

INFORMATION THEORETIC SELF-ORGANIZATION OF MULTIPLE AGENTS

By

RAJVIGNESH THOGULUA CHANDRASEKAR

A THESIS PRESENTED TO THE GRADUATE SCHOOL  
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT  
OF THE REQUIREMENTS FOR THE DEGREE OF  
MASTER OF SCIENCE

UNIVERSITY OF FLORIDA

2003

Dedicated to the almighty

## ACKNOWLEDGMENTS

I am grateful to my advisor, Dr. Jose C. Principe, for his guidance throughout the completion of this work. He has been incredibly helpful with his ideas and suggestions. I admire him very much for providing me the freedom to think independently which has made me a better researcher today.

I would like to thank Dr. John G. Harris and Dr. Fred J. Taylor for serving in my committee and for their useful comments.

I am greatly indebted to Dr. Deniz Erdogmus for his valuable thoughts and helpful suggestions. He has been instrumental in initiating the core idea and also in its improvements. This work would not have been possible without him.

Thanks are due to my group members including Yadunandana Rao, Kenneth Hild II and Anant Hegde. The group meetings helped in enhancing the quality of the thesis.

My sincere thanks go to my parents for their constant emotional support throughout my academic career. I wish to thank all my friends including RoopKishore Matavalam and Kartikeya Tripathi and my brother, for being enthusiastic about my work and for providing me some interesting questions and comments.

## TABLE OF CONTENTS

|  | <u>Page</u> |
|--|-------------|
| ACKNOWLEDGMENTS .....  | iii         |
| LIST OF TABLES .....   | vi          |
| LIST OF FIGURES .....  | vii         |
| ABSTRACT .....   | ix          |
| CHAPTER  |             |
| 1 INTRODUCTION .....   | 1           |
| 1.1 Swarms .....   | 1           |
| 1.1.1 Potential and Future of Swarms .....                   | 1           |
| 1.1.2 Biological Background .....                            | 2           |
| 1.2 Multiple Agents .....                                    | 2           |
| 1.2.1 Basics .....   | 2           |
| 1.2.2 Applications .....                                     | 3           |
| 1.3 Control Strategies in Self-Organization .....            | 3           |
| 1.3.1 Superiority of Decentralized Control .....             | 3           |
| 1.3.2 Decentralized Self-Organization .....                  | 4           |
| 1.4 Goals and Possible Applications .....                    | 5           |
| 1.5 Organization of Thesis .....                             | 5           |
| 2 PARTICLE POTENTIAL FIELDS .....                            | 7           |
| 2.1 Generalized Information Particle Interaction Model ..... | 7           |
| 2.1.1 Nonparametric Estimation of Renyi's Entropy .....      | 8           |
| 2.1.2 Information Particles .....                            | 9           |
| 2.2 Robots as Information Particles .....                    | 11          |
| 3 SELF-ORGANIZATION ALGORITHM .....                          | 14          |
| 3.1 Literature Survey .....                                  | 14          |
| 3.2 Spreading Algorithm for a Circular Region .....          | 15          |
| 3.2.1 Threshold Estimation .....                             | 16          |
| 3.2.1.1 Sphere packing problem .....                         | 16          |
| 3.2.1.2 Empirical estimation of threshold .....              | 17          |
| 3.2.2 Algorithm Details .....                                | 18          |

|  |    |
|--|----|
| 3.2.3 Results .....  | 19 |
| 3.2.3.1 Spreading algorithm .....                          | 19 |
| 3.2.3.2 Uniformity analysis .....                          | 22 |
| 3.2.3.3 Velocity variation analysis .....                  | 23 |
| 3.2.3.4 Noise analysis .....                               | 24 |
| 3.2.4 Obstacle Avoidance .....                             | 25 |
| 3.3 Spreading Algorithm for Piecewise Linear Regions ..... | 28 |
| 3.3.1 Jordan Curve Theorem .....                           | 28 |
| 3.3.2 Results .....  | 29 |
| 3.4 Comparison with the Nearest Neighbor Technique .....   | 31 |
| <br>   |    |
| 4 COLLECTIVE TARGET TRACKING .....                         | 35 |
| <br>   |    |
| 4.1 Requirements and Suitable Modifications .....          | 35 |
| 4.2 Decentralized Target Tracking Algorithm .....          | 36 |
| 4.3 Results .....  | 38 |
| 4.3.1 Stationary Target .....                              | 38 |
| 4.3.2 Moving Target .....                                  | 39 |
| <br>   |    |
| 5 TOWARD REAL-TIME IMPLEMENTATION .....                    | 42 |
| <br>   |    |
| 5.1 Implementation Issues .....                            | 42 |
| 5.1.1 Communication Scheme .....                           | 42 |
| 5.1.2 Gradient Calculation .....                           | 45 |
| 5.1.3 Results .....  | 45 |
| 5.2 Webots Simulation .....                                | 47 |
| 5.2.1 Webots – An Introduction .....                       | 48 |
| 5.2.2 Environment and Robot Design .....                   | 49 |
| 5.2.3 Controller Programming .....                         | 52 |
| 5.2.4 Results and Discussion .....                         | 54 |
| <br>   |    |
| 6 CONCLUSION AND FUTURE WORK .....                         | 59 |
| <br>   |    |
| 6.1 Conclusion .....                                       | 59 |
| 6.2 Future Research .....                                  | 60 |
| <br>   |    |
| LIST OF REFERENCES .....                                   | 62 |
| <br>   |    |
| BIOGRAPHICAL SKETCH .....                                  | 65 |

## LIST OF TABLES

| <u>Table</u>  | <u>page</u> |
|---|-------------|
| 3.1 Parameters used in the spreading algorithm for circular region.....   | 21          |
| 3.2 Parameters used in the obstacle avoidance algorithm.....  | 26          |
| 3.3 Parameters used in the spreading algorithm for star shaped region.....  | 29          |
| 5.1 Parameters used to verify spreading algorithm with the suggested solutions for<br>implementation issues ..... | 46          |
| 5.2 Parameters used in Webots simulation.....   | 54          |

## LIST OF FIGURES

| <u>Figure</u>  | <u>page</u> |
|--|-------------|
| 2.1 Potential field created by information particles – top view.....                       | 10          |
| 2.2 Potential field created by the information particles – 3D view.....                    | 11          |
| 2.3 Illustration of information particles and the forces acting on them .....              | 12          |
| 3.1 Sphere packing problem – N denotes number of small circles.....                        | 17          |
| 3.2 Empirical estimation of threshold .....  | 18          |
| 3.3 Spreading of robots over a circular region.....  | 20          |
| 3.4 Trajectories of the robots while spreading over the circle .....                       | 21          |
| 3.5 Estimated pdf of the final position of robots using Parzen window estimator .....      | 22          |
| 3.6 Side view of the estimated pdf .....   | 23          |
| 3.7 Analysis of the spreading algorithm with variation in velocity .....                   | 24          |
| 3.8 Noise analysis for spreading algorithm.....  | 25          |
| 3.9 Obstacle Avoidance.....  | 27          |
| 3.10 Jordan curve theorem .....  | 28          |
| 3.11 Spreading of robots over a star shaped region. ....                                   | 30          |
| 3.12 Spreading of robots over different regions.....                                       | 31          |
| 3.13 Comparison of speed of spreading (N = 10) .....                                       | 33          |
| 3.14 Comparison of speed of spreading (N = 30) .....                                       | 33          |
| 3.15 Comparison of speed of spreading (N = 50) .....                                       | 34          |
| 4.1 Technique for collective target tracking.....  | 37          |
| 4.2 Collective target tracking of a stationary target with robots in circular formation .. | 38          |

|     |  |    |
|-----|--|----|
| 4.3 | Collective target tracking of a stationary target with robots in star formation..... | 39 |
| 4.4 | Collective target tracking of a moving target with robots in circular formation..... | 40 |
| 5.1 | Spreading Speed with the use of M sequence and rectangular grid antenna .....        | 46 |
| 5.2 | Noise analysis with the use of M sequence and rectangular grid antenna .....         | 47 |
| 5.3 | Designed environment and robots in Webots .....                                      | 50 |
| 5.4 | Basic environment design .....   | 51 |
| 5.5 | Field values for the DESIREDCIRCLE node.....   | 52 |
| 5.6 | Field values for the ROBOT1 DifferentialWheels node.....                             | 53 |
| 5.7 | Webots simulation on spreading of robots.....  | 55 |

Abstract of Thesis Presented to the Graduate School  
of the University of Florida in Partial Fulfillment of the  
Requirements for the Degree of Master of Science

INFORMATION THEORETIC SELF-ORGANIZATION OF MULTIPLE AGENTS

By

Rajvignesh Thogulua Chandrasekar

August 2003

Chair: Jose C. Principe

Major Department: Electrical and Computer Engineering

This study intends to use the information theoretic interaction principle for self-organization of multiple agents. The impetus of this work comes from the recent increase in the application of swarm intelligence to versatile fields like defense, space and communication networks. One of the important problems in this exciting area is the coordination and control of individual agents in the swarm to produce desired collective behavior. Most of the available solutions for this problem are either heuristic or based on centralized approaches. Centralized approaches, often referred to as the leader-follower strategy, may not be applicable in many real-life situations. It may also lead to frailty in the overall system and increased costs.

In this work, a decentralized approach to solve this problem using an information theoretic interaction between the agents is suggested. This technique uses entropy maximization to spread the agents uniformly over a particular region of interest and also to guide the collective towards a target. By minimizing the communication between the

robots, the proposed algorithm minimizes the total cost and hardware required for its operation.

## CHAPTER 1 INTRODUCTION

### 1.1 Swarms

#### 1.1.1 Potential and Future of Swarms

Union is strength. This is clearly evident in the behavior of pack hunters, right from the velociraptors species of the Mesozoic era to many species of dogs and wolves of the present era. It seems now that the world is going to understand the above saying in a better way from swarms, a few years down the lane. As quoted in the famous novel *Prey* by Michael Crichton [Cri02], Farmer and Belin say,

Within fifty to a hundred years, a new class of organisms is likely to emerge. These organisms will be artificial in the sense that they will be designed by humans. However, they will reproduce, and will evolve into something other than their original form; they will be alive under any reasonable definition of the word. These organisms will evolve in a fundamentally different manner. ... The pace...will be extremely rapid. ... The impact on humanity and the biosphere could be enormous, larger than the industrial revolution, nuclear weapons, or environmental pollution. We must take steps now to shape the emergence of artificial organisms. [Far92, p. 815]

The power of swarms and numerous interesting avenues they offer are already creating waves in the scientific community. A new field coined as molecular robotics is emerging which combines swarm intelligence with another equally revolutionary idea – nanotechnology. Research in this field is mainly on making nanoswarms with each robot in the swarm measuring only nanometers. An article in Unisci [Cal02] says “The University of Southern California School of Engineering has received a \$1.5 million research grant from the National Science Foundation to create swarms of microscopic robots to monitor potentially dangerous microorganisms in the ocean.” It is clear now that

the researchers no more talk about “Artificial Intelligence” and the new term “Artificial Life” rules the roost in the recent conferences.

### **1.1.2 Biological Background**

Swarm Intelligence is biologically inspired by insect societies, where “complex collective behavior may emerge from interactions among individuals that exhibit simple behavior.” [Bon99, p. 6] Some examples of this collective activity in insects are as listed below.

- In Leafcutter ants (*Atta*), chains of ants work cooperatively in bringing cut leaves obtained far away, back to the nest.
- Hunting in army ants (*Eciton*) involves hundreds of thousands of workers.
- Wasps (*Parachartergus*, *Epipona*) and Termites (*Macrotermes*) can build very complex nests comprising concentric walls and ventilation ducts.

## **1.2 Multiple Agents**

The word “swarms” is used more in biological sense as seen in the papers on insect societies. But in the literature on artificial life, it is used interchangeably with another phrase – multiple agents. Hence, in this thesis too, although these two words mean the same, the more appropriate one will be used according to the context.

### **1.2.1 Basics**

Agents can be defined as man-made entities, which can perform particular tasks. Agents can be robots, as referred in this work and may also be software programs. The following are few reasons for the need of multiple agents [Kub00]:

- Multiple agents are necessary to perform few tasks, which are impossible for a single robot. Robot soccer and box pushing can be good examples.
- The individual agents may be easier to design and hence, using multiple agents may be more cost-efficient than using one complex agent.
- Multiple agents are more flexible and robust compared to a single agent.

### 1.2.2 Applications

Multiple agents are being applied to varied problems in versatile fields. Some of them are as follows:

- White and Pagurek [Whi98] have used multiple interacting swarms inspired by the foraging activities of ants to solve problems in networks.
- In oceanographic sampling networks, a group of agents have been applied to cooperatively sample a region over a particular period of time [Tur98].
- Miller [Mil90] has proposed a swarm of micro-robots for carrying out planetary missions.
- Genovese *et al.* [Gen92] have devised an architecture for a pack of miniature-robots to search pollutants in a hazardous zone.
- Cooperative material transportation by robots has been studied with centralized control [Sti93] and decentralized control [Kub00].

### 1.3 Control Strategies in Self-Organization

Two types of control strategies can be applied for the self-organization of multiple agents, namely centralized control and decentralized control. Centralized control is also called the leader-follower approach.

#### 1.3.1 Superiority of Decentralized Control

Although many researchers have attempted self-organization with centralized control, it is widely accepted now that decentralization is vital since it brings robustness to the system. The various reasons that speak for decentralized approach are

- Having a central controller may not be feasible in some tasks like mine sweeping, since the failure of the controller means failure of the whole system.
- High swarm population would necessitate the controller to have extensive communication with agents.
- The complexity of the controller will make it costlier than having many simple robots.

It seems that the choice of decentralized control also has some biological reasoning behind it. In his famous book called *Turtles, Termites and Traffic Jams*, Mitchel Resnick states

A flock of birds sweeps across the sky. Like a well-choreographed dance troupe, the birds veer to the left in unison. Then, suddenly, they all dart to the right and swoop down toward the ground. Each movement seems perfectly coordinated. The flock as a whole is as graceful—maybe more graceful—than any of the birds within it. Most people assume that birds play a game of follow-the-leader: the bird at the front of the flock leads, and the others follow. But that's not so... There is no special "leader bird." Rather, the flock is an example of what some people call "self organization." Each bird in the flock follows a set of simple rules, reacting to the movements of the birds nearby it. Orderly flock patterns arise from these simple, local interactions. None of the birds has a sense of the overall flock pattern... and... Bird flocks are not the only things that work that way. Ant colonies, highway traffic, market economies, immune systems – in all of these systems, patterns are determined not by some centralized authority but by local interactions among decentralized components. [Res97, p. 3]

Thus, the importance of decentralized control and its connection to self-organization can be understood.

### **1.3.2 Decentralized Self-Organization**

Here, a simple and effective way of bringing decentralized self-organization in multiple agents through an Information Particle Interaction Model (IPIM) is proposed. IPIM is based on the theory of physically interpreting data samples in entropy estimation as information particles (IPCs) and the interactions between data samples as information forces (IF) [Pri00]. This theory can be applied to the self-organization of multiple agents by considering each robot as an IPC interacting with each other in a potential field [Tho03]. Since this requires each robot to have only a simple transmitter and receiver, it is shown here that this technique greatly reduces the circuit complexity and hence the cost of the whole application.

### **1.4 Goals and Possible Applications**

This work tries to achieve the following goals through the application of IPIM to the decentralized self-organization of multiple agents:

- To distribute the multiple agents uniformly over a region of any shape.
- To move the agents collectively towards a particular target.

This technique has good potential to be applied for problems in various fields. Material transportation of huge components requires the spreading of agents uniformly over a region so that all the agents can share the weight evenly. Once the object is placed, the agents need to move cooperatively keeping the formation intact towards a destined location. A simple military application is the bombing of a particular region with multiple agents. Each robot can carry a bomb and spread themselves over the whole region and then simultaneously detonate their bombs. Another interesting application is that of air surveillance of enemy targets [Cri02]. A uniformly spread swarm of multiple agents with each agent carrying a small lens can transmit back their observations to a base station where the individual pictures can be made into one large picture of the whole region. Furthermore, the swarm cannot be shot down since they can easily disperse and re-unite. This application can be slightly modified for use in airborne traffic surveillance for large cities. The proposed technique can be of huge interest in space missions with multiple agents where no GPS is available for giving location information.

### **1.5 Organization of Thesis**

Chapter 2 explains the background of the proposed technique, describing IPIM and the possibility of applying it to self-organization of multiple agents. Chapter 3 deals with the first goal of spreading the agents uniformly over any region. Comparisons with the nearest neighbor technique are made wherever possible. A small introduction to obstacle

avoidance is also given. Chapter 4 describes the collective target tracking algorithm. Chapter 5 is a collection of implementation issues and suggested solutions. The simulation in Webots, software that can be used as an intermediate platform before the actual implementation of the whole system, is explained. Chapter 6 concludes the thesis and gives some future directions.

## CHAPTER 2 PARTICLE POTENTIAL FIELDS

The idea of self-organization in distributed systems through pair-wise interactions of particles emitting potential energies has its origin in the work done by Principe *et al.* [Pri00] on the Information Particle Interaction Model (IPIM). IPIM gives a physical interpretation of the process for finding Renyi's quadratic entropy by which any learning process can be seen as an interaction of the samples, considered as particles interacting in a potential field. IPIM has been successfully applied in many problems including independent component analysis (ICA), nonlinear principal component analysis (nonlinear PCA) and SAR image feature extraction. Generalized Information Particle Interaction Model (GIPIIM) is the generalization of IPIM, primarily developed for learning and system adaptation [Erd02]. Self-organization algorithm can be derived as a special case in GIPIIM, corresponding to a specific choice of particle potential function (elaborated later).

### **2.1 Generalized Information Particle Interaction Model**

The quest by Principe *et al.* [Pri00] for finding better cost functions than mean square error (MSE) led to the use of Renyi's quadratic entropy as an adaptation criterion in optimization problems. This inspired the emergence of a dynamic new branch in adaptive systems, termed Information Theoretic Learning (ITL). In ITL, Renyi's quadratic entropy was combined with nonparametric estimation of probability density functions (pdf) using Parzen windows, to develop a highly effective algorithm for computing entropy as described below.

### 2.1.1 Nonparametric Estimation of Renyi's Entropy

Renyi's entropy is a parametric family defined by [Ren70]

$$H_\alpha(X) = \frac{1}{1-\alpha} \log \int_{-\infty}^{\infty} f_X^\alpha(x) dx \quad (2.1)$$

where  $X$  is a random variable with the designated marginal and joint pdf, and  $\alpha$  is the order parameter. The entropy definition in (2.1) can also be written as,

$$H_\alpha(X) = \frac{1}{1-\alpha} \log E[f_X^{\alpha-1}(X)] \quad (2.2)$$

To find the pdf  $f_X(X)$  in the above equation, the famous non-parametric estimation using Parzen windowing is utilized. The Parzen window estimator [Par67] for the pdf of the data samples  $p_i, i=1,2,\dots,N$ , is evaluated using a kernel function  $\kappa_\sigma(\cdot)$ , where  $\sigma$  is a parameter that controls the width of the kernel function.

$$\hat{f}_X(x) = \frac{1}{N} \sum_{i=1}^N \kappa_\sigma(x - p_i) \quad (2.3)$$

In the multidimensional pdf estimation case, this can be a vector or the covariance matrix of the kernel function. In general, use of joint kernels of the type

$$\kappa_\Sigma(x) = \prod_{o=1}^n \kappa_{\sigma_o}(x^o) \quad (2.4)$$

where  $x^o$  is the  $o^{th}$  component of the input vector has been suggested. Replacing the expected value in (2.2) by the sample mean, the following nonparametric estimator for Renyi's entropy [Erd02] becomes,

$$H_\alpha(X) \approx \frac{1}{1-\alpha} \log \frac{1}{N^\alpha} \sum_{j=1}^N \left( \sum_{i=1}^N \kappa_\sigma(p_j - p_i) \right)^{\alpha-1} \quad (2.5)$$

This nonparametric estimator allows the designer to choose any entropy order  $\alpha$  and any kernel function  $\kappa_\sigma$ . For the special choices of quadratic entropy ( $\alpha = 2$ ) and Gaussian kernels, equation (2.5) reduces to the estimator defined by Principe *et al.* [Pri00] except for a change in kernel size.

$$H_2(X) \approx -\log V_2(X)$$

$$V_2(X) = \frac{1}{N^2} \sum_{j=1}^N \sum_{i=1}^N G_{\sigma\sqrt{2}}(p_j - p_i) \quad (2.6)$$

It is interesting to point out that the above definition is achieved without any sample approximations as in (2.5) due to the mathematical properties of the Gaussian kernel. It must also be noted here that in contrast to Shannon's entropy, Renyi's entropy is much easier to estimate directly from the data samples.

