Exploring the Split and Join Capabilities of a Robotic Collective Movement Framework

Regular Paper

Iñaki Navarro¹,*

¹ Universidad Politécnica de Madrid, Madrid, Spain
* Corresponding author E-mail: inaki.navarro@upm.es

Received 8 Jun 2012; Accepted 4 Dec 2012

DOI: 10.5772/55390

© 2013 Navarro; licensee InTech. This is an open access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/3.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Abstract The splitting and joining capabilities of a previously designed framework for the collective movement of mobile robots are studied. This framework is implemented in a distributed manner among the robots and allows scalability with increasing number of robots. It is expandable with new functionalities through the inclusion of new modules. Splitting functionality is created by including a new module, while joining does not need new structures. A total of 130 systematic experiments, grouped in six different sets, were undertaken to test the splitting and joining functions of the framework, yielding results in which robots behaved as expected. Four of the sets correspond to experiments in simulation using up to 40 robots. The remaining sets use eight real robots.

Keywords Split, Joining, Collective Movement, Flocking

1. Introduction

The collective movement of mobile robots is one of the basic and fundamental problems in multi-robot systems. It is the problem of how to make mobile robots move together as a group, behaving as a single entity. It represents a starting point for more complex tasks such as search tasks, collective mapping tele-operation by a single operator of a group of robots, etc. (Navarro & Matía, 2013). In this article, a previously designed framework for the collective movement of robots (Navarro & Matía, 2011) is expanded with a new module to allow splitting into sub-groups of robots. In addition, the joining capability with no need for new modules is analysed.

For a group of robots moving together it can be useful or necessary to divide or split into two or more groups. This can allow them to pass by a large obstacle so that one group leaves the obstacle on its left and the other group leaves it on its right. The decision on when and how to split might depend on the number of robots, the desired task that the group must perform or on the environment. For instance, a group of robots can split into sub-groups for a more efficient performance in a searching task.

Two or more groups of robots moving separately can join eventually and become a larger group that can behave as a single one. This can be useful if the task that robots
perform requires more members or, for instance, if the group of robots must be tele-operated to a desired place by only one human.

Several of the solutions for the collective movement problem imply the existence of a leader that moves while being followed by one or more robots that may also act as leaders for other robots (Balch & Arkin, 1998; Barfoot et al., 2002; Naffin & Sukhatme, 2004). The main disadvantage of this type of architecture is that it does not scale well with an increasing number of robots since the position error propagates cumulatively from leader to follower. In other solutions no leaders are designated and robots react to the positions of their neighbouring robots, while moving in an emergent direction (Antonelli et al., 2008; Spears et al., 2004). These last solutions usually need global information according to the desired direction of movement, which makes them not scalable.

The Physicomimetics Framework (PF) by Spears et al. (2004), allows creation of hexagonal lattices on a fully decentralized algorithm. As in our framework, robots react to the neighbouring positions with attractive or repulsive forces, but they need global information to move towards an objective. In our no global information is necessary thanks to the Distributed Orientation Agreement Mechanism explained later. PF is scalable with increasing number of robots, as it is shown in simulations. PF does not allow splitting into sub-groups, and potential joining is much harder than in our proposed solution thanks to the way neighbours are chosen.

Antonelli et al. (2008) propose a method that also creates collective movement placing robots in a hexagonal lattice but it also needs a virtual goal which limits the scalability and application on real robots. The control of real robots if performed off-board, using a camera that tracks the positions of the robots, and sending commands to the motors. Our proposed framework does not need to know the position of a virtual goal, and direction of movement is decided between all the robots. As in the PF, split is not implemented and potential rejoining does not work easily.

Turgut et al. (2008) present a scalable and distributed algorithm for robot flocking. It is based on the heading alignment of the robots and maintaining a desired inter-robot distance. The robots are provided with a compass and with wireless communications so that they can inform their neighbours of their heading with respect to their detected magnetic north. The algorithm is tested in simulation with hundreds of robots and with a small group of real robots. Çelikkanat & Şahin (2010) extend Turgut’s research, modifying the steering behaviour of the flock by introducing externally informed robots. They test the algorithm both in sets of systematic experiments using both physical and simulated robots for different ratios of informed robots. Ferrante et al. (2010) improve this work by replacing the compass with a light sensor and using local communication to explicit signalling in order to achieve heading agreement with static and dynamic target directions. The performance is tested in realistic simulations.

Ögren (2004) proposes a set of controllers for formations of robots where robots join or exit the formation depending on the quality of the movement. If the velocity of the formation is low or the error in position is high they leave the formation. If the velocity is high and robots are in a good position in the formation they join it. The method is tested only in simulation, with only one example.

