

---

# A General Approach to Swarm Coordination using Circle Formation

Karthikeyan Swaminathan<sup>1</sup> and Ali A. Minai<sup>2</sup>

<sup>1</sup> Department of Biomedical Engineering, University of Cincinnati, Cincinnati,  
Ohio 45221 [swamink@email.uc.edu](mailto:swamink@email.uc.edu)

<sup>2</sup> Department of Electrical & Computer Engineering and Computer Science,  
University of Cincinnati, Cincinnati, Ohio 45221 [aminai@ececs.uc.edu](mailto:aminai@ececs.uc.edu)

## 1 Summary

The field of collective robotics exploits the use of technologically simple robots, deployed in large numbers, to collectively perform complex tasks. Here, the real challenge is in developing simple algorithms which the robots can execute autonomously, based on data from their vicinity, to achieve global behavior. One such global task that many researchers (including the authors) have developed algorithms for is the formation of a circle. In this chapter, we discuss how the circle formation algorithm can be used as a means for solving other formation and organization problems in multi-robot systems. The idea behind this approach is that circle formation can be seen as a method of organizing the robots in a regular formation which can then be exploited. This involves identifying specific robots to achieve different geometric patterns like lines, semicircles, triangles and squares, and dividing the robots into subgroups, which can then perform specific group-wise tasks. The algorithms that achieve these tasks are entirely distributed and do not need any manual intervention. The results from these studies are presented here.

## 2 Collective Robotics

Collective robotics is the study of groups of relatively simple robots that are capable of moving around and accomplishing tasks collaboratively. The number of robots used can vary from tens to tens of thousands, based on the application. The goal is to use robots that are as simple — and therefore, as cheap — as possible, deploy them in large numbers and coordinate them to achieve complex tasks.

Groups of robots create a very complex coordination and control problem because of the extremely large configuration space created by numerous interacting agents. Centralized control methods are not feasible in this situation,

and the goal of much research in collective robotics has been to find efficient methods of distributed and decentralized control [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]. In particular, the idea of *self-organized control* arising from each robot/agent following its own set of behavioral rules has gained almost universal currency [1, 12, 13, 14, 15, 16]. In extremely large collections of robots, called swarms, the focus is on finding very simple behaviors at the robot level that still lead to the emergence of sophisticated behavior at the group level [18, 19, 20, 21, 22], but even with relatively small groups of robots, the self-organized approach has proved to be the most effective.

The key issue in collective robotics is how to specify rules of behavior and interaction at the level of individual robots such that coordination can be achieved automatically at the global level. This is called the coordination problem.

### 3 Pattern Formation in Collective Robotics

A fundamental problem in collective robotics is to have the group organize into global formations or patterns. These include simple patterns like circles, lines, uniform distribution within a circle or square, etc. In the presence of a central controller, these tasks are trivial, but this is not the case in a distributed system. The goal here is to have an entirely decentralized system in which each robot performs tasks *autonomously* based on information gathered by itself, preferably from its neighborhood. An important feature in most of these systems is that the individual entities (robots) are not explicitly aware that they are involved in pattern formation, and certainly cannot direct it globally. This also means that only a global observer can truly assess the performance of these systems, and the robots themselves must rely on limited — possibly incorrect — estimates to make their decisions. This last aspect has important implications for convergence in robot collectives and other decentralized systems.

The formation of patterns in multi-robot systems has several applications, some of which are listed below. These highlight the importance of pattern formation in the real world.

- Surrounding an object or feature in the environment. [23].
- Forming a uniform distribution of robots in a given area for protecting the area or surveillance. [23, 24].
- Election of a leader or follow-the-leader situations. [25, 26].
- Gathering to share information or for some other task. [25].
- Removal of mines or bomb disposal [27].
- Exploration and mapping in space, underwater, or in other hazardous environments. [28, 29].
- Formation of sensing grids. [30, 31].
- Managing processes in a manufacturing unit. [32].

- Carrying, moving and assembling objects. [33].
- Aiding emergency response and decongestion of traffic on highways. [34].

## 4 Background

This section briefly discusses the work of other researchers in the area of pattern formation. The reader is cautioned that the amount of research done in this field is immense and this section covers only a few papers that have directly influenced the work presented in this chapter. In particular, the formation of a circle has been studied by many. The work of Suzuki et al. stands out in this area [28, 35, 36]. Their multi-robot system (henceforth termed the Suzuki model) consists of around 50 robots, each of which is autonomous and mobile. Each is equipped with sensors that can detect the positions of *all* other robots instantaneously in the unlimited visibility case, and only its neighbors in the limited visibility case. Each robot is anonymous, meaning that it does not have any label, and executes the same algorithm as all other robots. The robots do not possess any communication capabilities and each has its own local coordinate system. In the unlimited visibility case, each robot senses the positions of all other robots and based on the observed coordinates, makes a move. Each robot determines its nearest and farthest robots and assumes the center of these robots to be the center of the circle to be formed and moves towards or away from it. All other robots similarly execute these simple tasks which eventually lead them to form a circle. In some cases, the original algorithm can lead to the robots forming a triangular shape instead of a circle, and a small correction was subsequently made to the algorithm by Tanaka [37] in order to rectify this. The group introduced a similar algorithm that helped the robots distribute themselves uniformly in a circle and a few more ways that needed manual intervention to help the system form a polygon, form a line and divide into groups. In the limited visibility case, they showed that the robots could converge to a point only when they were synchronous i.e. executed their tasks in unison.

In the Suzuki model, the different kinds of patterns can be formed only in the unlimited visibility case. When the application requires a large number of robots, the assumption that each robot can sense all other robots becomes unreasonable. A second group of researchers worked on a similar model of robots but explored the limited visibility case in detail. They showed that the robots could converge to a single point even in the asynchronous case, provided they had a sense of direction (possessed a compass) [25]. They analyzed the different shapes that are achievable by such a system, starting with the case when the robots have no common knowledge i.e. they do not share a common sense of direction or orientation, and increasing the knowledge step by step up to the case when they share everything [38, 27].

A refreshingly different approach to mobile multi-robot formation is provided by Gordon et al. [30, 31]. The robots here are actually Micro-Air Vehicles

(MAVs) that need to distribute uniformly in a hexagon or a square in order to form a sensing grid that can operate like a radar. The motion of the MAVs is governed by laws that mimic natural gravitational forces, which the authors term *artificial physics* forces. It is obvious that, in order to compute the virtual forces underlying their control, such robots need to have capabilities greater than those assumed in the Suzuki model.

