Circle formation algorithm for autonomous agents with local sensing

Andrew Mario Michael

Follow this and additional works at: http://scholarworks.rit.edu/theses

Recommended Citation
CIRCLE FORMATION ALGORITHM FOR AUTONOMOUS AGENTS WITH LOCAL SENSING

By

ANDREW MARIO MICHAEL

Thesis submitted to the Faculty of Rochester Institute of Technology in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

IN

ELECTRICAL ENGINEERING

Approved by

Thesis Advisor
Dr. Attimoottil Mathew

Thesis Committee
Dr. Ferat Sahin

Thesis Committee
Dr. Wayne Walter

Department Head
Dr. Robert J. Bowman

DEPARTMENT OF ELECTRICAL ENGINEERING, COLLEGE OF ENGINEERING
ROCHESTER INSTITUTE OF TECHNOLOGY, ROCHESTER, NEW YORK

July 2004
Thesis/Dissertation Author Permission Statement

Title of thesis or dissertation: Circle Formation Algorithm for Autonomous Agents with Local Sensing.

Name of author: Andrew Mario Michael
Degree: Master of Science
Program: Electrical Engineering
College: College of Engineering.

I understand that I must submit a print copy of my thesis or dissertation to the RIT Archives, per current RIT guidelines for the completion of my degree. I hereby grant to the Rochester Institute of Technology and its agents the non-exclusive license to archive and make accessible my thesis or dissertation in whole or in part in all forms of media in perpetuity. I retain all other ownership rights to the copyright of the thesis or dissertation. I also retain the right to use in future works (such as articles or books) all or part of this thesis or dissertation.

Print Reproduction Permission Granted:

I, Andrew M. Michael, hereby grant permission to the Rochester Institute Technology to reproduce my print thesis or dissertation in whole or in part. Any reproduction will not be for commercial use or profit.

Signature of Author: ___________________________ Date: 07/23/04

Print Reproduction Permission Denied:

I, Andrew M. Michael, hereby deny permission to the RIT Library of the Rochester Institute of Technology to reproduce my print thesis or dissertation in whole or in part.

Signature of Author: ___________________________ Date: 07/23/04
Acknowledgments

During my study at Rochester Institute of Technology (RIT), New York, there have been many individuals who have helped me in numerous ways. Dr. Attimoottil Mathew, my research advisor, stands out among them. He has tested me with his sharp questions that have made me think a great deal and helped me develop my thesis. Dr. Mathew has seldom declined to talk to me when I walk into his office without prior appointment! I thank him for giving me so much of time in spite of his busy schedule. Working with Dr. Mathew has been very challenging. It made me realize my short comings and helped me to develop as a researcher. Thank you Dr. Mathew.

I wish to thank the head, faculty and staff of the department of Electrical Engineering at RIT. The department funded most part of my research and helped me cover my expenses at RIT. I thank the department for providing me work space and equipment for my research. The faculty at RIT has been helpful in guiding me and sharing their expertise. Special thanks to my thesis committee members Dr. Ferat Sahin and Dr. Wayne Walter. Dr. Jayanti Venkataraman met me a couple of times to talk about the practical applicability of the algorithm. Thanks to her. I also wish to thank Mr. James Stefano and Ms. Patti Vicari for their help in many ways.

My heart felt gratitude goes out to my family and fiancée for being with me in spirit even though we are oceans apart. They have been my source of inspiration and encouragement at times of need. Their prayers have helped me survive many difficulties faced as an international student. Thank you so much once again to my dearest amma, appa and my fiancée Geethi.
CIRCLE FORMATION ALGORITHM FOR AUTONOMOUS AGENTS
WITH LOCAL SENSING

By

ANDREW MARIO MICHAEL

Master of Science in Electrical Engineering

Abstract

Research on cooperative robotics has increased radically over the past decade due to its simplicity and applicability in a variety of fields. Shape formation plays an important role in such cooperative behavior. Our work deals with the formation of a circle by a group of mobile agents (robots) that initially are randomly spread and randomly oriented in an unmapped terrain. The agents have simple characteristics and limited capabilities. They are autonomous, homogeneous, anonymous, and memory-less. They do not communicate with each other, but are able to measure the inter-agent distances and angels. The agents follow the same distributed algorithm synchronously without any central control. The existing algorithms make it necessary to scan all the agents over the whole terrain. The main advantage of our algorithm is that each agent makes use of local information collected from two neighboring partners. Our algorithm also results in a regularly distributed circle for any form of initial distribution. By changing a parameter in the algorithm, the circle can either be made to grow or shrink uniformly. Applications of this work can be made to a variety of areas such as space missions, military operations, in agriculture and fire fighting.
# Table of Contents

**Acknowledgments** .......................................................................................................................... ii

**Abstract** .......................................................................................................................................... iii

**Table of Contents** ........................................................................................................................... iv

**List of Figures** .................................................................................................................................... vii

**List of Tables** ..................................................................................................................................... x

1. **Introduction** ...................................................................................................................................... 1
   1.1 Related Work ................................................................................................................................. 2
   1.2 Advantages and Disadvantages of past work ............................................................................. 5
   1.3 Our Contribution ............................................................................................................................ 7
   1.4 Thesis Organization ....................................................................................................................... 10

2. **Cooperative Robotics** .................................................................................................................... 13
   2.1 Advantages of Cooperative Robots ............................................................................................ 13
   2.2 Examples of Formation in Nature ............................................................................................... 15
      2.2.1 Schools of Fish ......................................................................................................................... 16
      2.2.2 Flocks of Bird ........................................................................................................................ 16
      2.2.3 Termites and Army Ants ........................................................................................................ 17
   2.3 Importance of Self Organization in Cooperative Robots ............................................................. 17
   2.4 Importance of Shape Formation in Cooperative Robotics ............................................................. 18
Missing Page
3.14.3 Irregular formation....................................................... 63
3.14.4 Inappropriate Direction of Orientation ................................. 67
3.14.5 Very small span of scan ................................................... 70

4. Simulation and Results.................................................................. 74
4.1 Coordinate Axis Change ............................................................ 75
4.2 Flow Chart of the Algorithm ....................................................... 78
4.3 Convergence of the algorithm ..................................................... 80
4.4 Effect of step on Formation ....................................................... 84
4.5 Effect of span of scan on formation ............................................. 90
4.6 Effect of span of scan and step on final radius ............................. 96
4.7 Circle formation for various initial distributions ............................. 97
4.8 Shrinking and growing circle ...................................................... 100
4.9 Circle with a particular radius .................................................... 102
4.10 Algorithm comparison ............................................................ 105

5. Conclusion and Future work .......................................................... 108
5.1 Conclusions ............................................................................. 108
5.2 Future Work ............................................................................. 109

Reference ..................................................................................... 112
Appendix A .................................................................................. 116
Appendix B .................................................................................. 117
List of Figures

Figure 1-1 Initial agent distribution. Circle showing the position of the agent and the lines the direction it is oriented .......................................................... 1

Figure 1-2 A picture from the Sydney Olympics opening ceremony ....................... 7

Figure 2-1 “Spirit of Mars rover”, the robot that explored the Martian surface [38] ..... 15

Figure 3-1 A colony of agents distributed randomly with their directions of orientation and spans of scan ................................................................. 22

Figure 3-2 A regular hexagon (n =6) is divided into 4 (n-2) triangles ...................... 28

Figure 3-3 Partner selection by agent R_1 ......................................................... 29

Figure 3-4 Partners changing at each iteration ..................................................... 31

Figure 3-5 Nearest agents selected as partners ..................................................... 32

Figure 3-6 Path of five agents if the nearest agents are selected as partners .......... 34

Figure 3-7 Path of the agents if the smallest angle agents are selected as partners .... 34

Figure 3-8 Measurement of Inter-Agent angle and distance ............................... 36

Figure 3-9 Two possible positions R_{N1} and R_{N2} with \( \alpha = \theta \) and equidistant from partners ............................................................................................................................... 37

Figure 3-10 Internal and External agents .............................................................. 38

Figure 3-11 A distribution to explain the selection of correct position ................. 39

Figure 3-12 Position of partners in the local coordinate system .......................... 41

Figure 3-13 Three possible ways for partners to be when \( \alpha < \pi \) ....................... 42

Figure 3-14 Three possible ways for partners to be for \( \alpha > \pi \) ......................... 43

Figure 3-15 New position R_{N} of R_{i} when \( \alpha < \pi \), case (a) ............................ 43

Figure 3-16 New position R_{N} when \( \alpha > \pi \), case (d) ......................................... 46
Figure 3-17 New position $R_N$ of $R_i$ when slope of $R_LR_R > 0$ ......................... 48

Figure 3-18 Diagram showing the distance $D$ to travel and the angle $\gamma$ to turn .......... 51

Figure 3-19 Five agents in formation of a regular pentagon showing preferred direction of orientation ................................................................. 54

Figure 3-20 The angle $\varepsilon$ to be adjusted by $R_i$ for (a) $\alpha < \pi$ and (b) $\alpha > \pi$ ............ 56

Figure 3-21 A regular hexagon growing or shrinking depending on $\theta_d$ ......................... 58

Figure 3-22 Relation between inter-agent distance (ird) and circle radius (RAD) ............... 59

Figure 3-23 Direction of orientation for $R_i$ after moving a step .................................... 61

Figure 3-24 Two agents selecting the same agents as their partner .................................... 62

Figure 3-25 Four agents in a rectangle moving to new locations on another rectangle ... 64

Figure 3-26 Agents getting too close to each other .......................................................... 65

Figure 3-27 Simulation results for irregular shape formation ........................................... 66

Figure 3-28 Simulation results for change of internal external agents ................................. 66

Figure 3-29 A agent selecting inappropriate partner due to its direction of orientation ... 68

Figure 3-30 $R_7$ getting stagnant at the wrong position in the formation ......................... 69

Figure 1-31 Two agents trapped in the middle. Their path is also shown .......................... 70

Figure 3-32 Formation of two shrinking circle due to small span of scan ....................... 72

Figure 3-1 Change of axis with respect to the position and direction of $R_i$ ....................... 75

Figure 4-2 Distribution of the initial colony .................................................................... 80

Figure 4-3 Convergence graphs, $n = 50$, with unlimited step and range. Error for

iterations from (a) 1-100 (b) 101-200 (c) Inter-agent angle (d) final formation ...... 81

Figure 4-4 (a) Error vs. # of iterations. $n = 49$, with unlimited step and range. (b) Final

formation after 300 iterations ....................................................................................... 83
Figure 4-5 Effect of step size on formation. The right column shows the formation after 50 iterations and the left column shows the error graph. .................................................. 87

Figure 4-6 The error graph and the final formation for an optimal step ...................... 88

Figure 4-7 Number of agents that become external against number of steps ............. 89

Figure 4-8 The initial colony shown in figure 4-2 simulated with step =0.5 for first 10 iterations and 0.01 thereafter .......................................................................................................................... 90

Figure 4-9 Effect of span of scan on formation for the colony shown in figure 4-2. ...... 93

Figure 4-10 Maximum, mean and minimum distances from the centroid for the case when span of scan is 0.2 and 0.3 ................................................................................................................. 94

Figure 4-11 Final radius shown as a surface plot for different step and span of scan ..... 96

Figure 4-12 Final formation (right) for various initial distributions (left) ............... 100

Figure 4-13 Radius of the growing circle for different values of δθ ............................ 101

Figure 4-14 Radius of the shrinking circle for different values of δθ ......................... 102

Figure 4-15 Mean distance from the centroid and final formation for different required radii ......................................................................................................................................... 105

Figure 4-16 Distance from the centroid and the final formation in Suzuki’s algorithm. 105

Figure 4-17 Distance from the centroid and the final formation in the algorithm developed in this thesis .................................................................................................................. 106

Figure 4-17 Simulation result of Suzuki et al.’s algorithm when the initial distribution is a line ....................................................................................................................................... 107
List of Tables

Table 3-1 Theoretical and simulation values for the required radius and the corresponding inter-agent distance ................................................................. 103
1. Introduction

A group of robots, autonomous, simple and take steps as a result of changes in the environment to achieve a common goal with very little human intervention or supervision can be used in applications such as space missions, agriculture, fire fighting, search operations, landmine de-mining and many others [1-3]. Over the years engineers and roboticists have researched extensively on artificial intelligence of an individual robot. In the recent past the attention has been diverted to cooperative robotics, inspired by social insect colonies or swarms.

If $n$ number of autonomous robots are scattered and oriented randomly in an unmapped terrain (figure 1-1) ordering or arranging them becomes an issue of importance if these robots are to achieve a common task. Since the terrain is unmapped and there is no global coordinate system each robot will be basically lost in the wilderness. The distribution shown in the figure needs to organize and form a shape before the robots start to achieve a task.

![Initial agent distribution. Circle showing the position of the agent and the lines the direction it is oriented](image)

Figure 1-1 Initial agent distribution. Circle showing the position of the agent and the lines the direction it is oriented
Research on forming shapes with cooperative robotics has been of interest in the recent past. This serves as a starting point for cooperative task achievement for autonomous robots randomly spread in an unknown terrain.

1.1 Related Work

The study of swarms and their complex task of achieving a goal through simple steps has been done for a long period of time. Inspired by the swarms, over the past decade research on cooperative robotics has radically increased due to its application in a variety of fields.

The pioneers of shape formations in cooperative robotics were Suzuki et al. [1-5]. In their research they have developed algorithms for circle, line and simple polygon formations. The algorithms are developed with the assumption that robots have \textbf{global sensing} capacity. The robots need to know the positions of all the other robots in the colony. Their circle formation algorithm is as follows. Each robot $R$ monitors the farthest robot $R'$ and the nearest robot $R^*$. If $D$ is the diameter of the circle to be formed, $\delta$ a small positive constant and $d$ the distance between $R$ and $R^*$, robot $R$ then moves as follows.

If $d > D$, then $R$ moves towards $R^*$

If $d < D - \delta$, then $R$ moves away from $R^*$

If $D - \delta \leq d \leq D$, then $R$ moves away from $R^*$

The algorithm in their work is simple but the formation is not precise. The circle formed is an approximation of a circle and the robots are not distributed evenly on the
Introduction

Further, for certain initial symmetric distributions their circle algorithm does not make a circle but a shape called Reuleaux’s triangle [1].

Suzuki et al. in [3] give a formal discussion of the limitations of their algorithms under certain assumptions. In their later work they have modified their algorithm to represent a robot as a disc and have also included collision avoidance strategies [2].

Along the lines of Suzuki et al.’s work, extensive work has been done by Prencipe et al. [6-9]. In their research the robots are considered to be asynchronous and formations resulting in finite time. There is no common notion of time and a robot observes the environment at unpredictable time instants. The formation problem of cooperating and coordinating a group of independent robots is analyzed from a computational point of view. The main problems analyzed in their work are: arbitrary pattern formation, gathering and flocking. In the arbitrary pattern formation the robots have the capacity of global sensing, have a prior knowledge and agree upon a unit distance and a common direction. The robots are also given the coordinates of the pattern to be formed with respect to their local coordinate system. The problem is mathematically analyzed. In the gathering problem, the robots are supposed to gather at a certain point. Prior knowledge of a common direction by the use of a compass is exploited in the development of this algorithm. The need of a compass arises only if the robots have local sensing, but if the robots have global sensing the compass is done away with.

The flaws and imperfections in the work of Suzuki et al. has been modified in [10] As stated earlier in [1] the circle formation algorithm does not work well for certain initial distributions and results in imperfect formations. In [10] a different approach is employed but the characteristics of the robots remain the same as in [1].
The algorithm is as follows.

Robot $R$ determines the furthest robot $R_f$, the closest robot $R_{c1}$ and the second closest robot $R_{c2}$. Computes the coordinates of the centroid $p_m$, of $R_f, R_{c1}$ and $R_{c2}$. Moves to point $p_r$ which is $r$ distance away from $p_m$ on the line that passes through current position and $p_m$, where $r$ is the desired radius of the circle to be formed.

