2D shape it is sufficient for a free robot to compute its distance from two robots in the formation to achieve this objective. We validate our method using physics-based simulations and robotic experiments, showing consistent convergence and minimal formation placement error.

Keywords Swarm robotics · Pattern formation · Progressive deployment · Buzz

Guannan Li
State Key Laboratory of Robotics, Shenyang Institute of Automation, CAS, No. 114 Nanta Street, Shenhe District, Shenyang, China

University of Chinese Academy of Sciences 19 A Yuquan Rd, Shijingshan District, Beijing, China
E-mail: liguannan@sia.cn

Carlo Pinciroli
Worcester Polytechnic Institute, 100 Institute Rd, Worcester, MA 01609, USA.
E-mail: cpinciroli@wpi.edu

Andrea Gasparri
Università Roma Tre, Via della Vasca Navale 79, Roma, 00146, Italy.
E-mail: gasparri@dia.uniroma3.it

Emanuele Garone
Université Libre de Bruxelles, 50 Avenue F. D. Roosevelt, Bruxelles, 1050, Belgium.
E-mail: egarone@ulb.ac.be

Guannan Li, David St-Onge, and Giovanni Beltrame
Polytechnique Montréal, 2500 chemin de Polytechnique, Montréal, QC, Canada.
Tel.: +1 (514) 340 4711 ext 2370
E-mail: david.st-onge,giovanni.beltrame,guannan.li@polymtl.ca

1 Introduction

Multi-robot systems are becoming pervasive in society: whether it is self-driving cars or swarms of quadcopters, the idea of using large number of communicating robots to accomplish tasks is becoming more common. Coordinating large teams of robots can be accomplished in many ways, for example with robot swarms using decentralized techniques (Brambilla et al, 2013). In general, most approaches assume that all of the robots involved in a particular task, or executing a particular algorithm, are readily available at the beginning of its execution. In practice, however, this can be rather difficult to achieve for swarms containing more than a dozen robots: economic or technology constraints can severely limit the number of robots that are available for deployment at a given time. For instance, the current state-of-the-art in terms of planetary exploration, i.e., the current effort at understanding Mars, required hardware to be sent in waves, and each new wave is built on the capability of the previous. Currently, the rovers use the communication infrastructure provided by satellites of previous missions.

We believe that future mission involving complex, heterogeneous swarms will be deployed progressively, in phases. A first phase of expensive, complex robots can build or provide infrastructure that paves the way for simpler, cheaper, and more mission-specific robots. For these progressive deployments, we need a new class

of algorithms that focuses on swarms that grow over time (Beal, 2011). As a first attempt in this area, in this paper we propose a decentralized algorithm that can progressively form an arbitrary shape using a robot swarm. Shape formation is an important and well-studied application of robot swarms, with applications including environmental modeling of large areas (such as the optimal placement of sensors) or the creation and maintenance of mobile ad-hoc networks. We can envision robots beginning their tasks before the structure is completed, and still providing a useful service. In this context, the time needed to form a complete shape is bound to the frequency at which robots can be added to the system, which is possibly not completely controllable by the system designer. The time needed to correctly place the next robot and the accuracy of its placement are then far more important metrics.

Centralized methods allow robots to achieve any kind of shape. However, they might not be applicable in communication-challenged scenarios, and as the number of robots involved increases, these methods become less and less desirable. In addition, pre-assignment of robots to specific positions is not robust to individual failures and it becomes increasingly more cumbersome as the swarm size grows. We believe that a progressive, decentralized method such as the one presented in this paper can solve these issues.

Our main idea is to represent the target shape as an acyclic directed graph in which each robot can find its position using two other robots (called *parents*) in the shape as reference. We assume that all robots possess the graph representation, but none is initially assigned to a specific position. The overall shape is built dynamically and iteratively: each new robot joins the shape only after being granted permission by one of the parents, using local communication exclusively. The resulting algorithm is completely decentralized and parallel: multiple robots can join different parts of the shape at any given time.

The rest of the paper is organized as follows. We discuss related work in Section 2. In Section 3, we present an algorithm to construct an acyclic directed graph starting from a point cloud which represents the target shape. In Section 4, we illustrate a mathematical model that proves the convergence properties of our decentralized algorithm. In Section 5, we describe the behavior the robots follow to achieve the target shape. We report experimental evaluation in Sections 6 (simulation) and 7 (real robots). The paper is concluded by Section 8, in which we outline future research directions.

2 Related Work

The deployment of a swarm of robots in patterns is a foundation for various practical swarm applications, and has been widely studied. Many of these studies focus on special patterns that are suited for specific applications. For instance, (Güzel et al, 2017) and Sepulchre et al (2008) focus solely on circle formation and parallel formation, while Majid and Arshad (2015) focus on a 'V' shaped pattern, and Paley et al (2008) manage to stabilize symmetric formations. These kinds of patterns are usually achieved through well-designed control laws, and have different communication or sensors requirements. However, the focus is in general towards relatively simple patterns, which constrain their application domain.

In this paper, we focus on a pattern formation scheme for arbitary patterns. To represent these patterns, several schemes are proposed in literature: for instance, Hsieh and Kumar (2006) and Zhang and Leonard (2006) use curves described by a formula for 2D geometrical patterns. Cheah et al (2009) propose a region-based formation method shaping the target region from a continuous mathematical representation of the frontier. Robots are then deployed to cover this region. This is achieved with a control law derived using the shape of the region as a global objective for all robots, and maintaing a minimum distance between neighbors as a local objective function. Similarly, potential fields were also proposed (Hsieh et al (2008)). Zhang et al (2007) use smooth curve-based formation for ocean sampling and, in later work, design a control algorithm to form a pattern defined by a boundary curve without requiring a global positioning system (Zhang and Haq (2008)). Compared to our work, the existing methods either require robots to sense the position of all robots involved in the formation (and not only the neighbors), or do not support progressive formation. Cowley and Taylor (2007) introduced an incremental pattern formation that allows the construction of symmetrical and recurring shapes that can be described by particularly crafted functions. The work was not validated on real robots or in presence of noise, and it is limited in terms of the type of shapes that it can form. We show that our approach works with noise, and for formations defined by arbitrary point clouds.

Another way to represent the desired pattern is to use a point cloud to define the relative positions of each robot. Robots are then directed to the desired positions following a control law. The process of assigning each robot to a specified point is similar to a task allocation problem, as the control law to drive the robots to their destination is related to path planning. In fact,

Belta and Kumar (2002) focus on moving a swarm of robots while keeping it in formation, and Michael et al (2008b) propose a task allocation method that can be used in pattern formation. More recently, the work of Turpin et al (2013) considers the problem of assigning a swarm of robots to specific goals, and gives a trajectory planning method. Similar work is done by Turpin et al (2012b), in which the robots are required to move while keeping a desired shape, and each robot plans its own trajectory based on relative location information. While all of these methods have interesting applications, their experimentation is limited to simple shapes, and they do not address the problem of progressive deployment.

Closer to our approach, Desai et al (2001) uses graph theory to assign robot positions in a given formation. Robots in the formation determine their neighbors' relative position based only on distance or using both distance and bearing. Scalability of such methods was proven by Rubenstein and Shen (2008), in which they show control of a large number of robots in formation. This method also provides a mechanism to change the formation and to self-heal if a robot fails. The method is further implemented at large scales using physical robots in Rubenstein et al (2014): the swarm is initialized with static seed robots and then generates a coordinate frame to identify the location of each swarm member, and robots are then driven purely by virtual potentials. In comparison, our work does not need seed robots, and it can be progressively deployed.

The leader-follower method is also popular to manage swarm behaviors, where one or more robots are treated as a reference for the others (Tanner et al (2002)). For example, Fierro et al (2001a) focus on controlling a swarm of Unmanned Aerial Vehicles (UAVs) using a leader-follower structure; in their work, the trajectory is planned for the leader, and the followers only maintain their relative position. Fierro et al (2001b) also present a leader-follower structure, with robots switching between continuous-state different control laws to achieve the desired pattern.

Turpin et al (2012a) represent the desired pattern with shape vectors, and Anand et al (2014) generate a pattern to move objects, where a master robot directs all other robot slaves. Michael et al (2007) and Michael et al (2008a) use an aerial robot to control a team of ground robot to move as a pattern: by using a rough model of the formation, the aerial robot can control the formation without knowing details about each robot in the formation.

Compared to these works, our method also uses a leader-follower hierarchy based on a graph. Each robot needs to confirm its destination according to the positions of its predecessors in the graph. However, the allocation of robots on the graph is not predefined, and a robot joining the formation dynamically finds a suitable predecessor, thus providing flexibility and fault tolerance.

Virtual forces or virtual potentials are widely used in pattern formation. Zhang (2007) and Yang and Zhang (2010) present an approach based on Jacobi shape theory for robot swarms, reaching a formation with a set of virtual forces. Chen and Chu (2013) use a model based on attractive and repulsive interaction between robots. Methods presented in these works are similar to the *Joining* phase of our method, which in contrast is a whole deployment process consisting of multiple phases, and includes a strategy for robots to find proper predecessors. Besides, as the theoretical basis of these works are different, so are the virtual forces.

Additionally to the previously mentioned work, a handful of publications are addressing a similar question as our solution: achieving any given shape defined as a point cloud in a decentralized manner. Spletzer and Fierro (2005) reorganize a swarm of robots to form new shapes while minimizing the maximum distance any robot travels, or the total distance traveled by the group. Ravichandran et al (2007) provide a scalable distributed algorithm to achieve an arbitrary formation based on a distributed median consensus estimator, requiring only local communication. Yu and Nagpal (2008) focus on the theoretical study of decentralized control for sensing-based formation on modular multi-robot systems. Yu and Nagpal (2008) use local sensor constraints between neighboring robot agents to define the desired shape. Alonso-Mora et al (2011) provide a method to form arbitrary target patterns independently from the number of robots. Liu and Shell (2014) tackle the problem of changing smoothly between formations of spatially deployed multi-robot systems. The main difference between our approach and these methods is that we propose an incremental formation scheme, where new robots join the pattern without disrupting the partially formed structure.

