SELF-ORGANIZATION IN LARGE POPULATIONS OF MOBILE ROBOTS

by

Cem Ünsal

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science

in

Electrical Engineering

APPROVED

J. S. Bay, Chairman

W. T. Baumann

H. F. Van Ladingham

May 1993

Blacksburg, Virginia
SELF-ORGANIZATION IN LARGE POPULATIONS OF MOBILE ROBOTS

by

Cem Ünsal

J. S. Bay, Chairman
Electrical Engineering

(ABSTRACT)

A homogeneous population of robots described as an Army-ant swarm is to be realized for material transportation. Robots envisioned in the Army-ant scenario are relatively small, independent autonomous mobile robots which can cooperatively carry palletized loads. In this thesis, the agents are treated as a self-organizing system of moving points. This characteristic makes the Army-ant swarm a modular, adaptive and dynamic system.

Several algorithms for spatial self-organization of the robots are given. Self-organizing agents can arrange themselves geometrically in two- and three-dimensional space using only local information about teammates. The method is a distributed one: each agent uses only the information obtained by its own sensors. Algorithms are based on feasible assumptions. It is also shown possible to divide such a population into different groups around goals by communicating minimal data. Data transfer has a broadcast characteristic.
Behavioral self-organization in the Army-ant scenario is also investigated. Activation and inhibition relations between robots determine the behavior (position in a behavioral space) of the agents, while in spatial self-organization force fields are in effect. Several problems which may be encountered and the solution to some of these problems are outlined. Methods for communication and cooperative decision systems —such as coupled van der Pol oscillators— in finding and carrying the pallets are proposed. Sensors and communication systems which may be used in the Army-ant scenario are also briefly discussed.
To Aydın & Ayhan Ünsal,
"full-time parents" since November 1967
Acknowledgements

I wish to express my gratitude to a number of people who became involved with this thesis, one way or another. My thesis advisor, Dr. John S. Bay, a true gentlemen whose suggestions led me throughout this research, has always been willing to answer my questions. Dr. William T. Baumann and Dr. Hugh F. VanLandingham, who were also my instructors in the past two years, have both been kind enough to serve as members of my advisory committee. Han Kılıççöte and Ergin Güney, two exceptional friends, were so gracious in providing me some of the tools I needed for research and documentation. Aylin Barlas, a fellow graduate student who certainly understood what I have been through, was always available for an irrelevant question (or for a visit to Capri Twin). Also Elif Evin, *manita extraordinaire*, was always with me no matter how dubious my decisions were. And I believe I owe many thanks to my parents Ayhan & Aydı̈n Ünsal who were very supportive of my academic venture.
# TABLE OF CONTENTS

Abstract  
ii

Acknowledgments  
v

Table of Contents  
vi

List of Figures  
ix

1. Introduction  
1
   1.1 Motivation  
1
   1.2 Context  
2
   1.3 Scope and Structure of Thesis  
4

2. Review of Literature  
5
   2.1 Self-Organization  
5
      2.1.1 Definition  
5
      2.1.2 Characteristics of Self-Organizing Systems  
6
      2.1.3 Advantages of Self-Organization  
7
   2.2 Natural Systems  
8
      2.2.1 Wasp Colonies  
8
      2.2.2 Schools of Fish  
9
      2.2.3 Flocks of Birds  
10
      2.2.4 Termites  
11
      2.2.5 Army Ants  
12
   2.3 Multiple Mobile Robots  
13
      2.3.1 Autonomous Mobile Robots  
13
         2.3.1.1 Subsumption Architecture  
13
         2.3.1.2 Autonomous Navigation  
14
      2.3.2 Multirobot Systems  
16
   2.4 Learning and Decision-Making  
18
      2.4.1 Neural Networks and Fuzzy Systems  
18
      2.4.2 Unsupervised Learning: Self-Organizing Maps  
20

3. Spatial Self-Organization  
21
   3.1 Formation of a Circle Around a Goal  
21
   3.2 Alterations and interpretations of the algorithm  
23
3.2.1 Uniform Distribution Inside a Circle 23
3.2.2 Extension to 3-D 23
3.3 Formation of a Paraboloid 24
3.4 Multigoal situations 26
3.4.1 Introduction of "switching distance" $d_s$ 27
3.4.2 More intelligent agents: Increasing communications 28
3.4.3 Ideal case 30
3.5 Other issues 31
3.5.1 Many-body physics 31
3.5.2 Explicit Communications 32

4. Behavioral Self-Organization 64
4.1 Behavioral Model of Army-ant Robots 64
4.2 System-level analysis of Army-ant problem 65
4.2.1 Finding pallets in the warehouse 65
4.2.2 Navigating Aisles and Collision Avoidance 68
4.3 Team Coordination 69
4.3.1 Leader Characteristics 69
4.3.2 A Collective Behavior Example: Distributed Load Bearing 69
4.3.3 Carrying the pallets 73
4.3.4 Other Issues 74
4.3.4.1 Dispersal and Rearrangement to Other Teams 74
4.3.4.2 Battery Recharge 74
4.3.4.3 Complexity 74

5. Technical Assessment 96
5.1 A Communication Method: Binary Coded Signals 96
5.2 Sensors 97

6. Conclusions and Future Research 100
6.1 Conclusions 100
6.2 Areas of Further Development 101

Appendix 103
A. Programs 103
A.1 Algorithm Form-Circle 103
A.2 Algorithm Fill-Circle 104
A.3 Algorithm Form-Sphere 105

vii
A.4 Algorithm Form-Paraboloid 106
A.5 Algorithm Switching-distance 108
A.6 Algorithm Family-values 110
A.7 Algorithm Find-pallet 114
B. Matlab/Simulink Simulation Snapshots 117

Bibliography 124
Vita 128
# LIST OF FIGURES

| Figure 1.1 | Material transport using Army-ant robots | 3 |
| Figure 3.1.1 | Plot of the speed toward the goal as a function of distance to goal | 33 |
| Figure 3.1.2 | Plot of the regions defined for velocity/force vector functions | 33 |
| Figure 3.1.3 | Algorithm for formation of a circle around a goal | 34 |
| Figure 3.1.4 | Initial (x) and final positions of the agents for $t_f=50$ with $d=80$ | 35 |
| Figure 3.1.5 | Positions after 100 more time steps with same parameter values | 36 |
| Figure 3.1.6 | Initial (x) and final positions of the agents for $t_f=50$ with $d=80$ | 37 |
| Figure 3.1.7 | Positions after 100 more time steps with same parameter values | 38 |
| Figure 3.1.8 | Initial (x) and final positions for $t_f=200$ with $d=250$ | 39 |
| Figure 3.2.1 | Algorithm for distributing agent uniformly inside a circle | 40 |
| Figure 3.2.2 | Initial (x) and final positions for $t_f=50$ with $d=100$ | 41 |
| Figure 3.2.3 | Positions after 50 more time steps with same parameter values | 42 |
| Figure 3.2.4 | Initial (x) and final positions for $t_f=200$ with $d=100$ | 43 |
| Figure 3.2.5 | Initial (x) and final positions for $t_f=200$ with $d=100$ | 44 |
| Figure 3.2.6 | Final position of 30 agents around the beacon for $t_f=300$ and $d=150$ | 45 |
| Figure 3.2.7 | Final position of 30 agents around the beacon for $t_f=150$ and $d=100$ | 46 |
| Figure 3.3.1 | Definition of the imaginary coordinate axis | 47 |
| Figure 3.3.2 | 2-D representation of the regions defined in the algorithm | 47 |
| Figure 3.3.3 | Flowchart of the algorithm for formation of a paraboloid | 48 |
| Figure 3.3.4 | Force fields defined in the algorithm (Velocity functions are constant in all regions) | 49 |
| Figure 3.3.5 | Initial (top row) and final positions of 30 agents forming a paraboloid with $t_f=600$ | 50 |
| Figure 3.3.6 | Initial (top row) and final positions of 20 agents forming a paraboloid with $t_f=600$ | 51 |
| Figure 3.4.1 | Initial (x) and final positions for $t_f=150$: All agents have moved toward closest beacon | 52 |
| Figure 3.4.2 | Interpretation of forbidden region defined by $d_s$ | 53 |
| Figure 3.4.3 | Switching distance $d_s$ is decreasing with time | 53 |
| Figure 3.4.4 | Flowchart for use of switching distance $d_s$ | 54 |
Figure 3.4.5  Definition of switching distance and resulting final positions at $t_f=300$ (Initial positions defined by x's)  
Figure 3.4.6  Definition of switching distance and resulting final positions at $t_f=300$. (Similar function for $d_y$, different initial positions)  
Figure 3.4.7  Definition of piecewise constant switching distance and resulting final positions at $t_f=300$  
Figure 3.4.8  Flowchart of the $f_v$ algorithm  
Figure 3.4.9  Three first agents shifting their ID numbers  

Figure 3.4.10  Initial (x) and final positions for $t_f=200$ and goal allocation of robots (Initial positions and other parameters are the same as the example given in figure 3.4.6.)  
Figure 3.4.11  Initial (x) and final positions for $t_f=100$ and goal allocation of robots (Initial positions and other parameters are the same as the example given in figure 3.4.7.)  

Figure 3.4.12  Initial (x) and final positions for $t_f=100$ and goal allocation of robots (Initial positions and other parameters are the same as in Figure 3.4.12, only ID numbers of the robots are changed.)  
Figure 3.4.13  Initial and final position of the robots for $t_f=150$ and goal allocation versus time. (Detection region of the agents is 700 units while the beacons and robots are distributed on a 1000x1000 space)  

Figure 4.1.1  Behavior modules of the Army-ant robots  
Figure 4.2.1  Sample warehouse plans  
Figure 4.2.2  Initially only three robots detected the beacon. Three more robots were able to find the beacon by following one of those robots  
Figure 4.2.3  Although initially only one robot (*) detects the beacon, the rest of the pack finds the beacon following that robot, and other teammates  
Figure 4.2.4  One agent searching a beacon in the warehouse  
Figure 4.2.5  One agent searching a beacon in the warehouse  
Figure 4.2.6  Eight agents searching for a beacon  
Figure 4.2.7  First 5 steps and final positions of ten agents searching for a beacon  
Figure 4.2.8  Four typical paths from the simulation with initial conditions given in Figure 4.2.7.
Figure 4.2.9  *Wait-for-me* signal
Figure 4.2.10 Use of range sensors for navigation
Figure 4.3.1 Ring formation and coupling coefficients for eight agents
Figure 4.3.2 Difference between 1st and 4th sensor is quite large while all sensor readings between successive robots are in allowable range
Figure 4.3.3 Coupled VDP oscillator with added feedback gain
Figure 4.3.4 Mapping from force sensor output to feedback gain
Figure 4.3.5 Oscillator outputs (xi) and error signals (εi=x_i-x_{i-1}) of eight agents in ring formation
Figure 4.3.6a Oscillator feedback gains (λ_{ii}) and outputs of eight agents in ring formation
Figure 4.3.6b Oscillator feedback gains (λ_{ii}) and error signals of eight agents in ring formation
Figure 4.3.7a Oscillator feedback gains (λ_{ii}) and outputs of eight agents in ring formation
Figure 4.3.7b Oscillator feedback gains (λ_{ii}) of eight agents in ring formation and error signals of two agents
Figure 4.3.8 Oscillator feedback gains (λ_{ii}) of eight agents in ring formation and error signals of two agents
Figure 4.3.9 Oscillator feedback gains (λ_{ii}) of eight agents in ring formation and error signals of two agents
Figure 4.3.10 Oscillator feedback gains (λ_{ii}) of five agents in ring formation and error signal of an agent
Figure 4.3.11 A small obstacle is encountered during pallet transport
Figure 4.3.12 An example of beacon positioning and priority assignment
Figure B.1 Plot screen from program *Sphere*
Figure B.2 Plot screen from program *Paraboloid*
Figure B.3 *Unmasked* VDP oscillator block with time varying feedback gain.
Figure B.4 Simulink window for coupled Modified VDP Oscillators
Figure B.5 Screen snapshot of the Matlab program *ds*
Figure B.6 Screen snapshot of the Matlab program *famval*
1. Introduction

1.1 Motivation

Multirobot systems are becoming more and more significant in industrial, commercial and scientific applications. The number of robots currently being used in industrial projects is increasing fast. The rate of scientific and industrial development made way for the use of robots in many fields.

Control and communication methods for multiple-robot systems have been investigated by various researchers. Problems such as coordination of multiple manipulators, motion planning and coordination of multirobot systems are generally approached with a central (hierarchical) controller in mind. Until recently, most of the multirobot systems have been "fixed" systems without autonomously moving elements. They may consist of several types of robots or manipulators.

On the other hand, there is extensive research carried out on autonomous mobile robots. Many solutions to problems including path planning and obstacle avoidance were proposed and tested. However, most of the research on autonomous mobile robots was based on a single robot interacting with its environment.

Currently, there is an increasing interest in multiple autonomous mobile robot systems due to their applicability to various tasks such as space missions, operations in hazardous environments, and military operations. Such systems bring in the problems of both multiple robot coordination and autonomous navigation. Again, multiple mobile robots may be controlled by using a hierarchical (central) controller. However, tasks mentioned above obviously require many robots which are able to navigate autonomously. It is difficult to use a central controller or a hierarchical method, sometimes because of the large distances, sometimes due to robustness and versatility problems. The advantages of a decentralized system will be outlined in the next section where we introduce the Army-ant scenario.
1.2 Context

With a name like yours, you might be any shape, almost.
*Through the Looking-Glass, Ch. 6, LEWIS CARROLL*

The Army-ant scenario envisages a large population of identical mobile robots which are able to find and carry a relatively small number of palletized payloads from place to place. The locations of pallets are defined by beacons whose signals can be picked up by robots. Using the beacon signals, robots are able to group around a pallet and *self organize* to lift and carry it to its destination. After set-down, the robots will disperse to continue the operation.

Army-ant robots will be relatively small in size, and individually incapable of carrying the load; but they will be able to act cooperatively as a transporter, similar to ant colonies in foraging activity. We treat the Army-ant robots as a *self-organizing* system, because self-organization—an important characteristic of most insect societies—has many advantages, as we describe in depth in the next chapter. A self-organizing system can change its structure as a function of its experience with the environment, and may accomplish complex tasks with simple individual behavior. Changes in the individual characteristics can influence the overall behavior of the system. On the other hand, the environment may cause the system to generate a different task, without any effect on individual behavior.

Army-ant robots would also have the following characteristics:

- All robots are physically and functionally identical. Therefore, they can be manufactured inexpensively in large numbers, which would be the case.

- This homogeneous population of robots is a *modular* system. Any robot can replace any other robot. In case of failure, the absence of a hierarchical system would prevent total system failure. Furthermore, new agents can be added to the team whenever necessary.

- Army-ant robots are designed for a broad range of tasks, not for a specific task as is the case in current multirobot systems. They can be adapted to various tasks with minimal structural changes.
- Individually, robots have limited capabilities and limited knowledge of the environment. However, as a *swarm*, they can exhibit "intelligent behavior". Simple individual behavior will result in an intelligent swarm behavior provided that some type of direct or indirect communications between agents exists.

- Being a homogeneous and self-organizing population, Army-ant robots form a *dynamic* system. They are able to (cooperatively) adjust the team size to the size of the payload. The capabilities of a team (other than physical abilities such as lifting the pallet) do not depend on the number of agents since there are no hierarchical interactions between agents.

Obviously, most of the characteristics listed above cannot be achieved by using a central controller. The number of agents and the dynamic character of the teams make it impractical. Populations of mobile robots using decentralized control methods have many other applications. Mine sweeping, maintenance work in nuclear power plants, planetary surface missions and multisatellite defense systems are areas where teams of large numbers of autonomous agents are potentially advantageous. Some of the methods we describe for the Army-ant scenario in this thesis are also suitable for the above mentioned applications. The Army-ant approach differs from the ongoing research on multiple mobile robots in its design. We try to realize a complete system designed for a practical task, instead of creating a generic system to study the problems of multiple mobile robot populations.

