Uniform Circle Formation by Autonomous Mobile Robots

A Project Report Submitted

for the Course

MA498 Project I

by

Chirag Maheshwari

(Roll No. 10012315)

to the

DEPARTMENT OF MATHEMATICS

INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI

GUWAHATI - 781039, INDIA

November 2013

CERTIFICATE

This is to certify that the work contained in this project report entitled "**Uniform Circle Formation by Autonomous Mobile Robots**" submitted by **Chirag Maheshwari** (**Roll No.: 10012315**) to Department of Mathematics, Indian Institute of Technology Guwahati towards the requirement of the course **MA498 Project I** has been carried out by him/her under my supervision.

Guwahati - 781 039
November 2013

(Dr. P.S. Mandal)
Project Supervisor

ABSTRACT

The main aim of the project is to propose and test by simulations a distributed algorithm by which a set of autonomous, anonymous mobile robots roaming on a plane move to form a uniform circle. The robots are anonymous in the sense that they all execute the same algorithm and they cannot be distinguished by their appearances. Two algorithms proposed tries to solve the uniformity problem presuming that the robots already lie on a circle (non-uniform) and have a common sense of rotational orientation.

The distributed coordination and control of a set of autonomous mobile robots is a problem widely studied in a variety of fields, such as engineerig, artificial intelligence, artificial life, robotics. The problem is practically important, because, if the robots can form a given pattern, they can agree on their respective roles in any coordinated actions.

Contents

List of Figures

List of Algorithms

1

Introduction

Suppose that a schoolteacher wants her 100 children in the playground to form a circle so that, for instance, they can play a game. She might draw a circle on the ground as a guideline or even give each child a specific position to move to. What if the teacher does not provide such assistance? Even without such assistance, the children may still be able to form a sufficiently good approximation of a circle if each of them moves adaptively based on the movement of other children and knowledge of the shape of a circle. If successful, this method can be called a distributed solution to the circle formation problem for children. A similar distributed approach can be used for

controlling a group of multiple mobile robots.

An interesting trend has been observed in robotic research, both from engineering and behavioural viewpoints. They are readily moving away from the design and deployment of few, rather complex, usually expensive, application-specific robots. In fact, within this trend, the interest has shifted towards the design and use of a large number of "generic" robots which are very simple, with very limited capabilities, and thus relatively inexpensive but capable of performing (together) rather complex tasks. The main idea is to let each robot execute a simple algorithm and plan its motion adaptively based on the observed movement of other robots, so that the robots as a group will achieve the given goal.

The advantages of such an approach are clear and many. They include: reduced costs (due to simpler engineering and construction costs, faster development and deployment time, etc); ease of system expandability (just add a few more robots) which in turns allows for incremental and on-demand deployment (use only as few robots as you need and when you need them); simple and affordable fault-tolerance capabilities (replace just the faulty robots); re-usability of the robots in different applications (reprogram the system to perform a different task).

1.1 Model and Problem

The robots are modelled as *points* a 2D plane. They are "weak" robots: *homogeneous* (they all follow the same set of rules), *autonomous* (there is no a priori central authority, and each robots computing capabilities are independent from the others), *asynchronous* (there is no central clock, no a priori synchronization, no a priori bounds on processing or motorial speed), *mobile*

(robots are allowed to move on a plane), *anonymous* (they are a priori indistinguishable), oblivious (they do not explicitly remember the past). Besides, robots execute the same deterministic algorithm, are unable to communicate directly, and can only interact by observing each others' position.

With this weak model, we address the problem of forming an uniform circle by a group of mobile robots, for which two algorithms are proposed. The problem in particular has interesting applications. For instance, consider the context of space exploration and the initial preparation of a zone. A group of robots could be sent and after landing at random locations, would self-organize to form the initial infrastructure for later expeditions. Also, pattern formation is the first step towards flocking, i.e., allowing a group of robots to move in formation. Moreover, the formation of geometrical patterns and flocking are both useful in themselves for the self-positioning of mobile base stations in a mobile ad-hoc network and self-deployment of sensor rings.

Forming of an uniform circle by a group of autonomous robots can be divided into two parts:

1. Forming a circle, possibly an non-uniform one. (Circle Formation Problem)

2. Positioning the robots evenly on the boundary, i.e., forming a regular polygon. (Uniformity Transformation)

The Circle Formation Problem by a set of autonomous, anonymous, oblivious robots has many proposed solutions whereby robots deterministically form a circle. These solutions will be discussed in the next section of literature review. Unfortunately, the second part of the problem only has converging solutions. Although, deterministic solutions have been proposed, but they do not work for n(number of robots) = 4, 6, 8. Consequently, the two

algorithms presented here will try to solve the uniformity problem. Both the algorithm assume that all the robots have a common set of roational orientation. Former of the two algorithms presumes that one of the robot is faulty and cannot move from it's position after forming the circle. Latter of the two algorithms restricts the movement of all the robots only in the anti-clockwise direction. Both of them determinitically position all the robots evenly on the boundary of the circle forming a regular polygon and ultimately solving the uniformity problem.

The algorithms have not been rigorously proven, but an animated simulation in *webGL* of the working of the algorithm is constructed to test the hypothetical results. The results and images from the simulation are added in the subsequent section.

1.2 Definitions

We study the problem of coordinating a set of *autonomous*, mobile robots in the plane. The coordination mechanism must be totally decentralized, without any central control. The robots are *anonymous*, in the sense that a robot does not have an identity that it can use in a computation, and all robots execute the exact same algorithm. Each robot has its own, *local view* of the world. This view includes a local Cartesian coordinate system with origin, unit of length, and the directions of two coordinate axes, identified as x axis and y axis, together with their orientations, identified as the positive sides of the axes. The robots do not have a common understanding of the handedness (chirality) of the coordinate system that allows them to consistently infer the orientation of the y axis once the orientation of the x axis is given; instead, knowing North does not distinguish East from West.