3 Progressive Formation

This paper can be considered a case study in progressive swarm deployment: robots are progressively introduced in an environment and are required to form a complex shape, without the knowledge of all robot positions. In this context, the swarm members cannot share a global coordinate system, but they have access to a local coordinate system defined by their own frame of reference. Interaction among robots is achieved through situated communication, i.e., a robot can get the relative range
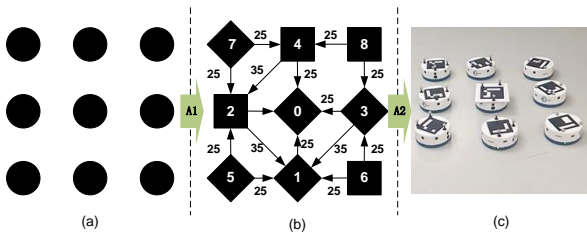
Fig. 1 Steps of the progressive formation approach proposed in this paper: (a) the desired formation depicted as a 2D point cloud, (b) the acyclic directed graph generated with algorithm A1, (c) from the behavior law A2, robots reach their final formation. In (b), numerical labels indicates the nodes in the graph, with the arrows identifying the parents of a node and their associated number representing the distances between robots. The markers' shape indicate the side a node is w.r.t. the vector between its parents (using the right-hand rule): squares are on the positive side, while diamonds are on the negative side.

and bearing (RAB) of the sender when it receives a message (Støy, 2001).

We propose a progressive formation approach. As shown in Fig. 1, this method consists of two stages:

1. We generate a directed acyclic graph (DAG) from a point cloud representing robot positions. This graph specifies a parent-child relationship between triads of positions (two parents, one child). As soon as a robot can identify both its parents, it can join the formation;

2. We propose a set of behavior laws to direct a robot to the target position based on the information sent from its parents, until it reaches its target position in the formation.

A robot is guided to its target position by robots that have already joined the formation. These robots are called *parents* of the robot. With the exception of the first two deployed robots (robots 0 and 1), our solution requires that each robot is able to sense 2 parents. We chose 2 parents, instead of another number, as a compromise between the communication requirements and the robustness of the method. With only 1 parent, robots need to exchange their relative positions to establish a common frame of reference. Using 2 parents does not require this exchange, lowering the communication requirements. When considering more than 2 parents, it is worth noting that a robot is unable to join the pattern until all of its parents reached their position (which would slow down the formation process), and the robot must be able to sense all of them (which gets harder as the number of parents increases).
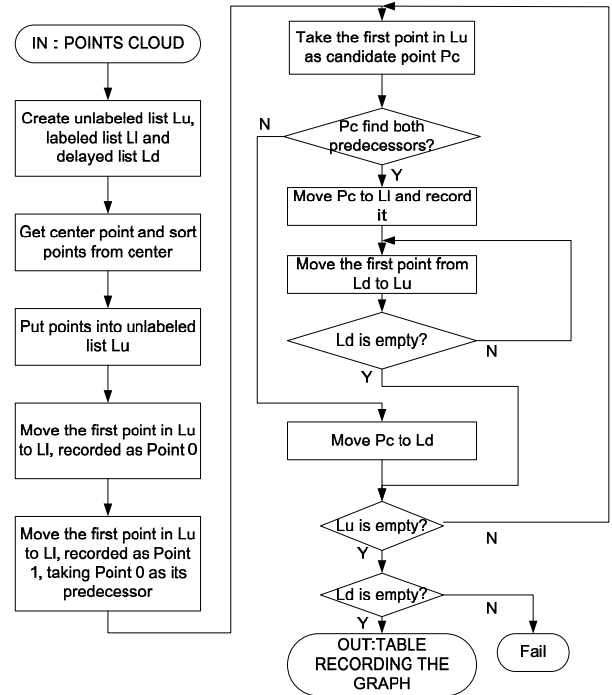
Fig. 2 Task flow of our graph generation from a point cloud algorithm

3.1 Directed Acyclic Graph Generation

We propose an algorithm to produce a DAG from a 2D point cloud, based on the flow graph shown in Fig. 2. Since individual robots must converge to a desired position in a global reference, the graph encodes two types of information:

1. The relative distance between a robot and its parents; and

2. The orientation of the robot with respect to the vector connecting its two parents.

To compute the graph, we use three lists over which we iterate until all points are labeled:

1. **Unlabeled** list: points that not have been assigned a label yet;

2. **Labeled** list: points with a label from the graph;

3. **Delayed** list: points that could not be labeled in the current iteration.

The algorithm starts by labeling point 0. This is the initial point for the formation process and it can be chosen based on different criteria such as distance, hierarchy, etc. In this implementation, we elect the nearest point to the center of the point cloud, so that the formation will expand from the inside out, gradually. Subsequently, all other points are sorted according to their distance to the center. The closest point to point

0 is labeled 1, and both are pushed in the *Labeled* list. Point 0 has no parents, while point 1 has one.

All the other points are pushed in the *Unlabeled* list and will be labeled through an iterative process to find their two parents. At each iteration, a candidate is pulled from the *Unlabeled* list assuming that all points in the *Labeled* list could be its parent. To select a pair of parents, we test pairs of points in *Labeled* as potential parents according to the distances to the candidate. The pairs of potential parents nearest to the candidate are tested first, and the first two points that pass the test are chosen as the parents. Assuming that no distance between two adjacent points in the graph is greater than the robot communication range, the robots are always able to communicate with their parents.

Pairs of points from the *Labeled* must pass two tests: (1) they form a triangle with the current candidate, and (2) no other points is present at the edges of this triangle. If this is the case, the candidate is moved from the *Unlabeled* list to the *Labeled* list, getting a label according to its position in the *Labeled* list. If the search fails to find a pair of parents, the point is transferred the *Delayed* list. When all points of the *Unlabeled* list are checked, the remaining points in the *Delayed* list are pushed back to the *Unlabeled* list and another iteration begins.

The algorithm ends after checking the *Delayed* list. If this list is empty, all points have been labeled; otherwise, the algorithm fails. Our 2-parent representation might not be able to represent every possible point cloud, such as cases in which too many points are placed along a single line. In practical applications, if acceptable, the point cloud could be modified to introduce slight perturbations that break the linearity of the shape.

Once a point is moved to the *Labeled* list, an element is added to the graph, which is recorded in a table to be later distributed to the swarm. Each element of this table contains six items: the label, parent 1, parent 2, distance to parent 1, distance to parent 2, and the side of the position with respect to the two parents. For instance, Tab. 1 depicts the graph of Fig. 1 with nine points. The point with label 0 in the graph has no parents, and point labeled 1 has only one. The value -1 is used to indicate an unused cell in the table. The main drawback of the current method is that, as the point labeled 1 has only one parent, the corresponding robot can end up anywhere on a circle around the robot labeled 0. As the rest of the formation is built around these two robots, the orientation of the formation is not fixed. This problem can be easily resolved by recording both the relative distance and orientation of the point labeled 1 w.r.t. the point labeled 0 (Li et al, 2017).

Table 1 The acyclic directed graph in Fig. 1(b), recorded as a table

Label	P1	Dis1	P2	Dis2	Side
0	-1	-1	-1	-1	1
1	0	25	-1	-1	1
2	0	25	1	35	0
3	0	25	1	35	1
4	0	25	2	35	0
5	1	25	2	25	1
6	1	25	3	25	0
7	2	25	4	25	1
8	3	25	4	25	0

4 Mathematical Model

We build a mathematical model to derive the control law that drives the robots to their positions on the graph. This model is used to guarantee that the desired formation can be reached.

4.1 Formulation

Consider a team of N robots labeled as $i \in \{0, ..., N-1\}$ moving in $\mathbb{R}^2$ space. With respect to a fixed global reference frame $\varnothing^g_{xy}$, the position of each robot can be represented by a vector $\mathbf{p}_i(t) = [x_i(t), y_i(t)]^T \in \mathbb{R}^2$. We assume that the dynamics of each robot is:

$$\dot{\mathbf{p}}_i = \mathbf{u}_i \qquad (1),$$

From the conditions described in Section 3, each robot i does not know its absolute position $\mathbf{p}_i(t)$. Situated communication (Støy, 2001) provides the relative distance $d_{ij}(t) = \|\mathbf{p}_i(t) - \mathbf{p}_j(t)\|$ and the relative bearing $\theta_{ij}(t)$ of the j neighbor robots in direct line-of-sight. Commercial robots such as the e-puck (Mondada et al, 2006) can be equipped with devices that perform situated communication. Based on the prerequisites of the earlier sections, the problem is defined as follows.

Problem 1. Given a desired formation expressed as a set of positions $\{\bar{\mathbf{q}}_0, \bar{\mathbf{q}}_1, ..., \bar{\mathbf{q}}_N\}$, determine a translation $\mathbf{r} \in \mathbb{R}^2$ and a rotation $R(\theta)$ such that

$$\lim_{t \to \infty} R(\theta)\mathbf{p}_i(t) + \mathbf{r} = \bar{\mathbf{q}}_i$$

It is preferable to create a target shape that can arbitrarily rotate and translate with respect to a fixed global reference, rather than forcing all robots to converge to specific positions. For the sake of simplicity, we consider $\bar{\mathbf{q}}_0 = [0, 0]^T$ and $\bar{\mathbf{q}}_1 = [x_1, 0]^T$ with $x_1 > 0$, without loss of generality, in the rest of this section. Under this assumption, $\mathbf{p}_i^{01}$ is treated as the coordinates of the i-th robot in the reference frame centered in $\mathbf{p}_0$. The x axis of the reference is oriented with the vector

connecting $\mathbf{p}_0$ to $\mathbf{p}_1$, while the y axis with a $\pi/2$ anti-clockwise rotation of x around $\mathbf{p}_0$. Problem 1 can then be expressed in new terms as:

Problem 2. Given a desired formation $\{\bar{\mathbf{q}}_0, \bar{\mathbf{q}}_1, ..., \bar{\mathbf{q}}_n\}$ where $\bar{\mathbf{q}}_0 = [0,0]^T$ and $\bar{\mathbf{q}}_1 = [x_1, 0]^T$ and $x_1 > 0$, ensure that $\dot{\mathbf{p}}_i \to 0$ and $\mathbf{p}_i \to \bar{\mathbf{q}}_i$ $\forall i \in \{0, 1, ..., N-1\}$.

