Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 151 (2019) 431–438

Procedia Computer Science

www.elsevier.com/locate/procedia

The 10th International Conference on Ambient Systems, Networks and Technologies (ANT)
April 29 - May 2, 2019, Leuven, Belgium

# Communication-free and Index-free Distributed Formation Control Algorithm for Multi-robot Systems

J. Peña Queralta[a], C. Mccord[a], T. N. Gia[a], H. Tenhunen[b], T. Westerlund[a]

[a]*Department of Future Technologies, University of Turku, Turku, Finland*
[b]*Deptarment of Electronics, KTH Royal Institute of Technology Stockholm, Sweden*

**Abstract**

Pattern formation algorithms for swarms of robots can find applications in many fields from surveillance and monitoring to rescue missions in post-disaster scenarios. Complex formation configurations can be of interest to be the central element in an exhibition or maximize surface coverage for surveillance of a specific area. Existing algorithms that enable complex configurations usually require a centralized control, a communication protocol among the swarm in order to achieve consensus, or predefined instructions for individual agents. Nonetheless, trivial shapes such as flocks can be accomplished with low sensing and interaction requirements. We propose a pattern formation algorithm that enables a variety of shape configurations with a distributed, communication-free and index-free implementation with collision avoidance. Our algorithm is based on a formation definition that does not require indexing of the agents. We show the potential of the algorithm by simulating the formation of non-trivial shapes such as a wedge and a T-shaped configuration. We compare the performance of the algorithm for single and double integrator models for the dynamics of the agents. Finally, we run a preliminary test of our algorithm by implementing it with a group of small cars equipped with a Lidar for sensing and orientation calculation.



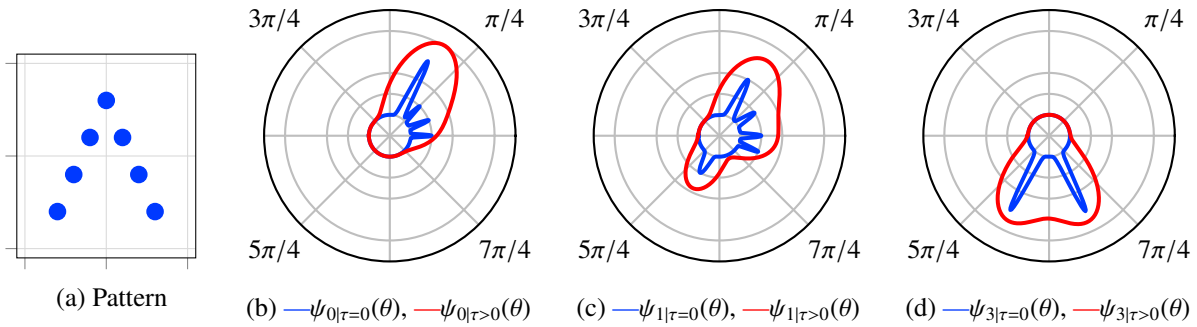
*Keywords:* Formation Control; Pattern Configuration; Coordination; Index-Free Control; Communication-Free; Swarm Robotics; Multi-Agent Systems; Multi-Robot Systems;

## 1. Introduction

Pattern configuration and formation control algorithms for swarms of robots have been extensively studied for the past decades [15, 9, 1, 2, 8, 4, 6, 7, 5]. We can classify pattern formation algorithms among those that require agent indexing or those in which agents are anonymous. Most of the work to date in formation control algorithms for multi-agent systems lies in the former category [3, 7, 14]. Nonetheless, in a swarm of robots where all units are indistinguishable, a more natural approach is to assume a homogeneous set of agents without identity.

*E-mail address:* jopequ@utu.fi

1877-0509 
10.1016/j.procs.2019.04.059




(a) Pattern (b) —$\psi_{0|\tau=0}(\theta)$, —$\psi_{0|\tau>0}(\theta)$ (c) —$\psi_{1|\tau=0}(\theta)$, —$\psi_{1|\tau>0}(\theta)$ (d) —$\psi_{3|\tau=0}(\theta)$, —$\psi_{3|\tau>0}(\theta)$

Fig. 1. Illustration of the spherical indicator distribution. Figure (a) shows a wedge configuration of 7 agents. Figures (b), (c) and (d) show the spherical indicator distribution of the first, second and fourth (top) agents, if indexed from left to right, for $\tau = 0$ and $\tau > 0$.