The robots observe the environment and move; this is their only means of communication and of expressing a decision that they have taken. The only thing that a robot can do is make a step, where a step is a sequence of three actions. First, the robot observes the positions of all other robots with respect to its local coordinate system. Each robot is viewed as a point, and therefore the observation returns a set of points to the observing robot. The robot cannot distinguish between its fellow robots; they all look identical. Second, the robot performs an arbitrary local computation according to its algorithm, based only on the common knowledge of the world (assumed e.g. to be stored in read-only-memory and to be read off from sensors of the environment) and the observed set of points. Since the robot does not memorize anything about the past, we call it *oblivious*. For simplicity, we assume that the algorithm is deterministic, but it will be obvious that all of our results hold for nondeterministic algorithms as well (randomization, however, makes things different). Third, as a result of the computation, the robot either stands still, or it moves (along any curve it likes). The movement is confined to some (potentially small) unpredictable, nonzero amount. Hence, the robot can only go towards its goal along a curve, but it cannot know a priori how far it will come in the current step. While it is on its continuous move, a robot may be seen an arbitrary number of times by other robots, even within one of its steps.

Let π be a predicate describing a geometric pattern, such as a point, a regular polygon, a line segment, etc. On the one hand, we say that an algorithm $\mathcal{A}$ solves the ***convergence problem*** for π if the robots distribution converges to one that satisfies π, regardless of the number n of robots, their initial distribution, and the timing with which they become active. On the other hand, we say that $\mathcal{A}$ solves the ***formation problem*** for π if the

robots eventually reach a distribution that satisfies π in a finite number of steps, regardless of n, their initial distribution, and the timing with which they become active.

The robots are *asynchronous*: the amount of time spent in observation, in computation, in movement, and in action is negligible but otherwise unpredictable. In particular, the robots do not (need to) have a common notion of time. Each robot makes steps at unpredictable time instants. The (global) time that passes between two successive steps of the same robot is finite; that is, any desired finite number of steps could have been made by any robot after some finite amount of time. In addition, we do not make any timing assumptions within a step: The time that passes after the robot has observed the positions of all others and before it starts moving is arbitrary, but finite. That is, the actual move of a robot may be based on a situation that lies arbitrarily far in the past, and therefore it may be totally different from the current situation. We feel that this assumption of asynchronicity within a step is important in a totally asynchronous environment, since we want to give each robot enough time to perform its local computation.

The robots are able to sense the complete plane: we say they have *Unlimited Visibility*. If they have limited visibility, then a visibiity graph for the positions of the robots is defined as:

Definition 1.2.1. *Visibility Graph.* The visibility graph $\mathrm{G} = (\mathrm{N}, \mathrm{E})$ is a graph whose node set N is the set of the input robots and $(r_i, r_j) \in \mathrm{E}$ iff $r_j \in \mathcal{C}_i$ and $r_i \in \mathcal{C}_j$, where $r_i, r_j, \mathcal{C}_i, \mathcal{C}_j$ are the two and robots and their visibility area from their initial positions.

The robots to be able to detect *multiplicity* (i.e. whether there is more than one robot on any of the observed points, included the position where the observing robot is.

1.2.1 Computational Cycle

The robots execute the same deterministic algorithm, which takes as input the observed positions of the robots within the visibility radius, and returns a destination point towards which the executing robot moves. A robot is initially in a waiting state (Wait); asynchronously and independently from the other robots, it observes the environment in its area of visibility (Look); it calculates its destination point based only on the observed locations of the robots in its (Compute); it then moves towards that point (Move); after the move it goes back to a waiting state. The sequence: Wait Look Compute Move will be called a computation cycle (or briefly cycle) of a robot. The operations performed by the robots in each state will be now described in more details.

Wait The robot is idle. A robot cannot stay idle indefinitely unless it is faulty. At the beginning all the robots are in the Wait state.

Look The robot observes the world by activating its sensors which will return a *snapshot* of the positions of all other robots with respect to its local coordinate system. Each robot is viewed as a point, hence its position in the plane is given by its coordinates, and the result of the snapshot is just a set of coordinates in its local coordinate system: this set forms the *view of the world* of r . More formally, the view of the world of r at time t is defined as the last snapshot taken at a time smaller than or equal to t.

Compute The robot performs a local computation according to its deterministic, oblivious algorithm $\mathcal{A}$. The result of the computation is a destination point; if this point is the current location, the robot stays still (null movement).

Move If the point computed in the previous state is the current location, the robot does not move; otherwise it moves towards the destination point. The robot moves by an unpredictable amount of space, which is assumed neither infinite, nor infinitesimally small. Hence, the robot can only go towards its goal, but it cannot predict how far it will go in the current cycle, because it can stop anytime during its movement; that is, a robot can stop before reaching its destination point.

1.2.2 Time Settings

Asynchronous. In this time setting, the global time that passes between two successive states of the same robot is finite but unpredictable. In addition, no time assumption within a state is made. This implies that the time that passes after the robot starts observing the positions of all others and before it starts moving is arbitrary, but finite. That is, the actual move of a robot may be based on a situation that was observed arbitrarily far in the past, and therefore it may be totally inaccurate in the current situation.

The system resulting from this time setting is *fully asynchronous*; in particular, the amount of time spent in *Wait, Look, Compute, Move*, and idle states is finite but otherwise unpredictable. As a result, the robots do not have a common notion of time, robots can be seen while moving, and computations can be made based on obsolete observations. This time setting is adopted in [8]; we will refer to it as Async. If the robots move according to this time setting, we say that they move according to an *asynchronous activation schedule.*

Atomic. In contrast, if the robots execute their activities in an atomic and instantaneous fashion, we say that the robots are *atomically synchronized*, and that they move according to an atomic activation schedule. It is referred

to as Atom

In an atomic activation schedule, at each time instant t, every robot r_i is either *active* or *inactive*. At least one robot is active at every time instant, and every robot becomes active at infinitely many unpredictable time instants. For any $t \geq 0$, if r_i is inactive, then $p_i\ (t + 1) = p_i\ (t)$; otherwise $p_i\ (t + 1) = p$, where $p_i\ (t)$ denotes the position of robot r_i at time instant t, and p is the point returned by $\mathcal{A}$.

1.2.3 Problem Definition

The problem addressed is the formation of a circle by a set of autonomous mobile robots. More rigorously the problem is defined as follows.

Definition 1.2.2. *Uniform Circle Formation.* Given a group of n robots $r_1, r_2, ..., r_n$ with distinct positions and located arbitrarily on a plane on the boundary of some non-degenrate circle (i.e., with finite radius greater than zero), eventually arrange them at regular intervals of the boundary of the circle.

A weaker problem is also considered as a part of the literature review which requires the robots to form a circle, but not necessarily be at regular intervals. This weaker problem is expressed more rigorously as follows.

Definition 1.2.3. *Circle Formation.* Given a group of n robots $r_1, r_2, ..., r_n$ with distinct positions and located arbitrarily on the plane, arrange them to eventually form a non-degenrate circle.