In this work the robots are also assumed to be having global sensing capacity. The algorithm does not explain how each robot will move from its present location to reach the centroid point.

In [11] the authors try to make a circle from a randomly distributed colony of robots. The algorithm is based on Voronoi cells and Smallest Enclosing Circle (SEC). The smallest enclosing circle, as the name suggests, is the circle with the smallest radius enclosing all the robots. Each robot determines the boundary of the smallest enclosing circle and moves to that point. Here too the robots need to have sensing of the whole terrain. The authors state “Although we believe that the algorithm could actually be used in practice, there are several important issues that must be addressed”. Each robot has to scan the positions of the rest of the colony and needs to make intensive computation to determine the SEC. Also, how the SEC can be practically determined in real time is not stated.

In [12] the circle formation problem is rather trivialized by the use of a beacon around which the circle is to be formed. If the robots have a prior knowledge of the radius of the circle and by measuring the distance between themselves and the beacon and then by moving accordingly circle can easily be formed. If there is a central beacon the terrain
is no more unknown. The beacon itself serves as the origin of a coordinate system around which a circle of desired radius is formed.

From our literature survey the above were the algorithms we found on circle formation in cooperative robotics. There are other formation algorithms but do not directly relate to circle formation. We also list those works below.

Balch et al. [12-14], deal with the problem of keeping a formation and avoiding obstacles while in motion rather than shape formation of a randomly distributed colony. Keeping the shape of a line, column, diamond and wedge for a group of robots in motion is addressed. Moreover the robots are not considered to be homogeneous since each robot has an identification number. The global positioning system (GPS) is used to transmit the coordinates of the robot positions making the system not simple. The use of world coordinates with the help of GPS makes the terrain totally mapped.

Matarić et al. [16, 17] have also worked on maintaining formations for a small group of robots. In their work the robots have local sensing but through simple communication they have access to the global goal. The algorithm is developed by each robot keeping a designated friend at a particular distance and angle by using a panning camera. Each robot has a unique ID and a protocol for communication purposes.

1.2 Advantages and Disadvantages of past work

Circle formation algorithms stated above have their own different advantages and disadvantages.

The advantages of the algorithms presented in [1-5] and [10] are that they are simple and very little mathematical calculation is needed by each robot.
The main disadvantage of all the above algorithms is that each robot needs to scan the whole terrain for the functioning of the algorithm. By global scanning each robot has to scan the positions of all the robots in the colony, store the information collected and make decisions depending on the stored information. Global scanning is disadvantages for the following reasons.

1. The battery power of the robot will die off faster, since scanning the whole terrain needs more energy. This will reduce the time period the robot can function.

2. To attain global sensing it becomes necessary to have more powerful transmitters and sensors. This makes the robots sophisticated and not simple or weak.

3. If the number of robots \( n \) is large the robots need to have a memory array to store the positions of the robots. Then the robots have to sort them to select the closest or the farthest robot.

4. When \( n \) is large scanning the whole terrain is time consuming resulting in a delay at each iterative step. This reduces the efficiency of the colony as whole.

Another draw back of past work is they do not explain how each agent will move at each iteration. The direction and the distance a robot will make at each iteration are not explained. In simulation it is easy to find the location of the new position of the robot using the global coordinate system. In a practical situation of an unknown terrain there is no such global coordinate system. A robot has to know how it will reach a new position at each iteration. This decision making on how each robot has to travel is not clearly explained.
The other disadvantage is that the above algorithms do not always result in a circle. The outcome is dependent on the initial distribution. The circle formed is also not evenly distributed.

1.3 Our Contribution

Imagine the situation during Olympic opening ceremonies (figure 1-2) and other sport displays the performers make various forms and shapes in a large unmarked field. Another situation is when a group of large number of people is asked to make a circle. In both these situations the members are able to form shapes with no central control in an unmarked terrain. Essentially the members observe the positions of the neighboring members and position themselves iteratively till a reasonably acceptable shape is formed.

Figure 1-2 A picture from the Sydney Olympics opening ceremony
In this thesis we try to make a circle in an unknown terrain with autonomous mobile agents with only local sensing capability. An agent observes the positions of the neighboring agents, one to its left and one to its right, computes the next position it has to be and moves there. The same iterative process of observe, calculate and move is simultaneously executed by all the agents in the colony till the colony forms a circle. In this thesis we present a circle formation algorithm for very simple agents.

- The agents are autonomous. Human intervention is not needed after initializing the circle formation algorithm.
- The agents in the colony are all identical. There is no leader or even a hierarchical structure.
- The agents do not communicate with each other. They do not have distinctive ID numbers.
- The agents do not have knowledge of a global coordinate system since they are spread in an unknown terrain. There are no landmarks or beacons around which the circle has to be formed.
- The agents do not use the world coordinate system with the help of GPS.
- The agents do not have a sense of common direction or use a compass.
- The main advantage of our circle formation algorithm is that the agents have limited scanning power.
- Even with local scanning the agents do not scan all the agents within its span of scan. An agent selects and collects information only from the two
neighboring partners. This makes it unnecessary to store information of all the agents in the colony.

- The agents are memory less. They do not store information about their past path points or about their past partners.

A novel idea of using the inter-agent angle information is exploited for the formation algorithm. The inter-agent angle and distance information is used with properties of regular geometric figures in the formation.

The agents move in iteration synchronously. Their functions at a particular iteration are: Move, Scan, Measure and Compute. In our algorithm we have used a higher computing power for the agents than other algorithms. An agent can perform algebraic and trigonometric calculations to obtain the angle and distance it has to turn and move respectively at each iteration.

We give a formal explanation mathematically and with the help of diagrams on how each agent has to move at every iteration. We mathematically derive equations for the distance and angle an agent has to take to reach the new position. The higher computing power is used to obtain these distances by substituting in the equations we derived.

Our simulation does not use the global coordinates. Each agent scans with respect to its local coordinate system with its position as its origin and direction of orientation as the positive x-axis. Simulations are made to represent how in a real situation an agent will observe the colony, select the partners, and calculate the distance and angle it has to take to move to the new position.
In [6] the author quotes, ‘Little is known about the solvability of other geometrical problems like spreading and exploration...’. In our work we present extensions of the circle algorithm, such as flocking or gathering at one point or foraging in the form of a growing circle. These are the extensions of the same circle algorithm but are achievable by altering a single parameter.

Finally our algorithm works well for all type of initial distributions resulting in evenly distributed circles.

1.4 Thesis Organization

Chapter 1

In this chapter we briefly explained the area of shape formation in cooperative robotics and its importance. Past work in shape formation and their advantages and disadvantages were noted. Finally we gave details of the contribution we have made in our circle formation algorithm.

Chapter 2

In chapter 2 we give an understanding of cooperative robots. Examples of cooperative robots in nature are stated. Advantages of cooperative robotics are given. The importance of self organization and shape formation in cooperative robotics is brought forward.
Chapter 3

Chapter 3 is dedicated for the development of the circle algorithm. We start with a description of the colony and the characteristics and capabilities of its robots. A mathematical nomenclature of the colony is given. The mathematical idea behind the circle algorithm is explained. Then we develop the understanding and some definitions needed for the algorithm. How each robot calculates the positions of other robots and the coordinates of its new position with respect to its local coordinate system is worked out. A mathematical derivation of the distance, the angle and the direction of orientation adjustment a robot has to take at each iteration is given. We then explain how this algorithm can be altered to make a growing or shrinking circle. Finally some draw backs of the algorithm and possible solutions on how to overcome them are addressed.

Chapter 4

In this chapter we obtain simulation results of the algorithm we developed. In simulation each robot uses its local coordinate system instead of making use of the global coordinates. The required coordinate axis change is explained. A flow chart of the algorithm is then given. The convergence of the algorithm to a regularly distributed circle is verified by plotting the inter-robot distances. The effect of step size and span of scan on the final circle formation is analyzed. Then we simulate the results for growing and shrinking circle. Forming a circle with a particular radius is also simulated. Finally we compare the results of the existing algorithm and the algorithm developed in this thesis.
Chapter 5

Chapter five is for conclusion and future work. Issues involved in practical implication will be stated.

Reference

Appendix
2. Cooperative Robotics

Swarm based robotics relies on a group of simple robots that are able to perform tasks without explicit representation of the environment and of the other robots [4]. In this approach extensive initial planning of achieving a task is minimized by allowing the robots to react to changes in the environment. Social insects are autonomous and communicate with each other through the environment. Indirect communication among insects through modifications of the environment was coined *stigmergy* by Grasse, an entomologist [4]. Stigmergy theory states that steps taken by a colony is regulated by the effect of the previous steps. It also states how activity can be regulated using only local perception and indirect communication through the environment for coordinating distributed behavior to achieve a global task.

Social insects are very simple and ineffective as an individual entity but as a colony they cooperate to attain global needs. In autonomous cooperative robotics the same idea is imitated by having several simple homogeneous robots that work together to achieve a user defined goal. Unlike the artificial intelligence of a single robot that is expensive, complicated, tailored to specific problems, cooperative robotics uses a different approach of using teams of simple, interacting robots to perform a wide range of tasks [5].

2.1 Advantages of Cooperative Robots

The fact that researchers are yet to invent a highly autonomous robot capable of functioning in a changing environment has lead them to propose the organization of several simpler robots into collections of task achieving populations [16]. Cooperative
Cooperative robotics has several advantages over individual robots with artificial intelligence. If a specific task is to be performed by an individual robot, it increases the robot complexity making it difficult to design and fabricate. On the contrary cooperative robots have elementary features making them easier to design and manufacture.

Due to the simplicity and the increased number of robots made, the cost of manufacturing cooperative robots is highly cost effective than that of an individual robot.

A group of cooperative robots is comparatively more fault tolerant than an individual robot. Since there are a number of robots, if one of them malfunctions the rest may carry on with the task. In our model we assume all the agents to be homogeneous and with no hierarchy. This makes the colony further more fault tolerant. If there is a hierarchy and there is a malfunction at the top of the hierarchy several or all the robots may be affected.

In cooperative robots the algorithm by which they are driven plays an important role in their functioning. It is comparatively easier to change the behavior of the colony by changing the algorithm than to change the performance of a single robot designed to meet specific goals. This makes cooperative robots more flexible.

When a group of robots are engaged in achieving the same task, efficiency of task completion increases. In applications such as space mission, search operation, lawn moving or harvesting, more area could be covered by the colony than the area covered by an individual robot.

In the recent space missions by NASA to explore the Martian surface individual robots are being employed. One such robot is shown in figure 2-1. In these single robot missions there could be numerable problems and the possibilities of a mission failure is
fairly high. By dropping a swarm of cooperative robots working together to explore the Martian surface, a high degree of fault tolerance may be achieved. Moreover the efficiency of exploration could also be significantly increased.

![Spirit Mars Exploration Rover](image)

**Figure 2-1 “Spirit of Mars rover”, the robot that explored the Martian surface**

For robots to function as a group in such autonomous tasks in unknown terrains it becomes a necessity that they are capable of “Self – Organizing” or work in formations. Social insects and animals that inspired cooperative robotics work as a group in formations or in an organized manner [16, 17, and 18].

### 2.2 Examples of Formation in Nature

Social insects are distributed systems in which colony level behavior emerges out of interactions among individuals [19]. Intricate phenomena such as foraging, nest building and path formation are achieved through the cooperation of insects with limited
capabilities. There are other social animals in nature which thrive as a group but as an individual entity may not be able to survive. In many such colonies there is no hierarchy or central control. Local information by neighbor observation is used to obtain global goals. We examine few such natural systems.

2.2.1 Schools of Fish

Partridge examines how schools of fish move in certain formations to increase the effectiveness of the school [20]. For example Tuna schools form a parabolic shape with concave side forward and swim parallel to its axis. A prey reacting to the curved school will be driven to the focus of the parabola which makes the capture easier. A school of fish moving in a certain formation has a higher chance of detecting a predator than a fish swimming individually. On the contrary predators also move in formations to increase the search area. This helps to share the food found by a particular member [21]. This idea can be applied in search robots for planetary and military applications.

Formations were maintained in these schools by individual fishes maintaining a particular angle and distance with the neighboring fishes. This idea is utilized in the algorithm of this thesis.

2.2.2 Flocks of Bird

Birds also fly in formations to increase the scanning area like schools of fish. Air force fighter pilots use this technique to direct their visual and radar search depending on their position in a formation [22]. Three mechanisms: collision avoidance, velocity matching with nearby flock-mates and flock centering in attempt to stay close to nearby flock mates are utilized [23].
2.2.3 Termites and Army Ants

Ants and termites are some of the most organized social insects. Construction of nest structures, finding the shortest path between two points, foraging efficiently as a swarm, bringing back the food to the nest are some examples of self organization [24]. They use minimal communication between the members of the colony through trails of pheromone. In these insects formation is in the form of established trails.

2.3 Importance of Self Organization in Cooperative Robots

When a group of autonomous robots are dropped in an unknown terrain, the group has to organize itself before it proceeds with task achieving. A self-organizing system is defined by Farley et al. [25] as a system that changes its basic structure as a function of its experience and environment. In self organizing system a change in the environment may influence the same system to generate a different task, without any change in the behavioral characteristics. Any small differences in individual behavior can influence the collective behavior of the system.

Self-Organizing autonomous robots have a few more advantages than cooperative robots with central control. In a large group of robots communication overhead is prohibitively high to collect all relevant information to a central location. It is also computationally infeasible for a central control to generate a schedule for the entire set of robots in real time [1]. Hence the need of autonomous robots with distributed computing arises.
In military applications, the whole army of robots is destabilized if the central control or the leader is destroyed. On the contrary if the robots are autonomous and homogeneous the mission continues even if a few robots are destroyed.

There are a few disadvantages in autonomous cooperative robotics. Due to lack of coordination there could be stagnation [16]. A group of robots could find themselves in a dead lock detrimental to the global task. The other disadvantage is that the miniaturization of robots severely limits their sensing and computing powers.

2.4 Importance of Shape Formation in Cooperative Robotics

1. If the agents are randomly spread and randomly oriented each agent will go about doing a task without the coordination of the rest of the colony. The behavior would be haphazard, chaotic and will not be directed towards a particular goal achievement.

2. This problem is important, because it provides a way for the agents to agree on a common origin and a common unit distance, for instance by forming a circle [1]. A flock of agents can converge to a point and use that point as the origin as a starting point to achieve various tasks.

3. Formations can be effectively utilized in exploring an unknown terrain. If the exploration is done in an unorganized manner it would be less efficient and inconclusive.

4. Formations help to increase the scanning range of a group of agents. In military applications where sensor assets are limited, formations help to cover a wider
area if individual members concentrate across a portion of the environment while their partners cover the rest [13].

5. Can be used in military operations: to form a barricade in the shape of a circle or surround an area and converge to capture.
3. Algorithm Development

This chapter is dedicated to the development of the circle formation algorithm. The chapter begins with an introduction of the colony and a description of the characteristics and capabilities of the agents. The colony and the way it functions are defined by a mathematical nomenclature.

The basic mathematical idea behind the algorithm is explained. Then we proceed to show how each agent will move to a new position using this mathematical idea so that the colony as a whole will perform the circle formation algorithm.

Recall that we are not using a global coordinate system. This makes it compulsory to scan the terrain with respect to a local coordinate system. The current position of an agent is selected as the origin and the current direction of orientation as the direction of positive x-axis of the local coordinate system.

An agent selects two partners, measures the inter-agent distances and the angles and moves to a new position based on these information. For this an agent has to first select two partners. Partner selection and why an agent selects partners in such a manner is described in detail. The two parameters to be measured from the partners: the inter-agent distance and the inter-agent angle are defined.