Defago and Konagaya [39] present a method for circle formation based on extensions of the Suzuki model. This method consists of two algorithms executed one after the other. The first algorithm places the robots along the circumference of a circle and the second uniformly distributes the robots along the circumference. In the circle formation algorithm, the robots are initially in arbitrary positions. The goal of these robots is to determine the *smallest enclosing circle (SEC)* and then move to occupy positions along this circle. The SEC can be defined often by two, and at most three, extremal robots. The robots can have their own local coordinate systems, and hence different views of the environment, but the smallest enclosing circle (SEC) every robot computes for a given configuration of robots will be the same. However, the configuration of the robots changes as they move and hence it becomes necessary to ensure that the robots move in such a way that the SEC remains the same. To achieve this, the robots move according to certain rules based on Voronoi tessellation of the space around the robots [39]. Once all the robots take positions along the circumference of the SEC, the second algorithm is executed to spread the robots along the circumference. Every robot tries to move half the distance towards the mid-point of its nearest left and right robots. This method of circle formation is computationally complex because of the calculation of the Voronoi tessellation [39].

In order to reduce the complexity of the above algorithm, another method was developed by Markou et al. [40]. In this method, once the SEC is computed, a given robot determines the point on the circumference of the SEC, that is closest to it and moves towards this point. When all the robots have taken positions along the circumference, they spread along the circumference as described earlier [40].

Yun et al. [41] present a novel method for circle formation called the *Merge then Circle* algorithm. All the robots move initially towards the midpoint between their nearest and farthest robots. This, when executed for a long period of time, brings all the robots together in a cluster. After this step, the robots sense the positions of other robots and move in a direction of empty space for a distance equal to the radius of the desired circle. Once this is done, each robot uses the positions of its two nearest robots to move towards their midpoint for a more uniform distribution.

## 5 A Communication-Based Multi-Robot System

Most of the methods presented in the previous section are based on the Suzuki model of robots, which makes the assumption that each robot can sense the positions of all other robots irrespective of their distance from the given robot. This assumption becomes unreasonable when there are more than a handful of robots. A more practical model based on wireless communication has recently been developed by us [42], and is described below. Two circle-formation algorithms — Batch Broadcast of Coordinates (BBC) and Individual Broadcast of Coordinates (IBC) — have been developed for this model based on the algorithms in [28, 35, 37].

In the communication-based model, the robots are not equipped with sensors, but are assumed to possess transmitters and receivers, using which they can communicate messages to each other. The messages are mostly coordinate positions of the robots. However, the robots are assumed to be able to broadcast messages only over a small distance, analogous to the limited visibility condition in the Suzuki model. In this way, a robot can transmit its position to its neighbors directly and to distant robots through these neighbors by propagation (hops). The robots are anonymous in the sense that they cannot be distinguished from each other either by appearance or by the programs they possess. They are autonomous in that they do not need intervention from other robots, humans or external controllers. They are assumed to be able to move small but definite distances called *steps*. There are ( $N$ ) robots ( $R_1, \dots, R_N$ ) placed initially in random positions on a unit grid and loaded with the same programs which, when executed by each of them, lead to the formation of different patterns. Coordinates are shared through communication and the robots are assumed to follow a global coordinate system. This can be achieved by either pre-initializing the coordinates externally followed by path integration, or by the robots themselves setting up a self-organized coordinate system [43, 44, 38, 45]. The external initializer can be the same as the one that deploys the robots and is not needed during the system's pattern formation phase.

Using this model, and the circle formation algorithms (BBC and IBC) as the base, other algorithms were developed that exploit the arrangement of robots in a circle to solve other pattern formation and coordination problems in such a multi-robot system. This work is presented in the following sections.

## 6 Beyond Circle Formation - Formation of Other Patterns and Groups

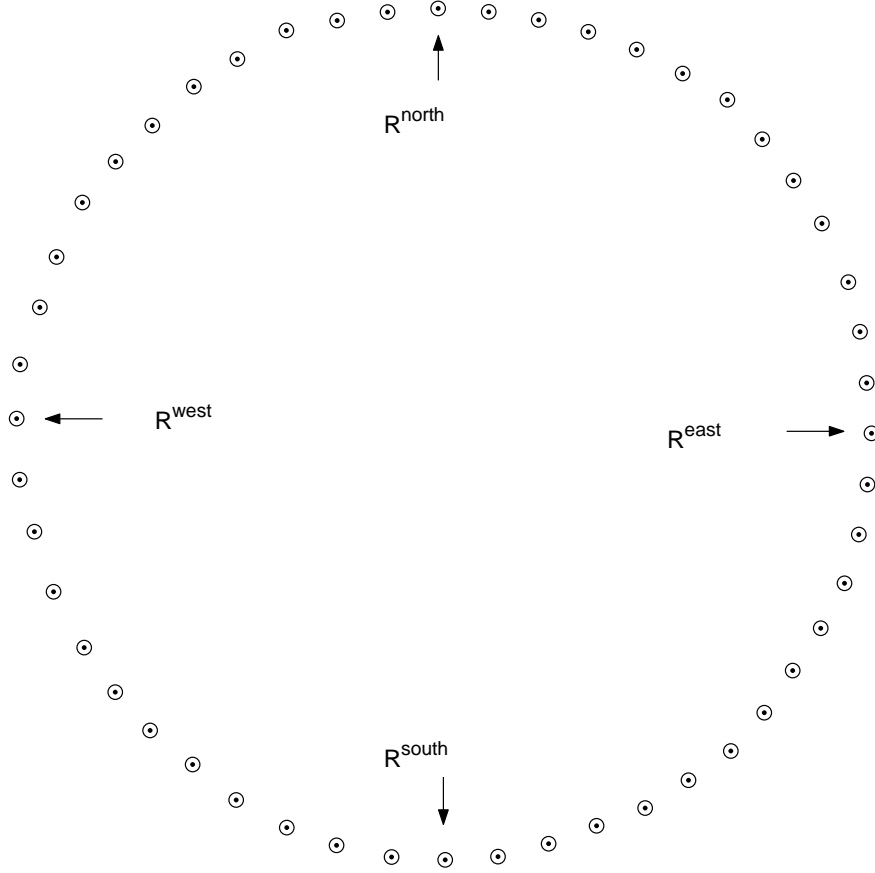
A circle is a very simple and perfectly symmetric geometric pattern, which is precisely why its formation is easy with oblivious and anonymous robots. The formation of other shapes is not so simple because they require designated vertices, orientations, etc. However, the ability to form a circle can be exploited

to form other shapes, and for achieving other coordination goals in multi-robot systems. Basically, forming a circle is a way of organizing an initially randomly scattered population of robots into a regular shape, which can then be used to localize and tag specific robots for special tasks. The goal, as always, is to do this in an entirely distributed manner. This section presents algorithms that allow robots to achieve non-circular formations without manual intervention, only using circle formation as the basic organizing principle. Robots can be pre-programmed so that they first form a circle and then switch to the rules for forming the desired shape. Alternatively, the robots could be switched to the algorithm for a particular shape through a one-time global message.