4.2 Proposed Solution

For the first robot, at the center of the formation, a simple solution to Problem 2 is to maintain its current position, i.e.,

$$\mathbf{u}_0 = \mathbf{0} \qquad (2)$$

For the second robot to join the formation, an equally simple solution is to apply any control law of the form

$$\mathbf{u}_1 = \mathbf{f}_1(d_{0,1}(t)) \qquad (3)$$

that ensures that $d_{0,1}(t)$ asymptotically tends to $||\bar{\mathbf{q}}_0(t) - \bar{\mathbf{q}}_1(t)||$. Here we use the Coulomb potential (Spears et al, 2004), a distance-based attraction/repulsion law, to derive $\mathbf{f}_1$.

The control law for the other robots ($i > 1$) is based on a continuously differentiable potential field $\Phi_i(\mathbf{p}_i, \mathbf{p}_j, \mathbf{p}_k)$, $\Phi_i : \mathbb{R}^2 \times \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}$, which can be considered as being generated by the two parent nodes j and k such that, for any two bounded $\mathbf{p}_j$ and $\mathbf{p}_k$, these assumptions hold:

A1: $\Phi_i(\mathbf{p}_i, \mathbf{p}_j, \mathbf{p}_k)$ is invariant to rototranslations:

$$\Phi_i(R(\theta)\mathbf{p}_i + \mathbf{r}, R(\theta)\mathbf{p}_j + \mathbf{r}, R(\theta)\mathbf{p}_k + \mathbf{r}) = \Phi_i(\mathbf{p}_i, \mathbf{p}_j, \mathbf{p}_k)$$

A2: $\Phi_i(\mathbf{p}_i, \mathbf{p}_j, \mathbf{p}_k)$ is unbounded for unbounded $\mathbf{p}_i$:

$$\lim_{\|\mathbf{p}_i\| \to \infty} \Phi_i(\mathbf{p}_i, \mathbf{p}_j, \mathbf{p}_k) = \infty$$

A3: If p_j and p_k are distinct, $\Phi_i(\mathbf{p}_i, \mathbf{p}_j, \mathbf{p}_k)$ admits only one stationary point, which we denote as $\bar{\mathbf{p}}_i(\mathbf{p}_j, \mathbf{p}_k)$, and that point is a minimum of the potential field. This corresponds to forcing robot i to only have one possible position to reach when communicating with robots j and k.

A4: Two finite scalars α_{max} and β_{min} exist such that for any $\mathbf{p}_j$, $\mathbf{p}_k$

$$-\nabla\Phi_i(\mathbf{p}_i, \mathbf{p}_j, \mathbf{p}_k)^T \nabla\Phi_i(\mathbf{p}_i, \mathbf{p}_j + \mathbf{d}_j, \mathbf{p}_k + \mathbf{d}_k) \le 0$$

for any $\|\mathbf{d}_j\|$, $\|\mathbf{d}_k\| \le \alpha_{max}$, and for any $\mathbf{p}_i = \bar{\mathbf{p}}_i(\mathbf{p}_j, \mathbf{p}_k)$ with $\|\mathbf{d}_i\| \ge \beta_{min}$.

This means that, for bounded perturbations of the parents $\mathbf{p}_j$ and $\mathbf{p}_k$, if $\mathbf{p}_i$ is sufficiently far from the equilibrium, the perturbed anti-gradient is still a direction of descent of the non-perturbed potential field.

We prove the following theorem by using a control law in the form:

$$\mathbf{u}_i = -\nabla\Phi_i(\mathbf{p}_i, \mathbf{p}_j, \mathbf{p}_k) \qquad (4)$$

Theorem 1. Consider the system described by Equation (1) and controlled by Equations (2)−(4) that comply with assumptions **A1**−**A4**. Under the constraints:

1. If j and k are parents of i, then $j < i$ and $k < i$;
2. $\bar{\mathbf{p}}_i(\bar{\mathbf{q}}_j, \bar{\mathbf{q}}_k) = \bar{\mathbf{q}}_i$ $\forall i \in \{0, 1, ..., N\}$;

Problem 2 is then solved for any set of initial conditions.

Proof. Due to Equation (2), at $t = 0$, robot 0 is already stationary in its final position. Concerning robot 1, due to Equation (3), $\mathbf{p}_1(t)$ converges asymptotically to $\bar{\mathbf{q}}_1$. As for the other robots ($i > 1$), by using Equation (4), we can write

$$\dot{\mathbf{p}}_i = -\nabla\Phi_i(\mathbf{p}_i, \mathbf{p}_j, \mathbf{p}_k).$$

For fixed and non-coincident $\mathbf{p}_j$ and $\mathbf{p}_k$, global asymptotic stability of the point of equilibrium $\bar{\mathbf{p}}_i(\mathbf{p}_j, \mathbf{p}_k)$ directly follows using

$$V = \Phi_i(\mathbf{p}_i, \mathbf{p}_j, \mathbf{p}_k) + \Phi_i(\bar{\mathbf{p}}_i(\mathbf{p}_j, \mathbf{p}_k), \mathbf{p}_j, \mathbf{p}_k)$$

as a Lyapunov function along with assumptions **A2** and **A3**. For robot 2, it is then possible to write

$$\dot{\mathbf{p}}_2^{0,1} = -\nabla\Phi_i(\mathbf{p}_2^{0,1}, \bar{\mathbf{q}}_0, \bar{\mathbf{q}}_1 + \boldsymbol{\delta}_2(t)) \qquad (5)$$

for which $\bar{\mathbf{q}}_1$ is Globally Asymptotically Stable (GAS) for $\boldsymbol{\delta}_2(t) = \mathbf{0}$. The term $\boldsymbol{\delta}_2(t)$ can be seen as a vanishing output of the system

$$\begin{cases} \dot{\mathbf{p}}_0^{0,1} = 0 \\ \dot{\mathbf{p}}_1^{0,1} = \mathbf{f}_1(\bar{\mathbf{q}}_0, \dot{\mathbf{p}}_1^{0,1}) \\ \boldsymbol{\delta}_2(t) = \dot{\mathbf{p}}_1^{0,1} - \dot{\mathbf{q}}_1. \end{cases} \qquad (6)$$

Equations (5) and (6) can be seen as the cascade of two GAS systems which, according to Theorem 1.1 in (Seibert and Suarez, 1990), is GAS if the trajectories of the resulting system are bounded. Since Equation (5) satisfies **A4**, the trajectories of the resulting systems are bounded (see also Theorem 1.2 in Seibert and Suarez, 1990). By repeating the same procedure for $i \in \{3, ..., N-1\}$ the theorem is proved. $\square$

Since every position $\bar{\mathbf{q}}_i$ in the target shape is expressed with respect to two other positions $\bar{\mathbf{q}}_j$ and $\bar{\mathbf{q}}_k$, and $j < i$ and $k < i$, the target shape can be presented as a directed acyclic graph in which every node i is connected to its parents j and k. Thus, the process of positioning a specific robot i can ideally be seen as a cascade of positioning processes, which starts with robot 1 and proceeds with all other robots. If every step of the positioning dynamics is globally asymptotically

stable, then the entire process converges to the desired shape. This is the core idea of this proof.

For some applications, assumptions **A3** and **A4** might prove to be too strong. For instance, the existence of a single stationary point topologically forbids the use of repulsion terms among robots that are usually used for collision avoidance. To fix this, consider the following relaxed condition:

A3bis: If $\mathbf{p}_j$ and $\mathbf{p}_k$ are distinct, $\bar{\mathbf{p}}_i(\mathbf{p}_j, \mathbf{p}_k)$ exists, which is an isolated minimum of $\Phi_i(\mathbf{p}_i, \mathbf{p}_j, \mathbf{p}_k)$.

By using **A3bis** instead of **A3** and dropping assumption **A4**, the formation is asymptotically stable. This means that a basin of attraction around the configuration of equilibrium $\dot{\mathbf{p}}_i^{0,1} = \bar{\mathbf{q}}_i$ exists for which Problem 2 is solved.

Theorem 2. Consider the system described by Equation (1) and controlled by Equations (2)−(4) that comply with assumptions **A1**, **A2**, and **A3bis**. Under the constraints:

1. If j and k are parents of i, then $j < i$ and $k < i$;
2. $\bar{\mathbf{p}}_i(\bar{\mathbf{q}}_j, \bar{\mathbf{q}}_k) = \bar{\mathbf{q}}_i \ \forall i \in \{0, 1, ..., N - 1\}$;

Problem 2 is solved for any set of initial conditions in a suitable neighborhood of the equilibrium configuration of $\dot{\mathbf{p}}_i^{0,1} = \mathbf{q}_i \ \forall i \in \{0, 1, ..., N - 1\}$.