![Figure 1.1 Material transport using Army-ant robots](image-url)
1.3 Scope and Structure of Thesis

Chapter 1 introduces the Army-ant scenario. The definition and advantages of self-organization, several self-organization examples in nature are given in Chapter 2, as well as previous and related work in the fields of multiple (mobile) robots.

We divide self-organization in Army-ant robots into two chapters: spatial and behavioral self-organizations. Spatial self-organization refers to an reactive method of treating the agents as a many-body system interacting according to specific laws of gravitation. By defining a set of gravitational rules, it is possible to force agents into geometric arrangements, or to divide them into groups/teams. On the other hand, in behavioral self-organization, Army-ant robots' behaviors are defined on a behavioral space, where the whole system's state consists of individual behavior modes of all agents. Changes in this space are due to the activation/inhibition forces generated by robot behaviors, beacons, and environmental conditions. State-space in behavioral self-organization can be "visualized" as a multidimensional space where the dimension is related to the number of behaviors, as opposed to the spatial self-organization dealing with two or three "physical" dimensions.

Chapter 3 deals with spatial self-organization, where geometric arrangement of agents, team formation in two- and three-dimensional spaces, and related assumptions on the knowledge and influence of robotic agents are investigated. A behavioral model of the robots, system-level analysis of the Army-ant problem and several aspects of team coordination are discussed in Chapter 4. As examples, several problems which may be encountered in the Army-ant scenario and solutions to some of these problems are outlined.

Chapter 5 is devoted to technical assessment; necessary devices for communication and sensing are briefly described. Feasibility of their application to our scenario is investigated. Chapter 6 draws conclusions from the work and makes suggestions for further research. Partial listings of the source code and screen snapshots of the simulation programs are given in the Appendix. The works we cited in the text are listed in the Bibliography.
2. Review of Literature

In this Chapter, we give an overview of topics related to this thesis. Some topics discussed here will be mentioned in more detail than others, since they are more closely related to our research. We organize this chapter as a collection of short introductions. Instead of simply stating the literature, we will comment on some aspects of the emphasized topic. Furthermore, we will also investigate possible future applications and state the differences from the approach taken in this work, whenever necessary. It is our intention to keep this chapter as interesting as possible.

2.1 Self-Organization

2.1.1 Definition

The term "self-organization" (or "self-organizing system," to be precise) is first defined by Farley and Clark of Lincoln Laboratory in 1954 [39]:

A self-organizing system is a system that changes its basic structure as a function of its experience and environment.

This definition clearly relates to today's "hot" topics of adaptive control, neural networks and genetic algorithms. We will also dwell upon neural networks and unsupervised learning briefly at the end of this chapter.

A self-organizing system has three main characteristics (or functions): affect, telos\footnote{Ultimate end, in Greek; the meaning here is 'evaluation.'} and effect [33]. To explain these three functions, we will use our Army-ant scenario as an
example: A robotic agent in a multirobot network observes its environment (affect), using these observations decides what to do next (telos) and executes according to its decision (effect). In a population of Army-Ant robots dispersed in an area where several loads are located, agents recognize the situation by observing the signals coming from other agents and "goals" (affect), compute the direction of movement at each step (telos) and move (effect). On a large scale, the whole population receives signals (affect) and, guided by decision algorithms (telos), acts (effect).

One encounters self-organization in many fields. Biology (insect societies, ecosystems), chemistry (thermodynamics), computer science (decision algorithms, neural networks and fuzzy logic), geology (tectonic movements), sociology (communication and migration) and economy (socio-spatial systems) are some areas where self-organizing systems are encountered often.

Nicolis and Prigogine [25], defining self-organization in nonequilibrium systems, stated that self-organization emphasizes the large scale coordination processes at many levels. Nonlinear processes and nonequilibrium conditions play a significant role in these processes.

Kauffman [19] believes that self-organization, an "inherent property of some complex systems," may be responsible for biological evolution along with selection. His computer models suggest that certain complex biological systems tend toward self-organization.

2.1.2 Characteristics of Self-Organizing Systems

Self-organization has three important characteristics. First, a self-organizing system can accomplish complex tasks with little and simple individual behavior. Secondly, a change in the environment may influence the same system to generate a different task, without any change in the behavioral characteristics. Finally, any small differences in individual behavior can influence the collective behavior of the system Therefore, social complexity of the system is compatible with simple and identical individuals, as long as communication among the members can provide the necessary amplifying mechanism. For example, as we mention in Chapter 4, our "swarm" of robotic agents gathering under a palletized load, can change their operation "phase" by a signal from any member of the group. This can be achieved by defining specific communication mechanisms.
In a self-organizing system, individual behavior need not be changed in order to have different collective behavior. This characteristic of self-organization is highly advantageous for a swarm of robots since simple individual behavior can be achieved with relatively cheap and simple designs.

2.1.3 Advantages of Self-Organization

What makes a self-organizing system advantageous over a preprogrammed, deterministic organization is that the former is based on individuals/agents requiring simple programming and autocatalytic communications. A large number of individuals can be coordinated into a collective system interacting with environment. And as stated above, this collective behavior will have an "adaptive" character. Such a system is therefore simple, reliable and adaptive while only a few basic rules are needed to define individual behavior and interactions.

Some animal societies and particularly social insects can achieve complex tasks that are impossible to complete individually. We will state some examples in the next section. On the other hand, simplicity (and homogeneity) of individual agents on a robotic swarm decreases the cost of production and the likelihood of the breakdown.

Furthermore, breakdown of one agent will not effect the activity of the whole robotic team, which may not be the case in a deterministic system such as a production chain. The simplicity would also be in software as well as in hardware. In a deterministic system, programs are highly complex, in order to operate in every possible situation harmful to the system, and it is still impossible to foresee them all. However in a self-organizing system, simpler programs can operate in unforeseen situations and adapt to changing conditions. For these reasons, self-organizing algorithms which have only partial (local) knowledge of the network are used to manage data networks of large numbers of users.

Advantages of self-organization and the efficiency in self-organizing behavior of some animal societies, as they became known, caused interest in the use of self-organization in robotics. To quote Deneubourg and Goss [11]:

Engineers are often, consciously or not, prisoners of the Cartesian and scientific positivist philosophy that dominates their education, and it is therefore not surprising that robot designers have chosen to develop expensive, complicated, deterministic robots, tailored to specific problems. We can now propose the
completely different approach of using teams of simple, interacting robots to perform a wide range of tasks.

As engineering society becomes more interested in adaptive, decision-making systems such as neural nets, fuzzy logic, etc., it is obvious that this approach will draw more attention in the future.

2.2 Natural Systems

Some animal societies such as colonies of ants and bees, flocks of birds, schools of fish, can be an inspiring model for a self-organizing robotic network. In this section, we will summarize some interesting characteristics of above-mentioned animal societies.

2.2.1 Wasp Colonies

Deneubourg and Goss [11, 37], authors of many articles about self-organization in insect societies, stated that the task organization in a colony (of bees and/or ants) appears to be a distributed function which does not require a central organizer. Elementary processes in insect societies can potentially be applied in coordination and self-organization of robotic "swarms" formed by agents with simple local computational properties.

It has been shown in [37] that elementary rules of individual behavior makes it possible for a society of Polistes wasps to make efficient decisions when certain types of external constraints are encountered, as well as to create complicated patterns. This, as we defined in the previous section, is an important characteristic of self-organizing systems. The model discussed in [37] is based on two different characteristics of individual interactions: (i) hierarchical and (ii) tropic.

Hierarchical interactions guarantee that, when two bees interact, one assumes a dominant role while the other is submissive. Some form of hierarchical interaction is also used in this research, as we attempt to define a coupled system of nonlinear oscillators in order to create a simple decision system in Chapter 4.

2 The hierarchical rank is related to the degree of mobility on the (honey) comb.
The thropic type of interaction controls the relationships between individuals and the environment, especially with the brood. For example, when food supply is insufficient, the intensity of the larval stimulation increases and consequently, the number of foragers also increases. This type of process has also been observed in ants [10, 38].

2.2.2 Schools of Fish

Another interesting self-organized behavior is found in schools of fish. Hundreds of fish, moving like a single organism, can disperse in a quick expansion in case of a danger (in form of a bigger fish probably) and then group again to reform the school. Schooling serves to reduce the risk of being eaten for a fish, since the probability of detection is reduced by forming a school. Also even if a school is detected by a predator, the odds of being eaten is still less for an individual fish [27]3.

Although most work done on schools of fish studied species of fish that are consumed, some predators also form schools. If a member of the school finds food, the other members can take advantage of the find. If the members of the school remain barely in the sight of one another, the search area is at a maximum. Application of this idea to populations of multiple mobile robots for searching pollutants, for planetary missions or for detecting missile launches, is obvious.

Partridge [27] determined an interesting coordination in tuna schools. Tuna schools of 50 or more members sometimes divide into smaller groups which consist of between 10 and 20 fish. These fishes spread out along a curve very similar to a parabola with concave side forward. Although achieving a regular distance between individuals along a parabola is difficult4, that form provides a considerable advantage in hunting. If the parabolic school swims parallel to its axis, any prey reacting to the curved school, will be driven to the focus of the parabola, which is the most convenient place for surrounding the prey.

---

3 Although it might seem that a school made of thousands (or millions) of fishes would be highly visible, the probability of detection of a school is not larger than a single isolated fish. The reason has to do with the optical character of the medium. "Contrast" is highly important for distinguishing an object from the background. Since scattering and absorption of light in water greatly reduces the contrast, maximum visibility distance is 200 m., even in exceptionally clean waters. In the open ocean a predator's chance of finding a school of one thousand fish is slightly greater than its chance of finding one fish.

4 Because distance and angle between each pair are different.
Fish schools do not have a regular geometric form; the structure is loose or probabilistic and it results from each fish's applying a few simple behavior rules. First rule is that each individual maintains an empty space around itself. In general, only one neighbor at a time is at the preferred distance from a particular fish. (In a regular geometric shape, neighboring fishes would be at the same distance.) Fishes also tend to keep their neighborhood at a particular preferred angle with respect to their body angle\(^5\). Most schools of fish are organized on the same lines: preferred distance and angle.

Experiments on pollock [27] showed that vision and lateral lines\(^6\) are two important senses fishes employ to match the speed and direction of other fishes. Blinded fishes and fishes whose lateral lines are removed were able to school. But blinded fishes swam farther from their nearest neighborhoods than pollock's ordinarily do, while fishes with lateral lines removed swam closer to the nearest schoolmate. Only when a fish was both blinded and had had its lateral lines removed it did fail to maintain its position in the school. Vision seemed to provide the "attractive force" between members while lateral lines provided the "repulsive force." Other research suggested that vision take precedence in case of contradictory information.

### 2.2.3 Flocks of Birds

Flocks of birds are organized more or less the same way as the schools of fish. Each member of the flock is attracted to the flock; at the same time, they are repelled from other member in the vicinity by an obstacle avoidance "mechanism." Computer simulations based on three simple rules, could create flocks of birds which seemed to correspond to our notion of what constitutes flock-like motion [30]. In order of precedence, these individual behavior rules are:

i) collision avoidance

ii) velocity matching with nearby flockmates

iii) flock centering in attempt to stay close to nearby flockmates

\(^5\)Preferred distance and preferred angle are statistical values since the spatial relations among fishes change constantly as the school changes its direction and speed.

\(^6\)An organ sensitive to displacement of water.
2.2.4 Termites

Another highly interesting self-organization example is encountered in termites: the periodic assembling of a nest by a population [22]. The nest building behavior of termites consists of several distinct phases of construction. In the first phase, building material are carried into the site and deposited randomly. This phase ends when preferred sites, which are fewer than original deposits, emerge. In the next phase, material buildup continues until deposit sites take the shape of pillars. When pillars reach certain size, third phase of construction starts. Two neighboring pillars mutually bend toward a virtual midpoint. End of the third phase is defined by formation of an arch. And in the final phase, construction of an arching dome that extends from the tops of arches takes place. These phases can be repeated on top of the dome if random deposition of material begins again.

The formation of this complex structure involves pheromones\(^7\). The insects follow two simple rules:

i) move in the direction of strongest smell

ii) deposit where the smell is strongest

Each deposit creates an "aromatic potential field." Because the number of insects is large, the likelihood that an insect will move in the direction of a recent deposit will increase. The more attractive a site becomes because of increasing pheromone concentration, the more frequent the deposits (of material and, therefore, pheromones) on that site, which in turn increases the pheromone concentration. This sequence requires a certain number of insects. Only above a critical number of insects, can the pheromone amplify and become effective, since it has a diffusive character.

When a pillar develops on a site of an original deposit, its uppermost region, being the deposition point, acts as a point attractor for insects. When two pillars are sufficiently close to each other, a virtual saddle point midway between the pillars results. Therefore, insects first approach the saddle point and then converge to one of the pillars from the direction of the other. This behavior leads to the formation of an arch. And the formation of arches, creating new attraction points, can result new saddle points that guarantee the

\(^7\)A chemical substance that is produced by an animal and serves especially as a stimulus to other individuals of the same species for one or more behavioral responses.
formation of a dome. The cycle can repeat when new deposit sites emerge on top of the dome.

2.2.5 Army Ants

Army ants, which are the inspiration for our robotic "swarm," are one of the most "organized" animal societies. In ant societies, activities like selection of most rewarding resource, finding of shortest route between two points, formation of a collective exploratory and exploitation patterns, the generation of synchronized rhythmic activity both in time and space, spatial specialization within a group of individuals without direct communication between them, and the collective construction of regular nest structures are examples of self-organization [11].

Army ants, like other self-organizing insect societies, exist in large colonies. "If 100 ants are placed in a flat surface, they will walk around and around in never decreasing circles until they die of exhaustion. In high number however it is a different story." [15] The collective intelligence of the Army ants is an emergent behavior of the collective communication.

Computer simulation based on nonlinear equations [12] showed that ant colonies are able achieve a collective spatial and temporal structure in raids without global coordination, but instead through the communication between foragers by laying down of trail pheromones, which they also react to. On the other hand, Deneubourg, et al, demonstrated that, although their mathematical model was very crude, certain amount of "noise" during foraging can be advantageous. Their model shows that a given amount of noise in the food recruitment process is needed to optimize food gathering when a multiple and aggregated source situation is present [9]. They also stated that the emergence of error could be regulated by the nature of the system of communication itself. In a similar work [28], it is demonstrated that trail recruitment to newly discovered food sources in ants can be simulated using unspecialized identical units with a very simple program, no memory and a large stochastic behavior as a model. Even when the characteristics of behavior and communication in ants are oversimplified, the mathematical model was able to generate the integrated behavior observed in experiments with ants.
2.3 Multiple Mobile Robots

In this section, some previous work on autonomous mobile robots, multirobot systems and robot behavior will be cited. We will try to highlight important ideas and significant achievements on above-mentioned fields.

2.3.1 Autonomous Mobile Robots

Since autonomous mobile robots are the basic elements of multiple mobile robot populations, we will first dwell upon autonomous mobile robots. Subsumption control architecture and several navigation techniques will be summarized in this subsection.

2.3.1.1 Subsumption Architecture

Subsumption architecture for controlling mobile robots is first introduced by Brooks [7]. In such architecture, layers of control system are built in order to let the robot operate at increasing levels of competence. Layers are made up of asynchronous modules that communicate over low-bandwidth channels. Each module is a simple computational machine, and higher level layers can suppress the output of lower levels (subsumption). But, lower levels continue to function as higher levels, which interfere with their data inputs, are added.

Each level generates a behavior and the competence of the robot is improved by addition of new layers. The subsumption architecture is based on decomposition of a mobile robot in terms of behavior rather than in terms of functional modules\(^8\). Since the overall control system can be viewed as a system of agents acting separately, there is no need for a central control module.

