Article

Decentralized Coordination of a Swarm of UAVs for Spatial Planar Shapes Formations

Etienne Petitprez [1,2,4], François Guérin [1,3]*, Frédéric Guinand [1,4,5], Florian Germain [1] and Nicolas Kerthe [1]

[1] Le Havre Normandy University, Le Havre France
[2] Squadrone System Grenoble, France
[3] GREAH Laboratory
[4] LITIS Laboratory
[5] Cardinal Stefan Wyszynski University, Warsaw, Poland
* Correspondence: francois.guerin@univ-lehavre.fr

Abstract: In this paper, a decentralized coordination of a swarm of UAVs is presented to surround planar virtual shapes in 3D. Shapes are designed by Fourier's descriptors from a reference point and can fluctuate over time. We show that UAVs equally distribute themselves angularly on the defined shape. They share position information to avoid collision between them while trying to reach their desired position. Simulations and experiments were conducted to test the swarm control and its performances.

Keywords: Swarm of UAVs, Decentralized coordination, Fourier's descriptor, Formation control, Obstacle avoidance

Citation: Petitprez, E.; Guérin, F.; Guinand, F.; Germain, F.; Kerthe, N. Decentralized UAV Swarm Formation. *Sensors* **2022**, *1*, 0. https://doi.org/

Received:
Accepted:
Published:

1. Introduction

Swarm robotics, inspired by collective behaviors observed in nature, has emerged as a promising concept with numerous applications across various domains [1]. The development of key elements underlying self-organization, namely collaboration, cooperation and robustness, has enabled breakthrough in many challenging and time-consuming tasks [2–4].

Robotic swarms demonstrate their efficiency by sharing tasks among robots. For instance, they can rapidly cover a designated geographic area or perform search and tracking, particularly in the context of rescue missions [5]. Along the same lines, data exchange (including images and positions) can prevent collisions and assist the group in navigating complex environments, as required for inspecting critical infrastructures [6].

Another advantage of swarms is their adaptability and robustness. Thanks to a decentralized approach, facilitated by communication among group members, robots can reorganize themselves in response to dynamic and often unpredictable changes, even when a few robots become inoperative. These properties prevent system failure. This makes swarms desirable for long-lasting tasks, such as environmental monitoring, where some may take over for others due to failure or a lack of energy [7].

Behind these few examples, thousands of scientific papers and books have been published over the last few decades[1] on the topic of swarm robotics. These works range from pure theoretical studies to real experiments conducted in harsh environmental conditions. They have explored both centralized and decentralized methods, as well as hybrid approaches. Researchers have considered a wide array of robotic platforms, including aerial, ground, and underwater robots, and have proposed applications spanning numerous domains. Many surveys have been written in an attempt to classify problems, approaches, methodologies, experimental platforms, and results.

[1] +15000 hits in google scholar for the last decade

The work presented in this paper focuses on swarms of Unmanned Aerial Vehicles (UAVs), also referred to as drones. While many works have been done in this field [8,9], this paper specifically addresses the problem of pattern formation.

In the context of this study, the goal consists in positioning a set of drones evenly along a predetermined planar shape, such as a circle, polygon, butterfly, etc. The term 'evenly' can be interpreted as maintaining equal distances between the drones or, alternatively, arranging them at regular angular intervals from a fixed reference point along the shape. In this work, we choose to explore the latter option. Additionally, unlike some previous works described in [3], the shape itself is defined, prior to the mission, by a human operator and does not result from interactions among the robots.

The original motivation for this work stemmed from an industrial accident that occurred in Normandy in 2019[2]. The Lubrizol petrochemical facctory and some neighborhing warehouses caught fire. The toxic and dense cloud quickly spreads high into the sky, close to the city of Rouen. Due to its size and the proximity of houses, firefighters encountered great difficulty in controlling the fire. Later on, the debrief of this accident sparked discussions about possible future strategies for improving the conditions of actions. One part of the discussion focused on the possibility of deploying a swarm of drones for observation purpose. The developed idea was to enable the swarm to surround the cloud, at a relevant distance, with each drone pointing its camera towards the source of the fire. The current paper presents a decentralized and flexible solution for achieving such a goal. The implementation of such a solution faces several problems.

Information should arrive continuously from the swarm, thus, due to limited battery capacities, the composition of the swarm may change during the mission. This constraint eliminates swarm-based solutions relying on a leader or on any form of centralization, like the one proposed by Raja and his colleagues [10]. Identify the interesting area to observe and define the shape delimiting this area is another issue. According to the situation, environment and physical constraints, areas of interest may be very different, invalidating all solutions targeting a specific set of shapes. In 2021, Huang and his colleagues [11] perform 3D representation with a UAVs fleet using graph theory. For N UAVs, each 3D model is meshed by N nodes, at precise coordinates. The fleet of UAVs moves from one shape to another, when an event transition is triggered, by solving a task assignment problem optimization and planning a collision-free trajectory. An artificial potential field was used to manage the obstacle avoidance in real-time. Even if this work shows a concrete realization of a robust system with 6 UAVs, the model for each desired shape is meshed by a central machine beforehand and the number of drones is fixed.

In order to define a wide variety of shapes delimiting potential areas of interest we propose to use Fourier Descriptors (FDs) [12–14]. FDs coupled with the Fourier Transform (FT) allow the definition of a planar formation shape as a virtual rigid body structure. Methods based on FDs ensure the representation and retrieval of a shape. They derived from a shape signature (in general a 1D function) representing 2D areas or boundaries. Zhang and Lu [15] compared some of them on a set of complex shapes. As a result, the centroid distance has better results in terms of robustness, convergence speed and computation complexity and can handle open curves. The FT approximates the shape signature with a finite number of harmonics to transform an unknown function into a low-computational sum of sinusoidal functions. For copying with reactivity and flexibility constraints of the decentralized network, we decided to use a composition of virtual actions [16–18] to ensure collision avoidance. Each UAV adjusts its velocity according to the presence of static or dynamic obstacles in a defined range.

In the Section 2, the FD and FT formulation to depict the desired formation is described. The decentralized allocation of bearing angle between robots is described and explained in Section 3. Then, the modeling and control is discussed along with the obstacle avoidance

[2] https://www.francetvinfo.fr/faits-divers/incendie/incendie-d-un-site-seveso-a-rouen/ (french tv news)

management in Section 4. Simulation protocols and experiments are presented in Section 5 and results are commented in Section 5.3.

2. Design planar shapes with Fourier descriptors

2.1. Problem statement

We desire to control the swarm/formation regardless the number of UAVs. The swarm/formation should be maintained when the number of UAVs varies over time. For some applications, conditions may vary during the execution of the mission: number of deployed UAVs, target movement, objectives, etc. These variations may also entail a change in the formation (size, position, orientation, etc). It would then be appropriate to find a way of describing the pattern irrespectively of its absolute position, orientation and size.

One way to ensure these statements is to use Fourier Descriptors (FDs) and Transform (FT). This method allows shape description through these modifications while maintaining a formation coherence via three parameters: amplitude, phase and reference point. It also ensures continuity of sampled or discontinuous functions (within limits) avoiding singularity on formation shape.

In our study, we choose to equally distribute UAVs (index i) around the shape with bearing angles (β_i) computed according to the number of UAVs available. The decentralized method enabling the determination and the allocation of these bearing angles between UAVs is described in Section 3. From now on, we consider that each UAV knows the bearing angle to reach, it is implemented in the high level control as a direction instruction.

Only knowing the reference point of the shape, we wanted them to deduce the coordinates of the point to reach (M_i). Therefore, we designed our functions to fit with a real cyclic period of 2π by describing them in polar coordinates from this point. These functions do not require to be continuous, they can be sampled or piecewise continuous. To be able to rotate in 3D the planar shape, we used a rotational quaternion which is given to all UAVs along side the harmonics from FT.

Each UAV has to regulate to zero its distance errors to the point to reach (M_i) of the shape while pointing toward it according to the desired direction β_i (Figure 1).

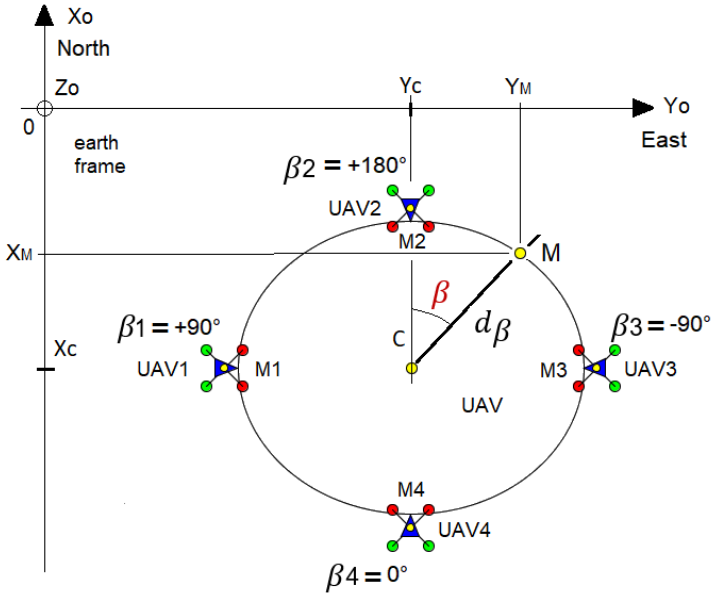

Figure 1. Example of an equidistribution of UAVs around an elliptical shape of center *C*.

