Course of Study: Computer Science

Examiner: Prof. Dr. rer. nat. habil. Paul Levi

Supervisor: Dr. rer. nat. Viktor Avrutin

Diplomarbeit Nº 2402

Modeling and Investigation of Robot Swarms

Heiko Hamann

Commenced: 01.09.2005

Completed: 23.02.2006

CR-Classification: I.2.9, I.2.10, I.2.11, I.6.0

University of Stuttgart
Institute of Parallel and
Distributed Systems
Universitätsstraße 38
D–70565 Stuttgart

Contents

Chapter 1

Introduction

"What a depressingly stupid machine,"
said Marvin and trudged away.

DOUGLAS ADAMS (1952-2001)
The Restaurant at the End of the Universe
(Chapter 6, p. 41)

1.1 Basic Ideas

1.1.1 Small World Robots

In traditional robotics the magnitudes of the robots are typically in the sizes of garbage cans, adult humans, or even bigger. This field is called macro-robotics and supplies robots with the computational power of at least one personal computer, high-capacity batteries, many sensors, and elaborated actuators. In short, these robots are characterized by complexity. They have complex software running on their machines and might even be learning agents. However, in case of a total break-down there is no recovery although they might have some kind of fault tolerant mechanisms or might compensate the loss of single components.
The idea is to see swarms in nature as a metaphor for robotics. A swarm of ants, termites, or honey bees survives easily the loss of some individuals. So we identify a high number of individuals as one precondition for an artificial swarm. However, to make the production of such swarms practically and financially feasible our artificial swarms will have to be much simpler and smaller than natural swarms. Additionally they will be completely or a big majority of them will be homogeneous. Only by restricting ourselves

to these properties we are able to achieve our aim by mass production. Our working hypothesis for this journey is that such artificial swarms of robots are useful for micro-manipulation.
In micro- or small world robotics we are working with robots that have very restricted capabilities. For a single swarm robot Marvin's statement in the quotation at the beginning of this chapter is quite true: It is a "depressingly stupid machine". But the swarm as a whole shows intelligent behavior.

1.1.2 Self-Organization

The search for an applicable way to control the artificial swarm is marked by the bounded abilities of the swarm robots. Due to the restricted capabilities especially in respect to communication a centralized control is not applicable. A distributed approach, which could for example be negotiation-based, is not applicable either due to the restricted capabilities in concern of memory and computational power. Our suggested solution is therefore a swarm that shows self-organized behavior. This approach is firmed by the mathematical theory of self-organization [Hak83b, Hak83a, NP77] because an artificial swarm has the typical properties of systems that show self-organized behavior:

- It consists of a large number...
- ...of simple sub-systems...
- ...connected with each other via short-range interactions.

Anticipated we can put it that way: The degree of order in our system could be defined by the equality of the robots' motion (both direction and velocity). Using this definition we have to be careful how the state of no motion needs to be defined. Additionally only scenarios without a successful state of no motion should be considered. We are adding energy to the system by the light source. The light is transfered into kinetic energy that enables the robots to influence the direction of their motion. By forming groups and moving in formations they increase the level of order.

1.1.3 Artificial Physics

Since we are simulating the I-SWARM scenario completely nothing is real – not the robots themselves, not the accelerations, and not the communication. However, we want to distinguish between the real world, like the robots that we are simulating, and totally virtual concepts that will exist later in the

implementation still only in software. Besides the simulated real physics like the acceleration, the velocity, and the communication we will introduce virtual forces later that are a useful model to control the robots but not real. Why could it be helpful to define virtual force laws? At first it might look like we are making things more complicated then they already are. Controlling a huge swarm of robots by local rules only is not a trivial task. But we define by these artificial forces a model that serves as both a guide that provides us with intuitive ways of thinking about our robot control and as a framework for the implementation of the control software.
How artificial physics can be used for a distributed control of agents was introduced by Spears and Gordon [SG99]. By defining artificial forces we are motivated by natural physical forces but not bound to them. The agents act as if the forces were real but of course they are only acting according to the defined model. Therefore the robots need to be equipped by sensors that are sufficient to perceive all facts of the real world that are necessary to compute the artificial forces.
By using the successful concepts of nature as a metaphor we hope to overcome the complexity of our task since in nature large groups of small entities show complex behavior by interacting according to simple and local rules. Obviously the concept of artificial physics is completely compatible to the concept of self-organization.

1.2 Previous Results

This work builds mainly up on a student project by Andreas Thomas Koch [Koc05]. Both his theoretical results and his implementation were reused here. In this section we want to introduce those results shortly.

1.2.1 Two Layer Approach

In order to develop control software that is easy to use and to maintain suitable software architecture is essential. Since we are stiffly restricted by the capabilities of the swarm robots a minimalistic solution is needed. Our approach is an architecture with only two layers named *decision layer* and *operational layer* (see figure 1.1). While on the decision layer we switch between different, more abstract modes of operation controlled by an internal state and perceptions, on the operational layer a set of rules is defined that controls directly the robots' basic actions based on the actual mode. The decision layer is modeled by hybrid automata and the operational layer by virtual force functions (see the following sections).

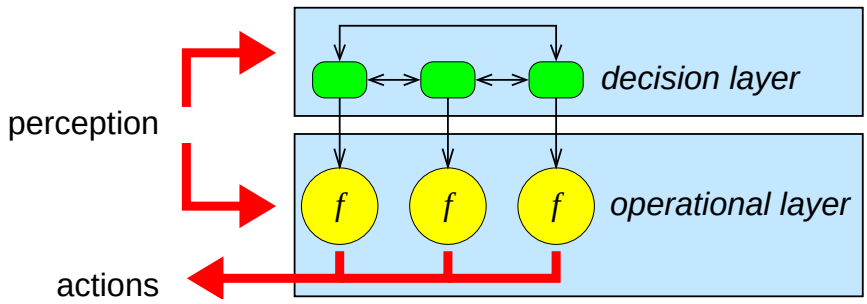


Fig. 1.1: Two layer architecture with the decision and the operational layer [Koc05], changed.

The operational layer is influenced by the decisional layer and governs the basic actions of the robots. The decision layer is influenced by the perceptions of the robot such as perceiving a special object or communication with another robot.

1.2.2 Hybrid Automata

The two layer software architecture can be modeled by a unified framework with a well elaborated mathematical background: hybrid automata (see [Avr04] and references therein).

A hybrid automaton is a mathematical model of a hybrid dynamical system, given by the set

$$H = (\vec{x}, (V, E), S, Jump) \tag{1.1}$$

with the following components:

1. a state vector $\vec{x} \in \Gamma \subset \mathbb{R}^n$.
2. a set of control modes V and a set of transitions E that together define a directed graph (V, E).
3. a set of functions (behavioral rules) R.
4. a set of transition conditions $Jump$.

Here a single robot is described by a hybrid automaton. Thus the model for the complete swarm is given by a set of coupled automata. The different modes of operation are given by the set V. For every mode a behavioral rule $\vec{f} \in R$ is specified. These functions manipulate the vector $\vec{x}$ and represent the operational layer of the model. The control modes are switched via

transitions $e \in E$. For each transition e a condition is specified in the set $Jump$. Thus the decision layer of the model is given by the graph (V, E) combined with the set of conditions $Jump$.

Hybrid automata have an important advantage with respect to software development because they support the method of incremental development. To increase the complexity of a given scenario an already existent hybrid automaton is extended by new modes of operation and new transitions. In this case large parts of the automaton persist and therefore time and costs in the development are reduced due to the reusability of the software.