For the agent to move to a new position it has to compute the distance \((D)\) it has to travel from its current position and the angle \((\gamma)\) it has to turn from its current direction of orientation. We mathematically derive formulae for these two physical quantities in terms of the known parameters. After the agent moves to its new position the direction of orientation of the agent has to be adjusted for appropriate partner selection at next scanning. An explanation of why the direction of orientation has to be adjusted and
how it is altered is explained. Finally the chapter deals with the short comings of the
circle algorithm and a brief explanation on how to over come them are given.

3.1 The Colony of Agents

The colony has $n$ agents that initially are randomly distributed in an unmapped
and unknown terrain. There is neither a global coordinate system nor landmarks that
could be identified by an agent. The agents in the colony are all identical in physical and
functional terms. They all start functioning synchronously in terms of scanning,
measuring, computing and moving.

Initially the agents are randomly spread and randomly oriented. The direction of
orientation of an agent is the direction from which it starts its scanning to its left and
right. The agents have a limited span of scan within which they can detect the positions
of other agents. By moving together synchronously in each iteration cycle, the final goal
is to form a regular polygon that is on the circumference of a circle.

Figure 3-1 shows the positions of a colony of agents. The agents, $R_1$, $R_2$..., are
represented by a circular disc. The line with arrow denotes the direction of orientation.
The agents are looking towards various different directions as shown by the lines with
arrow. The Dotted circles are the limits of spans of scan of the agents. Each agent selects
two partners within its span of scan and moves to a new location. This process continues
iteratively until a regular polygon is formed.
3.2 Agent Characteristics and Capabilities

The agents in the colony are considered to be having simple characteristics with weak or limited capabilities. They are,

**Characteristics:** Autonomous, Homogeneous, Anonymous, Memory less, Synchronous, Uncommunicative and Myopic.

**Capabilities:** Scan, Measure, Compute and Move

**Autonomous** – This is one of the main features and advantages of the agents in the colony. The colony is totally independent of any central control after *initialization* by the user. The agents are all initialized at the same time instant. Then the agents follow an algorithm iteratively until a user specified termination.

Figure 3-1 A colony of agents distributed randomly with their directions of orientation and spans of scan
Homogeneous – The agents have the same limited features and are identical in all sense. There are no physical or functional differences. They use the same distributed algorithm to determine their next position at every iteration.

Anonymous – As a result of homogeneity it is impossible to have a hierarchy in the colony or individual identification numbers for each agent. When an agent scans the terrain all the other agents appear the same.

Memory less – The agents do not have any form of memory. They do not remember their past locations in the terrain. They also do not remember their partners in the previous iteration and at each step the rest of the colony appears different.

Uncommunicative – The agents do not communicate with each other. Inter agent communication becomes a difficulty due to bandwidth limitations, especially when the number of agents is large. Communication is through the environment in an implicit manner rather than explicit communication between the agents. The agents only observe the positions of their neighbors with respect to their position and direction of orientation.

Myopic – The agents are limited with only local scanning facility. The agents are able to detect only the agents that fall within its limited span of scan.

Synchronous – The colony has a global clock which is launched at colony initialization. Thereafter all the agents act synchronously with respect to global time and also stop at the same time. The global timing plays a very important role in the iterative procedure of circle formation. The agents have to be stationary to measure inter-agent angle and distance. If the agents are not synchronous and some of them are in motion it is almost impossible to measure these quantities. The necessity of synchronicity could turnout to be a major drawback in practical applications.
The agents have the following limited capabilities.

**Scan** – While the colony of agents is stationary, each agent can scan along a plane for a range of \(2\pi\) radians. It is able to detect other agents in the colony while scanning. An agent scans to its left and right from its current direction of orientation. The instant it detects an agent to its left it stops scanning to the left and does the same to its right.

**Measure** – After detecting the agents to its left and right it is able to measure two physical quantities: *inter-agent distance* and the *inter-agent angle* of the left and right neighboring agents. More of how and why the neighboring agents are selected in such a manner will be explained in section 3.5. The definitions of inter-agent angle and inter-agent distance are given in section 3.6.

**Compute** – Using the distances and the angles measured, the agent is able to compute \(\gamma\) the angle by which it has to turn from its current direction of orientation, the distance \(D\) it has to travel from its current position and \(\varepsilon\) the angle adjustment to be made at the end of movement. The equations for \(D\), \(\gamma\) and \(\varepsilon\) will be derived in sections 3.10 3.12 respectively. The agents are capable of performing basic arithmetic and trigonometric calculations to obtain \(D\), \(\gamma\), and \(\varepsilon\).

**Move** – The agents in the proposed algorithm are able to traverse in any direction in a two dimensional horizontal plane. They first turn an angle \(\gamma\) and then move a user defined maximum distance of *step* or less in a unit time.

The agents in the colony synchronously do the above four steps at each iteration cycle. Then the cycle starts again with scanning then measuring and so on and moves to a new position.
3.3 Definitions and Notation

The notations we have used relate to the definitions given in [3]. The colony has \( n \) agents. Let \( C \) denote the set of agents in the colony and \( R_1, R_2, \ldots, R_n \) be the agents in the colony. \( R_i \) denotes the \( i^{th} \) agent of the colony.

\[
C = \{R_i \mid 1 \leq i \leq n\} \tag{3.1}
\]

The synchronous formation of the colony is timed by the global clock. After user initialization, the agents scan, measure, compute at discrete time instances.

\[
t = kT \tag{3.2}
\]

Here \( k \) is a positive integer, and \( T \) is the time period of one iteration cycle in the algorithm. One iteration cycle consist of four steps of scanning, measuring, computing and moving. Let us denote these time intervals for these steps as \( T_s, T_{ms}, T_c \) and \( T_{mv} \) respectively.

\[
T = T_s + T_{ms} + T_c + T_{mv} \tag{3.3}
\]

The time consumed for each of these steps by all the agents is the same to maintain synchronicity. If synchronicity is lost the implementation of the algorithm becomes difficult. For instance, during the scanning step the whole colony has to be stationary. If some of the agents are in motion it will be difficult to detect and measure the distances and angles of the agents in motion.

Let \( p_i(t) \) be the position of \( R_i \) in the global environment at time instant \( t \). Then we can write \( P(t) \), the set of positions of all the agents as,

\[
P(t) = \{p_i(t) \mid 1 \leq i \leq n\} \tag{3.4}
\]
$P(t)$ is the set of positions of all the agents in $C$ at various discrete time instances. Each agent $R_i$ observes $P(t)$ differently unless the distribution is perfectly symmetric and the agents’ orientations are directed in a uniform manner. If the agents in the colony are myopic or with limited span of scan then $R_i$ cannot observe $P(t)$. It will only be able to observe a subset of $P(t)$. Let us denote $V_i(t)$ to be the set of agent positions observable by $R_i$ at time instant $t$. Then,

$$V_i(t) \subseteq P(t)$$

$$V_i(t) = \{ p_v(t) \mid 1 \leq v \leq n \}$$

Since the agents in the colony are myopic $v$ is not equal to $n$. $v$ is the number of agents detectable by $R_i$ within its span of scan. If agents have global sensing capability then $v = n$. If the span of scan is small or if an agent is lost in the terrain, it cannot sense any other agent, then $v = 0$. $V_i(t)$ is different for all the agents unless in a very special case of a perfectly symmetric distribution with agents’ orientations directed uniformly. A formation in the shape of a regular polygon where the agents’ orientation are directed uniformly and radially inwards or outwards is one such distribution we can think of. The ultimate aim of the circle formation algorithm is to make $V_i(t)$ of all the agents same.

The step an agent needs to move depends totally on the previous configuration of the colony. Therefore, in a global sense $P(t)$ directly relates to $P(t-1)$. We could also define $P(t)$ in an alternative manner. $P(t)$ is also a function of $V_i(t)$. A step each agent takes depends solely on how it observes part of the colony at that time instant. Consequently, $P(t)$ is dependent on $V_1(t), V_2(t), \ldots, V_i(t), \ldots, V_n(t)$. We can say that $P(t)$ is a function of the set,
\[
V(t) = \{ V_i(t) | 1 \leq i \leq n \}
\]

For the circle formation algorithm let us denote this function as \( C \).

\[
P(t) = C_{\{V(t)\}} \tag{3.7}
\]

The function \( C \) for each agent \( R_i \) is calculating the angle \( \gamma \) it has to turn from its current orientation, the distance \( D \) it has to move from its current position and \( \varepsilon \) the angle adjustment needed at the end of each step. \( D \) and \( \gamma \) depends on \( V_i(t) \), particularly the inter-agent distances and the inter-agent angles of the partners selected by \( R_i \). Let us see on what mathematical idea \( D \) and \( \gamma \) are computed.

### 3.4 Mathematical Model

The mathematical idea behind our circle formation algorithm is simple and straightforward. Formation of a circle with \( n \) agents can be considered as forming an \( n \)-sided regular polygon (\( n \)-gon). In other words it can be proved that the vertices of a regular \( n \)-gon lie on the perimeter of a circle. If the number of agents, \( n \), is large then the polygon would appear to be distributed uniformly on the circumference of a circle. In our algorithm we are trying to make a regular polygon which necessarily is on the circumference of a circle.

The main idea behind the algorithm is to maintain a certain angle between the neighboring agents. If the agents are at the vertices of a regular polygon this angle is the same for all agents. For a regular polygon this angle can be easily found using basic geometry.
Figure 3-2 A regular hexagon \((n = 6)\) is divided into 4 \((n-2)\) triangles

Figure 3-2 shows a hexagon divided into four triangles. We can generalize it to a polygon with \(n\) sides to state that the polygon can be divided into \((n-2)\) triangles. The sum of the interior angles of a triangle is \(\pi\) radians. If a polygon with \(n\) sides can be divided into \((n-2)\) triangles then the sum of the interior angles of that polygon can be given by \((n-2)\pi\) radians. As we discussed earlier, since we are interested in forming an \(n\) sided regular polygon, the interior angles would all be the same for such a polygon. If we denote the interior angle of an \(n\) sided regular polygon by \(\theta\), it can be given by,

\[
\theta = \frac{(n-2)\pi}{n}
\]  

(3.8)

An algorithm that progressively makes all the interior vertex angles of the formation to be \(\theta\) and the neighbouring sides equal, in a colony of \(n\) randomly distributed agents will lead to a circle formation. In other words we can say that if all the agents try to make the internal angles between their neighbouring agents to be \(\theta\), then
gradually the formation will become a circle. To achieve this $R_i$ should first find two partners and move to a position where the angle will be $\theta$ between the partners and equidistant from them. In the next section we shall see how $R_i$ selects its partners.

3.5 Partner Selection

An agent has to identify two appropriate partners and move to a new position at every iteration. The new position has to be equidistant from its partners making the interior angle to be $\theta$. To select partners an agent has to scan the terrain within its span of scan. An agent can scan an angle of $2\pi$ radians within its span of scan. Our algorithm makes it unnecessary to scan this whole range. If each agent scans to its left and right from its current direction of orientation and collects information about its partners it is sufficient for the functioning of our algorithm. Here we will explain how the partner agents are selected.

Figure 3-3 Partner selection by agent $R_i$
Figure 3-3 shows a possible distribution of the positions of a few agents. Here agent $R_i$ is the agent of concern. The line with arrow is the direction of $R_i$'s orientation. Agents $R_L$, $R_R$, $R_1$ and $R_4$, are all within $R_i$'s span of scan and $R_2$ and $R_3$ are not. $R_i$ scans to its left and right as indicated in figure 3-3. The first agent it detects while scanning to its left side is its partner to the left and the first agents it detects while scanning to its right is its partner to the right. Let us name these partners as $R_L$ and $R_R$ respectively. Even though agent $R_1$ is closer to $R_i$, it is not selected as a partner. The partner selected to the right of $R_i$ is $R_R$ since $R_R$ makes the smallest angle from $R_i$'s direction of orientation. Similarly $R_L$ is selected as $R_i$'s partner to the left. Agents $R_2$ and $R_3$ make the smallest angle with the direction of orientation of $R_i$, but since they are out of the span of scan, $R_i$ will not be able to detect them.

From the above explanation we could define the partners of $R_i$ as the agents that are within the span of scan of $R_i$ and those that make the smallest angles to $R_i$'s left and right with respect to $R_i$'s current direction of orientation. Our algorithm functions with the information gathered from these partner agents.

An agent is unable to select the same two agents as its partners for the entire formation course. There are three reasons why an agent cannot carry on with the same partners. The first reason is that since the agents are all homogeneous there is no way to physically identify a previous partner. Recalling a partner is further made impossible since the agents are memory less. The second reason is that the agents are uncommunicative and as a result there is no way an agent could send a signal to its partner regarding its current location. The main reason is that the colony is changing at
each iteration. A partner at a particular iteration may not be so in the next. The partner may have selected two other agents as its partners and moved away to another location.

![Figure 3-4 Partners changing at each iteration](image)

Figure 3-4 shows a possible distribution of five agents, $R_1$, $R_2$, ..., $R_5$. Let us see how the partner selection will vary for agents $R_1$ and $R_3$. The partner selection depends solely in the direction of orientation of the agents. The arrowed lines show the directions of orientation for $R_1$ and $R_3$. According to the definition of partners $R_3$ selects $R_2$ and $R_1$, since they make the smallest angles to $R_3$’s left and right respectively. $R_1$ does not select $R_3$ but it selects $R_4$ and $R_5$. The new positions of $R_3$ and $R_1$ are $R_{3N}$ and $R_{1N}$ respectively. In the next iteration partners selected by $R_1$ and $R_3$ will depend on their direction of orientation at that time. We will discuss in section 3.12 why and how the directions of orientation are altered after the end of motion at each iteration. It is intuitive that after the agents have reached the perimeter of the circle they should select the same partners to maintain the circle. In a closed circular path, where all the agents are on the periphery of the circle, if the agents do not select the neighbours on either side as its partners the
formation will collapse. In section 3.12 we will see how, by altering the direction of orientation, we can make sure up to a certain extent, that the agents select the same partners they had selected in the previous iteration.

3.5.1 Why small angle partner selection

The agents in the colony as stated are simple and memory less. By selecting the first agents detected when R_i scans to its left and right makes scanning the whole terrain within its span of scan unnecessary. Suppose we say ‘select the two closest agents as R_i’s partners’, then R_i has to scan the full $2\pi$ radians within its span of scan. Moreover it has to store the distances and the angles of each agent it detected and sort to select the two shortest distances and the respective angles. This procedure of scanning, storing and sorting the data is made unnecessary if the first agents detected on either side are selected as partners. This increases the efficiency and reduces the simplicity of agents.

The main reason for not selecting the closest two agents is because the colony then would converge to a single point instead of forming a circle. Let us try to understand this phenomenon of converging to a point with the help of five agents trying to make a regular pentagon.

![Figure 3-5 Nearest agents selected as partners](image_url)
Figure 3-5 shows a possible distribution of five agents $R_1, R_2,...R_5$. $R_1$ will select $R_2$ and $R_3$ as its partners since they are the closest agents. It would then move to position C, equidistant from $R_2$ and $R_3$ making angel $\theta$. Similarly $R_2$’s partners will be $R_1$ and $R_3$ and it would move to B. $R_3$ will move to A selecting $R_1$ and $R_2$ as partners. $R_4$ will move to C after selecting $R_2$ and $R_3$ as partners. $R_5$ will select $R_1$ and $R_2$ and move to A. In the above explanation let us assume the partners selected are the closest two agents. After the first iteration the five agents would have converged to three points. Here we assume that the step size is large enough for the agents to reach their final destination without stopping after moving a unit step. For the second iteration agents at A will select agents at B and C and move to a point between B and C. Similarly C will move to a point between A and B and B between A and C. As we could see the inter-agent distances are progressively getting smaller and ultimately will converge to a single point.