Definition 1.2.4. *Gathering Problem.* Given a group of n robots $r_1, r_2, ..., r_n$ with distinct positions and located arbitrarily on the plane, eventually gather at one point.

Definition 1.2.5. *Pattern Formation Problem.* Given a group of n robots $r_1, r_2, ..., r_n$ located arbitrarily on the plane are said to form a given pattern, if, eventually the positions of the robots coincide, in everybody's local view, with the points of the pattern.

In terms of reaching agreement, Circle Formation provides an origin and a unit distance.

1.3 Future Prospects

The present work can be extended in the followinf directions:

- The assumptions taken with the algorithm can be relaxed to make weaker models and solve Uniform transformation problem in finite time.
- Different settings of obliviousness can be tested. For instance, a totally non-oblivious model, i.e., with unlimited amount of memory. Alternatively, equipping the robot with bounded memory (semi-obliviousness). Or the weakest model with no memory of past (oblivious).
- Weaker models can be constructed wherein the three actions (look-compute-move) may not be atomic.
- Limitless region of vision can be limited.
- Reducing the complexity of the algorithm.
- Instead of forming regular patterns, forming irregular/arbitrary patterns.
- Taking into consideration the *practical issues*. The algorithm presented here is more theoritical rather than practical. Infinite precision, unlim-

ited visibility, point robots, are impractical assumptions. Engineering issues is an important topic.

- fault tolerance. oblivious algorithms are by definition self-stabilizing but non-oblivious are not. Also, the algorithm should also work when the number of robots changes dynamically a finite number of time.

2

Literature Review

A vast amount of researches exist in the context of cooperative mobile robotics. Most of it uses diverse heuristics such as free market optimization [4] or swarm intelligence [7]. However, only few studies take the problem from a computational standpoint. This can be partly explained by the difficulty of the task, and the fact that heuristics are perceived as a way to circumvent that difficulty. Debest [2] briefly discusses the formation of a circle by a group of mobile robots as an illustration of self-stabilizing distributed algorithms. He discusses the problem, but does not really provide an algorithm. Sugihara and Suzuki [12] propose several algorithms for the formation of various

geometrical patterns. They propose an algorithm for the formation of an approximation of a circle, based on heuristics. In some cases, the shape obtained with their algorithm is a Reuleaux triangle (a hybrid shape, between a triangle and a circle) rather than a circle. Suzuki and Yamashita [13] propose a non-oblivious algorithm for the formation of a regular polygon. In other words, the robots eventually reach a configuration in which they are arranged at regular intervals on the boundary of a circle. To achieve this, they must however require the robots to be able to remember all past actions. Under the same model, Ando et al. [6] propose an algorithm by which robots with a limited range of vision gather to a point. Flocchini et al. [9] give an algorithm to solve the same problem in a slightly different model; dropping the assumption of instantaneous computation and movement, but assuming a common sense of direction as given by compasses. Flocchini et al. [8] study the solvability of the formation of arbitrary patterns, depending on how much common knowledge the robots initially have about a global coordinate system. Uny Cao et al. [14] provide a wide survey of researches in cooperative mobile robotics, and observe that only few researches take the problem from a computational point of view. This observation is later echoed by Flocchini et al. [9]. The subsequent section describes the assumptions and results collected from various research papers.

2.1 Gathering Problem

From the works of G. Prencipe [10], given n robots with the following capabilities cannot solve the gathering problem:

- Autonomous
- Anonymous

- Homogeneous
- Oblivious
- Asynchronous, Synchoronized
- Unlimited Visibilty
- Dimensionless robots (point robots)
- Do not detect *multiplicity*
- Initial positions are all distinct

Under the above assumptions G. Prencipe [10] says that,

Theorem 2.1.1. *In both the asynchronous and the atomic time setting, there exists no deterministic oblivious algorithm that solves the gathering problem in a finite number of cycles, hence in finite time, for a set of $n \geq 2$ robots.*

In another work by M. Yamashita [13], after taking an extra assumption of multiplicity detection, they proved that gathering problem is solvable. The assumptions taken by M. Yamashita [13] are as follows:

- Autonomous
- Anonymous
- Homogeneous
- Oblivious
- Asynchronous, Synchoronized
- Unlimited Visibilty

- Dimensionless robots (point robots)
- **Detect *multiplicity***
- Initial positions are all distinct

Under the above assumptions, M. Yamashita [13] quotes,

Theorem 2.1.2. *There exists an oblivious algorithm for solving gathering problem in a finite number of steps for $n \geq 3$.*

Proof. It suffices to give an oblivious algorithm $\mathcal{A}$ that solves the gathering problem in a finite number of cycles. The idea is the following. Starting from distinct initial positions, we move the robots in such a way that eventually there will be exactly one position, say, p, that two or more robots occupy. Once such a distribution is reached, all robots that are not located at p move toward p in such a way that no two robots will occupy the same position at any location other than p. Then all robots eventually occupy p, solving gathering problem in finite time.

Such a distribution can be obtained if each robot, each time it becomes active, determines which of the following cases applies and moves to a new position (or remains stationary) as specified. Since a robots action is based only on the current robot distribution, this strategy can be implemented as an oblivious algorithm.

Case 1. $n = 3$ p_1, p_2 and p_3 denote the positions of the three robots

1. If $n = 3$ and p_1, p_2, p_3 are collinear with p_2 in the middle, then the robots at p_1 and p_3 move toward p_2 while the robot at p_2 remains stationary. Then eventually two robots occupy p_2.

2. If $n = 3$ and p_1, p_2 and p_3 form an isosceles triangle with $\|\overline{p_1p_2}\| = \|\overline{p_1p_3}\| \neq \|\overline{p_2p_3}\|$, then the robot at p_1 moves toward the foot of the perpendicular drop from its current position to p_2 p_3 in such a way that the robots do not form an equilateral triangle at any time, while the robots at p_2 and p_3 remain stationary. Then eventually the robots become collinear and the problem is reduced to part 1.

3. If $n = 3$ and the lengths of the three sides of triangle $p_1p_2p_3$ are all different, say, $\|\overline{p_1p_2}\| > \|\overline{p_1p_3}\| > \|\overline{p_2p_3}\|$, then the robot at p_3 moves toward the foot of the perpendicular drop from its current position to p_1p_2 while the robots at p_1 and p_2 remain stationary. Then eventually the robots become collinear and the problem is reduced to part 1.