\(^8\)Here is an example decomposition of a mobile robot control system based on task achieving behaviors:

![Diagram](image.png)
An example of subsumption architecture is Squirt, a very small intelligent mobile robot\(^9\). Squirt acts as a bug, hiding in dark corners and venturing out in the direction of noises, only after noises are gone, looking for a new place to hide near where the previous set of noises came from. The most interesting fact about Squirt is the way in which its high-level behavior, mentioned above, emerges from a set of simple interactions with the environment. Squirt's lowest level of behavior causes the robot to search for darkness. The second level of behavior is triggered once a dark spot has been found. Monitoring two microphones, the direction from which the noises come is detected, and when a few minutes of silence follows a sharp pattern of noise, Squirt moves in the direction of the last heard noise, suppressing the desire to stay in the dark. After a time-period, the first level is no longer suppressed and becomes active. This "bug behavior" fits in 1300 bytes of code on an 8-bit microprocessor [8].

The subsumption architecture has also demonstrated robust navigation for mobile robots in dynamically changing environments. Its layered structure is well-adaptable for hardware implementation.

\subsection*{2.3.1.2 Autonomous Navigation}

The most important "function" (or the first layer of a subsumption control architecture) in a mobile robot is the ability to avoid obstacles, as it is in schools of fish and flocks of birds. An autonomous robot recognizes its environment using sensors and decides what to do next based on the sensor data. Rodin and Amin [32] defined the general structure of an intelligent navigational algorithm for solving the problem of real time control in an environment with moving obstacles as follows: it consists of identifier, goal selector and adapter levels.

The identifier constructs a local representation of the surroundings based on information obtained from sensors, and determines the speed of obstacles. Goal selector uses the map and speed of the obstacles and finds a locally optimal collision-free path satisfying other possible conditions. The adapter consists of two subsystems: one for path smoothing to avoid sharp turns and the other for determination of steering command (based on potential field path planning).

\footnote{Squirt weighs about 50 gr. and is about 5/4 cubic inches in volume. It has an 8-bit onboard microprocessor, power supply, three sensors and a propulsion system.}
Problems often encountered in autonomous navigation models are (i) delay in feedback information, (ii) sensor and servo errors, and (iii) limited sensor range [14]. Due to the large amount of computation required to process the sensor data, a delay is expected in obtaining the local map. For Army-ant robots, this would not be a problem since Army-ant scenario does not include any map and/or path finding algorithms. Again sensor and servo errors create a problem for map building robots. Limited sensor range may cause a problem in obstacle avoidance. However, it is possible to overcome this by adjusting the speed of rovers according to the visibility range of the sensors.

On the other hand, Arkin, describing path planning and navigation as a collection of behaviors, use *motor schemas*\(^\text{10}\) to obtain a reactive navigation method for autonomous robots. Motor schemas serve as the basic unit of behavior specification for the navigation; they are concurrent processes that operate in conjunction with associated perceptual schemas and contribute independently to the overall action of the robot [3]. A variant of the potential field method is used to produce the appropriate velocity and steering commands. Motor schemas, such as *move-ahead, move-to-goal, avoid-obstacle*, which can be visualized as vector fields, are represented as asynchronous computing agents in terms of addition and multiplication.

The output of a schema is a single velocity vector derived from a potential field formulation of the forces exerted upon the robot at any particular point in space. The entire potential field is never computed; thus, the computational demand for a single schema is small. The output of each motor schema is combined using vector summation, and then normalized. Arkin's model includes a low-magnitude random vector that changes at random time intervals in order to remove the robot from undesirable equilibrium points that arise when active motor schemas balance each other. Also, gains of schema outputs can be changed (depending on established real-time deadlines for goal attainment) in order to allow a blocked robot to bypass obstacles. Arkin states that what might appear to be a naive approach, the summing of individual vector outputs of the "schemas," works quite well, both in simulations and real world experiments.

Although most reactive systems are not concerned with the use of world knowledge (map), Arkin's autonomous robot architecture [4] includes *a priori* information about the environment. The Army-ant scenario is closer to Brooks' works, which avoid

\(^{10}\)Webster defines 'schema' as a mental codification of experience that includes a particular organized way of perceiving cognitively and responding to a complex situation or set of stimuli.
"world modeling" for individual insect-like robots (such as Squirt and Genghis, a six-legged robot).

### 2.3.2 Multirobot Systems

In this section we will dwell upon some previous work done on coordination and control of multirobot systems, excluding coordination of multiple manipulator systems — which is generally based on centralized control.

Systems of multiple mobile robots have gained interest in recent years when projects such as planetary surface mission and hazardous waste management, emerged. Large populations of mobile robots, as decentralized robotic systems (DRS), have many advantages over centralized systems, especially when high reliability is required, such as maintenance tasks in nuclear power plants.

The application of multiple mobile robots to planetary missions is outlined by Miller in [24]. This work states the fact that teams of small autonomous robots have advantages, such as lower cost, lower launch/landing mass and mission reliability, over larger robots. Behavior driven control methods described in the previous section are likely to be used in designing such small robots. The use of fixed radio beacons is also anticipated along with the necessity of leader selection for formation of a coordinated team. Leader selection can be achieved by assigning serial number to robots. Robots are assumed to be able to transmit/detect these numbers, and the one with higher serial number will be collectively elected as leader.

Miller also emphasize the fact that coded beacons and beacon readers on each robot, with other simple broadcast signals, could be sufficient to achieve a complex task with individual behavior, since navigation and homing techniques are well developed for autonomous mobile robots.

The term "distributed robotic system" (DRS) is sometimes used to describe a multirobot system based on instinctive responses and cooperation. Simulations of DRS designed for searching for pollutants are created by Genovese, et al. [16], based on biological systems and subsumption-like architectures. This design suggests a supervising user who can localize agents whenever necessary. In this context, the system is not a "swarm." Communication between agents is more complex than the one described by Miller and includes coded radio transmission on a single channel. Although this model
includes "acknowledge" messages, this type of communication proves to be advantageous, as we investigate in Chapter 4.

In [6], Beni and Wang claim that all agents involved in an operation must communicate to each other the intention to execute their part of operation. Stating that the "commitment protocols" are basic building blocks of distributed computing algorithms, such communication is defined as a required characteristic. On the contrary, biological systems described in Section 2.2 are able to operate as a self-organizing system without direct communication (of intentions). The key factor here is the large number of agents. But again, in our Army-ant scenario, there may be a situation where number of agents at each "goal" is not sufficient to start the next phase. A temporary "do-not-consider-this-goal" signal can be introduced by the leader to overcome this problem.

Previous experimental works on multiple mobile robots include ACTRESS (ACTor Based Robots and Equivalent Synthetic Systems) developed by Habib et al. [17], and Yamabica robots realized by Yuta and Premvuti [40]. ACTRESS, as an autonomous and decentralized robotic system, does not only have mobile robots, but also any kind of robotic system and/or computers. Mobile robots developed for ACTRESS have a portable computer instead of an onboard microprocessor and weigh 51 kg. They demonstrated intelligent navigation behavior.

Yamabica robots, more compact than above-mentioned robots, are used to determine a solution for a deadlock situation caused by multiple mobile robots with overlapping running courses in aisles. The method described in [40] provides a shunting process to solve the deadlock. However it requires constant broadcast of information (e.g., current position), a world knowledge, and is based on complex decision modules "managing" the information obtained from sensors. Both ACTRESS and Yamabico models differ from our approach to self-organization in many contexts.

Part of the work on spatial self-organization in this thesis (Chapter 3) is inspired by an interesting work by Sugihara and Suzuki [35]. They give a method for motion coordination of a group of mobile robots. Each robot plans its motion individually based upon a defined goal and detected position of other robots. This method is fully distributed in that sense and shows that intelligent behavior can emerge from simple individual behavior.

Sugihara and Suzuki were able to create different geometric shapes by defining simple algorithms to be executed by a large number of agents. Their simulation shows that
robots can form lines, circles, polygons and distributes themselves within a circle or convex polygon in the plane. Although the algorithms defined in [35] are shown to work quite well, most of these algorithms are based on the assumption that each robot can detect the distance from the farthest teammate (as well as the distance from the nearest teammate). In this research, we use the advantage of a goal beacon and eliminate the need for the distance from the farthest agent in computations.

2.4 Learning and Decision-Making

Neural networks, fuzzy systems and Kohonen networks, which are all examples of self-organization, are explained briefly in this section.

2.4.1 Neural Networks and Fuzzy Systems

Neural networks and fuzzy logic are definitive examples of self-organizing systems. Neural networks emerged from the idea of building an intelligent machine out of artificial neurons. The definition of an artificial "neuron" is given by McCulloch and Pitts in 1943, in order to create a network for computation of logic functions. Some early results on neural systems were achieved in 1950 and 1960's. During this period most of the articles are collected under self-organization titles. Research in neural networks came to virtual halt at the end of 60's, when networks under study were shown to be computationally weak. However, neural networks gained a world wide interest in 1980's, due to appearance of faster computers, and the discovery of new neural network architectures and learning algorithms.

Neural networks are characterized by having (i) a large number of neuronlike processing elements, (ii) a large number of weighted connection between these elements, (iii) highly parallel, distributed control and (iv) an emphasis on learning internal representations automatically [31]. Knowledge of the network is "encoded" in the weights between neurons.

Artificial neural networks are currently used in the fields of signal processing, speech recognition, visual perception, control and robotics. Neural networks help solve
problems encountered in these fields with natural mechanism of generalization, and have the most promise in real-world problems.

Based on a "prototype" model, NNs naturally develop a category structure, "classification by similarity", which enables the use of NNs in speech recognition, visual perception, etc. However, the same characteristic is a serious potential weakness. For example, it is highly difficult for a simple NN to detect the parity of bit vectors.

NNs deal with uncertainty, not by calculated design, but as a result of the parallel-distributed structure. It is equally possible to directly build insights about categorization into an artificial system. Fuzzy systems are the result of this approach. When such insights are properly quantified, fuzzy systems can be just as well defined and useful as the traditional formulations of probability and statistics. In 1965, Lotfi Zadeh formally developed multivalued set theory and introduced the term "fuzzy" into technical literature. Neural networks and fuzzy systems estimate input-output functions, and both are trainable dynamical systems. Sample data affects their time evolution. Unlike statistical estimators based on mathematical models, they are model-free estimators. They "learn from experience" [21].

Artificial neural networks can be programmed and trained to estimate sampled functions whose form is unknown. Many feedback neural networks can learn new patterns and recall old patterns simultaneously. Supervised neural networks can learn far more input-output pairs than the number of neurons and synapses in the network. Since there is no input-output model of the system, the same neural-network architecture and dynamics can be applied to different problems.

Fuzzy systems are based on storage of "common-sense" rules [21]. For example, a fuzzy Army-ant robot controller might have the fuzzy association "if load is heavy, then signal for help longer." Fuzzy phenomena admit degrees: some loads are heavier than others; some signal durations are longer than others. A single association (heavy,longer) encodes all combinations. Fuzzy systems are newer than neural systems. Yet there are successful applications of fuzzy systems in many areas, such as automation of subways, camera focusing, automobile cruise control, etc. Fuzzy systems "reason" with parallel associative inference. A fuzzy system reasons with multivalued sets, instead of true or false propositions, and it may adaptively modify its fuzzy associations from representative numerical samples.
2.4.2 Unsupervised Learning: Self-Organizing Maps

Self-organizing maps (Kohonen networks) are based on the idea that there is a meaningful order of processing units in the human brain, a property which is ignored in the "learning machines" [20]. The term "order" does not usually refer to spatial arrangement. The units being structurally identical, specialized roles are determined by parameters which can be changed in certain processes, producing meaningful organizations.

Kohonen introduced a self-organizing algorithm that produces a global ordering over the network. To every cell in the network, a reference vector is assigned and the same input vector is fed to all cells. The unit whose reference vector matches the input best, is chosen as "winner." To implement a process in which reference vectors are ordered spatially, it is necessary to update reference vectors in blocks, concentrated around the winners. Due to frequent overlap of such blocks in learning and similar correction imposed in each block, the values of reference vectors tend to be smoothed and to become ordered. Updating is always restricted to a topological neighborhood of cells. Size of the neighborhood decreases monotonically. At the end of the process the neighborhood for each cell contains the closest cells, not only the unit itself. If the neighborhood consists of only the unit itself, the map will degenerate into the zero-order topology model.

The number of training steps in the map method is approximately the same as in traditional neural networks. In [18], it is stated that statistical accuracy for phonemic recognition achieved with the map method and learning vector quantization algorithms exceeded the results provided by more conventional Neural Networks.
3. Spatial Self-Organization

First problem encountered in our Army-ant scenario is the formation of a team around a "goal" (marshaling). It is necessary that robots gather around a beacon (goal) to start the next phase (e.g., lifting). In this chapter, we attempt to define several algorithms for organization of mobile semi-intelligent agents into 2-D and 3-D arrangements. We also extend our algorithms to multigoal situations and try to find relatively better solutions by increasing the number of assumptions on the "intelligence" of agents. We will try to explain our approach while giving some examples of spatial self-organization.

We want the Army-ant "swarm" to be a self-organizing system because of the nonlinear characteristics such as the significant changes in the overall behavior of the system due to small changes in the individual behaviors. On the other hand, it is highly difficult (if not impossible) to obtain analytical results for a large-scale nonlinear system. Because of the analytical difficulties in large-scale nonlinear interactions, our work has an "algorithmic" characteristic.

3.1 Formation of a Circle Around a Goal

The approach taken in this thesis for formation of a circle around a goal by a group of robots is a distributed one. This method does not require a central controller. This is an important characteristic of a self-organizing system.

Our scenario envisages a beacon signaling to the robots the position of the pallet/goal. There are also several assumptions on the detection capabilities of the agents:

i) Each robot can detect signals from the others and the beacon, and can distinguish between beacon and agent signals.
ii) Each robot is able to compute the distance and direction of the signal source.
However, Army-ant robots have navigation characteristics different from other autonomous mobile robots discussed in Chapter 2; our scenario does not require the robots to build or possess maps of the environment. The robots do not need to know beacon and robot positions in absolute coordinates. The algorithm we used to form a circle is very simple: A robot approaches the beacon up to certain (predefined) neighborhood $d$, its velocity being a function of its distance from the beacon (Figure 3.1.1). For different regions, a different function is defined:

$$
\bar{v}_i = \begin{cases} 
    f_1(\tilde{d}_i) & \text{if } \|\tilde{d}_i\| > d + \varepsilon \\
    f_2(\tilde{d}_i) & \text{if } \|\tilde{d}_i\| - d \leq \varepsilon \\
    f_3(\tilde{d}_i) & \text{if } \|\tilde{d}_i\| < d - \varepsilon 
\end{cases}
$$

Note that both $v_i$ and $d_i$ are vectors. When a robot finds itself inside the circular region defined by $d$, it is "repelled" by the beacon. In the $\varepsilon$-region, each robot moves away from its closest mate. Our assumptions above guarantee the detection of the closest mate.

All agents knowing the value of $d$ and running the same algorithm, they form an almost perfect circle with radius $d$ around the beacon. This algorithm, with the advantage of using a beacon signal, does not require that the robots know the distance from the farthest teammate, as in [35]. The use of a beacon enables us to find a new method for 2-D arrangement that uses only localized information.

Simulations running this algorithm (Figure 3.1.3) gave very satisfactory results. Some of the results are shown in figures 3.1.4–3.1.8. As seen in figures 3.1.5, 3.1.7 and 3.1.8, all agents are in the defined $\varepsilon$-region and they are distributed on the circle uniformly. As long as the value of the displacement per time step (velocity/force function) is not much larger than the size of the $\varepsilon$-region, a satisfactory result is guaranteed.