2.2. Fourier descriptor

Fourier descriptors [12,13,15] consist in describing a planar function (d_β), defined by the position of a point $M(X_M, Y_M)$ with a single variable β (Figure 2). This one dimensional

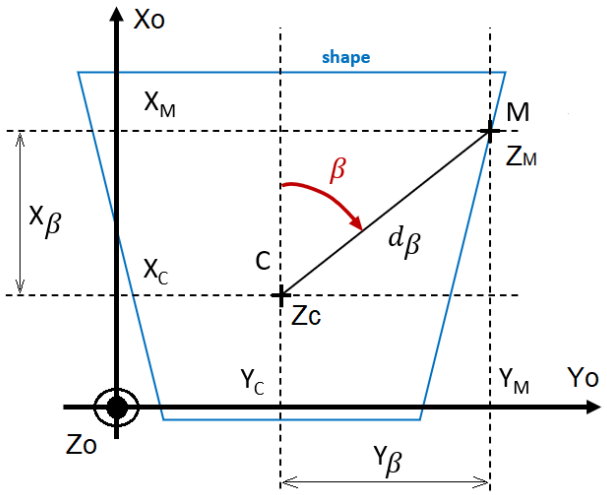

Figure 2. Example of a closed curve defined by d_β with FD from the reference point C.

function d_β is called shape signature and must be periodical.

In our study, we choose the centroid distance d_β to describe the signature shape of the desired curve (Figure 1, Figure 2). It captures and retrieves with the most accuracy complex patterns [15]. Additionally, it matches the polar description we wanted, to retrieve the desired position. It is expressed by the distance of the boundary points (M) from the centroid (C) of the shape. To match our problem statement, the reference point C can be any point as long as one angle gives one possible distance.

A Discrete Fourier Transformation (DFT) is applied to retrieve the complete definition of multiple harmonics (1). The number of harmonics f_{max} depends on the sampling frequency f_e of the continuous function of the shape signature to respect the Nyquist-Shannon condition (2). N denotes the number of samples generated and β_t the associated angle of the sample t, parameter of the signature function.

$$\begin{aligned} a_n &= \frac{1}{N}\sum_{t=1}^{N} d_{\beta_t} \exp\left(\frac{-j2\pi nt}{N}\right) \\ Re(a_n) &= \frac{1}{N}\sum_{t=1}^{N} d_{\beta_t} \cos\left(\frac{2\pi nt}{N}\right) \\ Im(a_n) &= \frac{1}{N}\sum_{t=1}^{N} d_{\beta_t} \sin\left(\frac{-2\pi nt}{N}\right) \end{aligned} \tag{1}$$

$$f_{max} < \frac{f_e}{2} \tag{2}$$

The Fourier coefficients are determined with the DFT, retrieving the distance from the reference point with the bearing angle β :

$$\begin{cases} \phi_n = atan2(Im(a_n), Re(a_n)) \\ X_\beta = \sum_{n=0}^{N+1} \epsilon |a_n| \cos(n\beta + \phi_n) \\ Y_\beta = \sum_{n=0}^{N+1} \epsilon |a_n| \sin(n\beta + \phi_n) \\ d_\beta = \sqrt{X_\beta^2 + Y_\beta^2} \end{cases} \tag{3}$$

ϕ_n : phase of the n^{th} harmonic,
d_β : distance from the reference point C to the point M,
β : angle between X (north) axis and the direction defined by the point M and the reference point C,
X_β : distance along X axis (north) of the point M from the reference point C,

Y_β : distance along Y axis (east) of the point M from the reference point C.

2.3. Spatial rotations

The reached position is at this state determined by the altitude of the reference point Z_c and is purely planar (Figure 2). To bring the third dimension on the planar shape and allow 3D rotations, we defined a quaternion to operate it. It only requires a revolutionary axis coupled with an angle.

Let q the initial quaternion resulting from DF and FT be defined as follows:

$$q = \begin{cases} a_1 = 1 \\ v_1 = \begin{cases} \cos(\beta) \\ \sin(\beta) \\ 0 \end{cases} \end{cases} \tag{4}$$

The rotational quaternion q_{rot} is interpreted from the rotational angle α_{rot}, normalized vector $(x_{rot}, y_{rot}, z_{rot})^T$:

$$q_{rot} = \begin{cases} a_2 = \cos(\alpha_{rot}/2) \\ v_2 = \begin{cases} \sin(\alpha_{rot}/2) * x_{rot} \\ \sin(\alpha_{rot}/2) * y_{rot} \\ \sin(\alpha_{rot}/2) * z_{rot} \end{cases} \end{cases} \tag{5}$$

The new normalized position vector $\underline{X}$, result of the rotation of q around q_{rot}, can be expressed as :

$$\underline{X} = 2 * (v_1 \cdot v_2) * v_2 + a_2^2 - (v_2 \cdot v_2) * v_1 + 2 * a_2 * (v_2 \otimes v_1) \tag{6}$$

One multiplies the normalized position vector $\underline{X}$ by the distance d_β to retrieve the desired position of the point M.

To conclude, the management of the swarm/formation will consist in leading each UAV to the points M defined around the 3D shape (β, d_β) by avoiding possible obstacles and collisions.

Before starting their mission, drones receive two crucial pieces of information: the reference point and the process for computing their position on the shape defined by the Fourier Descriptor. However, to calculate their positions accurately and orient themselves toward the source of the reference point, each drone requires its own bearing angles. Allocating bearing angles to drones as part of their individual parameters before the mission is one approach. Nevertheless, this method presents certain drawbacks. Indeed, angles must be individually transmitted to each drone by a central device assumed to possess all the necessary information. Consequently, each drone should be uniquely identified, and the requirement for each drone to possess a unique identifier runs counter to the core principles of decentralized algorithms. Furthermore, this method faces challenges if the number of drones changes during the mission. Drones cannot adjust their positions accordingly. As a result, if some drones leave the swarm, gaps may form in the shape description. Conversely, when new drones join the swarm, determining their positions becomes a non-trivial task.

Next section will discuss different decentralized algorithms for addressing these issues.

3. Decentralized Allocation of Bearing Angles

In this section, we present algorithms that enable drones to self-determine bearing angles through message exchange. These algorithms are executed asynchronously by each drone. In all cases, drones don't have identifiers making the messages anonymous. Three cases are investigated:

1. case static and reliable communication: the size of the swarm (number of drones) is known by every drone and no message is lost during the exchanges
2. case static and unreliable communication: swarm size is known but some messages may be lost during the communications
3. case dynamic and reliable communication: swarm size is unknown and may change during the mission, some drones may leave the formation while new drones may join the swarm, but, no message is lost.

Whatever the scenario, drones communicate by broadcasting messages and communication topology is a full-connected graph. Thus, no routing is needed. They all own a compass enabling every drone to consider the magnetic North as the reference null angle. Drones have to be equally distributed over the shape, so computing an angle is equivalent, for each of them, to choose a number between 0 and $N-1$, where N is the size of the swarm. These numbers are called *positions*, and two drones cannot have the same position. Note that for case 3 (dynamic-reliable-com), N changes during the mission. In this study, the term *consensus* refers to the situation where each position has been chosen by one and only one drone. Algorithms presented in the sequel are designed such that consensus is reached through a decentralized process.

3.1. Known Constant Swarm Size

For this first case, we assume several conditions fulfilled. We suppose drones communicate by broadcasting messages and that each broadcast message is received by the other drones. We suppose the number of drones initially equals to N and no drone is leaving or joining the swarm during the mission.

The principle lies on the use of an array of boolean values. In this array, cell i contains either $true$ if position i has been chosen by a drone or $false$ if no drone has chosen yet this value. When necessary, drones broadcast both their array of booleans and their chosen position. If two drones have chosen the same position there is a conflict. Thus, upon reception of such a message, the drone compares its position and the received one and in case of conflict, it performs a new random choice of position such that its corresponding cell in the array contains $false$. The method is formally written in Algorithm 1 (see Annex 8).

For measuring the performances of the algorithm, the asynchronous execution of the algorithm by each drone is obtained thanks to a multi-threaded simulation. Each drone is assigned to an independent thread that executes the algorithm. Two scenarios are considered. Message loss is not considered in the first scenario, while a substantial percentage of broadcast messages are lost in the second scenario.

Without message loss, the algorithm is rather efficient since an average number of 2.5 broadcast messages per drone is enough for reaching a consensus, that is, for each drone to choose a number in $[0, N-1]$, such that each number is associated to one and only one drone. Experimental results are reported on Figure 3.

In the second scenario, broadcast messages loss may lead to a deadlock. Indeed, in case of conflict, the two involved drones have to choose another number and broadcast their new choice to the swarm. If one of the two messages is lost, the other members of the swarm receive from the other drone one message with a cell containing "false", and, as the other message is never received the drones may wait forever. For coping with this issue, a timeout mechanism is introduced in the algorithm. If after a given period of time a drone does not receive any message and still have some cells at false within its array, it broadcasts again its positions array and its own position. This new broadcast will trigger the emission of the drone which position is marked false in the received array. So, the condition line 5 of Algorithm 1 becomes: **if** `change` OR `timeout` **then**.