Many efforts have been devoted to the study and replication of collective motion in nature, such as birds in a wedge formation or flocks of fishes. In these cases, agents are anonymous and homogeneous, and therefore interchangeable. Algorithms that model this behavior may use indexing for formulation but are not affected by an indexing permutation as all agents are considered equal. To the extent of our knowledge, algorithms that rely on anonymous definitions and do not require communication can only be applied in formations of trivial patterns such as flocks or regular polygons, where all positions are equivalent. In the case of flocks, the algorithms are usually based on distance-only measurements [5], while bearing-only measurements can be used to achieve regular polygon configurations [9, 1, 2]. A limitation of these algorithms is that the set of possible formations is restricted to those configurations in which all agent positions are equal. More complex configurations are possible with index-free control if communication is used throughout the system to ensure consensus [15]. Index-free refers to the fact that agents are anonymous and that a permutation of the indexes does not affect the calculation of the control inputs. However, wireless communications between robots used in these algorithms can cause a large overhead of latency and energy consumption. When the number of agents increases, this becomes a serious issue. Some research has been focused on generalizing index-free control by designing algorithms that are invariable to indexing permutations [16, 11, 10]. However, these often use an equivalence relation, or are significantly affected by the non-scalable size of the permutation space with respect to the number of agents in the configuration[13].

Distance-only measurements or bearing-only measurements also appear in the literature within algorithms that are able to achieve more complex formation configurations where the positions are not equivalent. For example, bearing-only measurements are used in [12] to allow non-trivial pattern formation. Alternatively, distance-only measurements are equally employed by [6, 14].

In this paper, we propose a new approach for defining a position within a formation based on the definition of a spatial distribution that takes into account the position of several agents at once. This approach helps to achieve complex formation configurations without agent indexing and without communication, given a favorable initial spatial distribution of the agents. The proposed approach is demonstrated with both simulations for two different pattern configurations and actual experiments with real robots. The algorithm is simulated using single and double integrator dynamics models in order to perform an initial analysis of its potential. The performance between both dynamics models is compared.

The paper is organized as follows: Section 2 introduces the definitions and key features in which the proposed algorithm relies. Section 3 develops the main formulas for different control inputs that enable an actual implementation of the algorithm. Section 4 covers numerical analysis that demonstrates the effectiveness of the proposed methods. Section 5 covers a preliminary experiment run with real robots. Section 6 concludes the work and discusses the future research direction.

## 2. Formation Definition

Formation control algorithms often define formation positions in a way that are directly related to the variables sensed by agents. Some widely used formation control algorithms are position-based, distance-based and bearing-

based algorithms [7]. Position-based formation control algorithms define the positions in the formation with a set of points in space, which are assigned to individual agents. Distance-based formation algorithms (e.g. algorithms for achieving flocking formation) define a position in the formation by distances which are calculated by the agent and its neighbors. Similarly, bearing-based formation control algorithms define a position by the set of bearings, given a common orientation for the measurement. In this paper, the proposed formation control algorithm is a distributed and index-free pattern formation algorithm considering the position of neighbor agents simultaneously.

Comparing to index-based formation control algorithms, an index-free algorithm has many advantages. For instance, it is challenging or even impossible for index-based robots to form an accurate formation when one of the robots cannot get into the predefined position due to a hardware failure or other errors. This issue of an inaccurate formation can be avoided with an appropriate index-free formation control algorithm as agents are interchangeable. Furthermore, the proposed algorithm is communication-free, which helps to avoid the overhead of large energy consumption and latency caused by wireless communicating between agents.

### 2.1. Spherical distribution

Our objective is to design an algorithm that requires, at most, position measurements, and provide a definition for a formation configuration that is independent of the agent indexing while allowing almost arbitrary patterns. Therefore, we introduce a new definition for each of the positions in the formation.

**Definition 2.1** (Spherical Indicator Distribution)**.** *Given a set of N agents with positions* $(x_i, y_i, z_i) \in \mathbb{R}^3\ \forall i = 1, \ldots, N$*, let* $N_i$ *be the set of agent i's neighbors. Their positions are measured within agent i's local reference frame and represented by* $(x^i_j, y^i_j, z^i_j) \equiv (r^i_j, \theta^j_i, \varphi^j_i)\ \forall j \in N_i$ *in Cartesian coordinates or spherical coordinates, respectively. Then we define agent i's spherical indicator distribution by*

$$\psi_i(\theta, \varphi) = \alpha \sum_{j \in N_i} w(r^i_j)(\theta, \varphi) * \delta(\theta - \theta^i_j, \varphi - \varphi^i_j) \tag{1}$$