In (Chaimowicz & Kumar, 2004), a method for splitting a group of robots and rejoining two groups of robots is explained. Each group of robots is leaded by an aerial blimp that tracks the robots. If two groups of robots have the same destination and one of the blimps is able to track both groups it will get in charge of both and rejoin the groups. If one group has to split into two sub-groups, the aerial blimp will lead one of them and ask a second blimp to track the other sub-group. As can be seen the high-level coordination is centralized in the aerial blimp and not distributed among the robots.

The rest of the article is organized as follows. The framework and its different relevant pieces are described in Section 2. A description and results of the set of experiments performed is given in Section 3. Finally the conclusions and future work can be found in Section 4.

2. The Framework

The aim of this article is to demonstrate the joining and splitting capabilities of a framework for the collective movement of mobile robots. The details and generalities of this framework are described in a previous piece of research (Navarro & Matía, 2011), but it is necessary to summarize here the framework in order to understand it. One pieces of the framework is the modules. In order allow splitting into sub-groups, the Obstacle Avoidance Module and Splitting Control Module must be present. The first is summarized in Section 2.3, since it was presented in the previous work. The second is described in detail in Section 2.4. The joining capability works directly without the inclusion of any module. The other important piece of the framework is the Flocking Controller, which is explained in Section 2.2

By running the framework in a group of robots, they move as a unique entity, being able to avoid obstacles, maintaining a desired inter-robot distance, splitting into
sub-groups, and/or deciding collectively the direction of movement and speed. The controllers are distributed in each robot, and both communications and sensing are local, which allows scalability with an increasing number of robots.

The framework is based on three different types of element:

- **Distributed Orientation Agreement Mechanism.** This provides every robot with a common reference orientation that is used by the controllers. The mechanism is described in Section 2.1 and analysed and tested in more detail in (Navarro & Matía, 2012). This could be replaced by other mechanisms such as a compass or other localization methods that allow the robots to share a reference orientation.

- **Controllers.** They control the movements of the robots. The **Flocking Controller** is distributed among the robots, making them move as a group in a determined direction while maintaining a desired inter-robot distance. It requires the robots to share a common reference orientation obtained by the Distributed Orientation Agreement Mechanism. The **Safety Controller** is in charge of avoiding collisions with near obstacles.

- **Modules.** They modify one or more of the parameters of the Flocking Controller in order to allow obstacle avoidance, speed control, splitting control or changing inter-robot distance.

The relations between these three types of element are shown in Figure 1. The arrows indicate the flow of information from the modules to the controllers, showing the parameter that is changed in the controller. The modules in grey are used in the splitting experiments, while those in white allow other functions not present in this study.

The robots used in the experiments are Khepera III, able to detect nearby obstacles by the use of ultrasound and infra-red sensors. In order for the framework to work, the robots must be able to estimate the relative position of neighbouring robots. This estimation is made by a relative positioning system (Pugh et al., 2009), based on the strength of the infra-red signals interchanged by neighbouring robots. The system is also able to communicate short messages between robots. Through the use of these messages, robots exchange a unique id, which is necessary for the proposed algorithms of the framework. The robots communicate periodically also using IEEE 802.11 wireless standard and UDP messages. Both communication systems make up two parallel networks, whose links are limited by the distance between robots in order to keep communications local. Messages are always sent from one robot to one of its communication neighbours. Information is spread by way of each robot receiving a message, processing it and sending new messages to its neighbours.

![Figure 1. Overview of the framework and the relations between its different elements. The modules (top) modify parameters of the Flocking Controller (centre). The Safety Controller (middle right) supersedes the Flocking Controller to avoid collisions. The Distributed Orientation Agreement Mechanism (bottom) provides the Flocking Controller with a reference orientation common to all the robots. Arrows indicate the flow of information. Modules in white are not described in this research.](www.intechopen.com)
2.1 Distributed Orientation Agreement Mechanism

The objective of the mechanism is to make a group of robots agree on a reference orientation without the use of any external reference, compass or global positioning system. As a result of applying the method, robots share a distributed virtual compass, i.e., robots agree on a reference orientation. The agreement is maintained while the algorithm is working, serving as a common reference.