Proof. Similarly to the proof of Theorem 1, the overall system can be seen as a cascade of N systems such that, in the absence of interconnection, $\dot{\mathbf{p}}_i^{0,1} = \mathbf{q}_i$ are asymptotically stable. This implies (see Theorem 4.1 in Seibert and Suarez, 1990) that $\dot{\mathbf{p}}_i^{0,1} = \mathbf{q}_i$ is asymptotically stable for the cascaded system. The theorem statement follows by definition of asymptotic stability. □

The above proof demonstrates that the controller will converge to the desired shape as far as all robots received their destination and identified the proper parents. Section 5 introduces a finite state machine that assigns each robot a proper destination and proper parents. Then the described control law is applied to drive robots to the destination, allowing the whole swarm to form the desired pattern in an incremental way. The convergence for the overall behavior is shown in experiments in Section 6 and 7.
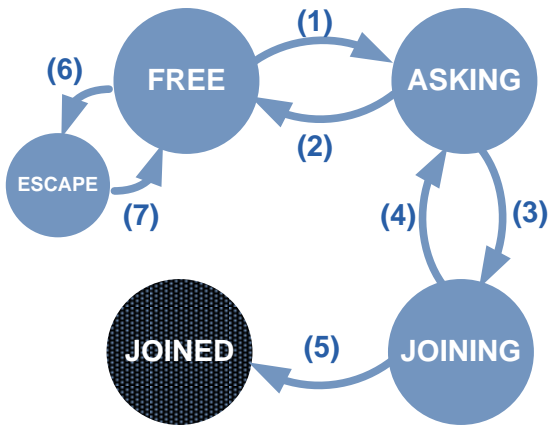
5 Behavior

We assume that all robots involved in the formation are aware of the formation graph, as described in Section 3. This can be achieved by downloading the table representing the graph before the robots' deployment, or through run-time broadcast. However, the robots are not pre-assigned to a specified label in the graph. The behavior allows them to find proper labels through simple local interactions with other robots, including robots already part of the formation and robots not yet in the formation. This process can drive free robots to participate in the formation gradually or, from the perspective of the formation, it attracts free robots to join from the edges of the current formation, allowing it to grow dynamically. Of course, the assumption that all robots are aware of the graph can be considered a limiting constraint. However, the graph can be broadcasted by robots already in the formation, so that nearby free robots receive the graph and then participate to the formation progressively.

As mentioned in Section 3, the formation process starts when a robot gets the label 0 in the graph, and is considered as part of the formation. In the context of progressive deployment, this robot is simply the first robot to enter the scene. If multiple robots are deployed simultaneously, position 0 can be determined using one of the many election schemes available in the literature (e.g. Dieudonné and Petit (2007); Petit (2009); Karpov and Karpova (2015)), or we can assign label 0 to a specially chosen robot. For example, robot 0 could be a stakeholder that allows a user to monitor and control the swarm, or it could be the first robot to reach the designated position for the center of the graph. Robot 0, as explained earlier, does not move, but all other robots gradually position themselves around it. Robot 0 can also act as the interface with a human to interact with the swarm and manually control the formation process or change its shape. It is worth noting that this does not prevent the whole formation process to be entirely distributed, as there is no central control node.

5.1 Behavior structure

The behavior is represented as a finite state machine, shown in Fig. 3. It consists of four states, which are *Free*, *Asking*, *Joining* and *Joined*. Tab. 2 shows the state transition conditions. A robot in state *Free* and *Asking* is still not part of the formation: it circles around the edge of the formation, namely the structure composed of *Joining* and *Joined* robots, and searches for a proper label in the graph. When such a label is found, and both its parents are in sight, the *Free* robot transitions to state *Asking*, sending a message to request the label. Once the request is approved by the *Joining* and *Joined* robots, the robot transitions to state *Joining*. From then on it becomes part of the formation and at the position designated by the acquired label. Based on the control law described in Section 4, the robot is attracted and repelled by its parents, moving to the target position in the formation and stopping there. If no obstacles are encountered and if the parents stay in

Fig. 3 The behavior represented as a finite state machine. Every robot joining the formation will experience states *Free*, *Asking*, *Joining* and *Joined* and finally *Joined*. The secondary state *Escape* is active when a free robot is trapped among *Joined* robots.

Table 2 State transition condition in Fig. 3

Marker	Description
(1)	Found a label
(2)	Request refused
(3)	Request granted
(4)	Lost parents
(5)	Reached target
(6)	Trapped inside joined robots
(7)	Escape from joined robots

sight, the robot finally converges to the desired position and switches to state *Joined*. In the *Asking* state, several robots may ask for the same label simultaneously. In this case, only one is conferred the label, while the rest transitions back to state *Free*. If a *Joining* robot loses communication with its parents, it transitions to state *Asking* to request the label again, from which it may receive the same label or have to transition back to state *Free*. It is possible that a *Free* robot gets trapped among joined robots. In this case, the robot switches to a secondary state, *Escape*, to evade its trapped location and find a safe spot outside the formation. The whole formation process is shown in Fig. 4.

5.2 States

Each of the states described in Fig. 3 are detailed in this section, together with the communication scheme and control law.
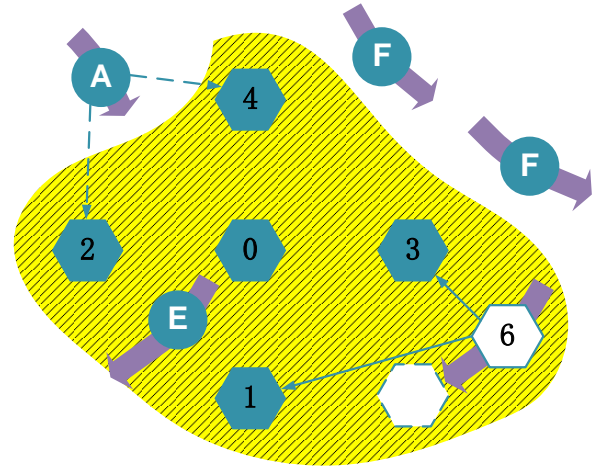
Fig. 4 The overall formation process. Round markers are robots not participating to the formation, with a letter indicating their state. Hexagon markers are robots already part of the formation, with indication of their label. Arrows show the predicted motion of each robot, and the shadow represents the virtual force field generated by the current formation. Letter 'F' indicates a *Free* robot, slipping at the edge of the virtual force field. Letter 'E' indicates a robot trapped in the formation aiming at a safer outside position. Letter 'A' indicates an *Asking* robot, trying to keep the two candidate parents within sight. The full hexagon markers are *Joined* robots that can act as parents, and the hollow hexagon marker is a *Joining* robot driven by the parents to the target position, represented without a numerical label.
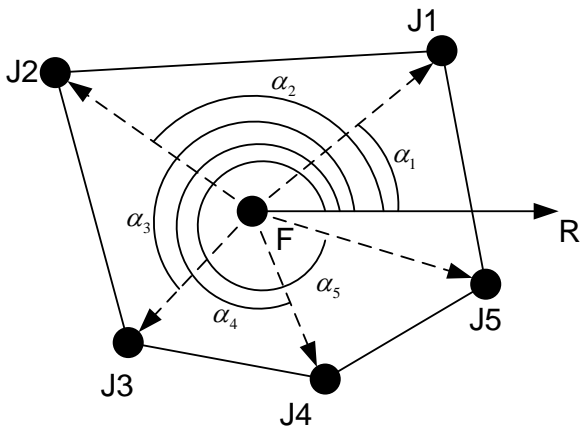
5.2.1 Free State

A robot in this state does not have a label because it is not yet part of the formation. Robots in this state update their graph according to the robots that have already joined the formation. They are notified if a labeled node in the graph is already occupied or not. A *Free* robot searches for an available label and when one is found, if both parents are also within sight, transits to the state *Asking* to send a request to join the formation with this label.

Free robots only broadcast their state, so that robots nearby can sense them and avoid collisions. They circle around the current formation composed of *Joined* robots and *Joining* robots. This is achieved by applying three virtual forces:

$$\mathbf{u} = f(\mathbf{F}_p, \mathbf{F}_v, \mathbf{F}_a),$$

where $\mathbf{F}_p$ is a force keeping the robot away from the structure, $\mathbf{F}_v$ is a force perpendicular to $\mathbf{F}_p$ that compels the robot to circle around the structure, and $\mathbf{F}_a$ is a force pushing *Free* robots away from nearby robots to avoid collisions. The function $f(\cdot)$ maps the virtual force to a navigation command.

Fig. 5 Schematic of the *Free* state test. The F point is a *Free* robot. Assuming it has five *Joined* neighbors, dots J1 to J5, it is trapped. Ray FR is the reference direction of the local coordinate of the *Free* robot, so α_1 to α_5 are the bearings of each *Joined* neighbor w.r.t. this local frame. Sorting the neighbors according to angle α_1 to angle α_5 and lining them up, a polygon consisting of 5 *Joined* neighbors of the *Free* robot is created.

5.2.2 Escape State

The *Escape* state is a secondary state under state *Free*. Robots in this state follow the same communication rule as a *Free* robot, but following a different behavior. If a *Free* robot is trapped, i.e., it is inside a polygon created by *Joined* robots, it is unable to move along the outer edge of the already formed pattern and find a proper position to join. To free itself from the polygon, the robot transits to state *Escape*. A *Free* robot tests if it is trapped at each control step. As in Fig. 5, a *Free* robot sorts all its *Joined* neighbors according to their relative bearing. By lining them up, we can obtain a polygon composed of the *Joined* neighbors. Then, we draw a ray from the *Free* robot. By counting the number of intersection points between the ray and the edges of the polygon, the *Free* robot can check if it is inside the polygon, which occurs when the number of intersections is odd. A trapped robot can free itself by moving towards the *Joined* neighbor with the highest label and then rotate around it. This escape strategy works because the labeling algorithm grows the target graph from the inside out, so labels with higher values are closer to the exterior boundary of the formation.

5.2.3 Asking State

Once a robot gets a proper label and both parents are within sight, it sends a message containing three items to request this label. The message is a tuple *(Asking, ReqLabel, ReqID)*.

