2008 IEEE International Conference on
Robotics and Automation
Pasadena, CA, USA, May 19-23, 2008

Robot formations: robots allocation and leader–follower pairs

Sérgio Monteiro Estela Bicho
Department of Industrial Electronics
University of Minho
4800–058 Azurém, Portugal
{sergio,estela}@dei.uminho.pt

Abstract— **In this paper we focus on the problem of assigning robots to places in a desired formation, considering random initial locations of the robots. Since we use a leader–follower strategy, we also address the task of choosing the leader to each follower. The result is a *formation matrix* that describes the relation between the robots and the desired formation shape. Simple algorithms are defined, that are based on the minimization of the distances of robots to places in the formation. All these algorithms are implemented in a decentralized way. We assume that communication is possible, but the requirements are of very–low bandwidth.**

I. INTRODUCTION

The problem of controlling a set of cooperating mobile robots is very important because of their applications in real scenarios. The transportation and manipulation of objects [1], [2], coverage and exploration of specific environments [3], [4] or localization and mapping [5], [6] are some of the tasks where the researchers have been focusing. Another important task is when the set of robots should navigate according to a prescribed geometric shape, in what is known as *formation control*. Several solutions have been proposed to this problem (e.g. [7], [8], [9], [10]). Yet, usually, authors focus their attention on path planning related issues, and neglect the formation initiation task.

This work appears as a consequence of our previous work [11], [12], [13], where we presented a framework based on the attractor dynamics approach, using a leader–follower strategy, that is able to stabilize and maintain a team of robots navigating according to a prescribed formation shape. Some of the key features of our work are: i) the ability to stabilize a desired formation from any initial state; ii) obstacle (either static or moving) avoidance; iii) implicit formation split–and–join (that can occur in the presence of obstacles); iv) commanded formation switches. Here we extend that work with the capability of automatically allocate robots to places. More specifically, considering a team of N robots that has a pre–assigned team leader (called the *lead robot*), to which is communicated a target location and a desired formation shape, we formulate the following two problems: a) which robot should be allocated to which place in the formation? b) how to construct the *leader–follower* hierarchy?

The considered assumptions are: all robots are able to communicate with the *lead robot* (for the centralized implementation) or with all other robots (in the distributed implementation); each robot is able to measure the distance and relative bearing to the *lead robot* (during the allocation process) and to its leader (during mission execution). No absolute reference frame is required.

The rest of the paper is structures as follows: Section II presents some of the relevant related work; in Section III we introduce the formation matrix, which is where we capture the formation description; the generation of this matrix is the subject of Section IV, while results are presented in SectionVI; in Section V we adress the topic of formation robustness and finalize presenting our conclusions and future work in Section VII.

II. RELATED WORK

The first problem, i.e. given a desired formation configuration, which robot to allocate to which position in the formation is of growing interest. It has been studied and few solutions have been proposed. Michaud et al. [14], for instance, used a cost function dependent on the distance between the robots. All the robots run the same allocation algorithm, as if they were team leaders. The one that reaches the smallest cost has its allocation assigned to the formation. Fredslund and Mataric [15] assigned robots to places, following an algorithm based on the robots ID. Since all robots know the same algorithm and have different IDs then they'll assign themselves to different positions in the formation. The work by Kosternik et al. [16] is, in general, similar to the previously described, but it adds *social roles* to the robots in the formation. These roles characterize the location in the formation (either to the left or right of the leader). It also adds a chain of communications (from followers to leaders) that ensures the leader with the complete knowledge of the formation, and it enables it to give orders to its followers (to balance the formation, for instance). In Brimble and Press [17] each robot negotiates with the others the allocation of a specific station (place within the formation), searching to minimize one of two costs: either total distance or maximum distance traveled. Two types of negotiation are also introduced: a pairwise one (only two robots negotiate each time) and a recursive one (a robot "consults" the others before deciding). This problem is also tackled by Gold et al. [18]. Each robot has information about the nearest target positions in the formation and the nearest robots. The decision is taken using two utility functions in a cost–benefit approach. A set of options, where the benefit is higher than the cost, emerges from this approach (the satisficing set). Any of these options can then be used.

When all the robots are allocated to the formation, the second problem arises (when using a leader–follower strategy): which robot should a follower follow? Or, should it follow more than one robot? Fierro and Das [19] present an algorithm that given an assigned leader and a desired formation geometry, it generates the leader–follower hierarchy of the entire formation. Their algorithm tries to minimize the path between the leader and the follower, and takes into account the sensor visibility of each robot. Kaminka and Glick [10] also focus on this problem, from the sensor usage efficiency perspective.

This paper has the following contributions: an algorithm (with a centralized and a distributed version) that allocates robots to places in a formation, given the desired shape and team–leader, that exhibits as advantages, due to its simplicity, the low requirement on exchanged messages (communication); an algorithm that enables a follower to choose its leader, that is tailored to our formation control framework.

