Stabilizing Flocking Via Leader Election in Robot Networks

Davide Canepa and Maria Gradinariu Potop-Butucaru

Université Pierre et Marie Curie (Paris 6), LIP6, CNRS, INRIA, France
canepa.davide@tiscali.it, maria.gradinariu@lip6.fr

Abstract. Flocking is the ability of a group of robots to follow a leader or head whenever it moves in a plane (two dimensional Cartesian space). In this paper we propose and prove correct an architecture for a self-organizing and stabilizing flocking system. Contrary to the existing work on this topic our flocking architecture does not rely on the existence of a specific leader *a priori* known to every robot in the network. In our approach robots are uniform, start in an arbitrary configuration and the head of the group is elected via algorithmic tools.

Our contribution is threefold. First, we propose novel probabilistic solutions for leader election in asynchronous settings under bounded schedulers. Additionally, we prove the impossibility of deterministic leader election when robots have no common coordinates and start in an arbitrary configuration. Secondly, we propose a collision free deterministic algorithm for circle formation designed for asynchronous networks. Thirdly, we propose a deterministic flocking algorithm totally independent of the existence of an *a priori* known leader. The proposed algorithm also works in asynchronous networks.

1 Introduction

Several applications like large-scale constructions, hazardous waste cleanup, space missions or exploration of dangerous or contaminated area motivate the research related to self-organized robot networks (multi-robot systems). The literature proposed so far a significant amount of research towards the operation of a single remote robot, however more work is required towards the operation of networks of autonomous robots. These systems provide interesting solutions to many real problems: manipulation of large objects, system redundancy, reducing time complexity for the targeted tasks, however they bring in discussion some specific difficulties. In particular, these robots should achieve their tasks without human intervention based only on the information provided by the robots in the same group. Moreover, they have to explore unknown or quasi unknown environments while avoiding collisions among themselves. Additionally, they have to be able to reorganize whenever one or more robots in the group stop to behave correctly.

In this paper we propose a self-organized and stabilizing flocking architecture. Flocking is the ability of a group of robots to follow a leader or a flock head whenever this one changes its position in plane. Our work is developed in

Corda model [1,2] one of the two theoretical models proposed so far for robot networks. The first model proposed in the literature was introduced by Suzuky and Yamashita [3,4,5]. In this model robots are oblivious and perform a cycles of elementary actions as follows : observation (the robot observes the environment), computation (the robot computes its next position based on the information collected in the observation phase) and motion (the robot changes its position by moving to the coordinates returned by the computation phase). In this model robots cannot be interrupted during the execution of a cycle. The Corda model breaks the execution cycle in elementary actions. That is, a robot can be activated/turned off while executing a cycle. Hence, robots are not anymore synchronized.

In both Corda and Suzuki-Yamashita model several problems have been studied under different assumptions on the environment (e.g. schedulers, fault-tolerance), robots visibility, accuracy of compasses: circle formation, pattern formation, gathering [6,7,8,9,10,11,12]. The *flocking problem* although largely discussed for real robots ([13,14] and [15]) was studied from theoretical point of view principally by Prencipe [16,17]. The authors propose non-uniform algorithms where robots play two roles: leader or follower. The leader is unique and all the followers know it. Obviously, when the leader crashes, disappears or duplicates the flock cannot finish its task. Our approach is different, the leader is not known *a priory* but it is elected via algorithmic tools. When the current leader disappears from the system another leader is elected and the network can finish its task. In order to be sound our flocking architecture includes as basic building block a leader election module.

The leader election problem has been studied under a broad class of models. Recent works propose solutions in the population protocol model, [18,19]. The same problem has also been studied in the mobile agents model [20]. These models may seem similar to the robots model however, in these models agents either have a point to point interaction with simultaneous change of their respective state or assume a specific topology of the network guesting the agents (e.g. rings) or make additional assumptions like the existence of whiteboards on the nodes visited by agents. In the robot networks there is no such assumptions since robots move in a Cartesian two dimensional space helped only by the information they can collect at each activation.

In robot networks leader election have been mainly studied in [5]. The authors propose a solution where robots share the same coordinate system. Further in [21] is proposed an algorithm for leader election based on Lyndon words which works if the number of robots is prime and robots are not disposed in a regular n-gon. The previously cited works focus the Suzuki-Yamashita model. In [22] the author prove the leader election impossibility in Corda model when the number of robots is even.