As reported on Figure 4, the number of broadcast messages increases with the percentage of loss. However, the method remains very robust since large loss percentages (up to 50%) do not prevent the algorithm to reach the consensus. Furthermore, the number of

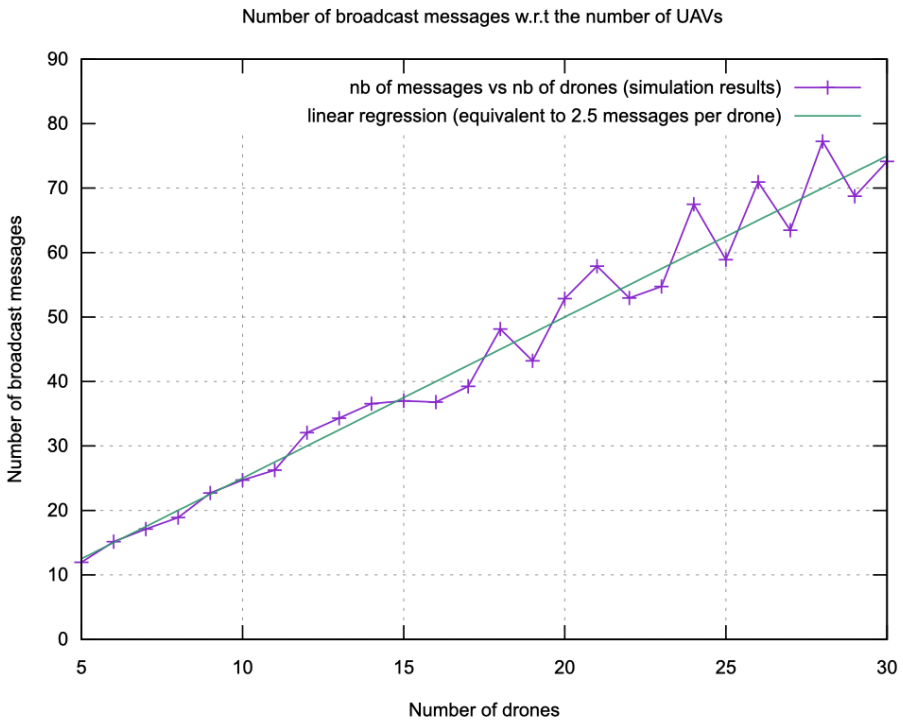

Figure 3. Average total number of broadcast messages by drones for reaching a consensus. On average, 2.5 messages are broadcast by drones. Each point on the graphics corresponds to 100 runs.

additional needed messages for coping with losses grows sub-linearly with the number of drones, making unlikely communications congestion.

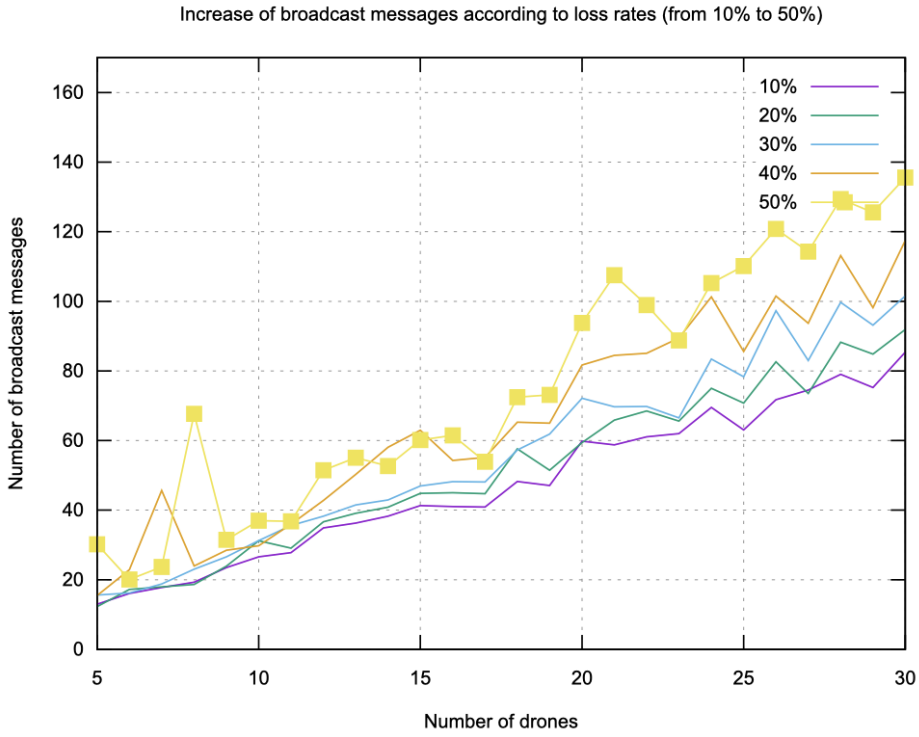

Figure 4. Average Number of Broadcast Messages needed when some messages are lost. Considered loss rates range from 10 to 50%.

3.2. Dynamic Swarm Size

For this last scenario composition of the swarm is dynamic. Two opposite events may occur: a drone joins the swarm, a drone leaves the swarm. We assume several conditions to be true:

- communication is assumed to be reliable: no message is lost,
- between two events (join or leave), there is enough time for the swarm to reach a consensus on drone's respective chosen position,
- the topology of the communication network is a full-connected graph: no routing is needed,
- it is assumed that messages are received in the chronological order of their emission. Specifically, a message sent at time t by drone D_i is expected to be received by all other drones before any message broadcast at a later time $t' > t$ by drone D_j. This assumption seems valid in the context of a fully-connected communication topology,
- drones process their message in First In First Out (FIFO) order.

Because of changes and unlike previous scenario, bearing angle allocation is a never-ending process. The method lies on the exchange of messages. Each message contains three fields: the chosen position by the drone, its array of positions (boolean values), and the type of the message.

My position	$[true, true, false, ..., true]$	MSG_TYPE

Three types of messages are considered:

- JOIN: when a drone wants to join the swarm, it broadcasts a JOIN-type message with its position (0 by default) and an array of only one cell containing *true*,
- LEAVE: when a drone leaves the swarm, it broadcasts a LEAVE-type message with its current position, and the corresponding array with *false* at its own position,
- UPDATE: several situations may lead to the emission of a UPDATE-type message:
  1. when a drone changes its position (caused by a conflict), it broadcasts a new UPDATE-type message with its newly chosen position and the new array with the boolean *false* in the cell of its former position,
  2. when a drone receives a JOIN-type or a LEAVE-type message, it modifies its array of positions according to the situation and broadcasts an UPDATE-type message,
  3. when a drone updates the size of its array after reception of an UPDATE-type message (only for newly arriving drones)

Example: consider drone D and its chosen position j. Its array of positions contains *true* at index i ($\neq j$), if and only if it receives a message from another drone claiming i as its own position. If D receives a message with *false* at index j in the array, it broadcasts a new UPDATE message claiming j as its position and with *true* at index j in the array.

After a first consensus has been found (drones have chosen their respective position), the decentralized method will position any new arriving drone at the last position. When a drone leaves the swarm, only drones with a larger position value will change it by decreasing its value by 1. In both cases, this ensures minimal distance changes for all drones and reduces the likelihood of collisions.

The algorithm is composed of two processes, a reception process that gathers received messages into a FIFO structure (mailbox in the Algorithms) and the main process described by Algorithms 2 and 3.

On Figure 5, we can observe that the number of broadcast messages increases almost linearly with the number of changes occurring within the swarm. However, this increase is mainly due to drones joining the swarm as illustrated by Figure 6.

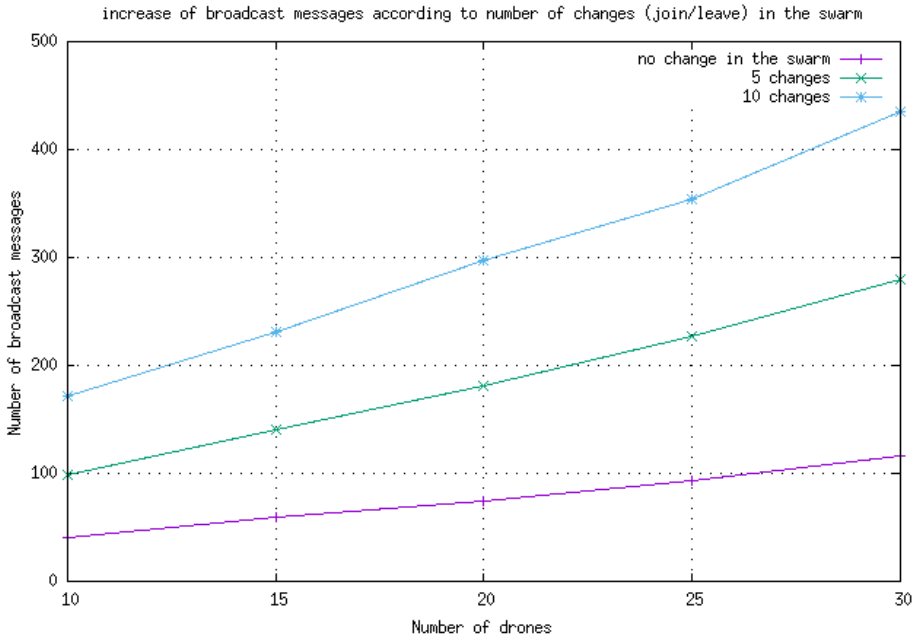

Figure 5. Average Number of Broadcast Messages when new drones are joining the swarm or when drones are leaving the swarm. Considered number of changes are 0, 5 and 10.

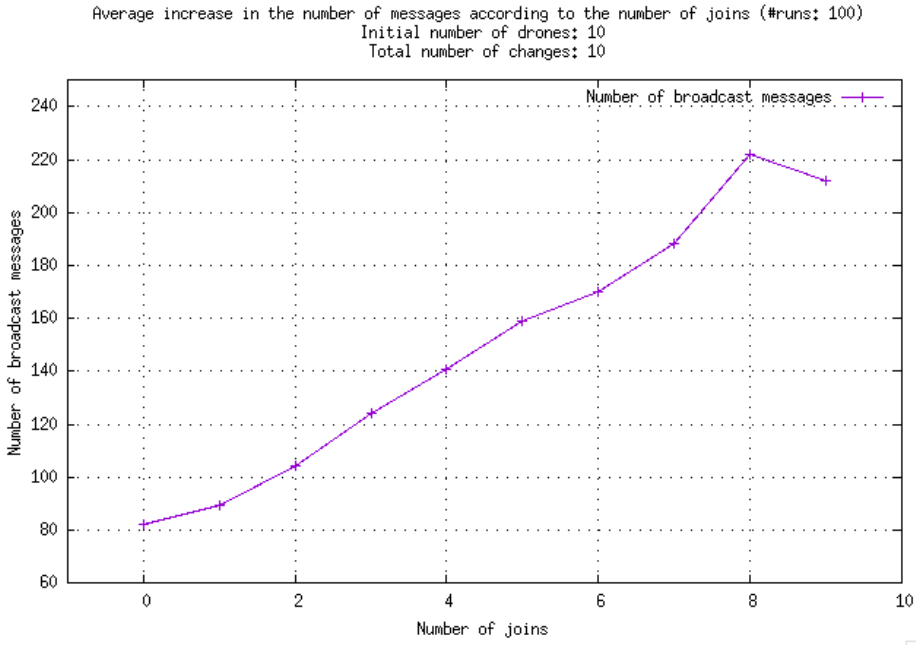

Figure 6. Increase of the broadcast message number with respect to the number of joins. Considered number of changes is always 10, thus 3 joins means that 3 drones joined the swarm while 7 left it.