III. Formation matrix

Given a desired formation shape that a number of robots should assume, it is necessary to translate it into a suitable internal representation that captures the desired pattern. In principle, to characterize a formation it is enough to state the number of places in it (with each place corresponding to one robot), how they relate to each other in terms of distance and orientation (since we follow a leader–follower strategy, it is necessary to state which is (are) the leader(s) to each follower), and which is the *lead robot*, i.e. the robot that “drags” the formation when it moves.

Figure 1 shows a series of possible formations, where the darker robot was chosen as the *lead robot*. The doted lines between the robots try to give an idea about the shape of the formation. We notice in that figure that almost any given formation shape can be outlined by a polygon. We can also identify intuitively some sort of dependence of a given robot to the others (at least some of the others) around it.

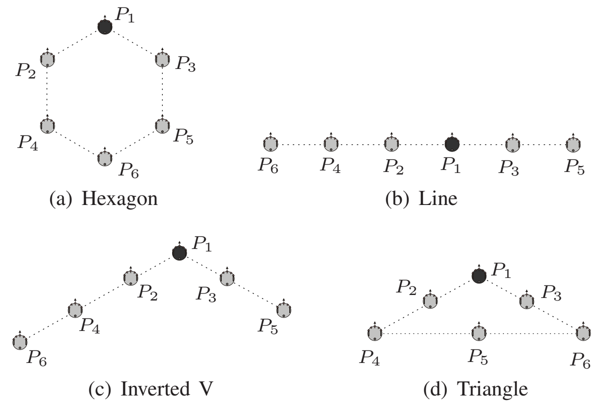


Fig. 1. Examples of possible formation shapes. The *lead robot* is represented by the darker circle.

In order to represent the information above, the complete team specification is described by means of a *formation matrix* [11] as follows

$$\mathbf{F} = \begin{pmatrix} L_1 & \Delta\psi_{1,d} & l_{1,d} \\ L_2 & \Delta\psi_{2,d} & l_{2,d} \\ \dots & \dots & \dots \\ L_N & \Delta\psi_{N,d} & l_{N,d} \end{pmatrix} \tag{1}$$

This matrix codes the shape of the formation in the following way: Row i $(= 1, 2, 3, ..., N)$ defines the pose of robot R_i in the formation. It is a vector $\mathbf{F}_i = \begin{pmatrix} L_i & \Delta\psi_{i,d} & l_{i,d} \end{pmatrix}$, where L_i $(L_i \neq R_i)$ identifies the *leader* robot for robot R_i, $\Delta\psi_{i,d}$ is the desired relative angle between robot R_i and its *leader* and $l_{i,d}$ the desired distance to its *leader*.

When robot R_i is the *lead robot* the parameters for its dynamics are $L_i = 0$, $\Delta\psi_{i,d} = 0$ and $l_{i,d}$ defines the distance at which it must stop from the target location.

For example, one formation matrix that determines the shape of a hexagon formation is $\mathbf{F_{hex}}$ in Figure 2.

$$\mathbf{F_{hex}} = \begin{pmatrix} 0 & 0 & 150 \\ 1 & -\pi/3 & 150 \\ 4 & 0 & 150 \\ 1 & \pi/3 & 150 \\ 1 & 0 & 300 \\ 3 & -\pi/2 & 260 \end{pmatrix}$$

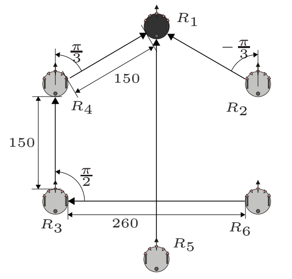


Fig. 2. Hexagon formation as determined by $\mathbf{F_{hex}}$.

In $\mathbf{F_{hex}}$, we assume that Robot R_1 is the *lead robot* (i.e. moves toward the target location), and that the desired distance between the robots is 150 cm. Considering two robots as an example we can see (in line 2) that robot R_2 follows R_1 on the left side and maintaining an oblique formation ($L_2 = 1$, $\Delta\psi_{2,d} = \pi/3$, $l_{2,d} = 150$) and that (in line 5) robot R_5 follows robot R_4 maintaining a line formation on the right ($L_5 = 4$, $\Delta\psi_{2,d} = -\pi/2$, $l_{5,d} = 150$). Figure 2 shows a representation of the referred hexagon pattern.

It is important to note that there are many formation matrices that generate the same geometric configuration for the formation. By proper manipulation of the $\mathbf{F}$ matrix, i.e. by changing its values, one can drive formation switches and cope with robot failure.

IV. Generation of the formation matrix

The shape of the formation and the leader–follower hierarchy for the complete team are both described by the formation matrix. When executing a mission all the robots in the team must have knowledge of this matrix. This is mainly for backup reasons in case of robot failure, because when actually running the mission, each robot just needs to know about its leader.

The formation matrix is generated in three distinct situations: i) prior to mission start, right after deploy; ii) by instruction of the *lead robot*, which as been ordered (by an higher level entity) a formation shape switch; iii) when one of the robots is found to be missing. In all situations, the generation of the *formation matrix* involves two steps: i) the allocation of robots to places in the formation and, after, ii) the definition of leader–follower pairs.