1.2.3 Virtual Forces

The most challenging task in the development of control software for robot swarms is inventing sophisticated strategies of sensor- and communication-induced behavior based on the information gathered from the environment. Examples of such behavior patterns are: avoid collisions, stay close to another robot, move in some defined direction, and so on. Since the electronics module of the robots has strictly limited capacities it is infeasible to come up with a complex algorithm that deals with every single pattern. Therefore a unified approach is needed. All the different movements like moving back from an obstacle or moving towards another robot should be controlled by one generic algorithm governed by parameters. Here we solve the problem using a generic force function $F(d)$ depending on the measured distance d to the object.
$F(d)$ is defined on the interval $[0, \infty)$ but the functional interesting range is $[d_{min}, d_{max}]$, where the minimal distance d_{min} is the robot size and the maximal distance d_{max} is the maximal sensor range. The function needs a repelling component $F(d) < 0$ at least for small distances $d = d_{min} + \epsilon$, with $\epsilon > 0$ to avoid collisions. We would like to have at least in some cases also an attracting component $F(d) > 0$ to keep robots together to cooperate. If such a function has both an attracting and a repelling component then in between at some point d_{opt} the slope has to be zero to keep the robot close to the equilibrium point and reduce undesirable oscillations.
We define the function depending on five parameters as the following:

$$F(d) = \begin{cases} -F_{rep} & \text{if } d < d_{min} \\ -F_{rep}\frac{(d-d_{opt})^2}{(d_{min}-d_{opt})^2} & \text{if } d_{min} < d \leq d_{opt} \\ F_{att}\frac{(d-d_{opt})^2}{(d_{max}-d_{opt})^2} & \text{if } d_{opt} < d < d_{max} \\ 0 & \text{if } d > d_{max} \end{cases},$$

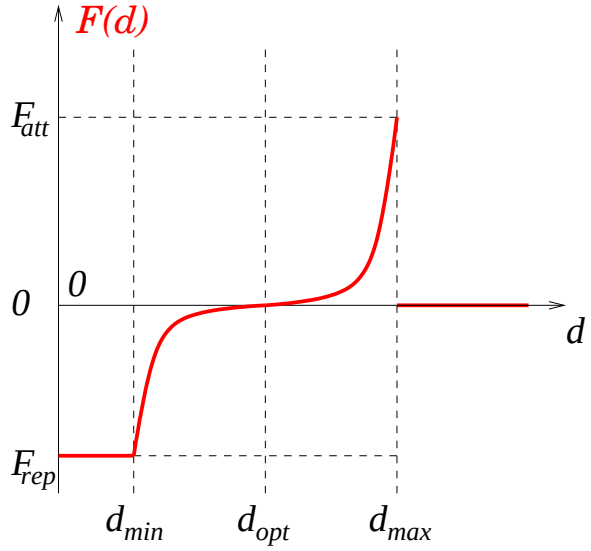


Fig. 1.2: Generic force function [Koc05], changed.

By the class of the observed object a generic function is defined that returns the force $F(d)$ virtually caused by this object depending on the measured distance d. A negative force has a repelling and a positive force an attracting effect on the robot. Thus using this function here the robot would stop in a position with a distance of d_{opt} to the object.

where F_{rep} denotes the maximal repelling force, F_{att} the maximal attracting force, d_{min} is, as already said above, the robot size, d_{max} the sensor range, d_{opt} the optimal or equilibrium point, which is the root of the derivation (see figure 1.2 for a graph). Values in the range of $d < d_{min}$ should not occur in the reality because they represent a collision but are defined for the simulation. In reality a value of $d > d_{max}$ cannot be measured because an object out of the sensor range cannot be perceived but for the simulation this value is defined because these physical bounds are not intrinsic in the simulation.
For each perceived object the value of the corresponding virtual force is computed. The parameters of the function $F(d)$ are defined by the class of the object and the state of the robot.

1.2.4 Simple Patterns in Robot Swarms

The simplest action a swarm robot can execute besides staying stopped is to move to one random direction until an obstacle blocks the way and it changes its direction according to some defined procedure. This can be

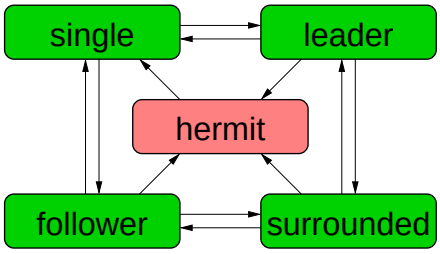


Fig. 1.3: Hybrid automaton for chain formation [Koc05].

A chain consists of one robot in LEADER mode, one in FOLLOWER mode and optionally some in the mode SURROUNDED. In the chain formation scenario the mode HERMIT serves as a fallback mode that is activated if problems occur. Then a robot switches to mode HERMIT for some cycles and will not join to any other robot for a short period.

called random walk and is what the robots in our model do in the initial mode. However, in order to accomplish demanding tasks with a swarm cooperation is needed. The first precondition for cooperation in a swarm is that at least two robots are within the sensor range of each other to make communication possible. Thus we want at least groups of robots to be close to each other. To ensure that such groups stay mobile and working the position of each robot within the group should be exactly defined to form an ordered formation rather than a cluster in that robots hinder each other. Although it is actually exactly the swarms that show us how it is possible to move efficiently in a crowd it turned out that to control robots a more goal-oriented behavior and motion is needed.

A very simple formation is the chain that is one robot leads and some follow. Some previous papers have shown that the chain formation is both simple and very efficient (see [TND05] and [ND04]). As introduced by Koch in [Koc05] a hybrid automaton for the chain formation can be seen in figure 1.3.

A more complex behavior is modeled in figure 1.4. Here the robots connect to form chains; if a chain has the nominal length then each member executes a transition to the next level (blue) representing complete chains; if the leader of such a complete chain perceives an object then each member of the chain performs a transition to the next level (pink) representing the object orbiting meta-mode. Although the structure of a complete chain and an orbiting chain is the same, an extra mode *LEADER*, for example, for the leader of an orbiting chain is needed to model different virtual force

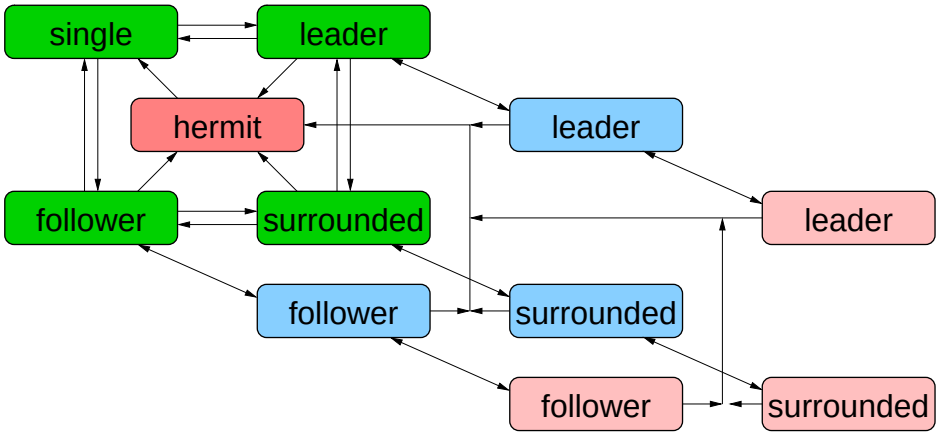


Fig. 1.4: Hybrid automaton for object orbiting [Koc05].

This automaton can be interpreted in a hierarchical way. We identify three levels: incomplete chains (green and mode HERMIT), chains of the full length (blue), and orbiting chains (pink).

functions. See figure 1.5 for a typical scene of the cooperative search for obstacles and the cooperative object orbiting.

1.3 Objectives

This work is done in the context of the European research project "I-SWARM" [IS06]. The main goal of this project is the production and control of hundreds to thousands of micro-robots that are able to accomplish tasks that could not be accomplished by a single nor by a small group of such robots. Attention has to be paid here to the intended size of the robots (about 2 times 2 times 3 mm^3) that is a novel dimension in modern robotics. Since the hardware of the planned robots is currently at development stage the abilities of the robot swarms need to be investigated using qualified simulation models.

The results of the preparatory work as summarized in section 1.2 are expedient. The model that simulates the ideal hardware and the complexity of the scenarios are both expandable while the analysis part is due to the restrictions in time of this preceding work quite compact.

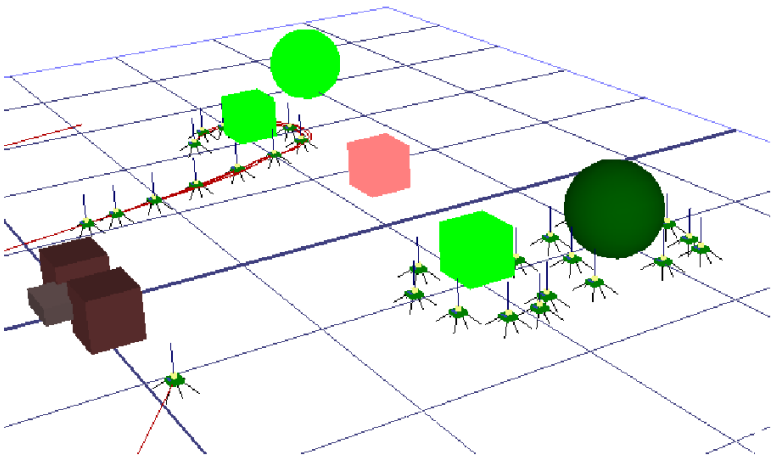


Fig. 1.5: Typical scene of the cooperative object search and orbiting.