*where* $\delta(\theta, \varphi)$ *is the Dirac delta function defined in in the subset* $[0, \pi) \times [0, 2\pi) \subset \mathbb{R}^2$*,* $\alpha > 0$ *is a constant and* $w(r)(\theta, \varphi)$ *is a function of the type* $w(r)(\theta, \varphi) = \exp\left(-(\theta, \varphi)^T Q(r)(\theta, \varphi)\right)$ *for a positive definite matrix* $Q(r) > 0$*.*

The definition of the spherical distribution in (1) is inspired by an agent's view of its environment. The Dirac delta marks the two-dimensional bearing of each of the neighbors while $Q(r)$ shapes the space around those points as a function of the distance. We have defined an agent's spherical distribution over $(\theta, \phi)$ only and not over $r$ as well because the bearing coordinates are defined over a compact subset of $\mathbb{R}^2$. Thus, they can be represented by two-dimensional matrices if we consider a discretization of $\mathbb{R}^2$. From a computational point of view, this means a finite and fixed memory allocation. In particular, for our algorithm formulation, we define $Q(r)$ in Definition 2.1 as

$$Q(r) = \beta \begin{pmatrix} r^2 & 0 \\ 0 & r^2 \end{pmatrix} + \begin{pmatrix} \tau & 0 \\ 0 & \tau \end{pmatrix} \tag{2}$$

for constants $\beta, \tau \in \mathbb{R}$, $\beta, \tau > 0$. The value of $\beta$ is the same for all agents and governs the variable width of $w(r)$ along $(\theta, \varphi)$, while $\tau$ adds a constant minimum width. We can think of $1/(\beta r^2)$ and $1/\tau$ as variances of two convoluted Gaussian distributions. We extend on the significance of $\tau$ later in this document. Then we can expand an agent's spherical distribution (1) into (3). illustrated in Figure 1,

$$\psi_i(\theta, \varphi) = \alpha \sum_{j \in N_i} \exp\left(-\left(\beta {r^i_j}^2 + \tau\right) \left\| \left(\theta - \theta^i_j, \varphi - \varphi^i_j\right) \right\|^2\right) \tag{3}$$

### 2.2. Autonomous Position Assignment

The first step in a formation control problem is the position assignment. In displacement-based control this is predefined beforehand, whereas in anonymous distance or bearing-based control this step is skipped due to all positions being equivalent, for instance. In our algorithm, agents perform autonomous self-position assignment as described by Definition 2.2.

**Definition 2.2** (Position objective)**.** *Given a desired position $\psi_j^*$ and an agent in a position $\psi_i$, we define the cost of achieving the position $j$ for agent $i$ as*

$$D_{\psi_i}(\psi_j^*) = \int_0^{2\pi} \int_0^{\pi} \|\psi_i(\theta,\varphi) - \psi_j^*(\theta,\varphi)\|^2 \, d\theta d\varphi \tag{4}$$

*and, therefore, an agent $i$ decides its objective position by minimizing*

$$\psi_i^{obj} = \psi_j^* : j = \underset{j=1,\dots,N}{\operatorname{argmin}} D_{\psi_i}(\psi_j^*), \;\; D_{\psi_i}^{obj} := D_{\psi_i}(\psi_i^{obj}) \tag{5}$$

*from where we define the cost of achieving the objective position*

In order to achieve the objective position, we give the following problem formulation. Given a formation shape defined by a set of $N$ positions $\{x_i\}_{i=1,\dots,N}$, and their corresponding spherical indicator distributions $\{\psi_i(\theta,\varphi)\}$, then

**Definition 2.3** (Formation objective)**.** *Given a pattern configuration defined by a set of $N$ positions, from which we can calculate their individual spherical distributions, the desired formation shape of a group of agents is represented by the set $\mathbb{E}_\psi = \{\psi_1^*, \dots, \psi_N^*\}$ of desired spherical indicator distributions. Therefore, the formation objective is to have a permutation $\sigma \in S_N$ such that $D_{\psi_i}(\psi_{\sigma(i)}^*) < \delta$ for a constant $\delta \in \mathbb{R}$, $\delta > 0$. The value of $\delta$ is the minimum error allowed to assume a successful convergence to the desired position. The set $\{\psi_i(\theta,\varphi)\}$ represents the agents' positions and $\{\psi_j^*(\theta,\varphi)\}$ the desired positions.*

The definition for the position objective in (2.2) does not ensure, in its current form, that there exists a permutation $\sigma$ that maps the set of agents into the set of formation positions. This paper presents a preliminary exploration of the potential of the position definitions introduced in this section. The results included in this document involving simulations and tests are obtained under the assumption that the initial distribution of agents in space ensures the existence of $\sigma$.