While applying the mechanism, every robot has an initial estimate of the reference orientation which represents the angular value the group of robots must agree on. Robots exchange this estimate among them and apply a consensus algorithm in order to agree. This reference orientation is expressed on each robot relative to its own coordinate system, so each pair of neighbouring robots that exchanges their estimates must know their relative headings. This is performed by using a relative positioning system and exchanging wireless messages. The steps of the algorithm, executed cyclically, are the following:

**Step 0**

Each robot \( r_i \) has an internal estimate of the reference orientation, expressed in its own coordinate system:

\[
\{ \theta_i \} \quad (1)
\]

**Step 1**

Each robot \( r_i \) reads the bearings to neighbouring robots \( r_j \), i.e., the relative angle to robot \( r_j \) expressed in the coordinate system of robot \( r_i \) (\( \beta_j \)). It is obtained using the relative positioning system and saved together with a time stamp, and the neighbour’s identification number (ID):

\[
\{ \beta_j(t_i)_{r_i}ID \} \quad (2)
\]

**Step 2**

Each robot \( r_i \) receives wireless messages from every neighbouring robot \( r_j \), which contain information about the estimate reference orientation of each neighbouring robot expressed in its own coordinate system (\( \theta_j \)), the bearing where robot \( r_i \) was seen from \( r_j \) in the coordinate system of \( r_i \) (\( \beta_j \)), the time stamp when bearing was measured (\( t_j \)), and the identification number of the neighbouring robot (ID), so later robot \( r_i \) can couple information. All this information is saved by robot \( r_i \) for each one of its neighbouring robots:

\[
\{ \theta_j, \beta_j(t_j)_{r_j}ID \} \quad (3)
\]

**Step 3**

The reference orientations received from neighbouring robots \( \theta_j(\theta_i) \) are expressed in the coordinate system of \( r_i \) and must be translated to the coordinate system of \( r_j(\theta_j) \). This can be calculated by robot \( r_i \) without considering the timing, as follows:

\[
\theta_i = \theta_j - \beta_j + \pi + \theta_i \quad (4)
\]

In Fig. 2 a schematic with the angles involved in this operation is outlined.

**Figure 2.** Angles involved in the calculation of \( \theta_i \) 

In practice robots move and messages suffer from delays, so for robot \( r_i \) to calculate \( \theta_i \), applying Eq. 4 needs to use \( \beta_j \) and \( \beta_j \), which correspond to exactly the same moment. We consider that the robots’ clocks are synchronized. When robot \( r_i \) receives \( \beta_j(t_i) \) from robot \( r_j \) it has suffered a delay, so it has to retrieve the value of \( \beta_j(t_i) \) corresponding to the same instant \( t_i = t_j \). If Eq. 4 is applied with these values, the resulting \( \theta_i \) will be for the time \( t_j \), which, in principle, due to the delays, is not the current time. So it is important to correct this using the change of heading \( \Delta_\text{odometry}(t_i^', t_i) \) of robot \( r_i \), from \( t_i \) to the moment \( t_i' \), when the \( \theta_i \) was estimated last. Then these values will be the inputs to the consensus algorithm. The following equation takes these details into account:

\[
\theta_i(t_i') = \theta_i(t_i) - \beta_j(t_i') + \pi + \theta_i - \Delta_\text{odometry}(t_i^', t_i) \quad (5)
\]

Robot \( r_i \) needs to get \( \beta_j(t_i) \) from its stored values. Since the value will probably not be stored for that exact moment, the robot will interpolate from the two samples with timing values around \( t_i \). The same applies for the odometry \( \Delta_\text{odometry}(t_i^', t_i) \).

Each robot \( r_i \) will calculate \( \theta_i(t_i') \) for every neighbour \( r_j \in N_{r_i} \), from which receives a wireless message containing \( \beta_j(t_i) \) and \( t_i' \).
Step 4

Each robot \( r_i \) applies a consensus algorithm using the neighbours’ reference orientation \( \theta_i(t_j) \) and its own estimate can be applied to compute the new estimate. This algorithm is implemented as a weighted average. The consensus algorithm can be summarized in the next equation:

\[
\theta_i(t') = f\theta_i(t'), \ldots, \theta_i(t'), \ldots \forall j \in N_i \quad (6)
\]

Step 5

The penultimate step in the cycle of the algorithm is for \( r_i \) to update \( \theta_i \) with the odometry for the current time \( t'_i \):

\[
\theta_i(t'_i) = \theta_i(t'_i) - \Delta \text{odometry}(t'_i, t'_j) \quad (7)
\]

Step 6

Lastly, each robot \( r_i \) creates and sends wireless messages to neighbouring robots. These messages include the reference orientation estimate \( \left( \theta_i(t') \right) \), the bearing to the neighbouring robot \( \left( \beta_i(t) \right) \), the time-stamp specifying the moment when the robot was observed, \( (t) \) and the identification number of the robot \( (ID_i) \). This information is summarized as follows:

\[
\left[ \theta_i, \beta_i(t), t, ID_i \right] \quad (8)
\]

This information corresponds to the received in step 2 from neighbouring robots.