4. Control framework

The swarm/formation of UAVs (marked with an index i) must surround a pre-defined shape computed by Fourier descriptors. The UAVs orient themselves toward the desired direction β_i (Figure 1) defined as the angle between the X axis and the direction of the UAV in the North/East/Down (NED) reference. The UAVs must reach their target position M_i while avoiding each others to prevent collisions. The GPS coordinates of the reference point (C) are known. The Fourier descriptors return the distance d_{β_i} and the altitude angle γ_i from the point C for a given angle β_i and a rotational quaternion (Section 2, Figure 7). Each UAV

has to regulate to zero its distance errors to the point to reach (M_i) while pointing toward it according to the desired direction β_i (Figure 1). The double exponential smoothing [19–21] is performed in real time to compute a filtered estimation and a prediction of the distance ($\hat{d}_i$) and the bearing angle ($\hat{\psi}_i$).

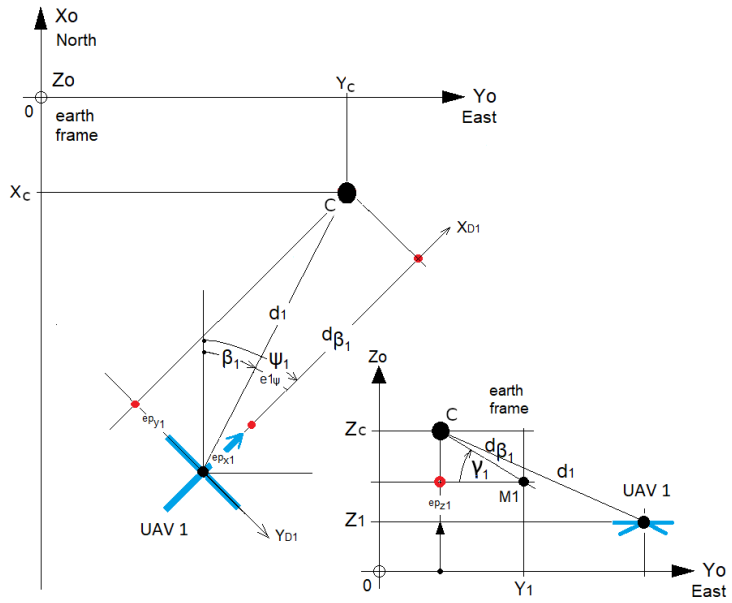

Figure 7. Definition of the position errors for UAV 1.

4.1. Control law

Attitude control has been designed by Squadrone System [22] and the control inputs (roll (Φ), pitch (Θ), yaw (Ψ), vertical acceleration (A_Z)) of the UAVs are available in the C++ API. We will describe the control law for UAV_1.

4.1.1. Position regulation

To reach the desired orientation (β_1) provided by the Fourier descriptors, we just have to give β_1 as reference value (Ψ_1) of the yaw control :

$$\Psi_1 = \beta_1$$

The filtered position errors $\hat{\epsilon p}_1$ of the UAV_1 are (Figure 7) :

$$\hat{\epsilon p}_1 : \begin{cases} \hat{ep}_{x1} = \hat{d}_1 \cos(\beta_1 - \hat{\psi}_1)\cos(\gamma_1) - d_{\beta_1} \\ \hat{ep}_{y1} = \hat{d}_1 \sin(\beta_1 - \hat{\psi}_1)\sin(\gamma_1) \\ \hat{ep}_{z1} = (Z_c - d_{\beta_1}\sin(\gamma_1)) - Z_1 \end{cases} \quad (7)$$

β_1 : bearing angle of the UAV_1 computed by the Fourier descriptors (Figure 2),
d_{β_1} : desired distance of the UAV_1 computed by the Fourier descriptors (Figure 2),
γ_1 : altitude angle between the target point $M1$ and the reference point C computed by the Fourier descriptors and the quaternion (Figure 2),
$\hat{d}_1$: filtered distance between the UAV_1 and the reference point C,
$\hat{\psi}_1$: filtered bearing angle (yaw) of the UAV_1,
Z_C, Z_1 : altitudes of the reference point C and of the UAV_1.

For the UAV_1, the position control objective is the following one (8):

$$\lim_{t\to\infty} \hat{\epsilon p}_1(t) = 0 : \begin{cases} \lim_{t\to\infty} \hat{\epsilon p}_{x1}(t) = 0 \\ \lim_{t\to\infty} \hat{\epsilon p}_{y1}(t) = 0 \\ \lim_{t\to\infty} \hat{\epsilon p}_{z1}(t) = 0 \end{cases} \tag{8}$$

To ensure the control objective (8), the following position control law (9) has been implemented :

$$\begin{bmatrix} \dot{\hat{ep}}_{x1} \\ \dot{\hat{ep}}_{y1} \\ \dot{\hat{ep}}_{z1} \end{bmatrix} = - \begin{bmatrix} f(\sum_{j=1}^{3} kp_{xj}.\epsilon p_{xj}, \bar{V}x_1, \bar{D}x_1, 0)) \\ f(\sum_{j=1}^{3} kp_{yj}.\epsilon p_{yj}, \bar{V}y_1, \bar{D}y_1, 0)) \\ f(\sum_{j=1}^{3} kp_{zj}.\epsilon p_{zj}, \bar{V}z_1, \bar{D}z_1, 0)) \end{bmatrix} = -\vec{V}_1 \tag{9}$$

We obtain :

$$\Longrightarrow \vec{V}_i = \begin{bmatrix} f(\sum_{j=1}^{3} kp_{xj}.\epsilon p_{xj}, \bar{V}x_1, \bar{D}x_1, 0)) \\ f(\sum_{j=1}^{3} kp_{yj}.\epsilon p_{yj}, \bar{V}y_1, \bar{D}y_1, 0)) \\ f(\sum_{j=1}^{3} kp_{zj}.\epsilon p_{zj}, \bar{V}z_1, \bar{D}z_1, 0)) \end{bmatrix} \tag{10}$$

$$\begin{array}{lll} \epsilon p_{x1} = \hat{ep}_{x1} \quad , & \epsilon p_{x2} = \dot{\hat{ep}}_{x1} \quad , & \epsilon \dot{p}_{x3} = \dot{\hat{ep}}_{x1} \quad ; \\ \epsilon p_{y1} = \hat{ep}_{y1} \quad , & \epsilon p_{y2} = \dot{\hat{ep}}_{y1} \quad , & \epsilon \dot{p}_{y3} = \dot{\hat{ep}}_{y1} \quad ; \\ \epsilon p_{z1} = \hat{ep}_{z1} \quad , & \epsilon p_{z2} = \dot{\hat{ep}}_{z1} \quad , & \epsilon \dot{p}_{z3} = \dot{\hat{ep}}_{z1} \quad ; \end{array}$$

$kp_{xj}, kp_{yj}, kp_{zj}$: coefficients to customize,
$\vec{V}_1 = [Vx_1\ , Vy_1, Vz_1]^T$: linear velocity vector of the UAV_1,
$\bar{V}x_1, \bar{V}y_1, \bar{V}z_1$: maximal linear velocities reached when the position errors are respectively higher or equal to the distance $\bar{D}x_1, \bar{D}y_1, \bar{D}z_1$.

4.1.2. Velocity regulation

The filtered velocity errors $\hat{\epsilon v}_1$ of the UAV_1 are (Figure 7) :

$$\dot{\epsilon v}_1 : \begin{cases} \dot{\hat{ev}}_{x1} = V_{x1} - \hat{V}_{x1} \\ \dot{\hat{ev}}_{y1} = V_{y1} - \hat{V}_{y1} \\ \dot{\hat{ev}}_{z1} = V_{z1} - \hat{V}_{z1} \end{cases} \tag{11}$$

For the UAV_1, the velocity control objective is the following one (12) :

$$\lim_{t\to\infty} \dot{\epsilon v}_1(t) = 0 : \begin{cases} \lim_{t\to\infty} \dot{\hat{ev}}_{x1}(t) = 0 \\ \lim_{t\to\infty} \dot{\hat{ev}}_{y1}(t) = 0 \\ \lim_{t\to\infty} \dot{\hat{ev}}_{z1}(t) = 0 \end{cases} \tag{12}$$

To ensure the control objective (12), the following velocity control law (13) has been implemented :

$$\vec{V}_{CMD1} = \begin{bmatrix} f(\sum_{j=1}^{3} kv_{xj}.\epsilon v_{xj}, \bar{\Theta}_1, \bar{V}x_1, 0) \\ f(\sum_{j=1}^{3} kv_{yj}.\epsilon v_{yj}, \bar{\Phi}_1, \bar{V}y_1, 0) \\ f(\sum_{j=1}^{3} kv_{zj}.\epsilon v_{zj}, \bar{Az}_1, \bar{V}z_1, g) \end{bmatrix} \tag{13}$$

$$\begin{array}{lll} \epsilon v_{x1} = \hat{ev}_{x1} \quad , & \epsilon v_{x2} = \dot{\hat{ev}}_{x1} \quad , & \epsilon \dot{v}_{x3} = \dot{\hat{ev}}_{x1} \quad ; \\ \epsilon v_{y1} = \hat{ev}_{y1} \quad , & \epsilon v_{y2} = \dot{\hat{ev}}_{y1} \quad , & \epsilon \dot{v}_{y3} = \dot{\hat{ev}}_{y1} \quad ; \\ \epsilon v_{z1} = \hat{ev}_{z1} \quad , & \epsilon v_{z2} = \dot{\hat{ev}}_{z1} \quad , & \epsilon \dot{v}_{z3} = \dot{\hat{ev}}_{z1} \quad ; \end{array}$$