4. If $n = 3$ and p_1, p_2, p_3 form an equilateral triangle, then every robot moves towards the center of the triangle. Since all robots can move up to at least a constant distance ϵ greater than 0 in one step, if part 4 continues to hold then eventually either the robots meet at the center, or the triangle they form becomes no longer equilateral and the problem is reduced to part 2 or part 3.

Case 2. $\boldsymbol{n \geq 4}$ C_t denotes the smallest enclosing circle of the robots at time t

1. If $n \geq 4$ and there is exactly one robot r in the interior of C_t, then r moves toward the position of any one robot, say, r , on the circumference of C_t while all other robots remain stationary. Then eventually r and r occupy the same position.

2. If $n \geq 4$ and there are two or more robots in the interior of C_t , then these robots move toward the center of C_t while all other robots

remain stationary (so that the center of C_t remains unchanged). Then eventually at least two robots reach the center.

3. If $n \geq 4$ and there are no robots in the interior of C_t , then every robot moves toward the center of C_t . Since all robots can move up to at least a constant distance ϵ greater than 0 in one step, if part 3 continues to hold, then eventually the radius of C_t becomes at most . Once this happens, then the next time some robot moves, say, at t' , either (i) two or more robots occupy the center of C_t or (ii) there is exactly one robot r at the center of C'_t , and therefore there is a robot that is not on C_t (and the problem is reduced to part 1 or part 2) since a cycle passing through r and a point on C_t intersects with C_t at most at two points.

□

2.2 Regular Pattern

From the works of M. Yamashita [13], given n robots can solve the pattern formation problem under the following assumptions:

- Autonomous
- Anonymous
- Homogeneous
- ***nonoblivious***
- Asynchronous, Synchoronized
- Unlimited Visibilty

- Dimensionless robots (point robots)
- **Detect *multiplicity***
- Initial positions are all distinct

M. Yamashita [13] says that,

Theorem 2.2.1. *There exists an algorithm for solving formation problem for a predicate π iff either $\pi = \pi_{point}$ or $\pi = \pi_{regular}$*

Although, M. Yamashita [13] proves that regular pattern formation is solvable if the robots are *nonoblivious*, he poses the same problem by *oblivious* robots regardless of initial state as an open problem.
In a research by P. Widmayer [5] some possibility and impossibility results on pattern formation were given under the following assumptions:

- Autonomous
- Anonymous
- Homogeneous
- ***oblivious***
- Asynchronous, Synchoronized
- Unlimited Visibilty
- Dimensionless robots (point robots)

The results can be summarized as:

Theorem 2.2.2. *1. With common knowledge of two axes directions and orientations, the robots can form an arbitrary given pattern.*

2. *With common knowledge on only one axis direction and orientation, the pattern formation problem is unsolvable when n is even; it can be solved if n is odd.*

3. *With common knowledge on only one axis direction, the robots can form an arbitrary pattern if n is odd.*

4. *With no common knowledge, the robots cannot form an arbitrary given pattern.*

Theorem 2.2.3. *With common knowledge on only one axis direction and orientation, there exists no deterministic algorithm that allows an even number of robots to form an asymmetric pattern. Moreover, in this case they can only form symmetric patterns that have at least one symmetric axis not passing through a vertex of the input pattern.*

Results on limited visibilty were also given in P. Widmayer [5]. These results can be summarized as:

Lemma 2.2.4. *If the visibility graph is disconnected, the pattern formation problem (or actually any problem) is unsolvable.*

Theorem 2.2.5. *There exists a deterministic algorithm that let the robots gather in one point in a finite number of movements, in the limited visibility setting and assuming common knowledge on direction and orientation of both axes.*

3

Uniform Circle Formation

Uniform circle Formation problem can be divided into two subparts as follows:

1. Forming a circle, possibly an non-uniform one. (Circle Formation Problem)

2. Positioning the robots evenly on the boundary, i.e., forming a regular polygon. (Uniform Transformation)

The algorithms to solve the above two problems have been given separately in the upcoming sections. Starting with the definitions and nota-

tions required, algorithm for circle formation and uniform transformation are given.

A deterministic algorithm to form a non-uniform circle is provided. Unfortunately, the uniform transformation problem is only solved as a "convergence problem". But, assuming some constraints and power on the robots, the uniform transformation problem can be solved deterministically in finite time. These assumptions are:

- Robots have a common sense of "rotational" orientation.
- The two oblivious algorithms presented separately assume the following:

    1. There is one faulty robot, i.e., a robot which cannot move when the robots are on the boundary of a circle.
    2. Robots can only move in the anti-clockwise direction. (All robots will move in the same direction as they have a common send of orientation).

3.1 Definitions and Notations

Some definition and notations used in the presentation of the algorithm.

Definition 3.1.1. *Position.* Given a robot r_i, $p_i(t)$ denotes its position at time t, according to some global x-y coordinate system, and $p_i(0)$ is its initial position. P $(t) = p_i$ (t) $1 \leq$ i $\leq n$ denotes the multiset of the positions of all robots at time t. When this is not ambiguous, we sometimes mention a robot, implicitly referring to its position rather than the robot itself.

Definition 3.1.2. *Voronoi Diagram.* The Voronoi diagram Voronoi(P) of a set of points P = $p_1, p_2, ..., p_n$ is a subdivision of the plane into n cells, one for each point in P. The cells have the property that a point q belongs to the Voronoi cell of point p_i, denoted $V_{cell_{p_i}}$ (P), if and only if, for any other point $p_j \in$ P, dist(q, p_i) ¡ dist(q, p_j), where dist(p, q) is the Euclidean distance between p and q. In particular, the strict inequality means that points located on the boundary of the Voronoi diagram do not belong to any Voronoi cell. A Voronoi diagram is for instance depicted on Figure 3.1. Significantly more details about Voronoi diagrams and their principal applications are surveyed by Aurenhammer [1].

Definition 3.1.3. *Smallest enclosing circle.* The smallest enclosing circle of a set of points P is denoted by SEC(P). It can be defined by either two opposite points, or by at least three points. The smallest enclosing circle is unique, and can be computed in O(n log n) [11].