In this work, we consider the formation of a vertical line, a triangle, a square and a semi-circle. We also look at the canonical problem of splitting the robot population into groups of approximately equal size. The key problem in all these cases is to get groups of robots to follow *different* rules — corresponding to different roles — with constraints on the relative sizes of the groups. Since the robots are initially anonymous and randomly distributed, it is difficult to assign roles systematically. Distributed algorithms for limited role assignment (such as cluster formation, leader election or gateway designation in ad-hoc networks) do exist, but these typically assign roles relative to the *actual* distribution of agents — e.g., assigning clusters based on agent density — whereas in formation problems, the roles are assigned relative to a future *desired* distribution. Relaxation-style algorithms for these assignments can be developed, but it is difficult to guarantee their correctness, convergence, etc. The algorithms based on circle-formation, on the other hand, allow systematic role assignments in ways that are much easier to analyze. Thus, the proposed algorithms should not be seen as solving a previously unsolvable problem, but as an elegant general approach to solving a broad class of important problems.

The basic idea is to use the circle as an ordering basis to split the robots into nearly equal groups and/or identify robots that define boundaries between these groups. For the cases presented here, the robots are split into four groups by dividing the circle into four quadrants. The fact that the robots follow a global coordinate system is used for this. The robots that have the highest y coordinate, lowest y coordinate, highest x coordinate and lowest x coordinate are termed as the  $R^{north}$ ,  $R^{south}$ ,  $R^{east}$  and  $R^{west}$  robots, respectively (Refer to Figure 1). This naming, however, is only for explanation purposes. Within the algorithms, the robots are still anonymous and autonomous until the algorithm below labels them. The identification of these four robots is common for all algorithms, and is done as follows. Every robot  $R_i$  executes the following steps:

- Step 1: Broadcast own coordinates  $(x_i, y_i)$ .
- Step 2: Receive coordinates from other robots. Store the first two distinct coordinates received as  $(x_i^{n1}, y_i^{n1})$  and  $(x_i^{n2}, y_i^{n2})$ .
- Step 3: Calculate distances from  $(x_i, y_i)$  to  $(x_i^{n1}, y_i^{n1})$  and  $(x_i^{n2}, y_i^{n2})$  as  $D_i^{n1}$  and  $D_i^{n2}$ , respectively.



**Fig. 1.** Diagram depicting the North, South, East and West robots in a circle formation of robots

- Step 4: If  $(D_i^{n2} > D_i^{n1})$ , swap  $(x_i^{n1}, y_i^{n1})$  and  $(x_i^{n2}, y_i^{n2})$ .
- Step 5: Broadcast own coordinates  $(x_i, y_i)$ .
- Step 6: Receive coordinates  $(x_j, y_j)$  from another robot. If these coordinates are the same as  $(x_i^{n1}, y_i^{n1})$  OR  $(x_i^{n2}, y_i^{n2})$ , go to Step 11.
- Step 7: Calculate distances from  $(x_i, y_i)$  to  $(x_i^{n1}, y_i^{n1})$  and  $(x_i^{n2}, y_i^{n2})$  as  $D_i^{n1}$  and  $D_i^{n2}$ , respectively.
- Step 8: Calculate distance between  $(x_i, y_i)$  and  $(x_j, y_j)$  as  $D_i^{temp}$ .
- Step 9: If  $(D_i^{temp} < D_i^{n1})$ , then set  $(x_i^{n2}, y_i^{n2})$  as  $(x_i^{n1}, y_i^{n1})$ , and then set  $(x_i^{n1}, y_i^{n1})$  as  $(x_j, y_j)$ . Go to Step 11.
- Step 10: If  $(D_i^{temp} < D_i^{n2})$ , then set  $(x_i^{n2}, y_i^{n2})$  as  $(x_j, y_j)$ .
- Step 11: Repeat Steps 5 to 10 for  $N^{iter1}$  iterations (chosen empirically).

- Step 12: If  $(y_i > y_i^{n1})$  AND  $(y_i > y_i^{n2})$ , then identify oneself as  $R^{north}$ .  
STOP.
- Step 13: If  $(y_i < y_i^{n1})$  AND  $(y_i < y_i^{n2})$ , then identify oneself as  $R^{south}$ .  
STOP.
- Step 14: If  $(x_i > x_i^{n1})$  AND  $(x_i > x_i^{n2})$ , then identify oneself as  $R^{east}$ .  
STOP.
- Step 15: If  $(x_i < x_i^{n1})$  AND  $(x_i < x_i^{n2})$ , then identify oneself as  $R^{west}$ .

Each robot executes the above steps to determine if it is one of the four robots. Following this, the North, South, East and West robots alone broadcast their coordinates for a few iterations. These messages are received by the other robots and stored. The other robots also rebroadcast these messages. It should be noted that the robots are in a circle, so these messages literally travel around the circle. Also, the messages broadcast by the North, South, East and West robots need not be tagged. Once the other robots have four distinct coordinates stored, they can easily determine which corresponds to North, South, East and West (the North robot will have the highest y coordinate among the four and so on). The above algorithm is termed here as the *NEWS algorithm*. At the end of the NEWS algorithm the north, south, east and west robots are first correctly identified and labelled. Then their coordinates are made available to all other robots. Each robot  $R_i$  stores these coordinates as  $(x_i^{north}, y_i^{north})$ ,  $(x_i^{south}, y_i^{south})$ ,  $(x_i^{east}, y_i^{east})$  and  $(x_i^{west}, y_i^{west})$ . Then the group can execute one of the following algorithms to form other shapes or dividing into groups, as described next.

### 6.1 Formation of a Vertical Line

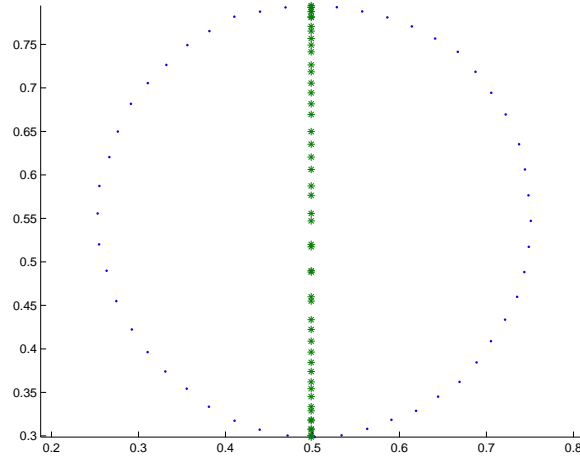
This algorithm is for a vertical line but could be used easily for a horizontal line as well. The idea behind line formation is that the North robot remains fixed, while all the other robots move along the direction of the x-axis to align themselves below the North robot. Every robot  $R_i$  executes the following steps:

- Step 1: Execute the NEWS algorithm.  
 Step 2: If  $(R^{north})$ , then do not move. STOP.  
 Step 3: Set the target coordinates  $(x_i^{tar}, y_i^{tar})$  to be reached as  $(x_i^{north}, y_i)$ .  
 Step 4: Take a step towards  $(x_i^{tar}, y_i^{tar})$ .  
 Step 5: Repeat Step 4 for  $N^{iter2}$  iterations (chosen empirically).