2.2 Flocking Controller

This controller is in charge of the basic collective movement of the robots. The controller’s output is a vector that indicates the desired direction of movement that is translated into wheel speed by kinematic controllers. It depends on several parameters and the positions of neighbouring robots. The main parameters of the Flocking Controller are \( d, \beta, K_{\text{forward}}, K_{\text{aggregation}} \) and \( \text{group} \). They are modified by the modules in order to change the interrobot distance, the group velocity and allow splitting.

In order to place robots into a hexagonal lattice, the robots react to attractive and repulsive forces depending on the positions of the neighbours. The identification of neighbouring robots is important for the correct working of the controller and for allowing a better grouping. Neighbouring robots are classified into two types: type 1 and type 2.

Robots \( r_i \) belong to type 1 when their distance to robot \( r_j \) is under a certain value \( \delta \). The chosen value for \( \delta \) is \( \delta = 1.25d_d \). Robots at a distance \( \sqrt{3}d_d \) or larger cannot be initially considered as neighbours, since clustering will appear when applying attractive forces between them.

If robots further than \( \delta = 1.25d_d \) are not considered, and initial positions among robots were further than \( \delta \), or a robot in the border of the group separates more than \( \delta \), the algorithm would not work. The solution for this includes robots of type 2. The algorithm takes as prospective robots of type 2 those robots \( r_j \) that fulfil the following equation:

\[
\delta < d_{ij} \leq r_{rm}d_d \quad (9)
\]

where \( r_{rm} \) is the relative maximum range which in the performed experiments is set to \( r_{rm} = 4 \). This means that prospective neighbouring robots are situated inside a ring of the given inner and outer radius around \( r_i \). The second condition is that the angular distance of robot \( r_j \) to every robot \( r_k \) of type 1 is larger than a threshold angle \( \theta_{\text{ang}} \):

\[
\left| \beta_{jk} \right| > \theta_{\text{ang}} \forall \left| r_k \right| \in \text{type1} \quad (10)
\]

In Figure 3, the different parameters involved in determining the type of neighbouring robots are shown.

The use of neighbouring robots of type 2 provides the robots with the ability to join into a single group even if their trajectories do not go one towards the other. This is thanks to the larger range of neighbouring consideration and represents an innovation compared to other frameworks based on neighbour-referenced algorithms. This also allows lost or isolated robots to join with a main group. In addition, this makes robots more compact and avoids undesired splitting of groups.

The controller output for a robot \( r_i \) is a virtual velocity made up of two vectors: \( v_{i, \text{aggregation}} \) that is in charge of maintaining the desired inter-robot distance; and \( v_{i, \text{group}} \) that imposes the group direction movement.

![Figure 3](image-url)

**Figure 3.** The sketch shows the different parameters involved in the determination of neighbouring robots of type 1 and type 2. Robots \( r_1, r_2 \) and \( r_3 \) are neighbouring robots of type 1 of robot \( r_4 \), while robots \( r_4, r_5 \) and \( r_6 \) are of type 2.
\[ \vec{v}_{i,\text{total}} = \vec{v}_{i,\text{group}} + \vec{v}_{i,\text{aggregation}} \]  

\( \vec{v}_{i,\text{group}} \) is the virtual velocity that moves robot \( r_i \) in the desired group direction \( \theta_g \). It is a vector with the orientation \( \theta_g \), translated to the reference system of the robot and with a magnitude weighted by the \( K_{\text{forward}} \) parameter:

\[ |\vec{v}_{i,\text{group}}| = K_{\text{forward}} \]  

\[ \arg(\vec{v}_{i,\text{group}}) = \theta_{\text{reference},i} - \theta_g \]  

\( \theta_{\text{reference},i} \) is the estimate of the reference orientation given by the Distributed Orientation Agreement Mechanism.

\( \vec{v}_{i,\text{aggregation}} \) of robot \( r_i \) is a sum of the virtual attractive/repulsive velocities (\( \vec{v}_{i,j} \)) for each one of the neighbouring robots \( r_j \) (type 1 or type 2).

\[ \vec{v}_{i,\text{aggregation}} = \sum_{\forall j \neq i \text{type 1}} \vec{v}_{i,j} + \sum_{\forall j \neq i \text{type 2}} \vec{v}_{i,j} \]  

The magnitude and angle of \( \vec{v}_{i,j} \) are defined as follows:

\[ |\vec{v}_{i,j}| = K_{\text{aggregation}} |d_{i,j} - d_d| \]  

\[ \arg(\vec{v}_{i,j}) = \begin{cases} \theta_{i,j} & \text{if } d_{i,j} > d_d \\ \theta_{i,j} + \pi & \text{if } d_{i,j} \leq d_d \end{cases} \]  

where \( d_{i,j} \) is the distance from robot \( r_i \) to the neighbouring robot \( r_j \), and \( \theta_{i,j} \) is the relative angular position of robot \( r_j \) seen from robot \( r_i \). So the virtual velocity \( \vec{v}_{i,j} \) is an attractive vector towards robot \( r_j \) if the robots are further than the desired distance, and repulsive if they are too close.