After the autonomous self-position assignment is made by each of the agents, the next step is to minimize some cost function in order to arrive to the convergence conditions given in Definition 2.3. When minimizing the cost of achieving the objective position introduced in (5), however, a problem arises from the nature of the definition we are using for $\psi_i(\theta,\varphi)$ in (3). If the value of $\tau$ is too small, as happens in the example illustrated in Figure 1, then the minimization problem can be non-convex. This might happen because $\psi_i(\theta,\varphi)$ is a function formed up by individual peaks with almost-zero intervals in between. Local minima exist when some of the peaks in an agent's spherical distribution and its objective position's spherical distribution are overlapped but others are not. Therefore, depending on the desired pattern configuration, the value of $\tau$ must be adjusted to prevent this from happening. We have developed a graphical interface for simulation purposes where we can adjust the parameters of the algorithm in real-time and during a simulation to see how this affects the position definitions as well as the algorithm performance.

### *2.3. Collision avoidance*

So far, our algorithm does not take into account collision avoidance; for example, in the limits $r_j \to 0$ and $r_j \to \infty$, the contribution of an agent $j$ to $\psi_i$ is reduced to a constant over $\theta$ or a Dirac delta function in $\theta_j$, respectively. In the limit $r_j \to 0$, this might lead to collisions between agents in the case of distributions in which the angular positions of neighbors are densely distributed across all $\theta$. In that case, $\psi_i$ would be an almost-constant function, and the error incurred when inter-agent is reduced would be negligible.

In order to introduce collision avoidance to our algorithm, we consider the following modification of the spherical indicator distribution for a position. Suppose an agent is considered to move safely when its distance with respect to all other agents is bigger than a threshold $r_s$. An agent is considered to be in collision danger if its distance with respect to a neighbor is below a threshold $r_d$. Then we can modify (1) into

$$\psi_i(\theta,\varphi) = \sum_{j\in N_i} \alpha_j \exp\left(-\left(\beta {r_j^i}^2 + \tau\right)\left\|\left(\theta-\theta_j^i, \varphi-\varphi_j^i\right)\right\|^2\right), \quad \alpha_j = \begin{cases} \alpha & if \;\; r_s < r_j \\ r_j/(r_j - r_d) & if \;\; r_d < r_j \le r_s \\ \infty & if \;\; r_d \le r_j \end{cases} \tag{6}$$

where we have introduced the constant $\alpha$ inside the sum and now varies for each of the agents, which always ensures that $r_j > r_d$ and therefore agents are safe from collisions.

## 3. Control Inputs

In this section, we introduce the control laws that define our algorithm for both a single and double integrator dynamics model for the agents. For simplicity, we reduce the problem from here on to the two dimensional case. Therefore, an agent's spherical distribution is given now in polar variables $\psi_i(\theta)$.

First, we suppose that the dynamics of each of the agents in the swarm follow a single-integrator model given by $p_i[k+1] = p_i[k] + T_s u_i[k]$ where $p_i[k], u_i[k] \in \mathbb{R}^2$ denote the position and velocity, respectively, of agent $i$, at the time step $k$, and $T_s$ is the sampling period. We use radial and angular control inputs $u_i[k] \equiv (u_{i_r}[k], u_{i_\theta}[k])$ that represent the speed and direction of movement, respectively. The polar input is well defined because all agents share a global orientation. Now we can calculate $r_j^i[k+1$ and $\theta_j^i[k+1]$ based on the control input for agent $i$, using basic trigonometry, and define a cost function.

$$r_j^i[k+1]^2 = r_j^i[k]^2 + T_s^2 u_{i_r}[k]^2 - 2T_s r_j^i[k] u_{i_r}[k] \cos\left(\theta_j^i[k] - u_{i_\theta}\right) \tag{7}$$

$$\theta_j^i[k+1] = \arccos\left(\frac{r_j^i[k]\cos\left(\theta_j^i[k]\right) - T_s u_{i_r}[k]\cos\left(u_{i_\theta}[k]\right)}{r_j^i[k+1]}\right) \tag{8}$$

**Definition 3.1** (Cost function)**.** *Given an agent with dynamics described by a single integrator, its position represented by $\psi_i(\theta)$, and its position objective by $\psi_i^{obj}(\theta)$, then we define its cost function at a time step k by*

$$J_i[k] = \int_0^{2\pi} \left(\psi_i - \psi_i^{obj}\right)^2 d\theta + \gamma \|u_i\|^2 = D_{\psi_i}^{obj}[k] + \gamma u_{i_r}[k] \tag{9}$$