The primary objectives of this thesis are:

- Development of more realistic models for the intended robots concerning the locomotion as well as the sensors of the robots.
- Design and implementation of a complex test scenario that is capable to show the abilities of the robot swarm.
- Investigation of the quantitative dependencies between the abilities of single robots and the behavior of the whole robot swarm. This should be done by detecting critical parameters in the robots' hardware as well as the determination of critical values for these parameters.

The implementation of the software that needs to be developed is to be done within the framework of the simulation software package AnT. In the implementation a special focus is to be set on expandability and reusability.

Part I

Hardware

Chapter 2

General Design, Powering, and Ego-Positioning

Science can amuse
and fascinate us all,
but it is engineering
that changes the world.

ISAAC ASIMOV (1920-1992)
Isaac Asimov's Book of Science
and Nature Quotations (p. 78)

The development of a robot of the intended magnitude is pioneering work and extremely difficult. Although no prototype is existing yet the hardware specification for the I-SWARM project is almost fully developed. Some facts that are of interest here are summarized below.

2.1 General Design

The size of the robots will be 3 times 3 mm^2 and they will be 3.5 mm thick (see figure 2.1). The programming will be done using the solar cells. The electronics module will have a 1 MHz clock, 8 kB ixprogram memory, 2 kB data memory and 256 Byte internal memory. The power consumption of the electronics module will be about 250 μW, the peak power consumption of the locomotion module will be about 100 μW and its average about 10 μW. Together with the sensors this is summing up to a power requirement of about 500 μW.The robot will also have a tool. That will be a vibrating needle that is integrated in the locomotion module (see figure 2.2).

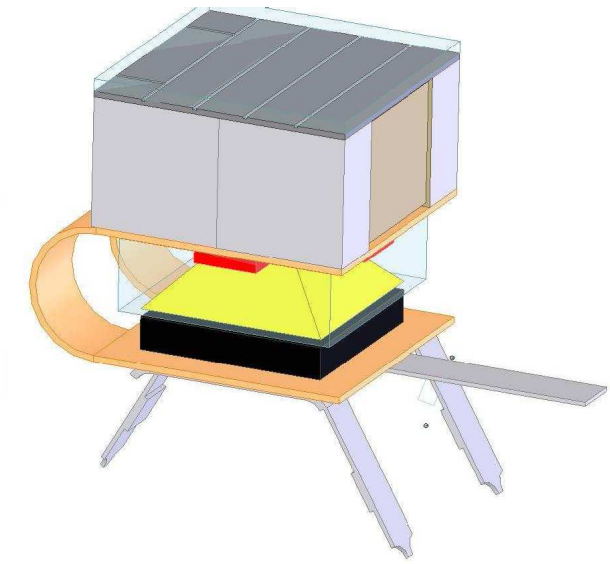


Fig. 2.1: Robot design [ISW05], by courtesy of the I-SWARM project coordinator.

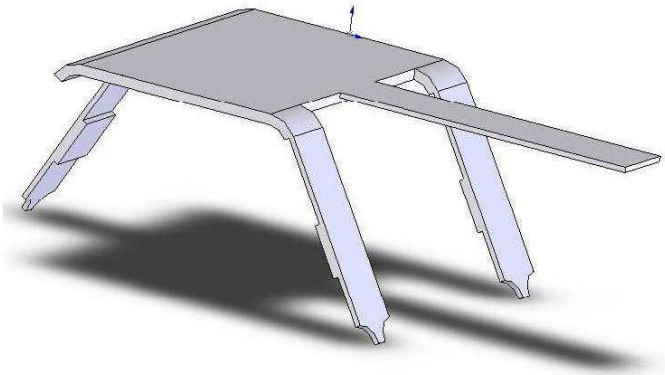


Fig. 2.2: Three legged locomotion module with vibrating needle tool [ISW05], by courtesy of the I-SWARM project coordinator.

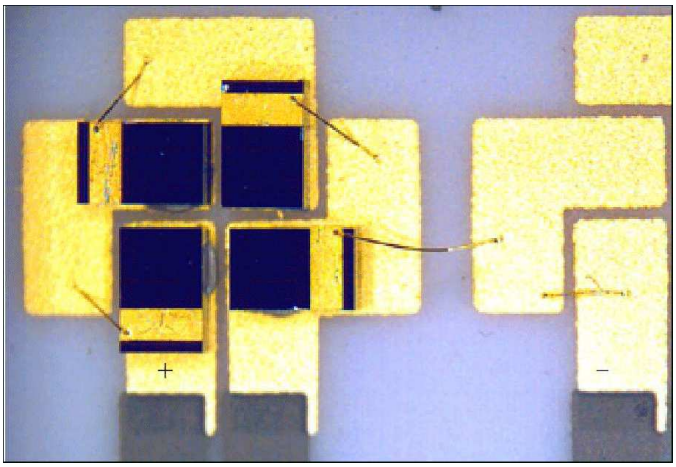


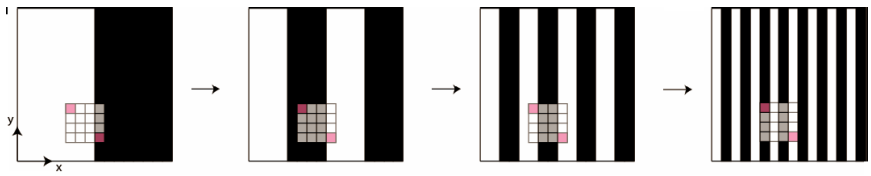
Fig. 2.3: Prototype of a solar cell module with $4mm^2$ active area [ISW05], by courtesy of the I-SWARM project coordinator.

2.2 Powering

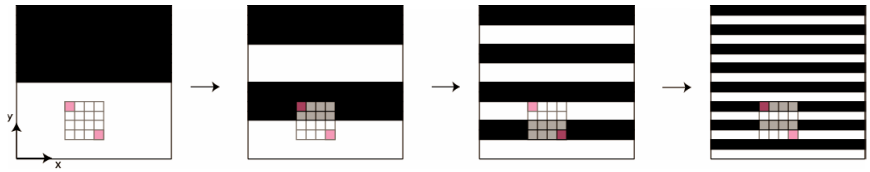
For the wireless power transfer solar cells will be used. The module will have a 4 mm^2 active area composed of four 1 x 1 mm^2 single cells (see figure 2.3). A beamer will serve as a virtual sun for the swarm. It will deliver 3000 ANSI Lumen that will result in a maximum of 121 $\mu W/mm^2$ on the area of a DIN A4 sheet of paper. But the irradiance will not be constant at every point of the arena and vary between 31 and 100 percent. The solar cell gives at least 250 μW and the use of an extra lamp might double this value.

2.3 Ego-Positioning

To perform an ego-positioning the beamer that is also used as the energy source for the wireless power transfer and a multi-segment solar cell on the back of the robot are needed. All robots have to stop during the procedure. Then a sequence of images is projected by the beamer onto the arena (see figure 2.4). These images are a series of alternating white and black lines. The number of these lines is doubled at each image until the minimum width is reached. This minimum width is ideally the size of the solar cell segment of the robot. The two lines of the first image correspond to the most significant bit of the robot's position and the lines of the last image to the



(a) vertical grids



(b) horizontal grids

Fig. 2.4: Sequence of vertical and horizontal grids projected on the arena [ISW05], by courtesy of the I-SWARM project coordinator.

The pictures show a sequence of grids that could be projected onto the arena using the beamer that is also used as energy source. Time goes from left to right. The robot is shown by the small grid symbolizing the solar cell on the robot's back.

least significant bit. The robots store their measurements of the brightness (bright or dark). From this data they can address their positions and maybe even their angles if a sufficient precision can be reached.
Although the possibility exists ego-positioning is not used in this work because it is contradictory to the philosophy of self-organization. Using Ego-positioning means using global information but for self-organized behavior only local information should be retrieved and all actions should be governed by short-range interactions only.

Chapter 3

Locomotion, Sensors and Communication

The fact that we can describe
the motions of the world
using Newtonian mechanics
tells us nothing about the world.
The fact that we do,
does tell us something
about the world.