By replacing  $R^{north}$  (or  $R^{south}$ ) by  $R^{east}$  (or  $R^{west}$ ), one can form a horizontal line instead of a vertical line. The algorithm was tested using simulations and results are depicted in Figure 2, where the dots (.) represent initial positions around a circle and the asterisks (\*) the final positions along a line.

### 6.2 Formation of a Triangle

The idea behind this algorithm is to form a triangle with the North, East and West robots as the vertices. The robots in the bottom two quadrants of the



**Fig. 2.** Result from an actual simulation of robots having executed the line formation algorithm

circle form the base, while the robots in each quadrant on top form one side each. This results in an isosceles triangle. Basically, each robot moves along the direction of the y-axis to reach a point on the line segment of its interest. The orientation of the triangle can be changed easily. Every robot  $R_i$  executes the following steps:

- Step 1: Execute the NEWS algorithm.
- Step 2: If  $(R^{north})$  OR  $(R^{east})$  OR  $(R^{west})$ , then do not move. STOP.
- Step 3: If  $(x_i > x_i^{north})$  AND  $(y_i > y_i^{east})$ , then set the target coordinates  $(x_i^{tar}, y_i^{tar})$  as the point on the line connecting  $(x_i^{north}, y_i^{north})$  and  $(x_i^{east}, y_i^{east})$  with x-coordinate  $x_i$ . Go to Step 6.
- Step 4: If  $(x_i < x_i^{north})$  AND  $(y_i > y_i^{west})$ , then set the target coordinates  $(x_i^{tar}, y_i^{tar})$  as the point on the line connecting  $(x_i^{north}, y_i^{north})$  and  $(x_i^{west}, y_i^{west})$  with x-coordinate  $x_i$ . Go to Step 6.
- Step 5: Set the target coordinates  $(x_i^{tar}, y_i^{tar})$  as the point on the line connecting  $(x_i^{west}, y_i^{west})$  and  $(x_i^{east}, y_i^{east})$  with x-coordinate  $x_i$ .
- Step 6: Take a step towards  $(x_i^{tar}, y_i^{tar})$ .
- Step 7: Repeat Step 6 for  $N^{iter3}$  iterations (chosen empirically).

By changing the choice of stationary robots (between  $R^{north}$ ,  $R^{south}$ ,  $R^{east}$  and  $R^{west}$ ) the orientation of the triangle can be changed.

### 6.3 Formation of a Square

The algorithm for a square is nearly the same as the triangle algorithm described earlier. The only difference is that, once a robot determines that it is one of the cardinal (North, South, East or West) robots, it does not move. All the other robots try to form lines with these robots as end points forming a quadrilateral that is approximately a square. Every robot  $R_i$  executes the following steps:

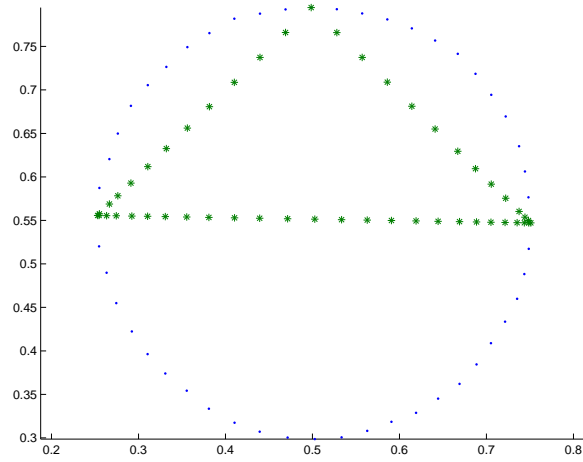
- Step 1: Execute the NEWS algorithm.
- Step 2: If  $(R^{north})$  OR  $(R^{south})$  OR  $(R^{east})$  OR  $(R^{west})$ , then do not move. STOP.
- Step 3: If  $(x_i > x_i^{north})$  AND  $(y_i > y_i^{east})$ , then set the target coordinates  $(x_i^{tar}, y_i^{tar})$  as the point on the line connecting  $(x_i^{north}, y_i^{north})$  and  $(x_i^{east}, y_i^{east})$  with x-coordinate  $x_i$ . Go to Step 7.
- Step 4: If  $(x_i < x_i^{north})$  AND  $(y_i > y_i^{west})$ , then set the target coordinates  $(x_i^{tar}, y_i^{tar})$  as the point on the line connecting  $(x_i^{north}, y_i^{north})$  and  $(x_i^{west}, y_i^{west})$  with x-coordinate  $x_i$ . Go to Step 7.
- Step 5: If  $(x_i < x_i^{south})$  AND  $(y_i < y_i^{west})$ , then set the target coordinates  $(x_i^{tar}, y_i^{tar})$  as the point on the line connecting  $(x_i^{south}, y_i^{south})$  and  $(x_i^{west}, y_i^{west})$  with x-coordinate  $x_i$ . Go to Step 7.
- Step 6: Set the target coordinates  $(x_i^{tar}, y_i^{tar})$  as the point on the line connecting  $(x_i^{south}, y_i^{south})$  and  $(x_i^{east}, y_i^{east})$  with x-coordinate  $x_i$ .
- Step 7: Take a step towards  $(x_i^{tar}, y_i^{tar})$ .
- Step 8: Repeat Step 7 for  $N^{iter4}$  iterations (chosen empirically).