### 2.1.2 Information Particles

The definition of Renyi's quadratic entropy as given in (2.6), has a very interesting physical interpretation. For simplicity, let us assume that we are dealing with single dimensional random variables, with extension to multi-variable case being trivial. If the data samples  $p_i$  are assumed as physical particles, each emanating a potential field, then they can be called as information particles (IPCs) [Pri00]. Then,  $V_2(X)$  in (2.6) can be viewed as the total potential energy due to the pair-wise interactions of the IPCs.

If the potential field that is generated by each IPC is  $v(\xi)$ , then this function is required to be continuous and differentiable (except possibly at the origin), and to satisfy the even symmetry condition  $v(\xi) = v(-\xi)$ . With these definitions, it can be observed that the potential energy or information potential (IP) of particle  $p_j$  due to particle  $p_i$  is  $V(p_j, p_i) = v(p_j - p_i)$ . The total potential energy of  $p_j$  due to all other particles is then given by

$$V(p_j) = \sum_{i=1, i \neq j}^N V(p_j, p_i) = \sum_{i=1, i \neq j}^N v(p_j - p_i) \quad (2.7)$$

Figure 2.1 shows the potential field created by a few IPCs. We can see that the peaks of the field are mostly at the positions of the IPCs. A three dimensional view of the same is provided in Figure 2.2 for better perception.

In analogy to physics, the information force (IF) between these particles can be defined as follows:

$$\begin{aligned} F(p_j, p_i) &= \partial V(p_j, p_i) / \partial p_j \\ &= \partial v(\xi) / \partial \xi \Big|_{\xi=(p_j - p_i)} = v'(p_j - p_i) \end{aligned} \quad (2.8)$$

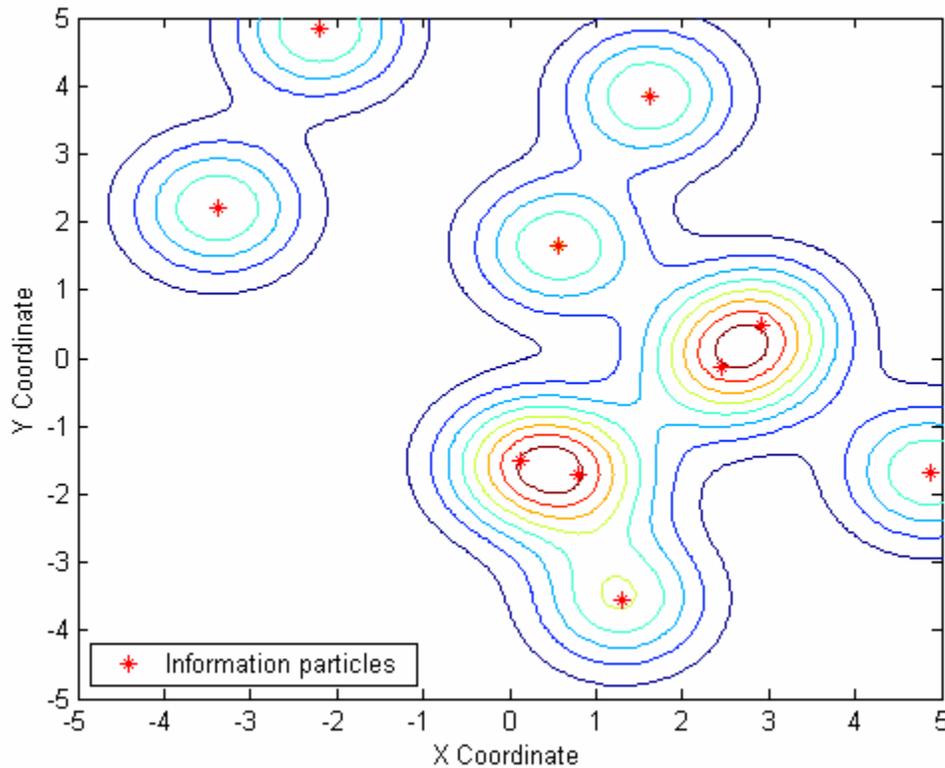


Figure 2.1 Potential field created by information particles – top view

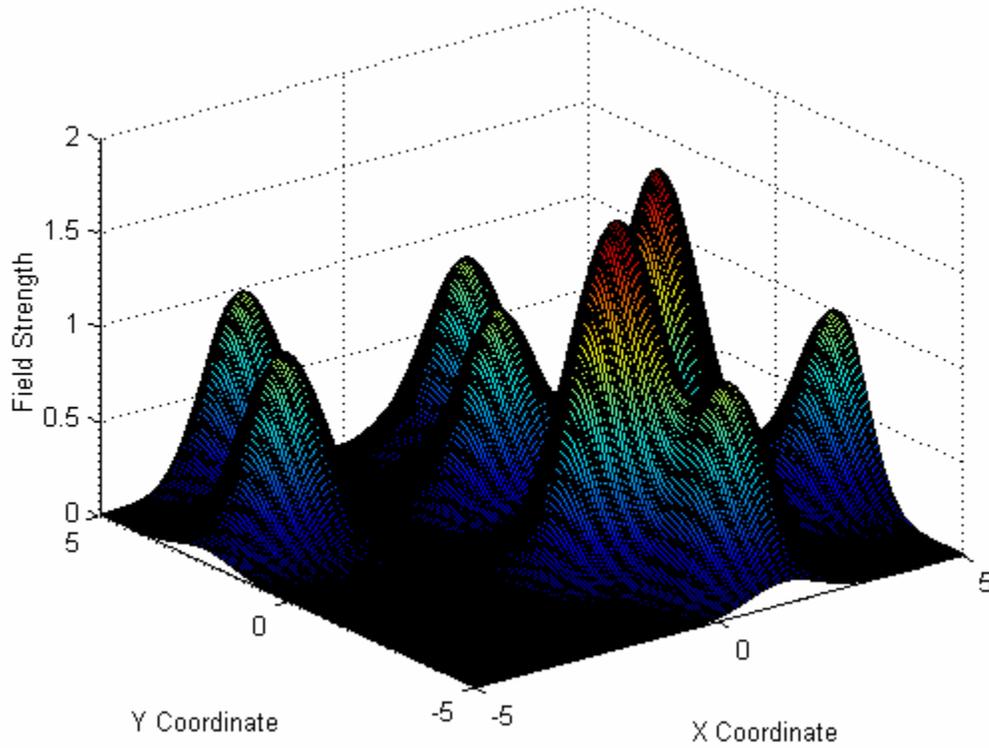


Figure 2.2 Potential field created by the information particles – 3D view

Then, the total force acting on particle  $p_j$

$$F(p_j) = \sum_{i=1, i \neq j}^N F(p_j, p_i) = \sum_{i=1, i \neq j}^N v'(p_j - p_i) \quad (2.9)$$

Notice that the force applied to a particle by itself is zero. A snapshot showing the IPCs and the information force acting on them is shown in figure 2.3.

## 2.2 Robots as Information Particles

The IPCs can be assumed as the robot position coordinate vectors in the current application. The information force, which is the gradient of the information potential with respect to the particle position, can then be immediately utilized as a command signal for the self-organization of agents.

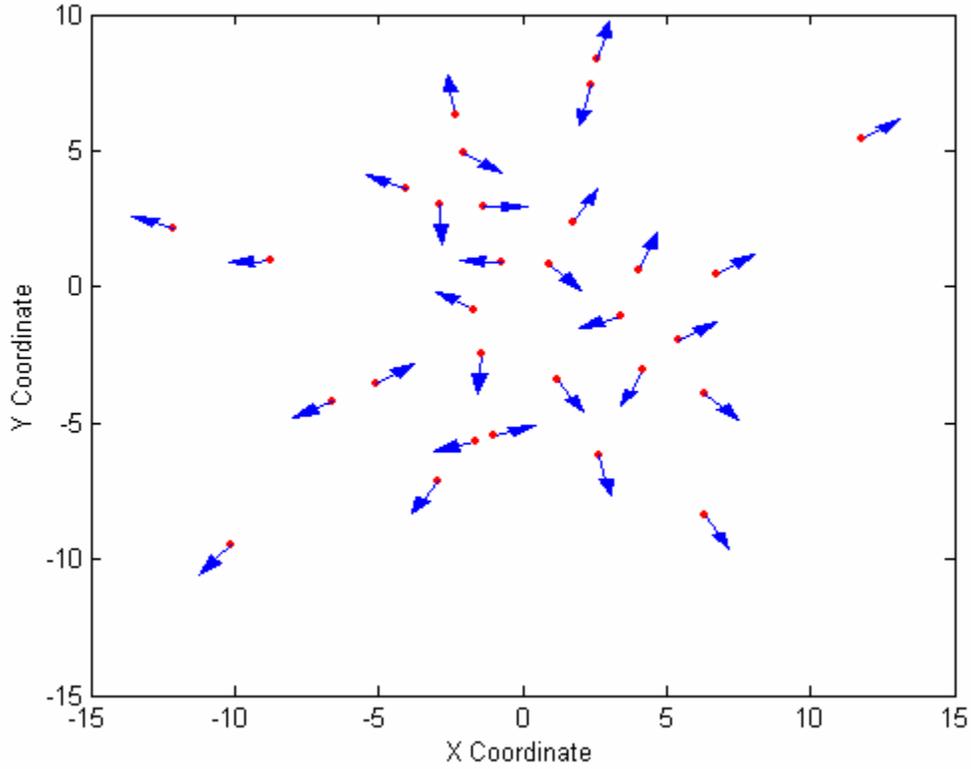


Figure 2.3 Illustration of information particles and the forces acting on them

Here, we assume that each robot in the swarm transmits an RF signal with certain amplitude and frequency. This signal may also be coded in order to improve the robustness of the system under noise and hostile jamming. Since the amplitude of any RF signal decays with the square of the distance, the potential field emanated by each particle is assumed as

$$v(\xi) = \frac{A}{\xi^T \xi} \quad (2.10)$$

Note that the above assumption satisfies all the properties of a potential field. Further, the fact that the above potential function is physically realizable motivates its use in this application. It is possible to show that maximization of entropy in a fixed region can be achieved using the above formulation [Erd02]. Since the maximization of entropy

leads to a uniform distribution of the particles, this principle can be effectively used to spread the agents uniformly over a particular region [Pri00].

## CHAPTER 3 SELF-ORGANIZATION ALGORITHM

In this chapter, the self-organization algorithm based on information theoretic interactions of IPCs is discussed. The self-organization algorithm is primarily used to distribute robots uniformly over a particular region. A literature survey on spreading of robots over a region or in a specific shape is presented in the next section.

### **3.1 Literature Survey**

In the field of multiple mobile robots, many researchers have attempted to arrange the robots in a particular shape or formation [Zha03]. This later led to the study of spreading of robots over a particular region. Fredslund and Mataric' [Fre02] have tried to make many interesting formations like line, wedge using an algorithm which asks each robot to keep a single friend at a desired angle and distance. They have used only local sensing in robots and have studied the stability and robustness issues in making the formations. The algorithm is very specific to each and every formation and it actually shows the basic AI approach for multiple robot formations.

Sugihara and Suzuki [Sug90] have used the distance from nearest and farthest robots to make circular and polygonal formations. It turns out that the circular formation is not accurate enough and most of the times lead actually to a Reuleaux triangle formation. Their technique uses relative instead of the absolute positions of the robots. Chen and Luh [Che94] continued this work and came out with a version that brings a more circular formation. They also studied obstacle avoidance of such formations.

Unsal and Bay [Uns94] used a goal beacon to arrange the robots in a circular fashion with the beacon as the center. This method eliminated the need to know the distance from the farthest robot and used only the nearest neighbor distance to achieve the same. They further modified their algorithm to distribute robots uniformly over a circular region, but still with the central beacon. A central beacon can be a realistic assumption in cases like material transportation but may not be feasible in certain military applications.

Batalin and Sukhatme [Bat02] discuss the deployment of a mobile sensor network over a region with the task of maximizing the sensor coverage. Two approaches namely informative and molecular have been discussed. Informative approach assigns local identities to robots with location information exchanged between them, where as molecular approach makes robots move away from their nearest neighbors.

Hence, it can be seen that most of the work is concentrated on using either nearest neighbor(s) technique or some kind of identification among robots for spreading them. The use of information theoretic interactions eliminates the need for identification of nearest robot(s) or farthest robot but still uses information from all the robots in a very simple way to spread them.

### **3.2 Spreading Algorithm for a Circular Region**

In most of the publications mentioned above, it is attempted to spread robots uniformly over a circular region or in a circular formation [Uns94]. This is because a circle has a perfect symmetry making it the simplest region to assume. Hence, the self-organization algorithm is used to spread robots uniformly over a circular region at first. The algorithm is later modified to spread robots over regions of any shape.

In the spreading algorithm explained here, the basic assumption is that all the robots are within the region at first and the total formation expands from inside the circular region. Hence, a controlling force at the boundary of the region is required to prevent the robots from moving outside the fixed region. When the region is a circle, it is easily possible, again due to its symmetry, to have a threshold for IF, which can act as a force at the boundary. The threshold, thus, eliminates the need to have any extrinsic force applied at the boundary, an unrealistic assumption in some applications.

### **3.2.1 Threshold Estimation**

As the robots are spreading themselves inside a circular region, the threshold is the total IF experienced by a robot from all other robots, when it reaches the boundary. Hence, it becomes necessary to know, given a particular number of robots, their most uniform distribution over the circle. Spreading a particular number of points over a disk is a complex optimization problem and is difficult to solve.

#### **3.2.1.1 Sphere packing problem**

It was thought that the problem at hand might be equivalent to the famous sphere packing problem. In sphere packing, a particular number of small circles are distributed uniformly over a bigger circle with the goal of maximizing the radius of the small circles [Spe02]. This enables us to consider the center of the small circles as the positions of the robots. Further investigation revealed that this solution does not give the most uniform distribution of robots possible. This can be explained better with figure 3.1. For a particular number of circles ( $N = 30$ ), the centers have been distributed uniformly. But in other cases ( $N = 31$  and  $32$ ), it is clear that the centers of the circles are not uniformly distributed.

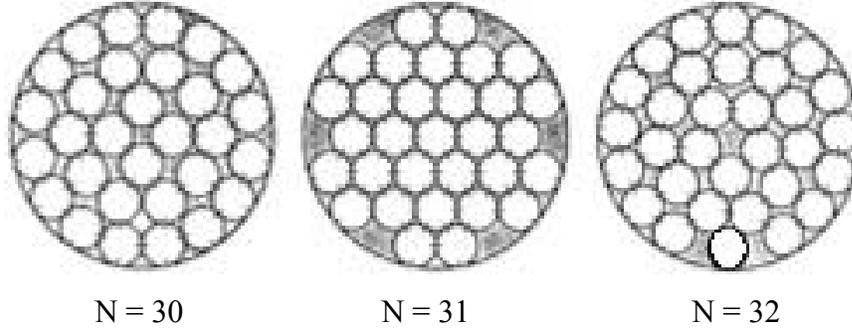


Figure 3.1 Sphere packing problem – N denotes number of small circles

### 3.2.1.2 Empirical estimation of threshold

Since an analytical expression for threshold is very difficult to find, the threshold was empirically estimated as follows. The self-organization algorithm was applied on a particular number of robots, which were initially positioned very close to each other. The IF between the robots results in an expansion of the formation of robots. At every time step, the approximate radius of the formation was calculated as the mean of the distance from the center of the formation to all robots, which form the outer boundary of the formation. This was continued till the estimated radius is equal to unity. Then the threshold was calculated as the mean of IFs experienced by all the robots at the boundary. The number of robots was varied and the process was repeated every time. A general expression for threshold for any number of robots was then found by fitting an exponential curve to the threshold data obtained from the above process, as shown in figure 3.2.

The threshold for any radii can then be obtained just by dividing the above found threshold for unit radius by the square of the radius. Thus, the general expression for the threshold, for a particular number of robots  $N$  and radius  $r$  is

$$\gamma = 0.21128 \cdot A \cdot N^{1.5107} / r^2 \quad (3.1)$$

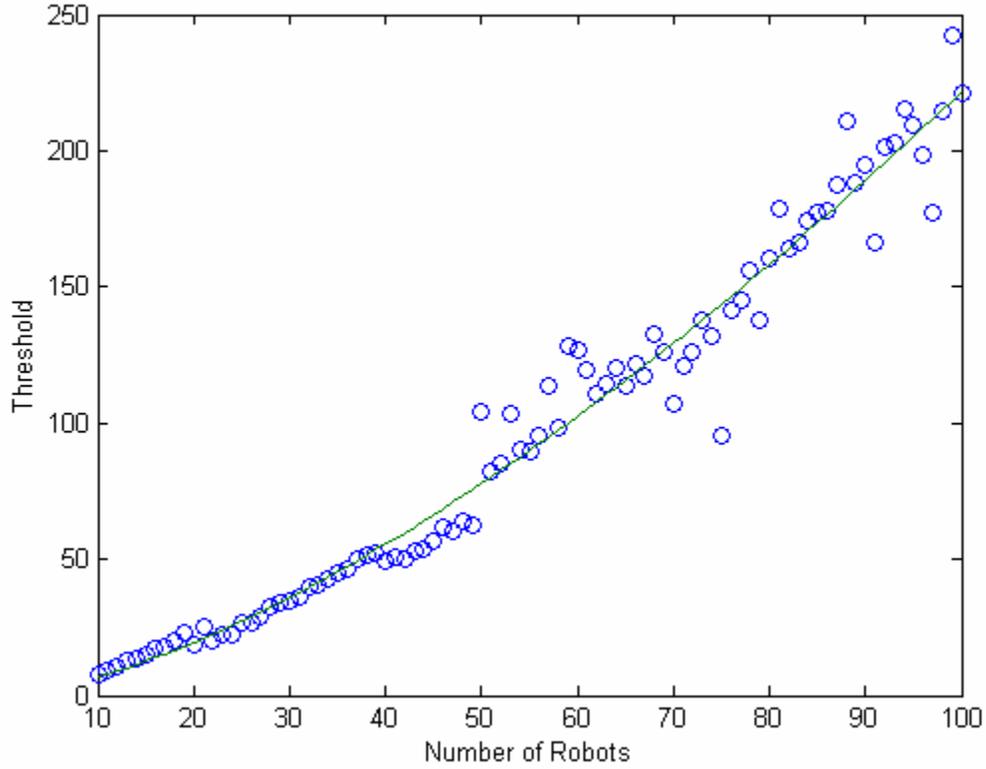


Figure 3.2 Empirical estimation of threshold

In figure 3.2, we can see that the variance of actual values from the approximation increases as the number of robots increase. This can be attributed to the fact that the simulation was run for the same number of iterations irrespective of the number of robots and hence the accuracy of the estimated thresholds decreases as the number of robots increase.