None of the algorithms we define in this chapter take into consideration the problem of collision avoidance among robots. The performance of the algorithms defined here should remain basically the same if this problem is considered. We also cannot give a performance analysis in terms of the quality of the resulting arrangements. The quality of the approximation in the implementation will depend on the accuracy of sensors and response time of actuators of the agents. These questions are to be addressed when actual mobile robots are built and tested.
3.2 Alterations and interpretations of the algorithm

The algorithm defined in previous section can be altered to obtain an almost uniform distribution in a circle or can be directly applied to 3-D.

3.2.1 Uniform Distribution Inside a Circle

If the algorithm used for circle formation around a goal is changed as in Figure 3.2.1, agents approaching the beacon will fill the circular region defined by \( d \) almost uniformly. The algorithm is simpler than before. Each agent entering the circular region starts to move away from the closest mate. Assumptions for this algorithm are the same as before, and there is no need to use a variable (\( e \)) to define a desired region.

As sample simulation results show (Figures 3.2.2-3.2.5), the algorithm provides a uniform final arrangement for relatively small number of robots. As the number of robots increases, final distribution inside the circle may become less uniform depending on the number of robots and their order of entering the region. Robots inside the circular region simply move one unit farther from the closest mate at each time step. A more complex definition for the velocity vector, perhaps by using two closest teammates, can provide faster and more satisfactory results.

3.2.2 Extension to 3-D

Although our Army-ant scenario requires 2-D spatial arrangements, self-organizing populations of large number of robotics agents can be applied to space applications. The mobile agents could be small satellites, space or underwater vehicles instead of Army-ant robots.

The algorithm defined in section 3.1 can be directly applied to 3-D without any alterations. In 3-D, the same algorithm will result formation of a sphere with the beacon in the center. As shown in figures 3.2.6-3.2.7, final positions resulting from randomly distributed initial conditions are again satisfactory. Although it may be hard to distinguish the spherical shape, views from different angles reveal that the mesh is quite exact. Simulations on Matlab display the agents that are not in the desired region by different colors. Also, it is possible to check the uniformity of the resulting mesh by simply
comparing the average value of the distances between a robot and its teammates. After several runs, we found that it is possible to form a sphere with the defined approximations, by suitably choosing parameters, such as velocity functions for different regions. For example, these functions must be defined so that the step size of velocity vector around \( \varepsilon \)-region is relatively smaller than the value \( 2\varepsilon \), in order to form an almost perfect circle. With the same parameters as in 2-D, the algorithm works slower in 3-D because of the introduction of one additional dimension.

Similarly, the algorithm used to distribute the agents uniformly inside a circular region can be applied to 3-D. However, we do not give such examples since it is visually and algebraically difficult to obtain a performance criterion for the final positions of the agents inside the sphere.

### 3.3 Formation of a Paraboloid

As we demonstrated in the previous section, it is possible to generalize our assumptions and algorithms to 3-D. Here we will demonstrate the formation of a paraboloid by robotic agents based on the same assumption as before. Namely, (i) each agent can detect signals from beacons and/or agents and (ii) each agent is able to determine the direction and distance of these signals' sources. The same assumptions and algorithms used here can also be applied to 2-D. However, we prefer to simulate the case in 3-D, which is slightly more complicated and has possible applications to space. A team of robotic agents in orbit may cooperatively act as a large antenna that can be oriented as desired.

To form a paraboloid, one must define its focal and vertex points. The paraboloid (parabola) is a surface (plane curve) generated by a point moving so that its distance \( a \) from a fixed point, namely focus, is equal to its distance from a fixed plane (line). The orthogonal distance between the fixed plane and the vertex is obviously equal the distance between the focal point and the vertex.

The variable \( a \) clearly defines the shape of the paraboloid and has to be fed to the agents forming it. Since our aim here is to create a parabolic antenna, not the whole paraboloid, another variable, say \( b \), is needed in order to define the radius of the antenna. \( b \) can be the direct distance from vertex to the perimeter of the antenna (See Figure 3.3.1).
Having variables $a$ and $b$ fed to the agents, we have to have two beacons at the focus and vertex of the paraboloid, or we have to direct to agents to these points. Here we use two agents, chosen based on their distance to those target points, as beacons and direct them accordingly. Agents have to be able to distinguish between signals emitted by beacons or by the chosen agents.

Defining the agent (beacon) at the focus as $F$, the distance from it as $d_f$ and the agent (beacon) at the vertex as $V$, distance as $d_v$, and considering the regions shown in Figure 3.3.2, our algorithm for formation of paraboloid is defined in Figure 3.3.3. The resulting velocity vector field is shown in Figure 3.3.4.

As shown in the flowchart, this algorithm, based on the assumptions given above, simply compares functions of "sensed" distances to some predefined variables. The decision for regions A and B are straightforward: comparison between pairs $(d_v, b)$ and $(d_v, d_f)$. The decision for $E$, $E^+$ and $E^-$ regions are slightly more complicated. Having values $d_f$, $d_v$, and $a$, we define a coordinate system such that the origin is at $V$ and $F$ is on the $z$-axis (See Figure 3.3.1).

Thus, the distance $d_v$ can be written as:

$$d_v^2 = x^2 + y^2 + z^2$$  \hspace{1cm} (3.3.1)

$x$, $y$, $z$ being the coordinates of an agent in the defined (imaginary) coordinate system.

On the other hand, the distance from the focal point is:

$$d_f^2 = x^2 + y^2 + (z - a)^2$$  \hspace{1cm} (3.3.2)

From (3.3.1) and (3.3.2) we get:

$$x^2 + y^2 + z^2 - 2az + a^2 = d_f^2 \Rightarrow d_v^2 - 2az + a^2 = d_f^2$$

$$\Rightarrow z = \frac{d_v^2 - d_f^2 + a^2}{2a}$$

and:

$$x^2 + y^2 = d_f^2 + z^2$$

For any point on the paraboloid, the values $z$ and $x^2 + y^2$ must satisfy the equation:

$$z = \frac{x^2 + y^2}{4a}$$

If $z > (x^2 + y^2)/4a$, then the point is in the half-space including the positive $z$-axis; otherwise it is in the half-space including the negative $z$-axis. Again an $E$-region, shown in Figure 3.3.2, is defined.
Therefore each agent, having "sensed" $d_f$ and $d_v$, can decide which direction to go using this abstraction. "Attraction" and "repulsion" vectors are defined as functions of the $d_v$ and/or $d_f$. This algorithm ensures the formation of an almost uniform paraboloid mesh. Matlab simulations with random initial conditions gave satisfactory final positions as seen in figures 3.3.5 and 3.3.6, even though the rules defined above do not positively contribute to the formation during the first steps of the simulation (since it takes time for the "beacon" agents to attain their desired positions). Also the velocity function inside the $\epsilon$-region is quite simple: Each robot moves away from closest teammate by a one unit at each time step. Again, choice of a more complex algorithm may provide a more uniform result in shorter time intervals. After (and also during) the formation, the shape and orientation of the parabolic mesh can be controlled by simply changing the position of the agent/beacon at the focus. The rest of the team will adjust according to the algorithms defined.

### 3.4 Multigoal situations

Examples given in the previous sections included a single beacon/goal. However, simultaneous handling of multiple pallets is featured in Army-ant scenario. Therefore, it is necessary that teams of agents group around several goals simultaneously. The methods described here will be explained in order of increasing assumptions (and, therefore, communications).

Assume that $m$ goals and $n$ agents are randomly distributed in an obstacle-free area. In the simplest case, agents can simply move toward the closest goal, assuming that they can distinguish between beacon signals. It is obvious that this straight-forward method may result in quite uneven distributions, as seen in Figure 3.4.1, where each one of the 40 robots moves toward the closest of four goals. (In the simulations defined hereafter, we will refer to this goal distribution as the "initial distribution," in order to have a performance criterion.) To obtain a more uniform distribution among goals/beacons, we introduce an additional variable $d_s$ that is assumed to be known by all agents.
3.4.1 Introduction of "switching distance" $d_s$

Switching distance $d_s$ is a variable that changes with time. It represents the radius of the forbidden circular region around each goal. This value or the variables defining $d_s$ as a function of time, are known by each agent moving toward its own designated goal. When an agent finds itself in the forbidden region (i.e., if its distance to the goal is smaller than $d_s$ at any given time $t$), it randomly changes its goal. Again we are assuming that all the beacons are in the detection region of all agents. When an agent changes its destination, the forbidden region also "moves" to the new destination. It is now defined around the new goal.

For example, in the situation shown in Figure 3.4.2, an agent approaching beacon B finds itself inside the forbidden region at time $t$ (distance to goal $d_i(t)$ is less than switching distance $d_{s}(t)$), and therefore, changes its destination randomly. Its goal at time $t+1$ becomes beacon A. Since the distance to the beacon A is greater than the distance to beacon B, the robot is now outside the forbidden region defined by $d_s(t)$, and continues to approach beacon A on the next step $t+2$. Although interpretation of the distance $d_s$ changes, the function (variables) defining $d_s$ stays the same.

The most logical choice for the function $d_s = f_{d_s}(t)$ seems to be a decreasing function of time. Advantages of the introduction of such a function are shown with simulations. The main simulation algorithm is given in figure 3.4.4.

As flowchart in Figure 3.4.4 shows, switching distance $d_s$ is effective if it is greater than $d_n$ at a given time. $d_n$ is again a predefined constant to define the neighborhood and to stop the switching algorithm. The algorithm is devised in such a way that the agents will be distributed uniformly in the circle defined by $d_n$ around each goal.

Simulation performed in Matlab enables us to define different time functions for $d_s$. We can also define parameters other than $d_s$, such as velocity as a function of distance, neighborhood $d_n$ around each goal, simulation time, as well as the number of agents and beacons.

A particular choice of function for $d_s$ does not always give the best results for every initial condition (position of robots and goals). However, improvement in the final goal allocations is distinctive. Effects of the parameters on the final allocation can be determined by experiments, but only for specific initial positions.

The result given in figure 3.4.5 shows that goal allocation of robots can be changed from highly uneven ([6 6 18] for 30 robots and 3 beacons) to an almost even
distribution ([1 1 19 10]) by defining $d_s$ as shown. A very similar definition for switching distance (Figure 3.4.6) again gives a satisfactory result (from [0 1 11 29] to [13 1 11 16]), close to best distribution ([13 1 14]) for different initial positions. A third example (Figure 3.4.7) shows quite different choice of function. Again, this constant $d_s$ provides distinctive improvement in the goal allocation. In simulation examples given in figures 3.4.5-3.4.7, switching distance $d_s$ was chosen so that we are able to show the final goal allocations, and therefore agents did not have sufficient time to fill uniformly the circular regions defined by $d_n$. However, we know that the spatial distribution of agents inside the neighborhood would be uniform or almost uniform if we have chosen the final time to be larger than 300.

### 3.4.2 More intelligent agents: Increasing communications

The algorithm using the notion of switching distance is based on relatively simple assumptions on the communication capabilities of the robots. We may increase the number of essential assumptions here (and therefore communication capabilities of the agents) to devise a new algorithm.

Assume that, in addition to assumptions we made in section 3.1, all robots are able to broadcast two values (namely, "ID number" and "family value"), and detect values signaled by others.

We use these two values to create teams of robots associated with each goal. Each robot, again, initially moves toward the closest detected beacon. When a robot finds itself inside the neighborhood (region defined by $d_n$), it checks if it is the first robot arriving. If it is the first one arriving into the neighborhood, the robot becomes a *first*, and starts running a different algorithm while trying to move away from the closest teammate (as they arrive). If there is already a *first* robot in the neighborhood, all other arriving robots compare their ID numbers with the ID number of the *first* agent. If the difference between two ID numbers is larger than family value $f_v$, then the robot randomly switches goal. Otherwise, it stays in the neighborhood and tries to move away from the closest mate, including the *first* (See flowchart in figure 3.4.8). Family value $f_v$ is the maximum possible difference between ID numbers of a team (family). Calculations are done in modulo $N$, where $N$ is the total number of robots.

For example, if robot number 45 in a population of 50 robots enters a neighborhood with a *first* robot with current ID number of 2 and family value of 10, it
stays in the neighborhood since \( |45-2|_{\text{mod}(50)} = 7 \) is less than 10. \( f_v \) is a predefined value computed and transmitted by \textit{firsts}. For our simulations, we choose the number to be:

\[
f_v = \frac{\text{Total # of robots}}{2(\# \text{ of firsts detected} + 1)}
\]

Therefore, receiving this value from the \textit{first}, arriving robots can "compute" whether they are allowed in this particular neighborhood or not. This definition requires another assumption. Each agent (not each \textit{first}, since any agent may become a first) must know the total number of robots\(^1\).

This definition of \( f_v \) does not guarantee an even distribution by itself. The \textit{first} robots run another algorithm to improve goal allocation. Each \textit{first} agent changes its ID number according to ID numbers of other \textit{firsts}\(^2\). A \textit{first} shifts its ID number so that the (modular) difference between its ID number and closest first ID number increases (See Figure 3.4.9 for an example).

Sample simulation results are given in the following pages. Results shown in figures 3.4.10 and 3.4.11 have the same initial conditions and parameters (e.g., velocity/force functions for different regions, neighborhood size, infinite detection region) as the simulation examples given in figures 3.4.6 and 3.4.7, respectively. Comparison of the results shows a better final distribution in both cases. When even allocation of robots is important, this method may be used — although the number of assumptions on communication abilities of the agents is larger. In both examples, one robot continuously switches its goal. This is due to behavior of the \textit{first} agents. Since we do not have any control over the ID number of the robots that become \textit{first}, the number of robots allowed in each family/team may not be the same (even if all the \textit{first} agents can detect each other) due to the ID number shifting process. However, the probability that one or two robots are not allowed in any team is relatively small, and in a large population of robots it is insignificant.

Figures 3.4.12 and 3.4.13 show two simulation results having the same initial conditions and parameter values. Only ID numbers of the robots are changed so that the ID numbers of the \textit{first} agents would be different in each simulation run. Since these numbers are different, goal allocation versus time plot shows different characteristics (it

\(^1\)Note that this requirement is a result of our definition of \( f_v \). As an alternative method, beacons can broadcast the desired number of robot for a particular pallet.

\(^2\)For the time being we assume that all agents can detect all beacons, and also all firsts can "see" other firsts.
takes more time to reach steady-state, and goal allocations change drastically in the first example), and in the second example one robot is again not allowed to any of the teams. However, the steady-state distributions are quite similar and almost even.

It is also possible to see the change in the ID numbers of the firsts and family sizes during the simulation. Simulations created in the Matlab 4.0 environment for this algorithm also show the detection map\(^3\) of the firsts.

As seen in the flowchart, we also assume that the total number of robots is known by all agents. This assumption decreases the versatility of the swarm. Instead of defining \(f\) as a function of total number of agents, we can simply choose it as a constant value fed to all agents. Another approach is to design beacons capable of signaling required family size to agents. This method also enables us to form teams with different sizes if the size and/or weight differ from payload to payload.

It is also possible that the distances between the neighborhoods of beacons prevent the firsts from detecting each other. Since \(f\) is defined as a function of the number of detected firsts, this may create a problem. However, if firsts robots cannot detect each other, that means not every robot can detect all beacons (assuming beacons have the same or lesser communication capabilities as robots). Again, the steady-state distribution is much better than initial distribution (See figure 3.4.14).

### 3.4.3 Ideal case

The methods explained in this chapter are listed in increasing complexity in communication and assumptions. One can observe that we are slowly crossing the line to behavioral self-organization. In this section, we will outline a method that we call the "ideal case," without simulations. This method requires very explicit communications.

We can design robot receivers in a way that they would be able to change their reception bandwidth, i.e. detection-band limits. They must also be able to send signals at the same frequency as the beacon they are moving toward. Therefore, the robots could be programmed, for example, to double their detection limit and signal their arrival if they arrive into the defined neighborhood\(^4\).

---

\(^3\)Detection map' shows whether a first agent can detect other firsts or not. See screen snapshots given in Figure B.6 of the Appendix.