The first element encodes the state of the robot, *ReqLabel* is the label being asked, and *ReqID* is a randomly generated unique ID used to counter the situation where multiple robots ask for the same label.

The reply received from a *Joined* robot is a tuple *(Joined, ReqLabel, ReqID, Reply)*. The *Asking* robot checks if the *ReqLabel* is the same as the label requested. If this is the case, the robot further checks the *Reply* value. If it is granted, and the *ReqID* is the same as the one it sent originally, then the robot takes the label and transitions to *Joining*. If not, it transitions back to *Free*, looking for another available label.

The control law during the *Asking* stage must keep the parents in sight. We use attraction to the joined robots and repulsion from the surrounding free robots through the following law:

$$\mathbf{u} = f(\mathbf{F}_p + \mathbf{F}_a)$$

where $\mathbf{F}_p$ is the virtual force generated by the joined robots, keeping the the *Asking* robot a given distance from the already formed pattern; and $\mathbf{F}_a$ is the virtual force generated by the nearby robots, pushing the *Asking* robot away to avoid collisions. Function $f(\cdot)$ maps the virtual force into velocity commands: it takes the vector of virtual force $v_{\text{force}} = (\alpha_{\text{force}}, d_{\text{force}})$ as input, with α_{force} being the direction of the vector, and d_{force} its magnitude. The output will also be a vector $v_{\text{velocity}} = (\alpha_{\text{velocity}}, d_{\text{velocity}})$, where:

$$\alpha_{\text{velocity}} = \alpha_{\text{force}}$$

and

$$d_{\text{velocity}} = \begin{cases} k \cdot d_{\text{force}} & \text{if } w \cdot d_{\text{force}} \leq v_{\text{threshold}} \\ v_{\text{threshold}} & \text{otherwise} \end{cases}$$

where w is a parameter to scale the magnitude (empirically determined) and $v_{\text{threshold}}$ is a speed threshold.

5.2.4 Joining State

According to the model described in Section 4, we derive a behavior as Equation (4) and prove that with this control law, robots can be driven to form the desired pattern. The behavior of a *Joining* robot is defined as

$$\mathbf{u}_i = -\nabla \Phi_i(\mathbf{p}_i, \mathbf{p}_j, \mathbf{p}_k)$$

The potential-field-based control law is interpreted as a set of virtual forces that are translated into a speed command. Based on $\mathbf{p}_i$, $\mathbf{p}_j$ and $\mathbf{p}_k$, robot i can get the position of parents j and k in its local coordinate frame as $\mathbf{p}_j^i = (d_j^i, \alpha_j^i)$ and $\mathbf{p}_k^i = (d_k^i, \alpha_k^i)$. Using the formation graph, the robot can also get the desired distance from the two parents as d_{ij} and d_{ik}. Robot

i first checks if it is on the correct side w.r.t. the two parents. This is done by normalizing vector $\mathbf{v}_{\mathrm{norm}} = (\mathbf{p}_k^i - \mathbf{p}_j^i)/\left\|\mathbf{p}_k^i - \mathbf{p}_j^i\right\|$, with $\mathbf{v}_{\mathrm{perp}}$ being obtained by rotating $\mathbf{v}_{\mathrm{norm}}$ anticlockwise by $\pi/2$. If $\mathbf{v}_{\mathrm{perp}} \cdot \mathbf{p}_j^i > 0$, the robot is on the correct side. Let us assume robot i is on the correct side: if it is sufficiently far from both parents, i.e., $\left|\left\|\mathbf{p}_j^i\right\| - d_{ij}\right| > d_{\mathrm{safe}}$ and $\left|\left\|\mathbf{p}_k^i\right\| - d_{ik}\right| > d_{\mathrm{safe}}$, or it is already between the two parents, which means $\mathbf{p}_j^i \cdot \mathbf{v}_{\mathrm{norm}} < 0$ and $\mathbf{p}_k^i \cdot \mathbf{v}_{\mathrm{norm}} > 0$, it is applied a virtual force represented as $\mathbf{F}_i = \mathbf{F}_j + \mathbf{F}_k$, with $\mathbf{F}_j$ and $\mathbf{F}_k$ being two attracting forces generated by the two parents. If the robot is already near the target, namely $\left|\left\|\mathbf{p}_j^i\right\| - d_{ij}\right| < d_{\mathrm{threshold}}$ and $\left|\left\|\mathbf{p}_k^i\right\| - d_{ik}\right| < d_{\mathrm{threshold}}$, we use the Lennard-Jones Potential to compute the driving virtual forces as:

$$\mathbf{F}_j = (f_{\mathrm{LJ}}(d_j^i, d_{ij}), \alpha_j^i)$$

$$\mathbf{F}_k = (f_{\mathrm{LJ}}(d_k^i, d_{ik}), \alpha_k^i)$$

where

$$f_{\mathrm{LJ}}(d_{\mathrm{dist}}, d_{\mathrm{tar}}) = \frac{K_{\mathrm{LJ}}}{d_{\mathrm{dist}}} \cdot \left(\left(\frac{d_{\mathrm{tar}}}{d_{\mathrm{dist}}}\right)^4 - \left(\frac{d_{\mathrm{tar}}}{d_{\mathrm{dist}}}\right)^2\right)$$

in which K_{LJ} is an empirically determined parameter. If the robot is far from the target, which means $\left|\left\|\mathbf{p}_j^i\right\| - d_{ij}\right| \geq d_{\mathrm{threshold}}$ or $\left|\left\|\mathbf{p}_k^i\right\| - d_{ik}\right| \geq d_{\mathrm{threshold}}$, we use a concave potential as:

$$\mathbf{F}_j = (f_{\mathrm{con}}(d_j^i, d_{ij}), \alpha_j^i)$$

$$\mathbf{F}_k = (f_{\mathrm{con}}(d_k^i, d_{ik}), \alpha_k^i)$$

where

$$f_{\mathrm{con}}(d_{\mathrm{dist}}, d_{\mathrm{tar}}) = K_{\mathrm{max}} \cdot (\frac{d_{\mathrm{tar}}}{d_{\mathrm{dist}}} - 1)^{2K_{\mathrm{exp}} - 1}$$

in which K_{max} is a parameter set to the maximum speed of the robot, and K_{exp} is another empirically-determined parameter.

If the robot is not between the parents, and is on the side of parent j, which means $\mathbf{p}_j^i \cdot \mathbf{v}_{\mathrm{norm}} > 0$, the virtual force is

$$\mathbf{F}_i = K_{\mathrm{max}} \cdot \mathbf{v}_{\mathrm{norm}}$$

If the robot is on the side parent k, namely $\mathbf{p}_k^i \cdot \mathbf{v}_{\mathrm{norm}} < 0$, the virtual force is then

$$\mathbf{F}_i = -K_{\mathrm{max}} \cdot \mathbf{v}_{\mathrm{norm}}.$$

If the robot is on the wrong side of the parents, if it is far enough from both parents, as $\left|\left\|\mathbf{p}_j^i\right\| - d_{ij}\right| > d_{\mathrm{safe}}$ and $\left|\left\|\mathbf{p}_k^i\right\| - d_{ik}\right| > d_{\mathrm{safe}}$, or it is already between the two parents, which means $\mathbf{p}_j^i \cdot \mathbf{v}_{\mathrm{norm}} < 0$ and $\mathbf{p}_k^i \cdot \mathbf{v}_{\mathrm{norm}} > 0$, the robot goes between the parents with

$$\mathbf{F}_i = K_{\mathrm{max}} \cdot (\mathbf{p}_j^i + \mathbf{p}_k^i),$$

(a) $t = 10\,\mathrm{sec}$ (b) $t = 400\,\mathrm{sec}$ (c) $t = 1500\,\mathrm{sec}$
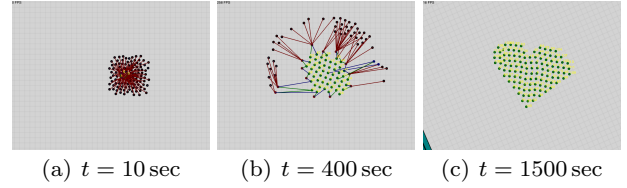
Fig. 6 Steps of the formation for a heart geometry formed by 114 robots. The black robots are *Free*, and the green-yellow robots are *Joined*. Screenshot are taken from the ARGoS simulator.

meaning it will move to the correct side.

At this stage, robots broadcast their state and label as the tuple *(Joining, Label)*. When the distance to the target is smaller than a tolerance ϵ, the robot transitions to state *Joined*. If for any reason it lost communication with its parents, it transitions back to *Asking*.

5.2.5 Joined State

A *Joined* robot is part of the formation, which means it can play the role of a parent for other robots. Such a robot keeps broadcasting its state and label with the tuple *(Joined, Label)*.

Once a request is received, a *Joined* robot checks if it is a parent of the point with the requested label. In this case, and if it is also the parent with the highest label, it is then responsible for sending a reply to the asking robot. If the point is already occupied by another robot, the *Joined* robot sends a refuse message *(Joined, ReqLabel,0,refuse)*.

If the position is still unoccupied, the joined robot will choose a robot among the candidates (if many are simultaneously asking) and reply as: *(Joined, ReqLabel, ReqID, granted)*.

In some cases, the free robot may be trapped in the middle of joined robots. The robot then switches to state *Escape* until it evades the trapped position.

6 Simulation Evaluation

The performance of the formation algorithm is evaluated with ARGoS (Pinciroli et al, 2012), a realistic, physics-based simulator for multi-robot systems. Fig. 6 shows how a heart-shaped formation is generated progressively (Pinciroli et al, 2016). A systematic evaluation follows.