We simulated the same initial distributions of five agents and observed their path for the two different cases. In the first case the nearest two agents are selected as partners and in the second case partners are selected according to our definition given in 3.5. The agents in the simulation move a unit step after selecting the partners without moving to the final position. Figure 3-6 shows the paths of five agents where the nearest agents are selected as partners and figure 3-7 shows the paths of the same initial distribution but here smallest angle neighbours are selected as partners. The circle mark denotes the initial position of the agents and the asterisk denotes the final position. As we can see from the diagrams, when nearest agents are selected as partners the agents gradually get closer and closer and finally converge to a single point. In the second case where smallest angle agents are selected as partners the agents end up making a regular pentagon.
Figure 3-6 Paths of five agents if the nearest agents are selected as partners

Figure 3-7 Paths of the agents if the smallest angle agents are selected as partners

After the appropriate partners are selected $R_i$ has to measure the inter-agent distances and the inter-agent angle between the partners. In the next section we will define and explain these two parameters.
3.6 Inter-Agent angle and Inter-Agent Distance

Figure 3-8 shows the positions of $R_i$ and its partners. The thick lines with arrows show two possible orientations of $R_i$. Let $O$ be the direction of orientation of $R_i$. After scanning to its left and right starting from its current direction of orientation $R_i$ detects its left partner $R_L$ and right partner $R_R$. The distances $R_L R_i$ and $R_R R_i$ are the inter-agent distances $R_i$ will measure. These distances are denoted by $d_L$ and $d_R$ respectively. Hence $d_L$ and $d_R$ are the distances between $R_i$'s current position and to the left and right partners respectively.

The angles $R_L R_i O$ and $R_R R_i O$ are the angles between $R_i$'s direction of orientation and to the left and right partners respectively. The angle to the left partner is denoted as $\alpha_L$ and to the right partner as $\alpha_R$.

$$\alpha_L = \angle R_L R_i O \quad \text{and} \quad \alpha_R = \angle R_R R_i O$$

Let us denote,

$$\alpha = \alpha_L + \alpha_R \quad (3.9)$$

$\alpha$ is the inter-agent angle made by the two partnering agents with the position of $R_i$ which includes the direction of orientation of $R_i$. It could be less than $\pi$ radians or greater depending on the direction of orientation of $R_i$. The orientation can be in any direction, but it is of importance to know if the direction of orientation is in the convex or the concave region.
Figure 3-8 Measurement of Inter-Agent angle and distance

(a) when $\alpha_L + \alpha_R < \pi$ and (b) when $\alpha_L + \alpha_R > \pi$

Figure 3-8(a) and figure 3-8(b) show three agents at the same locations but differing inter-agent angle $\alpha$ for $R_i$. $\alpha$ is different due to its differing directions of orientation of $R_i$. In figure 3-8(a) the direction of orientation is in the convex region, therefore $\alpha < \pi$. In figure 3-8(b) direction of orientation is in the concave region and $\alpha > \pi$.

The value of $\alpha$, i.e. whether $\alpha > \pi$ or $\alpha < \pi$ is of importance for $R_i$ to move to the right position. In the subsequent section we shall explain the importance of the value of $\alpha$. We shall use the value of $\alpha$ to define internal and external agents.

3.7 Internal and External Agents

The goal of $R_i$ at each iteration is to make the inter-agent angle $\alpha$ equal to $\theta$ and $d_L = d_R = d$. $d$ is not a user defined parameter but depends on the inter-agent distances.
and the inter-agent angle between the partner agents. After R_i detects its partners, there are two positions R_i could move to, so that the inter-agent angle is θ and is equidistant from its partners.

![Diagram of agent positions](image)

**Figure 3-9** Two possible positions R_{N1} and R_{N2} with α = θ and equidistant from partners

In figure 3-9 the old position of R_i is denoted by R_O. R_i detects R_L and R_R as its partners and measures d_L, d_R, α_L, and α_R. The two possible positions R_i could move to are shown in the figure. Let us name these positions as R_{N1} and R_{N2}. These positions are in the perpendicular bisector of R_LR_R and at both these positions, 

\[
R_L R_{N1} = R_R R_{N1} = R_L R_{N2} = R_R R_{N2} = d \quad \text{and}
\]

\[
\angle R_L R_{N1} R_R = \angle R_L R_{N2} R_R = \theta
\]

One possible position is on the same side of R_O with respect to the line joining R_i’s partners R_L and R_R. The other position is on the opposite side of R_O with respect to
this line. To choose which position \( R_i \) should select we make use of the inter-agent angle.

To understand this decision making we need to introduce two new terms: Internal Agent and External Agent.

![Figure 3-10 Internal and External agents](image)

In a randomly distributed colony of agents we could define Internal agent and External agent as follows. Imagine of connecting the agents with lines and making polygons so that all the agents in the colony are inside the polygon. We could form various polygons with different number of sides. Such a polygon with the smallest number of sides is of our interest. The agents at the vertices of the polygon with the smallest number of sides and that encompasses the rest of the colony can be defined as the External agents. The agents that are contained by this polygon are the Internal agents. In figure 3-10 the agents that are shaded inside are the external agents and the ones that are plain are the internal agents.

The objective of the colony of agents would be for all agents to become external. In other words we could say that the internal agents have to move towards the periphery of the encompassing polygon while the external agents position themselves so that the
polygon becomes regular. Let us look into a distribution and try to understand how agents would come to know if they are internal agents or external agents.

Figure 3-11 A distribution to explain the selection of correct position

Figure 3-11 shows an agent distribution of six agents. If these six agents are to make a regular hexagon let us examine how agents $R_1$ and $R_5$ should move so that the formation leads toward a hexagon. Let us assume that the directions of orientations for these agents are as shown by the arrows. Agent $R_1$ will detect $R_2$ and $R_6$ as its partners. For the formation to get closer to a hexagon $R_1$ has to be on the same side of its old position with respect to line $R_2R_6$. In the case of $R_5$ it will select $R_4$ and $R_6$ as its partners. Again for the formation to become a regular hexagon $R_5$ has to cross over line $R_4R_6$ and select the point on the others side of $R_4R_6$.

If we assume that the agents in the colony have unlimited span of scan we could definitely state that if $\alpha$, the inter-agent angle, is greater than $\pi$ radians then $R_i$ is an external agent. We cannot make a definite statement that if $\alpha$ is less than $\pi$ radians then $R_i$ will be an internal agent. It depends on the direction at which $R_i$ is oriented. From
figure 3-11, we can see that $\alpha$ for $R_3$ is less than $\pi$ radians even though it is an external agent.

With limited span of scan the problem is further convoluted. We cannot conclusively state that if $\alpha$ is greater than $\pi$ radians then that agent will be an external agent. Within the agent's span of scan it will be an external agent but not for the whole colony. If $R_i$ scans the full $2\pi$ radians and calculates the largest angle gap between the agents, and if that angle gap is less than $\pi$ radians it will be locally as well as globally internal. Again for the case when the angle gap is greater than $\pi$ radians we can come to the conclusion that it is locally external but may not be definitely true in a global sense.

To avoid this situation up to a certain extent, it is important to alter the direction of orientation after the end of movement at every iteration. In the case where a closed circular loop is formed with all the agents external, if the agents look outwards then the inter-agent angles will be greater than $\pi$ and the agents will need to stay on the same side of its old position. If this continues for a few iterations the formation will converge to a regular polygon. We shall discuss about this further in section 3.9.

As a rule we shall state that $R_i$ will stay on the same side of the line joining its partners with respect to its old position if $\alpha$ is greater than $\pi$ radians and cross over to the other side of the line if $\alpha$ is less than $\pi$ radians.

Up to this point we have come to the understanding of how the partners are selected and to which one of the two possible positions an agent has to move relative to its partners. In order for $R_i$ to measure the inter-agent distances and angles of the selected partners and move to a new position, the coordinates of the agents under its span of scan
should be relative to R_i’s local coordinate system. In the next section we will explain how the coordinates of the partners are determined relative to the R_i’s local coordinate system.

### 3.8 Local Coordinate System

When an agent R_i scans the terrain the measurement of distances and angles are made with respect to its local coordinate system. The local coordinate system of a R_i is defined as follows. The origin of the local coordinate is the current position of R_i and the positive x-axis is the current direction of orientation of R_i. Under this definition the coordinates of the partners can be determined as shown in figure 3-12.

![Figure 3-12 Position of partners in the local coordinate system](image)

The coordinates of the positions of the partners of R_i in the local coordinate system of R_i can be written as,
\[ x_L = d_L \cos(\alpha_L) \]
\[ y_L = d_L \sin(\alpha_L) \]
\[ x_R = d_R \cos(\alpha_R) \]
\[ y_R = -d_R \sin(\alpha_R) \]  

(3.10)

Using equations in 3.10 \( R_i \) can find the coordinates of its new position. We shall use the equations in 3.10 for our derivations in the later sections. There are six possible ways for the partners of \( R_i \) to be located in the four quadrants of \( R_i \)'s local coordinate system. Three for the case where \( \alpha < \pi \) and three for \( \alpha > \pi \). The three possible positions of \( R_i \)'s partners when \( \alpha < \pi \) are shown in figure 3-13.

![Diagram](image)

(a) Partners in the 1st and 4th quadrants  
(b) Partners in the 1st and 3rd quadrants  
(c) Partners in the 2nd and 4th quadrants

**Figure 3-13** Three possible ways for partners to be when \( \alpha < \pi \)

When \( R_i \) scans to the left and right from its current direction of orientation, \( R_L \) and \( R_R \) can be in the quadrants as shown in figure 3-13. In these positions \( \alpha_L + \alpha_R = \alpha < \pi \). Similarly there could be three possible positions for \( \alpha > \pi \). These possibilities are shown in figure 3-14.
The new position of \( R_i \) will vary according to different positions of its partners. In the next section we shall examine the coordinates of the new position of \( R_i \).

### 3.9 Coordinates of the New Position

![Diagram showing the new position of \( R_i \) when \( \alpha < \pi \), case (a)](image)

**Figure 3-15** New position \( R_N \) of \( R_i \) when \( \alpha < \pi \), case (a)
Figure 3-15 shows the old ($R_O$) and new ($R_N$) position of $R_i$ respectively. The partners are denoted by $R_L$ and $R_R$. $M$ is the midpoint of line $R_LR_R$.

\[ R_LR_N = R_RR_N \text{ and} \]
\[ \angle R_LR_NR_R = \theta \]

Since $M$ is the midpoint of $R_LR_R$ and $R_LR_N = R_RR_N$,

\[ R_NR_M \perp R_LR_R \text{ and} \]
\[ \angle R_LR_NR_M = \angle R_RR_NR_M = \frac{\theta}{2} \]

$R_LQ \parallel y$-axis and $R_RQ \parallel x$-axis

$R_NR_P \parallel y$-axis and $MP \parallel x$-axis

For simplicity let us denote,

$R_LR_R = A,$

$R_NR_M = d$

$\tan^{-1} (\text{slope of } R_NR_M) = a$

coordinates of $M = (x_m, y_m)$

From the diagram we can see that,

\[ \angle PMR_R = \frac{\pi}{2} - a \]

\[ \therefore \angle R_LR_RQ = \frac{\pi}{2} - a \quad (\because \ MP \parallel QR_R \text{ - alternate angles}) \]

\[ \therefore \angle R_RR_LQ = a \]
Again from the diagram,

\[ x_n = x_m + d \cos(a) \]  \hspace{1cm} (3.11)

\[ y_n = y_m + d \sin(a) \]  \hspace{1cm} (3.12)

In \( \Delta R_L R_N M \),

\[ \measuredangle R_L R_N R_M = \theta/2, \quad R_L M = \frac{A}{2} \text{ and } R_N M = d \]

\[ \therefore \quad d = \frac{A}{2 \tan\left(\frac{\theta}{2}\right)} \]  \hspace{1cm} (3.13)

In \( \Delta R_L R_R Q \),

\[ \measuredangle R_R R_L Q = a, \quad R_L R_R = A \]

\[ R_L Q = (y_L - y_R) \quad \therefore \quad y_R < 0 \text{ and } y_L > y_R \]  \hspace{1cm} (3.14)

\[ R_R Q = (x_R - x_L) \quad \therefore \quad x_R > x_L \]  \hspace{1cm} (3.15)

\[ \cos(a) = \frac{y_L - y_R}{A} \]  \hspace{1cm} (3.16)

\[ \sin(a) = \frac{x_R - x_L}{A} \]  \hspace{1cm} (3.17)

Substituting 3.13 and 3.16 in 3.11 we get,

\[ x_n = x_m + \frac{y_L - y_R}{2 \tan\left(\frac{\theta}{2}\right)} \]

And substituting 3.13 and 3.17 in 3.12,

\[ y_n = y_m + \frac{x_R - x_L}{2 \tan\left(\frac{\theta}{2}\right)} \]

Since \( M \) is the midpoint of \( R_L \) and \( R_R \).
\[ x_n = \left( \frac{x_L + x_R}{2} \right) + \left( \frac{y_L - y_R}{2 \tan \left( \frac{\theta}{2} \right)} \right) \]  
(3.18)

\[ y_n = \left( \frac{y_L + y_R}{2} \right) + \left( \frac{x_R - x_L}{2 \tan \left( \frac{\theta}{2} \right)} \right) \]  
(3.19)

Equations 3.18 and 3.19 give us the x and y coordinates respectively for case (a) where the partners are in the 1st and 4th quadrants. Case (a) is when \( \alpha < \pi \). Let us examine a case when \( \alpha > \pi \).

![Diagram showing new position \( R_N \) when \( \alpha > \pi \), case (d)](image)

**Figure 3-16 New position \( R_N \) when \( \alpha > \pi \), case (d)**

Figure 3-16 shows the case when \( \alpha > \pi \) and when the partners are in the 2nd and 3rd quadrants. When \( \alpha > \pi \), the agent stays on the same side of its old position with respect to line \( R_L R_R \), for reasons explained in section 3.7.
Figure 3.16 is comparable to figure 3.15 and in fact all the equations remain the same. If we rewrite those equations,

\[ x_n = x_m + d \cos(a) \]
\[ y_n = y_m + d \sin(a) \]

\[ d = \frac{A}{2 \tan\left(\frac{\theta}{2}\right)} \]

\[ \cos(a) = \frac{y_L - y_R}{A} \]
\[ \sin(a) = \frac{x_R - x_L}{A} \]

We can write the coordinates of the final position given by the equations given below and it is the same as equations 3.18 and 3.19.

\[ x_n = \left( \frac{x_L + x_R}{2} \right) + \left( \frac{y_L - y_R}{2 \tan\left(\frac{\theta}{2}\right)} \right) \]
\[ y_n = \left( \frac{y_L + y_R}{2} \right) + \left( \frac{x_R - x_L}{2 \tan\left(\frac{\theta}{2}\right)} \right) \]

From the above two derivations we can see that the coordinates of the final position is mathematically the same for both the cases of \( \alpha > \pi \) and \( \alpha < \pi \). This is really advantageous in an agent’s decision making. It does not have to have two different sets of equations for the two different cases.
In the above two cases the slope of line \( R_L R_R \) is negative in the local coordinate system. The derivation of the coordinates of the new position \( R_N \), does change a little when the slope of line \( R_L R_R \) is negative. Figure 3-17 shows such a situation for case (a).

\[ R_l(x_L, y_L) \]
\[ R_R(x_R, y_R) \]
\[ R_N(x_n, y_n) \]

**Figure 3-17 New position \( R_N \) of \( R_l \) when slope of \( R_L R_R \) > 0**

The above diagram shows the new position when the slope of \( R_L R_R \) > 0.