A. Allocation of robots to places

To solve i) in a distributed way, usually, there are two options: either by direct robot negotiation [17], [20] or by *robot identification* based assignment [15]. The advantage of the first option is that, depending on the negotiation effort and the allocation criteria, it can ensure optimal assignments. It has the drawback of requiring explicit communication and also with an increase in the number of agents the negotiation task can become overwhelming. By allocating robots to places based on robot ID, one overcomes the problem of the negotiation effort, because it is a quasi silent operation (explicit communication is only necessary to instruct the robots of the desired formation). The drawback is the rigidity of the assignment that does not take into account the actual location of the robots. Given some random initial configuration, assigning robots to places with this method can lead to highly suboptimal trajectories.

We will employ a method of allocation by negotiation, in an auction like process. We assume that the *lead robot* is assigned a priori and is the only one aware of the destination target. The *lead robot* is also the one with initial knowledge of the desired formation, that is mapped into a *shape matrix*, $\mathbf{S}$. This *shape matrix* can be communicated by a higher level entity, or constructed by the *lead robot* given some task constraints (not discussed in this paper). It assumes the following form:

$$\mathbf{S} = \begin{pmatrix} 0 & 0 \\ l_2 & \psi_2 \\ ... & ... \\ l_N & \psi_N \end{pmatrix} \qquad (2)$$

where row j, with $j = 2, 3, ...N$, describes the place P_j in the formation. l_j and ψ_j are the distance and orientation of place P_j taking the *lead robot* as reference. Since each row in the matrix relates to one place in the formation, thus it should have as many rows as there are places to fill in the formation, i.e. N. Place P_1 belongs to the *lead robot* and, as such, $l_1 = 0$ and $\psi_1 = 0$. Another rule to build the shape matrix is that places with lower IDs should be closer to the leader, in terms of *vertical distance*, than places with higher IDs (where vertical distance is $l_j \cos\psi_j$). We enforce this rule because it helps to speed up the algorithm of controller assignment. Figure 3 shows an example of a *shape matrix* together with its representation. An hexagon was chosen as example.

$$\mathbf{S_{hex}} = \begin{pmatrix} 0 & 0 \\ l_2 & \psi_2 \\ l_3 & \psi_3 \\ l_4 & \psi_4 \\ l_5 & \psi_5 \\ l_6 & \psi_6 \end{pmatrix}$$

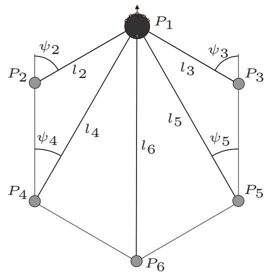


Fig. 3. Example of a *shape matrix* for an hexagon formation. The place of index *i* is described by row *i*. The first place always belong to the leader.

The place allocation algorithm is based on the distributed computation of a cost function and subsequent negotiation with the team mates. Because each robot needs to know the *shape matrix*, prior to executing the algorithm, the *lead robot* broadcasts it to all the robots. After, each robot, R_i, computes the distance separating it from each place, P_j, in the formation, according to the following equation:

$$\mathbf{D}_{i,j} = \sqrt{l_j^2 + l_{i,l}^2 - 2 l_j d_{i,j} \cos\left(\psi_j - \phi_l + \pi/2\right)} \qquad (3)$$

where l_j and ψ_j are directly extracted from the row j of the shape matrix $\mathbf{S}$, $l_{i,l}$ is the actual distance to the *lead robot*, and ϕ_l is the *lead robot's* heading, in the *follower's* reference frame. If the *lead robot's* heading is unknown to the follower, then it can use its own heading direction instead. This is a reasonable assumption because, while executing a mission, the robots move in the same direction with headings approximately equal. The problem lies at mission start, right after deploy, when the robots can have completely random heading directions.

At this moment we have a distributed matrix of distances between robots and places, $\mathbf{D}$, with each row located in different robots. Based on this matrix, our purpose is to assign to each place the robot that is closest to it, that is not already assigned. Here two alternatives are possible: i) either all robots communicate their entry in matrix $\mathbf{D}$ to one robot that is responsible for the assignment (the *lead robot* for instance); in this case only one robot executes the allocation algorithm for all the robots and at the end it communicates results to the team; ii) or the whole team engages in an auction biding for places in the formation.

Algorithm 1 shows the procedure for centralized assignment. This algorithm departs from the complete $\mathbf{D}$ matrix and searches for the robot closest to each place. Lines 1 and 2 of the algorithm serve to remove place P_1 and robot R_1 of possible assignment (we remove robots and places by increasing the corresponding cost to infinity), as these are already allocated (are the place leader and the *lead robot*). Then, a cycle to the remaining places is initiated. At every iterations, from the set of the not yet allocated robots and places, we find which is the pair that is closest to one another (line 4). That pair is the robot identified by the row index corresponding to the minimum value in the $\mathbf{D}$, while the place correspond to the column index of the same value. The selected robot is allocated to the selected place (line 7), and the pair is removed from the list of unassigned robots and places. The result is an allocation matrix, $\mathbf{A}$, with as many lines as there are robots, and with as many columns as there are places. If the robot R_i is allocated to place P_j, then, $\mathbf{A}_{i,j} = 1$, while the other elements in the same row and column all equal 0. The allocation process is terminated by the broadcast of $\mathbf{A}$ to all other robots in the team.