6.1 Performance

The evaluation focuses on three aspects of the formation process:

- **Completion time**: the time required to form a complete shape;
- **Joining time**: the time elapsed between two successive join events;
- **Absolute positioning error**: E_{ij} between robots i and j is calculated as $E_{ij} = \left| (d_{ij} - \bar{d}_{ij})/\bar{d}_{ij} \right|$ where $\bar{d}_{ij}$ is the expected distance between robots i and j, and d_{ij} is the actual distance measured at the end of the experiment.

6.2 Simulation Setup

While our algorithm can form generic shapes, to ensure comparability among different experimental conditions, the robots are tasked with the formation of a grid. In our simulation, we test three grid sizes: 5×5, 10×10, and 15×15.

M robots are uniformly distributed in the experimental arena, an empty square of side L. We consider three density configurations D: broad ($D = 0.01$), medium ($D = 0.05$), and tight ($D = 0.1$). The arena side L is calculated with $L = \sqrt{(M\pi R^2)/D}$, where R is the radius of the robot ($R = 8.5\,\mathrm{cm}$). We use the marXbot (Bonani et al, 2010) robot model for our simulations, with a range-and-bearing sensor for situated communication, and with its maximum speed set to $10\,\mathrm{cm/s}$.

We also evaluate the scalability of the algorithm with the number of robots N required to complete the formation (e.g., $N = 25$ for a 5×5 grid). We deploy $M = kN$ robots, with $k \in \{1, 2, 3\}$. Every configuration $\langle \text{grid size}, D, k \rangle$ was tested 30 times, for a total of 810 runs.

6.3 Simulation results and discussion

We observed 3 failures out of all the 810 runs, leading to a failure rate of 0.37%. All the failures occurred with the high-density configuration ($D = 0.1$) and were due to a robot trapped in the middle of the formation at the very beginning of the formation process. No failure occurred with low and middle density distributions.

Fig. 7 shows that both completion time and joining time decrease with the factor k: this is because with $k = 1$, every robot needs to find a position in the formation and at the final stages of the formation process, there are few *Free* robots that have to circle the formation looking for a label. With $k > 1$ the number of *Free* robots increases, and so does the chance that a robot is close to an available position. We also observe that completion time slightly decreases with the initial robot density (the parameter D), which is to be expected as

denser robots have to travel a shorter distance to find a label. It is worth noting that the configuration parameters do not significantly affect the positioning error, which is always lower than 8% (with a median around 4%). This is expected because the final position of each robot in the formation relies on the virtual force generated by its parents and the position tolerance: assuming a robot can get accurate distance and bearing information from its neighbours and can physically travel to the target position, then the formation error is only affected by the parameters of the force field and the formation tolerance value. Note that this is in general not the case for real robots, and for this reason we conducted the experiments presented in the following section.

7 Experimental Evaluation

We carried out experiments with the Khepera IV robots[1] to verify our formation control algorithm. Fig. 8 shows our experimental platform, which consists of an Optitrack system (an infrared camera-based tracking system), a software communication hub, and a swarm of Khepera IV. To emulate situated communication, the robots send their messages wirelessly to the software hub. The software hub processes the messages received from the robots and calculates relative positioning information from the tracking system data, and relays them back to the receiving robots.

We ran two sets of experiments to validate our approach: *(i)* the formation of a regular 3×3 grid, repeated 10 times from different initial positions; and *(ii)* 4 formations with the shapes of the letters M, I, S and T, which spell the name of one of the laboratories involved in this work.

The formation control algorithm is implemented in Buzz (Pinciroli and Beltrame, 2016). Our implementation, shown in Algorithm 1, first reads the graph file and sets the experiment parameters within the `reset()` function. The `step()` function is executed 10 times per second, and it implements our algorithm's state machine.

We measure the performance of the algorithm as the positioning error of the formation $E_{ij} = d_{ij} - \bar{d}_{ij}$, where d_{ij} is the distance between robots i and j when all robots are in state *Joined*, and $\bar{d}_{ij}$ is the desired distance between the two robots.

7.1 Experiment Setup

Fig. 10 shows the results for the 3×3 grid formation, using red diamonds for the target positions. In this ex-

---

[1] https://www.k-team.com/khepera-iv
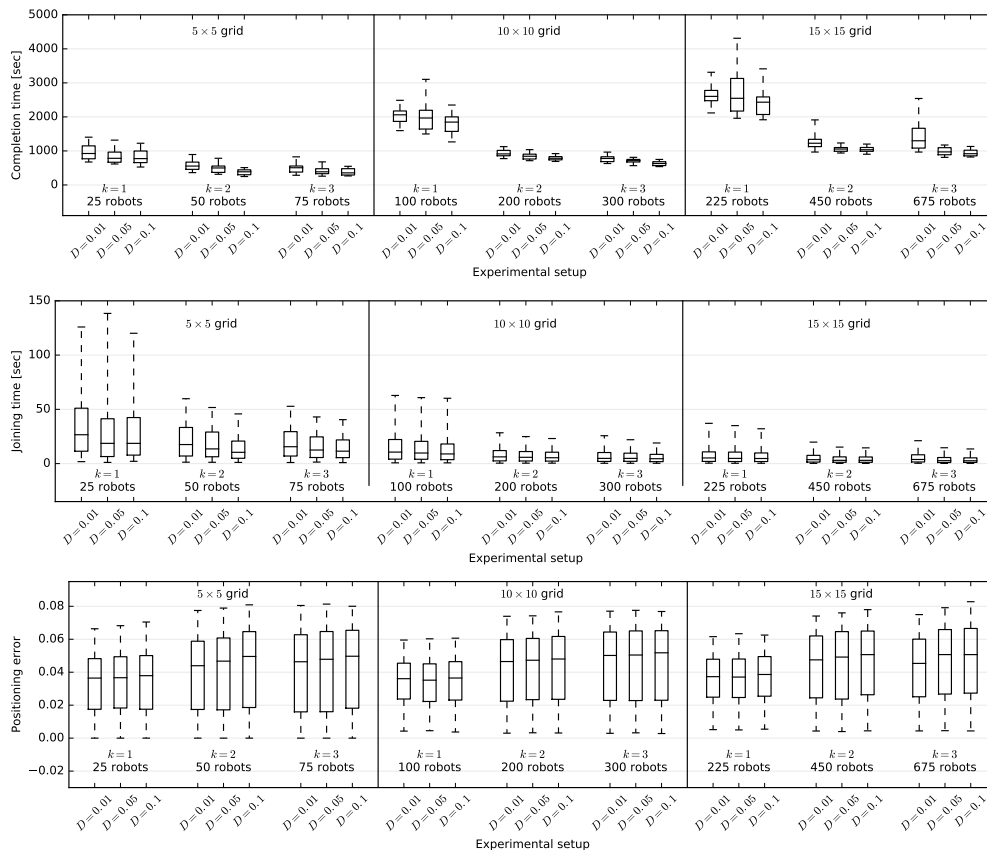
Fig. 7 Simulation results from (Pinciroli et al, 2016). Each box corresponds to the distribution of a performance measure for a specific experimental configuration ⟨grid size, D, k⟩ over 30 runs. The whiskers in the box plots indicate the 5th and 95th percentile. The top plot reports the completion time in seconds; the middle plot reports the joining time in seconds; and the bottom plot reports the positioning error.

periment, we fixed the robot with label 0 to be the Khepera with ID K01, and randomly selected the initial positions of all other robots shown in Fig. 9. The experiment was repeated 10 times, with different initial conditions.

For our second set of experiments (forming the "MIST" letters), the desired formation is shown in Fig. 13, with the targets marked with red diamonds. Each formation experiment was repeated twice, with random initial positions. The arena for the experiment was $2.5\,\mathrm{m} \times 2.5\,\mathrm{m}$ and the position tolerance was set to $3\,\mathrm{cm}$.

7.2 Experiment Results and Discussion

Fig. 9-11 report the results of the grid formation experiments. Fig. 9 shows tests with two initial positions as well as the final positions. Fig. 10 depicts the desired formation with red diamonds and the final positions of the robots of all tests; round markers with the same color indicate robots from the same test. As the orientation of the final formation cannot be fixed, we rotated and translated the graphs to overlay the

results. The final positions of robots converge around the targets, with an error in most cases smaller than the tolerance for joining the formation between -3 cm and 3 cm). The worst-case error was $\pm 6\,\mathrm{cm}$, and we attributed it to imprecise relative localization. Fig. 11 shows the details of the formation error for each test, as well as the aggregate error. Fig. 12 shows the results for the second set of experiments, where the robots form the shapes of the letters in the word "MIST". Fig. 13 depicts the final positions of the robots with round markers of different colors for different attempts. The robots converged to the targets with a formation error illustrated in Fig. 14, generally within the range of $\pm 2\,\mathrm{cm}$. A video for the experiments is available at `https://mistlab.ca/grid_MIST.avi`.

8 Conclusions

The progressive deployment of a swarm of robots can pave the way for many applications. In this paper, we proposed a case study of pattern formation, that can be applied to any shape described as a 2D point cloud.