LUDWIG WITTGENSTEIN (1889-1951)
John D. Barrow
The World within the World

3.1 Locomotion

3.1.1 Basic Concept

The robots are designed to have a polymer drive. The principle of this kind of drive is to have long and thin legs that are driven in resonance. This makes the tip of the legs move in elliptical trajectories. One can distinguish two states of function: In one state the legs' tips have contact to the ground which results in a horizontal and upward impetus. After that, in the other state, the legs' tips move back without contact to the ground. This results in a movement of the robot.
It is intended for the locomotion module to have one pair of legs at the front and only one at the back (see figure 3.1 and 2.2). The preferential direction

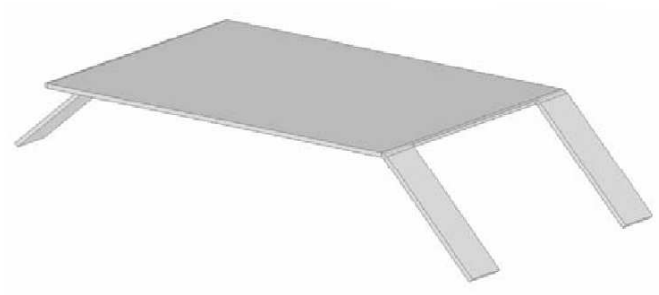


Fig. 3.1: The robot's polymer drive [ISW05], by courtesy of the I-SWARM project coordinator.

is defined by the pair of legs while the single leg on the other side is only used for backward movement. Turns can be performed by activating only one leg of the pair.
The theoretical maximal velocity of the intended motion module is as high as 60 mm/s. To save energy and to have a considerable well control it is advisable to use some kind of intermittent motion. See figure 3.2 for examples of possible movements.

3.1.2 Modeling

Compared to the ideal locomotion that is able to accelerate in all directions the main restriction of the designed locomotion is that the acceleration can only be performed in one direction – the direction that is defined by the position of the pair of legs.

The simulation of the drive is done in two steps per iteration: At first an ideal velocity vector is computed. That is the velocity vector that would be the perfect reaction to the virtual forces taking effect to the robot but in general it would only be possible to be realized with an ideal drive. In order to compute it both are needed the current velocity and the ideal acceleration which is the sum of all forces that take effect.
In the second step this ideal velocity is checked if it can be realized (see figure 3.3). The restricting value is the angle α between the current heading which is the current velocity vector and the ideal velocity (see figure 3.4). If this angle is bigger than the maximum turning angle the ideal velocity and the underlying ideal acceleration cannot be realized. Since for the turning

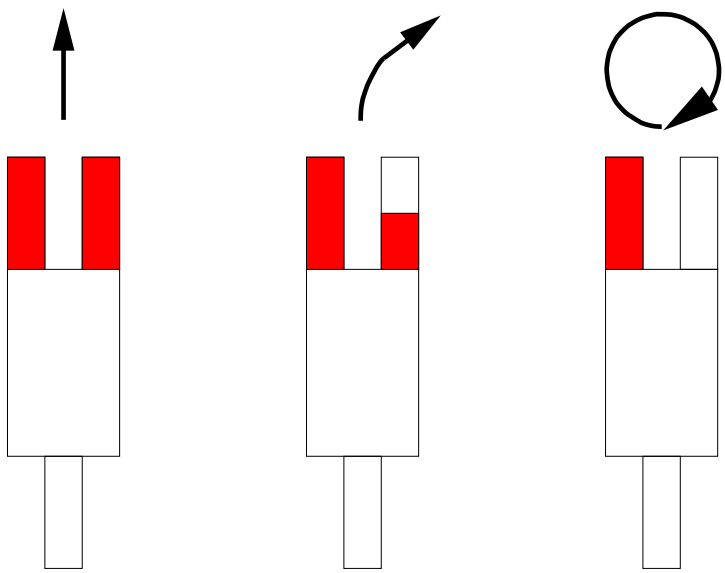


Fig. 3.2: Examples of possible movements.

Left: Full forward acceleration. Middle: An example of an intermittent motion that could also be seen as a little turn with forward acceleration afterwards. Right: Full acceleration on one leg for a turn.

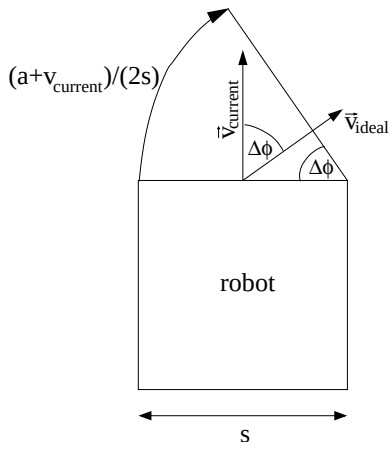


Fig. 3.3: Schematic sketch of the turning robot.

only one leg can be used the maximum angle ϕ_{max} in radians is defined by the half of the current velocity $v_{current}$, the half of the maximum acceleration a_{max}, and the robot's size s:

$$\phi_{max} = \frac{1}{2}\frac{(a_{max} + v_{current})}{s}.$$

In case the robot cannot turn that fast it is immediately stopped and turned as far as it is possible within a single time step in order to minimize the angle between the resulting heading and the ideal velocity vector. This is done without any gain of distance.
If the angle between the heading and the ideal velocity can be performed in one time step it is still needed to be checked whether the power demanded by the ideal acceleration can be delivered. The maximum forward acceleration a_{real} after having turned is defined by the turning angle $\Delta\phi$ and the maximum acceleration a_{max}:

$$a_{real} = \frac{\phi_{max} - \Delta\phi}{\phi_{max}} \cdot a_{max}.$$

If the ideal acceleration is bigger than a_{real} the acceleration is set to the remaining amount of acceleration a_{real}.

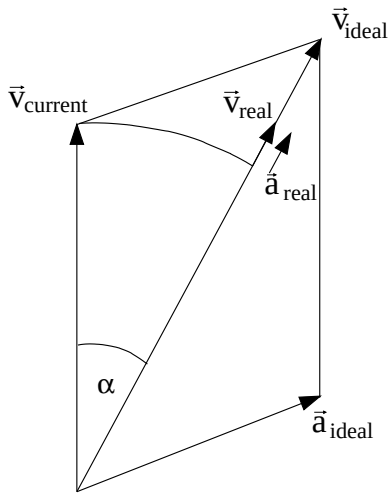


Fig. 3.4: Example of how the velocity of the next step is computed.

The new velocity $\vec{v}_{real}$ is computed using the velocity and the acceleration of an ideal locomotion; here $\alpha < \phi_{max}$, i.e. the robot is able to turn to the new heading and accelerate to this direction within one time step.

3.2 Sensing and Communication

3.2.1 Basic Concept

The sensing in the I-SWARM scenario can be divided into two tasks:

1. Proximity sensing: The sensors should provide measured distances to objects in the neighborhood of the robots. These sensors are placed around the robot so that in general it does not need to turn or perform other activities to measure.

2. Perception: The robots should be able to recognize objects and classify them for example as obstacles, other robots, or mission important objects.

Communication is actually distinctly different task compared to sensing. However, it is intended in the I-SWARM project to use optical sensors that will also be used for communication. Thus we can discuss all three tasks in one.
The robots will have four sensors that will cover the neighborhood of the robot only partially since every sensor covers only 60 degrees. The problem

of proximity sensing are the so-called indiscernible distances. The sensor cannot differentiate whether the object is on the central line and in a large distance, or the real distance is smaller and the object is displaced from the central line. This problem remains open and might be solved by undertaking several measurements in different directions.
The IR-based perception consists of sending an IR radiation beam and receiving the reflected light. For better performance a small opening angle and a "high energy" beam are considerable. Distance measurement can be done with low power for object recognition the robot should switch on the higher power IR-emitter and scan the object.

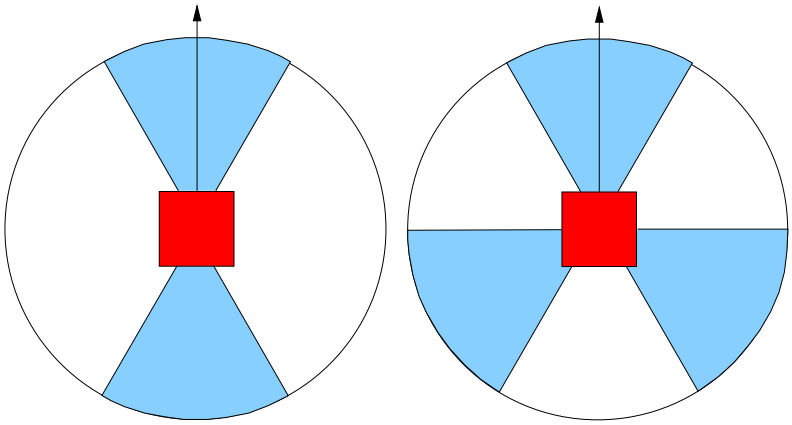
3.2.2 Modeling

The fact that the tasks of communication and sensing are united in one device is considered in the model by working with the same radius for both. While the proximity sensing is modeled comparatively close to the reality the model for communication is quite abstract and without any mentionable bounds except the range. The perception feature is only modeled indirectly: Objects that do not answer are considered to be obstacles and answering objects are obviously robots.