Our contribution. In this paper we propose and prove correct an architecture for a self-organizing and stabilizing flocking system. Contrary to existing work on this topic our flocking architecture does not rely on the existence of a specific leader *a priory* known to every robot in the network. In our approach robots are uniform, start in an arbitrary configuration and the head of the group is elected via algorithmic tools. Our architecture includes three modules: a leader election module, a preprocessing module and a motion module. The leader election module returns to each robot its status : leader or follower. The preprocessing module outputs a moving formation. The motion module provides the rules that will make the robots in the moving formation to change their positions whenever the leader moves. Every modification of robots position preserves the moving formation. For each of these modules we propose deterministic or probabilistic algorithms (in the case when a deterministic solution is impossible). Moreover, we prove their correctness in Corda model. The correctness[1] of the probabilistic algorithms considered in this paper assumes bounded schedulers.

2 Model

The notions and the model description presented in this section are borrowed from [1,11,16]. We consider a system of autonomous mobile robots that work in the Corda model [1]. Each robot is capable of observing its surrounding, computing a destination based on what it observed, and moving towards the computed destination: hence it performs an (endless) cycle of observing, computing, and moving. Each robot has its own local view of the world. This view includes a local Cartesian coordinate system having an origin, a unit of length, and the directions of two coordinate axes (which we will refer to as the x and y axes), together with their orientations, identified as the positive and negative sides of the axes.

The robots are modeled as processes with computational capabilities, which are able to freely move in the plane. They are equipped with sensors that let each robot observe the positions of the others with respect to their local coordinate system. Each robot is viewed as a point, and can see all the other robots in the system.

The robots act totally independently and asynchronously from each other, and do not rely on any centralized directives, nor on any common notion of time. Furthermore, they are oblivious, meaning that they do not remember any previous observation nor computations performed in the previous steps. Note that this feature gives to the algorithms designed in this model the nice property of self-stabilization [24]: in fact, every decision taken by a robot cannot depend on what happened in the system previously, and hence cannot be based on corrupted data stored in its local memory. The robots are anonymous, meaning that they are a priory indistinguishable by their appearances, and they do not have any kind of identifiers that can be used during the computation. Moreover, there are no explicit direct means of communication; hence the only way they

[1] Due to space restrictions, most of the proofs are proposed in the extended version of this work [23].

have to acquire information from their fellows is by observing their positions. The robots are uniform, meaning that they execute the same algorithm, which takes as input the observed positions of the robots, and returns a destination point towards which the executing robot moves.

Schedulers. A scheduler decides at each configuration the set of robots allowed to perform their actions. A scheduler is fair if, in an infinite execution, a robot is activated infinitely often. In this paper we consider the fair version of the following schedulers:

- *k-bounded*: between two consecutive activations of a robot, another robot can be activated at most k times;
- *arbitrary*: at each configuration an arbitrary subset of robots is activated.

In short, robots move asynchronously, are oblivious, anonymous and uniform. Additionally, their activation is managed by a scheduler who decides in each configuration the set of active robots. That is, in this paper we consider the Corda model refined with the above mentioned fair scheduling strategies (i.e. k-bounded and arbitrary).

3 Leader Election and Flocking Problems

Leader election creates an asymmetry whatever the initial configuration. Robots may be in one of the following states: leader or follower and the leader should be unique in the system.

Definition 1 (Leader Election). *A system of robots verifies the leader election specification iff the following two properties hold:*

- *Safety: The system is in a legal configuration where there is an unique robot in the state leader and all the other robots are in the state follower.*
- *Liveness: The legal configuration is reached in a finite number of steps.*

Leader election is the building block for a large class of problems. In this paper we focus on the flocking problem. Intuitively, a flock is a group of robots that moves in the plane in order to execute a task while maintaining a specific formation. The most current definition of the flocking implicitly assumes the existence of an unique leader of the group that will lead the group during the task execution. Robots have as input the same pattern representing the flock to be maintained which is described as a set of coordinates in the plane, relative to a point representing the leader.

Obviously, in order to achieve flocking robots need to re-organize their formation whenever the leader changes its position. Therefore the definition of flocking has to capture the mobility of the flock.

Formally, the flocking problem can be defined as follows:

Definition 2 (Flocking). *Let S be a system of robots and let $\mathcal{P}$ be the flocking pattern. S verifies the flocking specification iff the robots satisfy $\mathcal{P}$ infinitely often.*

4 Architecture of a Flocking System

In the following we define a possible architecture for a flocking system. The architecture is composed of three modules : the leader election module, the preprocessing module and the flocking module.