$kv_{xj}, kv_{yj}, kv_{zj}$: coefficients to customize,
$\vec{V}_{CMD1} = (\Theta_1, \Phi_1, Az_1)^T$: command vector of the UAV_1,

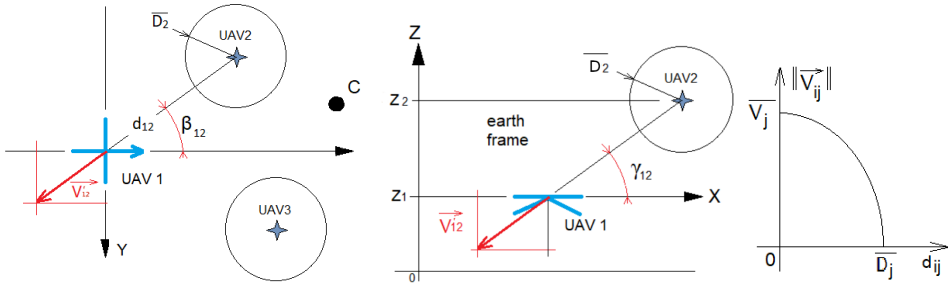

Figure 8. Collision avoidance scheme.

$\bar{\Theta}_1$, $\bar{\Phi}_1$, $\bar{A}z_1$: maximal angular positions and vertical acceleration, reached respectively when the velocity errors are higher or equal to the velocities $\bar{V}x_1$, $\bar{V}y_1$, $\bar{V}z_1$, g is the gravity acceleration in m/s^2.

Note : the full command vector of the UAV_1 is : $\vec{V}_{CMD1} = (\Theta_1, \Phi_1, \Psi_1, Az_1)^T$

The saturation function f is defined as follows:

$$f(\epsilon,\ \max,\ \lim, \text{offset}) = \begin{array}{ccc} -\max + \text{offset} & \text{if} & \epsilon < -\lim \\ \frac{\max}{\lim}\epsilon + \text{offset} & \text{if} & -\lim \leq \epsilon \leq \lim \\ \max + \text{offset} & \text{if} & \epsilon > \lim \end{array} \tag{14}$$

4.1.3. Collision avoidance

The proposed collision avoidance method can be used with static (poles,...) or dynamic (others UAVs) obstacles if their GPS coordinates are known or sent in real time. Let V_i the UAV_i velocity be the subtraction of a repulsive speed V_i^r to an attractive one V_i^a (13). V_i^r is the repulsive effect applied by neighbor UAV_j on UAV_i. When the UAV_i crosses the defined protected circular area of UAV_j (i.e. D_{ij} is below a threshold $\bar{D}_j$), the UAV_i is subject to a repulsive effect, homogeneous to a velocity, which intensity $|\vec{V}_{ij}|$ changes according to the inter-UAV distance d_{ij} (Figure 8). Each UAV computes its distance D_{ij}, altitude angle γ_{ij} and bearing angle β_{ij} with its neighbor j with the Haversine function using its emitted GPS coordinates in real time via the communication network. By including the collision avoidance method, equation (10) becomes:

$$\vec{V}_i = \begin{bmatrix} f(\sum_{j=1}^{3} kp_{xj}.\epsilon p_{xj}, \bar{V}x_1, \bar{D}x_1, 0)) - \sum_{j=1, j\neq i}^{n} \|\vec{V}_{ij}\| \cos(\beta_{ij} - \psi_i) \cos\gamma_{ij} \\ f(\sum_{j=1}^{3} kp_{yj}.\epsilon p_{yj}, \bar{V}y_1, \bar{D}y_1, 0)) - \sum_{j=1, j\neq i}^{n} \|\vec{V}_{ij}\| \sin(\beta_{ij} - \psi_i) \cos\gamma_{ij} \\ f(\sum_{j=1}^{3} kp_{zj}.\epsilon p_{zj}, \bar{V}z_1, \bar{D}z_1, 0)) - \sum_{j=1, j\neq i}^{n} \|\vec{V}_{ij}\| \sin\gamma_{ij} \end{bmatrix} \tag{15}$$

with :

$$\begin{aligned} \gamma_{ij} &= \arctan(Z_j - Z_i, d_{ij}) \\ D_{ij} &= \sqrt{d_{ij}^2 + (Z_j - Z_i)^2} \end{aligned} \tag{16}$$

The intensity of the repulsive effect (Figure 8) applies when :

$$\|\vec{V}_{ij}\| = \begin{cases} \bar{V}_j \cos\left(\frac{\pi D_{ij}}{2\bar{D}_j}\right) & \text{if} \quad D_{ij} < \bar{D}_j \\ 0 & \text{else} \end{cases} \tag{17}$$

5. Simulations and experiments

5.1. Simulations description

To demonstrate the decentralized coordination efficiency using FD and FT, several simulations were conducted on two different shapes. In the previous explanations, we state to restrict shapes to be polar, continuous and 2π periodical. It allows drones to compute their desired position knowing the reference point and their desired bearing angle. Thus, a geoidal and an astroidal shape have been selected (18). On the astroidal formation, we varied parameters to validate the non variation of shape design in 3D space through transformations.

$$\begin{aligned} &\text{astroidal: } f(x) = \frac{|\frac{1}{\cos(x)}|}{(1+(\tan(x))^{2/3})^{3/2}} \; ; \forall x \in \left]\frac{-\pi}{2}; \frac{\pi}{2}\right[\\ &\text{geoidal: } f(x) = 5 + \frac{\cos(3x)}{6} \; ; \forall x \in \left[0; \frac{\pi}{2}\right] \end{aligned} \tag{18}$$

Figure 9. The geoidal (left) and astroidal (right) shape.

Each signature shape is sampled from 0 to 2π by 1000 points allowing up to 500 harmonics without folding effect (Nyquist-Shannon criteria). For all experiences, we used 250 harmonics to retrieve the described shape. Retrieval accuracy optimization according the number of harmonics is not the object of our research. Parameters were set for each simulation to a scale factor of 20, no rotation and $(0, 0)$ as the reference point.

On the first runs, we mandate the swarm to equally distribute themselves angularly on the shape border, facing the reference point, starting from their origin. Then, to depict the total shape and ensure shape continuity, we made runs in which one drone has to follow the shape border by continuously increment its desired bearing angle ($+0.5^\circ$ per second).

5.2. Flight simulations

The first simulations were conducted on Processing [23] to simulate any number of drones. The second set of runs was using a device composed of:

1. 4 "MiniSim" simulators (designed by Squadrone System [22]) of type "hardware in the loop" reproducing the dynamic behaviour of the UAVs (Figure 10),
2. 4 embedded companion computers: RaspBerry PI 3B+, on which our algorithms are implemented in C++. The same embedded computers are installed on the real UAVs,
3. an operator computer to connect the 4 embedded computers together for programming or executing algorithms,
4. a flight simulator on a separate computer on which are linked via rooter the "MiniSim" simumators. The open source software FlightGear [24] is used to visualize the flight of the drone. Several sessions can be run simultaneously to observe the flight of the swarm (one UAV per window as illustrated on Figure 11).

This whole system works and behaves like real drones on the field test without communication hazards.

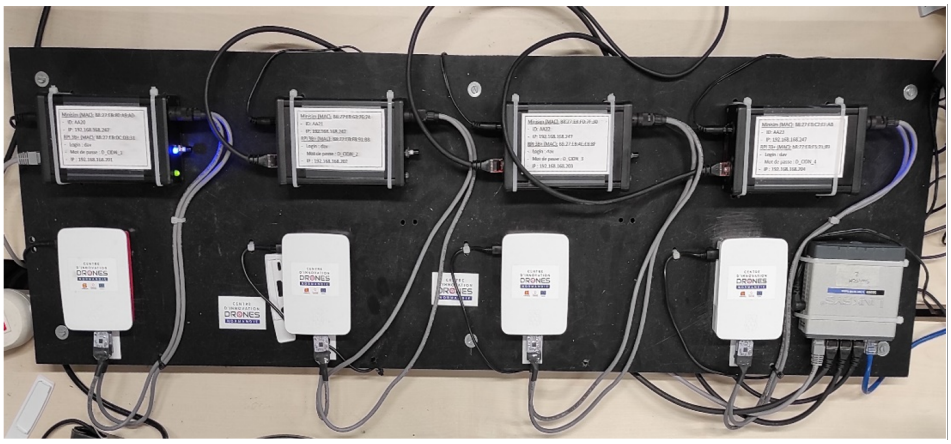

Figure 10. Squadrone Systems "MiniSim" simulators and their Raspberry PI 3B+ companion computers.