The number of sensors can be specified arbitrarily by the parameter S. The arrangement, however, is fixed and defined by the number: One sensor covers always the area directly in front of the robot (in the preferential moving direction) and all the others are arranged in a way that results in an equal distribution of the uncovered area (see figure 3.5 for some examples). The actual geometry of a typical sensor is approximated by circular sectors as seen in figure 3.6. Also the ratio of the covered area to the uncovered can be specified in a parameter in the interval from zero (a blind robot) to one (a robot with full circumferential visibility). In the following this parameter will be called *visibility ratio* v.

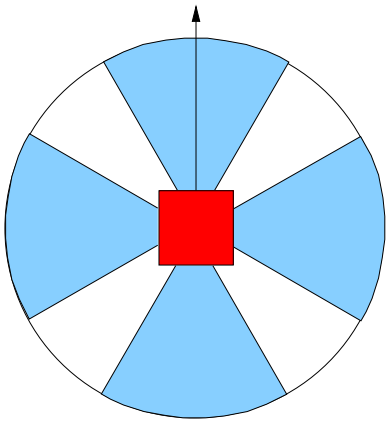
Only if an object is within the range and in the covered area the sensors deliver data about it. That will be the bearing of the object and the distance. Since a single sensor of the kind used here is not able to measure angles it reports only the direction of its own. For example the front sensor always reports bearings of 0° for all the objects in its range. Thus stated in a more abstract way a sensor that covers an angle of

$$\gamma = \frac{360^o \cdot v}{S}$$

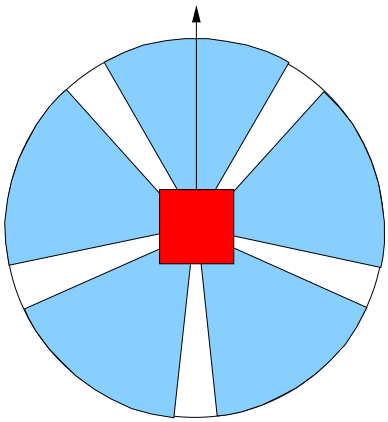


(a) 2 sensors

(b) 3 sensors



(c) 4 sensors



(d) 5 sensors

Fig. 3.5: Sensor Arrangement for 2,3,4, and 5 sensors.

By the red square the robot is symbolized and the arrow shows its preferential moving direction. For an uneven number of sensors it might be a drawback that there is no view to the back.

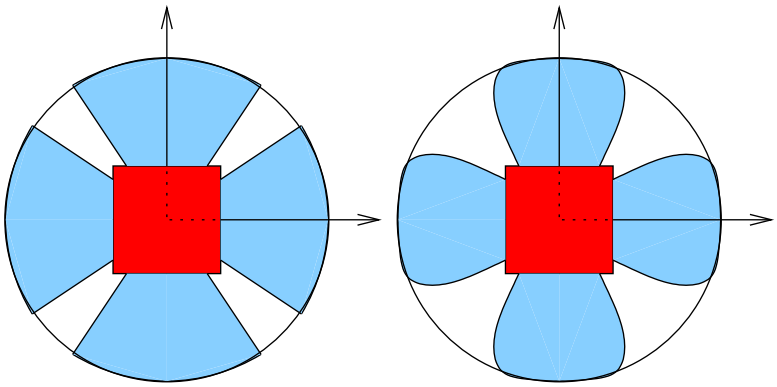


Fig. 3.6: Circular sectors (left) as an approximation of our model to the actual area covered by a sensor (right).

delivers only angles in a gradation of

$$\Delta\phi = \frac{\gamma}{2} = \frac{180^o \cdot v}{S}.$$

This will be called the *angle uncertainty* in the following.

Only a very simple model of noise has been implemented. Noise can be applied to both the actual angle information and the actual distances. That means that in the simulation at first the data of the actual situation is computed, then noise is added to the angle information and based on this noisy data the model decides which sensor notifies an object depending on the noisy bearing. After that noise is added to the distance. The error is Gaussian and is controlled by the deviation parameter σ. There is an extra distribution and a parameter for both the distances and the angles. The samples are drawn from a finite interval from -3σ to 3σ only. Results of this noise are:

1. The distance is noisy but the sensor always notifies a distance if the object is within the coverage of the sensor.

2. The noise to the angle information might be too small to move the object out of the range of the sensor. Thus the notification of the sensor would not change at all.

3. The noise on the angle might be big enough that the object is out of the covered area and no sensor delivers any data about this object. This is interpreted as a sensor malfunction.

Since the noise in one cycle is not correlated with the noise in the next cycle our experiments showed that this model of noise has almost no effects to the performance in any scenario. Because of the high frequency of measurements (at each cycle) the noise is simply averaged and thus has almost no effect.
For the model of communication it is assumed that sending and receiving works perfectly within the coverage and that at least a single byte can be sent.

Part II

Base-Target-Scenario

Chapter 4

Overview

This quest may be attempted by the weak
with as much hope as the strong.
Yet such is oft the course of deeds
that move the wheels of the world:
small hands do them because they must,
while the eyes of the great are elsewhere.

Elrond
JOHN RONALD REUEL TOLKIEN (1892-1973)
The Fellowship of the Ring (p. 353)

4.1 Motivation

A characteristic property of the small sized robots, considered here, is the strictly bounded range of their perception. To let them operate and cooperate successfully in a relatively big area, it will often be necessary to establish some kind of reliable "long distance" communication. Whereas by long distance we consider distances that might be low values of centimeters but they are multiples of the robots' reach. Since our motives are localization and self-organization, we want to omit the use of global positioning. Thus the option of sending a single robot to a given absolute point as messenger ceases to exist, since storing and communicating information like directions or distances is not reliable, if actually feasible. This makes it inevitable to come up with some other scheme of more intense cooperation. As a consequence thereof we will need many robots for this task.
If the only allowed communication media is the robot itself and thus other possibilities, like for example artificial pheromones, are not taken

into account, the obvious solution is of course the use of chains of robots. These chains allow other robots to find a marked object or make the communication between two arbitrary points possible.
The theme of the scenario discussed here is the overcoming of spatial distances, as well as the exploration of space, searching and marking of objects with the strictly bounded sensor ranges of the robots. The presented techniques might be useful for application in situations that are characterized by limited view and/or limited communication conditions like operations in deep water, small pipes, debris, or even the human body.

4.2 Outline

The whole scenario takes place within a quadratic arena. In this arena there are at least two objects besides the robots. One of these objects is called *base* the others are the *target objects*. At the beginning the robots are uniformly distributed over the whole arena. As a variant they are initially concentrated in one place. The task consists of finding every object, encircling them, and connecting them by lines of robots that should be as short as possible. This problem is related to the Steiner tree problem [FHW92, Hau04, PS02].
The method that is used here to solve this problem can be broken down into three phases:

1. Exploration: Build chains, find the objects and encircle them.

2. Formation of trees: Position one chain at the circle around the object and begin to build a tree out of chains using the open end of this first chain as the root.

3. Reduction of the tree: After all objects have been connected to every other object, i.e. the graph consists of only one component, unneeded chains should leave the tree. Additionally the remaining lines of robots should be reduced to the minimal necessary number of robots needed to connect the objects.

Although these three phases are not strictly separated chronologically their beginnings are chronologically ordered but the phases might overlap. For example the formation of a first tree might already begin at one object while another one has not been found yet.

4.2.1 Exploration

At the beginning the robots form chains of a defined length. This length is hard-coded into the rule set of the robot. It should not be too short since that would make the formation of the trees inefficient. The probability that a chain encounters a tree would be high but the tree would grow quite slowly per added chain. However, it should not be too long as well since that would bring down the probability of a chain encountering a tree although the tree size would grow a lot per added chain.
These chains of a given length are dispersing all over the arena. Also incomplete chains and single robots that have not met enough or any other robots yet depart from the starting position heading to a random direction. If a complete chain perceives an object, it starts to encircle it. After a given time it stops circling and serves now as a sentinel that allows the docking of only one other chain.

4.2.2 Formation of Trees

As soon as a complete chain meets a robot of a circle around an object that is not connected to another chain yet the formation of trees begins. The circle and the chain connect, the circle does not accept the docking of any other chain any more and the open end of the chain serves as the root of a new tree. This end accepts up to three other chains (see chapter 6 for the definition of x- and y-trees) and the same is true for their open ends. In order to maximize the reach of the tree two chains always try to maximize the angle between them. This should lead to angles of 180° if at that position only two chains are connected, 120° if there are three chains and 90° if there are four.
Since the robots should form a tree it is not allowed for two chain ends of the same tree to connect. But it is allowed for two chains that are parts of two different trees. If that happens two different objects have been connected successfully.