The generated virtual velocity \( \vec{v}_{i,\text{total}} \) must be translated into wheel movement, by using two controllers. The Kinematic Controller B allows backwards movement. In the Kinematic Controller A, robots just move forward and if \( \vec{v}_{i,\text{total}} \) points backwards they turn around before making any translation movement. Both kinematic controllers are described in (Navarro & Matía, 2011).

### 2.3 Obstacle Avoidance Module

The aim of this module is to control the group direction of movement (\( \theta_g \)) in order to avoid the collisions with obstacles and allow splitting into sub-groups when the Splitting Control Module is present. The module, explained in more detail in (Navarro & Matía, 2011), is based on a distributed consensus algorithm. This allows the robots to decide collectively the group direction of movement (\( \theta_g \)), agreeing on it at the same time that more importance is given to those robots with more relevant information.

When an obstacle is detected by a robot, a new desired direction of movement - which aims to avoid the obstacle - is calculated. This new value of \( \theta_g \) is transmitted to neighbouring robots together with a variable representing the relevance of that information. Neighbouring robots update their \( \theta_g \) with the new direction, but also take into account the relevance. In addition, these neighbouring robots send this information to their neighbours, spreading it to the whole group and making all the robots agree on the direction decided by the robot that found the wall.

### 2.4 Splitting Control Module

This module allows the splitting of a group of robots into two sub-groups in combination with the Obstacle Avoidance Module. The aim of the module is simply to demonstrate that the framework and the controllers are able to make the group of robots split, the decision on when to split and in which manner is given arbitrarily by one of the robots. The decision is not related to the perceived environment.

The splitting mechanism is done by making a part of the group of robots belong to one group and the other part to a different one. This is shown in the group variable. If one robot detects a nearby robot of a different group from its own group then it does not consider it as a neighbouring robot. Then the Flocking Controller and the Distributed Orientation Agreement Mechanism will not take into account that nearby robot. In addition, the Obstacle Avoidance Module will consider that robot as an obstacle and will try to avoid it. So, if in a group of robots some of them change their group variable, the group will split and the new two sub-groups will separate from each other.

In order to split the group, one of the robots establishes a line that divides the group; robots at one side belong to one sub-group (group = 1) and robots at the other side to the other (group = 2). This line is defined by its orientation (\( \theta_{\text{line}} \)) relative to the common orientation reference, and the minimum distance (\( d_{\text{line},j} \)) ,positive or negative, to the line from the robot, is the perpendicular vector from a robot to the line. A schematic showing this is depicted in Figure 4.

The robot that decides to split calculates the \( d_{\text{line},j} \) of its neighbouring robots, using the relative bearing to them and \( d_{\text{line},j} \) and \( \theta_{\text{line}} \). Then it sends to them \( d_{\text{line},j} \) together with the group numbers and \( \theta_{\text{line}} \). These neighbouring robots do the same with its neighbours and the information propagates and robots set their new group. After that the group split into two sub-groups that move away from each other thanks to the presence of the Obstacle Avoidance Module.
distance error (MDE) indicates the error in the expected inter-robot distance. It is the average of the inter-robot distance error with every neighbour averaged for every robot. Path length ratio (PLR) is the ratio between path length followed by the centre of mass of the group and the straight line distance from the initial to the end point.

| Num. Robots | Set1-8-sim (joining) | Set1-40-sim (joining) | Set1-18-real (splitting) | Set2-8-sim (splitting) | Set2-40-sim (splitting) | Set2-7-real (splitting) |
|-------------|----------------------|-----------------------|--------------------------|------------------------|------------------------|------------------------|
|             | 8                    | 8                     | 8                        | 8                      | 8                      | 8                      |
| Num. Exp    | 30                   | 30                    | 30                       | 30                     | 30                     | 30                     |
| Time of Exp. (s) | 60                 | Variable              | Variable                 |                        |                        |                        |

Table 1. Characteristics of experiments performed in the different sets of experiments where the joining or splitting capabilities of the framework are tested.

For the Khepera III robots used, if a robot is perfectly placed with respect to its neighbouring robots, then it should move forward at a speed of 0.178 km/s. This value is used as a reference to compare the ideal group speed (idealGS).