When, instead of a running the algorithm on a single robot, the choice is to have an auction by all the team, each agent follows the procedure in algorithm 2.

Algorithm 1 Allocation of robots to places – centralized

```
1: $\mathbf{D}_{\mathbf{row}\#\mathbf{1}} \leftarrow \infty$
2: $\mathbf{D}_{\mathbf{col}\#\mathbf{1}} \leftarrow \infty$
3: for $k = 2$ to $N$ do
4:     $i, j \leftarrow \text{index}(\min(\mathbf{D}))$
5:     $\mathbf{D}_{\mathbf{row}\#\mathbf{i}} \leftarrow \infty$
6:     $\mathbf{D}_{\mathbf{col}\#\mathbf{j}} \leftarrow \infty$
7:     $\mathbf{A}_{i,j} \leftarrow 1$
8: end for
```

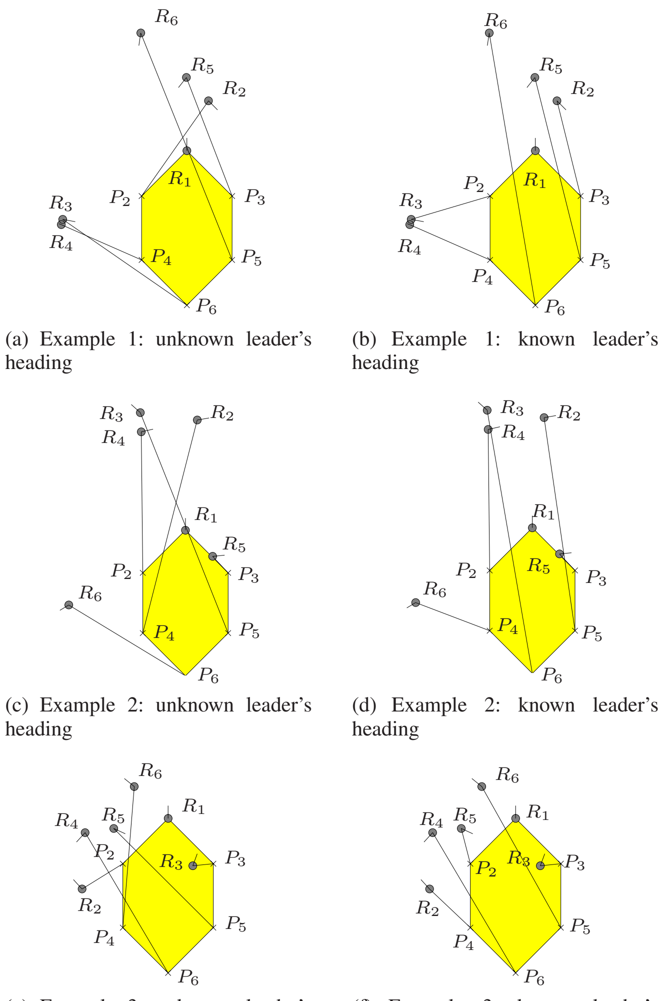
Algorithm 2 Allocation of robots to places – distributed

```
1: $j \leftarrow \text{index}(\min(\mathbf{D}_i))$
2: SEND (i,j,$\mathbf{D}_{i,j}$)
3: while not received all messages from all robots do
4:     $\mathbf{D}_{i,j}$ ←RECEIVE_MESSAGE
5: end while
6: if min(D)=$\mathbf{D}_{i,j}$ then
7:     $\mathbf{A}_{i,j} \leftarrow 1$
8:     SEND (i,j,$\mathbf{D}_{i,j}$)
9:     remove itself from negotiation
10: else
11:     $\mathbf{D}_{k,m}$=RECEIVE_MESSAGE
12:     $\mathbf{A}_{k,m} \leftarrow 1$
13: end if
```

This algorithm can be seen as a distributed implementation of the previous one. At each iteration, each robot selects the place to which is closest, and broadcasts that information to the team mates. The message consist of the emitter robot ID, place ID and distance to that place. At the same time it also listens to the other robots communicated information. After all robots have informed the team mates, the robot with lower distance to place assigns itself to that place, informs the team mates, as confirmation, and steps out of the auction process. The remaining robots remove that place from their list (by assigning an infinite distance to it), proceed the auction until there are no more places to assign.

For the same distance matrix, $\mathbf{D}$, and shape matrix, $\mathbf{S}$, the generated allocation is always the same, and is independent of the used algorithm. If inter–robot communication is to be minimized, then alg. 1 should be used. Else, if agent autonomy is a requirement then one should go for alg. 2.