*where $\gamma > 0$ controls the weight of the closed loop control.*

From Definition 3.1 we propose a control law that minimizes (9). First, we use numerical methods to calculate the angular control input, obtaining $\psi_i\left[k+1 \mid u_{i_\theta}[k] = \widetilde{\theta}\right]$ from (7) and (8). Regarding the control input $u_{i_r}$, we propose a control law based on the instantaneous position error.

$$u_{i_\theta}[k] = \widetilde{\theta} : \widetilde{\theta} = \underset{\widetilde{\theta}\in[0,2\pi)}{\operatorname{argmin}} \int_0^{2\pi} \left(\psi_i\left[\mathrm{k}+1 \mid u_{i_\theta}[k] = \widetilde{\theta}\right] - \psi_i^{obj}[k]\right)^2 d\theta \tag{10}$$

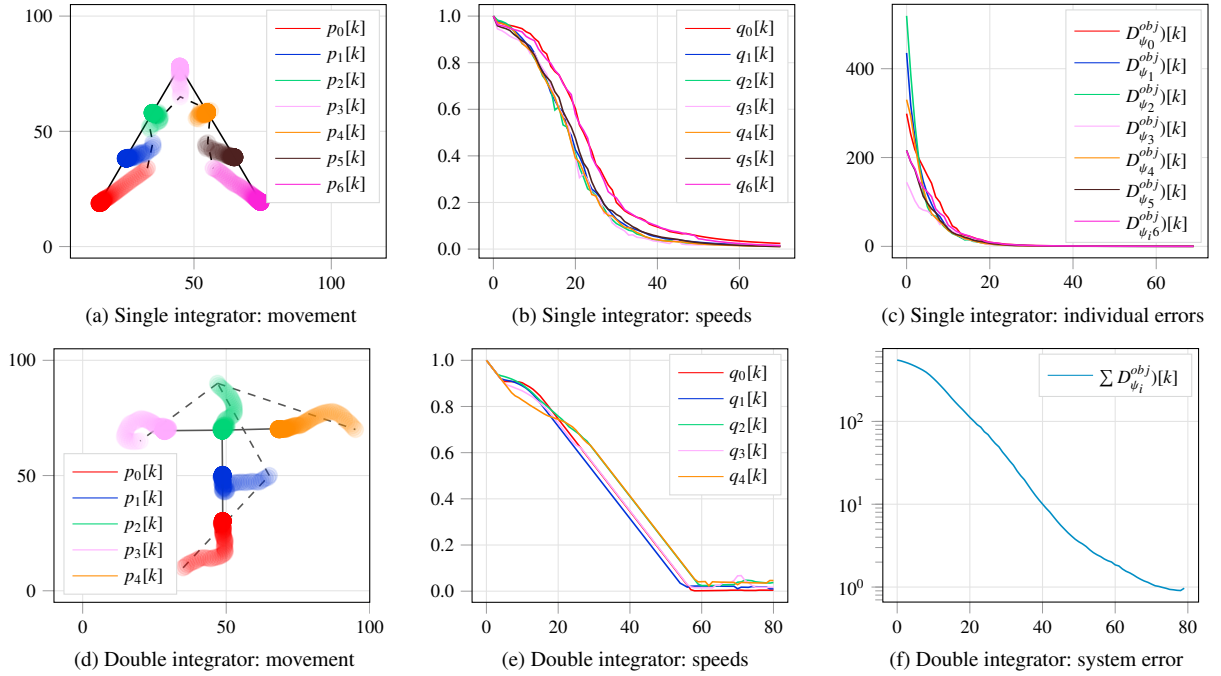
$$u_{i_r}[k] = \nu \frac{D_{\psi_i}^{obj}}{10\delta + D_{\psi_i}^{obj}} \tag{11}$$

where $\delta$ is the maximum error allowed for each position to assume that the system has converged to its formation objective, as defined in the problem formulation in Definition 2.3. The factor 10 introduced in the equation is motivated by a preferred faster convergence of $u_{i_r}^2$ to 0, with respect to the position error term of the cost function (9).