\(^4\)We are still assuming that robots can detect the distance to signal sources.
Let us assume that the beacons defining the position of the pallets can also receive signals from agents in the neighborhood, can change their signal frequency, and are able to run simple algorithms such as counting. If agents entering the region can signal\(^5\) that they are now in the neighborhood, the beacon will be able to count the number of agents assigned to the pallet at a given time.

When the count reaches the predefined number of robots\(^6\), the beacon becomes invisible to robots outside by shifting its broadcasting frequency beyond the detection limits of agents which are not in the neighborhood. Since robots in the neighborhood have their detection band doubled, this change in the frequency can be used to start another phase.

This scenario brings another problem besides increased complexity in communications and beacon characteristics: Two or more robots arriving into the neighborhood at the same or in a very short interval, may create miscount. Although the solution to this problem lies in the concept of semaphores\(^7\), realizing a system which eliminates the problem, as in on-line reservation systems, means more complexity in the design of robots and beacons.

### 3.5 Other issues

#### 3.5.1 Many-body physics

Most of the algorithms defined in this section are based on the assumptions that agents can compute directions and distances to beacons and other agents. The displacements at each time step (i.e., velocities) of the agents are functions of these distance vectors. On the other hand, the last algorithm, which requires more explicit communications, uses ID numbers assigned to agents (along with the distance vectors) to compute the velocity vectors.

Since the velocity vectors computed at each time step are functions of distance vectors and/or ID numbers associated with agents, we can think of this model as a many-

\(^5\)Using the same frequency so that no other beacon detect the signal.

\(^6\)Programmed into the beacon.

\(^7\)Here is a definition of semaphores in operating systems theory: A semaphore, first defined by Dijkstra in 1965, is a protected variable whose value can be accessed and altered only by certain operations.
body physics model. Our many-body model in Army-ant scenario has an advantage: we can define our laws of gravitation as we want.

3.5.2 Explicit Communications

As simulations show, an increase in the amount and type of transferred data between agents leads to an increase in the "intelligence" of the swarm. On the other hand, each new assumption on the communication abilities of the agents (and beacons) will probably increase the cost and complexity of the "system." For example, the scenario defined as "ideal case" in section 3.4.4, based on more assumptions, requires very explicit communication capabilities. Problems such as two or more robots arriving into the neighborhood at the same time, can be solved given that each robot is equipped with devices able to solve the "source sharing problem."  

Also, there may be cases that beacons are situated outside the detection region of the agents. Possible solutions of such a problem again require that robot detect not only the distance and direction, but also the "status" of other agents. Status can be another broadcast signal telling others that a particular robot is going toward a beacon. Using this signal along with detected distance value, robots can "school" together to find beacons, as do predator fishes.

The simulations in this chapter did not include any obstacles. However, Army-ant robots working in a warehouse encounter many problems that we did not consider here. As we explain in the next chapter, a status signal can be very useful when there are many obstacles creating "alleys."

---

8 An agent, before signaling its arrival to the beacon must assure that the beacon is not busy receiving another arrival signal.
Figure 3.1.1 Plot of the speed toward the goal as a function of distance to the goal

Figure 3.1.2 Plot of the regions defined for velocity/force vector functions
Figure 3.1.3 Algorithm for formation of a circle around a goal
Figure 3.1.4 Initial (x) and final positions of the agents for t_f=50 with d=80.
Figure 3.1.5 Positions after 100 more time steps with same parameter values.
Figure 3.16 Initial (x) and final positions of the agents for t=50 with d=80.
Figure 3.1.7 Positions after 100 more time steps with same parameter values.
Figure 3.148 Initial (x) and final positions for $t=200$ with $d=250$. 
for $i = 1$ to $n_{\text{robot}}$

- $d_i > d$?
  - Y: move twd. goal
  - N: move away from closest teammate

**Figure 3.2.1** Algorithm for distributing agent uniformly inside a circle
Figure 3.2.2 Initial (x) and final positions for t=50 with d=100.
Figure 3.2.3 Positions after 50 more time steps with same parameter values.
Figure 3.2.4 Initial (x) and final positions for t=200 with d=100.
Figure 3.2.5 Initial (x) and final positions for $t_f=200$ with $d=100$. 
Figure 3.2.6 Final position of 30 agents around the beacon with $t=300$ and $d=150$. 
Figure 3.2.7 Final position of 30 agents around the beacon with $t_f=150$ and $d=100$. 
Figure 3.3.1 Definition of the imaginary coordinate axis

Figure 3.3.2 2-D representation of the regions defined in the algorithm
for $i = 1$ to nrobot

- **Y:** in region B?
  - Y: move twd. V
  - N: in region A?
    - Y: move twd. V
    - N: outside $\xi$?
      - Y: move away from closest teammate
      - N: in region $\xi^+$?
        - Y: move twd. V and away from F
        - N: move twd. F and away from V

- **N:**

Figure 3.3.3  Flowchart of the algorithm for formation of a paraboloid
Figure 3.3.4 Force fields defined in the algorithm (Velocity functions are constant in all regions).
Figure 3.3.5 Initial (top row) and final positions of 30 agents forming a paraboloid with $t_f=600$. 
Figure 3.3.6 Initial (top row) and final positions of 20 agents forming a paraboloid with $t_f=600$. 
Figure 3.4.1 Initial (x) and final positions for $t=150$: All agents have moved toward closest beacon.
Figure 3.4.2 Interpretation of forbidden region defined by $d_s$

Figure 3.4.3 Switching distance $d_s$ is decreasing with time
($d_n$: radius of the predefined neighborhood)
for $i = 1$ to $n_{\text{robot}}$

$\begin{align*}
\text{choose a random goal} & \quad \text{if } d_i(t) < d_g(t) \& d_g(t) > d_n \text{?} \\
\text{move away from closest teammate} & \quad \text{if } d_i(t) < d_n \text{?} \\
\text{move twd. goal} & \quad \text{else}
\end{align*}$

$d_i$ : distance between $i^{th}$ agent and its goal  
$d_g$ : switching distance  
$d_n$ : radius of the predefined neighborhood around goals

**Figure 3.4.4** Flowchart for use of switching distance $d_s$
Figure 3.4.5 Definition of switching distance and resulting final positions at $t_f=300$ (Initial positions defined by x’s).
Figure 3.4.6  Definition of switching distance and resulting final positions at \( t_f = 300 \).
(Similar function for \( d_S \), different initial positions)
Figure 3.4.7 Definition of piecewise constant switching distance and resulting final positions at $t_f=300$. 
for $i = 1$ to nrobot

$\begin{align*}
\text{if } d_i(t) > d_n^* \text{ then } & \text{ move twd. goal} \\
& \text{become a first} \\
\text{else} & \text{ \exists a first in the nei. d?} \\
& \begin{cases} 
\text{Y} & |n_i - n_1| < f_v \text{ then } \text{move away from closest teammate} \\
\text{N} & \text{choose a random goal}
\end{cases}
\end{align*}$

$d_i$: distance between $i^{th}$ agent and its goal
$d_n$: switching distance
$d_n^*$: radius of the predefined neighborhood

$N$: number of robots
$n_i$: ID # of $i^{th}$ agent
$n_1$: ID # if the first

**Figure 3.4.8** Flowchart of the $f_v$ algorithm

**Figure 3.4.9** Three $first$ agents shifting their ID numbers
Figure 3.4.10 Initial (x) and final positions for \( t_f = 200 \) and goal allocation of robots. (Initial positions and other parameters are the same as the example given in figure 3.4.6.)
Figure 3.4.11 Initial (x) and final positions for $t_f=100$ and goal allocation of robots. (Initial positions and other parameters are the same as the example given in figure 3.4.7.)
Figure 3.4.12 Initial (x) and final positions for $t_f=100$ and goal allocation of robots.
Figure 3.4.13 Initial (x) and final positions for $t_f=100$ and goal allocation of robots. (Initial positions and other parameters are the same as in Figure 3.4.12, only ID numbers of the robots are changed.)
Figure 3.4.14 Initial and final position of the robots for $t_f=150$ and goal allocation versus time. (Detection region of the agents is 700 units while the beacons and robots are distributed on a 1000x1000 space)
4. Behavioral Self-Organization

In this chapter, we will describe several problems which may be encountered in Army-ant scenario, and propose solutions to some of these problems. Other than spatial self-organization, Army-ant robots have to act cooperatively to find and transport the pallets. In order to accomplish such tasks, they must form a team which can operate as a single entity, i.e. they have to achieve a behavioral self-organization. A robot will have several behavioral modules\(^1\) which enable it to carry some functions. The data a robot receives from its teammates, directly affects these modules, and they affect the teammates' behaviors reciprocally.

4.1 Behavioral Model of Army-ant Robots

Similar to the gravitational forces in spatial self-organization, each robot's position in the behavioral space is affected by activation or inhibition forces coming from teammates (and obstacles). Figure 4.1.1 shows the behavioral modules of an Army-ant robot as discussed in this chapter. As seen in the figure, robot behaviors are affected by other agents' status/behaviors as well as their own sensory data. The lines define the activation caused by a specific behavior module or sensory data. For example, avoid-obstacles behavior is induced by search and move-to-goal behaviors, and is also affected by sensor data signaling the presence of other agents and/or obstacles. On the other hand, the overall behavior of the team being affected by each individual behavior to some degree, the size of the team and the environment changes as the robots operate. Therefore, Army-ant robot behaviors have to be modeled (if possible) as a complex dynamic system with continuously changing parameters. Although there are artificial intelligence methods

\(^1\)A behavioral module is actually a subroutine that monitors specific inputs/variables, and initiates some other subroutine (or behavioral module) when necessary.
to define decision systems by using behavioral models, the behavioral model of an Army-ant team will be much more complicated than a single agent case [26].

4.2 System-level analysis of Army-ant problem

4.2.1 Finding pallets in the warehouse

Although positions of each pallet to be transported are signaled by beacons, it may still not be easy for army-ant robots to locate the pallets in a warehouse because of obstacles. We will explain this problem by using the sample warehouse plans, starting with the most simple case (Figure 4.2.1). If there were no obstacles in the warehouse (Figure 4.2.1a), finding the pallets would not be a problem for robots. If the size of the warehouse is relatively larger than the detection region of the robots, robots can still find the pallets by following teammates which are able to detect a goal. If the robots are able to signal their status (i.e., whether they are detecting a beacon or not), agents that cannot see any beacon, can follow the agents that detect a beacon. The decision tree for such a scenario will be:

```plaintext
if a beacon is detected
  move toward beacon
else if one or more robots that move toward a beacon are detected
  move toward the closest robot
else if
  search for pallet
end
```

This simple algorithm alone can increase the detection region of the "swarm" significantly. In the example shown in Figure 4.2.2, robots spread in a 1000x1000 unit space have a detection region defined by a radius of 250 units, and agents that cannot detect the goal initially (marked by +) can find the goal by following the agents which are initially moving toward the goal (marked by *).

However, following robots and wandering may not always lead to a satisfactory result if the warehouse has a more complex floor plan as in figures 4.2.1b and 4.2.1c.
Fortunately, robots can find the beacons by following the walls for these types of floor plans. Yet again, following signaling teammates and walls is not enough to find the beacons if the floor plan of the warehouse is closer to a "maze" (Figure 4.2.1d). As shown in the figure, robot A which follows the outer wall can reach the beacon, while robot B is circling the lower left block without seeing the beacon. If robot B keeps following the same wall, its only chance is to detect robot A as it moves toward the beacon. It is obvious that, for a more realistic case (Figure 4.2.1e), finding the pallets by using simple navigational rules (such as following other teammates and walls) will be highly difficult for Army-ant robots.

One method which may improve the result is letting the robots that are following their teammates to signal. Signals from the agents moving toward the beacon will have higher priority than the signals coming from the agents that are following their teammates. Using moving-toward-goal and moving-toward-robot signals, it is possible to create a network which we hope will include all agents. Also the moving-toward-robot signals will have different priorities depending on the signal coming from the robot followed. In a way, signals will indicate the number of nodes between a robot and the goal? As seen in figure 4.2.3, only one robot can detect the beacon initially. From the rest, one is following that robot, another is tracking this second robot, etc. Robots are following their teammate at different levels, and are able to find the beacon. When a robot following a teammate detects another agent with higher level signal, it automatically starts following this higher level signal (Figure 4.2.3). This method also proves to be highly rewarding in more complex cases given in figures 4.2.1d and 4.2.1e.

However, again, enabling robots to signal their status cannot completely solve the problem of finding the pallets in warehouse. To increase the chances of finding a pallet or a signaling robot, we can define more elaborate navigation algorithms or increase the communication abilities of the agents.

In order to explain what we mean by more elaborate navigation models, we will give some examples. In the warehouse plans shown in figures 4.2.4-4.2.8, initial positions of agents are marked by *'s. Defining the algorithm, we assume that robots can signal their status, and that they are able to follow signaling teammates. Agents that can not detect any signal, simply follow the walls (left or right, defined randomly or by given ID?

---

i.e., number of agents linking the robot and the beacon. The moving-toward-goal signal is actually the same as the moving-toward-robot; the only difference is the data-number of nodes- embedded in the signal.
numbers). However, if a robot following the walls cannot still detect any signal after a number of turns with an angle greater than a predefined value, it enters the first alley on the right or on the left, to stop a possible vicious circle. In Matlab simulations given here, this behavior becomes active after five 60-degree turns.

Figures 4.2.4 and 4.2.5 show one agent searching for a beacon in a warehouse. As seen from the traces, robots were able to find the beacon (pallet) using the described behavior instead of continuously circling the warehouse.

When the number of agents increases, the (average) time to find the beacon decreases. Figure 4.2.6 shows eight agents searching for a beacon. After a short time, seven of those agents have found the pallet; some by themselves, some by following other robots. Figures 4.2.7 and 4.2.8 show the first five steps, final positions, and four typical paths followed by agents. Eight out of ten agents have found the pallet after 300 steps. Some of the robots, as seen in figure 4.2.8, were able to find the pallet because of the search behavior we defined. These paths are obviously not the closest routes to the beacon; but considering the lack of map knowledge, the result is quite satisfactory.

Although the behavior we defined for these simulations is quite simple, floor area scanned by robots increased significantly in contrast with simple wall-following algorithms. By defining more complex algorithms, the search method can be improved. It is also possible to create algorithms with self-adjusting parameters, in order to have a swarm which can adapt to its environment for faster search phases.

Furthermore, with a more complex communication structure, agents can broadcast messages, and can affect the behavior of other agents. In the examples given in figures 4.2.6-4.2.8, each agent can follow each other; but there is always the possibility of losing the track of a teammate for a robot, when the former disappears behind an obstacle. Although we force the robots that lose the signal coming from another agent to continue in the same direction—hoping that they will redetect that agent—, this may not be enough. However, broadcasting specific signals enables robots to alert their teammates that they have been followed. A robot following a teammate which moves toward another beacon (goal or robot) may send signals to stop its teammate until the former moves and detects the higher level signal seen by its teammate\(^3\). Upon detection of the higher level signal, the teammate can be "released" by another coded message (See Figure 4.2.9). Content of such a coded signal is explained in Chapter 5.

\(^3\) This behavior is the same as the human behavior of shouting to someone while following him/her.
Another problem we did not consider here is the structure of the obstacles. They may physically stop the agents, but pass beacon signals. In such a case, the agents must be able to find their way to the signal source around the obstacle. However, most of the problems we listed here may not be highly problematic if the number of robots is large, which will be the case in the Army-ant scenario.

4.2.2 Navigating Aisles and Collision Avoidance

Since the navigation of Army-ant robots is not based on a map, there is no need to keep track of the distance traveled. Lack of a map simplifies navigation procedures while creating a problem in finding destination points — such as pallets, set-down points — in the warehouse. Army-ant robots, equipped with two range finders in front (one on the left, the other on the right), will be able to change their direction by simply comparing the output of the two range detectors. In addition to two detectors in front, it is possible to design the robots with additional range detectors on both sides. Such detectors can be used to find alleys on the sides, as is necessary for the algorithm described in the previous section (Figure 4.2.10).