### 6.4 Formation of a Semi-circle

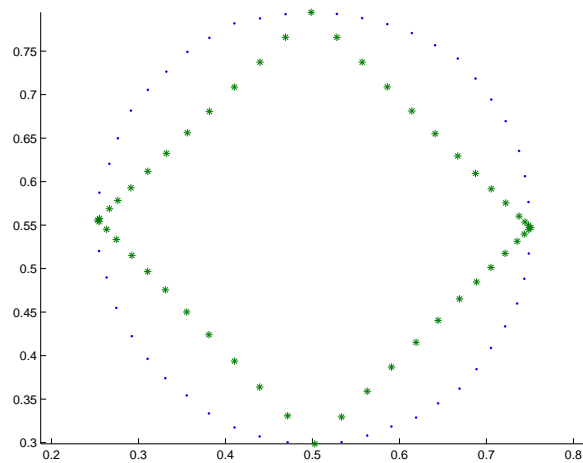
For a semicircle, the robots in the upper two quadrants remain in place, while the robots in the bottom half form a line. Every robot  $R_i$  executes the following steps:

- Step 1: Execute the NEWS algorithm.
- Step 2: If  $(R^{east})$  OR  $(R^{west})$ , do not move. STOP.
- Step 3: If  $(y_i > y_i^{west})$  OR  $(y_i > y_i^{east})$ , do not move. STOP.
- Step 4: Set the target coordinates  $(x_i^{tar}, y_i^{tar})$  as the point on the line connecting  $(x_i^{west}, y_i^{west})$  and  $(x_i^{east}, y_i^{east})$  with x-coordinate  $x_i$ .
- Step 5: Take a step towards  $(x_i^{tar}, y_i^{tar})$ .
- Step 6: Repeat Step 5 for  $N^{iter5}$  iterations (chosen empirically).

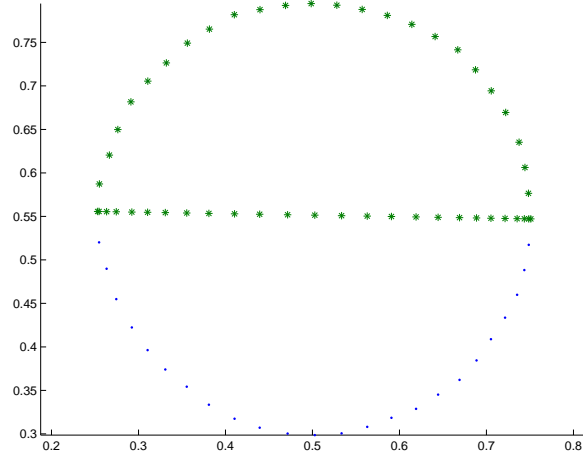
The results for formation of triangle, square and semi-circle are presented in Figure 3, Figure 4 and Figure 5, respectively. It can be observed that the distribution of the robots along the edges of these patterns is not uniform. This was expected, since distribution was not addressed in the algorithms. The focus was only on achieving different patterns.



**Fig. 3.** Result from an actual simulation of robots having executed the triangle formation algorithm



**Fig. 4.** Result from an actual simulation of robots having executed the square formation algorithm



**Fig. 5.** Result from an actual simulation of robots having executed the semicircle formation algorithm

### 6.5 Splitting into Four Groups

This method is similar to the algorithms discussed earlier. The robots first determine the quadrant, and thus the group in which they belong. Following this, they make sure they coordinate only with the robots in their group for future tasks. To achieve this, they tag messages they broadcast with a group identification based on the quadrant in which they fall. Using this group identification, they make use of messages from their group and ignore messages from members of other groups. Each group can form any pattern based on all the algorithms explained in the earlier sections. For the group determination algorithm, every robot  $R_i$  executes the following steps (this is similar to the steps used in the square formation algorithm, but is presented here for sake of completeness):

- Step 1: Execute the NEWS algorithm.
- Step 2: If  $(R^{north})$  OR  $(R^{west})$  OR  $(R^{south})$  OR  $(R^{east})$ , assign oneself a value of 1, 2, 3 and 4 as the group identification, respectively. STOP.
- Step 3: If  $(x_i > x_i^{north})$  AND  $(y_i > y_i^{east})$ , then assign oneself a group identification of 1. STOP.
- Step 4: If  $(x_i < x_i^{north})$  AND  $(y_i > y_i^{west})$ , then assign oneself a group identification of 2. STOP.
- Step 5: If  $(x_i < x_i^{south})$  AND  $(y_i < y_i^{west})$ , then assign oneself a group identification of 3. STOP.

Step 6: If  $(x_i > x_i^{south})$  AND  $(y_i < y_i^{east})$ , then assign oneself a group identification of 4.

Now every robot has a group identification and hence can execute separate behaviors by coordinating only with robots which have the same group identification. It should be noted that execution of the *same* algorithm by each robot results in it assigning itself a group identification. This does not give an identity to the individual robots, which still remain anonymous. It only serves the purpose of helping the robots distinguish the messages coming from members in their group from those coming from other groups. The simulation results are presented in Figure 6 where the robots, after determining their groups, interact with only their group members and execute the circle formation algorithm (diameter of the smaller circles can be chosen to be a fraction of the larger circle). Circle formation by groups is just an example and they can perform other tasks as well.

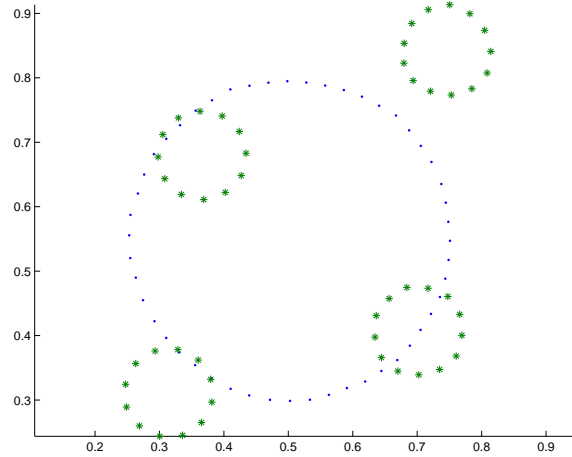


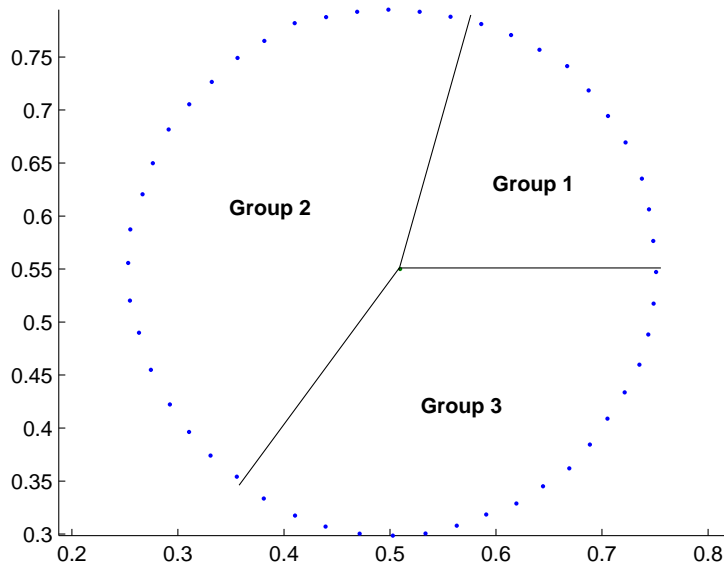
Fig. 6. Result from an actual simulation of robots having executed the splitting into groups and performing group tasks algorithm

### 6.6 Splitting into Groups — Generalized Case

While splitting robots into four equal groups is interesting, many applications require a different number of possibly unequal groups. In this section, we discuss how circle formation can be exploited to achieve this.