Figure 4 shows examples of the allocation results when using the previous algorithms (both the centralized and the distributed algorithms have exactly the same result, i.e. generate the same allocation) in an hexagon shape. The initial pose of each robot is set randomly. We ran the algorithm twice: first by assuming that the leader's heading is the same as the one of the robot computing the cost, and secondly, by assuming that each robot is able to determine the exact leader's heading direction. As expected the results when the leader's heading is known are much better, as the overall number of trajectory crossings decreases, and the overall distance to traverse also decreases, thus inducing faster stabilization times.



(a) Example 1: unknown leader's heading
(b) Example 1: known leader's heading
(c) Example 2: unknown leader's heading
(d) Example 2: known leader's heading
(e) Example 3: unknown leader's heading
(f) Example 3: known leader's heading

Fig. 4. Examples of robot allocations. Robots are depicted by circles with a dash indicating the initial heading. Places are depicted by crosses. A line connecting R_i to P_j indicates the pair assignment. The desired shape is an hexagon. Three initial (random) situations are presented.

B. Definition of leader–follower pairs

After the process of allocating robots to places, it is now time for each robot to select which will be its leader. At this moment, each robot possesses information about the formation shape and also knows which robot is in each place. The procedure each robot follows to select its leader, is based on choosing the leader from the set of eligible leaders. The set of eligible leader's is the set of robots, to the front and sides of the *follower*, but distant from it no more than d_{aloc}. This distance should be such to enable the set to contain at least one robot, and the larger it becomes it enables the followers to follow robots that are several levels above them. We limited this value to enforce the *followers* to choose a leader immediately at the above level. The selected *leader* is the one that causes the *follower* to follow it in a column formation, or closest to it. Algorithm 3 implements the described procedure.

As example, we show in figure 5 the assignment result for a formation with an hexagon shape. Independently of which

Algorithm 3 Algorithm for robot R_i to choose its leader.

```
1: $j \leftarrow$INDEX($\mathbf{A}_{row\#i} = 1$)
2: **if** $R_i$ at top level **then**
3:     $m \leftarrow j - 1$
4:     $k \leftarrow$INDEX($\mathbf{A}_{col\#m} = 1$)
5:     $dx \leftarrow (S_{j,1}\sin(S_{j,2}) - S_{m,1}\sin(S_{m,2}))$
6:     $dy \leftarrow (S_{j,1}\cos(S_{j,2}) - S_{m,1}\cos(S_{m,2}))$
7:     $\mathbf{F}_{i,1} \leftarrow k$
8:     $\mathbf{F}_{i,2} \leftarrow \arctan\frac{dy}{dx}$
9:     $\mathbf{F}_{i,3} \leftarrow \sqrt{dx^2 + dy^2}$
10: **else**
11:     $m \leftarrow j - 1$
12:     **while** $m \geq 1$ **do**
13:         $dx \leftarrow (S_{j,1}\sin(S_{j,2}) - S_{m,1}\sin(S_{m,2}))$
14:         $dy \leftarrow (S_{j,1}\cos(S_{j,2}) - S_{m,1}\cos(S_{m,2}))$
15:         $l \leftarrow 1$
16:         **if** $(\sqrt{dx^2 + dy^2 - d_{aloc}}) < 0$ **then**
17:             $k \leftarrow$INDEX($\mathbf{A}_{col\#m} = 1$)
18:             $\mathbf{leader\_set}_{l,1} \leftarrow k$
19:             $\mathbf{leader\_set}_{l,2} \leftarrow \arctan\frac{dy}{dx}$
20:             $\mathbf{leader\_set}_{l,3} \leftarrow \sqrt{dx^2 + dy^2}$
21:         **end if**
22:     **end while**
23:     $k \leftarrow$INDEX(min($\mathbf{leader\_set}_{col\#2}$))
24:     $\mathbf{F}_{i,1} \leftarrow \mathbf{leader\_set}_{k,1}$
25:     $\mathbf{F}_{i,2} \leftarrow \mathbf{leader\_set}_{k,2}$
26:     $\mathbf{F}_{i,3} \leftarrow \mathbf{leader\_set}_{k,3}$
27: **end if**
```

robot is at each place, the assignment (of leaders), in terms of places, will always be the same, i.e. in this example the robot that is allocated to place P_4, for instance, will always follow the robot that is allocated to place P_2, in a column configuration.

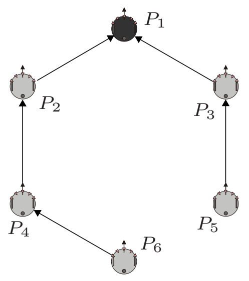


Fig. 5. The result of the controller assignment, following the procedure described in algorithm 3, for an hexagon formation.

The outcome of both these algorithms is a complete formation matrix as described by eq. 1 (including distances and relatives bearings).

V. Formation robustness

To guarantee robustness against robot failure, every robot is required to emit an *alive* signal (it can be a visual cue or a radio signal). Whenever a *follower* fails to receive its *leader* signal, for a predetermined amount of time, it sends an alarm message to the team requiring a formation update. Since initially, the number of places in the formation equals the number of robots in the team, the failure of one robot causes one of the places to be unattended. Depending on the mission instructions, the *lead robot* has three options: i) either continues the present shape, but with that place empty, ii) or commands a formation shape change, iii) or aborts the mission.