Of course, a special decision routine has to be added to enable robots to change their direction in case of head-on encounter. Robots are able to maneuver in small spaces since the navigation mechanism is based on tracks.

Other than walls and pallets, robots also create (moving) obstacles for other teammates. There are several methods for coordination of multiple mobile robots in surrounding with moving obstacles. However, all these methods are based on central/hierarchical controller and/or map knowledge. Our scenario having a decentralized control approach, we can only use range sensors, and the robots' ability to sense teammates for obstacle avoidance and to solve deadlocks in robot-to-robot confrontations. Assigned ID numbers can be very useful to solve the "right-of-way" problem.
4.3 Team Coordination

4.3.1 Leader Characteristics

Although the Army-ant swarm is a homogeneous population, it is sometimes necessary to define a "leader". Army-ant robots carrying a pallet toward a destination point need to elect a leader which is responsible for beacon detection. Even though all agents are able to detect the beacon, election of a leader is necessary due to the physical limitation of sensors which complicates the determination of the direction of travel [34]. Furthermore, the last algorithm defined in Chapter 3 requires a leader (a *first*) for formation of teams around beacons.

Leader election can be based on different criteria. It may be related to assigned ID numbers, or agents accomplishing a task first can be awarded as leader. Any robot may become a leader, and leadership of a robot is temporary; it ends after set-down of the pallet, if not ended earlier by a malfunction.

Besides the role assigned in Chapter 3, a leader will have other functions. Suppose that a team of robots could crawl under a pallet, but they are incapable of lifting and/or carrying the load. The robot chosen as a leader can broadcast a high priority signal, which force other robots moving toward other beacons to change their destination and join the team. A leader may also *free* robots that are committed to a particular pallet. This can be done by sending a *do-not-consider-this-beacon* signal to other robots or by silencing the beacon (depending on the team formation procedures and communication abilities).

4.3.2 A Collective Behavior Example: Distributed Load Bearing

The next phase after formation of a team around a beacon/pallet is lifting and carrying it. Robots need to act together to establish a stable transportation mechanism. A method for controlling the collective behavior of Army-ant robots for load transportation is described in [34]. It is achieved by selecting a leader, without direct communications, using only force sensors mounted at the point of contact with the pallet. The method described in [34] results in stable behavior in the horizontal movement of agents under the pallet.
On the other hand, during lifting and transportation of pallets it may be important that the pallet be supported equally (to some degree) by all robots. It is possible to detect this condition by comparing the sensed vertical force on agents\(^4\). However, since there is no central controller, this has to be done collectively.

Assume that each robot under the pallet can receive signals from a particular teammate, and robots can form a ring network\(^5\) where the signals correspond to vertical force sensors of each robot (Figure 4.3.1). Using the incoming signal and output of the vertical sensor signal, robots can determine the difference between their own sensor and the sensor of the "previous" robot. Robots that detect a difference beyond the predefined parameters can broadcast a warning signal. Transportation starts and continues as long as there is no robots broadcast a warning signal caused by this or any other problem. However, this simple approach is far from guaranteeing the detection of an undesirable configuration. For a given error range, it is possible that the forces sensed by team members differ greatly while no robot is detecting violation (Figure 4.3.2). Therefore robots must somehow be able to detect the range violations no matter the relative positions of the two robots on the communication ring.

To solve this problem, we will use the idea of coupled van der Pol oscillators. VDP oscillators have been used to model both electrical and biological systems (such as vacuum-tube circuits and locomotion generation in fishes). Another aspect of VDP oscillator networks is also useful in Army-ant scenario: It may be important to synchronize the robots for a particular task. Bay [5] developed the idea of "heartbeat" for Army-ant robots, where each robot initially oscillates at a different frequency —related to its pre-

\(^4\) Achieving a situation where all vertical forces are the same does not necessarily mean that the most stable configuration is achieved. In both situations shown below, the vertical forces sensed by robots are the same, but the configuration in (a) is more stable.

![Diagram](image)

One straightforward solution to obtain a more stable situation is to force robots to move away from the closest teammate (with the use of proximity sensors) while staying under the pallet. If there is a need for a decision method to obtain the most stable configuration, devising an algorithm for this purpose will be extremely difficult. Besides the simulation difficulties (because of the non-invertibility of the moment matrix for cases with more than three robots), the algorithm must have adaptive properties due to the "fuzzy" character of the problem.

\(^5\)Probably by using assigned ID numbers.
assigned ID number. Then a team of robots can reach a common oscillator frequency (entrainment) when coupled together. Several coupling methods and results are given in [5].

For our purpose, we will assume the coupling is done in a ring configuration, i.e., $\lambda_{ij} \neq 0$ if $i=j+1 \mod N$ (Figure 4.3.1). We are forced to use a ring configuration, because full coupling inhibits agents to detect the variations accurately. VDP oscillators used here are slightly different from a general VDP oscillator. We added a feedback gain factor ($\lambda_{ij}$) which can be changed over time as a function of the output of the force sensor. Therefore, feedback gain of the each oscillator will change according to the vertical force component sensed by the robot. Oscillator outputs $x_i$ will be broadcast signals "heard" by agents with non-zero coupling coefficients ($\lambda_{ij} \neq 0$).

We define a mapping from sensed force to feedback gain $\lambda_{ii}$ as shown in Figure 4.3.4. In a ring formation with $\lambda_{ij}$'s are chosen between 0 and 0.3, the entrainment is reached after a time with frequencies in the range 0.8-1.2 [5]. For our purpose, we set all oscillator frequencies $\omega^2$ to 1. However, the amplitude of the oscillations will differ from agent to agent if the feedback gains are not the same. Each robot is again able to compare the output of its force sensor with the output of the coupled agent. In addition, the ring configuration enables each robot to detect the sensor output variations occurring in any other teammate by comparing its sensor output only to the previous robot in the configuration.

We will try to demonstrate the performance of this method by some examples. $\lambda_{ij}$'s are taken as 0.3 in all simulations given here. Output and error signals (i.e., $[x_i - x_{i-1}]_{\mod N}$) of the eight oscillators in ring formation are shown in figure 4.3.5. Starting with different initial conditions, the oscillator outputs match one another and error signals approach zero in time with all feedback gains $\lambda_{ii}$ are set to 1.

Figures 4.3.6a and 4.3.6b show the changes of oscillator outputs and error signals when feedback gains (or force sensor outputs) of the eight oscillators are changing in time. If the gains approach to a common value (1 in the simulations), all error signals approach zero (Figure 3.4.6a).

To see the effect of one oscillator on others, we change the gain of one oscillator (number 1) after all gains reach the steady-state (Figure 4.3.7a)\textsuperscript{6}. Oscillation magnitudes still being approximately 1.5, the change in oscillator 1 is detected by all agents. Figure

\textsuperscript{6} Such a change in the feedback gain corresponds to a case where robots move over uneven surfaces.
4.3.7b shows the error signal computed by agents 5 and 8. There is a time lag between
detection times of agent 5 and agent 8; it takes time for a change in the feedback gain of
one agent to affect other agents.

In the case of two oscillators with "significantly" different gain factors, we found
that at least one agent can detect the problem. Figure 4.3.8 shows that the gains of
oscillators 1 and 2 differ from the others. Although the difference between oscillators 8
and 1 or oscillators 2 and 3 are not large, the difference between oscillator gains 1 and 2
is. As the error signals for agents 3 and 4 reveal, the problem is detected by agent 3 (as
well as agents 1 and 2). Even after the feedback gain of the second oscillator changes to
decrease the gain (force) range, error signal 4 shows that forces are not equal.

The errors in feedback gains (force sensors) can be detected by other oscillators,
even if they are not nearby. In a more problematic case, three agents' sensor outputs differ
from their teammates as shown in Figure 4.3.9. Again error signals of agents 4 and 8 show
the changes as well as agents 1, 2, and 3. The ring coupling of the oscillator outputs
enables a team of robots to sense the "stability" of pallet-robot system. Robots may adjust
their oscillator feedback gains by exerting more of less force — thereby equalizing vertical
forces until oscillators agree.

It is obviously necessary to adjust the allowable ranges for error signals. It should
be adjusted so that the problems may be detected no matter what the number of robots is.
Figure 4.3.10 shows a similar case to simulation given in Figure 4.3.7b with only five
agents. Again the error signal in the "farthest" agent shows the problem in one agent, and
the level of the error signal is slightly greater than the case with eight agents. As long as
the number of agents is more than or equal to 5, the same parameters for allowable ranges
are likely to work for all cases.

When additional oscillators are successively added to the system, the change in the
signal levels is less and less. Addition of new oscillators to ring network of eight
oscillators does not have a perceivable effect. Similar characteristics are observed in the
period and amplitude of the entrained oscillations by Bay [5]. Furthermore, a discrete
mapping function from sensor output to the feedback gain may improve the detection of
range violations. Yet another problem is that the parameters working successfully in the
horizontal floor will not work on an inclined surface. Use of inclinometers may solve the
problem. Inclinometer output can be used to adapt the range parameters.

---

7The "significance" of the changes is related to the parameters which define the allowable force range,
and to mapping function.
Although we defined the feedback gain range to be 0.7-1.3 and assumed that the gains generally approach unity, this will not be the case in Army-ant operation, because the average load on robots will change depending on the payload and number of agents. However, the method described here works no matter where the average value of the gain is with respect to range limits.

4.3.3 Carrying the pallets

In transporting the pallets, Army-ant robots may encounter several problems. Besides the problems of aligning all robots along a common direction of travel [34] or distributing the load equally, there still exists problems such as acceleration/deceleration or obstacle avoidance during a pallet transport. A group of robots forming a team has to move in accord.

The idea of "heartbeats" seems to be a solution to this kind of problem. Again, a broadcast warning signal can be used to create a team "conscience." Yet, the use of such a signal must be analyzed in depth. Suppose one of the robots encounters a small obstacle (perhaps a malfunctioning robot) creating problem for only that robot during the transportation of a pallet. Should the robot "warn" its teammates and force the whole team to stop and take action (Figure 4.3.11a), or should it simply "leave" the team temporarily to solve the problem by itself —by circling the obstacle? (Figure 4.3.11b)

The second approach seems to be more appropriate for our scenario —although it may be more difficult to carry out. Enabling agents to stop a whole team may destroy the robustness of this decentralized system. For example, a malfunctioning agent can stop a pallet transportation by an erroneous warning signal. Therefore, we must keep the "responsibility" of the agents as low as possible. A team should be able to leave that agent and continue the process.

Furthermore, two teams of robots carrying pallets may come across one another and block each other's way. In such a case, ID numbers of the leaders may be used to solve the conflict. If an individual robot encounters a team carrying a pallet, it should give the right-of-way to the team by sensing the leader's signal indicating that it is committed to load transportation.

Another problem may be finding the set-down point in the warehouse. Use of extra beacons defining specific points in the warehouse may be chosen as a solution. Teams carrying loads can find their way to the destination point by following special beacons
deployed so that robots can find the set-down point if they can detect one of the beacon signals initially. This method is explained in more detail in section 4.3.4.2.

4.3.4 Other Issues

4.3.4.1 Dispersal and Rearrangement to Other Teams

It will probably be the leader's task to decide whether the set-down point is reached or not. The set-down condition will be signaled by the leader, and another specific signal can be used to "release" the agents. Receiving "release" signals, a search behavior will again become active, suppressing the leader-following behavior, and agents will disperse in the warehouse.

4.3.4.2 Battery Recharge

Army-ant robots will be powered with batteries. Therefore, it will be necessary to recharge the units during operation. Robots can check their battery levels before engaging in a search or a team forming, and decide when to go to the recharge site. Since the Army-ant robots do not have a map knowledge of the warehouse, it may be difficult to find this site. We may define a behavior (e.g., search-for-recharge-site) which becomes active if the battery level is below a preset value. This behavior will force the agent to look for a different type of beacon which is deployed in the warehouse. It will work as a "reflex." A robot will approach the first special beacon it sees, and then lock onto the next beacon signal with higher "priority." Assigning priorities to beacons will enable robots to move toward the recharge site, no matter which beacon they detect.

4.3.4.3 Complexity

In this chapter, we offered solutions to some of the problems that will probably be encountered while realizing Army-ant robots. Several problems in different phases of the scenario can be solved using beacons, communication methods, and simple decision algorithms. The Army-ant swarm will take its final shape during realization. We expect more problems to emerge. On the other hand, some of the problems described here may become insignificant due to technical capabilities of Army-ant robots. A brief chapter of technical assessment, related to solutions given here, follows.
Figure 4.1.1 Behavior modules of the Army-ant robots as discussed in this chapter
Figure 4.2.1 Sample warehouse plans
Figure 4.2.2 Initially only three robots detected the beacon. Three more robots were able to find the beacon by following one of those robots.
Figure 4.2.3 Although initially only one robot (\*) detects the beacon, the rest of the pack finds the beacon following that robot, and other teammates.
Figure 4.2.4 One agent searching a beacon in the warehouse
(+: initial position of the agent, *: beacon position).
**Figure 4.2.5** One agent searching a beacon in the warehouse. 
(+: initial position of the agent, *: beacon position)
Figure 4.2.6 Eight agents searching for a beacon
(+: initial positions, *: beacon)
Figure 4.2.7 First five steps and final positions of ten agents searching for a beacon. (+: initial positions, *: beacon)
Figure 4.2.8 Four typical paths from the simulation with initial conditions given in Figure 4.2.7.
**Figure 4.2.9** *Wait-for-me* signal: Agent B receiving the signal from agent A waits, and upon detection of the goal by agent A, agent B is released.

**Figure 4.2.10** Use of range sensors for navigation
Figure 4.3.1 Ring formation and coupling coefficients for eight agents

Figure 4.3.2 Difference between 1st and 4th sensor is quite large while all sensor readings between successive robots are in allowable range
Figure 4.3.3 Coupled VDP oscillator with added feedback gain

Figure 4.3.4 Mapping from force sensor output to feedback gain
Figure 4.3.5 Oscillator outputs ($x_i$) and error signals ($e_i = x_i - x_{i-1}$) of eight agents in ring formation (all feedback gains $\lambda_{ii}$ are 1, all coupling coefficients $\lambda_{ij}$ are +0.3).
Figure 4.3.6a Oscillator feedback gains $\lambda_{ij}$ and outputs of eight agents in ring formation (All coupling coefficients $\lambda_{ij}$ are +0.3).
Figure 4.3.6b Oscillator feedback gains $\lambda_{ii}$ and error signals of eight agents in ring formation (All coupling coefficients $\lambda_{ij}$ are +0.3).
**Figure 4.3.7a** Oscillator feedback gains $\lambda_{ii}$ and outputs of eight agents in ring formation (All coupling coefficients $\lambda_{ij}$ are +0.3).
Figure 4.3.7b Oscillator feedback gains $\lambda_{ij}$ of eight agents in ring formation and error signals of two agents (All coupling coefficients $\lambda_{ij}$ are +0.3).
Figure 4.3.8 Oscillator feedback gains $\lambda_{ii}$ of eight agents in ring formation and error signals of two agents (All coupling coefficients $\lambda_{ij}$ are 0.3).
Figure 4.3.9 Oscillator feedback gains $\lambda_{ii}$ of eight agents in ring formation and error signals of two agents (All coupling coefficients $\lambda_{ij}$ are +0.3).
Figure 4.3.10 Oscillator feedback gains $\lambda_{ii}$ of five agents in ring formation and error signals of an agent (All coupling coefficients $\lambda_{ij}$ are $+0.3$).
Figure 4.3.11 A small obstacle is encountered during pallet transport

Figure 4.3.12 An example of beacon positioning and priority assignment
5. Technical Assessment

This chapter is a brief discussion of sensors and communication methods applicable to the Army-ant scenario. In Army-ant robots, there would be three main modules for interaction with the environment:

i) communication module
ii) beacon detection module
iii) obstacle avoidance module
iv) locomotion module
v) force/position detection module

We do not discuss force sensors, inclinometers, or locomotion devices—which are necessary for modules iv and v—here. Present technology used in robot manipulators and mobile robots is adequate for application to Army-ant robots. We will discuss several types of sensors that can be applied in realizing the first three modules. Our proposal of the devices and methods specified here is based on the scenario described in Chapters 3 and 4. The list of the devices to be used in Army-ant robots will take shape in the process of realizing these robots. Of course, the main decision factor will be the cost per agent.