In a circle, the robots are distributed uniformly, so one can employ a splitting technique based on the angular position of the robots with respect

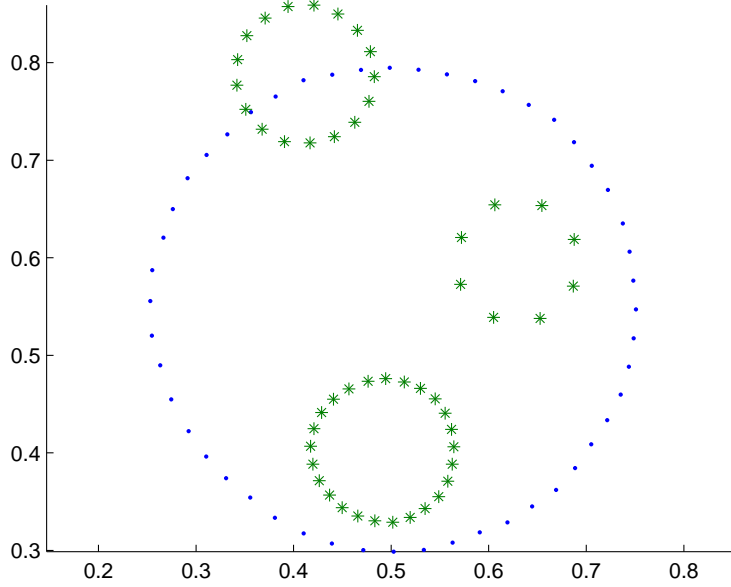
to the center of the circle. The robots are aware of the positions of the cardinal (North, South, East and West) robots and using their position can determine their angle with respect to the line segment between the center of the circle and the East robot. (The center of the circle can be taken as the geometric center of the cardinal robot positions). Depending on the application, the robots can be pre-programmed with certain angles so as to divide the circle into sectors, and hence groups. Based on the sector in which a robot lies, it can assign itself a pre-specified group identification. Then as shown in the earlier subsections, the robots can coordinate only with its group members and perform tasks assigned to the group. Note again that, while robots do determine special labels in the field and act accordingly, no *specific* robot is *pre-tasked* with a particular behavior. All behavioral differentiation emerges through self-organization.



**Fig. 7.** Diagram depicting splitting of robots in a circular formation based on angular constraints

Figure 7, depicts this technique. In this case, the robots are split into three groups. The robots need to given two angles for splitting into three groups. In the special case in which the angles of the sectors are specified such that they are equal, the resulting groups are also equal.

An example of such a splitting technique is presented in Figure 8. In this example, the angles specified were 60 and 120 degrees and hence resulted in



**Fig. 8.** Result from an actual simulation of robots having executed the splitting into 3 unequal groups algorithm

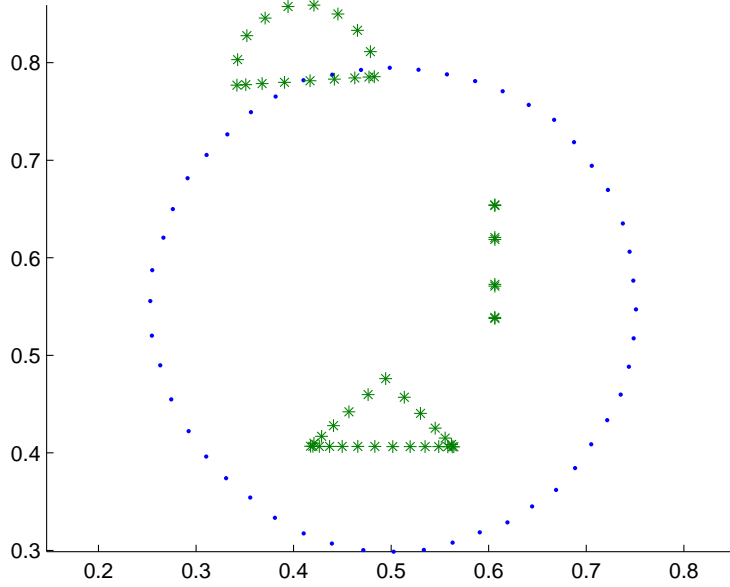
sectors with angles 60, 120 and 180 degrees. The groups then performed circle formation algorithms (with one-fourth the original diameter). The distribution of the three circles can be seen to vary depending on the number of robots that ended up in each group, which is directly proportional to the magnitude of the angles. Since circle-formation is the basis of forming other shapes, robots in each small circle can then form different shapes, This is shown in Figure 9.

## 7 Programming the Robots

The real power of multi-robot systems lies in distributed control and it is essential that each robot be made entirely autonomous with no need for manual intervention. Thus, if the robots are to perform complex tasks comprising several subtasks such as circle formation, group formation, etc., there must be a *canonical procedure program* that can be pre-loaded into each robot such that, as each robot executes its program, the result is the performance of the complex task. One possible approach to this is discussed here.

The general program followed by each robot can be described as follows:

- Perform a circle formation algorithm for a given number of iterations.



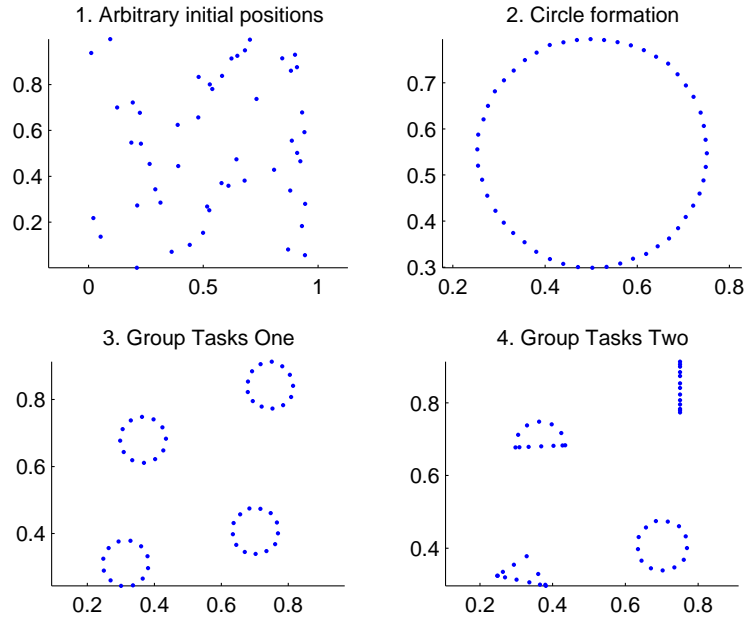
**Fig. 9.** Result from an actual simulation for splitting into 3 unequal groups and performing different group tasks

- Perform the NEWS algorithm for a given number of iterations.
- Determine the group identity.
- Perform the task assigned to the group.