### 3.2.2 Algorithm Details

Let us assume that the amplitude of the received signal by  $i^{\text{th}}$  robot from  $k^{\text{th}}$  robot is  $A_{ik}$ . For a spherical propagation model, we have  $A_{ik} = A/d_{ik}^2$ , where  $d_{ik}$  is the distance between the  $i^{\text{th}}$  and  $k^{\text{th}}$  robots. From now on, it is assumed that  $A = 1$  for simplicity. The

total information potential  $V_i$  for the  $i^{\text{th}}$  robot is the sum of the received signal amplitudes  $A_{ik}$  from all other robots.

$$V_i = \sum_{k=1, k \neq i}^N A_{ik} = \sum_{k=1, k \neq i}^N \frac{1}{d_{ik}^2 + \sigma} \quad (3.2)$$

where  $\sigma$  is a regularity constant to avoid  $V_i$  peaking to infinity when  $d_{ik} \approx 0$ .

This is compared to the threshold calculated according to the desired radius to specify the sign on the direction of the IF, which is the gradient of the potential. Approximating the discontinuous *sign* function with *arctan*, the expression for the IF for the  $i^{\text{th}}$  robot becomes

$$F_i = \arctan(\gamma - V_i) \frac{\partial V_i}{\partial p_i} \quad (3.3)$$

### 3.2.3 Results

The application of IPIM for spreading of robots eliminates the need for them to know their positions explicitly. But many implementation issues like the mode of communication as well as the way to find the gradient for IF must be formulated for the simulation of the algorithm completely. Hence, the results presented in this section were obtained by transmitting relative position information between robots. Transmitting the relative positions instead of absolute positions allows the center of the circle to vary according to the initial position of the robots.

#### 3.2.3.1 Spreading algorithm

Figure 3.3 shows the spreading of robots starting from their initial positions to the final positions. Figure 3.4 shows the trajectories of all the robots. It can be seen that the spreading is almost uniform at the end and the robots form approximately concentric circles around the center of the circle. It can also be observed that the radius of the

formation is a little more than the desired radius of 1. This is because of the inaccuracies in the estimated threshold  $\gamma$ .

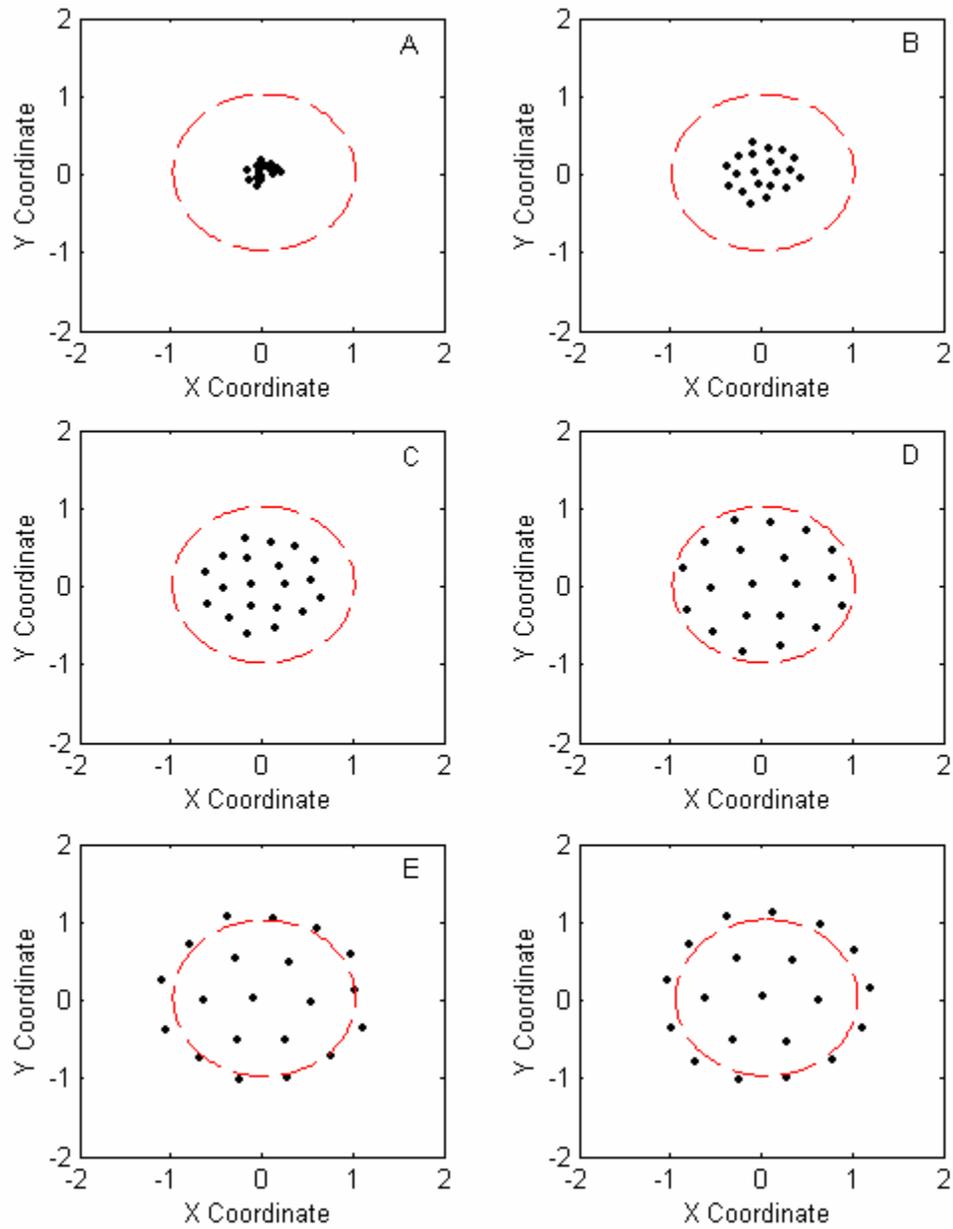


Figure 3.3 Spreading of robots over a circular region. A) Initial position of robots. B – D) Spreading of robots. E and F) Robots reaching the boundary and the group getting stabilized.

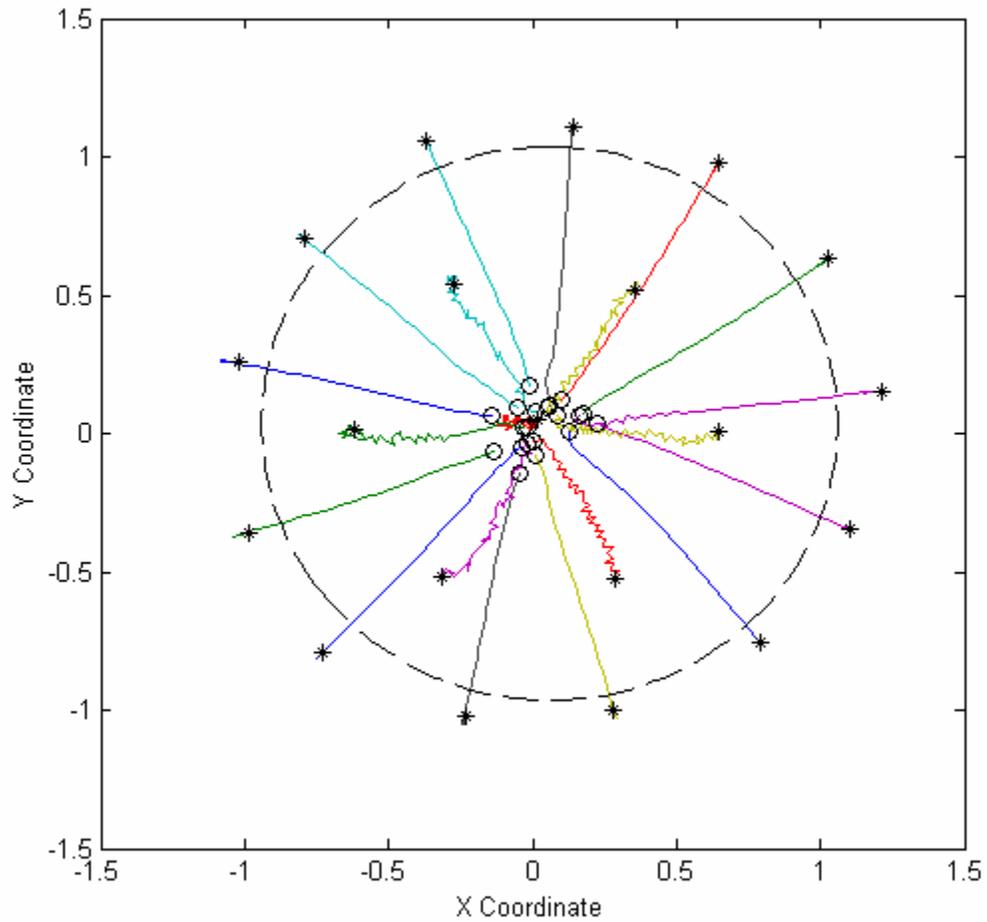


Figure 3.4 Trajectories of the robots while spreading over the circle

The various parameters used in the above simulation are listed in table 3.1. Note that the unit length is assumed as one meter and unit time is assumed as one second.

Table 3.1 Parameters used in the spreading algorithm for circular region

| Parameter              | Value  |
|------------------------|--------|
| Number of robots       | 20     |
| Radius of the circle   | 1 m    |
| Number of iterations   | 40     |
| Time step              | 0.03 s |
| Velocity of the robots | 1 m/s  |

### 3.2.3.2 Uniformity analysis

To measure the uniformity of the distribution, the pdf of the final position of robots is estimated non-parametrically using the Parzen window estimator [Par67]. The estimated pdf of the distribution is shown in figure 3.5. The side view of the same is shown in figure 3.6 to display the variation in the pdf in better light. From these two figures, we can clearly see that the distribution is almost uniform over the circular region. In the simulation, the variance of the Gaussian distribution used in the estimation was fixed as 0.1. The variance of the estimated pdf over the circular region was calculated and found to be  $3.0196e-004$ , which is very small justifying the uniformity of the final distribution of robots.

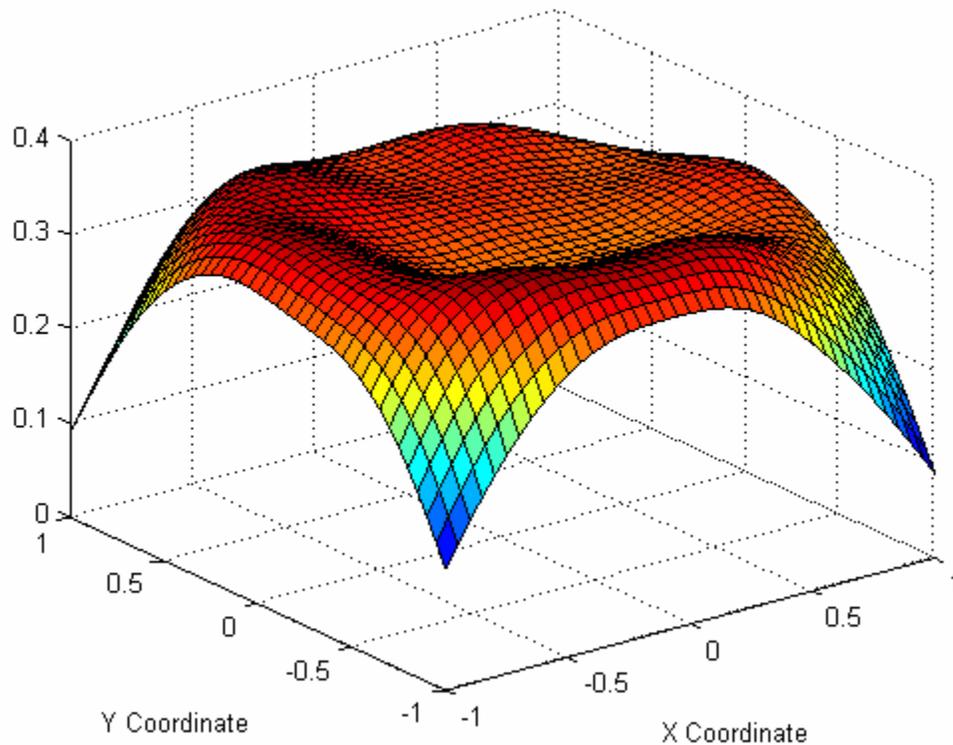


Figure 3.5 Estimated pdf of the final position of robots using Parzen window estimator

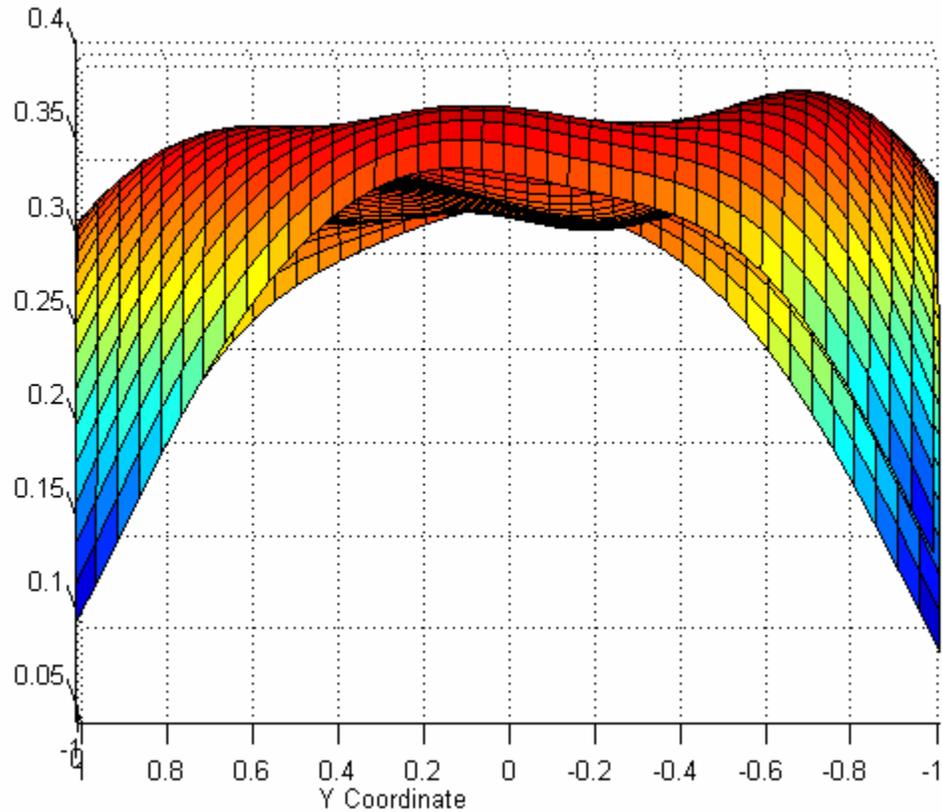


Figure 3.6 Side view of the estimated pdf

### 3.2.3.3 Velocity variation analysis

In the listed parameters, unit velocity means that every robot will move unit distance in unit time. However, the velocity can be varied to see the time the algorithm takes to spread over the circular region. In the following simulation, the velocity was varied from 0.01 to 0.1 and the spreading algorithm was allowed to execute till the radius of the formation becomes almost constant, which is desired to be unity. For every velocity, the spreading algorithm was executed for many runs to improve accuracy in the calculations. The number of robots was varied from 10 to 30 and 50 and the above process was repeated. The result is shown in figure 3.7.

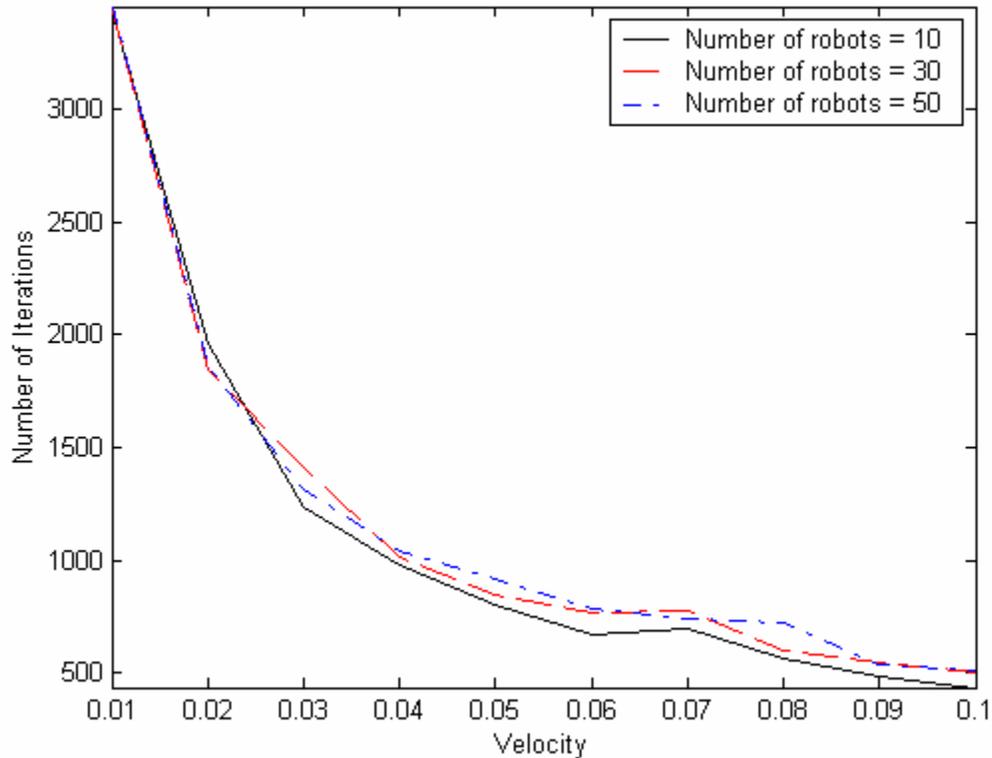


Figure 3.7 Analysis of the spreading algorithm with variation in velocity

An exponential decrease in the number of iterations (time) can be noted. Also, independent of the number of robots, the amount of time it takes to spread is almost the same.

#### 3.2.3.4 Noise analysis

In all the above results of the spreading algorithm, it has been assumed that there is no noise present in the environment. This assumption is not true for any environment encountered in reality. The spreading algorithm was hence simulated with the presence of noise and the uniformity of the resulting distribution was calculated in the same way as in section 3.2.3.2. The parameters used for this simulation are same as in table 3.1. For every iteration, additive white noise of a particular power was added in the position information of robots. The standard deviation of the estimated pdf for various noise

powers is calculated to compare with the desired value of the ideal uniform pdf, which is equal to  $1/(\pi \cdot \text{radius of the circle})$ . Figure 3.8 shows the variation of this new uniformity criterion for various noise powers. It can be seen that the curve is almost constant till  $-30$  db after which it increases drastically.