5.1 A Communication Method: Binary Coded Signals

As explained in previous chapters, Army-ant robots need to communicate with each other in order to share information available to some of the team members. Binary coded signals can be used to inform teammates about the goal and problems during the pallet transport. Request signals described in Chapter 4 (warning, wait-for-me and status of agents) can be coded and transmitted using short range radio communications. It is possible to address a specific agent using this coded signals. For example, an agent can
broadcast *wait-for-me* signal to all robots. The structure of the coded message can be arranged so that only the robots whose behavioral condition allow them to respond, can stop for their (would-be) teammate. Those agents may continue when the same signal is not received again in some predefined time interval.

Besides these messages, it is also possible for agents to send information that is necessary for the coupled VDP oscillators method described in Chapter 4. Ring formation can be achieved by exchanging data identifying the agents. However, oscillator (or other continuous analog) signals will probably be transmitted by RF links. To keep the number of necessary channels to minimum, agent beacons can assume different roles at different phases.

Agents may share a single channel provided that they can detect whether the channel is busy or not. Genovese et al, defined the necessary strategy for such a communication method in [16]: Each robot would have to determine the presence of the carrier signal using a radio transceiver module before transmitting any data in order to avoid "collisions," and a control bit added to the end of the binary coded signal have to be used to detect errors to achieve "robust" communications.

Digital communications via RF and IR links are becoming cheaper and there are a variety of companies providing solutions. Several radio modems that use fixed or spread spectrum techniques are available. Some of those devices can even be directly plugged into the serial ports of the microprocessors. Necessary devices for an RF link (i.e., FSK Data Modulator/Demodulators, Digital fixed Frequency Transmitter/Receivers, Computer Controlled Synthesizers, etc.) can be obtained around $50-100 each.

The amount of the data transferred by agents at each transmission would be quite small, considering that we can fit all data needed into a few bytes. Therefore, the time interval between transmission attempt and end of transmission should be short.

### 5.2 Sensors

In addition to the communication system for complex data transfer, Army-ant robots have to be equipped with sensors to detect obstacles and beacons for goal search and navigation. Acoustic and radiowave (microwave) sensors are the two most commonly used sensors when there is a need for computing the distance to an obstacle. These
sensors measure the time between emission and reception of the wave (time-of-flight method) to calculate the distance. They are both "active"\footnote{A sensor is active if it emits an information which has to interact with the environment before being analysed.} sensors, and can be used as short-range proximity sensors for obstacle avoidance and teammate detection.

Although the time-of-flight method is useful to detect obstacles and teammates in the Army-ant scenario, agents also have to detect the distance to a beacon --where time-of-flight methods are impracticable. However, there exist some short-range radars (radiowave sensors) realized by RCA, and a system using reflected energy instead of the time of flight is presented in [1]. This method should enable Army-ant robots to calculate their distance to the pallet and teammates.

Acoustic sensors have several advantages over radiowave sensors: they are smaller and not sensitive to the nature of the target, and they can be used for underwater applications. On the other hand, for space applications, radiowave sensors are the only choice, their physical principle being usable in space and for large distances.

Ultrasonic sensors are likely to be used in Army-ant robots. Ultrasonic sensors may use a large frequency range (100 KHz-1Mhz) very effectively. Resolution on distance measurements can be as low as 0.01 mm in air [23]. Qualities of ultrasonic sensors important to robotics applications are sensitivity, dynamic range, linearity, space and time resolution and signal processing. However, they are not flawless; acoustical waves projected onto a surface under an angle are deflected like a light by a mirror, and no echo returns to the receiver. They are also susceptible to wind --which may be an important factor in outdoor applications.

The most widely used acoustic sensor is the Polaroid acoustic sensor. It has a range between 26 cm. and 10.7 m, and +-01 resolution over the entire range, and requires 6.0 V DC. Its acceptance angle is 20 degrees. Transducer and ranging modules weight 8.2 and 18.4 gm respectively. There are also other sonar ranging modules under $50. With these specifications, sonar sensors seem to be a feasible choice.

One of the most important problems in Army-ant robot navigation is to detect the direction of the beacon signals. Again, two types of signals can be used for direction finding: radio or ultrasonic waves.

The direction of ultrasonic beacons can be found by phase comparison between pairs of acoustical receivers. A navigation system developed in MEL, Tsukuba, Japan uses three independently rotating pairs of receivers to find the direction of three ultrasonic
beacons [2]. However, the application of this idea to Army-ant robots seems difficult since they are to carry payloads.

Other than rotating acoustic sensors, geometrical arrangements of multiple sensors or an array of acoustical transducers are well-known methods for environment perception. These methods can also be adapted to Army-ant robots for beacon detection purposes. Arrays of acoustical transducers require interpretation of the signal received, but this can be realized by a small microprocessor. In [13], it is stated that acoustical phased arrays are low cost, and are useful for applications with mobile robots to give a rough perspective of the environment. Therefore, such a system must be adequate to detect the direction of a beacon.

Another type of sensor that we do not consider is laser range finders. Although they are more precise than other range finding devices, the inexpensive ones have maximum accuracy of 1/4 inch, and cost approximately $600, four to five times more expensive than sonar range finders. A more futuristic approach is to use gas sensors. Pelosi and Persaud state that gas sensors, based on polymers of aromatic and heteroaromatic molecules that exhibit electrical conducting properties, can be used to detect specific odors [29]. Presently, reducing and oxidizing gases can be monitored by the impedance of SnO$_2$, ZnO, TiO$_2$, and other oxide surfaces. A gas chromatographic air analyzer was fabricated on a silicon wafer using solid state electronic techniques [36].

Considering the number of robots, there will be a large number of signals on all the used channels. While it is possible to 'queue' the data transmission in Digital RF or IR communication, beacon detection may be highly difficult due to the number of signals emitted and/or reflected. It is difficult to discuss the performance of such a scenario before realizing Army-ant robots. On the other hand, we know that a certain amount of noise, whose emergence can be regulated by the communication system, can even be advantageous for pallet search as in the foraging process of ants.
6. Conclusions and Future Research

This chapter briefly summarizes the major results of this study, discusses their significance and potential applications, and outlines suggestions for future work.

6.1 Conclusions

The work reported in this thesis enables us to summarize the following ideas:

- Modeling Army-ant robots as a self-organizing system has many advantages. Adaptive, collective and "complex" systems resulting from simple individual behavior are what the Army-ant scenario envisages. Simplicity of the individual agents is an important factor in implementation. However, the size and nonlinear character of self-organization leaves little possibility for definitive analysis.

- It is possible to geometrically arrange Army-ant robots by using a distributed approach. Robots can spatially organize themselves around a goal using only local information transferred by broadcast signals. This method is more advantageous (and faster) than a conventional centralized control methods, especially when the number of agents is large. The methods described in the text can be applied directly to other multirobot systems in underwater, planetary surface and space missions.

- It is also possible to separate the agents into different teams around different goals. The size of these teams can be determined by the difficulty of the assigned task. The team formation can again be achieved by using broadcast signals. Although there is no hierarchy between agents, temporary "leader" assignments seem to be necessary to overcome several problems. However, this not a violation of the homogeneous character of the population since all agents may became one and replace the leader.
• Use of communication channels enables cooperating robots to form a decision mechanism. Army-ant robots can share individual information using the coupled VDP oscillator scheme, and consequently "act" intelligent. The method described for collective load bearing may have several other applications in the Army-ant scenario.

• Driven by several behavioral modules, Army-ant robots form a large dynamic system. Interaction "rules" between agents have to be adjusted (or have to self-adjust) carefully to the environment and/or tasks to define the "responsibilities" of agents during different phases. This necessary for system robustness.

• The robot beacon signals have to carry some information other than indicating the relative position of the agent. Army-ant robots must signal their "status" to their teammates in order to find goals in areas with size larger than the detection region of the agents and in warehouses with "maze" structure. These signals are to be used in a similar way pheromone fields are used in insect societies.

• Self-organizing mobile robots need to be equipped with communication devices as well as beacons and detectors. Binary coded signals transferred at RF may prove to be highly useful at different stages of the Army-ant scenario. Implementation of such a system with broadcast characteristics is feasible. However, the computation of the distance to beacon(s) is a problem to be solved, considering that there will be many interfering signals.

6.2 Areas of Further Development

Several topics briefly mentioned in the text deserve further attention:

• Simulations described in Chapter 3 do not take into account parameters such as inertial constraints of the agents. The algorithms devised here can be extended to more complex programs for more realistic simulations incorporating robots'
characteristics and collision avoidance. Such algorithms may be realized by object-oriented programming in an X-window environment.

- Using Lego Dacta robots to test the performance of the algorithms in the text may be very informative. Performance analysis under obstacle avoidance constraints can be analyzed better using actual mobile robots than computer simulations. Such an implementation would also provide useful information about signal interference and distance and/or direction detection.

- Topics related to multiagent systems, such as Petri nets, cellular automata, Kohonen networks and genetic algorithms, have to be investigated for possible applications to Army-ant scenario. Petri net and cellular automata techniques can have applications in understanding the swarm behavior resulting for individual models. Again, genetic algorithm applications may be suitable for changing the behavior models (i.e. parameters acting on behavior modules and connecting them) to create an adaptive system.

- The use and feasibility of non-linear "heartbeats" in Army-ant robots have to be explored in depth. The variable size of robot teams may be a problem for oscillator couplings.

- The use of gas sensors as a distant alternative to radio and acoustic signals must be kept in mind. On going research on detection of specific odors may prove to be useful in mobile multirobot systems since pheromone fields play an important role in self-organizing insects.
Appendix

A. Programs

Main algorithms of the Matlab programs created for simulations are given here. Subroutines for screen plots and user interfaces are omitted. Parameter values used during simulations are given to provide information to those who may want to reproduce our results.

A.1 Algorithm Form-Circle

% cl.m
% circle forming
.
.
gs1=3;
gs2=10000;
.
.
for t=1:nt,
    t
    for i=1:nrob,
        distg=norm(goal-rob(:,i));
        gvec(:,i)=goal-rob(:,i);
        goalvec(:,i)=(goal-rob(:,i))/distg; % vector to goal
    end
    for i=1:nrob,
        if norm(gvec(:,i)>d+eps,
            gs=gs1+(norm(gvec(:,i))^2)/gs2;
            rr=0;rtrv(:,i)=[0;0];minx=1;
        elseif norm(gvec(:,i))<d-eps,
            gs=gs1;
            %+gs1/2 /3
            rr=0;rtrv(:,i)=[0;0];minx=1,
        else
            for j=1:nrob,
if j~=i,
    rtrd(j)=norm(rob(:,i)-rob(:,j));
    rtrv(:,j)=(rob(:,i)-rob(:,j))/rtrd(j);
end
end
rtrd(i)=100;
[clst,minx]=min(rtrd);
gs=0; rr=1;
end
nxrob(:,i)=rob(:,i)+gs*goalvec(:,i)+rr*rtrv(:,minx);
end

rob=nxrob;
end

A.2 Algorithm Fill-Circle

% c2.m
% algorithm fill-circle

gs1=3;
gs2=10000;

for t=1:nt,
    t
    for i=1:nrob,
        distg=norm(goal-rob(:,i));
        gvec(:,i)=goal-rob(:,i);
        goalvec(:,i)=(goal-rob(:,i))/distg; % vector to goal
    end
    for i=1:nrob,
        if norm(gvec(:,i))>d,
            gs=gs1+(norm(gvec(:,i))^2)/gs2;
            rr=0;rtrv(:,i)=[0;0];minx=1;
        else
            for j=1:nrob,
                if j~=i,
                    rtrd(j)=norm(rob(:,i)-rob(:,j));
                    rtrv(:,j)=(rob(:,i)-rob(:,j))/rtrd(j);
                end
            end
        end
    end
end
end
rtrd(i)=100;
[clst,minx]=min(rtrd);
gs=0; rr=2;
end
nxrob(:,i)=rob(:,i)+gs*goalvec(:,i)+rr*rtrv(:,minx);
end

robnxrob;
end %t

A.3 Algorithm Form-Sphere

% sphere.m
% sphere forming
.
.
eps=3;
gs1=4;
gs2=10000;
.
.
for t=1:mt,
t

for i=1:nrob,
    distg=norm(goal-rob(:,i));
gvec(:,i)=goal-rob(:,i);
goalvec(:,i)=(goal-rob(:,i))/distg; % vector to goal
end
for i=1:nrob,
    if norm(gvec(:,i))>d+eps,
gs=gs1+(norm(gvec(:,i))^2)/gs2;
    rr=0;rtrv(:,i)=[0;0;0];minx=1;
    elseif norm(gvec(:,i))<d-eps,
gs=-gs1/2;
    rr=0;rtrv(:,i)=[0;0;0];minx=1;
    else
        for j=1:nrob,
            if j~=i,
                rtrd(j)=norm(rob(:,i)-rob(:,j));
rtrv(:,j)=(rob(:,i)-rob(:,j))/rtrd(j);
            end
            elseif norm(gvec(:,i))>d+eps,


A.4 Algorithm Form-Paraboloid

% program paraboloid.m

ep=1;

r1=ceil(rand*nrobot);
r2=r1;
while r2==r1,
    r2=ceil(rand*nrobot);
end

p2=[500;500;150]; % final positions for r1 and r2
p1=[500;500;850]; % (vector p1-p2 must be // to z-axis)
a=norm(p1-p2); % assumed to be fed to robots
b=300; % neig.d of r2 : fed to robots

for i=1:nrobot,
    dp1(i)=norm(robot(:,:,i)-p1);
    dp2(i)=norm(robot(:,:,i)-p2);
end

[xx,r1]=min(dp1);
[xx,r2]=min(dp2);

% user interface 'Go'
subr=['p51'];
goc=uicontrol('style','push','units','normal','string','Go','pos',[0.9 0.7 0.08 0.07],'callback',subr);

---
% program p51.m
% callback subroutine of GOC in paraboloid.m

for t=1:ntime,
  
t
  nxrobot(:,r1)=robot(:,r1)+(1+norm(p1-robot(:,r1))^2/3000)*(p1-
robot(:,r1))/norm(p1-robot(:,r1));
  nxrobot(:,r2)=robot(:,r2)+(1+norm(p2-robot(:,r2))^2/3000)*(p2-
robot(:,r2))/norm(p2-robot(:,r2));
  
  for i=1:nrobot,
    
      if i==r1 & i==r2,
        
v1=nxrobot(:,r1)-robot(:,i);d1=norm(v1);
  
v2=nxrobot(:,r2)-robot(:,i);d2=norm(v2);
  
z=(a^2-d1^2+d2^2)/(2*a);
  
x2y2=(d2^2-z^2);
  
        if d2>d1,

          nxrobot(:,i)=robot(:,i)+(1+d2^2/8000)*v2/d2;
          % move twd r2

        elseif d2>b,

          nxrobot(:,i)=robot(:,i)+(1+d2^2/8000)*v2/d2
          % move twd r2

        elseif z>x2y2/(4*a)+ep,

          nxrobot(:,i)=robot(:,i)-(1+(d1-a)^2/5000)*v1/d1+2*v2/d2;
          % move away from r1 to r2

        elseif z<x2y2/(4*a)-ep,

          nxrobot(:,i)=robot(:,i)+(1+(d1-a)^2/5000)*v1/d1-2*v2/d2;
          % move twd r1 + away from r2

        else

          for j=1:nrobot,
          
            if j==i & j==r1, % closest mate including r2
              
               rtrd(j)=norm(robot(:,j)-robot(:,i));
            
            else

            rtrd(j)=1000;

          end

          end

          [minrtrd,xnd]=min(rtrd);

          nxrobot(:,i)=robot(:,i)-2*(robot(:,xnd)-robot(:,i))/minrtrd;

        end

      end

    end

  end % time loop ends
% set color of the robots not on the defined region
for i=1:nrobot,
    if robot(3,i)-((robot(1,i)-500)^2+(robot(2,i)-500)^2)/((4*a)+200)>ep |
        norm(robot(:,i)-p2)> b & i=r1,
        set(plr(i),'xdata',robot(1,i), 'ydata',robot(2,i), 'zdata',robot(3,i), 'vis','on');
end