In the first scenario (option i)), only the robots that were following the failed robot need to modify their controller specification, by selecting another leader, i.e. they rerun algorithm 3 to update their entry in the formation matrix. The remaining robots are left out of this process.

If the choice is to change the shape of the formation (option ii)), then a complete new formation matrix has to be generated. This choice requires that the *lead robot* is able to produce a new formation shape, i.e., a new *shape matrix*. To produce this new shape matrix, the *lead robot* has to be supplied with sufficient knowledge at mission start. This knowledge comes in the form of *contingency plans*. These contingency plans can assume the form of different shape matrices (the follower can be informed of, for instance, three different shape matrices S_0, the original, S_1, when one robot is missing, and S_2 when two robots are missing), or the form of directives on how to construct a new shape matrix. Examples of directives can be, for instance: "distribute the places evenly along a circumference with radius r", or "produce a column where the distance between consecutive place is equal", or even "in case of failure of *n* robots, abort mission and return home". Another good example of formation directives, in our understanding, is the concept of *queues* as defined by [21].

The failure of one robot is conveniently treated by the described method in the previous paragraphs, unless the failing robot is the *lead robot*. In this case, a new *lead robot* has to be assigned and a new formation matrix has to be generated. Since every place is identified by an ID, and the place occupied by the *lead robot*, P_1, always is the first, in case of its failure the new team leader will be the robot in place P_2. This new *lead robot* needs to be informed about the mission specifications, in terms of target destination and contingency plans. Since the previous *lead robot* is '*dead*, it is not able to share the required information with the new one. To overcome this problem and to cope with the possibility of failure of more than one *lead robot*, the complete mission specification should be provided to all robots in the team as a backup strategy. During mission execution, only the *lead robot* makes use of it.

VI. Results

One important feature supported by our framework is the ability to perform ordered formations switches. Here we will describe a test where this feature is emphasized. The robots that compose the team have a differential drive model, and are equipped with 5 infra–red sensors for obstacle avoidance (located at the front of the robot with 30 deg spacing). They are assumed to be equipped with sensors that enable them to measure the distance and relative bearing to a given robot. Furthermore, when negotiating for a *formation matrix* it is assumed that all *followers* are able to see the *lead robot*. We will use a team of six robots placed at random initial locations (the initial status of the team can be seen in figure 6, at time instant 0). Four formation shapes are provided to the *lead robot*. These shapes are described by $\mathbf{S_{hex}}$, $\mathbf{S_{lin}}$, $\mathbf{S_v}$ and $\mathbf{S_{tri}}$, which are writen as follows:

$$\mathbf{S_{hex}} = \begin{pmatrix} 0 & 0 \\ 150 & \pi/4 \\ 150 & -\pi/4 \\ 277 & \pi/8 \\ 277 & -\pi/8 \\ 362 & 0 \end{pmatrix} \qquad \mathbf{S_{lin}} = \begin{pmatrix} 0 & 0 \\ 125 & \pi/2 \\ 125 & -\pi/2 \\ 250 & \pi/2 \\ 250 & -\pi/2 \\ 375 & \pi/2 \end{pmatrix}$$

$$\mathbf{S_v} = \begin{pmatrix} 0 & 0 \\ 150 & \pi/4 \\ 150 & -\pi/4 \\ 300 & \pi/4 \\ 300 & -\pi/4 \\ 450 & \pi/4 \end{pmatrix} \qquad \mathbf{S_{tri}} = \begin{pmatrix} 0 & 0 \\ 150 & \pi/4 \\ 150 & -\pi/4 \\ 300 & \pi/4 \\ 212 & 0 \\ 300 & -\pi/4 \end{pmatrix}$$

A sketch of those shapes is presented in figure 1. The *team leader* is ordered to move towards the target, starting in an hexagon formation ($\mathbf{F_{hex}}$ and figure 1(a)). After 36 sec of mission time, it should switch to a line shape ($\mathbf{F_{lin}}$ and figure 1(b)). It should navigate in line during 44 sec and then switch to an inverted V ($\mathbf{F_v}$ and figure 1(c)). After another 24 sec, again a formation switch is imposed. Now, the robots should stabilize a triangle formation ($\mathbf{F_{tri}}$ and figure 1(d)). The mission ends when the *lead robot* is in the neighborhood of the target.