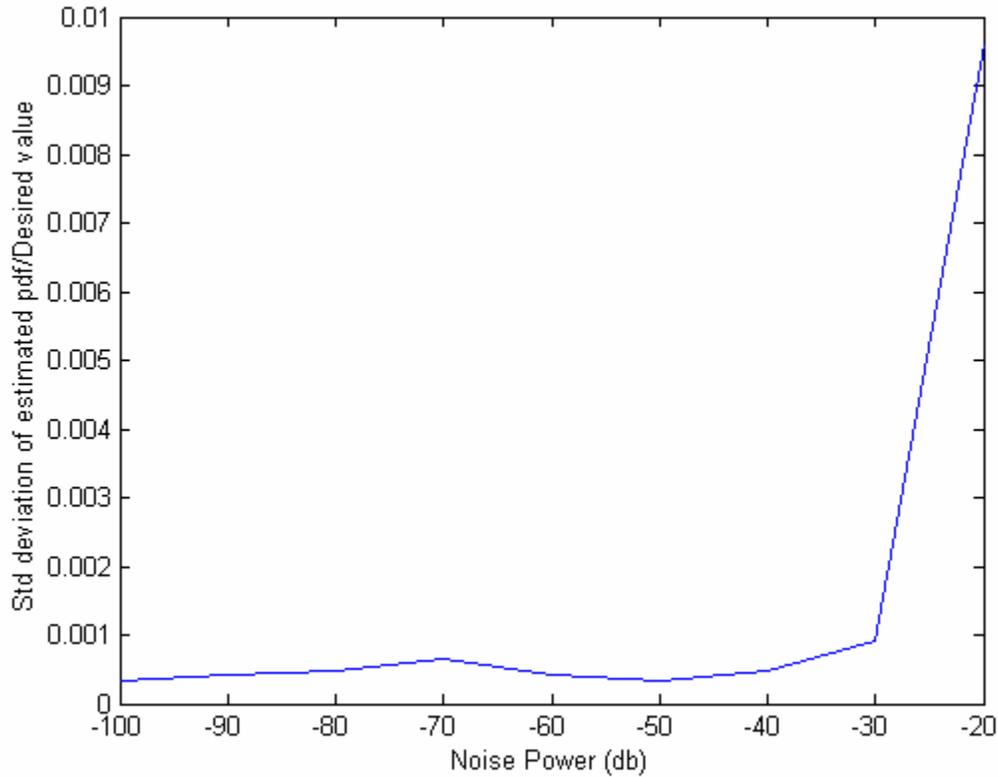


Figure 3.8 Noise analysis for spreading algorithm

### 3.2.4 Obstacle Avoidance

Since it is possible for the robots to spread without knowing their positions, an obstacle avoidance technique that does not use position information is required. Here, an attempt has been made to use the same IPIM for obstacle avoidance. It is assumed that each robot has a camera in its front. When the robots are proceeding towards a target, they can sense the obstacle using the camera in the front. From the image got from the camera, the robots measure the angle between their orientation and the edge of the

obstacle. Depending on this angle, the robots move in a direction away from the obstacle as if they are experiencing a repelling IF from the obstacle. The IF given by the obstacle has to be carefully tuned so that the robots do not get away from the group and at the same time, do not hit the obstacle. In the simulations, the IF given by the obstacle was assumed to be equivalent to placing an IPC at the location of the obstacle. As the robots close on to the obstacle, the repelling IF gets higher and higher. Once the robots cross the obstacle, they no more experience any repelling force and the robots come back into the group by experiencing only the IF from other robots according to the preset threshold  $\gamma$ . To facilitate regrouping of robots after they cross the obstacle, the IF between them is doubled for a brief time. The simulation results of the obstacle avoidance algorithm are shown in figure 3.9. As seen in figure 3.9A and B, the robots first spread themselves as they proceed in the 'x' direction. Then, they clearly avoid the obstacle as shown in figure 3.9C. Figure 3.9D through F show us the robots regrouping themselves. The iteration count is denoted in the figure as  $i$ . The parameters used in this simulation are shown in table 3.2.

Table 3.2 Parameters used in the obstacle avoidance algorithm

| <b>Parameter</b>       | <b>Value</b> |
|------------------------|--------------|
| Number of robots       | 30           |
| Radius of the circle   | 1 m          |
| Number of iterations   | 3501         |
| Time step              | 3 $\mu$ s    |
| Velocity of the robots | 1 m/s        |
| Length of the obstacle | 0.2 m        |

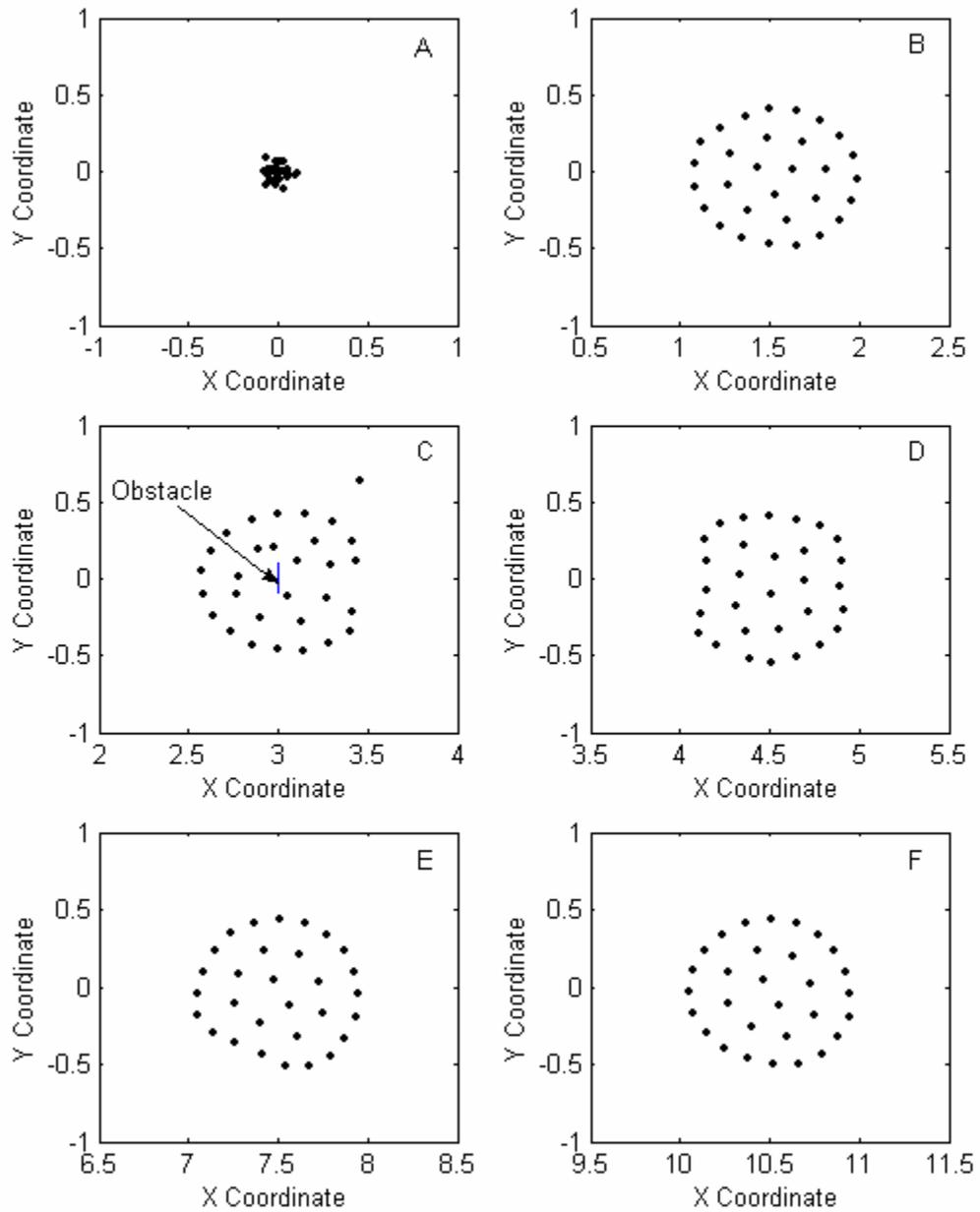


Figure 3.9 Obstacle Avoidance. A) Initial Position of robots ( $i = 1$ ). B) Robots uniformly spread ( $i = 501$ ). C) Robots avoiding the obstacle ( $i = 1001$ ). D-F) Regrouping of robots ( $i = 1501, 2501, 3501$  respectively).

### 3.3 Spreading Algorithm for Piecewise Linear Regions

The problem of spreading of robots over any region boils down to the mathematical paradigm – whether a point is inside a curve or not. The solution for this problem can be very complex depending of the arbitrary shape of the curve and a general analytic solution is not known. But if it can be assumed that any curve can be sufficiently approximated with a piece-wise linear curve, then the solution is given by a very simple theorem called Jordan curve theorem [Fra02].

Unlike the spreading algorithm for circular region, the above solution warrants the robots to know their own absolute position as well as the position of the vertices of the curve to spread themselves inside the given region. Since this is the simplest way to implement spreading over any piecewise linear region, the increase in cost due to the above requirement may be compromised.

#### 3.3.1 Jordan Curve Theorem

A semi-infinite ray going out in any direction from the testing point (figure 3.10) will cross the edges of the curve odd number of times if the point is inside the curve and even number of times if the point is outside the curve. At each crossing, the ray switches between inside and outside. This theorem is true for any closed curve, whether it is convex or concave.

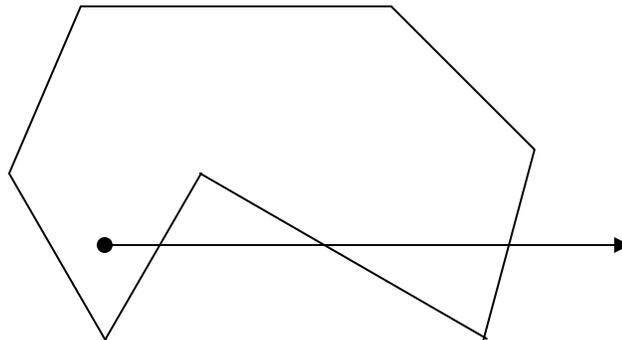


Figure 3.10 Jordan curve theorem

In the simulations, a ray going vertically upwards from the point is assumed although the theorem does not impose any restrictions on the direction of the ray. Then, the ray is tested against each edge of the region by checking whether:

1. The point in the half-plane is below the extended edge
2. The point's X coordinate is within the edge's X-range

Handling of points on the edges will be tricky since the solution may turn out to be incorrect. In any case, the number of points exactly on any one of the edges will be very few and the algorithm works well in spite of these points.

### 3.3.2 Results

Spreading in a star shaped region is considered here. The parameters used in the above simulation are in table 3.3. The result is shown in figure 3.11. As seen in this figure, the robots initially spread in a circular fashion and as they reach the boundary, few of them start moving towards the narrow edges. One can see that the formation gets well defined as the edges are reached gradually.

Table 3.3 Parameters used in the spreading algorithm for star shaped region

| Parameter              | Value  |
|------------------------|--------|
| Number of robots       | 100    |
| Number of iterations   | 100    |
| Time step              | 0.03 s |
| Velocity of the robots | 1 m/s  |

Figure 3.12 illustrates the application of spreading algorithm for various shapes. It can be inferred that as the edges becomes narrow, the robots struggle to reach the edges and hence the distribution is not completely uniform in those regions as compared to

other regions. This is more visible in elongated star formation and wedge formation as seen in figure 3.12C and D.

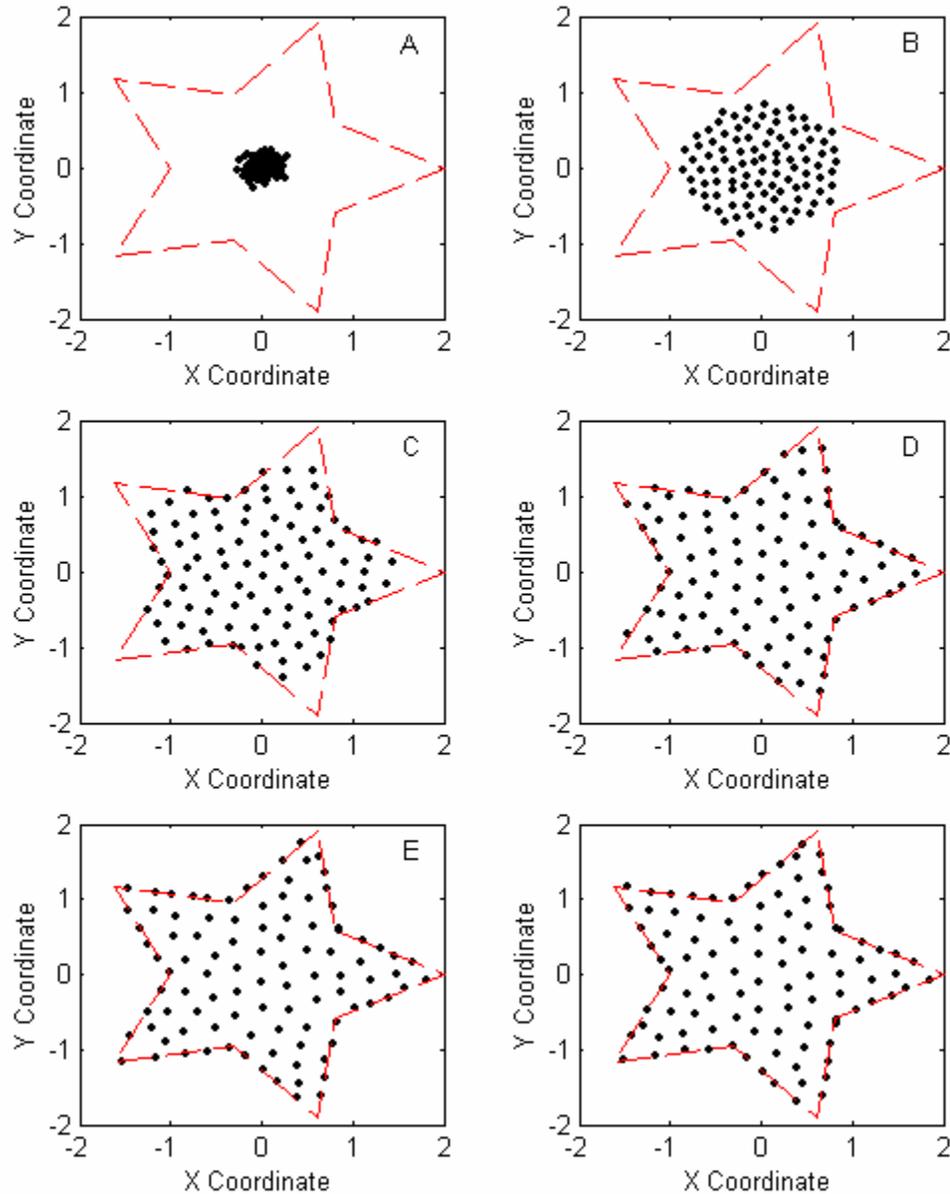


Figure 3.11 Spreading of robots over a star shaped region. A) Initial position of robots. B) Robots spreading in a circular shape. C and D) Robots slowly reaching the edges. E and F) Refinement of the edges.

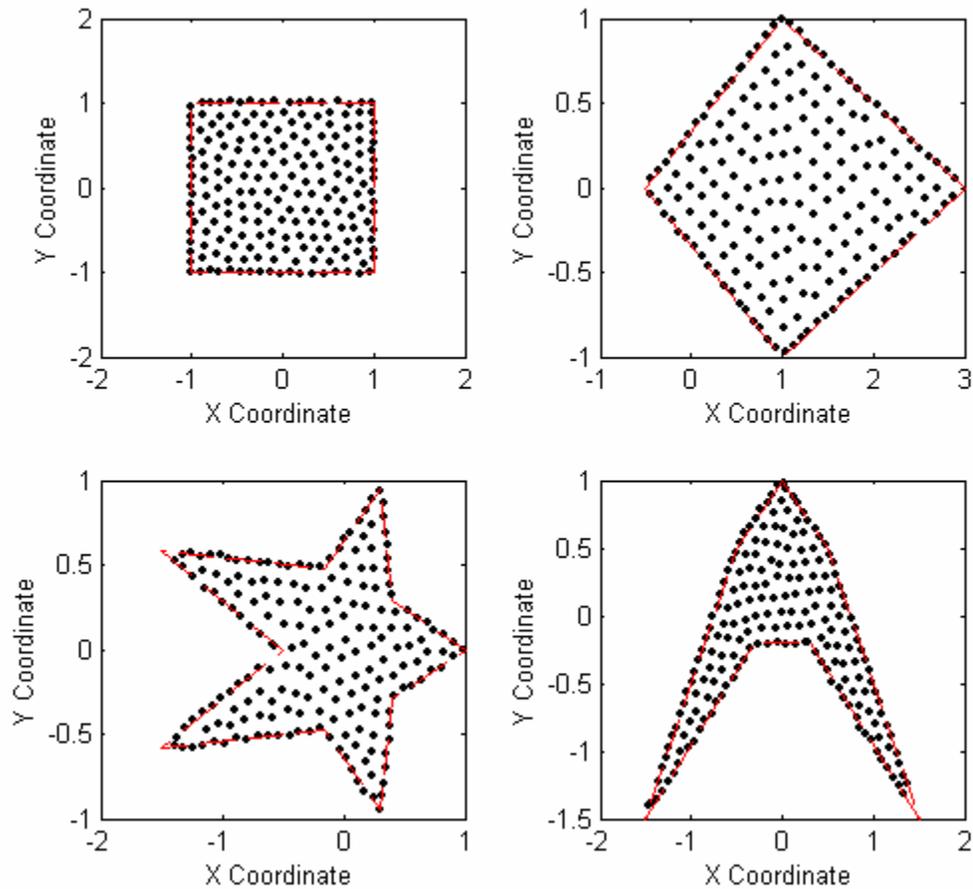


Figure 3.12 Spreading of robots over different regions. A) Square. B) Quadrilateral. C) Elongated Star. D) Wedge

### 3.4 Comparison with the Nearest Neighbor Technique

Some researchers have used the nearest neighbor technique for uniform spreading of robots over a particular region. In this technique, the robots spread themselves by moving away from their nearest neighbor(s). The suggested self-organization algorithm is superior to the nearest neighbor technique in many aspects. The nearest neighbor technique always requires the identification of the nearest neighbor. In the suggested technique, all robots are considered the same and no identification is required. Moreover, in the case of a circular region, it is possible to implement the spreading with the robots

not knowing their absolute positions. In the self-organization algorithm, the robots never collide since collision avoidance is intrinsically built in the technique with each robot repelling all other robots by transmitting the IP.

The following simulation compares the spreading speed between the IPIM based self-organization algorithm and the nearest neighbor technique. Both the techniques are applied for spreading of robots over a circular region with a unit radius. The uniformity of the resulting distribution of robots was calculated using the same technique explained in section 3.2.3.2 and the variances are shown in the figures 3.13 through 3.15 for various number of robots. It is clear that the self-organization algorithm brings uniformity in the distribution faster than the nearest neighbor technique. Also the final variance achieved by the self-organization algorithm is always smaller, which is visible when the number of robots  $N = 30$  and  $50$ . It should be mentioned that the speed of the robots was kept to be same for a fair comparison.

Above all, IPIM is a framework which offers more flexibility in self-organization. The robots can be organized in different ways by using different types of IPs. Moreover, the nearest neighbor technique can be viewed as a part of the IPIM based self-organization where the IP transmitted by the robots can reach only the nearest neighbor(s) around them located within a stipulated distance. Hence, IPIM can be seen as a generalized framework, with many parameters that can be suitably tuned according to the requirements of the application on hand.