- The leader election module is the base of the architecture. This module accepts as input a set of robots arbitrarily distributed in the plane and elects a leader. Results related to the impossibility of leader election and detailed description of probabilistic solutions for leader election are proposed in Section 5.
- The preprocessing module prepares the group of robots for the moving formation. All robots but the leader are placed on the smallest enclosing circle. Then, all robots on the smallest enclosing circle form a circular moving formation using as reference point the leader computed by the leader election module. One robot in this set will further act as the head of the flock. The preprocessing module is propose in Section 6.
- The flocking module receives as input a moving formation which initially has a circular form defined by a reference robot and a head and provides the necessary rules to move this formation in the plane whenever the head changes its position. The objective of the flocking module is to ensure the formation moving while keeping its properties. The algorithms for moving the formation are proposed in Section 7.

5 Leader Election Module

In this section we prove the impossibility of deterministic leader election. Generally, the impossibility results can be circumvent by using randomization. In the following we show that probabilistic leader election is impossible for 2 robots systems. However, the probabilistic leader election is possible for systems of size greater than 3.

5.1 Impossibility Results for Leader Election

In this section we prove the deterministic leader election impossible in Suzuki-Yamashita and Corda models.[2]

Theorem 1. *Deterministic leader election is impossible.*

Proof (sketch). Lets consider n robots forming a regular n-gon with the local $x - y$ coordinates of each robot such that the y positive axis is directed towards the next robot in clockwise. Assume also the x positive axis is such that the n-gon has no value of x less than 0. Consider all robots have the same unit of length. Without restraining the generality we consider in the following an equilateral triangle. For a deeper comprehension, lets consider Figure 1.

[2] Note that in [22] is proved the impossibility of leader election for n even, while in [21] is shown that leader election can be deterministically solved for n prime and robots not disposed in a n-gon.

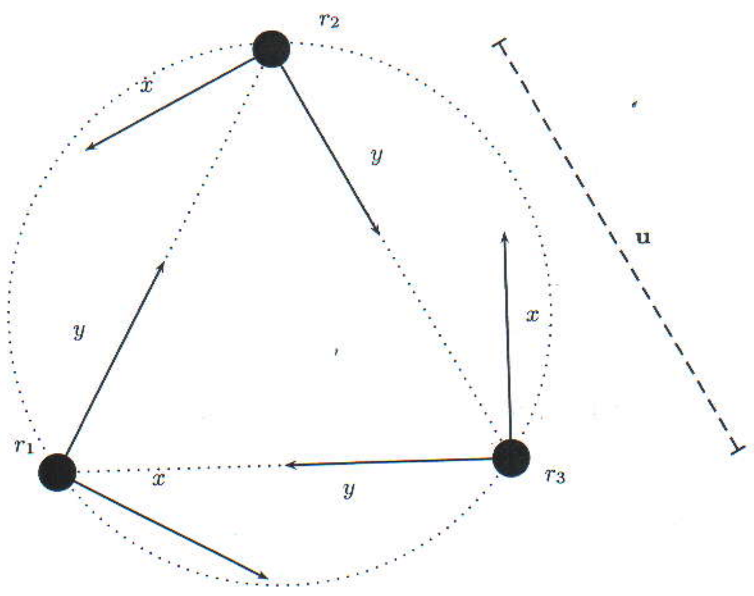

Fig. 1. Symmetric Configuration

Each robot can see a robot in (0,0) (itself) and other two robots in $(\frac{u}{2}, \frac{\sqrt{3}}{2}u)$ and in $(u, 0)$. Note that the three robots have the same view.

Assume a configuration such that the leader is the robot in $(u,1)$. In our example for r_1 the leader is r_2, for r_2 the leader is r_3 and for r_3 the leader is r_1. Each robot sees a different leader. Therefore, the safety property is violated.

Assume an initial configuration where there is no leader. In order to reach a legal leader election configuration robots should move. Assume the algorithm executed by each robot makes them move towards a point (x', y') of their system of coordinates and assume the scheduler chooses all robots to move concurrently. The system reaches a configuration where the n-gon structure is maintained. Moreover, in the new configuration robots have the same view. So, each deterministic movement from a symmetric configuration leads to a symmetric configuration. Hence, the system never converges to a legal configuration.

Lemma 1. *There is no probabilistic 2-robots leader election.*

5.2 Probabilistic Leader Election

In this section we propose probabilistic solutions for leader election for systems with three or more robots.