3.2 Circle Formation

Assumptions taken:

- Autonomous
- Anonymous
- Homogeneous
- Oblivious
- Asynchronous, Synchoronized
- Unlimited Visibilty

- Dimensionless robots (point robots)
- *multiplicity*
- Initial positions are all distinct

In addition to the above assumptions, we impose some restriction on the movement of the robots to solve the problem. Doing so ensures that (1) no two robots occupy the same position simultaneously (Restr. 1), and that (2) the smallest circle enclosing all robots remains invariant (Restr. 24). The restrictions are (they are also shown as figures):

1. A robot always moves toward a point that is inside its Voronoi cell.
2. No robot ever moves beyond the boundary of the smallest circle enclosing all robots.
3. All robots located on the boundary of the smallest enclosing circle remain on that boundary.
4. Robots located on the circumference of the smallest enclosing circle do not move unless there are at least three such robots with distinct positions.

"Algorithm 1" given by A. Konagaya [3] for circle formation. The given oblivious algorithm deterministically solves the circle formation problem in a finite number of cycles.

3.3 Uniform Transformation

The "Algorithm 2" given by A. Konagaya [3] converges to the uniform transformation. It may or may not ever.

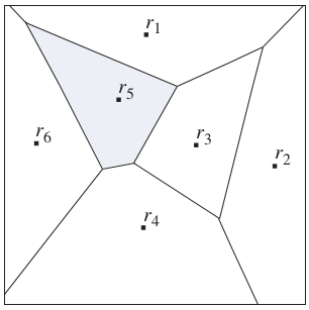

Figure 3.1: R
estriction 1: The movement of r_5 is constrained by the interior of its voronoi cell

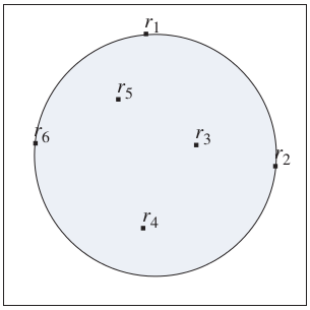

Figure 3.2: R
estriction 2: The movement of r_5 is constrained by smallest enclosing circle

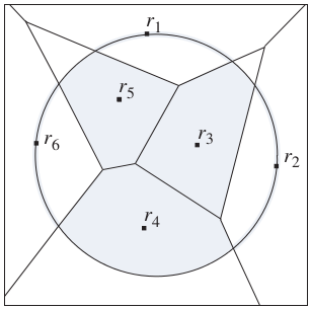

Figure 3.3: R
estriction 1-4: r_1, r_2, r_6 move on the boundary of the enclosing circle (Restriction 3-4): r_3, r_4, r_5 move in interior

Algorithm 1 Formation of an (arbitrary) circle (code executed by robot r_i)

function $\mathcal{A}_{circle}$(P, p_i)

1: **if** $p_i \in$ SEC(P) **then**
   {r_i already on boundary.}
   stay still
2: **else**
3: **if** $V_{cell_{p_i}}(\mathrm{P}) \bigcap \mathrm{SEC(P)} \neq \Phi$ **then**
4: target $\leftarrow V_{cell_{p_i}}(\mathrm{P}) \bigcup \mathrm{SEC(P)} \bigcup$ Voronoi(P - $\{p_i\}$)
      move to target
5: **else**
        {Voronoi cell of p_i inside circle}
        compute points in $V_{cell_{p_i}}$(P) closest to SEC(P).
        if exactly one candidate exists **then** move toward that point
6: **else**
        several candidates exists select the coordinate with the greatest x-coordinate and then y-coordinate. move toward that point
8: **end if**
9: **end if**
10: **end if**

Algorithm 2 Convergence towards a uniform transformation

function $\mathcal{A}_{uniform}$(P, p_i)

Require: Assume all robots are on the boundary of SEC(P)

1: prev(p_i) $\leftarrow$ direct neighbour of p_i counterclockwise
2: next(p_i) $\leftarrow$ direct neighbour of p_i clockwise
3: midpoint(p_i) $\leftarrow$ midpoint of arc prev(p_i) & next(p_i)
4: target $\leftarrow$ midpoint of midpoint(p_i) move toward target.

Now the algorithms presented have been proposed for some special situations.

3.3.1 Situation 1

Assumming all the robots form a circle and suddenly exactly one of the robots stops moving (maybe it is faulty or intentional). Now, the given algorithm in "Algorithm 3" deterministically solves the uniform transformation problem in a finite number of cycles.

Algorithm 3 uniform transformation with one faulty robot

function $\mathcal{A}_{uniform}$(P, p_i)
Require: Assume all robots are on the boundary of SEC(P)
Ensure: One of the robots is faulty (cannot move)

```
1: if r_i is falty then
2:    print  This robot cannot move.
3: else
4:    next(r_i) ← direct neighbour of r_i clockwise
5:    n ← total number of robots
6:    requiredAngularDiff ← 2*π/n
7:    angularDiff ← (arc between next(r_i))/(radius of circle)
8:    if requiredAngularDiff > angularDiff then
9:       target ← (requiredAngularDiff - angularDiff)*(radius of circle) in
         anticlockwise direction move toward target.
10:   else
11:      if requiredAngularDiff < angularDiff then
12:         target ← (angularDiff - requiredAngularDiff)*(radius of circle) in
            clockwise direction move toward target
13:      else
14:         target ← currentPosition stand still.
15:      end if
16:   end if
17: end if
```

Proof. "Algorithm 3" deterministically solves the problem of uniform transformation. Consider the robot just prev of the "immobile" robot, say r_f.

r_f robot will deterministically position itself in a finite number of cycles in its prescribed position as its neighbour robot is not moving, thus, fixing its target position for all the cycles. Now this r_f robot also becomes "immobile" as it is on its target position. Recursively considering the robot r_{f-1}, this will also deteministically position itself on the target position in a finite number of cycles. Similarly, all robots will move to their target position and will form a regular-polygon solving the uniform transformation problem.

This algorithm was simulated, and as derived, the movement of the robots stopped after a finite number of cycles forming an uniform circle/regular polygon. This result is shown in the pictures.

□

3.3.2 Situation 2

Assumming all the robots form a circle and we restrict the movement in only counterclockwise direction. Then "Algorithm 4" deterministically solves the problem of uniform transformation.