4.2.3 Reduction of the Tree

At the moment when all objects are connect to all other objects the tree will be reduced. This is done by two different operations: All robots that are open ends push off and leave the tree. This starts a chain reaction of many robots leaving. The purpose of the other operation is to tighten the

remaining connections between the objects. This is done by letting robots push off that are directly connected to the circles around the objects and that were able to make sure that their neighbors are close enough to the circle to ensure that the connection will not break (see figure 5.7). When the reduction is finished successfully all objects are connected by a minimum number of robots.

Chapter 5

Implementation

Beware of bugs in the above code;
I have only proved it correct,
not tried it.

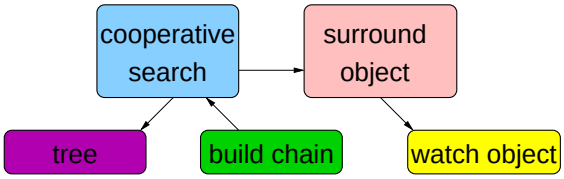
DONALD ERVIN KNUTH (1938-)
Notes on the van Emde Boas
construction of priority deques (p. 5)

In the following the three different phases of the scenario will be discussed in detail. Figure 5.1 shows an overview and the complete hybrid automaton of the base-target-scenario.

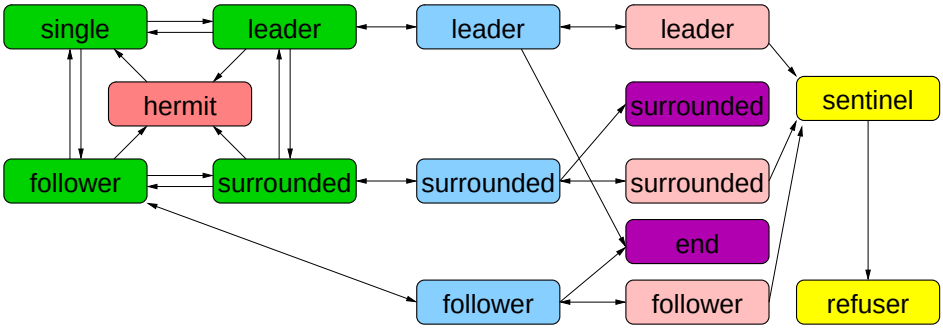
5.1 Exploration Phase

Almost only the basic features like random walk and the formation of chains are used in the exploration phase. Only one issue had to be treated to speed up the whole process. The individual robot has no internal motivation to accelerate to a high velocity in the original definition of the virtual force laws. By interacting with other robots only he is accelerated.

In the following it will turn out that it is important for the overall performance how the force function of the robots in the state *LEADER* in a chain is defined. We will discuss two variants that are called *symmetric* and *asymmetric* force functions. The symmetric force is the more intuitive one: The neighbor of the chain leader (always a robot in state *SURROUNDED*) has an effect on the leader by attracting and repelling it depending on its distance. This is intuitive because this assures that the chain stays



(a) Overview



(b) Complete automaton

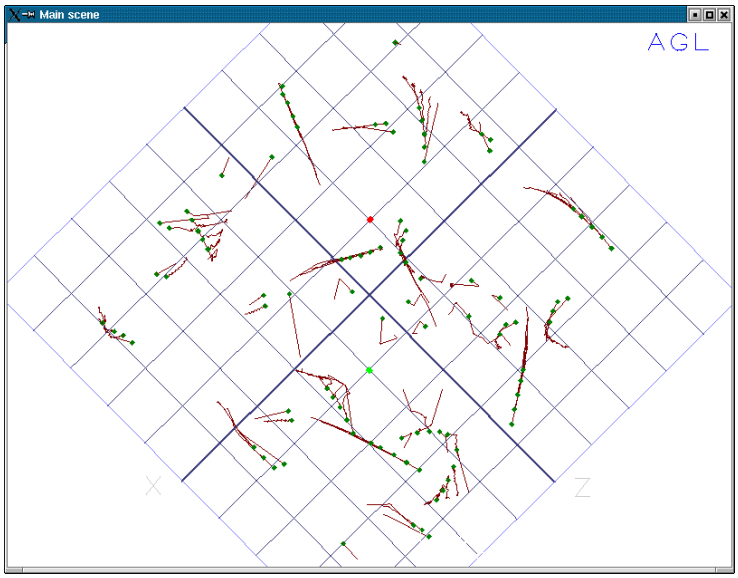
Fig. 5.1: An overview and the complete hybrid automaton of the base-target-scenario [Koc05], changed.

close together with equidistant gaps between the robots and it works as a collective although the leader leads.
The asymmetric force function omits the influence of the neighbor to the leader totally. The result is a completely autonomous leader that has just an appendage following it. The gaps are not equidistant, which increases the risk a bit that the chain breaks apart.

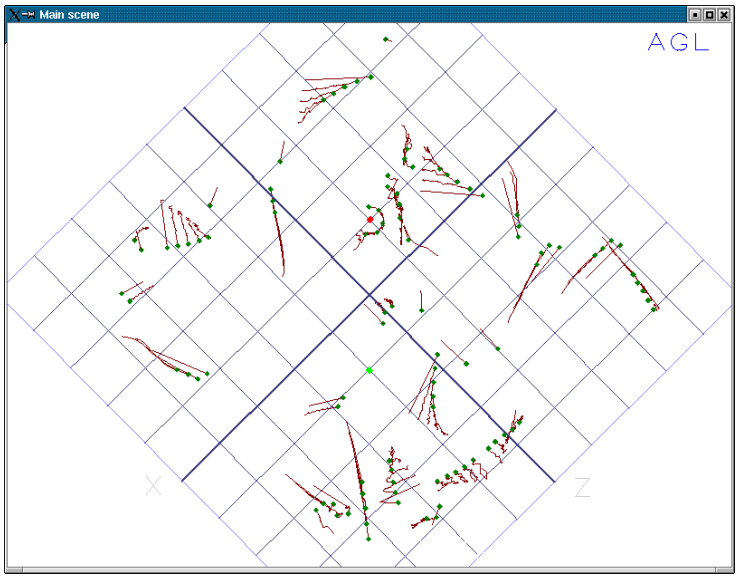
In the following we will consider both force sets and their effects: Since the objects are typically situated closer to the center of the arena than to the borders, it is essential that the probability of a chain being in the center is higher than the probability of being close to a wall. However, the chains determined by the symmetric forces tend to stay close to the walls. This is caused by the process that takes place when such a chain hits the border. The bigger the angle of entrance is, the closer the leading robot will be reflected into his direct neighbor in the chain. But using the symmetric forces the leading robot will be decelerated and accelerated again in a direction that is similar to the original angle of entrance. After oscillating between the wall and its neighbor, the robot will travel in a direction that is almost parallel to the wall.
We overcome this problem by using the asymmetric forces. But this means again that the leading robots get decelerated to slow velocities from time to time because their neighbors have no repelling effects that would accelerate them in the direction of traveling. By providing the robots with a timer that counts the cycles they were not exposed to any force, we are able to let the robots accelerate to the nominal velocity after a time of unaffected trajectory. This mechanism makes sure that chains do not interfere with each other too much and lets them move fast when they are unimpeded. It is like cars that have to drive slowly on urban streets and are allowed to drive faster on rural ones. Thus one could say that we implemented the lead-foot for our robots here. See figure 5.2 for two typical situations in the exploration phase.

5.2 Tree Formation Phase

This is the main phase of this scenario and thus the main part of the implementation. We want to make sure that only one tree is built at each object, new states for the robots in the tree are needed, the area covered by the tree should be maximized, and we need to guarantee that the tree stays loop-free.



(a) Many incomplete, some complete chains; the objects have not been found yet.



(b) The target object (red) has been found and is encircled.

Fig. 5.2: Screen-shots of the simulation showing the exploration phase.

5.2.1 Allowing Only One Chain per Object

By now the objects are encircled by chains of robots and if another chain meets such a circle, it should connect to it. This should only be possible for the first chain. All other chains that arrive later are not allowed to connect directly to the circle. The simple solution to this is done by introducing two new states: one for the robots of the circle that is free to everyone to connect (*SENTINEL*) and one for circles that have already connected to a chain (*REFUSER*). Thus the transitions of a chain that encircles an object are quite clear: At first they change from *LEADER*, *SURROUNDED*, or *FOLLOWER* to *OBLEADING*, *OBSURROUNDED*, and *OBFOLLOWER* respectively. After having stopped in a (semi-)circle around the object their state becomes *SENTINEL*. Robots in this state will never move again. Finally, after a chain has connected to one robot of the *SENTINELs* their new state is *REFUSER*. That is a state that has no transition to any other state and also robots in this state do still not move at all.