Probabilistic leader election with 3 robots. The algorithm idea is to exploit the asymmetry of a triangle. We choose as leader candidate the robot with the smallest angle or the robot different from the other two robots in the case of an isosceles triangle. The randomization is used only to break the symmetry of equilateral triangles. For this particular case we use randomization in order to create an asymmetric triangle on which we apply the method described above.

```
1) Compute the angle between every two robots.
2) if $my\_angle$ is the smallest then become $Leader$.
3) else if $my\_angle$ is not the smallest but the other two are identical
        then become $Leader$.
4) else if All the angles are identical
        then move perpendicular to segment linking the other
             two robots in opposite direction with probability $\frac{1}{3}$
```

Algorithm 5.1. Leader election algorithm

Lemma 2. *Algorithm 5.1 converges to the leader election specification in finite number of steps in expectation in the Corda model refined with a k-bounded scheduler.*

Probabilistic leader election with more than 3 robots. In the following we propose a leader election algorithm for systems with more than three robots. Intuitively, the leader robot will be the robot whose position is the closest to the center of the smallest enclosing circle (SEC). Additionally, we would like the leader to define a second reference together with the center of SEC. Therefore, the leader should not be placed on the center position. If a robot is initially positioned in the center of the smallest enclosing circle then a preprocessing phase is executed. The robot in the center moves to a free position chosen non-deterministically inside the SEC. The leader election algorithm idea is as follows. Robots randomly change their positions until only one of them is the closest to the SEC.

```
1) Compute the smallest enclosing circle $SEC$.
2) Compute the distance $d\_myself$ to the center of $SEC$.
3) if ($d\_myself < d_k\ \forall$ k$\neq$ $myself$, where 1$\leq$k$\leq$n )
     then { become leader;
            exit; }
4) if ($d\_myself \leq d_k\ \forall$ k$\neq$ $myself$, where 1$\leq$k$\leq$n )
     then { move to the center of $SEC$ with probability $p = \frac{1}{n}$ of a
            distance $d\_myself \cdot p$)}
```

Algorithm 5.2. Leader election in systems of size $n > 3$

Definition 3 (Leader election legitimate configuration). *A legitimate configuration for Leader Election is a configuration with an unique robot closest to the center of the smallest enclosing circle.*

Lemma 3. *Algorithm 5.2 converges to a legitimate configuration for the leader election problem in a finite number of steps in expectation in the Corda model refined with a k-bounded scheduler.*

6 Preprocessing Module: Setting a Moving Formation

In this section we gradually set the motion pattern used further in the flocking algorithm. We build on top of the leader election algorithms proposed in Section 5.2.The construction takes two phases. First, all robots but the leader are placed on the smallest enclosing circle. Then, the robots on the circle will be placed in their final positions for motion.

6.1 Phase 1: Placement on the Smallest Enclosing Circle

In this section we propose an algorithm for placing robots on the smallest enclosing circle. This algorithm uses as building block the leader election algorithm proposed in the previous section. Once this algorithm is stabilized all robots but the leader are placed on the smallest enclosing circle. Note that the leader does not change during this phase.

The algorithm works "in waves". First, the robots closest to the bounds of the smallest enclosing circle are placed. Then, recursively all the other robots but the leader are placed. The robots that should occupy a position that is already occupied by another robot will be placed on a free position between the robot that occupied their position and the next one on the smallest enclosing circle. We assume the robots agree on the same direction of the Ox axis given by the center of SEC and the leader position and the same direction of Oy axis. Our algorithm is collision free and works in the Corda model with arbitrary fair scheduler. Note that in [25] the authors propose similar deterministic solutions for Suzuki-Yamashita model. Interestingly, our algorithm has the same time complexity as the solution proposed in [25].

The following definitions introduce key functions used by Algorithm 6.1.

Definition 4 (FreeToMove). *Let $FreeToMove$ be the set of robots without robots between themselves and the SEC (including the border) along the radius passing through them, and that do not belong to the border of SEC.*

Definition 5 (Placed). *Let Placed be the set of robots belonging to the border of the SEC.*

Definition 6. *A legitimate configuration for Algorithm 6.1 is a configuration where all robots but the leader are Placed.*

Note that the algorithm does not change the leader position neither the position of *Placed* robots. Moreover, there is no robot between the leader and the SEC. Otherwise this robot is the closest to the center of the SEC hence the real leader.