Algorithm 4 uniform transformation with movement restriction in one direction

function $\mathcal{A}_{uniform}$(P, p_i)
Require: Assume all robots are on the boundary of SEC(P)
Ensure: Robots can only move counterclockwise
1: prev(p_i) ← direct neighbour of p_i counterclockwise
2: next(p_i) ← direct neighbour of p_i clockwise
3: midpoint(p_i) ← midpoint of arc prev(p_i) & next(p_i)
4: target ← midpoint of midpoint(p_i)
5: **if** target in counterclockwise direction **then**
    Move the robot r_i to target
6: **else**
    stand still.
7: **end if**

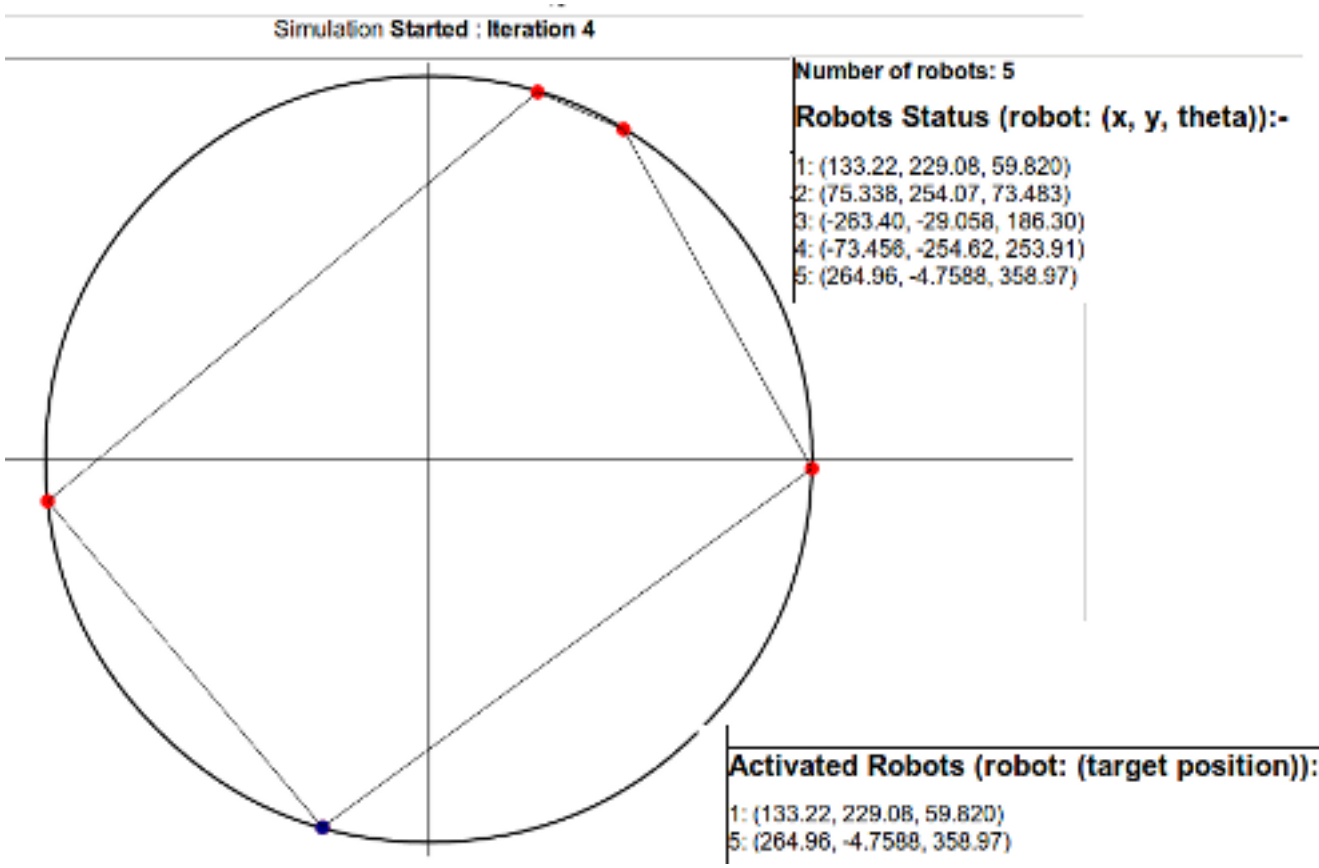

Figure 3.4: I
nitial states of all the robots. The faulty robots is shown in blue color
(although it is anonymous in theory)