The simulator used in the simulated experiments was Webots (Michel, 2004), using a dynamic model of the Khepera III robot. In the simulations, the robots read the relative positions of their neighbours, update their sensor readings and update their motor commands every 128 ms. They exchange wireless messages every 128 ms, which are received by their neighbouring robots with a delay of 128 ms. The bearing accuracy of the relative positioning system is modelled with an additive Gaussian noise of standard deviation of $\sigma = 0.2$ rad, while the distance has a standard deviation 15%.

The sets of experiments with real robots are as similar as possible to those in the simulation in order to compare them. In the simulations, the robots perform a complete cycle of the algorithm in a fixed period, but in reality this period varies from cycle to cycle and from robot to robot. This period was measured in several experiments and was around 200 ms.

Real robot experiments were recorded using an overhead camera for a later extraction of the robots’ positions with a periodicity of 40 ms. A white marker was placed on top of each robot in order to track them. The videos were processed using the SwisTrack tracking tool (Lochmatter et al., 2008).
As a result of non-perfect segmentation of the images and the effect of discretization given by pixels, noise appears in the output data. The robots' headings can not be extracted since the robots are marked with a single circle. As a consequence MOE cannot be calculated and analysed in the real robot experiments. The duration of experiments is variable in the sets with real robots, since the robots eventually moved out of the camera's range of vision.

3.1 Joining Experiments

In this section, the joining capability of the framework is tested in three sets of experiments. One set of experiments (Set1-40-sim) was carried out using 40 simulated robots, with the aim of demonstrating the scalability with an increasing number of robots. Another set (Set1-8-sim) makes use of only eight simulated robots in order to compare it with experiments of the third set (Set1-8-real) in which only eight real robots are present. A summary of the sets is presented in Table 1.

In every set experiment of these three sets, robots are placed in two separated groups facing one another with an angle of \( \pi/2 \). In the simulation, the robots are initially placed to form a square lattice with a 0.5 m distance between robots.

In Figure 5(a), the robots' trajectories are for an experiment of Set1-40-sim. As can be seen, the robots are placed in two different groups and after a certain period of time they found each other and join. In the final positions all the robots form a compact group and they are placed forming a hexagonal lattice. The graphs show the metrics plotted for the two groups of robots for \( t<20 \) s, and as a single group for \( t>20 \) s. The group speed is shown in Figure 5(b) being almost constant except for a high peak just before joining, and slightly lower than the average during the 10 s after it. This is due to the joining process taking place. Mean orientation error (Figure 5(c)) is very low in the beginning since the robots are initially aligned. When the joining process takes place the value increases, later decreasing at a slow rate. Mean distance error (Figure 5(d)) decreases with time while the robots try to occupy the desired positions among them.

---

**Figure 5.** Relevant metrics and robots' trajectories shown for an experiment of Set1-40-sim, where the joining function is tested in simulation.
Table 2. Relevant statistics of experiments performed in Set1-8-sim, Set1-40-sim, Set1-8-real, Set2-8-sim, Set2-40-sim and Set2-8-real.

| Statistic         | Set1-8-sim 8 robots | Set1-40-sim 40 robots | Set1-8-real 8 robots | Set2-8-sim 8 robots | Set2-40-sim 40 robots | Set2-8-real 8 robots |
|-------------------|---------------------|-----------------------|---------------------|---------------------|-----------------------|---------------------|
| Ave. GS (m/s)     | 0.181               | 0.168                 | 0.166               | 0.173               | 0.159                 | 0.156               |
| Std. dev. GS (m/s)| 0.025               | 0.014                 | 0.039               | 0.015               | 0.024                 | 0.032               |
| Ave. GS / Ideal GS| 1.016               | 0.943                 | 0.931               | 0.971               | 0.892                 | 0.875               |
| Ave. MDE (m)      | 0.042               | 0.088                 | 0.111               | 0.032               | 0.074                 | 0.084               |
| Std. dev. MDE (m) | 0.025               | 0.023                 | 0.034               | 0.023               | 0.026                 | 0.030               |
| Max. MDE (m)      | 0.161               | 0.126                 | 0.281               | 0.135               | 0.142                 | 0.231               |

Figure 6. Relevant metrics and robots’ trajectories shown for an experiment of Set1-8-real, where the joining function is tested with real robots.

The five experiments with real robots of Set1-8-real allow validating in reality the results from the simulations and comparing them. The duration of the experiments is variable since the robots can exit the area covered by the tracking system. In Figure 6(a), the robots’ trajectories for an experiment of Set1-8-real are shown. The robots are placed in two different groups and after some time they found each other and joined, forming a compact group. Group speed of the same experiment is plotted in Figure 6(b). While there are two different groups, speed is slightly higher than the ideal group speed, this is due to the joining process taking place. After that the group speed takes values of around the ideal speed. Mean distance error is plotted in Figure 6(c). This is small during the whole experiment.