The correctness of Algorithm 6.1 comes from the following lemmas.

```
∀$r_i$ compute the value of the radius passing through $r_i$. Let $rad_{r_i}$ be the value
of the angle between my radius ($rad_{myself} = 0$) and the radius of robot $r_i$, in
clockwise direction
∀$r_i$ compute the value of $dist_{r_i}$, distance of the robot $r_i$ to the border of the
smallest enclosing circle ($SEC$)

Predicates:
$Leader(myself) \equiv \forall r_i$ with $i \neq myself$, $dist_i < dist_{myself}$

Functions:
$OccupiedPosition(rad_{myself})$ : returns $r_i, i \neq myself, dist_{r_i} = 0$ and $rad_{r_i} = rad_{myself}$ otherwise $\perp$
$NextToMove$ : returns the set of closest robots $r$ to the $SEC$ with $dist_r \neq 0$

1) if ( $\neg Leader(myself) \wedge myself \in FreeToMove$)
   then { move to $SEC$ with distance $dist_{myself}$}
2) if ($\neg Leader(myself) \wedge (myself \in NextToMove) \wedge (FreeToMove = \emptyset) \wedge (OccupiedPosition(rad_{myself}) \neq \perp)$)
   then { Move to the middle point of the arc between robot
          OccupiedPosition($rad_{myself}$) and robot $r_j$ belonging to the $SEC$ such
          that $rad_j$ is minimum.}
```

Algorithm 6.1. Positioning Algorithm executed by robot my_self

Lemma 4. *If two robots r_i and r_j belong to the set $FreeToMove$, then their final position will be different.*

Lemma 5. *A robot always moves towards a free position on the SEC.*

Lemma 6. *Algorithm 6.1 is collisions free (two robots never move towards the same free position).*

Lemma 7. *Algorithm 6.1 converges in a finite number of steps, $O(n)$, to a legitimate configuration.*

6.2 Phase 2: Setting the Flocking Configuration

In this section we propose an algorithm that starting from the final configuration of Algorithm 6.1 reaches a flocking pattern or moving formation having the singularity property detailed later.

Initially, we place robots in a circular moving formation then in the final moving formation. The circular moving formation has the following shape: r_0 is inside SEC (the one computed by Algorithm 6.1) and all the other robots are placed on its border. These robots are placed as follows: a robot r_1 is in the position $SEC \cap [O, r_0)$ and the others, uniformly disposed on the semi-circle that does not contain r_1 and that ends in the points given by the intersection of SEC with the perpendicular on $[O, r_0)$ that passes through O ($SEC \cap (\perp [O, r_0)$ *on* O)). In the following this configuration will be referred as *circular moving formation* (see Figure 2 for a seven robots example).

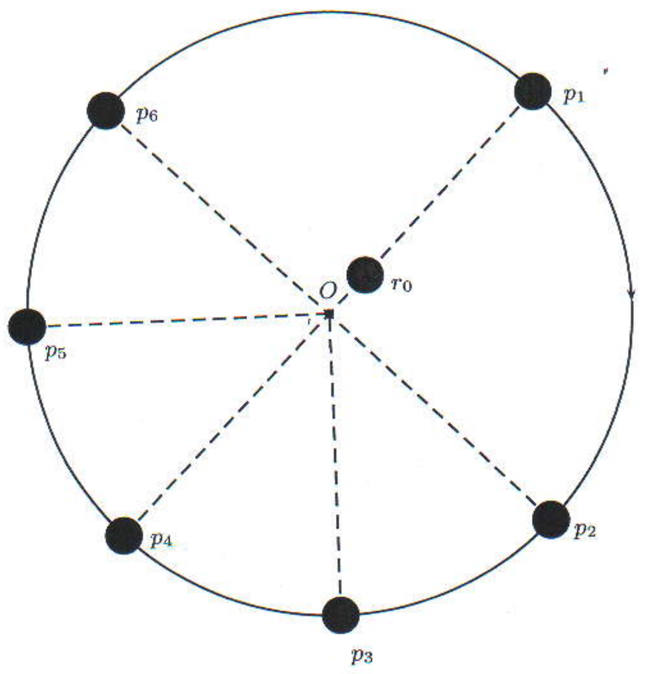

Fig. 2. Circular moving formation

In order to construct the circular moving formation we use the concept of oriented configuration [21]:

Definition 7 (oriented circular configuration). *A configuration is called circular oriented if the following conditions hold:*

1. *All robots are at distinct positions on the same circle SEC, except only one of them, called r_0, located inside SEC ;*
2. *r_0 is not located at the center of SEC;*

Note Algorithm 6.1 verifies point 1 of the above definition and is collisions free contrary to the solution proposed in [21]. Note also the leader election algorithm chooses a leader such that it is the closest to the center of SEC without reaching this center. If the leader is initially in the center, we recall that a preprocessing is performed in order to take care of this particular case. The leader election algorithm is executed only after the end of the preprocessing phase.