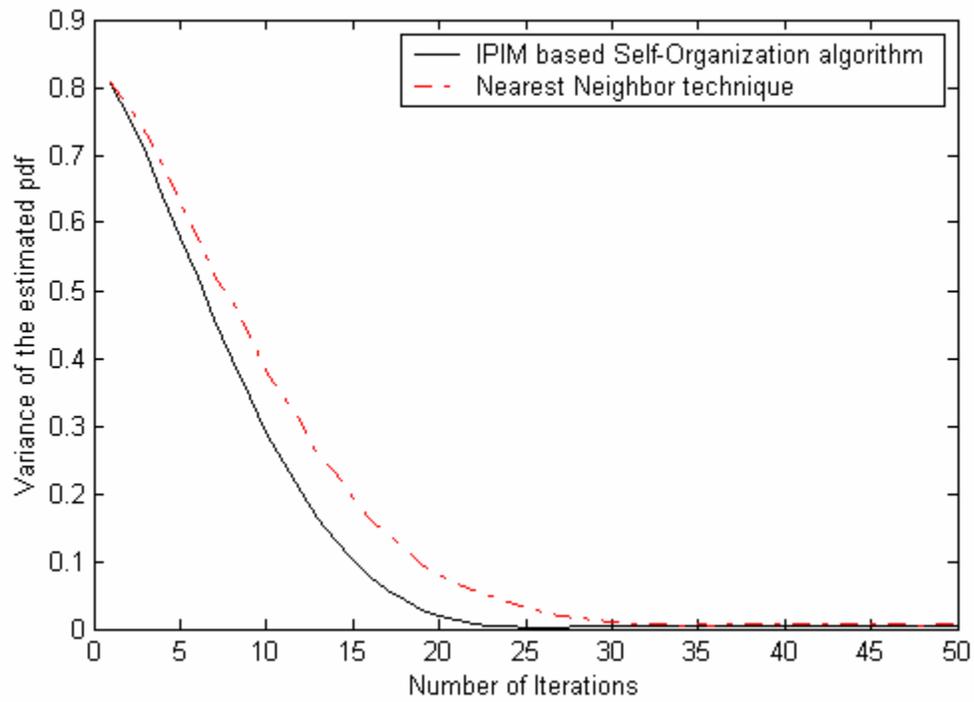


Figure 3.13 Comparison of speed of spreading ( $N = 10$ )

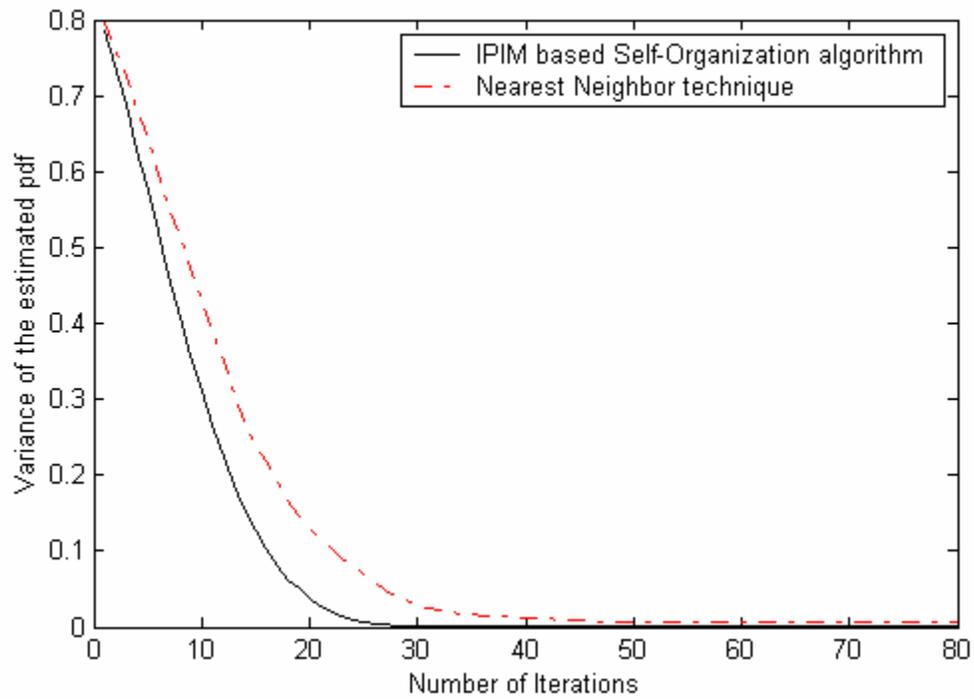


Figure 3.14 Comparison of speed of spreading ( $N = 30$ )

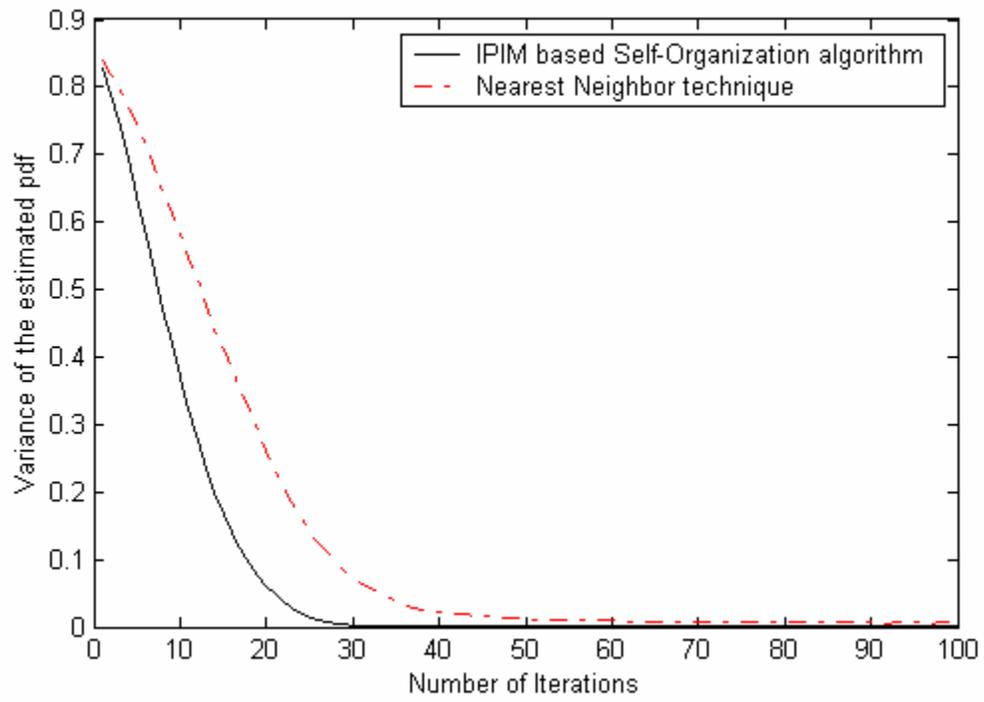


Figure 3.15 Comparison of speed of spreading ( $N = 50$ )

## CHAPTER 4 COLLECTIVE TARGET TRACKING

The self-organization algorithm for spreading the robots uniformly over any region has been discussed and analyzed in Chapter 3. The next goal of moving the robots collectively towards a target is taken for discussion in this chapter.

### **4.1 Requirements and Suitable Modifications**

Application of target tracking algorithm demands the understanding of the proper environment where it is being applied. According to the requirements of the intended usage, the mode of application of the algorithm needs to be modified. Below are some of the different situations encountered in the target tracking scenario.

The first and the simplest situation is the one where the robots know their own positions as well as the position of the target. Then, the solution becomes trivial.

Secondly, a centralized approach may be designed in which one leader robot knows the position of the target and it guides the group towards it. As explained in Chapter 1, centralized strategy can have many disadvantages and hence, not a preferred one. One of the ways this technique can be improved is by having more than one leader in the group and periodically rotating the leader in a move to make the group robust to failures.

In some applications, the target itself issues a beacon signal or it reflects signals from the robots, intimating them on the direction in which it is located. Eisler *et. al.* [Eis02] have analyzed the issues related to the tracking of unknown acoustic wave reflecting objects using a swarm of pressure sensors. In their technique, an emitter sends an acoustic pulse and the object reflects this pulse. The swarm of mobile sensors receives

both the transmitted and reflected waves. Then, using a localized cooperative pressure measurement the sensors detect the direction of maximum intensity in which they move and finally reach the object. Here the spacing between the sensors needs to be the same for accurate pressure measurements. Although this requirement is satisfied, the group moves in no particular formation towards the target. Moreover, the sensors need to share their positions and pressure measurements in order to move collectively. The proposed self-organization algorithm minimizes the sharing of information between agents.

The most difficult situation arises when the robots have no knowledge about the nature and position of the target and a decentralized strategy is required. Hence, some kind of external guidance needs to be given to the robots to move them towards the target. The above requirements suit for applications where minimal risk and high indestructibility (ex. bombing of a target), are desired. The target tracking algorithm for this situation is investigated in the next section.

#### **4.2 Decentralized Target Tracking Algorithm**

The technique used for decentralized and collective target tracking can be discussed with the help of figure 4.1. Two external base stations are used to transmit the direction information to the swarm. The two base stations can also be assumed as stationary IPCs transmitting IF to the swarm. But this IF needs to be the same for all the agents in the swarm and has to be rotated properly in order to move the swarm properly towards the target. This can be achieved as follows. Each base station is assumed to have a simple radar through which it detects the position of the target as well as the position of the center of the swarm. With the above position information, each base station then decides independently on whether the center of the swarm is to the left or right of the line of sight (LOS) with the target. Accordingly, this direction information is transmitted to the

swarm, perhaps, by changing the sign of the transmitted IP. As shown in the figure 4.1, the whole area is divided into 4 regions. The IF is given proper rotation depending on the region in which the swarm center is located, so that the swarm moves towards the target.

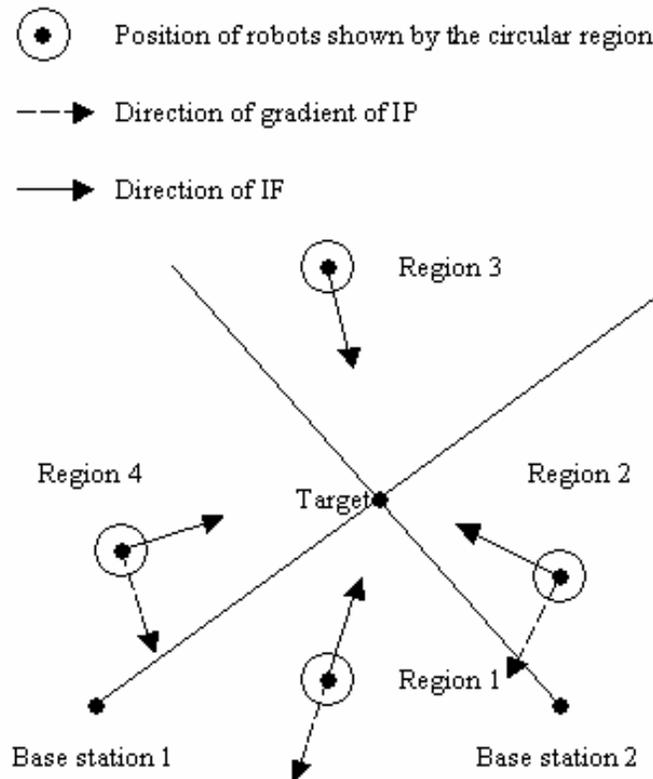


Figure 4.1 Technique for collective target tracking

Thus, each agent in the swarm receives signals from other agents as well as from the two base stations. First, the gradient of the IP from the two base stations is calculated and then rotated properly to get the IF for moving the group towards the target. The total IF experienced by each agent is the sum of the IF from the base stations and the IF from other agents.

### 4.3 Results

#### 4.3.1 Stationary Target

We demonstrate the target tracking algorithm on a stationary target and present the results for a particular case where the agents reaches the target in a circular formation.

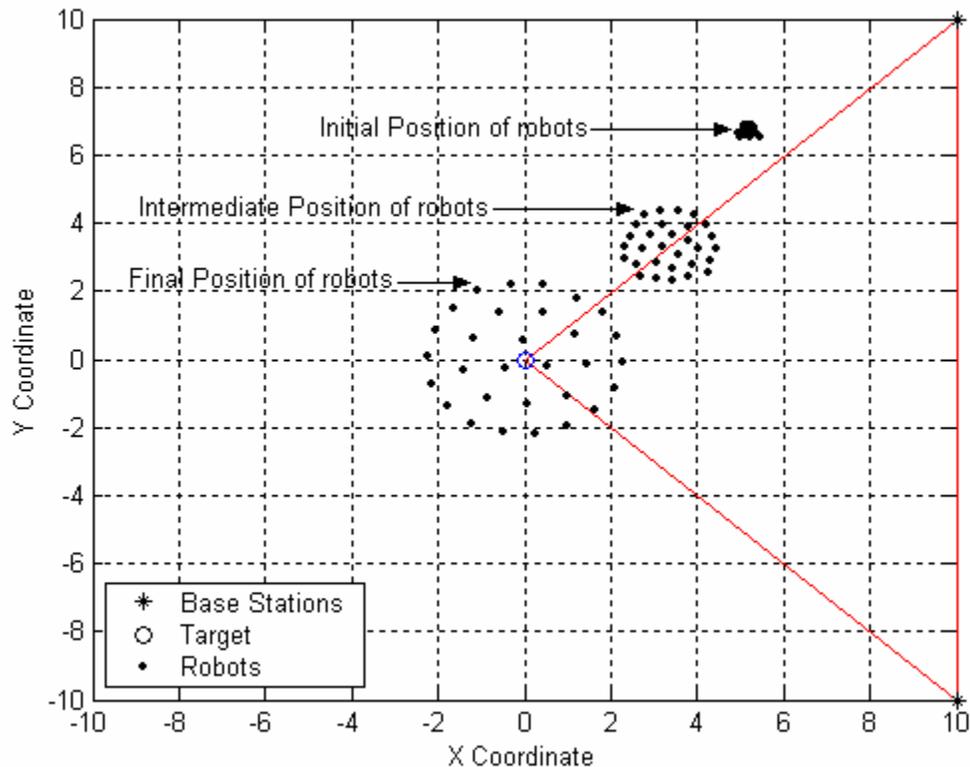


Figure 4.2 Collective target tracking of a stationary target with robots in circular formation

Figure 4.2 shows the two base stations, the stationary target and the LOS. It must be noted that the two processes namely spreading of the agents and the tracking of the target take place simultaneously. It is clear from the figure that the group first reaches the LOS of one of the base stations on its way towards the target. This behavior is due to the use of gradient indirectly as IF and hence, is similar to that of gradient minimization algorithms.

Next, we show that the formation can be changed to any shape while using the target tracking algorithm. In order to achieve, it is important to use the proper spreading algorithm explained in Chapter 3. Figure 4.3 shows the target tracking technique applied to swarm in a star formation.

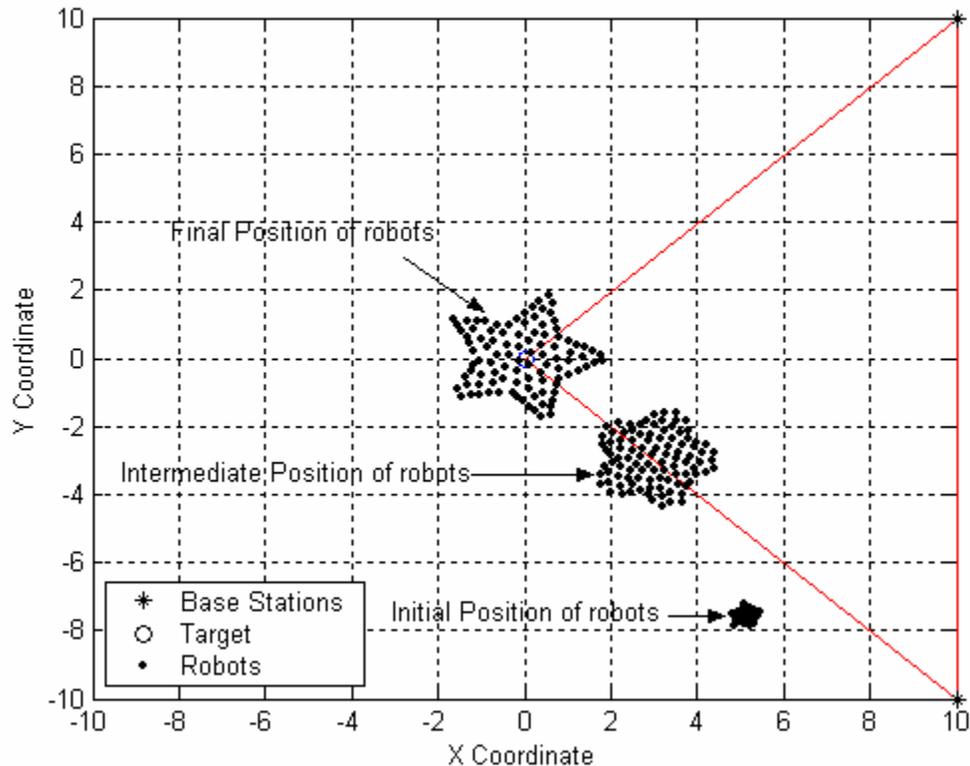


Figure 4.3 Collective target tracking of a stationary target with robots in star formation

#### 4.3.2 Moving Target

All the principles applied on tracking of a stationary target hold for moving targets too. We present here an example in which the target is moving in a kind of convoluted sinusoidal trajectory.

At the first intermediate position of the robots shown in figure 4.4, the robots orient along the trajectory of the target. But as one can observe, still they have not completely reached the moving target since the target is not at the center of the formation. At the

second intermediate position, the agents lock on to the target and also keep in pace with the target, tracing it perfectly till the end.

To show the robustness of this idea, the simulation is carried out such that some randomly chosen agents fail during the course of tracking. Still, the agents are successful in regrouping themselves back in the required formation and tracking the target easily. Although the robots will not be colliding with each other due to the IP emitted by them, the robots can still collide with the inactive or failed robots. Hence, a good obstacle avoidance scheme is required.

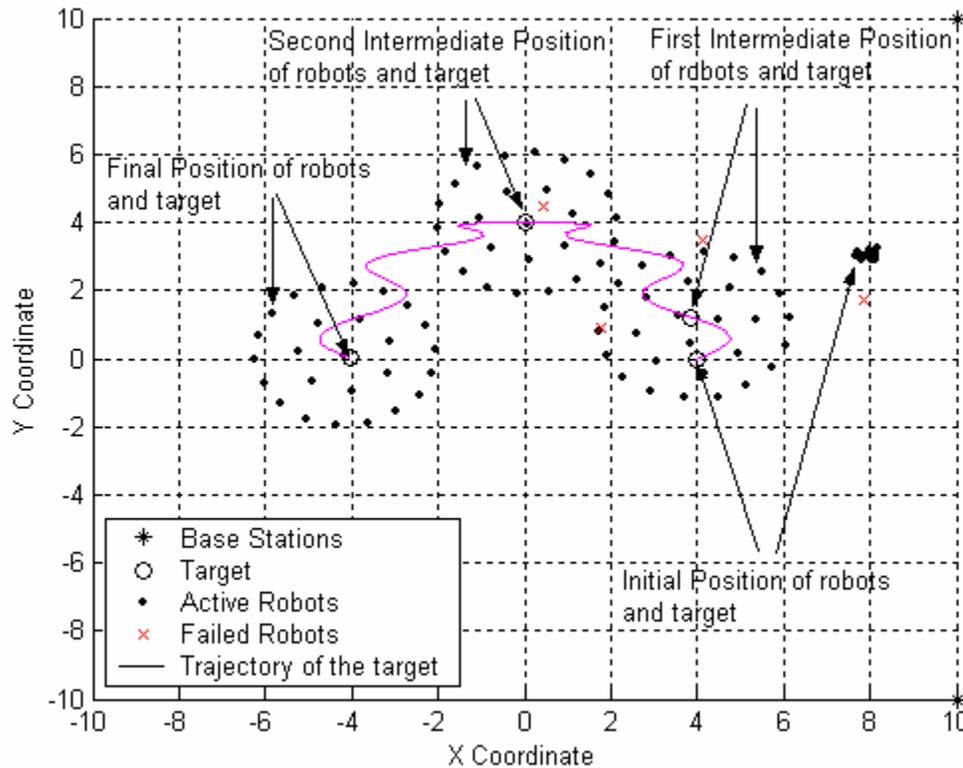


Figure 4.4 Collective target tracking of a moving target with robots in circular formation

When the target is moving, the radars at the base stations need to constantly track the position of the target and move their LOS accordingly. Further, only when the speed of the target is less than the maximum speed of robots, the swarm will be able to lock on

to the target and track it. In the above simulation, the speed of the robots was 1 m/s while the speed of the target was approximately 0.28 m/s.

## CHAPTER 5 TOWARD REAL-TIME IMPLEMENTATION

In this chapter, some implementation issues and the suggested solutions are discussed. Also, as a step towards real-time implementation, the algorithm is implemented in software called Webots, in which the kinetics of the robot movement can be simulated. This is also described later in this chapter.

### **5.1 Implementation Issues**

#### **5.1.1 Communication Scheme**

The requirements of the intended application can dictate the scheme used for communication between robots. One of the simple and cost-effective schemes is to allow each robot transmit at a different frequency with the frequencies evenly spaced. Each robot must be provided with apriori information on the number of robots in the swarm and their transmission frequencies. Then, from the received signal at any instant, the individual received amplitudes from every other robot can be extracted using a bank of band pass filters. This basic transmission scheme can be enhanced appropriately using any coding/ modulation technique. This scheme may suffice for many civilian applications like material transportation.