Algorithm 1 Structure of the robot control code

```
function RESET()
    READGRAPH()
    INITIALSET()
    msg = LISTEN()
end function

function STEP()
    if state == Free then
        DOFREE()
    else if state == Asking then
        DOASKING()
    else if state == Joining then
        DOJOINING()
    else if state == Joined then
        DOJOINED()
    end if
    BROADCAST(msg)
end function

function DOFREE()
    SEARCHPROPERLABEL()
    if SEEPARENTS() and ONCORRECTSIDE() then
        state = Asking
    end if
    if ISTRAPPED() then
        DOESCAPE()
    else
        CIRCLEAROUNDPATTERN()
    end if
end function

function DOASKING()
    SENDREQUESTMESSAGE()
    if REQUESTAPPROVED() then
        state = Joining
    else if REQUESTDENIED() or TIMEROUT() then
        state = Free
    end if
    UPDATETIMER()
    STAY()
end function

function DOJOINING()
    if SEEPARENTS() then
        if label == 1 then
            NAVIGATEACCORDINGTOPARENT()
        else
            NAVIGATEACCORDINGTOPOTENTIALFIELD()
        end if
    else if REACHEDTARGET() then
        state = Joined
    else if TIMEROUT() then
        state = Asking
    end if
end function

function DOJOINED()
    if RECEIVEREQUEST() then
        CHOOSECHILD()
        APPROVEREQUEST()
    end if
end function
```
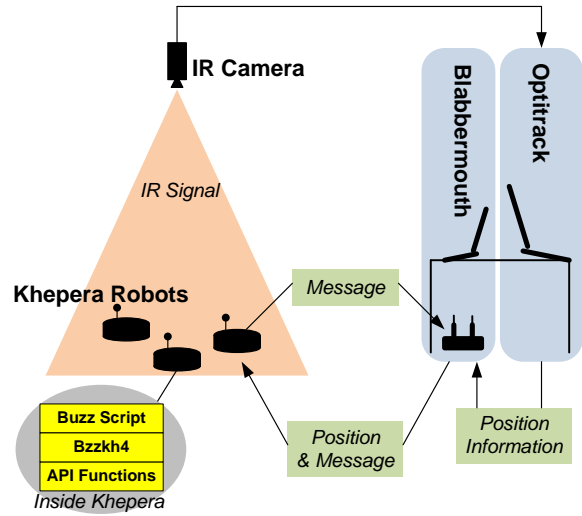

Fig. 8 Structure of the experimental platform. An Optitrack system gets the position of each robot through IR cameras. Blabbermouth acts as a communication hub, combining the communication messages and position information. The Khepera robots receive position-tagged messages, emulating situated communication. Our algorithm is implemented with a Buzz script.
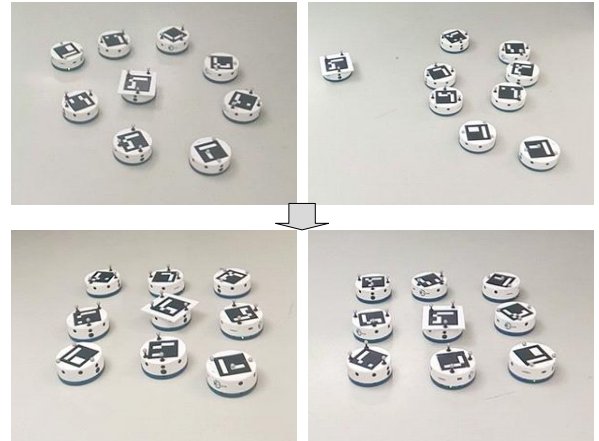

Fig. 9 Experiments with typical initial positions. The left side shows the initial and final position with robots initially placed around the robot with label 0. The right side show robots initially placed on one side of the robot with label 0.

The point cloud is transformed into a directed acyclic graph that is shared among the swarm members. When robots are deployed, they agree on their assigned position on the graph using only local communication and navigate there. This is a first step towards the definition of behaviors for progressively deployed swarms, and it shows how a formation can gradually grow in time, with guaranteed convergence for the joining process. In this paper, we provide a solid theoretical foun-
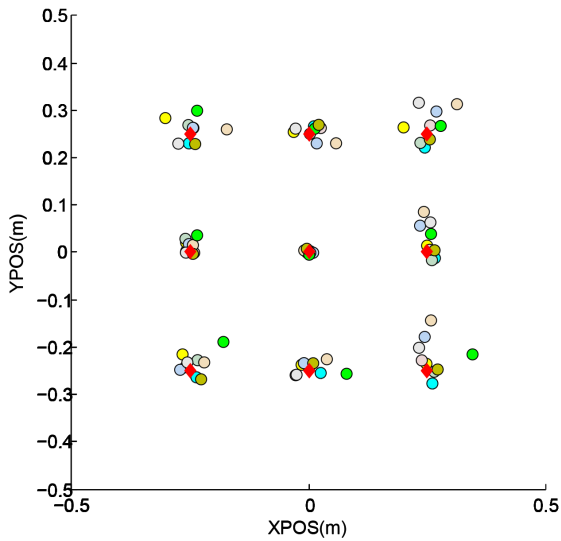
Fig. 10 Final positions of the robots after 10 tests. The red diamonds indicate the desired formation, while the round markers show the experimental results. Markers with the same color represent the same test.
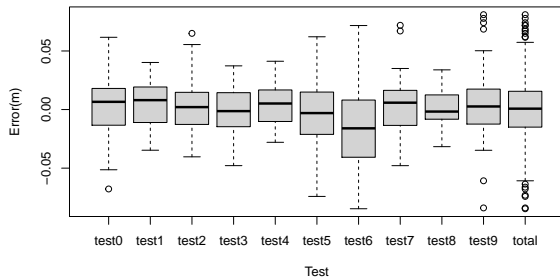
Fig. 11 Position error for the grid formation. The aggregate formation error is shown in the rightmost column.
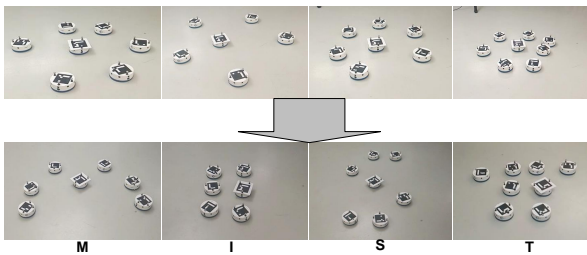
Fig. 12 Experiment results of the "MIST" formation. Robots are tasked to form four letters ("M", "I", "S" and "T") from random initial positions. The first row shows the initial positions of robots, while the second shows the final formations.

dation, a mathematical model, and sufficient conditions for convergence.

The proposed algorithm was verified with extensive simulations and with experiments on real robots. Our simulations study the effects of swarm size (up to hundreds of robots), available robots, and the density of the initial robot distribution. To validate our simulated experiments with real robots, we have implemented the complete algorithm using the Buzz language and tested it on a set of 10 Khepera IV robots. The results indicate that our algorithm is robust to noise and that it can handle different formations and shapes. The overall placement error for our algorithm is below 5%.

We plan to apply this algorithm to practical applications, such as the deployment of sensor networks. For instance, we are currently porting the algorithm to a fleet of heterogeneous unmanned vehicles, for which graphs can ease the zone coverage tasks for mapping and to maintain network connectivity. Further research includes considering robots that need to be in constant motion, and also includes additional theoretical work on progressive deployment, such as considering adaptive task allocation for heterogeneous swarms.

References

Alonso-Mora J, Breitenmoser A, Rufli M, Siegwart R, Beardsley P (2011) Multi-robot system for artistic pattern formation. In: IEEE International Conference on Robotics and Automation (ICRA), pp 4512–4517

Anand A, Nithya M, Sudarshan T (2014) Coordination of mobile robots with master-slave architecture for a service application. In: IEEE International Conference on Contemporary Computing and Informatics (IC3I), pp 539–543

Beal J (2011) Functional blueprints: An approach to modularity in grown systems. Swarm Intelligence 5(3):257–281

Belta C, Kumar V (2002) Trajectory design for formations of robots by kinetic energy shaping. In: IEEE International Conference on Robotics and Automation (ICRA), vol 3, pp 2593–2598

Bonani M, Longchamp V, Magnenat S, Rétornaz P, Burnier D, Roulet G, Vaussard F, Bleuler H, Mondada F (2010) The marXbot, a miniature mobile
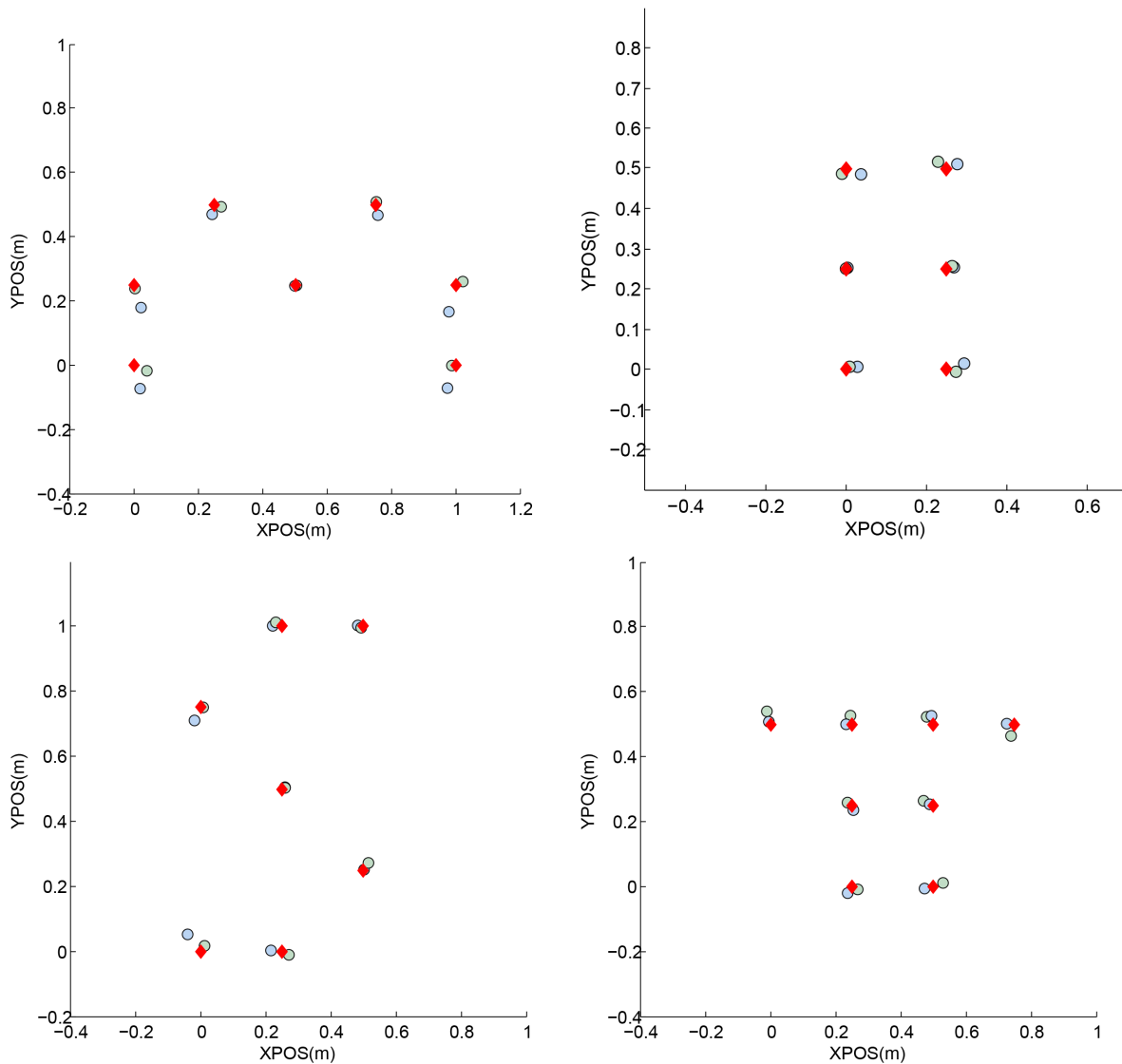
Fig. 13 Final positions of robots in the experiments for the MIST letters formation, with each letter repeated twice. The red diamonds show the desired formation, while the round markers with the same color show the final positions of robots from the same test.