We now describe Algorithm 6.2. The algorithm makes use of the following function: FinalPositions(SEC, p_1). This function returns, when invoked by a robot, the set of positions in the circular moving formation with respect to SEC and the point p_1. p_1 is the intersection between the segment $[O, r_0)$ and the circle SEC. The order of positions and robots is given clockwise starting with position p_1. Started in an oriented configuration Algorithm 6.2 eventually converges to a configuration where robots are disposed on SEC following the restrictions imposed by the $FinalPositions$ function.

```
Functions:
    $get\_number(myself)$ returns the number of robots between $myself$ and
                position $p_1$ (including robot myself) clockwise
    $get\_position(myself)$ returns the position $get\_number(myself)$
                in $FinalPositions(SEC, p_1)$
    $FreeToMove(myself)$ returns true if there are no robots between $myself$
                and $get\_position(myself)$

Motion Rule:
        if FreeToMove(myself) then
            move to $get\_position(myself)$
```

Algorithm 6.2. Setting the moving formation executed by robot $myself$

The idea of the algorithm is as follows. Robots started in an oriented configuration reach their final positions. If a robot is blocked by some other robot then it waits until this robot is placed in its final position. In the following we prove no robot is blocked infinitely.

Lemma 8. *In a system with n robots Algorithm 6.2 started in an oriented configuration converges in finite number of steps, $O(n)$, to a configuration where all robots reach their final positions computed via FinalPositions function.*

We formally define the moving formation as follows:

Definition 8 (moving formation). *A set of $n > 4$ robots, $r_0, \dots r_n$, is a moving formation if:*

- *r_1 and r_0 define the Oy axis of the system such that: the y coordinate of r_0 equals 0 and the positive values are in the r_1 direction;*
- *the axis Ox is perpendicular to Oy in r_0 and has positive values at the right of Oy;*
- *all the other robots are such that:*

  1. *$\forall\ r_i \neq r_l$ and $r_i \neq r_0 \Rightarrow y_{r_i} < 0$*
  2. *$\forall r_i, r_j,\ x_{r_i} \neq x_{r_j}$;*
  3. *$\forall\ r_i,\ \exists\ r_j$ such that $x_{r_i} = -x_{r_j}$*
  4. *if $|x_{r_i}| > |x_{r_j}|$ then $|y_{r_i}| < |y_{r_j}|$*
  5. *there exists an unique robot with $x = 0$ and $y < 0$*

The following theorem states the singularity property of the moving formation defined above. More precisely, we show that there is only one formation that satisfies Definition 8 when $n > 4$. Note that for the case $n \leq 4$ the formation defined by Definition 8 is not unique. In the sequel we consider systems with more than 4 robots. For the case $n \leq 4$ simple adhoc algorithms can be designed on top of the algorithms proposed in Section 5.2.

Theorem 2. *The moving formation defined by Definition 8 is singular when* $n > 4$.

Corollary 1. *Algorithm 6.2 started in an oriented configuration eventually places n robots in a circular moving formation if FindPositions returns a set of positions verifying Definition 8.*

7 Flocking Module

In this section we propose a flocking algorithm. The flock of robots verifies the moving formation defined in Definition 8 and follow the head robot (the robot referred as robot r_1) whenever this head changes its position. In the following the robot r_0 of the moving formation will be called *reference robot* and the robot r_1 *leader*. The only constraint imposed to the system is: the leader cannot move quicker than the slowest robot in the set. The algorithm idea is as follows. When the head of the group moves, it is followed within a distance δ (a parameter of the algorithm) by the reference robot. Then the closest robots to the reference move within a distance ϵ (another parameter of the algorithm) to the reference and so on till all the robots in the group move. Note the algorithm has three parameters: the speed of the leader, the distance between the leader and the reference and the distance ϵ between the successive rows of robots. The moving formation can be seen as a virtual tree where levels are linked to each other via virtual springs (Figure 3).

```
Input: $r_0, r_1 \ldots r_n$ a moving formation

Functions:
        $TheMostExterior(r_{myself})$ returns true if $|x_{r_{myself}}| \geq |x_{r_i}|\ \forall r_i$
        $YClosestExterior(r_{myself})$ returns the y coordinate of $r_{ext}$, the robot
        such that $(|x_{r_{ext}}| - |x_{r_{myself}}|)$ is minimum and positive

1. if $(r_{myself} == r_1)$: { move ahead at a speed $< v_{max}$ }, $v_{max}$ is a
   parameter of the algorithm

2. if $(r_{myself} == r_0)$: { follow the leader within a distance $\delta$ }

3. if $(r_{myself} \neq r_0, r_1$ & $TheMostExterior(r_{myself})$ ): { move ahead
   following $y = y_{r_j}$ towards the point $(x_{r_j}, -\epsilon)$; }

4. $(r_{myself} \neq r_0, r_1$ &$\neg TheMostExterior(r_{myself}))$: { move ahead following
   $y = y_{r_j}$ towards the point $(x_{r_j}, YClosestExterior(r_{myself}) - \epsilon)$; }
```