5.2.2 Additional States for the Tree

To realize the trees we introduce two new states: *TREEEND* and *TREESURROUNDED*. If a chain connects to a *SENTINEL*, its *SURROUNDEDs* become *TREESURROUNDEDs* and both the *LEADER* and the *FOLLOWER* become *TREEENDs*. Thus within the tree no direction is defined. A *TREEEND* in general can connect to up to three other *TREEENDs* (see chapter 6 for a definition of x- and y-trees). Only if it is connected to a *REFUSER*, any other connection is forbidden. Also for robots in the state *TREESURROUNDED* it is forbidden to connect to any robot except their two neighbors in the chain. See table 5.1 for an overview of the introduced states.

5.2.3 Maximizing the Area Covered by the Trees

Now we have defined the organization of the tree but almost more important is its geometry. The spatial extension of the tree determines its efficiency. The purpose of the tree is to connect to other trees and by that to other objects. In order to push the probability of one tree encountering another one as high as possible the amount of disjunct space that is covered by the sensors of robots in the state *TREEEND* should be as high as possible as well as the maximum distance between robots in the state *TREEEND* (see also chapter 6).
This is done by the definition of a new force that is effective to every robot

State	Description
SENTINEL	Robot in a circle around an object that is free for other chains to connect to. A robot in this state will never move in this scenario again.
REFUSER	A former robot in state SENTINEL in a circle around an object that is not free for other chains anymore because already one chain is connected to it. A robot in this state will never move in this scenario again.
TREEEND	A former leader or tail of a chain that is part of a tree now, connects to other robots in state TREEEND, and forms the joints of the tree.
TREESURROUNDED	Robot that was in state SURROUNDED before its chain became part of a tree.

Table 5.1: Overview of the introduced states.

in a tree (the definition of the distance between robots in the tree will be discussed at the end of this section). We will call it the angle maximizing force. The aim is that the angle between two neighboring chains of robots in the tree is maximized, i.e. for two connected chains we want therefore to have an angle of about 180° between them, for three we want 120° and for four we want 90°.
We present at first the algorithm to compute the angle maximizing force in two steps by the help of figure 5.3 and without considering any needs of distributed computation or for synchronization. As an additional simplification we make use of the global positions of the robots here but we will provide a way of computing the force by using local information only later on. Note that the robots A, C, and E must be in state *TREEEND* while robots B and D are in state *TREESURROUNDED*. Say that the position vector $\overrightarrow{w}$ of a robot X is given by $\overrightarrow{w} = \overrightarrow{pos}(X)$.

1. All connected robots R_i in state *TREEEND* for $0 \leq i \leq 4$ (in figure 5.3: A, C, and E) compute the direction to their neighbor N_i in the chain (here: the direction from A to B and from C to E):

$$\overrightarrow{d_i} = \frac{1}{||\overrightarrow{pos}(N_i) - \overrightarrow{pos}(R_i)||}(\overrightarrow{pos}(N_i) - \overrightarrow{pos}(R_i)).$$

2. Under the assumption that every R_i knows all the $\overrightarrow{d_i}$ they are able to

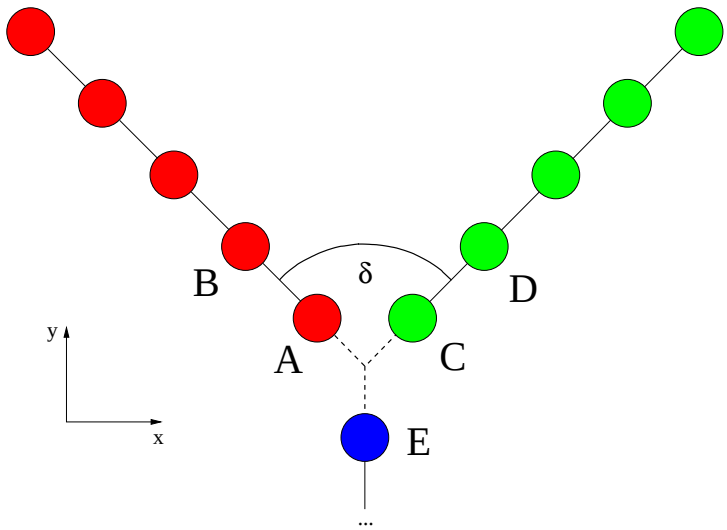


Fig. 5.3: Example for three chains in a tree.

Up to four connected chains in a tree are allowed. In order to cover an as big as possible area the angles δ between the chains need to be maximized. Having only three chains here we want to have $\delta \approx 120^o$.

compute the direction and the magnitude of the force by

$$\overrightarrow{F_i} = c(\overrightarrow{d_i} - \sum_{j=0, i \neq j}^{m} \overrightarrow{d_j}),$$

with m is the number of the connected robots R_i and c is some coefficient by which the intensity of the force can be fitted.

Before we investigate how the use of global information can be avoided and whether we need some kind of synchronization we want to consider the situation of figure 5.3 as an example. Say the positions are given by $\overrightarrow{pos}(A) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$, $\overrightarrow{pos}(B) = \begin{pmatrix} -1 \\ 1 \end{pmatrix}$, $\overrightarrow{pos}(C) = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$, and $\overrightarrow{pos}(D) = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$.

First, we have to compute the difference vectors $\overrightarrow{d_i}$ for every $R_i \in \{A, C, E\}$. Starting with A we get

$$\overrightarrow{d_{AB}} = \frac{1}{\sqrt{2}} \begin{pmatrix} -1-0 \\ 1-0 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} -1 \\ 1 \end{pmatrix}.$$

For C we get

$$\overrightarrow{d_{CD}} = \frac{1}{\sqrt{2}} \begin{pmatrix} 2-1 \\ 1-0 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

Say that we get for E

$$\overrightarrow{d_B} = \frac{1}{\sqrt{2}} \begin{pmatrix} 0 \\ -1 \end{pmatrix}$$

Second, we need to compute the force itself by subtracting the sum of all other difference vectors from our considered one. For A we get:

$$\overrightarrow{F_A} = \frac{1}{\sqrt{2}} \begin{pmatrix} -1-(1+0) \\ 1-(1-1) \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} -2 \\ 1 \end{pmatrix},$$

for C:

$$\overrightarrow{F_C} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1-(-1+0) \\ 1-(1-1) \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 2 \\ 1 \end{pmatrix},$$

and for E:

$$\overrightarrow{F_E} = \frac{1}{\sqrt{2}} \begin{pmatrix} 0-(-1+1) \\ 1-(1+1) \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 0 \\ -1 \end{pmatrix}.$$

How these forces affect the robots can clearly be seen in figure 5.4. In order to have an effect to the movement of the whole chain it needs to influence the movement of every single robot in the chain except the connected robot. In the above example force $\overrightarrow{F_A}$ has no effect to robot A but to B and every other robots in the chain of A (marked in red in figure 5.3). The component of the force that is parallel to the alignment of the chain has almost no influence to the robot's movement. For example $\overrightarrow{F_E}$ has only very little effect to the robots in the chain of E because the distances between them are governed by a different and bigger force.

We have left two things open: How can we get the same results without using global information and how have the robots to communicate the information needed. The force can be computed by using relative position data only (see figure 5.5): Both robots A and B know the bearing of the other one (α_1 and β_1) as well as A knows the bearing of C (α_2) and B the bearing of D (β_2). To communicate from B to A the direction of robot D seen from B (the normalized difference vector) B needs only to send the bearing of D relative to the bearing of A. In this example B would simply send $\beta_{send} = \beta_1 + \beta_2$. From this information A is in the position to compute the angle maximizing force.
Now we want to consider the communication needs. Taking figure 5.3 as

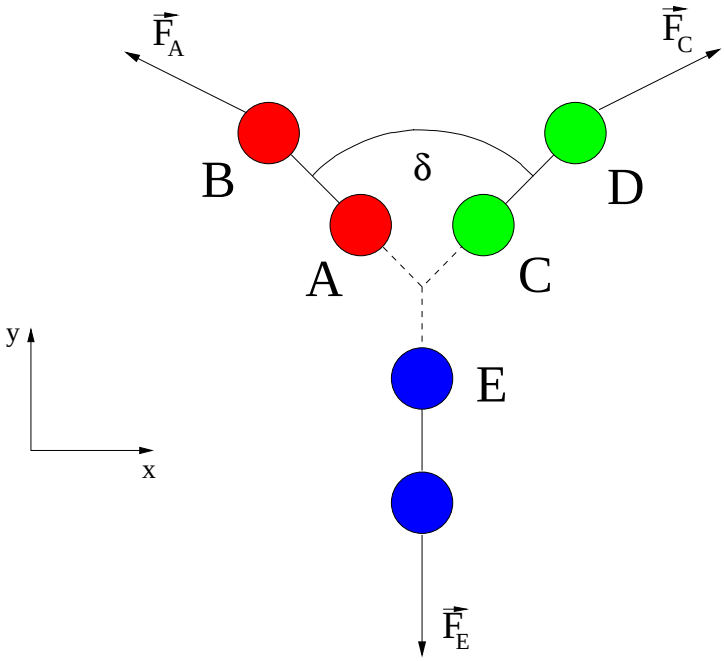


Fig. 5.4: The resulting angle maximizing forces for the example of figure 5.3.