If we suppose that the dynamics of each of the agents follow a discrete double-integrator model, then the position of a agent at the next time step is given by $p_i[k+1] = p_i[k] + T_s q_i[k]$, $q_i[k+1] = q_i[k] + T_s u_i[k]$, where $p_i[k]$, $q_i[k]$ and $u_i[k]$ are agent $i$'s position, velocity and acceleration (control input), respectively. Then, we adapt the cost function (9) to take into account the agent's velocity as $J_i[k] = D_{\psi_i}^{obj}[k] + \gamma_r u_{i_r}[k] + \gamma_\theta \left|u_{i_\theta}[k]\right|$. Now $u_{i_r}$ and $u_{i_\theta}$ represent the radial and angular acceleration, respectively. We take into account the radial acceleration to avoid rotations. In this case, instead of defining the control inputs at each time step by a closed formula, we adapt the desired angular speed from (10) into

$$q_{i_\theta}[k] = \widetilde{\theta} : \widetilde{\theta} = \underset{\widetilde{\theta}\in\left[q_{i_\theta} - u_{i_\theta}^{max}, q_{i_\theta} + u_{i_\theta}^{max}\right)}{\operatorname{argmin}} \int_0^{2\pi} \left(\psi_i\left[k+1 \mid u_{i_\theta}[k] = \widetilde{\theta}\right] - \psi_i^{obj}[k]\right)^2 d\theta \tag{12}$$

where we have only changed the minimization interval by adding the maximum allowed radial acceleration and taking into account that the operations $u_{i_\theta} - u_{i_\theta}^{max}$ and $u_{i_\theta} + u_{i_\theta}^{max}$ are modulus $2\pi$. Equivalently, we calculate the ideal radial speed using (11) and then limit the radial acceleration. Then the control inputs are simply $u_{i_\theta} = q_{i_\theta}[k] - q_{i_\theta}[k-1]$ and $u_{i_r} = q_{i_r}[k] - q_{i_r}[k-1]$.




(a) Single integrator: movement (b) Single integrator: speeds (c) Single integrator: individual errors

(d) Double integrator: movement (e) Double integrator: speeds (f) Double integrator: system error

Fig. 2. A wedge configuration of 7 agents is achieved in (a), where the movement of the agents is displayed with increasing opacity over time. Graphs in (b) and (c) show the evolution of each of the agents speed and position error. A single integrator model is used for the agents dynamics. In the second row, a T-shaped configuration of 5 agents is achieved, using double integrator models. Graphs in (e) and (f) now show the speed of the agents and the system error over time, respectively.

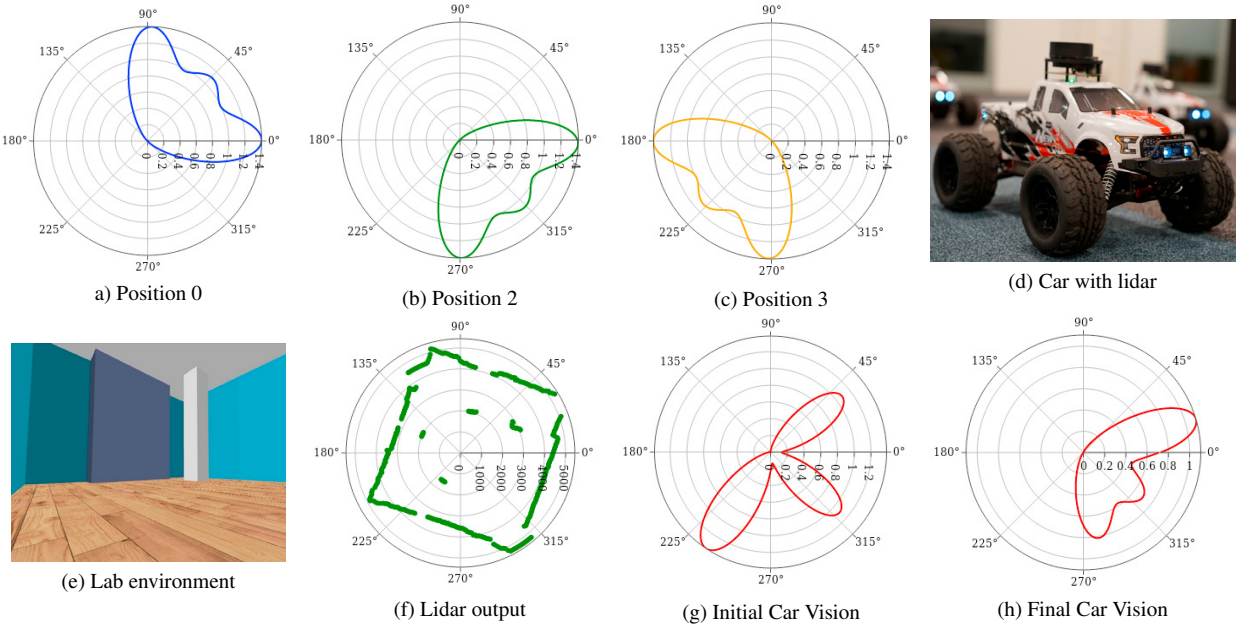
## 4. Numerical Analysis

In this section, we show the computational simulations results that we have obtained after implementing our algorithm for non-trivial T-shaped and wedge formation configurations. The algorithm has been implemented using Python. The implementation models the robots as single points in space.