robot opening new perspectives for the collective-robotic research. In: IEEE International Conference on Intelligent Robots and Systems (IROS), pp 4187–4193

Brambilla M, Ferrante E, Birattari M, Dorigo M (2013) Swarm robotics: a review from the swarm engineering perspective. Swarm Intelligence 7(1):1–41

Cheah CC, Hou SP, Slotine JJE (2009) Region-based shape control for a swarm of robots. Automatica 45(10):2406–2411

Chen Z, Chu T (2013) Multi-agent system model with mixed coupling topologies for pattern formation and formation splitting. Mathematical and Computer Modelling of Dynamical Systems 19(4):388–400

Cowley A, Taylor CJ (2007) Orchestrating concurrency in robot swarms. In: IEEE International Conference on Intelligent Robots and Systems (IROS), pp 945–950

Desai JP, Ostrowski JP, Kumar V (2001) Modeling and control of formations of nonholonomic mobile robots. IEEE Transactions on Robotics and Automation 17(6):905–908

Dieudonné Y, Petit F (2007) Deterministic leader election in anonymous sensor networks without common coordinated system. In: International Conference on Principles of Distributed Systems (ICPDS), pp 132–142
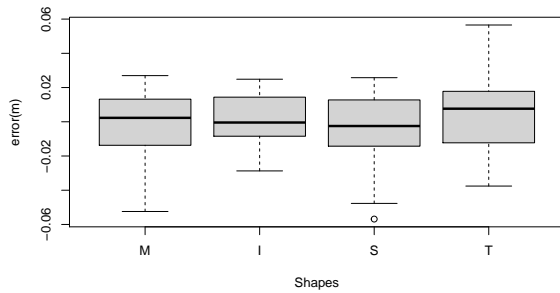
Fig. 14 Formation error from the MIST experiment

Fierro R, Belta C, Desai JP, Kumar V (2001a) On controlling aircraft formations. In: IEEE Conference on Decision and Control, vol 2, pp 1065–1070

Fierro R, Das AK, Kumar V, Ostrowski JP (2001b) Hybrid control of formations of robots. In: IEEE International Conference on Robotics and Automation (ICRA), vol 1, pp 157–162

Güzel MS, Gezer EC, Ajabshir VB, Bostancı E (2017) An adaptive pattern formation approach for swarm robots. In: IEEE International Conference on Electrical and Electronic Engineering (ICEEE), pp 194–198

Hsieh A, Kumar V (2006) Pattern generation with multiple robots. In: IEEE International Conference on Robotics and Automation, pp 2442–2447

Hsieh MA, Kumar V, Chaimowicz L (2008) Decentralized controllers for shape generation with robotic swarms. Robotica 26(5):691–701

Karpov V, Karpova I (2015) Leader election algorithms for static swarms. Biologically Inspired Cognitive Architectures 12:54–64

Li G, Sogor I, Beltrame G (2017) Self-adaptive pattern formation with battery-powered robot swarms. In: NASA/ESA Adaptive Hardware and Systems Conference (AHS)

Liu L, Shell DA (2014) Multi-robot formation morphing through a graph matching problem. In: International Symposium on Distributed Autonomous Robotic Systems (DARS), pp 291–306

Majid M, Arshad M (2015) Hydrodynamic effect on V-shape pattern formation of swarm autonomous surface vehicles (ASVs). Procedia Computer Science 76:186–191

Michael N, Fink J, Kumar V (2007) Controlling a team of ground robots via an aerial robot. In: IEEE International Conference on Intelligent Robots and Systems (IROS), pp 965–970

Michael N, Fink J, Kumar V (2008a) Controlling ensembles of robots via a supervisory aerial robot. Advanced Robotics 22(12):1361–1377

Michael N, Zavlanos MM, Kumar V, Pappas GJ (2008b) Distributed multi-robot task assignment and formation control. In: IEEE International Conference on Robotics and Automation (ICRA), pp 128–133

Mondada F, Bonani M, Raemy X, Pugh J, Cianci C, Klaptocz A, Zufferey JC, Floreano D, Martinoli A (2006) The e-puck: a robot designed for education in engineering. In: Conference on Autonomous Robot Systems and Competitions (Robotica), vol 1, pp 59–65

Paley DA, Leonard NE, Sepulchre R (2008) Stabilization of symmetric formations to motion around convex loops. Systems & Control Letters 57(3):209–215

Petit F (2009) Tutorial 1-3: Leader election and pattern formation in swarms of deterministic robots. In: International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT)

Pinciroli C, Beltrame G (2016) Buzz: An extensible programming language for heterogeneous swarm robotics. In: IEEE International Conference on Intelligent Robots and Systems (IROS), pp 3794–3800

Pinciroli C, Trianni V, O'Grady R, Pini G, Brutschy A, Brambilla M, Mathews N, Ferrante E, Di Caro G, Ducatelle F, Birattari M, Gambardella LM, Dorigo M (2012) ARGoS: a modular, parallel, multi-engine simulator for multi-robot systems. Swarm Intelligence 6(4):271–295

Pinciroli C, Gasparri A, Garone E, Beltrame G (2016) Decentralized progressive shape formation with robot swarms. In: International Symposium on Distributed Autonomous Robotic Systems (DARS), pp 433–445

Ravichandran R, Gordon G, Goldstein S (2007) A scalable distributed algorithm for shape transformation in multi-robot systems. In: International Conference on Intelligent Robots and Systems (IROS), pp 4188–4193

Rubenstein M, Shen WM (2008) A scalable and distributed model for self-organization and self-healing. In: International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS), pp 1179–1182

Rubenstein M, Cornejo A, Nagpal R (2014) Programmable self-assembly in a thousand-robot swarm. Science 345(6198):795–799

Seibert P, Suarez R (1990) Global stabilization of nonlinear cascade systems. Systems & control letters 14(4):347–352

Sepulchre R, Paley DA, Leonard NE (2008) Stabilization of planar collective motion with limited communication. IEEE Transactions on Automatic Control 53(3):706–719

Spears WM, Spears DF, Hamann JC, Heil R (2004) Distributed, Physics-Based Control of Swarms of Vehicles. Autonomous Robots 17(2/3):137–162

Spletzer J, Fierro R (2005) Optimal positioning strategies for shape changes in robot teams. In: IEEE International Conference on Robotics and Automation, pp 742–747

Støy K (2001) Using situated communication in distributed autonomous mobile robots. In: Scandinavian Conference on Artificial Intelligence (SCAI), pp 44–52

Tanner HG, Kumar V, Pappas GJ (2002) The effect of feedback and feedforward on formation iss. In: IEEE International Conference on Robotics and Automation (ICRA), vol 4, pp 3448–3453

Turpin M, Michael N, Kumar V (2012a) Decentralized formation control with variable shapes for aerial robots. In: IEEE International Conference on Robotics and Automation (ICRA), pp 23–30

Turpin M, Michael N, Kumar V (2012b) Trajectory design and control for aggressive formation flight with quadrotors. Autonomous Robots 33(1-2):143–156

Turpin M, Michael N, Kumar V (2013) Trajectory planning and assignment in multirobot systems. In: Frazzoli E, Lozano-Perez T, Roy N, Rus D (eds) Algorithmic Foundations of Robotics X, Springer, pp 175–190

Yang H, Zhang F (2010) Geometric formation control for autonomous underwater vehicles. In: IEEE International Conference on Robotics and Automation (ICRA), pp 4288–4293

Yu CH, Nagpal R (2008) Sensing-based shape formation on modular multi-robot systems: A theoretical study. In: International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS), pp 71–78

Zhang F (2007) Cooperative shape control of particle formations. In: IEEE Conference on Decision and Control, pp 2516–2521

Zhang F, Haq S (2008) Boundary following by robot formations without GPS. In: IEEE International Conference on Robotics and Automation, pp 152–157

Zhang F, Leonard NE (2006) Coordinated patterns on smooth curves. In: IEEE International Conference on Networking, Sensing and Control (ICNSC), pp 434–439

Zhang F, Fratantoni DM, Paley DA, Lund JM, Leonard NE (2007) Control of coordinated patterns for ocean sampling. International Journal of Control 80(7):1186–1199