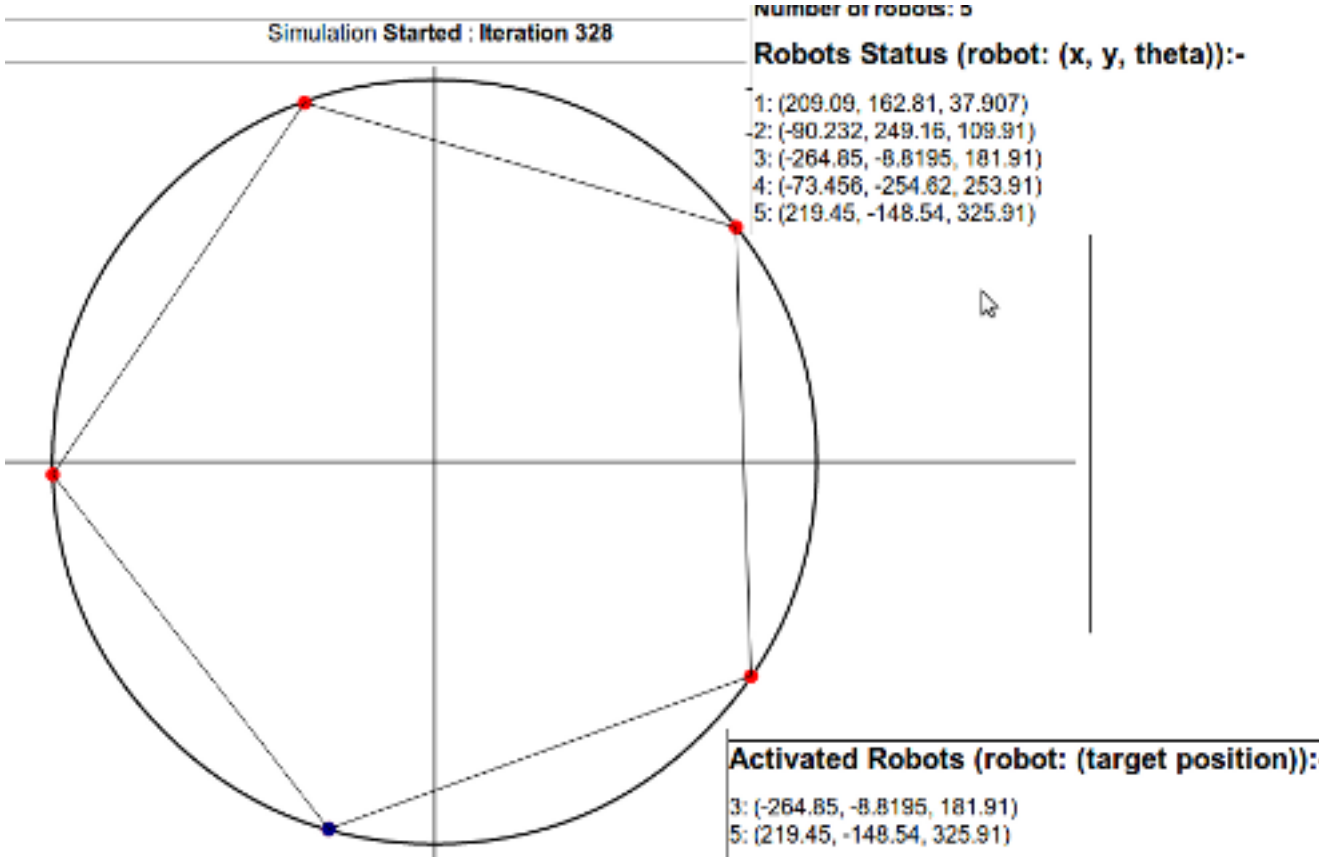

Figure 3.5: F
inal states of all the robots. The faulty robots is shown in blue color (although it is anonymous in theory). As shown, after a finite number of iterations, they form a regular polygon.

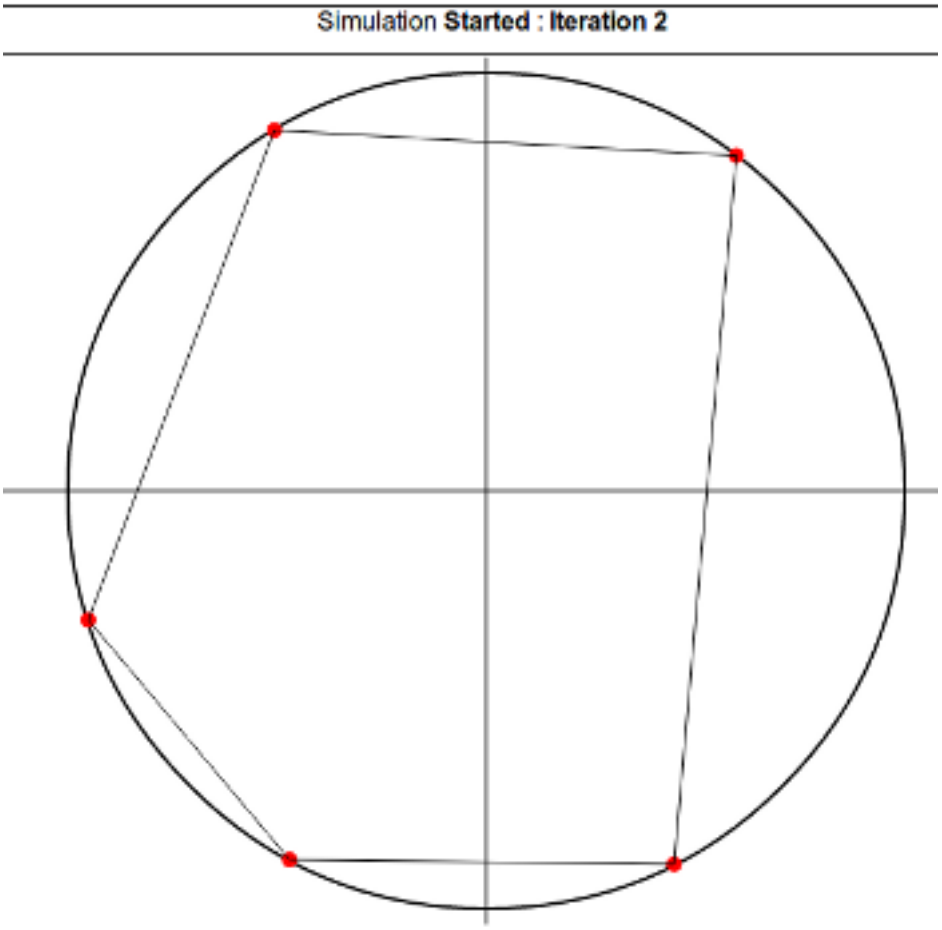

Figure 3.6: I
nitial states of all the robots. Algorithm 4.

Proof. "Algorithm 4" deterministically solves the problem of uniform transformation. No rigorous prrof hasn been worked out. But, this algorithm was simulated, and as in the diagrams shown, the movement of the robots stopped after a finite number of cycles forming an uniform circle/regular polygon.