But for military applications, the communication scheme needs to be highly noise and jamming resistant. One of the possible solutions is the use of pseudo-noise (PN) signals, since they offer a secure way of transmission and robustness to jamming. PN sequences used in CDMA based wireless communication must have low cross-correlation since the receiver must be able to de-spread only one particular PN sequence from many

PN sequences for better reception of the desired signal. In this scenario, however, we require that the robots do not use different seeded PN sequences, because then, as the swarm size increases, hardware complexity of the receivers will also increase. Hence, all the robots must use the same PN sequence with different phases and autocorrelation property of the PN sequence assumes importance. The autocorrelation function of an m-sequence [Hay94] is constant at a very small value ( $-1/M$ , where  $M$  is the sequence period) for non-zero lags and is 1 at zero lag. Hence, they can be beneficially used in this application.

Since every robot is at different distances from a particular robot, the received signal will have the same PN sequence arriving in different phase shifts. Then, when there are  $N$  robots, the received signal  $r_i(t)$  at the  $i^{\text{th}}$  robot can be written as

$$r_i(t) = \sum_{k=1, k \neq i}^N A_{ik} g(t - \Delta_{ik}) \quad (5.1)$$

where  $A_{ik}$  is the amplitude of the signal coming from robot  $k$ . Again,  $A_{ik} = A / d_{ik}^2$ , where  $d_{ik}$  is the distance between the  $i^{\text{th}}$  and  $k^{\text{th}}$  robots. Also, we let  $g(t)$  be the assumed PN sequence, which is common to all the robots. The phase  $\Delta_{ik}$  depends on the time-of-arrival of the signal, which is a function of the distance  $d_{ik}$ .

If the PN sequence is  $g(t)$ , then the decoded signal will be at a particular phase shift  $L$  of the PN sequence

$$y_i^L = E[r_i(t)g(t-L)] \quad (5.2)$$

Then, substituting the received signal in the above equation gives

$$y_i^L = E \left[ \sum_{k=1}^N A_{ik} g(t - \Delta_{ik}) g(t - L) \right] \quad (5.3)$$

The autocorrelation of the PN sequence,

$$C_{ik}^L = E[g(t - \Delta_{ik})g(t - L)] \quad (5.4)$$

For the specific case of an m-sequence,

$$C_{ik}^L = \begin{cases} 1 & \text{if } L = \Delta_k + zM \\ -\frac{1}{M} & \text{if } L \neq \Delta_k + zM \end{cases} \quad (5.5)$$

where  $M$  is the length of PN sequence and  $z$  is any integer. Then, the decoded received signal becomes as shown in (5.6), where  $j=1,2,\dots,N$ . Here,  $N$  is the number of robots.

Thus, a reasonable threshold can be set to get the amplitudes  $A_{ij}, j=1,2,\dots,N$ .

$$y_i^L = \sum_{k=1}^N A_{ik} C_{ik}^L = \begin{cases} -\frac{1}{M} \sum_{k=1, k \neq i}^N A_{ik} & \text{if } L \neq \Delta_1, \dots, \Delta_N \\ A_{ij} - \frac{1}{M} \sum_{k=1, k \neq i}^N A_{ik} & \text{if } L = \Delta_j \end{cases} \quad (5.6)$$

Notice that  $M \gg N$  is required for this to work properly all the time. Then, the potential  $V_i$  for a particular robot is the sum of the received signal amplitudes  $A_{jk}$  from all other robots.

$$V_i = \sum_{k=1}^N A_{ik} \quad (5.7)$$

Continuing in the same way, when the target tracking algorithm is applied, the two base stations can use two PN sequences, which are different from the one used for inter-robot communication. The sign of the PN sequences can be used to send the direction information.

Though according to the IPIM model, only the sum of the received signal amplitudes is required for interaction between IPCs, the communication schemes

explained above provide us with the individual received amplitudes. Hence, a better communication scheme where a simple receiver can detect directly the IP at the robot's position is required.

### 5.1.2 Gradient Calculation

Assuming a receiver that can directly detect the IP at the robot's position, a method to find IF at that position is formulated here. The gradient of the potential with respect to the robot's position  $p_i$  can be estimated from the measurements taken by three antennas on the robot using first order difference approximation of the derivatives. To improve accuracy in the calculations, a rectangular grid of receiver antennas may be used to calculate the gradient in the same way.

### 5.1.3 Results

The spreading algorithm for circular region was found to be working satisfactorily with the use of an M sequence for communication between robots and a rectangular grid antenna for gradient calculation. The parameters used in the simulation are tabulated in table 5.1. The spreading speed of the spreading algorithm was also verified by calculating the variance of the estimated pdf over the circular region at every iteration as shown in figure 5.1. It can be seen that the spreading speed is almost the same here as compared to the spreading speed when the position information of robots were transmitted (figures 3.13, 3.14 and 3.15). The variance of the estimated pdf after the robots reach their final positions was found to be 4.5246e-004. This is slightly higher than the variance obtained when the spreading algorithm was simulated by transmitting position information of robots.

Table 5.1 Parameters used to verify spreading algorithm with the suggested solutions for implementation issues

| Parameter  | Value  |
|--|--------|
| Number of robots                                   | 20     |
| Radius of the circle                               | 1 m    |
| Number of iterations                               | 40     |
| Time step  | 0.03 s |
| Velocity of the robots                             | 1 m/s  |
| Length of M sequence                               | 127    |
| Number of elements in the rectangular grid antenna | 9      |

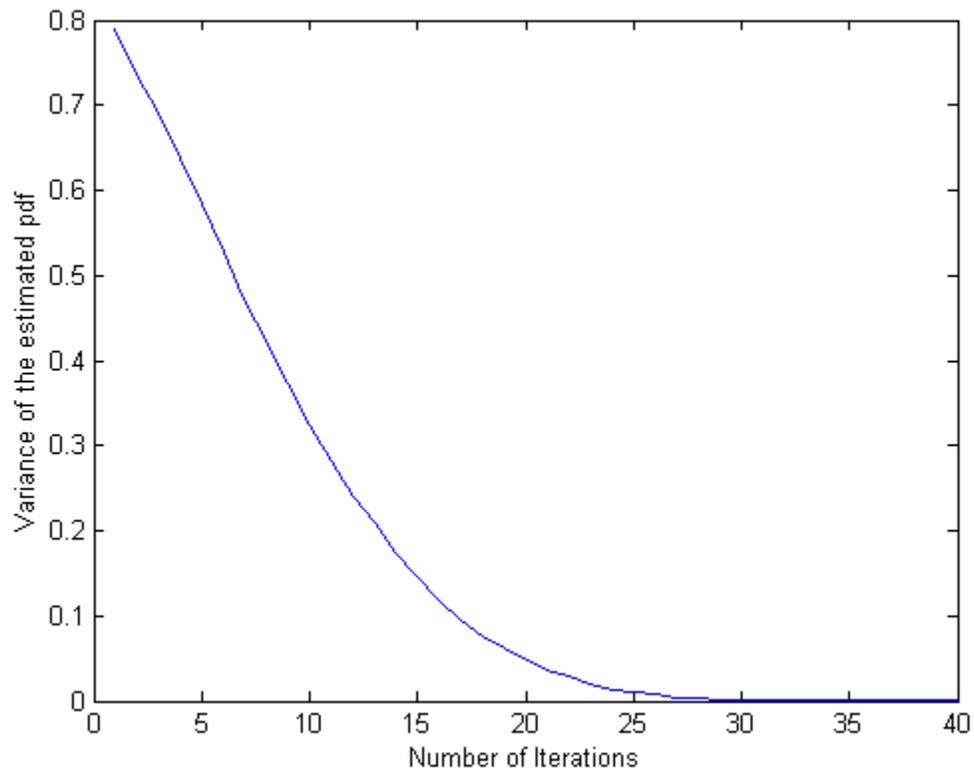


Figure 5.1 Spreading Speed with the use of M sequence and rectangular grid antenna

As in section 3.2.3.4, the standard deviation of the estimated pdf for various noise powers was calculated for various noise powers and was divided by the desired value of

the ideal uniform pdf (figure 5.2). Here, additive white noise of a particular power was added directly to the received signal at each antenna in the grid. It can be seen that the curve is almost constant till 40 db after which it shows an exponential increase.

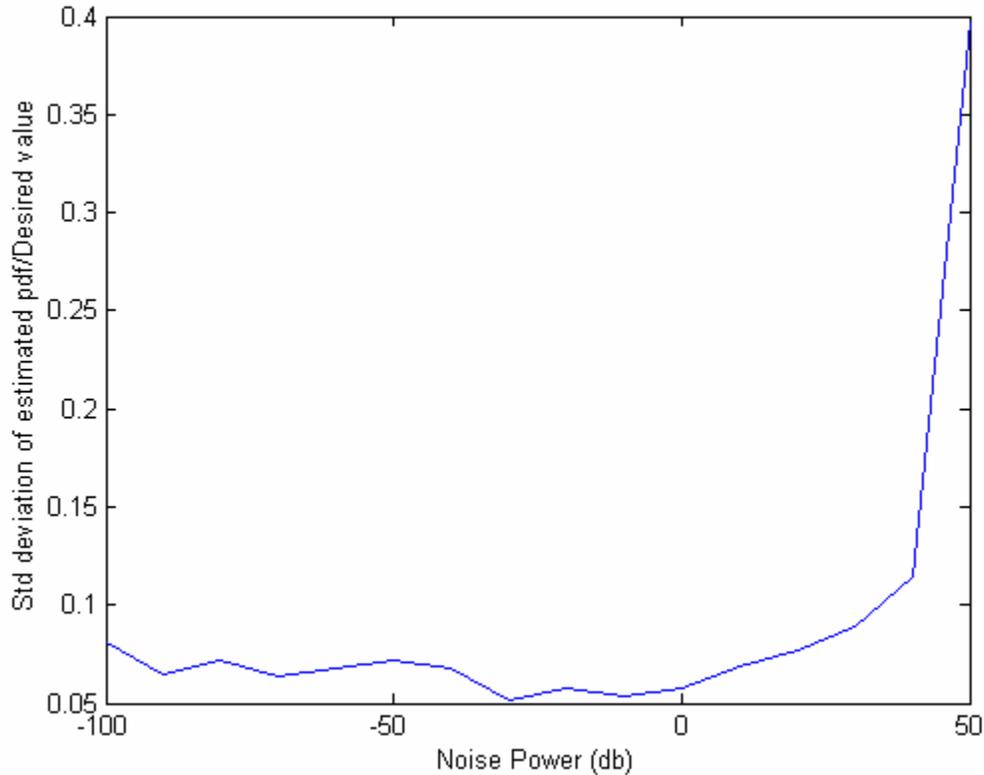


Figure 5.2 Noise analysis with the use of M sequence and rectangular grid antenna

## 5.2 Webots Simulation

Webots 3.0 provided by Cyberbotics is an intermediate stand-alone simulation platform used to study any collective robotics algorithm before its actual hardware implementation. A step-by-step simulation of the spreading algorithm in Webots 3.0, first using the transmission of position information and then, using the above explained solutions for implementation issues was planned. The simulation could not be made stand-alone due to few constraints and bugs in Webots 3.0 as detailed below.

- To determine the location of the robot dynamically, there is a function called `robot_get_position`, but it has bugs and hence could not be used.
- The receiver on each robot, requires to receive signals in multiple frequencies, but the receiver designed in Webots 3.0 can receive only at a single frequency.

Due to the above reasons, Webots is used now only to provide a better visualization for the spreading algorithm. The algorithm is hence first executed in Matlab and then the results are loaded into Webots forcing the robots move to preset locations. Still, this simulation can be effectively used to study the kinematics of the robot movements. The above constraints have been communicated to Cyberbotics. The suggested changes and improvements are expected to be implemented in Webots 4.0, the first version of which has been released recently in May 2003. In future, the self-organization algorithm may be completely simulated in Webots 4.0.

### **5.2.1 Webots – An Introduction**

The Webots mobile robotics simulation software provides us with an interactive graphic environment to conduct studies on collective mobile robots [Cyb03a]. Each object in the Webots set-up has a number of properties like shape, color and texture. Webots 3.0 has the capability to design a differential wheeled robot that can then be equipped with many sensors and actuators [Cyb03b]. The robots can be controlled using controller programs developed in C. Finally, after testing any algorithm in Webots, the controllers can also be transferred onto some of the real robots like koala and khypera.

Below are some of the important features of Webots [Cyb03a].

- A complete sensor library equipped with distance sensors, range sensors, light sensors, touch sensors, GPS, cameras, receivers, position sensors for servos and incremental encoders for wheels is available.
- Similarly, Webots has a complete actuator library with differential wheel motor unit, independent wheel motors, servos, LEDs, emitters and grippers.

- With Webots, complex environments for mobile robot simulations can be created, using advanced hardware accelerated OpenGL technologies, including lighting, smooth shading, texture mapping. Environments as large as needed can be created and optimized for fast simulations.
- The simulation system is accurate and uses virtual time mechanism, making it possible to run simulations much faster than it would take on a real robot. Depending on the complexity of the setup and the power of the workstation, simulations can run up to 300 times faster than the real robot when the fast simulation mode is used.
- The graphical user interface (GUI) of Webots provides excellent interaction with the simulation while it is running. The position, orientation and zoom of viewpoints and objects can be changed with just a click of the mouse.

Some of the applications that have been developed by various researchers with the help of Webots are summarized below [Cyb03a]:

- At Autonomous Systems Lab, Swiss Federal Institute of Technology (EPFL), Laussane [Zuf02], research on the evolution of cooperation and labor division in artificial ants is being done.
- Ijspeert *et. al.* [Ijs01] have worked on stick pulling using a group of hypera robots.
- Alice robots have been used to play soccer by Caprari [Cap01].
- Zhang and Martinoli [Zha02] have used Swarm intelligence to improve individual and collective traffic safety.

The following are the various steps for simulation in Webots:

1. Design the environment and robot in the Scene Tree Window (STW) provided by Webots. STW is composed of a list of nodes, each containing fields. Fields can contain values or nodes.
2. Develop a controller program in C using any of the available C compilers. Although GNU GCC compiler is suggested, any other compiler can be used. Microsoft Visual C++ was used as the compiler here.
3. Link the controller with the environment in Webots. Run and debug the controller.

### **5.2.2 Environment and Robot Design**

The environment and the robots designed in STW are shown in figure 5.3.

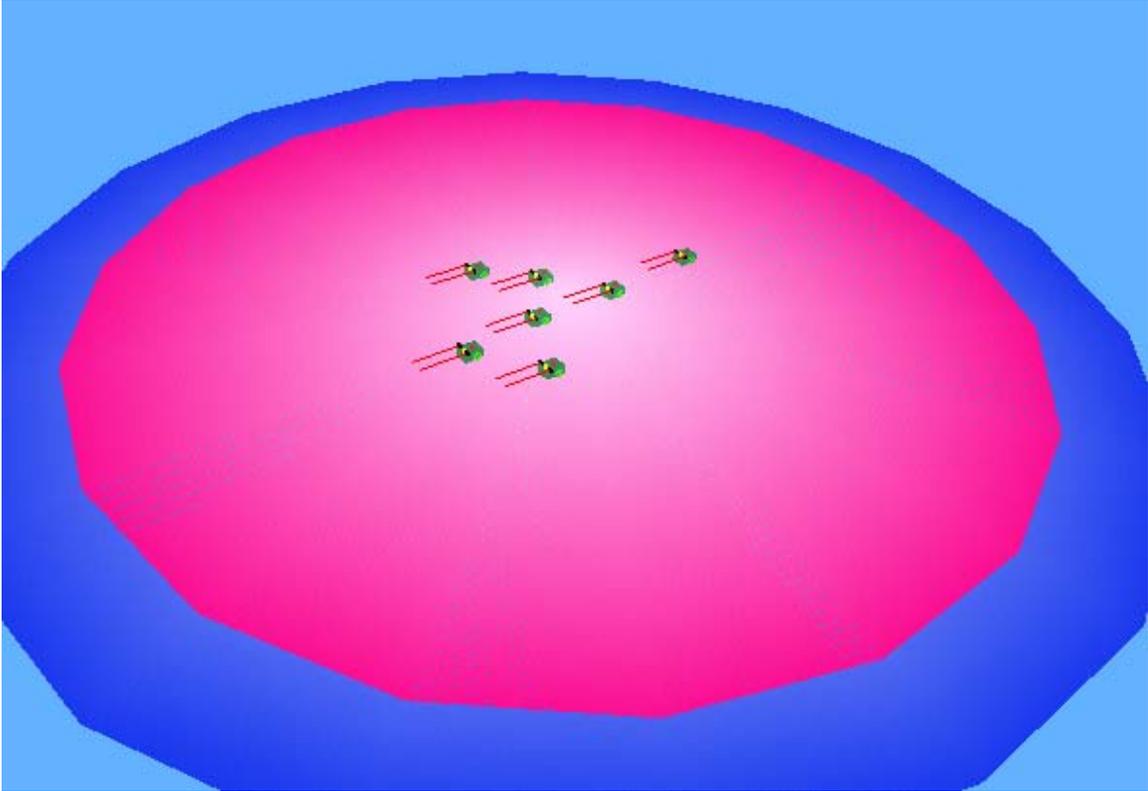


Figure 5.3 Designed environment and robots in Webots

The design process is as follows. In STW, the basic environment was first designed using Viewpoint, Background and PointLight nodes. Viewpoint node gives the initial viewpoint of the environment; Background node specifies the background color; PointLight node defines the desired lighting effect. The nodes and their various fields are shown in figure 5.4.

The circular region for spreading of robots is designed using two solid nodes. The DESIREDCIRCLE solid node defines all the properties for the desired circular region having a radius of 5 meters. The OUTERCIRCLE solid node acts as a bigger stage since the robots may spread outside the desired circular region. The DESIREDCIRCLE solid node and its field values are shown in figures 5.5. The OUTERCIRCLE node has almost same properties except that it has different color and the radius field is set to 6 meters.

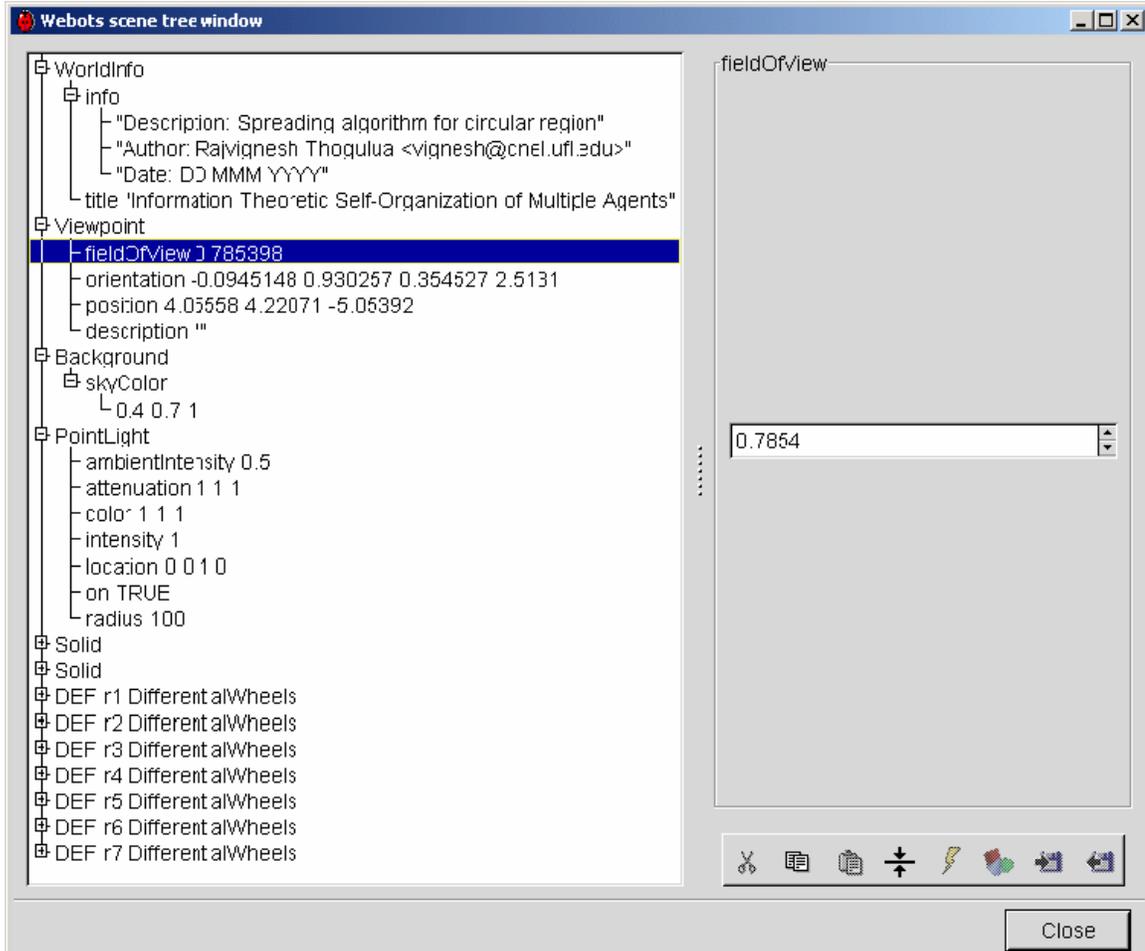


Figure 5.4 Basic environment design

All the robots have two-wheel differential steering designed using DifferentialWheels node. The DifferentialWheels node for the first robot with its fields is shown in figure 5.6. The robot has a body, two wheels, two IR sensors for collision avoidance and a transmitter-receiver pair as seen from the children nodes. The transmitter-receiver pair is not used in the controller program but however, it is included for future use.