Figure 2 shows the result of two simulations. The wedge formation is achieved using a single integrator model for the dynamics of the agents, while the T-shaped formation is achieved with a double integrator dynamics. We have tested both single and double integrator models for those two and other formation configurations, including a square, a square with a fifth agent in the center, and a diamond. We have found little difference in the performance of the algorithm under the single and double integrator models. The main difference can be seen in Figure 2, graphs (b) and (e), where we can see that there is an acceleration limit.

We set the maximum speed to $v = 1$, and the maximum position error allowed to assume convergence is set to $\delta = 1$. This means that when the position error of an agent is $D^{obj}_{\psi_i} = 1$, then its radial speed is $q_{i_r} = v \cdot D^{obj}_{\psi_i}/(10+D^{obj}_{\psi_i}) \approx 0.11$. We can see that this effectively happens in Figure 2, (e) and (f), around iteration 50, in which the total error of the system is approximately $\sum_i D^{obj}_{\psi_i} \approx 5$ and the speeds are $q_{i_r} \approx 0.1$.

We also partially test the role of the neighborhood set, $N_i$, in the convergence of the algorithm. In the wedge configuration simulation, we assume that agents are able to sense the position of all other agents. In the case of the T-shaped configuration, we introduce a sensing radius, and define each position in the formation based only on the *visible* positions. Throughout the simulation, we impose that Agents 0, 3 and 4 are not able to *see* each other, and the same for Agents 0 and 2. Agent 1 is the only one able to sense the position of all other agents. The definitions that we have given so far do not take into account the possibility that the number of neighbor positions that is used to define a given position might differ from the number of agents that an agent trying to reach that same position is able to sense. This is an important aspect of the algorithm under study and more results will be reported in future work.




a) Position 0 (b) Position 2 (c) Position 3 (d) Car with lidar

(e) Lab environment (f) Lidar output (g) Initial Car Vision (h) Final Car Vision

Fig. 3. Experiment with the MonsterTruck cars. Figures (a) to (c) show the definition of 3 positions in a square configuration (they are all equivalent with respect to a rotation). Figure (d) shows the car used in the experiment and (e) illustrates the testing environment, where the white column is used as an orientation reference. Figure (f) shows the lidar output, with one point for each degree, for a car placed near the center of the room detecting three other cars and the reference column. Figures (g) and (h) show the view of the car, $\psi(\theta)$, at iterations $it = 0$ and $it = 5$, respectively. All the figures, except (d) and (e), are taken from a web application that runs in each of the cars and is used for monitoring.

## 5. Testing in a lab environment

We have made an initial test of the proposed algorithm with real robots. In order to analyze the effectiveness of our algorithm in a real scenario, we use a *1:10 Elektro-Monstertruck "NEW1" BL* model, a radio controlled car where we have replaced the radio receiver by a Raspberry Pi. The motion of the car is controlled via two servo motors, one for the turning direction and one for speed control. The turning direction is limited to the interval $(-0.35rad, 0.35rad)$, of about 40 degrees, 20 in each direction. This is the value of the maximum radial acceleration $u_{i_\theta}^{max}$. While each iteration of the algorithm needs a relatively low execution time, reading and processing the lidar data takes more computing resources. Moreover, we run a web server on the Raspberry Pi in order to have real time monitoring and visualization of the lidar data and agent view, as well as calculated control inputs. As a result, instead of keeping the car moving in a continuous way, we move the car in steps. The car speed given by the control input is translated into the distance that is moved in each step. This can be improved with a more powerful computing platform.

To obtain the position of other cars we use a RPLiDAR A1M8 lidar with a range of 12m that offers a 360-degree view of the car environment. In order to calculate the car orientation, we use a column in the testing environment as a reference. We calculate the instantaneous car orientation supposing that the two walls closer to the column represent north and east directions. In a real scenario, potentially unknown, agents would be equipped with digital magnetometers or other sensors in order to ensure a shared global orientation. However, the settings we use are enough for the purpose of the experiment.

The results of our test are shown in Figure 3. We use 4 cars and choose a square pattern configuration as the formation objective. The first four graphs, (a) to (d) show the definition of the four positions in the square. The lab environment is illustrated in (e), with the white column used as an orientation reference. The column is not used for localization but only for aligning the orientation of local refence frames. We follow the movement of a car placed near the center of the room. The initial position of the car and the other three cars can be inferred from the initial lidar view shown in (f), where the car being monitored is in the center of the graph. The four cars are initially placed in a nearly-triangular configuration where one of the cars is near the center of the other two.