Once, the parameters like, number of iterations and the task for each group are given prior to deploying the robots, there is no need for any manual intervention. Also, it is important to note that all robots are given *identical* programs. Differences in behavior arise only as a result of labelling generated during *execution*, and are *not* specifically assigned to individual robots a priori.

This idea is illustrated in Figure 10. Initially, the robots are in arbitrary positions. They then execute the circle formation algorithm, followed by the self-determination of groups. Then the robots execute (say) the circle formation algorithm as a group task and from there, they can form different shapes, which they know beforehand.

Similar programs can be written for other complex tasks. When a group of robots is to perform that task, the appropriate program can be loaded into all the robots or, more likely, triggered by circumstances. In the latter scenario, a large number of programs would be stored in each robot — the same ones for each — and would be triggered through an external instruction or observations made by the robots themselves. This will raise issues of consistency, i.e., how



**Fig. 10.** Combination figure showing results from simulations performed successively. 1-Robots are randomly distributed, 2-Robots execute circle formation algorithm, 3-Robots split into four groups and each forms a circle, 4-Each group then forms other patterns

to ensure that all robots switch to the same program, but this is beyond the scope of the present work. Similar issues exist even with groups of human agents, which must be told to switch tasks. The focus in this work is on developing algorithms where these “external imperatives” can be made as independent of the detailed state of the robot group as possible (e.g., not requiring information on identities or positions of specific robots).

## 8 Discussion and Conclusion

The work described in this chapter has built upon circle-formation algorithms developed by other researchers to explore whether this can be used as a generic coordination mechanism for solving other, more complex organization problems. We have described methods to make robots form different shapes like lines, semicircles, triangles and squares, and to split into groups of specified size to perform specialized group tasks. A key point is that, using this approach, each robot for a specific organization task can be *pre-loaded* with

a generic program whose execution by all individual robots will lead to the desired organization *without need for manual intervention*.

Most problems in formation or organization are solved by the assignment of *roles* or *tasks* to specific individuals. For example, a square can be formed if four robots are assigned to be the corners and the rest then use them to line up along the edges. However, in the absence of a central controller and with anonymous robots, there is no objective basis on which to make these assignments; all agents execute the same algorithm, and none is inherently disposed to behave in a special way. One possible way to get around this is to adapt *leader-election algorithms* [46, 47, 48, 49, 50, 25, 26] developed in the distributed systems literature for various purposes. These algorithms distinguish certain agents/robots for specific tasks (e.g., cluster-heads or gateways in ad-hoc networks) through a distributed process. Once these leaders are elected, they can perform functions such as indicating vertices or serving as beacons, and the rest can execute appropriate algorithms to fall in place using these indicators. However, different groups of agents must behave differently relative to the leaders/beacons to achieve the formation, e.g., four different groups of robots must choose a different edge to line up along in order to form a square. Thus, the decentralized organization process involves three steps: 1) Selection of specific leader/beacon agents; 2) Assignment of different behaviors to appropriate subgroups of agents; and 3) Execution of the designated behaviors by all agents. The assignments made in Steps 1 and 2 must satisfy certain criteria, e.g., number of beacons and relatively equal sizes of groups. Our work explores the use of circle formation as a generic mechanism integrating both leader-election and subsequent group assignment in a reliable and efficient way. The circle is especially appropriate for this function because of its uniquely homogeneous, arbitrarily symmetric shape. It provides a natural manifold for defining a problem-specific coordinate system for a wide class of problems. In work not presented here we have also studied the robustness of our algorithms and found them to be quite robust to many, though not all, types of errors [42] The results presented in this chapter demonstrate clearly that our approach is both powerful and generic. However, several issues, such as the positioning of groups, remain open and will be addressed in future work.

## References

1. BROOKS R (1991). *Science*, 253(5025):1227–1232.
2. Brooks R, Connell J (1987) SPIE Conference on Mobile Robots, Cambridge, MA (USA), pp. 77–84.
3. Cao Y, Fukunaga A, Kahng A (1997) *Autonomous Robots*. 4(1):1–23.
4. Parke LE (2000) Proceedings of the 5th International Symposium on Distributed Autonomous Robotic Systems, DARS, Springer Verlag, 4:3–12.
5. Yamaguchi H (1997) IEEE International Conference on Robotics and Automation, 3:2300–2305.