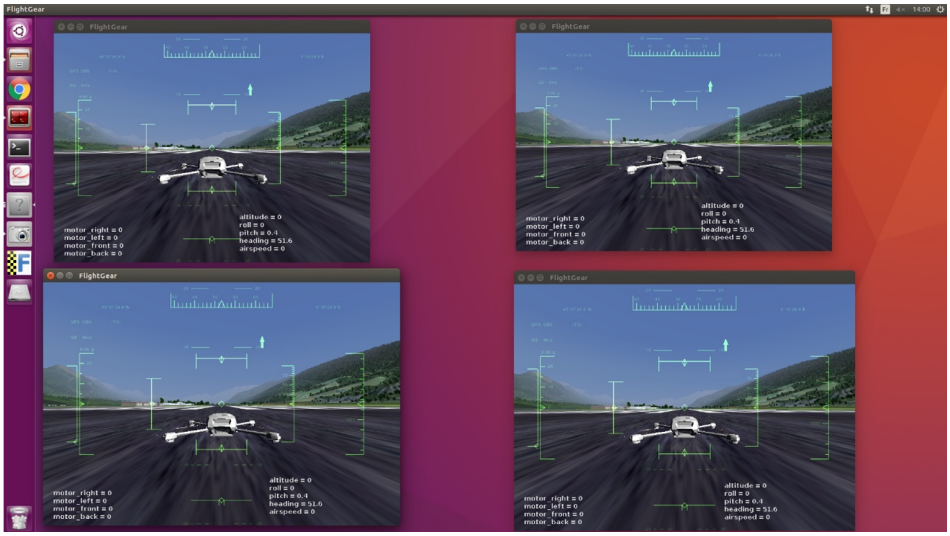

Figure 11. Flights visualisation on FlightGear [24]

5.3. Simulations results

On Matlab 2015B, we visualized the shapes representation with transformation applied on it. As a result, we were able to dynamically transform the shape (translation, rotation and size) without losing continuity nor deforming it (Figure 12).

Thus, one can perform a dynamic formation control without re-computing harmonics nor re-sampling the initial set of points which fed the DFT. We prove our method to be saving computation time when applying transformation on the same shape, opposed to standard method of point controlling each drone which need to compute new drones coordinate each time.

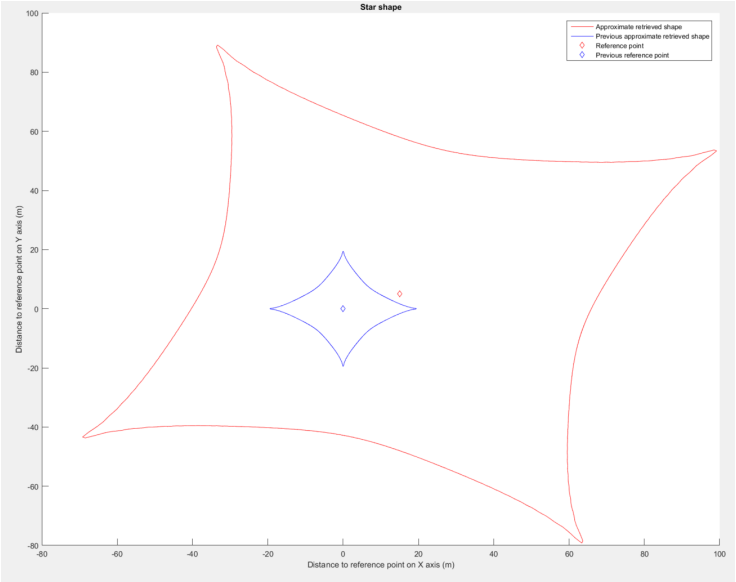

Figure 12. Astroidal shape transformation from previous parameters to new ones: scale = 100, rotation = $\pi/3$, reference point = $(15, 5)$

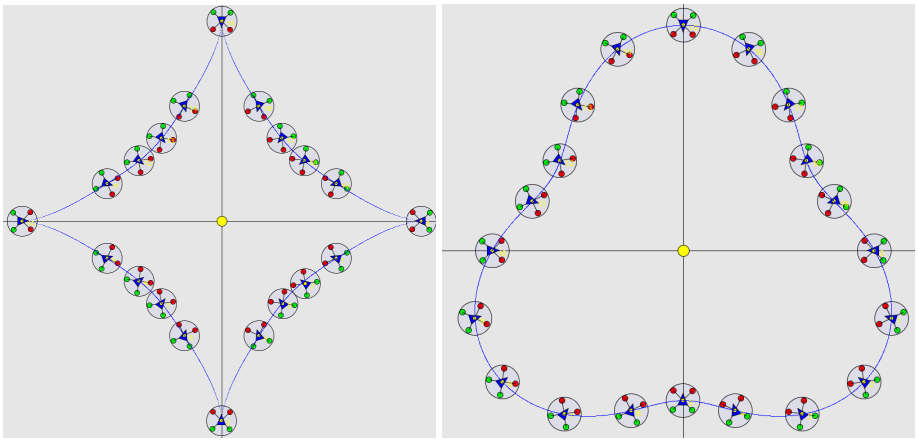

Figure 13. Astroidal and geoidal shape simulation with 20 drones.

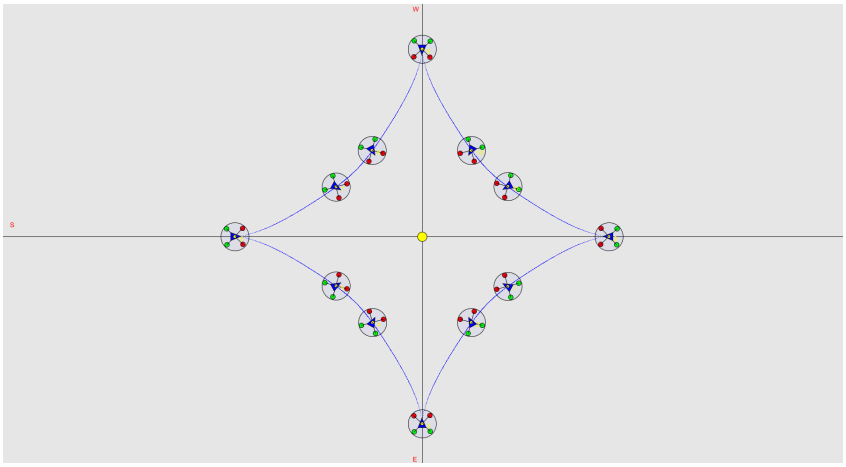

Figure 14. Astroidal shape simulation with 20 drones after losing 8 drones in the formation.

On Processing [23], we simulate the formation control of 20 drones with the astroidal shape (Figure 13). UAVs were able to navigate to their self-computed position without collision. They are angularly distributed around the shape by only giving them shape's harmonics. When we retrieved 8 drones, they autonomously manage the rearrangement (Figure 14).

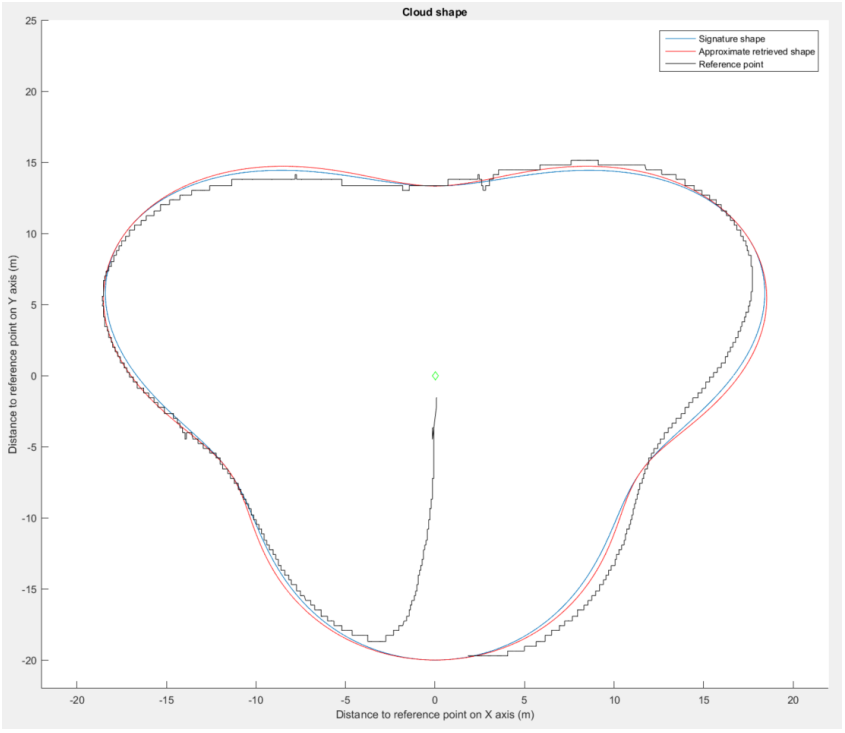

Figure 15. Geoidal shape simulation with the signature shape (blue), the approximated shape (red) and drone's path during the simulation (black).

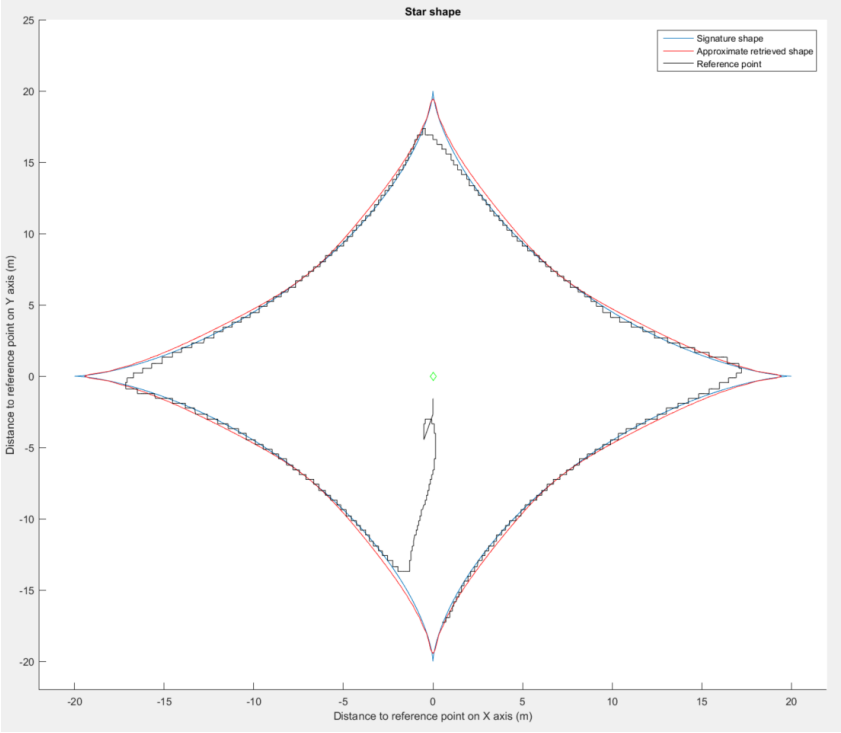

Figure 16. Astroidal shape simulation with the signature shape (blue), the approximated shape (red) and drone's path during the simulation (black).