If \( \text{slope}(R_L R_R) > 0 \)

Then, \( \text{slope}(R_N M) < 0 \) \( (\because R_L R_R \perp R_N M) \)

\[ a = \tan^{-1}(\text{slope of } R_N M) \]

\[ \because \ a < 0 \]

Equations 3.11 and 3.12 will change in the following way.

\[ x_n = x_m + d \cos(-a) = x_m + d \cos(a) \quad (3.20) \]

\[ y_n = y_m + d \sin(-a) = y_m - d \sin(a) \quad (3.21) \]

Equation 3.13 remains the same as,
\[
d = \frac{A}{2 \tan(\theta/2)} \quad (3.22)
\]

Equation 3.14 is the same,

\[
R_L Q = (y_L - y_R) \quad \because y_R < 0 \text{ and } y_L > y_R \quad (3.23)
\]

But equation 3.15 will change as

\[
R_R Q = (x_L - x_R) \quad \because x_L > x_R \quad (3.24)
\]

\[
R_R Q = -(x_R - x_L)
\]

\[
\cos(a) = \frac{y_L - y_R}{A} \quad (3.25)
\]

\[
\sin(a) = \frac{-(x_R - x_L)}{A} \quad (3.26)
\]

Substituting 3.22 and 3.25 in 3.20 we get,

\[
x_n = x_m + \frac{y_L - y_R}{2 \tan(\theta/2)}
\]

And substituting 3.22 and 3.26 in 3.21,

\[
y_n = y_m + \frac{x_R - x_L}{2 \tan(\theta/2)}
\]

\[
\therefore x_n = \left( \frac{x_L + x_R}{2} \right) + \left( \frac{y_L - y_R}{2 \tan(\theta/2)} \right)
\]

\[
y_n = \left( \frac{y_L + y_R}{2} \right) + \left( \frac{x_R - x_L}{2 \tan(\theta/2)} \right)
\]
The above derivations addressed the two cases of $\alpha < \pi$ and $\alpha > \pi$. The derivations for the two cases were almost the same and we proved that the coordinates of the new position are given by the same equations.

When the slope of line $R_LR_R$ is negative the derivation is slightly different but the coordinates of the new position are given by the same as equations.

The rest of the cases fall into the derivations we did above. We can come to conclusion that however the partners are located or however the direction of the agent is oriented the new position will be given by the same set of equations. These equations are in terms of the coordinates of the partnering agents. The coordinates of the partnering agents are given by equation 3.10 in terms of the inter-agent distances and the angles of the partnering agents. Therefore an agent can find the coordinates of its new position with respect to its local coordinate system with the information collected from its partners.

For the agent to reach the new position it should know by what angle ($\gamma$) it has to turn and by what distance ($D$) it has to travel from its current position. In the subsequent sections we derive the formulae for these quantities.
3.10 Distance to Travel \((D)\)

The distance \(R_i\) has to travel from its old position to its new position can be easily found using the new coordinates of \(R_i\). The coordinates of \(R_O\), the old position of \(R_i\) is the origin of \(R_i\)'s local coordinate system. Distance \(R_OR_N\) is the distance \(D\), \(R_i\) has to travel to reach the new position. \(D\) can be expressed as follows using the coordinates of new position.

\[
D^2 = (x_n - 0)^2 + (y_n - 0)^2
\]

Substituting for \(x_n\) and \(y_n\) using equations 3.18 and 3.19 respectively,

\[
D^2 = \left[\frac{x_L + x_R}{2}\right]^2 + \left[\frac{y_L - y_R}{2 \tan\left(\frac{\theta}{2}\right)}\right]^2 + \left[\frac{y_L + y_R}{2}\right]^2 + \left[\frac{x_R - x_L}{2 \tan\left(\frac{\theta}{2}\right)}\right]^2
\]

\[
D^2 = \left[\frac{\sin\left(\frac{\theta}{2}\right)x_L + \cos\left(\frac{\theta}{2}\right)(y_L - y_R)}{2\sin\left(\frac{\theta}{2}\right)}\right]^2 + \left[\frac{\sin\left(\frac{\theta}{2}\right)(y_L + y_R) + \cos\left(\frac{\theta}{2}\right)(x_R - x_L)}{2\sin\left(\frac{\theta}{2}\right)}\right]^2
\]
Algorithm Development

D₂ = sm(θ)(x_L + x_R) + cos(θ)(y_L - y_R) K

K = 2 sin(θ)

Expanding and reducing using the trigonometric identities of:

sin²A + cos²A = 1

D₂ = [x_L² + y_L² + x_R + y_R - 2 cos(θ)(x_L x_R + y_L y_R)]

Using equation 3.20, we can further simplify:

x_L + y_L = [d_Lcos(θ)] + [d_Lsin(θ)]

y_R = [d_Rcos(θ)] + [-d_Rsin(θ)]

x_Ry_L - x_Ly_R = [d_Rcos(θ)][d_Lsin(θ)] - [d_Lcos(θ)][-d_Rsin(θ)]

= d_Ld_R sin(θ + θ)

= d_Ld_R sin(θ)

K² D₂ = [j² + i² - 2j(j cos(θ) - sin(θ))]

7/21/04
Dept. of Electrical Engineering
Equation 3.28 gives the distance $D$, $R_i$ has to travel from its current position to the new position. The equation is in terms of the known quantities of $R_i$, the inter-agent distances, inter-agent angle and constants $K$ and $\theta$. At each step $R_i$ measures the inter-agent distances and angles and computes $D$, the distance to move. For $R_i$ to travel a distance $D$, it needs to know the angle it has to turn from its current direction of orientation. We shall denote the angle to turn as $\gamma$. In the next section we will derive a formula for $\gamma$.

3.11 Angle to turn ($\gamma$)

$\gamma$ is the angle to be turned by $R_i$ from its current direction of orientation. This is the angle line $R_O R_N$ (see figure 3-18) makes with the positive x-axis of $R_i$'s local coordinate system. By finding the slope of line $R_O R_N$ we can find angle $\gamma$.

$$\tan(\gamma) = \frac{y_n}{x_n}$$

Substituting for $y_n$ and $x_n$ using equations 3.18 and 3.19 respectively,

$$\tan(\gamma) = \frac{\frac{y_L + y_R}{2} + \frac{x_R - x_L}{2\tan(\frac{\theta}{2})}}{\frac{x_L + x_R}{2} + \frac{y_L - y_R}{2\tan(\frac{\theta}{2})}}$$

$$\tan(\gamma) = \frac{\sin(\frac{\theta}{2})(y_L + y_R) + \cos(\frac{\theta}{2})(x_R - x_L)}{\sin(\frac{\theta}{2})(x_L + x_R) + \cos(\frac{\theta}{2})(y_L - y_R)}$$

Substituting for $x_L, y_L, x_R$ and $y_R$ using equation 3.10 and simplifying
\[
\tan(\gamma) = \frac{d_R \cos\left(\alpha_R + \frac{\theta}{2}\right) - d_L \cos\left(\alpha_L + \frac{\theta}{2}\right)}{d_R \sin\left(\alpha_R + \frac{\theta}{2}\right) + d_L \sin\left(\alpha_L + \frac{\theta}{2}\right)}
\]

\[
\gamma = \tan^{-1} \left[ \frac{d_R \cos\left(\alpha_R + \frac{\theta}{2}\right) - d_L \cos\left(\alpha_L + \frac{\theta}{2}\right)}{d_R \sin\left(\alpha_R + \frac{\theta}{2}\right) + d_L \sin\left(\alpha_L + \frac{\theta}{2}\right)} \right]
\] (3.29)

3.12 Direction of Orientation Adjustment

Imagine the situation when the agents in the colony have formed a convex polygon. After a convex polygon has been formed it is important that all the agents select the partners that are on either side of them at every iteration. If at each iteration an agent selects different partners then the formation will be unstable and will not converge to a regular polygon.

Figure 3-19 Five agents in formation of a regular pentagon showing preferred direction of orientation
Figure 3-19 shows five agents in formation of a regular polygon. If the agents have the positions shown in the figure the agents have to stick with the same partners for the figure to converge to a regular pentagon. In other words in a formation if all the agents are external agents then the agents have to make sure that they select the same partners in the next iteration. For this to happen the directions of orientation of the agents have to be in the convex region of the formation. In other words it can be said that the agents have to be looking outwards of the formation. If an agent is looking inwards as shown by the arrowed dotted line for agent $R_1$, then it will select the wrong partners and disrupt the formation. In the case of $R_1$ looking inwards, it will select $R_3$ and $R_4$ as its partners and move towards line $R_3R_4$. This movement will go against the formation of the pentagon.

Even when the agents are internal it is preferable to select the same partners. If the agents keep selecting different agents as their partners the movement will be very haphazard. By adjusting the direction of orientation to some extent it is possible to select the same partners. As shown in figure 3-19 when the agent is an external agent the direction of orientation should be looking outwards of the formation. $R_i$ after moving from the old position $R_O$ to the new position $R_N$, has to turn a certain angle to make the necessary alteration to the direction of orientation. If the new direction of orientation is on the perpendicular bisector of line $R_RR_L$, to some extent we can expect that the same partners will be selected in the next iteration provided that the partners have not moved far away.

Let us denote the angle $R_i$ has to turn by as $\varepsilon$. This angle is shown for the two different cases of $\alpha < \pi$ and $\alpha > \pi$ in figure 3-18. From the figure it is clear that we can
express this angle as the differences of arc tangent of the slopes of lines $R_O R_N$ and $M R_N$. If we denote the arc tangent of the slope of line $R_N M$ as $\xi$ we can express $\varepsilon$ in terms of $\xi$ and $\gamma$, the arc tangent of line $R_O R_N$.

\[ \varepsilon = \xi - \gamma \quad (3.30) \]

Coordinates of $R_N$ are $(x_N, y_N)$ and of $M$ are $(x_m, y_m)$.

\[ \xi = \tan^{-1}\left(\text{tangent}(R_N M)\right) \]

From equations 3.18 and 3.19

\[
\begin{align*}
    x_n &= \left(\frac{x_L + x_R}{2}\right) + \frac{y_L - y_R}{2 \tan \left(\frac{\theta}{2}\right)} \\
    y_n &= \left(\frac{y_L + y_R}{2}\right) + \frac{x_R - x_L}{2 \tan \left(\frac{\theta}{2}\right)} \\
    x_m &= \frac{x_L + x_R}{2} \quad \text{and} \quad y_m = \frac{y_L + y_R}{2}
\end{align*}
\]
\[
\therefore \xi = \tan^{-1}\left[\frac{\left(\frac{y_L + y_R}{2}\right) + \left(\frac{x_R - x_L}{2\tan(\theta/2)}\right) - \left(\frac{y_L + y_R}{2}\right)}{\left(\frac{x_L + x_R}{2}\right) + \left(\frac{y_L - y_R}{2\tan(\theta/2)}\right) - \left(\frac{x_L + x_R}{2}\right)}\right]
\]

Simplifying,

\[
\xi = \tan^{-1}\left[\frac{x_L - x_R}{y_L - y_R}\right]
\]

Substituting for \(x_L, y_L, x_R\) and \(y_R\) using equation 3.10 and simplifying

\[
\xi = \tan^{-1}\left[-\frac{d_L \cos(\alpha_L) - d_R \cos(\alpha_R)}{d_L \sin(\alpha_L) - d_R \sin(\alpha_R)}\right]
\]

(3.31)

\(\varepsilon = \xi - \gamma\) and in figure 3-20(a) \(\xi > \gamma\), and therefore \(\varepsilon > 0\). In figure 3.20(b) \(\xi < \gamma\) and \(\varepsilon < 0\). As a convention we could state that if \(\varepsilon > 0\) \(R_i\) should turn counter clock wise and if \(\varepsilon < 0\) \(R_i\) should turn clock wise.

### 3.13 Modifications of the circle algorithm

With the algorithm we developed above we can make a randomly distributed colony of agents to form a uniform circle. In the next chapter of simulation and results we will show that it converges to a circle where all the agents are equidistant from their partners. After the colony has converged to a circle, by altering one parameter in the algorithm we can modify the circle according to the need of the user.
3.13.1 Circle with increasing and decreasing radius

The circle algorithm can be altered to make it either to uniformly grow larger by increasing the radius or shrink smaller by reducing the radius.

The inter-agent angle each agent is trying to make with its partners is made to be $\theta$ where $\theta = \frac{(n-2)\pi}{n}$. By slightly increasing or decreasing the value of $\theta$, the radius of the circle can be made to increase or decrease. Let us denote the slightly changed $\theta$ as $\theta_d$. If $\theta_d$ is made smaller than $\theta$ at each step $R_i$ will move outwards of the circle to make the inter-agent angle smaller. Similarly if $\theta_d$ is made larger than $\theta$ at each step $R_i$ will move inwards of the circle to make the inter-agent angle larger. When the whole colony of agents does the same the circle will grow or shrink.

![Diagram](image)

Figure 3-21 A regular hexagon growing or shrinking depending on $\theta_d$
In Figure 3-21 a regular hexagon is shown by the dark lines. R₁ is an agent located at the vertex of the regular hexagon. In the case where \( \theta_{d1} < \theta \), R₁ goes outward of the formation locating itself on the vertex of a bigger hexagon. When \( \theta_{d1} > \theta \), R₁ moves inwards. The angle adjustment made at the end of each iteration makes sure that all the agents look out wards of the formation. Since all the agents in the formation will do the same the locations of the agents in the next iteration will be on the vertices of another regular hexagon. When this happens repeatedly the hexagon will either grow or shrink gradually.

3.13.2 Circle with a specific radius

In the previous section we discussed how we can make a circle grow or shrink by changing the value of \( \theta \). This could be used to make a circle of a required radius. In a circle with \( n \) agents, or in other words in a regular polygon with \( n \) sides we can find a relationship between the length of a side and the radius of a circle.

![Diagram showing relation between inter-agent distance (ird) and circle radius (RAD)](image)

Figure 3-22 Relation between inter-agent distance (ird) and circle radius (RAD)
Figure 3-22 shows the inter-agent distance (ird) and the radius of the circle (RAD) for a hexagon. For a \( n \) sided regular polygon the interior angle at the center can be written as \( \frac{2\pi}{n} \). Applying cosine rule for the triangle we get,

\[
ird^2 = 2RAD^2 - 2RAD^2 \cos\left(\frac{2\pi}{n}\right)
\]

\[
ird = RAD \sqrt{2\left(1-\cos\left(\frac{2\pi}{n}\right)\right)} \tag{3.32}
\]

By this equation if all the agents make their inter-agent distances equal to a certain distance then the formation will be on a circle of required radius. In the next chapter we will simulate to verify this statement.

3.14 Problems in the algorithm

Conceptually the algorithm we have developed seems to be acceptable for circle formation. During simulation we came to know some of the drawbacks. In this section we shall address these problems and give possible remedies.

3.14.1 Direction of orientation at step distance

When \( R_i \) moves a step distance and not the whole distance \( D \), then the new position of \( R_i \) will not be on the perpendicular bisector of \( R_i R_R \). But the direction of orientation of \( R_i \) will still be in the direction of the perpendicular bisector. Figure 3-23 shows such a situation. \( R_N \) is the position of \( R_i \) after moving step and \( R_{final} \) is the position it should have moved to. The main idea of direction of orientation adjustment is to make sure to some extent the selection of same partners. The direction of orientation after
moving step, $R_i$ would still select the same agents as its partners if they had not moved to other locations.

![Diagram of orientation](image)

**Figure 3-23 Direction of orientation for $R_i$ after moving a step**

### 3.14.2 Same partner selection

Due to the positioning and orientation of agents, two or more agents could select the same agents as their partners. If the same two agents are selected as partners by more than one agent then the new position of those agents will be the same. These agents will then proceed to the same location and collide. Figure 3-24 shows such a situation where agents $R_3$ and $R_4$ select $R_1$ and $R_2$ as their partners. Both these agents will move to point A and collide.
In the model we have developed, a partial solution is inherent in the model itself. In the model, agents have limited span of scan. Therefore the number of agents detected by $R_i$ is reduced only to the agents within its span of scan. This reduces the chances of agents selecting the same partners. If a particular agent can be detected by the rest of the colony the probability of that agent being selected as a partner is higher. With limited span of scan the number of agents that would detect it will be lesser thereby reducing the chances of it being selected as a partner.

The issue is also overcome by $R_i$ not moving the calculated distance completely but by moving just a distance of $step$. When the agents move a distance of only a $step$ in the next iteration the partners may have moved and $R_3$ and $R_4$ may select some other agents as their partners. Even if $R_1$ and $R_2$ stay at the same place, when $R_3$ and $R_4$ gets closer and closer to $A$ at one point $R_3$ will select $R_4$ as its partner and the same for $R_4$. When that happens the collision of $R_3$ and $R_4$ will be avoided.