The relevant statistics for group speed and mean distance error for all the sets are summarized in Table 2. They are calculated for each set for all the experiments for $t > 5$ s. In the three sets where joining is studied, group speed is very similar and close to the ideal group speed. This is slightly larger than the ideal speed in Set1-8-sim which can be due to the acceleration during the joining process. Mean distance error is very low in these three cases. It is larger when experimenting with 40 robots (Set1-40-sim) than with eight in simulation (Set1-8-sim). The reason is that for 40 robots it takes them some time to occupy the desired positions. This is also larger with real robots (Set1-8-real) than in simulation (Set1-8-sim).

3.2 Splitting Experiments

The splitting capability of the framework is presented in this Section. As in the joining experiments, three different sets of experiments were performed: two in simulation using eight (Set2-8-sim) and 40 (Set2-40-sim) robots, and a third one using eight real robots (Set2-8-real). This allows demonstrating the scalability with an increasing number of robots, to prove the functionality in reality and also to compare real robot experiments with simulations. In all the
sets, the robots use the *Splitting Control Module* and the *Splitting Control Module* to create the splitting functionality. The simulation sets consist of 30 experiments of 60 s, while Set2-8-real is composed of five experiments of variable duration. All the sets run the same combination of modules and parameters. They are summarized in Table 1.

In all the sets of experiments the robots are initially placed forming a square lattice with a 0.5 m distance between robots and with randomly uniformly distributed headings. The splitting trigger takes place at \( t = 8 \) s and the robot in charge of triggering the split is placed approximately in the centre of the group.

In Figure 7(a), the robots’ trajectories for an experiment of Set2-40-sim are shown. As it can be seen, the robots start moving together and after some seconds they split into two compact groups. Splitting is detected by the algorithm in charge of deciding how to group robots in order to plot them and calculate statistics at \( t = 23 \) s. This indicates that it takes the robots 15 s to completely split. The final positions of the robots form perfect hexagonal lattices in both sub-groups. The evolution of group speed is plotted for the same experiment in Figure 7(b).

Group speed is almost constant and of around 0.17 m/s (ideal) during the whole experiment. It is only lower for \( 10 < t < 23 \) s, that is when the group is splitting, but the separation into two groups is not yet detected by the algorithm in charge of calculating the metrics. Mean orientation error, shown in Figure 7(c), is low except when splitting is taking place since the two groups of robots are heading in two different directions. Mean distance error, shown in Figure 7(d), decreases during the whole experiment with time, reaching very low values in the end.

The robots’ trajectories for a splitting experiment with real robots (Set2-8-real) are plotted in Figure 8(a). It can be seen how the robots after splitting end in two differentiated groups. Group speed, shown in Figure 8(b), is very close to the ideal value of 0.17 m/s during the whole experiment. Mean distance error is plotted in Figure 8(c). This is low during the whole experiments with high peaks in the beginning of the experiment and during the splitting process.

As in the joining sets of experiments, the metrics for group speed and mean distance error are presented in Table 2 for the sets of experiments of splitting. They are calculated for each set for all the experiments for \( t > 5 \) s. The results in all the cases are successful. \( \text{Ave.GS}/\text{IdealGS} \) is for the three sets of around 1. This is worse in Set2-40-sim and Set2-8-real than in Set2-8-sim. Average of mean distance error is very low in the three sets.

![Figure 7](image-url)  
(a) Trajectories of the robots during an experiment of Set2-40-sim, where the splitting function is tested in simulation. The initial positions of the robots are marked with a small circle (bottom). The final positions, where robots are split into two groups (top left and top right), are marked with a square.

![Figure 7](image-url)  
(b) Group speed for an experiment.

![Figure 7](image-url)  
(c) Mean orientation error for the same experiment.

![Figure 7](image-url)  
(d) Mean distance error for the same experiment.

Figure 7. Relevant metrics and robots’ trajectories shown for an experiment of Set2-40-sim, where the splitting function is tested in simulation.
The joining and splitting capabilities of the framework have been successfully tested both in simulation and using real robots. Experiments with real robots were carried out using eight Khepera III robots, demonstrating that it works correctly in reality. In simulation, up to 40 Khepera III robots were used, demonstrating experimentally the scalability with an increasing number of robots.

The inclusion of the new Splitting Control Module proves that the framework is expandable to allow new functionalities.