A.5 Algorithm Switching-Distance

% ds.m
% p1 (matlab 4.0 program 1 switching ds)
% call subroutines: p11.m, p12.m p13.m
global subr

ng=3; % number of goals
nrobot=20; % number of robots
eyea=1000; % detection region
ntime=50; % final time.
dn=20; % neighborhood
ttt=2; % repulsion vector magnitude
gs=5;
gs2=1500;
formula='gs1+DTG^2/gs2'; % displacement formula for region d>dn

goc=uitcontrol('style','push','units','normal','pos',[0.03,.19,.20,.06], 'string','Sim','call',
    subr);

for c=1:ng+1, % initial goal configuration
gv(1,c)=0;
    for cc=1:nrobot,
        if gv(cc)==c,
            gv(1,c)=gv(1,c)+1;
        end
    end
end

for t=1:ntime, % time loop
ds=findtime(sdfunc,t,deg);

set(f1p3,'xdata',t,'ydata',ds)

---

% program p11.m
% callback subroutine of user interface GOC in p1 (ds.m)

for i=1:nrobot,
    %signal assignments
    for k=1:ngoal,
        distg(k)=norm(goal(:,k)-robot(:,i));
    end
    [clog,ind]=min(distg);
    if clog<=eyea,
        gv(i)=ind;
    else
        gv(i)=ngoal+1;  % if robot can't see any goal, assign
        % to virtual goal at [3000,3000]
    end
end

for i=1:nrobot,
    distg=norm(goal(:,gv(i))-robot(:,i));
    gvec(:,i)=goal(:,gv(i))-robot(:,i);
    goalvec(:,i)=(goal(:,gv(i))-robot(:,i))/distg;  % vector to closest goal
    if norm(gvec(:,i))<=dn,
        signal(i)=gv(i);  % if in dn, signal
    else
        signal(i)=0;  % if not, do not signal
    end
end

for i=1:nrobot,
    if norm(gvec(:,i))>1500,  % no goal in 'eye', then move random
        rr=0;rtvec(:,i)=[0;0];
    elseif norm(gvec(:,i))>dn,  % can be changed
        DTG=norm(gvec(:,i));
        gs=eval(formul);
        rr=0;rtvec(:,i)=[0;0];
    else
        mates=[0;0];rtrd=0;rtvec(:,i)=[0;0];tt=1;
        for j=1:nrobot,
            if signal(j)==signal(i) & j~=i,
                mates(:,tt)=robot(:,j);
                tt=tt+1;
            end
end
end
% teammates in d known
if mates(:,1)==[0,0], % no mates
    rtrvec(:,i)=[0,0];
else
    for j=1:tt-1, % there are some mates around
        rtrd(j)=norm(mates(:,j)-robot(:,i));
    end
    [dummy, minx]=min(rtrd);
    rtrvec(:,i)=(mates(:,minx)-robot(:,i))/rtrd(minx);
end
% move away from closest teammate
rr=rtrr, gs=0;
end
nxtrob(:,i)=robot(:,i)+gs*goalvec(:,i)-rr*rtrvec(:,i);
% change goal if inside ds
if (norm(nxtrob(:,i)-goal(:,gv(i)))<=ds & ds>dn) | (norm(nxtrob(:,i)-goal(:,gv(i)))>1500),
    tr=1;
    gv(i)=ceil(rand*ngao);
    while tr<=20 & norm(nxtrob(:,i)-goal(:,gv(i)))>eyea,
        gv(i)=ceil(rand*ngao);
        tr=tr+1; % ideal case: tr=very large
    end
    if norm(nxtrob(:,i)-goal(:,gv(i)))>eyea, %if still no goal "around"
        gv(i)=ngao+1;
    end
    st=t;
end
robot=nxtrob;
.
.
end % time loop ends
.
.

A.6 Algorithm Family values

% program p3 (famval.m)
% main program for simulation famval
global subr
.
.
% initial values for parameters
ngao=3;
nrobot=20;
nctime=30;
dn=20;
fv=[1:nrobot 0];
first=zeros(1,nrobot);  % vector for "first" def.
gs1=4;
gs2=1500;
trt=2;
eye=1500;  % robot detection region of all agents
eyeg=1500;  % goal detection region of all agents
formula='gs1+DTG^2/gs2',  % displacement formula for r>d
.
goc=uicontrol('style','push','units','normal','pos',[0.03,.22,.20,.06],'string','Sim','callback',
  subr);

---

% program p31.m
% callback subroutine for user interface GOC in famval.m
.
for i=1:nrobot,  % signal assignments
  for k=1:ngoal,
    distg(k)=norm(goal(:,k)-robot(:,i));
  end
  [elog,ind]=min(distg);
  if eelog<=eye,
    gv(i)=ind;
  else
    gv(i)=ngaoal+1  % if robot can't see any goal assign
  end
  % virtual goal at [3000,3000]
end

for c=1:ngoal+1,  % initial goal configuration
  gvt(1,c)=0;
  for cc=1:nrobot,
    if gvc(cc)==c,
      gvt(1,c)=gvt(1,c)+1;
    end
  end
end
.
for t=1:nctime,
  set(f1p2,'xdata',[-.6 -.6+1.2*t/nctime])

111
for i=1:nrobot,
    distg=norm(goal(:,gv(i))-robot(:,i));
    gvec(:,i)=goal(:,gv(i))-robot(:,i);
    goalvec(:,i)=(goal(:,gv(i))-robot(:,i))/distg; % vector to closest goal
    if norm(gvec(:,i))<=dn, % if in dn, signal
        signal(i)=gv(i);
        if first(i)==0 & length(find(first==gv(i)))==1,
            % if you're the first one in neig.d
            first(i)=gv(i);
        end
    else
        signal(i)=0; % if not, do not signal
    end
end

fx=find(first>0); % fam.val. shifting
for i=1:length(fx),
    numseen(i)=0;
    for j=1:length(fx),
        if j==i & norm(robot(:,fx(i))-robot(:,fx(j)))<eyer,
            % if you're seeing jth "first" agent
            numseen(i)=numseen(i)+1;
            dist(i,j)=abs(fv(fx(i))-fv(fx(j)));
            if dist(i,j)>nrobot/2,
                dist(i,j)=abs(fv(fx(i))-nrobot-fv(fx(j)));
            end
            if dist(i,j)>nrobot/2,
                dist(i,j)=abs(fv(fx(i))+nrobot-fv(fx(j)));
            end
            else
                dist(i,j)=100;
            end
        end
    end
[clst,clstx]=min(dist(:,:));
if clst==100,
    nfv(fx(i))=fv(fx(i));
elseif fv(fx(i))>fv(fx(clstx)),
    if abs(fv(fx(i))-fv(fx(clstx)))>nrobot/2,
        nfv(fx(i))=fv(fx(i))-5;
    else
        nfv(fx(i))=fv(fx(i))+5;
    end
else
    if fv(fx(i))=fv(fx(clstx)),
        nfv(fx(i))=fv(fx(i))-5+round(rand);
elseif abs(fv(fx(i))-fv(fx(clstx)))>nrobot/2,
    nfv(fx(i))=fv(fx(i))+.5;
else
    nfv(fx(i))=fv(fx(i))-5;
end
if nfv(fx(i))>nrobot,
    nfv(fx(i))=nfv(fx(i))-nrobot;
end
if nfv(fx(i))<0.5,
    nfv(fx(i))=nfv(fx(i))+nrobot;
end
fam(fx(i))=nrobot/(numseen(i)+1);
end
if exist('nfv')>0,
    fv(fx)=nfv(fx);
end

for i=1:nrobot, % next locations
    if norm(gvec(:,i))>1500, % if no goal in 'eye', then move random
        rr=0; rtrvec(:,i)=[0;0];
        goalvec(:,i)=-1+2*rand(2,1); gs=4; % can be changed
    elseif norm(gvec(:,i))>dn, % go twd goal
        gs=gs1+(norm(gvec(:,i)))^2/gs2;
        rr=0; rtrvec(:,i)=[0;0];
    else
        mates=[0;0];rtrd=0; rtrvec(:,i)=[0;0]; tt=1;
        for j=1:nrobot,
            if signal(j)==signal(i) & j~=i,
                mates(:,tt)=robot(:,j);
                tt=tt+1;
            end
        end
        % teammates in d known
        if mates(:,1)==[0;0], % no mates
            rtrvec(:,i)=[0;0];
        else
            for j=1:tt-1, % there are some mates around
                rtrd(j)=norm(mates(:,j)-robot(:,i));
            end
            [dummy, minx]=min(rtrd);
            rtrvec(:,i)=(mates(:,minx)-robot(:,i))/rtrd(minx);
        end
        % move away from closest teammate
        rr=rtrr; gs=0;

    end
end

nxtrob(:,i)=robot(:,i)+gs*goalvec(:,i)-tr*trvec(:,i);
% change goal if (first's fv-yours)>fam/2
ix=find(first==gv(i));
if length(ix)==0,
    ix=nrobot+1; fam(ix)=10000;
end
if norm(goalvec(:,i)<=eyer & first(i)==0 & abs(fv(i)-fv(ix))>(.5+fam(ix))/2 & ((abs(fv(i)-
    fv(ix))+nrobot)>(.5+fam(ix))/2 & fv(i)<fv(ix)) | (abs(fv(i)-fv(ix)-nrobot)>(.5+fam(ix))/2 &
    fv(i)>fv(ix))),
    tr=1;
    gv(i)=ceil(rand*ngao);% ideal case: tr=very large
    while tr<=20 & norm(nxtrob(:,i)-goal(:,gv(i)))>eyeg,
        gv(i)=ceil(rand*ngao);
        tr=tr+1;
    end
end
if norm(nxtrob(:,i)-goal(:,gv(i)))>eyeg, %if no goal "around"
    gv(i)=ngao+1;
end
st=t;
end

robot=nxtrob;
end % time loop

A.7 Algorithm Find-pallet

% finding pallets in the warehouse
% simulation for beacon, robot interactions and possible
% strategies for finding pallets

% general algorithm:
% 1. if see pallet, move twd. pallet
% 2. if see no pallet but another agent going to destination, follow that agent
% 3. if no pallet nor agent, follow wall


global obsm

% robot and goal properties
nrobot=10;
ngao=1;
neig=60;
eyeg=300;eyer=1200;
th=rand(1,nrobot)*pi;
dir=[cos(th),sin(th)];
step=30;
angstep=36; % i.e. theta=180/18=10
dirch=zeros(1,nrobot);
maxdirch=5;
robot=zeros(4,nrobot);
ntime=300;

% detection vectors
seegr=zeros(2,nrobot); % =max if goal seen, <=max-1 if another agent seen

% row1 status row2=ID of agent seen

seeegr=seege;
glev=20;

for t=1:ntime,

t seege

% set signals according to detection status
seege=zeros(2,nrobot);
for i=1:nrobot,
    if all([norm(robot(:,i)-goal)<=eye see(robot(:,i),goal)==1])==1,
        seege(i,i)=glev;
    end
end

for lev=glev:-1:glev-nrobot+1,
    for i=1:nrobot,
        if seege(i,i)==0, % if no detection yet
            maxx=2000;
            for j=1:nrobot,
                if all([norm(robot(:,i)-robot(:,j))<=min(eyer,maxx)
                    see(robot(:,i),robot(:,j))==1 i=j seege(1,j)==lev+1])==1,
                    seege(i,i)=lev;seege(2,i)=j;maxx=norm(robot(:,i)-
                    robot(:,j))
                end
            end
        end
    end
end

% here we know all agents' detection status
% set direction depending on the detection
for i=1:nrobot,
    if seegr(1,i)==glev,
        dir(:,i)=-(robot(:,i)-goal)/norm(robot(:,i)-goal);  % to goal if seen
        dirch(i)=0;
    else
        if seegr(1,i)==0,
            dir(:,i)=-(robot(:,i)-robot(:,seegr(2,i)))/norm(robot(:,i)-
                     robot(:,seegr(2,i))),
            dirch(i)=0;
        end
    end
end
if dirch(i)>maxdirch,
    left=robot(:,i)+100*[0 -1; 1 0]*dir(:,i);
    right=robot(:,i)+100*[0 1; -1 0]*dir(:,i);
    if min(left)>0 & max(left)<1200,
        if obsm(left(1),left(2))==0
            dir(:,i)=[0 -1; 1 0]*dir(:,i); dirch(i)=0;
        end
    elseif min(right)>0 & max(right)<1200,
        if obsm(right(1),right(2))==0,
            dir(:,i)=[0 1; -1 0]*dir(:,i); dirch(i)=0;
        end
    end
end
end

% move in the direction computed above (adjust direction if necessary)
for i=1:nrobot,
    if norm(robot(:,i)-goal)>neig,
        newp=round(robot(:,i)+step*dir(:,i));
        if (max(newp)>1200 | min(newp)<1),
            newp=[1 1];
        end
        if see(robot(:,i),newp)==0,
            newp=[1 1];
        end
        turn=1;
        tpos=robot(size(robot,1)-5:size(robot,1),i);
        while obsm(newp(1),newp(2))==1,
            % if obstacle deducted at step range, change direction
            th(i)=th(i)+turn*sign(isodd(turn)-.5)*pi/angstep;
            % scan +/- starting with ccw (+angstep)
            dir(:,i)=[cos(th(i));sin(th(i))];
            newp=ceil(robot(:,i)+step*dir(:,i));
        end
    end
end
if (max(newp)>1200 | min(newp)<1),
    newp=[1; 1];
end
if see(robot(:,i),newp)==0,
    newp=[1; 1];
end
    turn=turn+1;
end
    if ((tpos(1)-tpos(3))^2+(tpos(2)-tpos(6))^2)^.5<=step*sqrt(3),
        % 60 or more turn
        dirch(i)=dirch(i)+1;
    end
    robot(:,i)=newp;
end

robt=[robt;robot];
end

B. Matlab/Simulink Simulation Snapshots

Screen snapshots of the following Mathworks' Matlab 4.0 and Simulink 2.0 simulations are given here:

- Matlab program for formation of a sphere.
- Matlab program for formation of a paraboloid.
- Modified VDP Oscillator block.
- Simulation window for coupled Modified VDP Oscillators.
- Matlab program for multigoal distribution using *switching distance*.
- Matlab program for multigoal distribution using *family values*.
Figure B.1 Plot screen from program Sphere.
Figure B.3 Unmasked VDP oscillator block with time varying feedback gain
Figure B.6 Screen snapshot of the Matlab program formxyld
Bibliography


Vita

Cem Ünsal was born on November 4, 1967 in Ankara, Türkiye. He graduated from Galatasaray Lisesi (Lycée de Galatasaray) in Istanbul (1986) and then received his B.S. degree in Electrical and Electronics Engineering with honors from Bogazici University, Istanbul, Türkiye. He then began his graduate studies at Virginia Polytechnic Institute and State University and received a M.S. degree in Electrical Engineering in June 1993. He will start his studies toward a Ph.D. degree in Electrical Engineering and a M.S. degree in Mathematics at VaTech in Fall 1993.