However, even within a $step$ movement if the agents are moving towards reaching the same point then collision cannot be avoided. To overcome the problem we have used a strategy in which agents do not move the complete distance if there is a possible
collision ahead. Instead one agent moves less than the distance it has to move. In a practical situation we need to have a sensor on \( R_i \) to detect a possible collision. If \( R_i \) senses a possible collision then it moves a distance less than it had calculated. By reducing the distance to travel at one point \( R_i \) will detect the colliding agent as its partner and move away from it.

### 3.14.3 Irregular formation

When the *step* size is large sometimes the formation was found to result in irregularly distributed circles. The formation, instead of converging to a regularly distributed circle resulted in a non converging unevenly distributed circle. When the *step* size is large the agents' change of position is too far, making the formation unstable. The main reason for this phenomenon is because even though at a particular iteration each agent tries to make the inter-agent distances equal, it is not guaranteed that in the new position the inter-agent distances between the partners will be equal. The reason for this is that the partners also would have moved to other locations. If each agent moves serially, one after the other, then the algorithm will result in a regular polygon. When agents move serially the new position of \( R_i \) is guaranteed to be equidistant from its partners since the partners are stationary. At the next iteration when another agent scans the terrain the new position of the agent that moved serially will be scanned. To implement serial movement the agents have to be numbered and then the colony will no more be homogeneous.

Figure 3-25 shows an example of a formation that will not result in a regular polygon. If four agents are in a rectangle with their inter-agent angle equal to \( \frac{\pi}{2} \) radians
the formation at the next iterative step will also be a rectangle. For instance agent $R_1$ will select agent $R_2$ and $R_4$ as partners and move to a position $R_{N1}$ such that,

$$R_{N1} = R_4 \quad \text{and} \quad R_{N1} = R_2 \quad \text{and also}$$

$$R_4 R_{N1} R_2 = \frac{\pi}{2}$$

Similarly $R_2$, $R_3$ and $R_4$ will move to $R_{N2}$, $R_{N3}$ and $R_{N4}$ respectfully. From the diagram we can see that the diagonals of $R_1 R_2 R_3 R_4$ and $R_{N1} R_{N2} R_{N3} R_{N4}$ will be equal and the interior angles of the diagonals will also be the same. Therefore we can conclude that $R_{N1} R_{N2} R_{N3} R_{N4}$ will also be a rectangle. The four agents will therefore switch between the two rectangles without converging to a square. This is just an example to explain the phenomenon of the colony not forming a regular polygon.

Figure 3-25 Four agents in a rectangle moving to new locations on another rectangle
Figure 3-26 Agents getting too close to each other

Figure 3-26 shows situation of four agents. This diagram will give another explanation of forming irregular polygons when the step is large. $R_2$ will select $R_1$ and $R_3$ as its partners and move to its new position $R_{2N}$. Similarly $R_3$ will select $R_2$ and $R_4$ as its partners and move to its new position $R_{3N}$. In the next iteration $R_2$ and $R_3$ are close to each other and farther away from their other partners $R_1$ and $R_4$. This situation arises when an agent is too close to one partner than the other. Figure 3-27 shows the simulation result of such a situation. The agents on the right side of the formation are closer to one partner than to the other. The large step size makes them in the next iteration get closer to the partner that was further away move away from the one that was closer. This continues resulting in an irregular unstable polygon.
Another reason for irregular polygon is that agents become internal and external alternatively. The internal agents select the external agents as their partners and vice-versa. As a result the internal agents move outwards and become external agents and vice versa. Figure 3-28 shows such a situation and this phenomenon keeps repeating in subsequent iterations.

Figure 3-27 Simulation results for irregular shape formation

Figure 3-28 Simulation results for change of internal external agents
One possible solution for problem is to have small step size. When the step size is small it takes a long time for the agents to reach the periphery of the circle. When the step size is large the formation is irregular and does not converge. Both these problems can be overcome by having two different step sizes. Step can be made large initially to bring the agents to the periphery of the formation faster and then made small to make the formation a regular polygon. The ideal time to switch the step size will be when all the agents have become external. This varies from distribution to distribution, the number of agents and also in the initial step size. We shall address the point at which the step size has to change in the next chapter when we analyze the algorithm.

3.14.4 Inappropriate Direction of Orientation

The direction of orientation of some agents may make it to travel a longer distance than it should. This in turn affects the formation of the whole colony. For instance an external agent due to inappropriate orientation will select partners that will make it to move to the interior of the formation. An internal agent positioned close to the periphery of the formation may select agents that are in the opposite side of the formation and move there. In figure 3-29, R₅ is close to the periphery of the formation. If the direction of orientation is as shown by the arrowed dotted line then it would have selected R₄ and R₆ as its partners and moved towards line R₄R₆. If the direction of orientation is as shown by the solid arrowed line then it will select R₂ and R₃ as its partners and move towards line R₂R₃. This is inappropriate and results in a non optimal solution. In section 3.9, by adjusting the angle of orientation we have tried to minimize this problem.
Another situation when the direction of orientation affects the formation is shown in figure 3-30. When the span of scan of the agent is small an agent can find itself in a situation shown in figure 3-30. Here R₇, due to its direction of orientation and limited span of scan selects R₁ and R₆ as its partners. Since the inter-agent angle \(\alpha\) is greater than \(\pi\) radians it does not cross to the other side of its partners. R₇ gets stuck at the location whereas the other agents continue to form a regular polygon. We can assure that if the span of scan is greater than the radius of the circle formed then this stagnation will not occur. If the span of scan is greater than the radius of the circle then R₇ will select the agents on the opposite periphery of the circle and move toward them. For this case agents R₃ and R₄ will be selected and R₇ will move towards line R₃R₄. If the radius of span of scan is greater than the circle formed then R₇ will be able to scan all the agents resulting in global scanning, which is not what we are trying to achieve.

A solution for this problem is to change the direction of orientation of R₇ by \(\pi\) radians if the partnering agents are greater than a certain distance. If R₇ changes the
direction of orientation then it will detect the appropriate partners. To make a check on the distance $R_7$ should know how close its partners should be. We will discuss about this issue further in the next chapter section 3.5.

![Diagram of agents in a circle]

**Figure 3-30** $R_7$ getting stagnant at the wrong position in the formation

Inappropriate direction of orientation along with small span of scan some times gives rise to another situation. For some initial distributions more than one agent gets trapped in the middle. When this happens, the agents that are trapped keep moving around in the middle part of the circle without reaching the periphery of it. Figure 3-31 shows such a situation where two agents get trapped and move around in the middle part of the circle. The thick lines show the path of the trapped agents. This problem can also be eradicated by the method we explained above. After a certain number of steps if all the agents do a check on how far away both their partners are they can come to know about their position in the circle. The agents that are on periphery of the circle will have their partners closer than a certain limit and the partners of the agents that are in the middle
will be further away than that limit. A discussion on how the limit is calculated is given on section 3.5

![Figure 1-31 Two agents trapped in the middle. Their path is also shown](image)

3.14.5 Very small span of scan

When the span of scan is small, apart from the agents getting stuck in the middle can also create another problem. Some of the agents may get lost in the terrain without being able to detect any partners. Each agent in the colony should be able to detect at least two partners for it to move according to the algorithm. We could visualize this by saying that the circles of span of scans of all the agents should intersect with each other.

If an agent is lost in the terrain it also affects the rest of the colony. Each agent moves after calculating $D$ and $\gamma$. These are calculated using $\theta$. $\theta$ is dependent on $n$, the total number of agents, in the colony. If an agent is lost in the active colony then the number of agents will be $n-1$. There is no way for the agents in the colony to know about the loss of an agent and will continue to make $\theta$, with its partners.
If $\theta_n$ is the inter-agent angle for $n$ agents and $\theta_{n-1}$ is the angle for $n-1$ agents we can write these by the following equations.

$$\theta_n = \frac{(n-1)\pi}{n} \quad \theta_{n-1} = \frac{(n-1)-1}{n-1} \pi$$

$$\theta_n = \pi \left[ 1 - \frac{1}{n} \right] \quad \theta_{n-1} = \pi \left[ 1 - \frac{1}{n-1} \right]$$

From the equations it is clear that $\theta_n > \theta_{n-1}$. $n-1$ agents are trying to make an inter-agent angle of $\theta_n$ which is greater than the appropriate angle $\theta_{n-1}$. At each iteration the agents are trying to make an angle greater than the appropriate angle. To do so, the agents have to move inwards of the formation. This will make the formation smaller and smaller.

Small span of scan can also result in agents forming two circles instead of one. These circles will shrink gradually for the reason explained above. The inter-agent angle each agent will try to make will be the angle for the whole colony which is greater than the angle for the formation of one of the circles. A simulation result is given in figure 3-32.
Figure 3-32 Formation of two shrinking circle due to small span of scan
Missing Page
4. Simulation and Results

In this chapter we will analyze the simulation results of the algorithm we developed in the previous chapter. Simulations were obtained using MATLAB software. The simulations were made to match how a real agent would go about in performing the circle formation algorithm. Each agent in the colony will measure the distances and the angles of the partners, calculate the angle to turn and the distance to travel and then move to the new position. In the simulations each agent is represented as a point and assumed to be not taking any physical space. Consequently collision avoidance techniques are not incorporated into the formation algorithm.

$n$ agent positions and their respective directions of orientations were generated using the `rand` function in MATLAB. `rand(l,n)` generates a 1-by-n matrix with random entries, chosen from a uniform distribution on the interval (0.0,1.0). Therefore the colony will be randomly spread in a square area of size 1-by-1 with random initial orientations.

In simulation, for $R_i$ to measure the distances and angles of its partners the coordinates of the agent positions under its span of scan have to be relative to its current position. For this measurement the coordinates of those agents have to be changed to a new coordinate system and that would be the local coordinate system of $R_i$. The origin of the local coordinate system will be the current position of $R_i$ and the positive x-axis will be the direction of orientation of $R_i$. In simulation the global coordinates have to be changed to local coordinates. In the next section we will see how this axis change is done.
4.1 Coordinate Axis Change

Figure 3-1 Change of axis with respect to the position and direction of R_i

Figure 3-1 shows the position of agents R_i and R_1. Let R_i be the scanning agent and R_1 be an agent that is scanned. R_i's current position and direction of orientation, denoted by DIR_i, will be the origin and direction of positive x-axis respectively of the local coordinate system of R_i. R_iR_1 will be the inter-agent distance and the inter-agent angle will be the angle between R_iR_1 and X_{local}, denoted by A_1.

For simulation, we converted the global coordinates of the agents to a local coordinate system with respect to the position and direction of the agent that scans. In the global coordinate system we denote the coordinates of R_i and R_1 as (x_i, y_i) and (x_1, y_1) respectively. In the local coordinate system the coordinates of R_i will be (0, 0) and let the coordinates of R_1 be (x_{1,local}, y_{1,local}).

We can write the slope of line R_iR_1 in the global coordinate system as,
\[
\tan(A_i + \text{DIR}_i) = \frac{y_i - y_1}{x_i - x_1}
\]

\[
A_i = \tan^{-1}\left(\frac{y_i - y_1}{x_i - x_1}\right) - \text{DIR}_i
\]

This will be the inter-agent angle made with R_1 by R_i. The inter-agent distance R_iR_1 can be written as,

\[
R_iR_1 = \left[ (x_i - x_1)^2 + (y_i - y_1)^2 \right]^{\frac{1}{2}}
\]

Therefore the local coordinates of R_1 in the local coordinate system of R_i can be given by,

\[
x_{\text{local}} = R_iR_1 \cos(A_i)
\]
\[
y_{\text{local}} = R_iR_1 \sin(A_i)
\]

This conversion is done to all the agents under the span of scan of R_i. The agents with the smallest angle and largest are selected as the partners of R_i. This will necessarily mean that if R_i scans to its left it will detect the agent with the smallest angle and if it scans to the right it will detect the agent with the greatest angle. Then the inter-agent distance is calculated as,

\[
R_iR_1 = \left( x_{\text{local}}^2 + y_{\text{local}}^2 \right)^{\frac{1}{2}}
\]

After determining the distance (D) R_i has to travel from its current position and the angle (\gamma) it has to turn from its current direction of orientation we can find the coordinates of new position with respect to R_i's local coordinate system. This coordinates have to be changed back to the global coordinate system to plot the new position.

In the global system the new direction of orientation will be,
$$\text{DIR}_{\text{new}} = \text{DIR}_{i} + \gamma$$

The new coordinates of $R_i$ can be written as,

$$x_{\text{new}} = x_i + D \cos(\text{DIR}_{\text{new}})$$

$$y_{\text{new}} = y_i + D \sin(\text{DIR}_{\text{new}})$$

We have set up the necessary steps for the simulation program. In the next section we present the flow chart of the algorithm.
4.2 Flow Chart of the Algorithm

Initialization

for i = 1 to n

Scan and convert to local coordinates

Select two partners

Measure $d_L, d_R, \alpha_L, \alpha_R$

Compute $D, \gamma, \varepsilon$

If $D > \text{step}$

$D = \text{step}$

Find the new position

Update the global coordinate system. If two robots occupy the same position place one of them at the mid point of its old and new position

$n$ - # of robots

$x, y$ coordinates of colony

direction of orientation of robots

$\text{step}$ size

range - span of scan

$\theta$ Eqn 2.8

$K$ Eqn 2.27

Robots under the span of scan

Smallest and largest angle robots

Inter-robot distances and angles

$D$ Distance to travel. Eqn 2.28

$\gamma$ Angle to turn. Eqn 2.29

$\varepsilon$ Angle adjustment 2.30
The above algorithm is followed iteratively till the formation converges to a circle. The ultimate circle formed and the manner in which it is formed depends on several factors. They are,

1. Number of agents (n)
2. The initial distribution of the colony
   a) The positions of the agents
   b) The direction of orientation of the agents
3. span of scan
4. step size

To analyze the effect of these factors on the formation we need to keep some of them constant and change the rest. For most of our simulation we have kept the number of agents in the colony as \( n = 50 \). The initial distribution of the colony is also kept fixed. The coordinate values of the positions of this distribution and the direction of orientation of the agents are given in appendix A.1. The initial distribution of this colony is shown in figure 4-2.
4.3 Convergence of the algorithm

Before we start analyzing the effect of various factors on formation we need to make sure that the algorithm converges to a regular polygon. Convergence can be checked by examining the inter-agent distances and the inter-agent angles made with the partnering agents for each agent. If the inter-agent distances and the inter-agent angles are the same for all the agents then we can say that the formation is a regular polygon. By plotting the differences of distances between the two partnering agents against number of iteration we can check the convergence. We shall call the difference of distance between the two partnering agents as error. Error will be large for every agent initially since the partnering agents will not be equidistant. Error should get smaller with the number of iteration as each agent is trying to move to a point equidistant from its partners. To
examine the pattern of error we plotted the maximum, the minimum and the mean of error of the agents against number of iteration. The inter-agent angle value was also plotted. We simulated with the initial colony shown in figure 4-2. The results are given in figure 4-3.

Figure 4-3 Convergence graphs, \( n = 50 \), with unlimited step and range. Error for iterations from (a) 1-100 (b) 101-200 (c) Inter-agent angle (d) final formation
The results shown in figure 4-3 were obtained for a distribution where \( n = 50 \). The *step* was made to be unlimited (very large) and span of scan was also made to be unlimited. Figure 4-3 (a) shows that initially the *error* is very high and slowly they get lesser. It takes about 72 iterations for the algorithm to converge to a regular polygon. Figure 4.3 (b) shows that the *error* reduces considerably, in the order of \( 10^{-3} \), after the 100\(^{th} \) iteration.

After 200 iterations the values of *errors* are as follows.