6. Mataric MJ (1995) *Robotics and Autonomous Systems*, 16:321–331.
7. Mataric MJ (1991) A Distributed Model for Mobile Robot Environment-Learning and Navigation. Technical Report TR-1228, Artificial Intelligence Laboratory, Massachusetts Institute of Technology.
8. Mataric MJ (1998) *Journal of Experimental and Theoretical Artificial Intelligence*, 10(3):357–369.
9. Mataric MJ (1998) In *IEEE Intelligent Systems*, 6–9.
10. Goldberg D, Mataric MJ (2000) Technical Report IRIS-00-387, USC Institute for Robotics and Intelligent Systems.
11. Ostergaard E, Mataric MJ, Sukhatme GS (2001) In *IEEE/RSJ International Conference on Intelligent Robots and Systems(IROS)*, 2:821–826.
12. Mataric MJ, Fredslund J (2002) *IEEE Transactions on Robotics and Automation*, 18(5):837–846.
13. Mataric MJ, Jones C (2004) In *Proceedings of the Hawaii International Conference on Computer Sciences, Waikiki, Hawaii*, 27–32.
14. Nolfi S(1998) *Connection Science*, 10(3-4):167–184.
15. Unsal C, Bay JS(1994) *IEEE International Symposium on Intelligent Control*, Columbus, Ohio, 249–254.
16. Shen W, Chuong C, Will P (2002) *IEEE/RSJ International Conference on Intelligent Robots and System*, 3:2776–2781.
17. Baldassarre G, Nolfi S, Parisi D (2003) *Artificial Life*, 9(3):255-267.
18. Bonabeau E, Dorigo M, Theraulaz G (1999) *Swarm Intelligence: From Natural to Artificial Systems*, NY: Oxford University Press Inc.
19. Beni G Wang J (1989) *Swarm intelligence in cellular robotics systems*. Proceeding of NATO Advanced Workshop on Robots and Biological System, Il Ciocco, Tuscany, Italy.
20. Arkin RC (1998) *Behavior-Based Robotics*, Cambridge, MA: MIT Press.
21. Millonas M (1994) *Swarms, phase transitions, and collective intelligence*. In: Palaniswami M, Attikiouzel Y, Marks R, Fogel D, Fukuda T (eds) *Computational Intelligence: A Dynamic System Perspective*, pp.137–151. IEEE Press, Piscataway, NJ.
22. Dudek G, Jenkin M, Milios E, Wilkes D (1993) A taxonomy for swarm robots, *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Japan, 1:441–447.
23. Sugihara K, Suzuki I (1990) *Distributed Motion Coordination of Multiple Mobile Robots*, In *IEEE International Symposium on Intelligent Control*, pp. 138–143.
24. Dudenhoeffer DD, Jones MP (2000) A Formation Behavior for Large-Scale Micro-Robot Force Deployment, In *Proceedings of the 2000 Winter Simulation Conference*, pp. 972–982.
25. Flocchini P, Prencipe G, Santoro N, Widmayer P (2001) *Gathering of Asynchronous Mobile Robots with Limited Visibility*. In *18th International Symposium on Theoretical Aspects of Computer Science (STACS)*, 331(1-3):147–168.
26. Desai J, Kumar V, Ostrowski J (1999) *Control of changes in formation for a team of mobile robots*, In *Proceedings of 1999 International Conference on Robotics and Automation*, pp.1556–1561.
27. Flocchini P, Prencipe G, Santoro N, Widmayer P. *Pattern Formation by Autonomous Mobile Robots*. *Interjournal of Complex Systems*, Article 395, (on line publication <http://www.interjournal.org>).
28. Sugihara K, Suzuki I (1996) *Journal of Robotic Systems*, 13:127–139.

29. Wang PKC (1989) Navigation Strategies For Multiple Autonomous Mobile Robots Moving In Formation, IEEE/RSJ International Workshop on Intelligent Robots and Systems, pp. 486–493.
30. Gordon FD, Spears MW (1999) Using Artificial Physics to Control Agents. In Proceedings of IEEE International Conference on Information, Intelligence and Systems.
31. Gordon FD, Spears MW, Sokolsky W, Lee I (1999) Distributed Spatial Control, Global Monitoring and Steering of Mobile Agents. In Proceedings of IEEE International Conference on Information, Intelligence and Systems (ICIIS).
32. Molnár P, Starke J (2001) Control of distributed autonomous robotic systems using principles of pattern formation in nature and pedestrian behavior. IEEE Transactions on Systems, Man, and Cybernetics, Part B, 31(3): 433–435.
33. Chen Q, Luh J (1994) Coordination and control of a group of small mobile robots. In Proceedings of the IEEE International Conference on Robotics and Automation, pp.2315–2320.
34. Trivedi M, Hall B, Kogut G, Roche S (2000) Web-based teleautonomy and telepresence, In 45th SPIE Optical Science and Technology Conference, Applications and Science of neural networks, fuzzy systems and evolutionary computation III, San Diego, v.4120.
35. Suzuki I, Yamashita M (1999) Distributed Anonymous Mobile Robots: Formation of Geometric Patterns, Siam J. Comput., 28(4):1347–1363.
36. Suzuki I, Yamashita M (1996) Distributed Anonymous Mobile Robots- Formation and Agreement Problems, In Proc. Third Colloq. On Struc. Information and Communication Complexity (SIROCCO), pp 313–330.
37. Tanaka O (1992) Forming a Circle by Distributed Anonymous Mobile Robots. Bachelor thesis, Department of Electrical Engineering, Hiroshima University, Hiroshima, Japan.
38. Flocchini P, Prencipe G, Santoro N, Widmayer P (1999) Hard Tasks for Weak Robots: The Role of Common Knowledge In Pattern Formation by Autonomous Mobile Robots. In 10th International Symposium on Algorithm and Computation (ISAAC), pp.93–102.
39. Defago X, Konagaya A (2002) Circle Formation for Oblivious Anonymous Mobile Robots with No Common Sense of Orientation. In Proceedings of the second ACM international workshop on Principles of mobile computing, pp.97–104.
40. Chatzigiannakis I, Markou M, Nikolettseas S (2004) Distributed Circle Formation for Anonymous Oblivious Robots. In 3rd Workshop on Efficient and Experimental Algorithms, Lecture Notes in Computer Science, 3059:159–174.
41. Yun X, Alptekin G, Albayrak O (1997) Line and circle formation of distributed physical mobile robots. Journal of Robotic Systems, 14(2):63–76.
42. Swaminathan K (2005) Self-organized formation of geometric patterns in multi-robot swarms using wireless communication. MS Thesis, University of Cincinnati, Cincinnati.
43. Ando H, Suzuki I, Yamashita M (1995) Formation and agreement problems for synchronous mobile robots with limited visibility. In Proceedings of the 1995 IEEE International Symposium on Intelligent Control, pp.453–460.
44. Gordon N, Wagner IA, Brucks AM (2003) Discrete Bee Dance Algorithms for Pattern Formation on a Grid. In IEEE/WIC International Conference on Intelligent Agent Technology, pp.545–549.

45. Dudek G, Jenkin M, Milios E, Wilkes D (1993) Robust Positioning with a Multi-Agent Robotic System. In Proceedings of the International Joint Conference of Artificial Intelligence (IJCAI) on Dynamically Interacting Robots, Chambéry, France, pp.118-123.
46. Arora A, Kulkarni S (1995) Designing masking fault-tolerance via nonmasking fault-tolerance, 14TH Symposium on Reliable Distributed Systems, pp.174.
47. Arora A, Gouda M. Distributed Reset. IEEE Transactions on Computers 43(9):1026-1038.
48. Nagpal R. Programmable Self-Assembly Using Biologically -Inspired Multi-robot Control (2002). ACM Joint Conference on Autonomous Agents and Multi-agent Systems, Bologna.
49. Coore D, Nagpal R, Weiss R (1997) Paradigms for Structure in an Amorphous Computer. MIT Artificial Intelligence Laboratory Memo 1614.
50. Nagpal R, Coore D (1998) An algorithm for group formation in an amorphous computer. Proceedings of the Tenth International Conference on Parallel and Distributed Systems (PDCS).



---

## Index

collective robotics, 1, 2

communication based multi-robot  
system, 5

pattern formation, 2, 3, 5

self-organization, 2, 14  
swarms, 2