The formation matrices generated (at each formation switch) are the following:

$$\mathbf{F_{hex}} = \begin{pmatrix} 0 & 0 & 150 \\ 1 & -\pi/4 & 150 \\ 1 & \pi/4 & 150 \\ 3 & 0 & 150 \\ 2 & -\pi/2 & 150 \\ 4 & 0 & 150 \end{pmatrix} \qquad \mathbf{F_{lin}} = \begin{pmatrix} 0 & 0 & 150 \\ 1 & -\pi/2 & 125 \\ 1 & \pi/2 & 125 \\ 3 & \pi/2 & 125 \\ 2 & -\pi/2 & 125 \\ 4 & \pi/2 & 125 \end{pmatrix}$$

$$\mathbf{F_v} = \begin{pmatrix} 0 & 0 & 150 \\ 1 & -\pi/4 & 150 \\ 1 & \pi/4 & 150 \\ 3 & \pi/4 & 150 \\ 2 & -\pi/4 & 150 \\ 4 & \pi/4 & 150 \end{pmatrix} \qquad \mathbf{F_{tri}} = \begin{pmatrix} 0 & 0 & 150 \\ 1 & -\pi/4 & 150 \\ 1 & \pi/4 & 150 \\ 3 & \pi/4 & 150 \\ 2 & -\pi/4 & 150 \\ 2 & \pi/4 & 150 \end{pmatrix}$$

Figure 6 depicts the simulated trajectory evolution of the team, using the control architectures described in [11]. Snapshots are provided at each time instant prior to formation change. Figure 7 shows each robot position error together with the average position error of the team (*formation*). This position error is the distance between the desired location of the robot and the actual location at which it is [22].

The mission ends with the team reaching its goal in the desired shape and with low formation error.

In summary, we have an approach that does not require an absolute reference frame ([14], [17] require it) by exploiting the fact that when traveling all robots have the same heading. We do require explicit communication between the robots, but the number of exchanged messages is fixed and dependent on the number of robots ,N: $2N$ (for the centralized implementation) or $\sum_{m=1}^{N} m$ (for the distributed

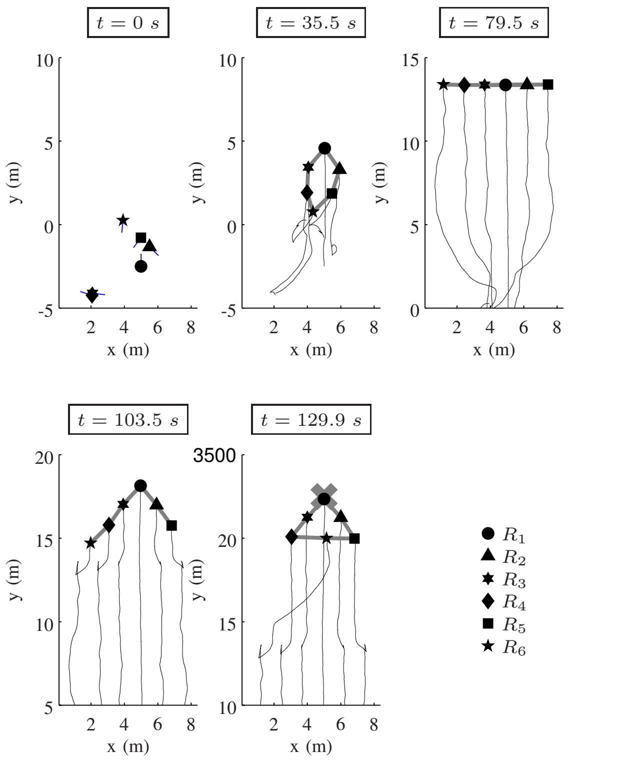


Fig. 6. Simulation of a team of six robots performing a mission where several formation switches occur.

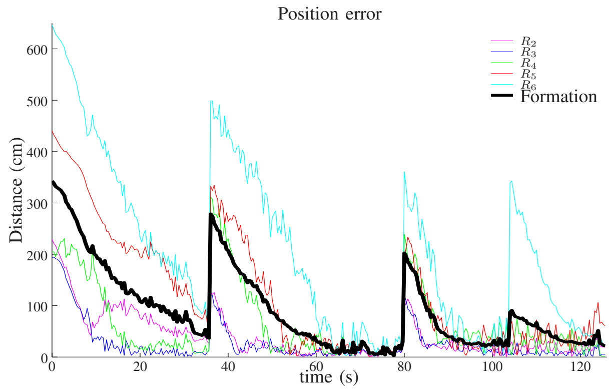


Fig. 7. Formation error of the experiment depicted in Figure 6.

implementation). On the contrary, in [17] this number is dependent on the quality of the result and can increase to very high values. In terms of required time to reach a solution, in our case is directly proportional to N. In [14] since it implements a search algorithm for all possible assignments, it will expectably take longer to conclude. So will the approach in [17] because of the negotiation process. The advantage of those is that no preassigned *lead robot* is necessary, and that with more complex alocation strategies probably better alocations might be accomplished. Another comparable solution is the one in [15]. As advantages it has the minimal required communication, no preassigned *lead robot* and the absence of absolute reference frames. Nevertheless the alocation strategy is very rigid (dependent on robots ID) leading to possible highly suboptimal allocations and also has problems with some formation switches (as the authors recognize in the switch from diamond to column formation).

VII. CONCLUSIONS AND FUTURE WORK

A. Conclusions

We have developed two algorithms (a centralized and a distributed version) for allocating robots to places in a formation, given its desired shape and team leader. These algorithms are based on a negotiation by auction. It do not achieve optimal assignments, nor is that its purpose. Instead we aimed at simple, easy to implement algorithms, with low–bandwidth requirements. Also, a decentralized algorithm for the definition of leader–follower hierarchy was proposed. Its outcome is a complete formation matrix, that stores information about the team hierarchy and shape. We have shown one simulation result, where these algorithms are used to drive four formation switches.