The most important properties of the DifferentialWheels node are wheelRadius and axleLength. The body of the robot as well as the wheels should be designed accordingly.

Also, the controller program `groundcomm.c` makes use of these two values for developing the vehicle dynamics of the robot, as described in the next section.

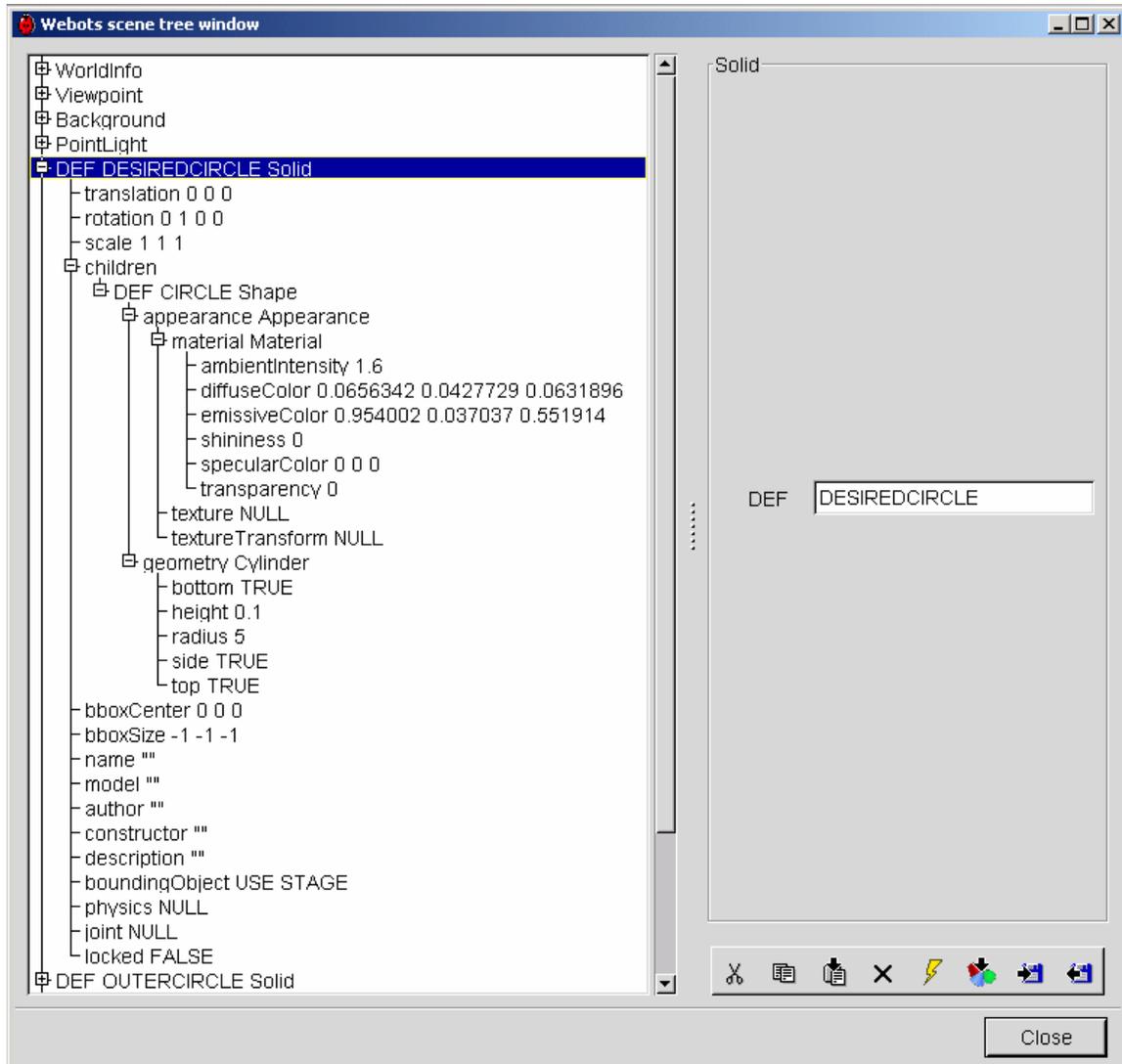


Figure 5.5 Field values for the DESIREDCIRCLE node

### 5.2.3 Controller Programming

All the kinematics of the robot movements is incorporated in the controller program. Since the spreading principles are the same for all the robots, only one controller needs to be developed. This controller program can then be linked to each robot in the STW.

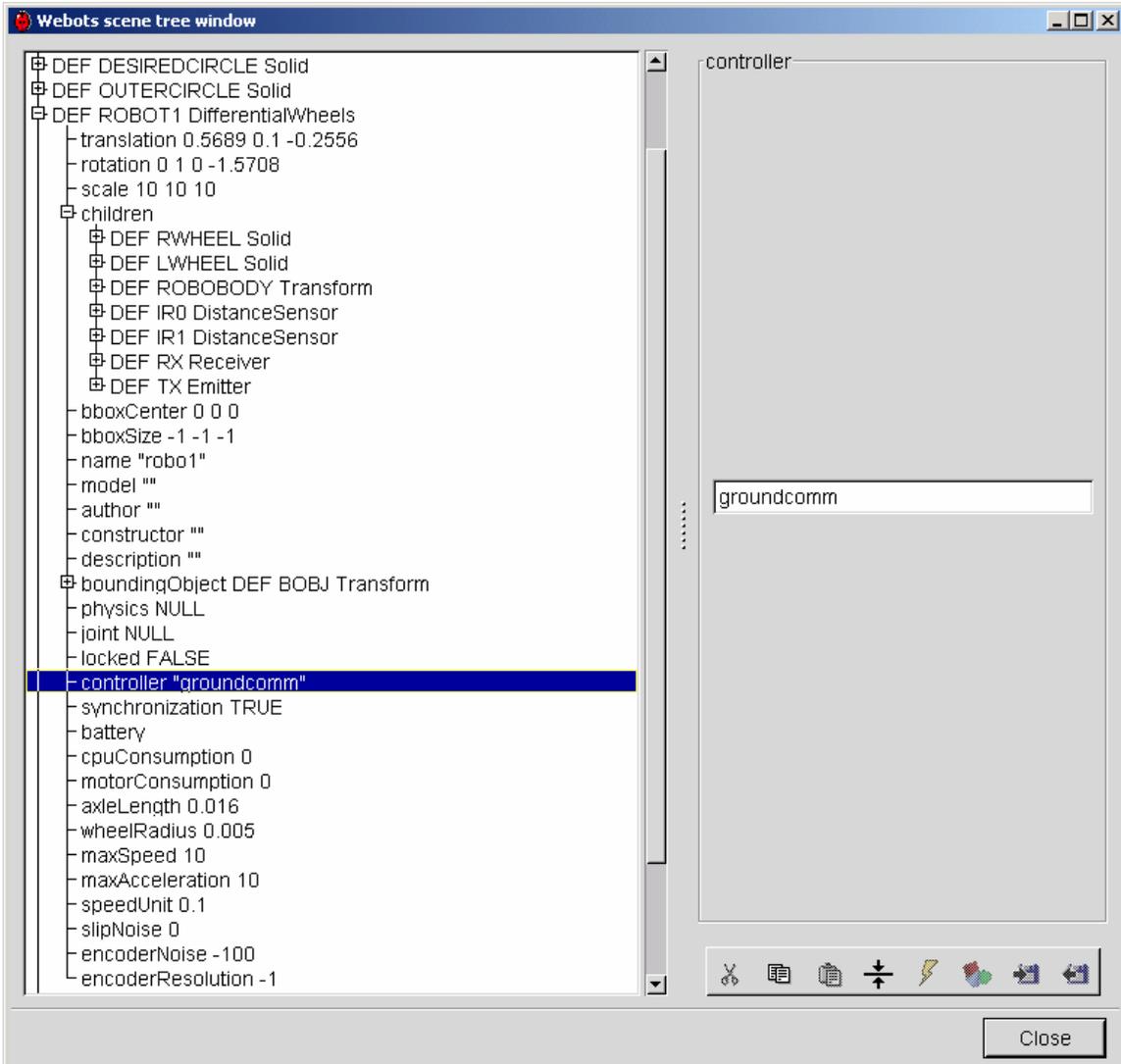


Figure 5.6 Field values for the ROBOT1 DifferentialWheels node

During the execution time, Webots uses multi-threads to execute the individual controller programs linked to the robots simultaneously. In our application, although the same controller program is getting executed multiple times simultaneously, Webots takes care of the conflicts and the overall program execution is smooth.

The parameters required as input are the number of robots ( $N$ ), radius of the circular region and the number of iterations the algorithm takes to spread the robots. Then, for every iteration, depending on the current position of the robot and the position

it needs to move, the change in the position and orientation of the robot is calculated. This information is changed into proper wheel rotations in radians using the axle and wheel lengths of the robot. Based on the robot speed, the time required to move the robot is calculated. This time duration is given as input to the function ‘differential\_wheels\_set\_speed’, which rotates the wheels accordingly. This is repeated for all the iterations and the total time for spreading is calculated and stored.

#### 5.2.4 Results and Discussion

The demonstration shown here uses the parameters listed in table 5.2. The controller program can be downloaded to readymade robots like koala in the future. But care must be taken while modifying the parameters properly for the chosen model of the robot.

Table 5.2 Parameters used in Webots simulation

| Parameter  | Value          |
|--|----------------|
| Number of robots   | 7              |
| Radius of the circle                                     | 4 m            |
| Number of iterations                                     | 60             |
| Time step (when the robot speed is 1 m/s)                | 3 microseconds |
| Wheel Length   | 0.05 m         |
| Axle Length  | 0.16 m         |
| Minimum speed of the robot (used for rotating the robot) | 0.1 rad/s      |
| Maximum speed of the robot (used to move the robot)      | 0.5 rad/s      |

Figure 5.7 shows the spreading of robots as seen in Webots GUI. The total time taken by the robots for spreading is found to be 2013.583 seconds  $\approx$  33 minutes 33 seconds. Spreading time can be reduced by increasing the speed of the robot. But it must be realized that the minimum speed of the robot used while bringing the correct

orientation should be increased with proportionate increase in the size of the robot. Otherwise, it will result in loss of accuracy of positions while moving the robot.

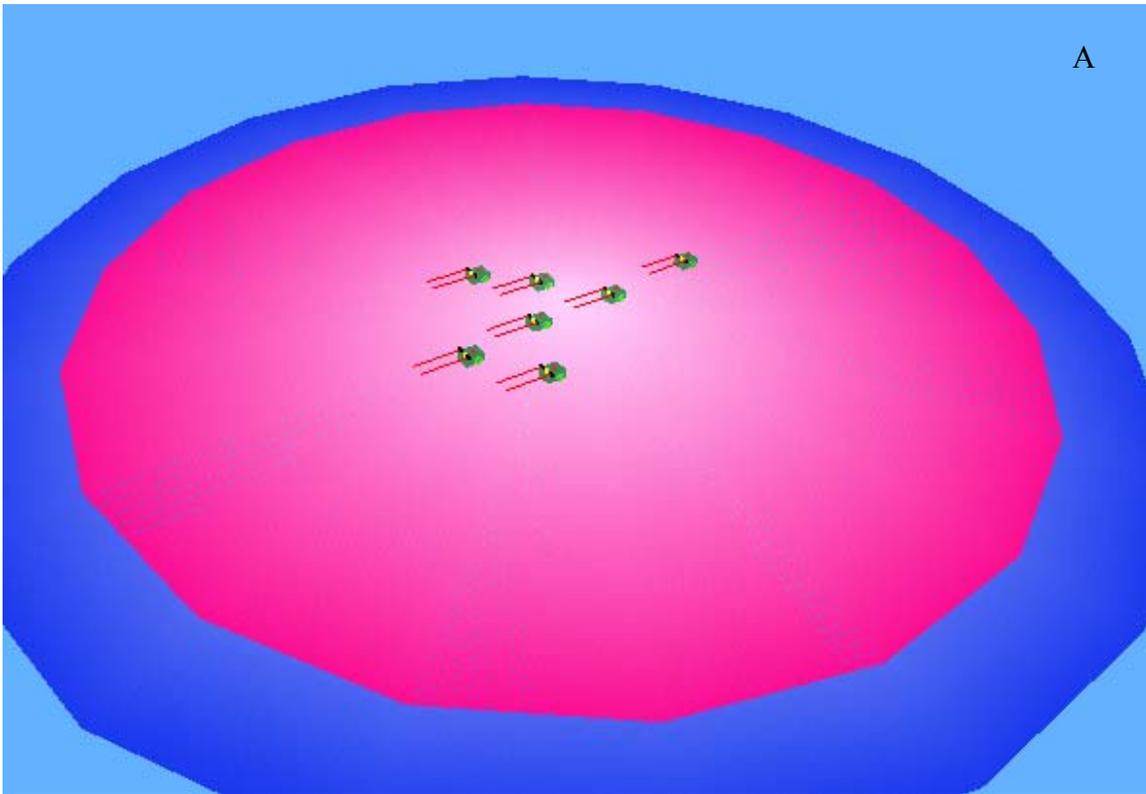


Figure 5.7 Webots simulation on spreading of robots. A) Initial position of robots. B – E) Robots spreading slowly. F) Final position of robots.

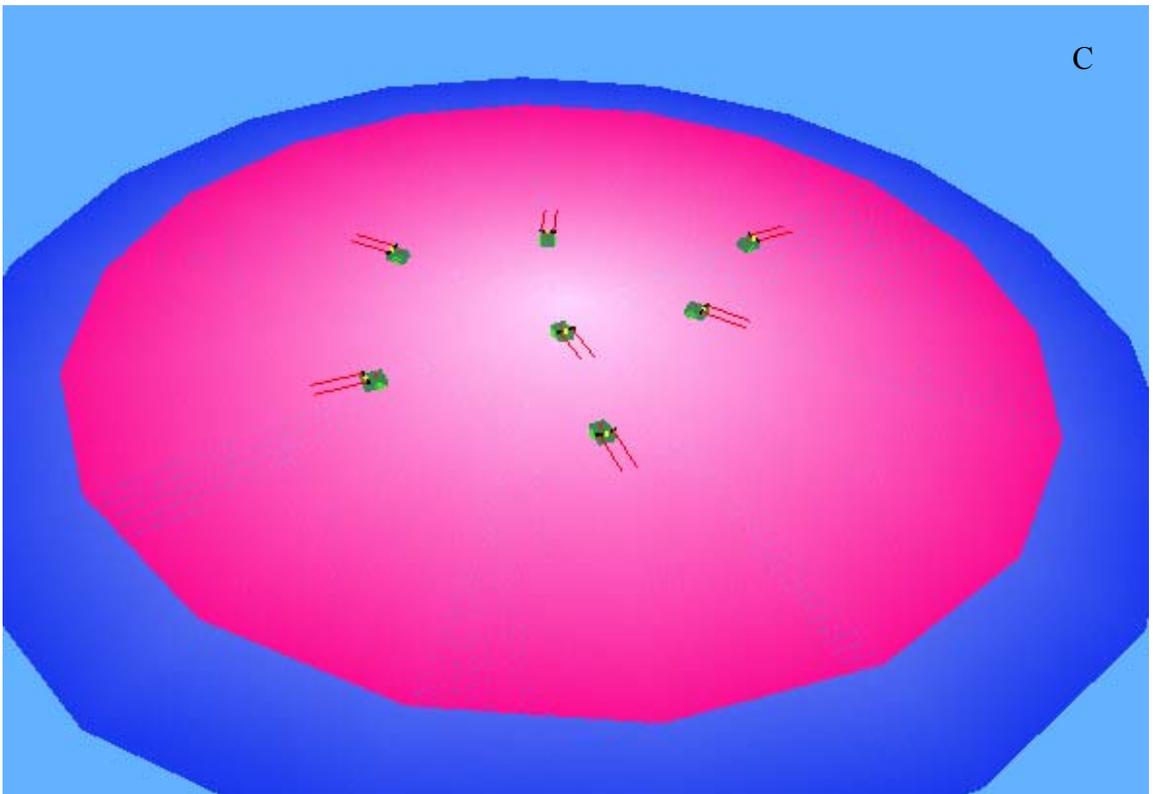
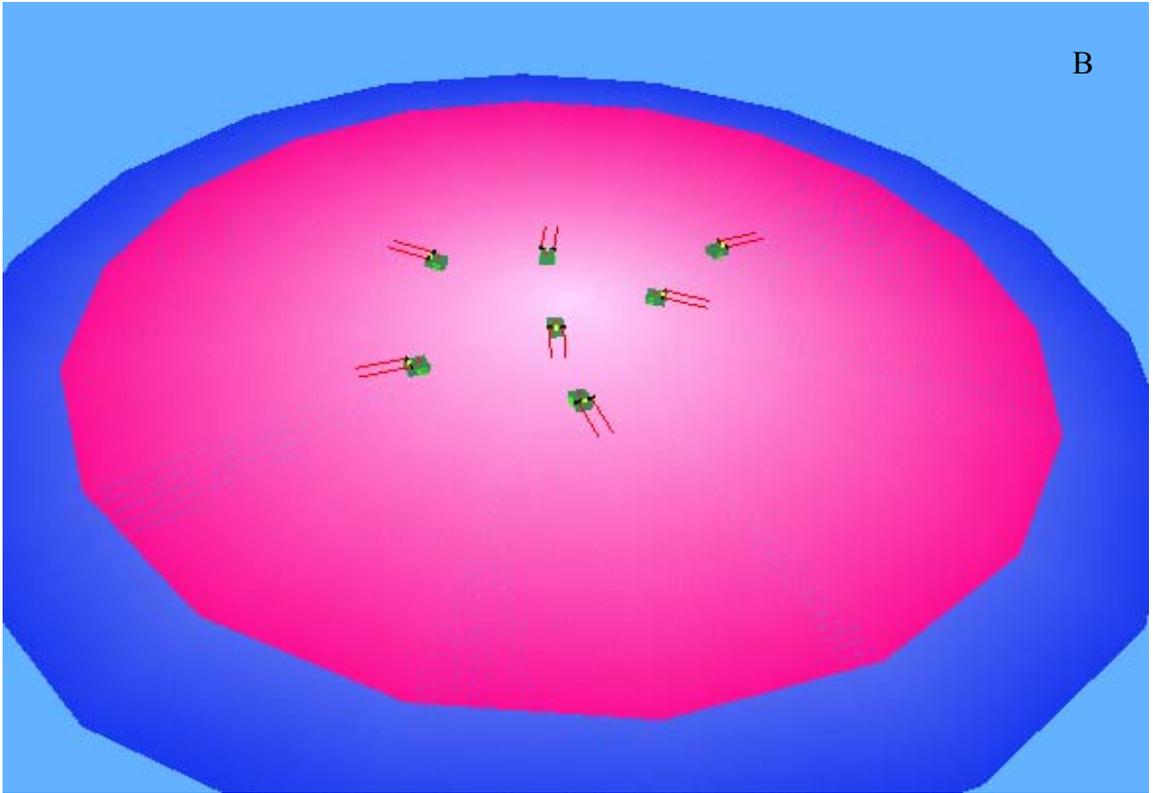