- Maximum error = \( 3.3153 \times 10^{-4} \)
- Mean error = \( 1.1741 \times 10^{-4} \)
- Minimum error = \( 4.1804 \times 10^{-7} \)

Figure 4.3 (c) shows the maximum, mean and minimum value of the inter-agent angle (\( \alpha \)). All the three graphs converge to a certain value.

- Maximum angle = \( 3.4010 \)
- Mean angle = \( 3.2044 \)
- Minimum angle = \( 3.0078 \)

The explanation given in section 3.14.3 is the reason why the three graphs do not to converge to a single line. Agents in the colony alternatively become *external* or *internal*. When an agent is *internal* the value of the inter-agent angle will be less than \( \pi \) radians and when an agent is *external* the value of inter-agent angle will be greater than \( \pi \) radians, which is why we get two different values for maximum and minimum inter-agent angle. The mean value is close to the required angle (3.2673) to be made by the agents to form a 50 sided regular polygon.
From the convergence of the error graphs to zero and the convergence of the inter-agent angle graphs over a period of time we can surely say that the formation converges to a regular polygon.

An interesting result we found in simulation was that when $n$ is an odd number the error did not converge to zero. After a certain point the error graphs follow a certain pattern without converging to zero. We simulated the same distribution we used for the earlier case leaving the 50th agent. Therefore, in the new distribution $n = 49$ and is an odd number.

![Error vs. # of iterations](image1)

**Figure 4-4** (a) Error vs. # of iterations. $n = 49$, with unlimited step and range. (b) Final formation after 300 iterations

Figure 4-4(a) shows the result of error when $n = 49$. After a certain point the error graph follows a repetitive pattern. Figure 4-3(b) shows the final formation after 300 iterations. In the final formation the agents on the left side of the formation are closer to one partner than to the other. In the next iteration they get closer to the partner that was
far away and further away to the partner that was closer. This phenomenon keeps repeating for reasons explained in section 3.14.3. If the step size is made small to the order of \(10^{-2}\) convergence can be obtained even for a colony with odd number of agents.

The problem in having a small step is that the convergence will be slower since it takes a while for the agents to reach the perimeter of the circle. When step is large the agents get to the perimeter of the circle faster but take a while to form a regular polygon when \(n\) is even and never when \(n\) is odd. In figure 4-3(a) we can notice that it takes about 72 iterations for the formation to converge for a random distribution when \(n = 50\). The problems of non-convergence when \(n\) is odd and the delay in convergence when \(n\) is even can be overcome by the strategy explained in section 3.14.3. In the next section we shall address the effect of step on the formation.

4.4 Effect of step on Formation

The distance \(R_i\) can travel in a unit time or the step size affects the speed of formation and the final formation. If step is too large the formation is not regular and if it is small it takes a longer time for the colony to form the circle. To examine the effect of step we simulated the same initial colony (figure 4-2) with various step sizes. Here we use unlimited scanning for the agents to negate the influence of span of scan. The number of agents was taken as an odd number \((n = 49)\) since this is when the algorithm did not converge. We can see the effects of various step sizes by comparing the speed of convergence by examining the error graphs and the final formation.
Simulation and Results

(a) step = 0.01

(b) step = 0.025

(c) step = 0.05
Simulation and Results

(d) $step = 0.075$

(e) $step = 0.1$

(f) $step = 0.25$
Simulation and Results

Figure 4-5 Effect of step size on formation. The right column shows the formation after 50 iterations and the left column shows the error graph.

In figures 4-5(a) and 4-5(b) we observe that since step is small the final formation is a regularly distributed circle but the speed of convergence is slow. When the step size is increased the agents come to the periphery of the formation faster but they do not form a regularly distributed circle. To check the optimal step size we simulated the algorithm for step sizes between the range of 0.012 and 0.04. The optimal step we found for this distribution was 0.026. The error graph started to converge around the 30th step. The error graph and the formation are given in figure 4-6. When the step was increased the
convergence rate became slower and when \textit{step} was 0.032 and above the error graph did not converge.

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{error_graph.png}
\caption{The error graph and the final formation for an optimal \textit{step}}
\end{figure}

\textit{step} = 0.026

We cannot conclude from the simulation of a particular colony about the optimal \textit{step} size for the algorithm. The optimal \textit{step} depends a great deal in the initial distribution. A conclusion we can make about the formation algorithm is that when \textit{step} is small it takes a longer period for the agents to reach the perimeter of the circle and when it is large the circle does not or takes a long time to converge to a regularly distributed circle.

A strategy to avoid this trade off is to have varying \textit{step} sizes. If \textit{step} is large initially it allows the agents to reach the periphery faster and then, when \textit{step} is made smaller the agents can adjust themselves to distribute regularly on the periphery of the circle. The best instant to switch from large \textit{step} to small \textit{step} is when all the agents in the colony are on the periphery or close to the periphery of the circle. If the span of scan is unlimited and if an agent is \textit{external} then we can definitely say that the agent is on the
periphery of the formation. The iteration at which all the agents in the colony become external will be the best moment to switch step size. This point again depends on the initial step size and also the span of scan of agents. With limited span of scan it is difficult to come to a conclusion if an agent is external or not. To understand the point or the iteration at which all the agents in the colony become external we simulated the same initial colony (figure. 4-2) with different step sizes and unlimited span of scan. The number of agents that are external was plotted against iteration for different step sizes. As predicted when step is large agents become external sooner. Figure 4-7 shows the graph of the number of agents that are external at certain iteration. The graphs depend considerably on the distribution of the initial colony. We plotted the graph for different randomly generated distributions and the graphs were not exactly the same as shown in figure 4-7(a), but the nature of the graphs remained the same.

(a) For the distribution of figure 4-2  
(b) For another random distribution

Figure 4-7 Number of agents that become external against number of steps

For faster convergence and precise formation the step size can be made large till the agents become external and made small thereafter. We can see from the graph that
Simulation and Results

around the 10th step all the agents are external except for the case when step is 0.1. To make convergence faster we simulated with large step for the first ten iterations and then made step to be 0.01 for the rest of the iterations. We were able to get faster convergence than we got for the optimal unvarying step. The best result was obtained when step was 0.5 for the first ten steps and made to be 0.01 for the rest of the iterations. The point of convergence was found to be around the 15th iteration. The results are shown in figure 4-8. The figure shows the final formation and the error graph. It is clear that the formation is regular and the convergence is much faster than the graph shown in figure 4.6 which was for the unvarying optimal step.

![Figure 4-8](image)

Figure 4-8 The initial colony shown in figure 4-2 simulated with step =0.5 for first 10 iterations and 0.01 thereafter.

4.5 Effect of span of scan on formation

The other main factor that influences the shape formation is the Span of scan. When the span of scan is made to be very small to the extent that an agent cannot detect another then that agent will basically be lost in the terrain. Span of scan has to be
reasonably large so that the agents are interconnected with each other. Figure 4-1 shows how the circles of span of scan should intersect each other. If a circle of span of an agent doest not intersects with another, the agent will be lost in the terrain. Therefore the span of scan has to be reasonably large enough for the algorithm to form a circle.

To study the effect of span of scan on formation we simulated the same distribution shown in figure 4-2 (n = 50) with different ranges of span of scan. For these simulations we take \textit{step} to be equal to 0.01. The results of the simulations are shown in figure 4-9 for varying spans of scan from 0.1 to unlimited.
Simulation and Results

(a) span of scan = 0.2

(b) span of scan = 0.3

(c) span of scan = 0.4
Simulation and Results

Figure 4-9 Effect of span of scan on formation for the colony shown in figure 4-2.

Figure 4-9 (a) shows the formation and the error graph when the span of scan is 0.2. When the span of scan is so small the agents gather into two groups and form two different circles. Moreover the circles will progressively shrink in radius for reasons explained in section 2.14.5. The agents in the two circles are trying to form an angle that is calculated for the circle formation of fifty agents. The number of agents in each circle is smaller than fifty and therefore the appropriate angle for the agents in those circles are smaller than the one for fifty agents. Hence at each step by moving inside the circle an
agent tries to make a larger angle than the one appropriate for that particular circle. This explains why the circles shrink. The error graph does not show this phenomenon since in the formation of two circles also the partnering agents are equidistant. We need to look at another graph to observe this phenomenon.

If we calculate the distance of each agent from the centroid of the colony and plot the maximum, minimum and mean of the distances from the centroid we can observe the situation where the agents even though are equidistant from the partners do not converge into one circle.

![Graph showing maximum, mean and minimum distances from the centroid](image)

Figure 4-10 Maximum, mean and minimum distances from the centroid for the case when span of scan is 0.2 and 0.3

Figure 4-10 (a) shows the distance of agents from the centroid of the colony for the case when span of scan is 0.2. The graph does not converge since the agents are shrinking in two circles. The formation of two shrinking circles can be avoided if the span of scan is made a little larger. It can also be avoided if all the agents are interconnected within the whole colony. This is not within our control since it depends on the initial distribution.
When the span of scan is made a little larger to 0.3 a situation of one agent getting trapped in the middle arises. The reason for this situation was explained in section 12.14.4 and 12.14.5. The error graph again does not show this situation by its non convergence. If we look at the distance from the centroid graph we can clearly see that even though the maximum distance and mean distance are converging together the minimum distance does not because of the stagnation of one agent in the middle.

This situation can be avoided if the agent that gets stuck in the middle changes its direction of orientation. From the simulations of different colonies one definite conclusion we can make is that if the agents are spread in a 1 x 1 square area the radius of the circle will be less than 0.5. If the radius is less than 0.5 then the circumference of the circle will be less than $\pi$. Therefore the distance to a partnering agent after convergence will be approximately equal to $\frac{\pi}{n}$. In the situation where an agent gets stuck in the middle, its partners will be further away than $\frac{\pi}{n}$. The accuracy of this number was checked with simulation result. After convergence, by checking the distance with the partnering agent, an agent would come to know if it is stuck in the middle or not. If the partnering agents distance is much greater than $\frac{\pi}{n}$, then by changing its direction of orientation by $\pi$ radians the stagnant agent can be made to move to the periphery of the circle.

Another conclusion we can make from the graphs in figure 4-7 (c) – (e) is that the rate of convergence is slower when the span of scan is smaller.
4.6 Effect of span of scan and step on final radius

The step size and span of scan influence the final radius of the circle formed. The radius greatly depends on the initial distribution and the number of agents. Even though the final radius depends on the initial distribution it followed a general pattern. The final radii were large when span of scan and step were large. We calculated the final radius of the initial distribution in figure 4-2. The final radius is given as a surface plot for different step sizes and spans of scan and shown in figure 4-11.

![Surface plot of final radii](image)

**Figure 4-11** Final radii shown as a surface plot for different step and span of scan
4.7 Circle formation for various initial distributions

In this section we shall test our algorithm for different initial distributions. The algorithm works well for any random distribution. It is of concern to test the algorithm for other geometric figures. If the initial distribution is in the shape of a polygon forming a circle from that distribution becomes easier. In figure 4-12 we show the initial distribution and the final formation.

Figure 4-12 shows various initial distributions and their final formations. We tried to select the initial distributions from which it will be difficult to make a circle. The speed of formation and the final formation differs. For instance it takes a long time for the formation to converge to a circle when the initial distribution is a line and the radius of final formation is smaller. If the initial distribution is closer to a line or in other words if the number of sides in the polygon are less it is harder to form a circle.

When the initial distribution is widely spread as in the case of figure 4-12(b) the final radius of the formation is large. When the initial distribution is clustered as seen in figure 4-12(a) and (b) the agents need to have a greater scanning range. If the range is small the clusters form different circles among themselves.
Simulation and Results

(a) Distribution in two clusters

(b) Distribution in four clusters

(c) All the agents looking in one direction
Simulation and Results

(d) Line with random orientation

(e) Line with one direction

(f) Two lines at a distant
Simulation and Results

4.8 Shrinking and growing circle

In section 2.13 we explained the possibility of making the circle to grow or shrink in radius uniformly. The circle can be made to grow or shrink by changing one parameter in the formation algorithm. By altering the angle $\theta$, to be made by each agent with its partnering agents the circle can be made to grow or shrink. If angle $\theta$ is made larger than the required interior angle of a regular polygon then the circle will shrink and if it is made...
smaller the circle will grow. We changed the interior angle $\theta$ by increasing or decreasing it by $\delta\theta$. The mean radius graph was plotted for different values of $\delta\theta$.

![Graph showing the mean distance from the centroid vs. number of iterations for different values of $\delta\theta$.](image)

**Figure 4-13 Radius of the growing circle for different values of $\delta\theta$**

Figure 4-13 shows the graph of the mean of the distances from the centroid of the colony for different values of $\delta\theta$. The initial distribution shown in figure 4-2 was initially made to form a circle and after convergence by decreasing $\theta$ by $\delta\theta$ the circle was made to grow. Here the step size was made to be large. When $\delta\theta$ is large each agent has to move a larger distance to make the inter-agent angle equal to $(\theta - \delta\theta)$. The distance to be moved also depends on the inter-agent distance. If the inter-agent distance gets larger, because of the growing circle, agents need to travel further to make the inter-agent angle equal to $(\theta - \delta\theta)$. This is the reason for the parabolic nature of the graph. If the step size is fixed to a value such as 0.01 the radius graph will be a straight line even for larger values of $\delta\theta$. 

7/21/04

Dept of Electrical Engineering
For the circle to shrink in radius $\delta \theta$ has to be added to $\theta$. The value of $\theta$ when $n = 50$ is 3.0159. Let us examine the shrinking circle when $\delta \theta$ is incremented by 0.02. Figure 4-14 shows the graph of shrinking circles. We cannot increase $\theta$ by a large margin since if we do so the agents will move two far into the circle and make it unstable.

![Graph of shrinking circles](image)

**Figure 4-14 Radius of the shrinking circle for different values of $\delta \theta$**

4.9 Circle with a particular radius

The idea of shrinking and growing circle can be used to form a circle of a particular radius. The colony of agents shown in figure 4-2 after forming the circle can be made to grow or shrink and stop at a particular radius. Each agent will know if the circle formed is bigger than the circle with the required radius or not by examining the inter-agent distance between its neighbors. If the inter-agent distances in larger than the inter-agent distance of the required circle to be formed then $\theta$ is incremented for the agents to move inwards. $\theta$ is made smaller if a larger circle is to be formed. Once the required
inter-agent distance is achieved growing or shrinking is stopped by getting back to the original $\theta$ for the $n$ sided polygon.

We simulated the distribution shown in figure 4-2 to form circles of radius 0.3 and 0.2, for shrinking circles and 0.6 and 0.8 for the growing circle. Table 4.1 shows the calculated result for the inter-agent distance using equation 2.32 and the results obtained through simulation. The simulation result of inter-agent distance was obtained by taking the mean of all the inter-agent distances. We also list the final radii of the circles formed in our simulations.

<table>
<thead>
<tr>
<th>Radius Required</th>
<th>Radius formed in simulation</th>
<th>Theoretical inter-agent distance</th>
<th>Simulation Inter-agent distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.3</td>
<td>0.3006</td>
<td>0.0377</td>
<td>0.0378</td>
</tr>
<tr>
<td>0.4</td>
<td>0.4008</td>
<td>0.0502</td>
<td>0.0504</td>
</tr>
<tr>
<td>0.5</td>
<td>0.4991</td>
<td>0.0628</td>
<td>0.0628</td>
</tr>
<tr>
<td>0.6</td>
<td>0.5990</td>
<td>0.0753</td>
<td>0.0753</td>
</tr>
</tbody>
</table>

Table 4-1 Theoretical and simulation values for the required radius and the corresponding inter-agent distance

Figure 4-15 shows the graph of the mean of the distance from the centroid of the colony and the final formation.
Simulation and Results

(a) Required radius = 0.3

(b) Required radius = 0.4

(c) Required radius = 0.5
Simulation and Results

Figure 4-15 Mean distance from the centroid and final formation for different required radii

4.10 Algorithm comparison

In this section we shall compare the algorithm we have developed and the existing circle algorithm developed by Suzuki et al. In figures 4-16 and 4-17 we show the difference between the two algorithms.