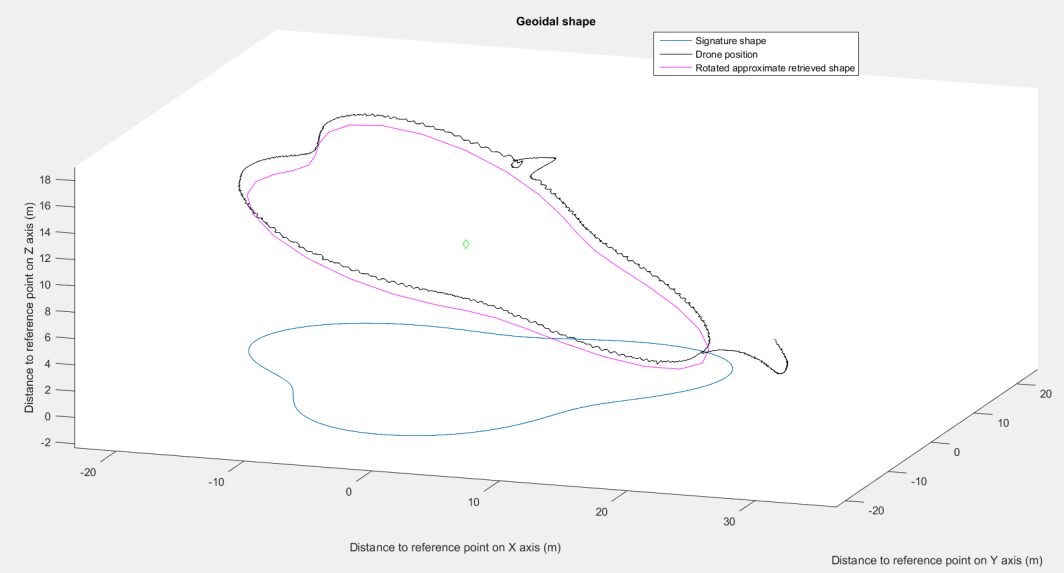

Figure 17. Geoidal shape simulation with the signature shape (blue), the rotated approximated shape (pink) and drone's path during the simulation (black).

We ran a simulation in which one drone navigates around the shape. It validates the method with satisfying placements even on spatial rotations, less than 10% of position deviations from the approximated shape. These deviations are caused by the control law of the drone and inaccuracy of GPS. FD and FT method returns an approximated shape with a deviation lower than 1% in both cases (Figure 15, 16 and 17). Therefore, we prove shape retrieving continuity without borders effect due to discontinuity of the sampled shape. Finally we show that collisions are avoided thanks to the implemented method as it can be seen on Figure 18.

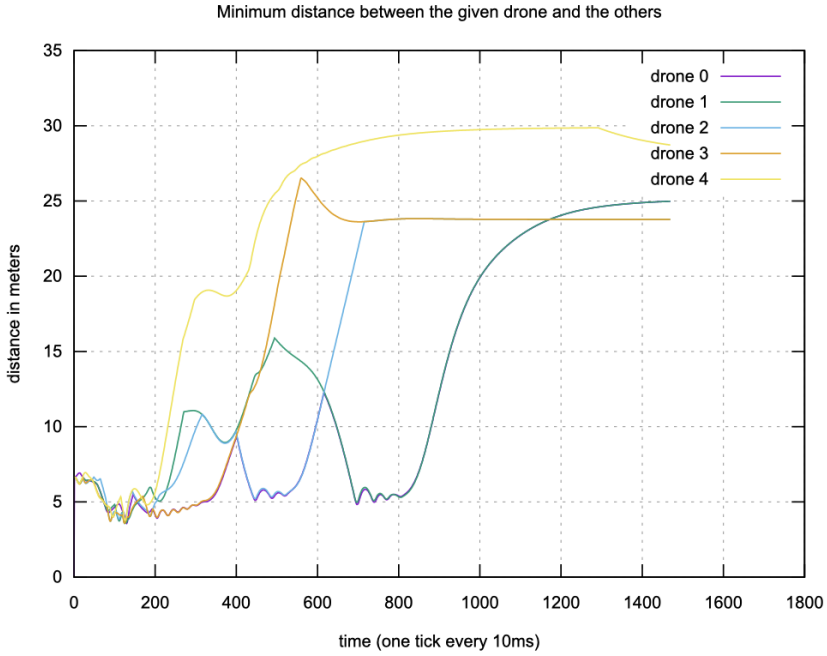

Figure 18. For each drone, minimum distance between itself and the other drones.

5.4. Experimentations

Experiments have been carried out by using DJI F450 UAVs (Figure 19) equipped with a flight controller and an API designed by Squadrone System [22]. Flight controller and API are identical to those available in the Minisim Hardware in the Loop simulators [22]. Fourier descriptors and control laws have been implemented in C++ language in the Raspberry PI 3B+ embedded computer. Note that after successful simulations, the same C++ code can be downloaded in the UAVs. The UAVs communicate with the ground station with a MicroHard 2.4GHz radio modules.

The goal of the experiments was to check if one UAV was able to follow the contours of a particular shape described by means of Fourier descriptors. The experimental procedure consists of choosing a shape and its characteristics (dimensions, inclination, etc.) and programming the increase of β_1 between 0 and 360 in small steps (about 2/s). The calculation of the values of d_{β_1} and γ_1 is carried out using the Fourier descriptors method. An inclined geoidal shape (10°,50m) has been chosen for the experiments. We have registered the GPS coordinates during the flight in order to represent the followed trajectory by the UAV (Figure 20).

To validate collision avoidance, we did an experiment (Figure 21) in which two UAVs flight to reach two opposite points (A and B) by avoiding a central point (C). Initially, the first UAV is at the point A and the second one at the point B. The two drones repeat this operation in a loop until the operator gives them the order to land. Taking into account the disturbances (wind, etc.), they are avoided by randomly going around to the right, to the left, from above or from below. Both of them are able to send its information (GPS coordinates, velocities,...) through the communication network by means of UDP multicast protocol and threads. We have registered the GPS coordinates during the flight in order to represent the followed trajectory by the UAV (Figure 21).

Figure 19. Experimentations with DJI F450

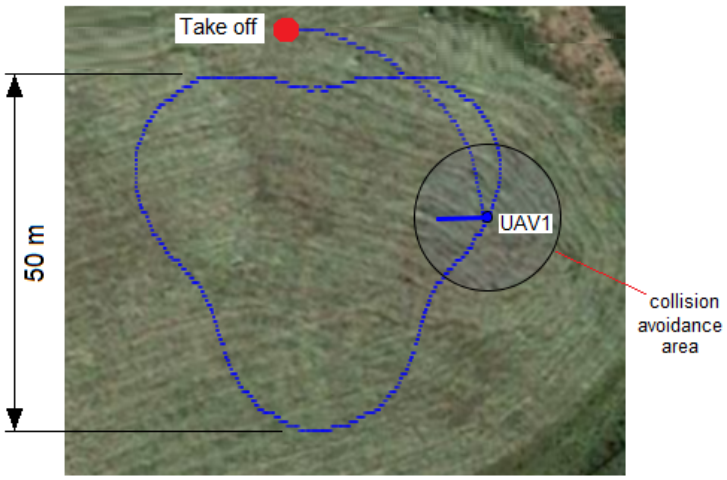

Figure 20. Geoidal shape followed by the UAV

6. Conclusion and perspectives

In the aftermath of the industrial accident at Lubrizol in Normandy, France, which occurred in 2019, numerous questions were raised regarding the potential assistance drones could provide in supporting the efforts of firefighting teams. It became evident that a swarm of drones could potentially provide a comprehensive 360-degree view of the operational theater, provided that it could offer such coverage. To achieve this objective and considering the source of the fire's origin, several challenges must be overcome: (i) precisely describing the area to be enclosed, (ii) enabling the surrounding shape to scale and change orientation, and (iii) ensuring continuous 360-degree vision service by allowing the modification of the swarm's composition to enable the arrival and departure of drones.

In this context, we have proposed a method for swarm formation control. This method relies on Fourier descriptors and transform and on decentralized algorithms for allocating drone positions within the swarm. The decentralized algorithms for allocating bearing angles have only been tested through simulation. However, the restricted number of exchanged messages (w.r.t the number of drones) required to reach a consensus provides optimism for their performance in real-world settings. For the description of the surrounding shape, two polar signature functions have been tested out as formations and, for the astroid shape, some transformations have been applied. In addition, a continuous rotation around the shape border have been given for the swarm of UAVs. As a result, the method demonstrates great accuracy in the shape reproduction even with transformations. The simulated swarm validates the method with satisfying placements. Further experiments and developments could be achieved on virtual actions to smoother UAV's movements and prevent deadlock situations and on an optimization of the number of harmonics to reduce the calculation time. The last but not least, one could use this system to approximately model objects by referencing one by one points at its border.