The joining capability works in good conditions and groups of robots start to join even if the robots are far from each other thanks to the inclusion of robots of type 2. The proposed framework can be augmented by adding new modules or by developing some of the already described ones to increase their autonomy. New modules could allow for more complicated tasks such as odour search or mapping.

The Splitting Control Module can be augmented to decide when and in which place in the group the splitting should take place. This must be done according to the information read by the robot’s sensors and shared among them. The decision can be done in a distributed manner, maybe making use of a consensus algorithm.

In addition, the framework could be modified to cope with movement in three dimensions. The basic workings of the framework would be the same. The Flocking Controller should be modified to allow 3-dimensional vectors. New modules in charge of modifying some of the parameters of the new Flocking Controller should be implemented. Obstacle Avoidance Module should cope with modifying the group direction with 3-dimensional vectors. Splitting Control Module should modify the way of determining the group, for instance by dividing the sub-groups with a plane instead of using a line.

5. References

[1] Antonelli, G., Arrighiello, F. & Chiaverini, S. (2008). Flocking for multi-robot systems via the null-space-based behavioral control, Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE Press, Piscataway, NJ, pp. 1409–1414.
[2] Balch, T. & Arkin, R. (1998). Behavior-based formation control for multi-robot teams, IEEE Transactions on Robotics and Automation 14(6): 926–939.
[3] Barfoot, T. D., Christopher M. Clark, Rock, S. M. & D’Eleuterio, G. M. T. (2002). Kinematic path-planning for formations of mobile robots with a nonholonomic constraint, Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Vol. 3, IEEE Press, Piscataway, NJ, pp. 2819 – 2824.

[4] Çelikkanat, H. & S.ahin, E. (2010). Steering self-organized robot flocks through externally guided individuals, Neural Computing and Applications 19(6): 849–865.

[5] Chaimowicz, L. & Kumar, V. (2004). Aerial shepherds: Coordination among uavs and swarms of robots, 7th International Symposium on Distributed Autonomous Robotic Systems, Toulouse.

[6] Ferrante, E., Turgut, A. E., Mathews, N., Birattari, M. & Dorigo, M. (2010). Parallel Problem Solving from Nature, PPSN XI, Springer-Verlag, Berlin, Germany, chapter Flocking in Stationary and Non-stationary Environments: A Novel Communication Strategy for Heading Alignment, pp. 331–340.

[7] Kelly, I. D. & Keating, D. A. (1996). Flocking by the fusion of sonar and active infrared sensors on physical autonomous mobile robots, Proceedings of the The Third International Conference on Mechatronics and Machine Vision in Practice, pp. 14–17.

[8] Lochmatter, T., Roduit, P., Cianci, C., Correll, N., Jacot, J. & Martinoli, A. (2008). SwisTrack - a flexible open source tracking software for multi-agent systems, IEEE/RSJ 2008 International Conference on Intelligent Robots and Systems (IROS 2008), IEEE, pp. 4004–4010.

[9] Michel, O. (2004). Webots: Professional mobile robot simulation, Journal of Advanced Robotics Systems 1(1): 39–42.

[10] Naffin, D. J. & Sukhatme, G. S. (2004). Negotiated formations, International Conference on Intelligent Autonomous Systems, pp. 181–190.

[11] Navarro, I. & Matía, F. (2009). A proposal of a set of metrics for collective movement of robots, Proceedings of Workshop on Good Experimental Methodology in Robotics, Robotics Science and Systems 2009.

[12] Navarro, I. & Matía, F. (2011). A framework for collective movement of mobile robots based on distributed decisions, Robotics and Autonomous Systems 59(10): 685–697.

[13] Navarro, I. & Matía, F. (2012). Distributed orientation agreement in a group of robots, Autonomous Robots 22(4): 445–465.

[14] Navarro, I. & Matía, F. (2013). A survey of collective movement of mobile robots, International Journal of Advanced Robotic Systems 10.

[15] Ögren, P. (2004). Split and join of vehicle formations doing obstacle avoidance, Proceedings of the 2004 IEEE International Conference on Robotics and Automation.

[16] Pugh, J., Raemy, X., Favre, C., Falconi, R. & Martinoli, A. (2009). A fast on-board relative positioning module for multi-robot systems, IEEE/ASME Transactions on Mechatronics, Focused Section on Mechatronics in Multi Robot Systems 14(2): 151–162.

[17] Spears, W., Spears, D., Hamann, J. & Heil, R. (2004). Distributed, physics-based control of swarms of vehicles, Autonomous Robots 17: 137–162.

[18] Turgut, A. E., Çelikkanat, H., Gökçe, F. & Şahin, E. (2008). Self-organized flocking in mobile robot swarms, Swarm Intelligence 2(2-4): 97–120.