Algorithm 7.1. Flocking executed by robot r_{myself}

Lemma 9. *Algorithm 7.1 preserves the moving formation (Definition 8).*

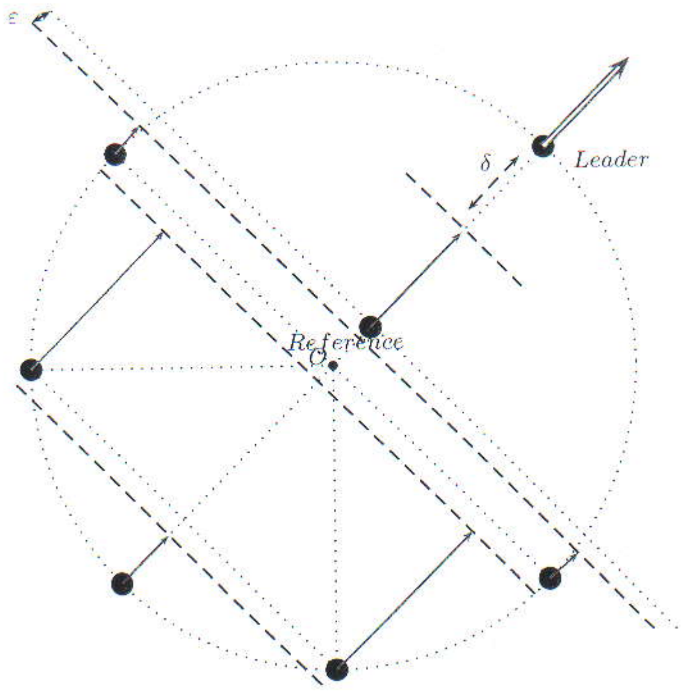

Fig. 3. Animation for Algorithm 7.1

8 Conclusions and Open Problems

In this paper we proposed an architecture for building a self-organizing and stabilizing flocking architecture. Contrary to the existing work on this topic our flocking architecture does not assume the existence of a specific leader a priory known to the robots in the network. In our approach robots are anonymous and uniform.

Our architecture includes three modules: a leader election module, a preprocessing module and a motion module. For each of these modules we propose deterministic or probabilistic algorithms (in the case when deterministic solutions are impossible).

This work can be seen as a preliminary study for the design of a general fault-tolerant flocking architecture where the group of robots verify a generic pattern and follow the head whatever its direction. Additionally, we currently investigate probabilistic algorithms that improve the leader election part of our architecture. In particular, we are looking for leader election solutions in the Corda model refined with an arbitrary scheduler. The idea of these algorithms is to use analogical strategies for election. That is, the robots candidate to a leader position choose probabilistically a free position on their corresponding radius in the smallest enclosing circle.

Acknowledgments

The authors would like to thank Xavier Defago for detailed discussions related to the current work, the suggested open problems and the various hints to solve these problems as discussed in Section 8. We also would like to thank Sebastien Tixeuil and anonymous reviewers for all their comments that greatly improved the presentation of the current work.