B. Future Work

Our plans for future work include the use of cognitive functions [23] to improve the algorithms of formation matrix generation. The purpose, here, is to guess and anticipate other team members position, as they move towards it, and in this way completely avoid the necessity for explicit communication.

REFERENCES

[1] N. Miyata, J. Ota, T. Arai, and H. Asama, "Cooperative transport by multiple mobile robots in unknown static environments associated with real–time task assignment," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 5, pp. 769–780, October 2002.

[2] R. Soares, E. Bicho, T. Machado, and W. Erlhagen, "Object transportation by multiple mobile robots controlled by attractor dynamics: theory and implementation," in *Proc. of the IEEE/RSJ Intl. Conference on Intelligent Robots and Systems, to appear*, 2007.

[3] W. Burgard, M. Moors, C. Stachniss, and F. Schneider, "Coordinated multi–robot exploration," *IEEE Transactions on Robotics*, vol. 21, no. 3, pp. 376–386, June 2005.

[4] J. Cortés, S. Martinez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 2, pp. 243–255, April 2004.

[5] F. Thomas and L. Ros, "Revisiting trilateration for robot localization," *IEEE Transactions on Robotics*, vol. 21, no. 1, pp. 93–101, February 2005.

[6] A. Howard, L. Parker, and G. Sukhatme, "Experiments with large heterogeneous team: exploration, mapping, deployment and detection," *The International Journal of Robotics Research*, vol. 25, no. 5–6, pp. 431–447, 2006.

[7] T. Balch and R. C. Arkin, "Behavior-based formation control for multirobot teams," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 6, pp. 926–939, December 1998.

[8] J. Desai, J. Ostrowski, and V. Kumar, "Modeling and control of formations of nonholonomic mobile robots," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 6, pp. 905–908, December 2001.

[9] T. Barfoot and C. Clark, "Motion planning for formations of mobile robots," *Robotics and Autonomous Systems*, vol. 46, pp. 65–78, 2004.

[10] G. A. Kaminka and R. Glick, "Towards robust multi–robot formations," in *Proc. IEEE Int. Conf. Robotics and Automation*, Orlando, FL, 2006.

[11] E. Bicho and S. Monteiro, "Formation control for multiple mobile robots: a non-linear attractor dynamics approach," in *2003 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Las Vegas, NV, October 27-31 2003, pp. 2016–2022.

[12] S. Monteiro, M. Vaz, and E. Bicho, "Attractor dynamics generates robot formations: from theory to implementation," in *Proc. IEEE Intl. Conference on Robotics and Automation*, New Orleans, LA, 2004.

[13] E. Bicho, A. Moreira, S. Diegues, M. Carvalheira, and S. Monteiro, "Airship formation control," in *3rd Int. Conf. on Informatics in Control, Automation and Robotics, in Workshop Multi-Agent Robotic Systems (MARS 2006)*, Setubal, portugal, August 1–5 2006.

[14] F. Michaud, D. Letourneau, M. Guilbert, and J. Valin, "Dynamic robot formations using directional visual perception," in *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, EPFL Lausanne, Switzerland, October 2002, pp. 2740–2745.

[15] J. Fredslund and M. Matarić, "A general local algorithm for robot formations," *IEEE Transactions on Robotics and Automation, special issue on Multirobot systems*, vol. 18, no. 5, pp. 837–846, October 2002.

[16] P. Kostelnik, M. Samulka, and M. Janosik, "Scalable multi–robot formations using local sensing and communication," in *Proc. of the Third Intl. Workshop on Robot Motion and Control*, November 9–1 2002, pp. 319–324.

[17] R. Brimble and J. Press, "Minimal cost robot formations," in *Proc. of the 11th International Conference on Advanced Robotics*, Coimbra, Portugal, June 30 – July 3 2003, pp. 1487–1495.

[18] T. Gold, J. Archibald, and R. Frost, "A utility approach to multi–agent coordination," in *Proc. of the Intl. Conference on Robotics and Automation*, San Francisco, USA, April 2000, pp. 2052–2057.

[19] R. Fierro and A. Das, "A modular architecture for formation control," in *Proc. of the Third Intl. Workshop on Robot Motion and Control*, November 9–11 2002, pp. 285–290.

[20] M. Lemay, F. Michaud, D. Létourneau, and J. M. Valin, "Autonomous initialization of robot formations," in *Proc. IEEE Int. Conf. Robotics and Automation*, New Orleans, LA, 2004.

[21] S. Ge and C. Fua, "Queues and artificial potentials trenches for multirobot formations," *IEEE Transactions on Robotics*, vol. 21, no. 4, pp. 646–656, August 2005.

[22] S. Monteiro and E. Bicho, "Attractor dynamics approach to formation control: theory and application," submitted.

[23] W. Erlhagen and E. Bicho, "The dynamic neural field approach to cognitive robotics," *Journal of Neural Engineering*, vol. 3, pp. 36–54, September 2006.