□

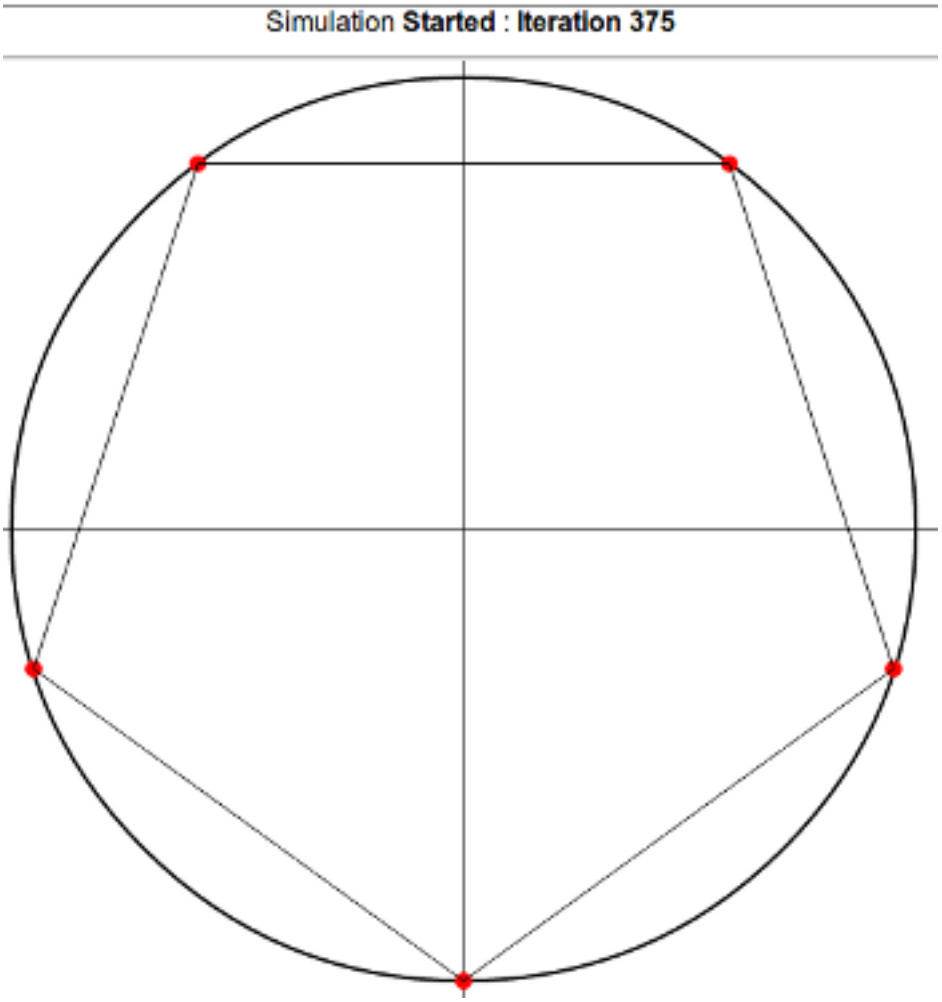

Figure 3.7: F
inal states of all the robots. Algorithm 4. They form a regular polygon.

3.4 Combining the two parts

Now the sub-parts constructed can be joined to make one solution to a bigger problem which is Uniform Circle Formation. This is shown in "Algorithm 5". When any robot is inside the circle, we run the algorithm for circle formation. If all the robots are on the boundary of the smallest enclosed circle, it will run one of the algorithms from 2, 3, 4 depending on the situation.

Algorithm 5 uniform circle formation (Combining the parts)

function $\mathcal{A}_{uniform-circle}$(P, p_i)

1: **if** p_i is in the interior of SEC(P) **then**
   run Algorithm 1
   $\mathcal{A}_{circle}$(P, p_i)
2: **else**
   {All robots are on SEC(P)}
   Algorithm 2, 3, 4 depending on the situation
   $\mathcal{A}_{uniform}$(P, p_i)
3: **end if**

4

Conclusion

Results from references were collected about the possibilities and impossibilities of the solutions for sub problems in pattern formation by autonomous, anonymous, oblivious/non-oblivious and homogeneous robots. In order to provide a deterministic and finite solution to the Uniform Circle Formation problem, some strong assumptions were made in the model.

The result of the project can be summarised as follows:

- Uniform Circle Formation problem can be sub-divided into two parts(Circle Formation and Uniform Transformation problems) and they can be solved separately to get the solution to the bigger problem.

- Circle Formation Problem can be solved for n $\geq$ 3 robots in a finite number of cycles with oblivious robots.

- Currently, no deterministic algorithm for oblivious, anonymous mobile robots exist to solve the Uniform transformation problem which terminates in a *finite number of cycles*.

- Taking a stronger assumption on the model makes the Uniform transformation problem solvable by oblivious robots in a finite number of cycles.

- ***First assumption***: Exactly one robot on the circle is "immobile". Now, the Uniform transformation problem is solvable by oblivious robots in a finite number of cycles by the algorithm *Algorithm 3* for $n \geq 2$.

- ***Second assumption***: Robots can only move in one direction (anticlockwise). Again, the Uniform transformation problem is solvable by oblivious robots in a finite number of cycles by the algorithm *Algorithm 3* for $n \geq 3$.

Bibliography

[1] F. Aurenhammer. Voronoi diagramsa survey of a fundamental geometric data structure. *ACM Comput. Surv*, 23(3):345–405, 1991.

[2] X. A. Debest. Remark about self-stabilizing systems. *Commun. ACM*, 38(2):115–117, 1995.

[3] Xavier Défago and Akihiko Konagaya. 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*, POMC '02, pages 97–104, New York, NY, USA, 2002. ACM.

[4] M. B. Dias and A. Stentz. A free market architecture for distributed control of a multirobot system. *Proc. 6th Intl Conf. on Intelligent Autonomous Systems*, pages 115–122, 2000.

[5] P. Flocchini, G. Prencipe, N. Santoro, and P. Widmayer. Distributed coordination of a set of autonomous mobile robots. *Intelligent Vehicles Symposium*, pages 480–485, 2000.

[6] I. Suzuki H. Ando, Y. Oasa and M. Yamashita. Distributed memoryless point convergence algorithm for mobile robots with limited visibility. *EEE Trans. on Robotics and Automation*, 15(5):818–828, 1999.

[7] J.-B. Billeter M. J. B. Krieger and L. Keller. Ant-like task allocation and recruitment in cooperative robots. *Nature*, 406(6799):992–995, 2000.

[8] N. Santoro P. Flocchini, G. Prencipe and P. Widmayer. Hard tasks for weak robots: The role of common knowledge in pattern formation by autonomous mobile robots. *Proc. 10th Intl Symp. on Algorithms and Computation (ISAAC99)*, 1741 of LNCS:93–102, 1999.

[9] N. Santoro P. Flocchini, G. Prencipe and P. Widmayer. Gathering of asynchronous oblivious robots with limited visibility. *Proc. 18th Annual Symp. on Theoretical Aspects of Computer Science (STACS 2001)*, 2010 of LNCS:247–258, 2001.

[10] Giuseppe Prencipe. Impossibility of gathering by a set of autonomous mobile robots. *Theoretical Computer Science*, 384:222–231, 2007.

[11] S. Skyum. A simple algorithm for computing the smallest enclosing circle. *Information Processing Letters*, 37(3):121–125, 1991.

[12] K. Sugihara and I. Suzuki. Distributed algorithms for formation of geometric patterns with many mobile robots. *Journal on Robotics Systems*, 3(13):127–139, 1996.

[13] I. Suzuki and M. Yamashita. Distributed anonymous mobile robots: Formation of geometric patterns. *SIAM Journal of Computing*, 28(4):1347–1363, 1999.

[14] A. S. Fukunaga Y. Uny Cao and A. B. Kahng. Cooperative mobile robotics: Antecedents and directions. *Autonomous Robots*, (4):1–23, 1997.