Due to the non-negligible size of the cars compared to the room space, and the fact that they are not small robots that can turn around while keeping their position, the formation can only be achieved to some extent, and the maxi-

mum required error is achieved in just 5 iterations. Graphs (g) and (h) show the initial and final agent view after those 5 iterations, where the car is trying to achieve the position defined in (b). This experiment proves that convergence to the desired configuration is possible in a real scenario without communication or predefined assignments. The angular distribution of the four position definitions makes the role assignment simpler in this case, but more complex configurations have been simulated. Future work will include extensive testing in larger environments and more complex formation configurations. We will also take into account more realistic models for the agent dynamics.

## 6. Conclusion and Future Work

In this paper, we propose an advanced formation control algorithm that enables almost arbitrary shapes to be formed up without the need of communication between the agents and without identifying each of the agents with a unique label. This algorithm is based on an index-free definition for a position within a formation that requires distance and bearing measurements to express the position of a neighbor agent in spherical coordinates. Moreover, the definitions can be adapted to avoid collision between agents.

The algorithm introduced in this paper consists of a set of preliminary ideas and is in an early stage of development. Therefore, further development is required for a robust formulation and implementation. We have assumed the existence of a permutation that ensures a surjective mapping to the set of desired positions. We base this assumption in the use of an initial distribution that naturally produces such surjection. Moreover, the role of the neighborhood set in the convergence has not been studied in depth. We are continuing to study those and other aspects involving the algorithm convergence and results will be shown in future work.

To the extent of our knowledge, other distributed algorithms that are able to achieve wedge configurations, or a T-shaped formation, require either communication among the agents to achieve consensus or agent labeling and a predefined assignment of the positions in the formation to specific agents. The algorithm presented in this paper is inspired by the anonymous nature of a homogeneous system. It only requires measurement of the position of other agents, and a shared global orientation. The former can be obtained through multiple solutions such as lidars or cameras together with computer vision, while the latter can be easily achieved by using a digital compass or similar sensors.

## References

[1] A. N. Bishop, 2011. A very relaxed control law for bearing-only triangular formation control. IFAC Proceedings Volumes 44, 5991 – 5998.
[2] A. N. Bishop *et al.*, 2010. Bearing-only triangular formation control on the plane and the sphere, in: 2010 18th MED.
[3] Barogh, S.A., Werner, H., 2016. Cascaded formation control using angle and distance between agents with orientation control (part 1 and part 2), in: 2016 UKACC 11th International Conference on Control (CONTROL), pp. 1–6.
[4] D. Morgan *et al.*, 2016. Swarm assignment and trajectory optimization using variable-swarm, distributed auction assignment and sequential convex programming. The International Journal of Robotics Research 35, 1261–1285.
[5] G. Vásárhelyi *et al.*, 2014. Outdoor flocking and formation flight with autonomous aerial robots, in: 2014 IEEE/RSJ IROS.
[6] K. K. Oh *et al.*, 2011. Formation control of mobile agents based on inter-agent distance dynamics. Automatica 47.
[7] K. K. Oh *et al.*, 2014. A survey of multi-agent formation control. Automatica 53.
[8] L Smith *et al.*, 2006. Stabilizing a multi-agent system to an equilateral polygon formation .
[9] M Basiri *et al.*, 2010. Distributed control of triangular formations with angle-only constraints. Systems & Control Letters 59, 147 – 154.
[10] M. M. Zavlanos *et al.*, 2007. Distributed formation control with permutation symmetries, in: 2007 46th IEEE CDC.
[11] N. P. Hyun *et al.*, 2016. Collision free and permutation invariant formation control using the root locus principle, in: 2016 ACC.
[12] N. Shiell *et al.*, 2016. A bearing-only pattern formation algorithm for swarm robotics, in: Swarm Intelligence, Springer International Publishing.
[13] P. Kingston *et al.*, 2010. Index-free multi-agent systems: An eulerian approach. IFAC Proceedings Volumes 43, 215 – 220.
[14] Park, M.C., Sun, Z., Trinh, M.H., Anderson, B.D.O., Ahn, H.S., 2016. Distance-based control of k4 formation with almost global convergence, in: 2016 IEEE 55th Conference on Decision and Control (CDC), pp. 904–909.
[15] S. Bandyopadhyay *et al.*, 2017. Probabilistic and distributed control of a large-scale swarm of autonomous agents. IEEE Transactions on Robotics 33, 1103–1123.
[16] S. Kloder *et al.*, 2004. A configuration space for permutation-invariant multi-robot formations, in: 2004 ICRA.