Figure 4-16 Distances from the centroid and the final formation in Suzuki’s algorithm
For the simulation of these algorithms we took the initial step as 0.025 for the first 15 steps and then it was made to be 0.01. Suzuki et al.’s algorithm has an unlimited span of scan and our algorithm uses a span of scan of 0.3 for each agent. Figure 4-16 and 4-17 show the distance from the centroid graph and the final formation after 100 steps for Suzuki et al.’s algorithm and our algorithm respectively.

The main advantage of our algorithm is that we use local scanning with information from only two partnering agents. Suzuki et al’s algorithm makes it necessary to scan all the agents in the terrain to find the furthest agent. The final formation as it can be seen is not a regularly distributed circle. Suzuki et al.’s algorithm does not deal with angles or direction of orientation of the agents. In our algorithm we make use of this information. They deal only with the inter-agent distances. The final direction of orientation of the agents is not known in their algorithm. In our algorithm, as we could see from the figure, the agents are uniformly looking outwards of the formation making the formation easy to grow or shrink.
Suzuki et al.’s algorithm does not work for some initial distributions. For certain random distributions, they state, the final formation is not a circle but in the form of Reauleaux’s triangle. Further we simulated their algorithm if the initial distribution was in the form of a line. The formation after 50 steps converges to two points instead of forming a circle. Our algorithm forms a circle as shown in figure 4-12(e).

![Graphs](image)

(a) Initial Distribution  (b) Final formation

**Figure 4-17** Simulation result of Suzuki et al.’s algorithm when the initial distribution is a line

The use of the inter-agent angle information for the functioning of the algorithm makes it necessary to have a hardware device that facilitates this capability. The algorithm is also computationally more complex than Suzuki et al.’s algorithm. Each agent needs to be able to compute the angle, the distance and the angle adjustment to make at each iteration. This involves the ability to do trigonometric and algebraic calculations given in equations 3.28, 3.29 and 3.31.
5. Conclusion and Future work

5.1 Conclusions

In this thesis we developed an algorithm for the formation of a circle by a group of autonomous mobile agents in an unknown terrain. The main advantages of this work in comparison to the other existing work are as follows.

1. Local sensing of the agents was used instead of global sensing.
2. Scanning all the agents in the whole terrain is done away with information from only two partnering agents.
3. The final formation is a perfectly distributed circle.
4. The algorithm works for any form of initial distribution resulting always in a circle.
5. By altering $\theta$, the inter-agent angle the agents are trying to make, the circle can be made to grow or shrink.

The basic algorithm is that each agent tries to make the inter-agent angle equal to $\theta$, where $\theta = \frac{(n-2)\pi}{n}$. To calculate $\theta$ each agent has to know $n$, the total number of agents in the colony.

The algorithm of making the inter-agent angle equal to $\theta$ has to be tweaked to overcome certain problems. The main problems were,

1. The uneven spreading of agents on the periphery of the circle due to large step size.
2. One or more than one agents get stuck in the middle of the circle.
The uneven spreading of the agent on the periphery of the circle can be eliminated if the step size is made small. When the step is small, obviously it takes a longer time for the agents to reach the periphery. This can be overcome by having a large step initially and changing to a smaller step. The point of change should ideally be when all the agents in the colony become external. We found that this happens around the 10th iteration. Using this technique we can make the colony to converge faster and form a precise evenly distributed circle.

The other problem of one or more agents getting stuck in the middle is caused when the span of scan does not cover the whole terrain. This problem can be overcome using the inter-agent distance. The partners of the agents that are stuck in the middle will be further away than a certain limit after a certain number of iterations. If an agent's partners are further than that limit it needs to turn by \( \pi \) radians to reach the periphery of the circle.

We also showed how the algorithm can be altered to make growing or shrinking circles by changing one parameter. This idea can also be used to make a circle of a desired radius.

5.2 Future Work

In this work we developed an algorithm and showed that it works through simulation. The algorithm depends a lot on randomness, but it is mathematically possible to predict the formation of the colony. For instance, it is possible to mathematically predict the radius of the circle or the location of the center to a certain approximation.
The algorithm developed in this thesis considers an agent to be a point. The dimensions of the agent were not taken into consideration. We have tried to minimize two agents taking the same position by one agent not completely traveling the whole distance. However there could be collisions while the robots are in motion. This has not been taken care of in our algorithm.

The real challenge will be to implement the algorithm practically. A practical way to measure the inter-robot angle, which was used for the first time to our knowledge in robotic formations, needs to be found. Using light sensors with a long tube on a rotor is a possibility. The ways of using sound waves or omni directional cameras need to be explored.

When the algorithm is employed on practical robots numerous issues will arise. The dimension of the robot and collision avoidance is a major concern. In real hardware there will be sensor and motor errors. These need to be addressed. Another issue is the robots turning any given angle. In a practical situation, if the wheels of the robots are turned by a certain angle then the robots will not travel in a straight line. The equations derived to reach the new position will have to be amended according to the specifications of the robot.

Initializing the colony to start the circle formation algorithm needs to be devised. An idea is to send out a signal strong enough to reach all the robots.

The algorithm termination also needs to be addressed. If the robots are asked to form a circle of a required radius then a robot would come to know about it by the inter-robot distances with its partners. After making that distance a robot can cease to function.
Maintaining the synchronicity is also an issue. If there are hardware delays and a robot cannot keep up with the rest of the colony it will create a problem. Works of Prencipe et al. [18-21] address the issues of asynchronicity. Ideas can probably be drawn from their work.
Reference


International Symposium on Theoretical Aspects of Computer Science, pp. 247-258, February 2001


Reference


Appendix A

Coordinate values and the direction of orientation for the distribution shown in figure 3.2

### x coordinate values

<p>| | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0.7020</td>
<td>0.8750</td>
<td>0.9862</td>
<td>0.8853</td>
<td>0.4048</td>
<td>0.6271</td>
<td>0.3855</td>
<td>0.8479</td>
<td>0.5268</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.8074</td>
<td>0.3935</td>
<td>0.9617</td>
<td>0.0301</td>
<td>0.9537</td>
<td>0.7144</td>
<td>0.6465</td>
<td>0.4467</td>
<td>0.1746</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.8352</td>
<td>0.9701</td>
<td>0.1350</td>
<td>0.2510</td>
<td>0.9100</td>
<td>0.6765</td>
<td>0.6232</td>
<td>0.5122</td>
<td>0.0035</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.2269</td>
<td>0.9785</td>
<td>0.8613</td>
<td>0.0144</td>
<td>0.4858</td>
<td>0.4164</td>
<td>0.7729</td>
<td>0.4881</td>
<td>0.5226</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.7820</td>
<td>0.5912</td>
<td>0.1264</td>
<td>0.1097</td>
<td>0.6629</td>
<td>0.9971</td>
<td>0.3462</td>
<td>0.1761</td>
<td>0.0679</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.3094</td>
<td>0.3348</td>
<td>0.3762</td>
<td>0.9522</td>
<td>0.7193</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### y coordinate values

<p>| | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0.7793</td>
<td>0.6177</td>
<td>0.6492</td>
<td>0.7563</td>
<td>0.1478</td>
<td>0.5995</td>
<td>0.8986</td>
<td>0.1719</td>
<td>0.8189</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0693</td>
<td>0.9557</td>
<td>0.3173</td>
<td>0.0052</td>
<td>0.7599</td>
<td>0.3087</td>
<td>0.7153</td>
<td>0.0809</td>
<td>0.8459</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.7184</td>
<td>0.8704</td>
<td>0.8722</td>
<td>0.7616</td>
<td>0.6695</td>
<td>0.9020</td>
<td>0.8215</td>
<td>0.8327</td>
<td>0.6399</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.1562</td>
<td>0.7832</td>
<td>0.7314</td>
<td>0.5647</td>
<td>0.7233</td>
<td>0.2397</td>
<td>0.4899</td>
<td>0.4236</td>
<td>0.7957</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.7634</td>
<td>0.2389</td>
<td>0.6351</td>
<td>0.2315</td>
<td>0.6159</td>
<td>0.2685</td>
<td>0.9912</td>
<td>0.7603</td>
<td>0.4822</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.9452</td>
<td>0.3607</td>
<td>0.0844</td>
<td>0.9485</td>
<td>0.6908</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### direction of orientation in radians

<p>| | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>3.8674</td>
<td>5.6233</td>
<td>0.0788</td>
<td>1.8907</td>
<td>6.0638</td>
<td>0.6415</td>
<td>3.7449</td>
<td>2.9467</td>
<td>4.5071</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.3964</td>
<td>1.1661</td>
<td>2.9052</td>
<td>5.6756</td>
<td>0.1389</td>
<td>4.7778</td>
<td>5.0005</td>
<td>1.9720</td>
<td>1.4817</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.1067</td>
<td>4.3056</td>
<td>5.9353</td>
<td>2.7603</td>
<td>3.1255</td>
<td>0.6196</td>
<td>3.4836</td>
<td>2.7375</td>
<td>1.1389</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.9664</td>
<td>5.8668</td>
<td>0.8562</td>
<td>6.2740</td>
<td>4.3393</td>
<td>6.0384</td>
<td>4.8726</td>
<td>0.0045</td>
<td>2.0450</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.4187</td>
<td>1.9258</td>
<td>4.2267</td>
<td>2.1722</td>
<td>6.1613</td>
<td>2.2240</td>
<td>0.1226</td>
<td>5.1067</td>
<td>6.1752</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.9593</td>
<td>2.3808</td>
<td>4.7000</td>
<td>2.7824</td>
<td>0.6276</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Appendix B

Circle formation algorithm code

clear all   
clf   
n=50; % # of robots   
x=rand(1,n); %global x coordinates   
y=rand(1,n); %global y coordinates   
dir=2*pi*rand(1,n); %global direction of orientation   
range=0.3; %span of scan   
step=0.1; %step size   

delta=0; %increment in Teta   
xE=zeros(1,n); %count of robots on the periphery   

Teta=((n-2)*pi)/n + delta; %angle to be made   
K=l/(2*sin(Teta/2)); %constant   

for m=1:100 %# of iterations   
  if m<15 % step size adjustment   
    step=step;   
  else   
    step=0.01;   
  end   

clf   
plot(x,y,'.r') %plot the positions of robots   
axis equal   
axis([-0.25 1.25 -0.25 1.25])   
for k=1:n %plot directions of orientation of robots   
  line([x(k) x(k)+0.1*cos(dir(k))],[y(k) y(k)+0.1*sin(dir(k))])   
end   

for i=1:n; %ith robot   
  r=1;   
  x11=x;   
  y11=y;   
  x11(i)=[];   
  y11(i)=[];   
  for q=1:n-1 %scan the robots under span of scan
Appendix

\[ \text{dis} = \sqrt{((x11(q)-x(i))^2 + (y11(q)-y(i))^2)^0.5}; \]

if \( \text{dis} < \text{range} \)
\[
\begin{align*}
    xr(r) &= x11(q); \\
    yr(r) &= y11(q); \\
    r &= r+1; \\
\end{align*}
\]
end
end

if \( r > 2 \)
\[
\begin{align*}
    \text{clear } x11, y11 \\
    x11 &= xr; \\
    y11 &= yr; \\
\end{align*}
\]
\[
\begin{align*}
    \text{beta} &= \text{atan2} \left( (y11-y(i)), (x11-x(i)) \right) - \text{dir}(i); \quad \% \text{inter-robot angle} \\
    \text{beta} &= \text{mod} \left( \text{beta}, 2\pi \right); \\
\end{align*}
\]
for \( k=1:(r-1) \) \% position in the local coordinate system
\[
\begin{align*}
    x0(k) &= \sqrt{((x11(k)-x(i))^2 + (y11(k)-y(i))^2)^0.5}\cos(\text{beta}(k)); \\
    y0(k) &= \sqrt{((x11(k)-x(i))^2 + (y11(k)-y(i))^2)^0.5}\sin(\text{beta}(k)); \\
\end{align*}
\]
end
\[
\begin{align*}
    [\text{betas}, p] &= \text{sort}(\text{beta}); \quad \% \text{arrange in the order of angle} \\
\end{align*}
\]
\[
\begin{align*}
    pL &= p(1); \quad \% \text{partner to the left} \\
    pR &= p(r-1); \quad \% \text{partner to the right} \\
\end{align*}
\]
\[
\begin{align*}
    x1 &= x0(pL); \quad \% \text{coordinates of the partners} \\
    y1 &= y0(pL); \\
    x2 &= x0(pR); \\
    y2 &= y0(pR); \\
\end{align*}
\]
\[
\begin{align*}
    dL &= (x1^2 + y1^2)^0.5; \quad \% \text{distance to the left partner} \\
    dR &= (x2^2 + y2^2)^0.5; \quad \% \text{distance to the right partner} \\
\end{align*}
\]
\[
\begin{align*}
    aL &= \text{beta}(pL); \quad \% \text{angle to the left partner} \\
    aR &= 2\pi - \text{beta}(pR); \quad \% \text{angle to the right partner} \\
    \text{alpha} &= (aR + aL); \quad \% \text{inter-robot angle} \\
\end{align*}
\]
\[
\begin{align*}
    D &= K \times (dL^2 + dR^2 - 2 \times dL \times dR \times \cos(\text{alpha} + \text{Teta}))^0.5; \quad \% \text{distance to move} \\
    \text{GAMMA} &= \text{atan2} \left( (\text{dR} \times \cos(\text{alpha} + \text{Teta}/2)) - (\text{dL} \times \cos(\text{alpha} + \text{Teta}/2)), (\text{dR} \times \sin(aR + \text{Teta}/2)) + (\text{dL} \times \sin(aL + \text{Teta}/2)) \right); \quad \% \text{angle to turn} \\
\end{align*}
\]
\[
\begin{align*}
    xn &= D \times \cos(\text{GAMMA}); \quad \% \text{new position} \\
    yn &= D \times \sin(\text{GAMMA}); \\
\end{align*}
\]
\[
\begin{align*}
    xm &= (dL \times \cos(aL) + dR \times \cos(aR))/2; \quad \% \text{midpoint of the partners} \\
    ym &= (dL \times \sin(aL) + (-dR \times \sin(aR)))/2; \\
\end{align*}
\]
EPSILON=atan2((yn-ym),(xn-xm)); %angle adjustment

if D>step %move less than step
    DD=step;
else
    DD=D;
end

if xE(i)==0
    dir(i)=(dir(i) + GAMMA); %switch back to global
    xU(i)=x(i)+ DD*cos (dir(i));
    yU(i)=y(i)+ DD*sin (dir(i));
    dir(i)=mod((dir(i) - GAMMA + EPSILON),2*pi); %angle adjustment
else
    xU(i)=x(i)+ DD*cos (dir(i));
    yU(i)=y(i)+ DD*sin (dir(i));
end

if m>50 %trapped robots getting back
    if (dL> (pi/n) || dR> (pi/n)) && xE(i)==0
        dir(i)=dir(i) + pi;
        xE(i)=1;
    end
    if dL < 0.1 && dR < 0.1
        xE(i)=0;
    end
end
else
    xU(i)=x(i);
    yU(i)=y(i);
    dir(i)=dir(i);
end

clear x11 y11 xr yr beta betas betadif x0 y0 a p
end

for s=1:n %avoiding two robots taking the same position
    samex=find(abs(xU(s) - xU(1:n))<10^(-2));
    [a p]=find(samex==s);
    samex(p)=[];
    if (length(samex))>0
        for j=1:length(samex)
            if abs(yU(s) - yU(samex(j)))<10^(-2)
                xU(samex(j))=(xU(samex(j)) + x(samex(j)))/2;
                yU(samex(j))=(yU(samex(j)) + y(samex(j)))/2;
            end
        end
    end
end
end
end
end
clear samex;
end

x=xU;
y=yU;
pause(0.1)
end