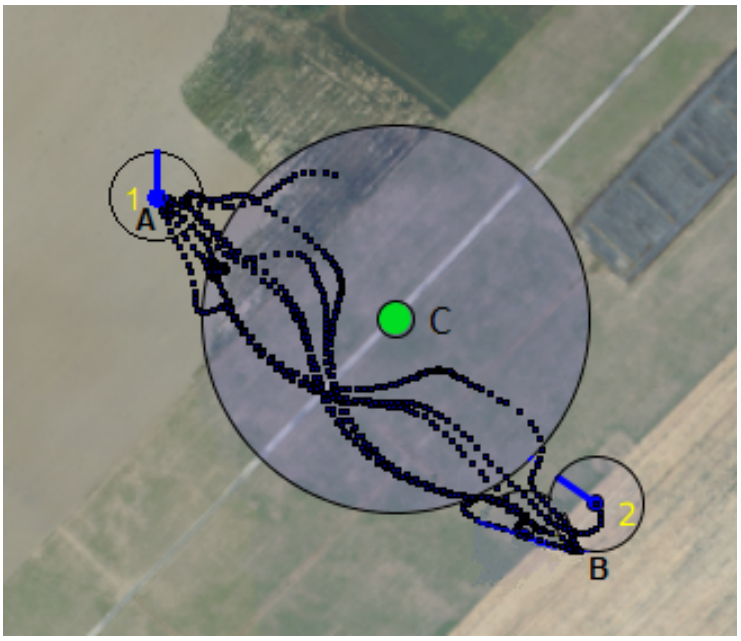

Figure 21. Collision avoidance

7. References

1. Dorigo, M.; Theraulaz, G.; Trianni, V. Swarm Robotics: Past, Present, and Future [Point of View]. *Proceedings of the IEEE* **2021**, *109*, 1152–1165. https://doi.org/10.1109/jproc.2021.3072740.
2. Brambilla, M.; Ferrante, E.; Birattari, M.; Dorigo, M. Swarm robotics: a review from the swarm engineering perspective. *Swarm Intelligence* **2013**, *7*, 1–41. https://doi.org/10.1007/s11721-012-0075-2.
3. Oh, H.; Shirazi, A.R.; Sun, C.; Jin, Y. Bio-inspired self-organising multi-robot pattern formation: A review. *Robotics and Autonomous Systems* **2017**, *91*, 83–100. https://doi.org/10.1016/j.robot.2016.12.006.
4. Debie, E.; Kasmarik, K.; Garratt, M. Swarm robotics: A Survey from a Multi-tasking Perspective. *ACM Computing Surveys* **2023**. https://doi.org/10.1145/3611652.
5. Senanayake, M.; Senthooran, I.; Barca, J.C.; Chung, H.; Kamruzzaman, J.; Murshed, M. Search and tracking algorithms for swarms of robots: A survey. *Robotics and Autonomous Systems* **2016**, *75*, 422–434. https://doi.org/10.1016/j.robot.2015.08.010.
6. Yasin, J.N.; Mohamed, S.A.S.; Haghbayan, M.H.; Heikkonen, J.; Tenhunen, H.; Plosila, J. Unmanned Aerial Vehicles (UAVs): Collision Avoidance Systems and Approaches. *IEEE Access* **2020**, *8*, 105139–105155. https://doi.org/10.1109/access.2020.3000064.
7. Bjerknes, J.D.; Winfield, A.F. On fault tolerance and scalability of swarm robotic systems. In Proceedings of the Distributed Autonomous Robotic Systems: The 10th International Symposium. Springer, 2013, pp. 431–444.
8. Abdelkader, M.; Güler, S.; Jaleel, H.; Shamma, J.S. Aerial Swarms: Recent Applications and Challenges. *Current Robotics Reports* **2021**, *2*, 309–320. https://doi.org/10.1007/s43154-021-00063-4.
9. Chung, S.; Paranjape, A.A.; Dames, P.; Shen, S.; Kumar, V. A Survey on Aerial Swarm Robotics. *IEEE Transactions on Robotics* **2018**, *34*, 837–855.
10. Raja, G.; Kottursamy, K.; Theetharappan, A.; Cengiz, K.; Ganapathisubramaniyan, A.; Kharel, R.; Yu, K. Dynamic Polygon Generation for Flexible Pattern Formation in Large-Scale UAV Swarm Networks. In Proceedings of the 2020 IEEE Globecom Workshops (GC Wkshps. IEEE, 2020. https://doi.org/10.1109/gcwkshps50303.2020.9367501.
11. Huang, J.; Tian, G.; Zhang, J.; Chen, Y. On Unmanned Aerial Vehicles Light Show Systems: Algorithms, Software and Hardware. *Applied Sciences* **2021**, *11*, 7687.
12. Zahn, C.T.; Roskies, R.Z. Fourier descriptors for plane closed curves. *IEEE Transactions on computers* **1972**, *100*, 269–281.

13. Uesaka, Y. A new Fourier descriptor applicable to open curves. *Electronics and Communications in Japan (Part I: Communications)* **1984**, *67*, 1–10.
14. Rui, Y.; She, A.C.; Huang, T.S. Modified Fourier descriptors for shape representation-a practical approach. In Proceedings of the Proc of First International Workshop on Image Databases and Multi Media Search. Citeseer, 1996, pp. 22–23.
15. Zhang, D.; Lu, G.; et al. A comparative study of Fourier descriptors for shape representation and retrieval. In Proceedings of the Proc. 5th Asian Conference on Computer Vision. Citeseer, 2002, p. 35.
16. Barnes, L.E.; Fields, M.A.; Valavanis, K.P. Swarm formation control utilizing elliptical surfaces and limiting functions. *IEEE Transactions on Systems, Man, and Cybernetics* **2009**, *39*, 1434–1445.
17. Rezaee, H.; Abdollahi, F. A decentralized cooperative control scheme with obstacle avoidance for a team of mobile robots. *IEEE Transactions on Industrial Electronics* **2014**, *61*, 347–354.
18. Falomir, E.; Chaumette, S.; Guerrini, G. Mobility strategies based on virtual forces for swarms of autonomous UAVs in constrained environments. In Proceedings of the International Conference on Informatics in Control, Automation and Robotics (ICINCO 2017), 2017, Vol. 1, pp. 221–229.
19. LaViola, J.J. Double exponential smoothing: an alternative to Kalman filter-based predictive tracking. In Proceedings of the Proceedings of the workshop on Virtual environments 2003, 2003, pp. 199–206.
20. LaViola, J.J. An experiment comparing double exponential smoothing and Kalman filter-based predictive tracking algorithms. In Proceedings of the IEEE Virtual Reality, 2003. Proceedings. IEEE, 2003, pp. 283–284.
21. Bastourous, M.; Guerin, F.; Guinand, F.; Lemains, E. Decentralized high level controller for formation flight control of uavs. In Proceedings of the 2020 6th International Conference on Mechatronics and Robotics Engineering (ICMRE). IEEE, 2020, pp. 226–231.
22. Squadrone-system (https://squadrone-system.com/ last visited: 2023-05-19).
23. Processing (https://processing.org/).
24. FlightGear (https://www.flightgear.org/).

8. Annex: Algorithms

8.1. Algorithm for the Known Swarm Size case

Algorithm 1 Algorithm executed by each drone for the scenario of known swarm size (N)

create and initialize $Positions[N]$
$myPosition \leftarrow$ random(N)
change $\leftarrow$ *true*
while $\exists i, Positions[i] = false$ **do**
    if change **then**
        $Positions[myPosition] \leftarrow true$
        broadcast $Positions[]$ and $myPosition$
        change $\leftarrow false$
    else
        if $receivedPositions[]$ and $peerPosition$ have been received **then**
            if $Positions[] \neq receivedPositions[]$ **then**
                merge $Positions[]$ and $receivedPositions[]$
                /* if $receivedPosition[j]$ is true then $Positions[j]$ becomes true */
                change $\leftarrow true$
            end if
            if $myPosition = peerPosition$ **then** /* conflict */
                $myPosition \leftarrow random(N)$ such that $Positions[myPosition] = false$
                $Positions[myPosition] \leftarrow true$
                change $\leftarrow true$
            end if
        end if
    end if
end while
broadcast $Positions[]$ and $myPosition$
compute the angle then the destination point
move to the destination point

8.2. Algorithms for the Dynamic Swarm Size case

Algorithm 2 Algorithm executed by each drone for the scenario of changing swarm size

```
mailbox ← ∅
N ← 1
create and initialize Positions[1]
myPosition ← 1
change ← true
broadcast (1, Positions, JOIN)
newToTheSwarm ← true
while mission is not finished do
    if mailbox has some messages then
        /* messages are received into the mailbox by another process asynchronously */
        change ← false
        while mailbox has some messages do
            change ← Process Messages (Algo 3)
        end while
    else
        if all cells of Positions[] are true then
            compute bearing angle
        else
            change ← true
        end if
    end if
    if change then
        broadcast (myPosition, Positions[], UPDATE)
        change ← false
    end if
end while
Positions[myPosition] ← false
broadcast (myPosition, Positions[], LEAVE)
```

Algorithm 3 Process Messages

```
read next message: uavPosition, receivedPositions, msgType
if msgType is JOIN then
    N ← N + 1
    Positions[N] ← false
    change ← true
end if
if msgType is LEAVE then
    N ← N − 1
    if uavPosition < myPosition then
        myPosition ← myPosition − −
        update Positions[]
    end if
    change ← true
end if
if msgType is UPDATE then
    if newToTheSwarm AND size of receivedPositions[] > 1 then
        update N
        update Positions[]
        change ← true
    else
        if myPosition = uavPosition then
            myPosition ← find a random free position
            update Positions[]
            change ← true
        else
            update Positions[]
            if receivedPositions[myPosition] = false then
                change ← true
            end if
        end if
    end if
end if
newToTheSwarm ← false
return change
```