References

1. Prencipe, G.: Corda: Distributed coordination of a set of autonomous mobile robots. In: Proc. ERSADS 2001, pp. 185–190 (2001)
2. Flocchini, P., Prencipe, G., Santoro, N., Widmayer, P.: Distributed coordination of a set of autonomous mobile robots. In: IVS 2000. IEEE Intelligent Vehicles Symposium, pp. 480–485 (2000)
3. Suzuki, I., Yamashita, M.: Distributed anonymous mobile robots—formation and agreement problems. In: SIROCCO 1996. Proceedings of the 3rd International Colloquium on Structural Information and Communication Complexity (1996)
4. Suzuki, I., Yamashita, M.: Distributed anonymous mobile robots: Formation of geometric patterns. SIAM Journal on Computing 28(4), 1347–1363 (1999)
5. Suzuki, I., Yamashita, M.: A theory of distributed anonymous mobile robots formation and agreement problems. Technical report, Wisconsin Univ. Milwakee, Dep. of Electrical Engineering and Computer Science, 6 (1994)
6. Flocchini, P., Prencipe, G., Santoro, N., Widmayer, P.: Gathering of asynchronous mobile robots with limited visibility. Theoretical Computer Science 337, 147–168 (2005)
7. Souissi, S., Défago, X., Yamashita, M.: Eventually consistent compasses for robust gathering of asynchronous mobile robots with limited visibility. Research Report IS-RR-2005-010, JAIST, Ishikawa, Japan (2005)
8. Ando, H., Oasa, Y., Suzuki, I., Yamashita, M.: Distributed memoryless point convergence algorithm for mobile robots with limited visibility. IEEE Trans. on Robotics and Automation 15(5), 818–828 (1999)
9. Cohen, R., Peleg, D.: Convergence of autonomous mobile robots with inaccurate sensors and movements. In: Durand, B., Thomas, W. (eds.) STACS 2006. LNCS, vol. 3884, pp. 549–560. Springer, Heidelberg (2006)
10. Agmon, N., Peleg, D.: Fault-tolerant gathering algorithms for autonomous mobile robots. In: SODA 2004. Proc. 15th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 1070–1078 (2004)
11. Défago, X., Gradinariu, M., Messika, S., Parvédy, P.R.: Fault-tolerant and self-stabilizing mobile robots gathering. In: Dolev, S. (ed.) DISC 2006. LNCS, vol. 4167, pp. 46–60. Springer, Heidelberg (2006)
12. Kasuya, M., Ito, N., Inuzuka, N., Wada, K.: A pattern formation algorithm for a set of autonomous distributed robots with agreement on orientation along one axis. Systems and Computers in Japan 37(10), 89–100 (2006)
13. Qadi, A., Huang, J., Farritor, S.M., Goddard, S.: Localization and follow-the-leader control of a heterogeneous group of mobile robots. IEEE/ASME Transactions on Mechatronics 11, 205–215 (2006)
14. Renaud, P., Cervera, E., Martiner, P.: Towards a reliable vision-based mobile robot formation control. In: IROS. IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 3176–3181 (2004)

15. Lindhe, M.: A flocking and obstacle avoidance algorithm for mobile robots. PhD thesis, KTH Stockholm (2004)
16. Gervasi, V., Prencipe, G.: Flocking by a set of autonomous mobile robots. Technical Report TR-01-24, Universitat di Pisa (2001)
17. Gervasi, V., Prencipe, G.: Coordination without communication: the case of the flocking problem. Discrete Appl. Math. 144(3), 324–344 (2004)
18. Angluin, D., Aspnes, J., Fischer, M., Jiang, H.: Self-stabilizing population protocols. In: Anderson, J.H., Prencipe, G., Wattenhofer, R. (eds.) OPODIS 2005. LNCS, vol. 3974, pp. 79–90. Springer, Heidelberg (2006)
19. Fischer, M., Jiang, H.: Self-stabilizing leader election in networks of finite-state anonymous agents. In: Shvartsman, A.A. (ed.) OPODIS 2006. LNCS, vol. 4305, pp. 395–409. Springer, Heidelberg (2006)
20. Barri, L., Flocchini, P., Fraigniaud, P., Santoro, N.: Electing a leader among anonymous mobile agents in anonymous networks with sense-of-direction. Technical Report 1310, Technical Report LRI, Laboratoire de recherche en Informatique, Université Paris-Sud, France, (April 2002)
21. Dieudonne, Y., Petit, F.: Circle formation of weak robots and lyndon words. Inf. Process. Lett. 101(4) (2007)
22. Prencipe, G.: Achievable patterns by an even number of autonomous mobile robots. Technical Report TR-00-11, Universitat di Pisa (2000)
23. Canepa, D., Gradinariu, M.: Stabilizing flocking via leader election in robot networks. Technical Report 6268, INRIA, France (2007)
24. Dolev, S.: Self-stabilization. MIT Press, Cambridge (2000)
25. Suissi, S.: Fault resilient cooperation of Autonomous Mobile robots with unreliable compass sensors. PhD thesis, JAIST, Japon (2007)

Abst
Wha
becau
or ex
of op
dard
and
stabi
nal i
two
egy
cont
syst
plica

Key

1 Intr

System de
performar
in many s
of operati
ating cont
is achieve
continuou
changes, a
state, as t
region of
property

The st
of the cu
tually co
system w
to mainta
mality. St

T. Masuzav
© Springer

Toshimitsu Masuzawa
Sébastien Tixeuil (Eds.)

LNCS 4838

Stabilization, Safety, and Security of Distributed Systems

9th International Symposium, SSS 2007
Paris, France, November 2007
Proceedings

SSS
2007

Springer