Figure 5.7 Continued

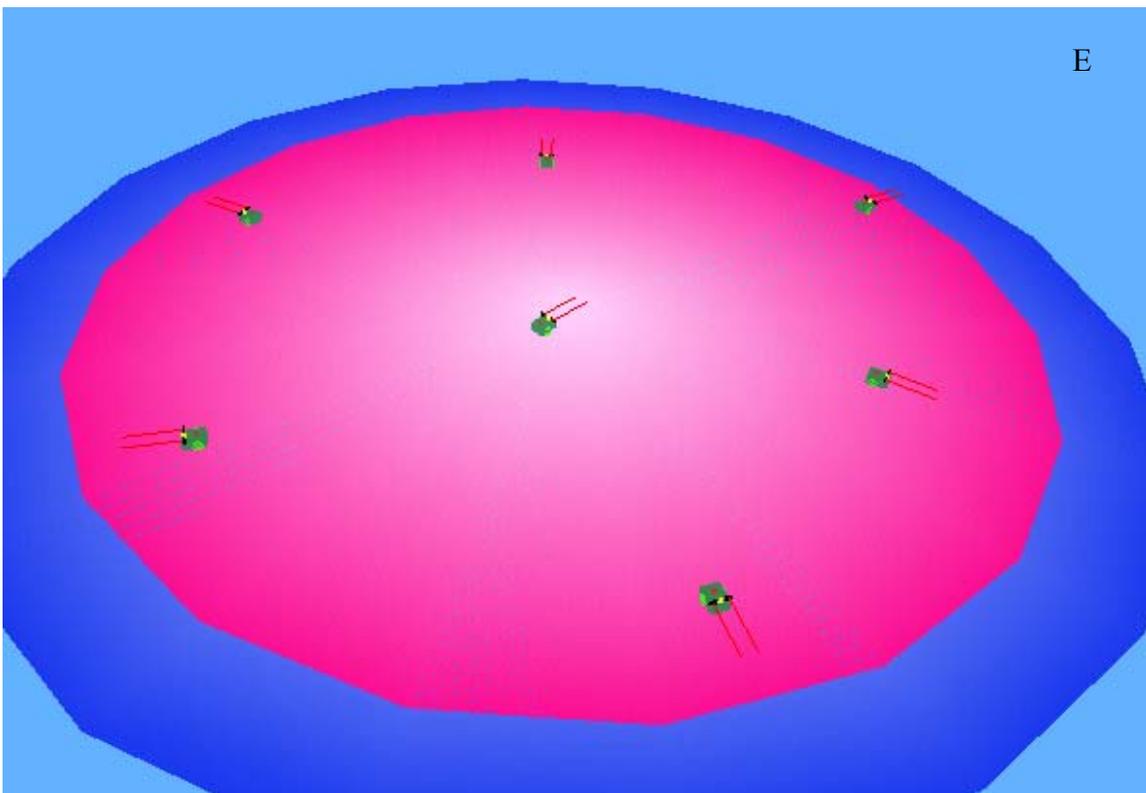
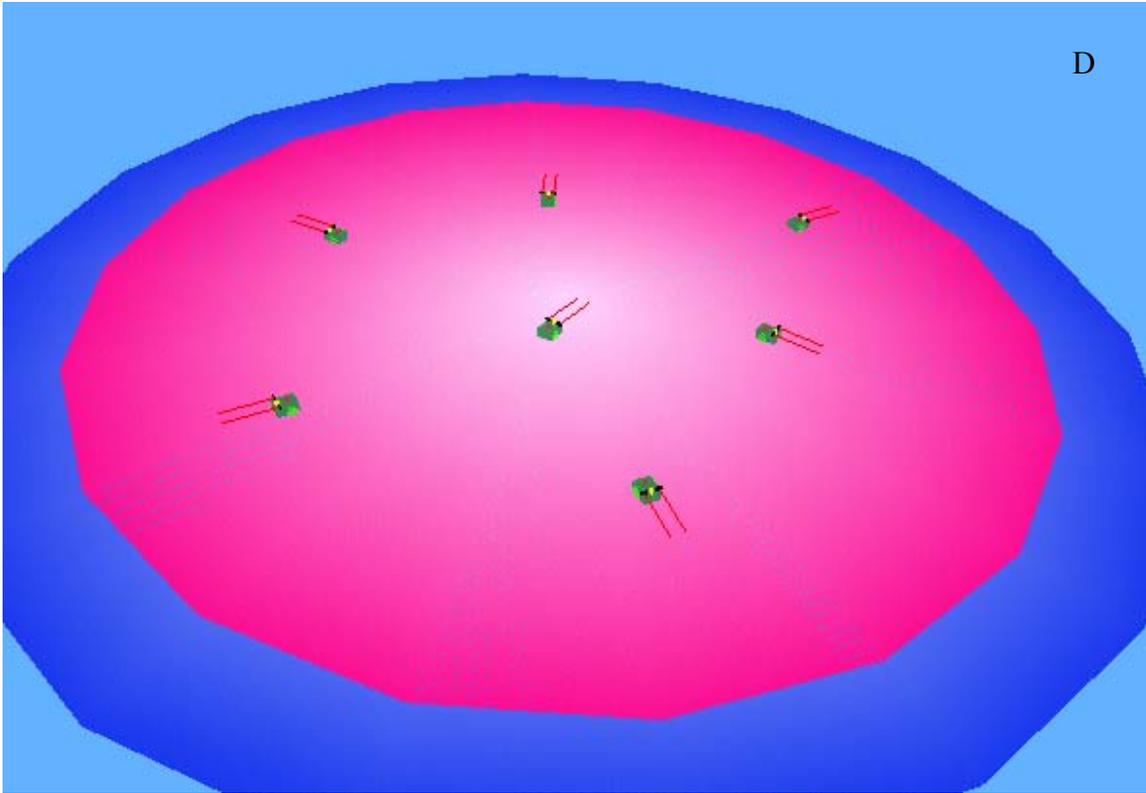


Figure 5.7 Continued

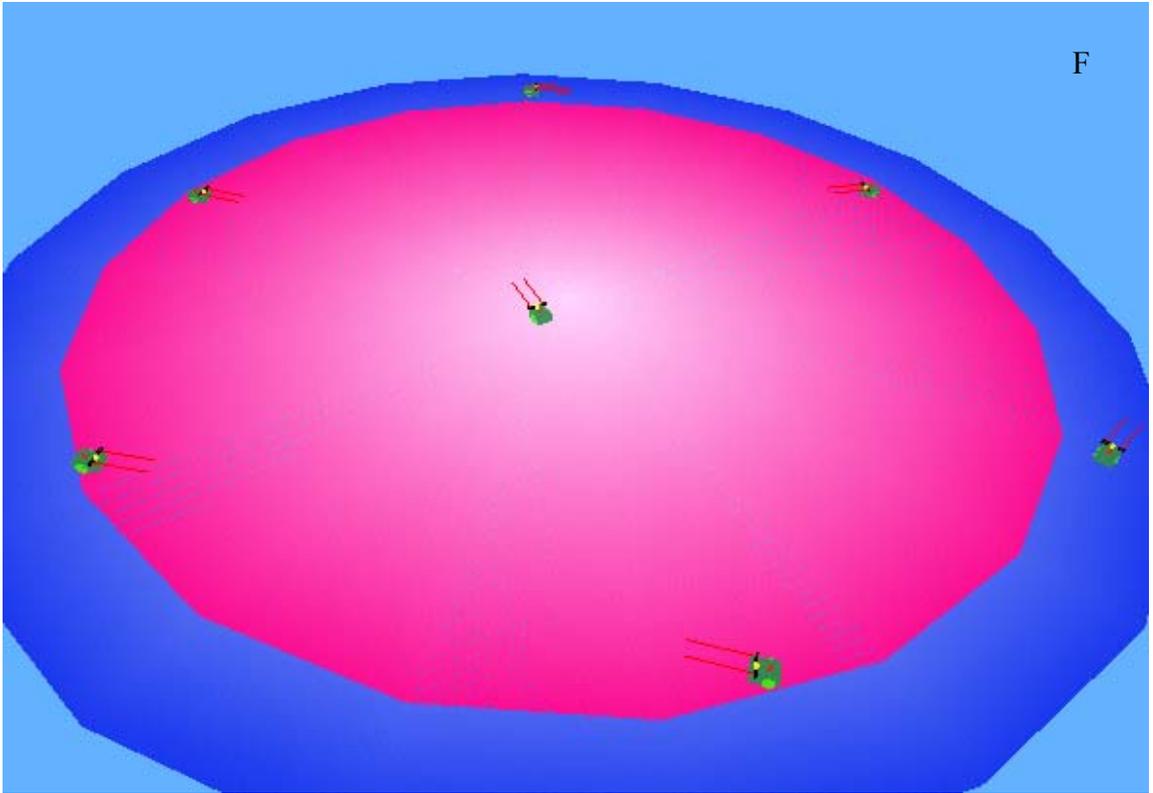


Figure 5.7 Continued.

## CHAPTER 6 CONCLUSION AND FUTURE WORK

### 6.1 Conclusion

Research on multiple agents and collective robotics is continuously growing in its popularity due to the various new applications they offer. The focus is currently on designing effective and efficient self-organizing algorithms that would enable a group of agents to accomplish a task collectively. This requires communication and cooperation between the individuals of the group.

In this work, a decentralized self-organization algorithm based on particle interactions through a predefined interaction law has been proposed. The approach is based on the recently developed particle interaction and information particle principles in adaptation. In this approach, each agent is considered an information particle, which emanates a potential field that allows communication with the other agents in the collective. We presented numerous simulations to illustrate how the robots organize themselves in a specific formation and spread over a given region. We also demonstrated the superiority of our algorithm over the nearest neighbor technique in terms of the speed of spreading.

A target tracking algorithm, which is also decentralized in its control, has been developed and the results have been demonstrated. The proposed self-organization and target tracking algorithms are tailored to be cost efficient since:

1. Each robot needs to have a simple transmitter – receiver pair for communication.

2. The circuit complexity of the receiver does not increase with increasing number of robots.
3. IPIM brings the possibility to implement the algorithms with no absolute positioning mechanisms on any of the robots.
4. Inter-robot collision avoidance is naturally built-in although other obstacle avoidance techniques would be required for practical applications.

Few implementation issues have been addressed and the suggested solutions are elaborated. The self-organization algorithm has been simulated in Webots, to facilitate practical implementation of the proposed system.

## **6.2 Future Research**

We recognize that this study opened up a lot of avenues for future explorations. The communication scheme must be simple to implement and resistant to noise and jamming. As already explained, the first scheme of allowing robots to transmit at different frequencies is simple to implement but not resistant to jamming. On the other hand, the use of PN sequences brings security for transmitted signals while at the same time, making the receiver complex. Hence, a better communication scheme is required.

Quantization errors can be an issue in real-time implementation of the algorithm. The magnitude of the error depends on the particular robot selected for implementation. The minimum speed in which the robot can rotate and the accuracy of the time steps contribute to quantization errors. A study on quantization errors can be conducted before any real-time implementation.

Although Webots can act as a separate simulation platform, due to few constraints in Webots 3.0, the software is now used only as a visualization tool. Cyberbotics has promised to implement the requested corrections and improvements in Webots 4.0. Such a development would ignite the process of implementing target tracking algorithm and

incorporating obstacle avoidance algorithms in Webots, thus making the controller complete. Finally, the real-time implementation of the algorithm on any available robots can be carried out after a thorough study of the behavior of the algorithm in Webots.

## LIST OF REFERENCES

- [Bat02] M.A. Batalin, G.S. Sukhatme, "Spreading Out: A Local Approach to Multi-robot Coverage," Proceedings of DARS'02, pp.373-382, Fukuoka, Japan, 2002.
- [Bon99] E. Bonabeau, M. Dorigo, G. Theraulaz, *Swarm Intelligence: From Natural to Artificial Systems*, Oxford University Press, New York, NY, 1999.
- [Cal02] B. Calverley, 2002, "Swarms Of Tiny Robots To Monitor Water Pollution," Uniscience News Net, Cape Coral, FL, <http://unisci.com/stories/20021/0114026.htm>, last accessed in July 2003.
- [Cap01] G. Caprari, 2001, "Alice Soccer," Autonomous Systems Lab, The Swiss Federal Institute of Technology, Lausanne, Switzerland, <http://dmtwww.epfl.ch/isr/asl/systems/AliceSoccer.html>, last accessed in July 2003.
- [Che94] Q. Chen, J.Y.S. Luh, "Distributed Motion Coordination of Multiple Robots," Proceedings of IROS'94, pp. 1493-1500, Munich, Germany, 1994.
- [Cri02] M. Crichton, *Prey*, HarperCollins, New York, NY, 2002.
- [Cyb03a] Cyberbotics, 2003, "Webots 4 Overview," Cyberbotics, Lausanne, Switzerland, <http://www.cyberbotics.com/products/webots/webots4.html>, last accessed in July 2003.
- [Cyb03b] Cyberbotics, 2003, "Webots User Guide," Cyberbotics, Lausanne, Switzerland, <http://cyberboticspc1.epfl.ch/cdrom/common/doc/webots/guide/guide.html>, last accessed in July 2003.
- [Eis02] G.R. Eisler, J.L. Dohner, B.J. Driessen, J. Hurtado, "Cooperative Control of Vehicle Swarms for Acoustic Target Recognition by Measurement of Energy Flows," SCOT Fifth International Symposium – Technology and the Mine Problem, Monterey, CA, 2002.
- [Erd02] D. Erdogmus, J.C. Principe, L. Vielva, D. Luengo, "Potential Energy and Particle Interaction Approach for Learning in Adaptive Systems," Proceedings of ICANN'02, pp. 456-461, Madrid, Spain, 2002.

- [Far92] D.J. Farmer, A. Belin, “Artificial Life: The Coming Evolution,” in *Artificial Life II*, C.G. Langton, C. Taylor, J.D. Farmer, S. Rasmussen Editors, Proceedings of Santa Fe Institute Studies in the Sciences of Complexity, vol. X, pp. 815, Redwood City, CA, 1992.
- [Fra02] W.R. Franklin, “PNPOLY – Point Inclusion in Polygon Test,” Department of Electrical, Computer and Systems Engineering, Rensselaer Polytechnic Institute, Troy, NY,  
<http://www.ecse.rpi.edu/Homepages/wrf/research/geom/pnpoly.html>, last accessed in July 2003.
- [Fre02] J. Fredslund, M.J. Mataric, “A General, Local Algorithm for Robot Formations,” *IEEE Transactions on Robotics and Automation*, Vol. 18, No. 5, pp. 837-846, 2002.
- [Gen92] V. Genovese, P. Dario, R. Magni, L. Odetti, “Self-Organizing Behaviour and Swarm Intelligence in a Pack of Mobile Miniature Robots in Search of Pollutants,” Proceedings of IROS’92, pp. 1575-1582, Raleigh, NC, 1992.
- [Hay94] S. Haykin, *Communication Systems*, 3<sup>rd</sup> ed., John Wiley & Son, Inc., New York, NY, 1994.
- [Ijs01] A. J. Ijspeert, A. Martinoli, A. Billard, and L. M. Gambardella, “Collaboration through the Exploitation of Local Interactions in Autonomous Collective Robotics: The Stick Pulling Experiment,” *Autonomous Robots*, vol.11, No.2, pp. 149-171, 2001.
- [Kub00] C.R. Kube, E. Bonabeau, “Co-operative Transport by Ants and Robots,” *Robotics and Automation Systems*, vol. 30, pp. 85-101, 2000.
- [Mil90] D.P. Miller, “Multiple Behaviour-Controlled Micro-Robots for Planetary Surface Missions,” Proceedings of IEEE Int. Conf. Systems, Man and Cybernetics, pp. 281-292, Los Angeles, CA, 1990.
- [Par67] E. Parzen, “On Estimation of a Probability Density Function and Mode,” *Time Series Analysis Papers*, Holden-Day, Inc., CA, 1967.
- [Pri00] J.C. Principe, D. Xu, J. Fisher, “Information Theoretic Learning,” in *Unsupervised Adaptive Filtering*, S. Haykin Editor, vol I, pp. 265-319, John Wiley & Son, Inc., New York, NY, 2000.
- [Ren70] A. Renyi, *Probability Theory*, American Elsevier Publishing Company Inc., New York, NY, 1970.
- [Res97] M. Resnick, *Turtles, Termites and Traffic Jams: Explorations in Massively Parallel Microworlds*, MIT Press, Cambridge, MA, 1997.

- [Spe02] E. Specht, 2002, "The Best Known Packings of Equal Circles in the Unit Circle," Packomania, <http://hydra.nat.uni-magdeburg.de/packing/cci/cci.html>, last accessed in July 2003.
- [Sti93] D.J. Stilwell, and J.S. Bay, "Toward the Development of a Material Transport System using Swarms of Ant-like Robots," Proceedings of ICRA'93, pp. 766-771, Atlanta, GA, 1993.
- [Sug90] K. Sugihara, and I. Suzuki, "Distributed Motion Coordination of Multiple Mobile Robot," Proceedings of ISIC'90, pp. 138-143, Philadelphia, PA, 1990.
- [Tho03] R. Thogulua, D. Erdogmus, J. Principe, "Self-Organization of Multiple Agents using Information Theoretic Interactions," Proceedings of ICONS'03, pp. 574-579, Faro, Portugal, 2003.
- [Tur98] R.M. Turner, E.H. Turner, "Organization and Reorganization of Autonomous Oceanographic Sampling Networks," Proceedings of ICRA'98, pp. 2060-2067, Leuven, Belgium, 1998.
- [Uns94] C. Unsal, J.S. Bay, "Spatial Self-Organization in Large Populations of Mobile Robots," Proceedings of ISIC'94, pp. 249-254, Columbus, OH, 1994.
- [Whi98] T. White, B. Pagurek, "Towards Multi-Swarm Problem Solving in Networks," Proceedings of ICMAS'98, pp. 333-340, Paris, France, 1998.
- [Zha02] Y. Zhang, A. Martinoli, 2002, "Swarm Intelligence and Traffic Safety," Center for Neuromorphic Systems Engineering, California Institute of Technology, Pasadena, CA, <http://www.cnse.caltech.edu/Research02/reports/zhang1full.html>, last accessed in July 2003.
- [Zha03] F. Zhang, M. Goldgeier and P.S. Krishnaprasad, "Control of Small Formations using Shape Coordinates," Proceedings of ICRA'03, Taipei, Taiwan, 2003.
- [Zuf02] J.C. Zufferey, 2002, "Evolution of Cooperation and Division of Labor in Artificial Ants," Autonomous Systems Lab, The Swiss Federal Institute of Technology, Lausanne, Switzerland, <http://asl.epfl.ch/index.html?content=education/courses/BioinspiredAdaptiveMachines/zufferey/>, last accessed in July 2003.

## BIOGRAPHICAL SKETCH

Rajvignesh Thogulua Chandrasekar got his Bachelor of Engineering degree in electronics and communication engineering from Thiagarajar College of Engineering, Madurai Kamaraj University, Madurai, India, in May 2001. He joined the University of Florida for his master's in fall 2001 and has been working at the Computational NeuroEngineering Laboratory under the guidance of Dr. Jose C. Principe from spring 2002. His interests are in digital signal processing, digital communication system design, information theory and collective robotics.