The angle between the chains of A and C is too small ($\delta \approx 90^o$ but it should be 120°). But the computed angle maximizing force will affect the ends of the chains of A and B to drift apart and the angle will be increased while the force has basically no effect to the chain of E.

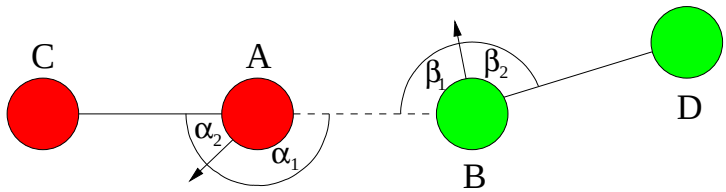


Fig. 5.5: Example of how two connected robots can communicate the bearing of the next in their chain by relative angles and without using global information.

The drawn arrows show the orientation of the robots. The angles α_1 and β_1 represent the bearing of the other robot (A and B respectively). α_2 and β_2 are the bearings of the next robot (C and D) in the chain of robot A and B respectively. From that information it is possible to communicate the bearing of C and D relative to α_1 and β_1.

our example again we want to start with the assumption that the ends of the chains of A and C are open (not connected to any other robot). In this case it could be enough to communicate the angle maximizing forces only to B and D respectively. If these forces would dominate the movements of B and D, we could expect that the other robots in the chains would align to them because of forces that straighten the chains. In fact that would be possible but there are technical arguments from the implementation of the control software point of view against this solution. But it turned out to be quite important that the ends of the chains (robots in state *TREEEND*) are not too much dominated by their neighbor in the chain (robots in state *TREESURROUNDED*) to avoid oscillations of whole chains. Therefore we have to come up with a more complicated solution. We send the information about the angle maximizing force through the whole chain and store it in every robot. This guarantees that the chains in the tree align quickly and as a whole according to the angle maximizing force.
This leads directly to the question what we should do in the case of a chain with two tightened ends. Now each robot in state *TREESURROUNDED* has to distinguish between the angle maximizing force coming from the one end and another force from the other end of the chain. The information of both forces are stored and both in principle affect the movement of the robot. All cases are imaginable: The two forces strengthen, totally compensate each other, or just sum up to some intermediate direction.
The communication between the robots in state *TREEEND* is limited to the exchange of the relative bearings of their neighboring robots. These updates occur basically every round but no synchronization is needed. Since the forces that keep the *TREEENDs* together are quite strong and defined in a way that makes them sticking together it is assured that they are within communication range at all time. The relative bearings (difference vectors) of the up to three other connected robots in state *TREEEND* can be stored and if an update does not take place for some cycles, the movement of its chain will still be relatively ordered and won't be critical to the overall performance.

The distance D between two robots in the tree is adjusted by the root of the derivation of the virtual force function. This can in principle be chosen arbitrarily. Since we want to maximize the area covered by the tree we want this distance to be as big as possible. On the other hand, however, we want to minimize the risk that the tree breaks apart. If we define the distance to be just a bit smaller than the sensor range ($D + \epsilon \approx r$), the risk is high that for some reason a robot in the tree loses its neighbor. Here the distance is

chosen conservatively as one half of the sensor range: $D = r/2$. In this case the risk that the tree breaks apart is very low and the robot can sometimes even perceive the other neighbor of its neighbor, which can be exploited to increase the distance on demand as done here in the tree reduction phase.

5.2.4 Keeping the Tree Loop-Free

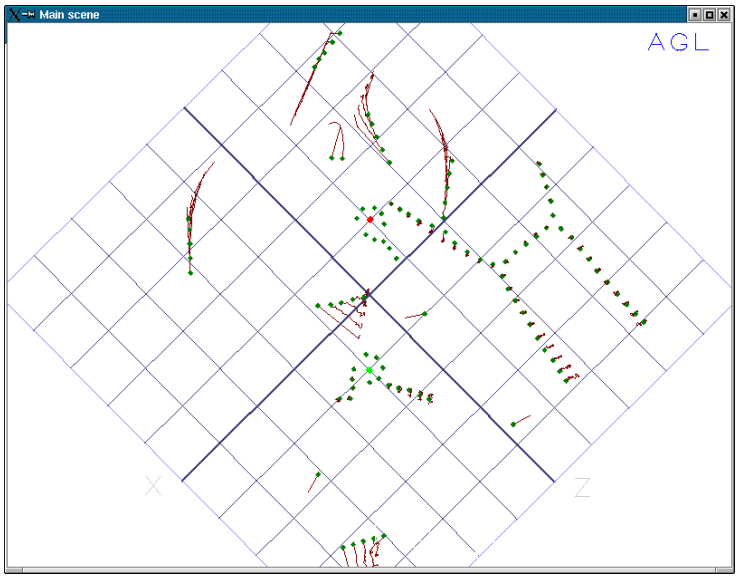
A tree is a graph without loops. It is useful to preserve this property here because without loops we obtain higher efficiency. Otherwise we could get for example two chains that are connected at both ends. These two chains would cover an area that could also be covered by a single chain, i.e. by half the number of robots.
To keep the tree loop-free we introduce another internal variable that stores an ID of the component of the overall graph the robot belongs to. For robots that are not part of a tree this variable is undefined. At the transition to *SENTINEL* the former leader of the chain generates a new component ID that is simply its own ID. Its neighbors adopt this component ID. The robots of a chain that might connected to the *SENTINELs* later will adopt it, too. By that we assure that every robot of a tree has the same unique component ID. In a situation where two different trees become connected a new component ID is generated again. Now it should represent additionally how many objects are connected by this component. This can for example be done by adding the number of robots to the former component ID of one of the two connecting trees, i.e. using higher unused bits. This new component ID needs to be propagated through the whole tree.
See figure 5.6 for typical situations in the tree formation phase.

5.3 Reduction Phase

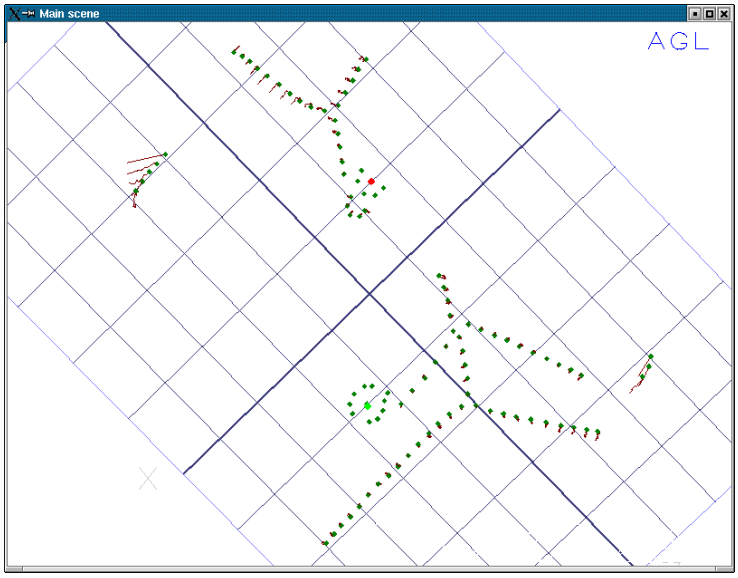
In this phase we have basically done most of the job already. All objects are connected but we want to reduce the number of robots involved. Additionally we want to find the approximately shortest path between each object. The bypassed distance is of course directly correlated to the number of robots involved. To achieve this we implement two different methods:

1. Tree leaves (robots in state *TREEEND* that are not connected to any other robot in state *TREEEND*) cut the connection to their neighbor and push off the tree.

2. Robots that are connected to a circle around an object might push off after they have made sure that the next in the tree is able to fill

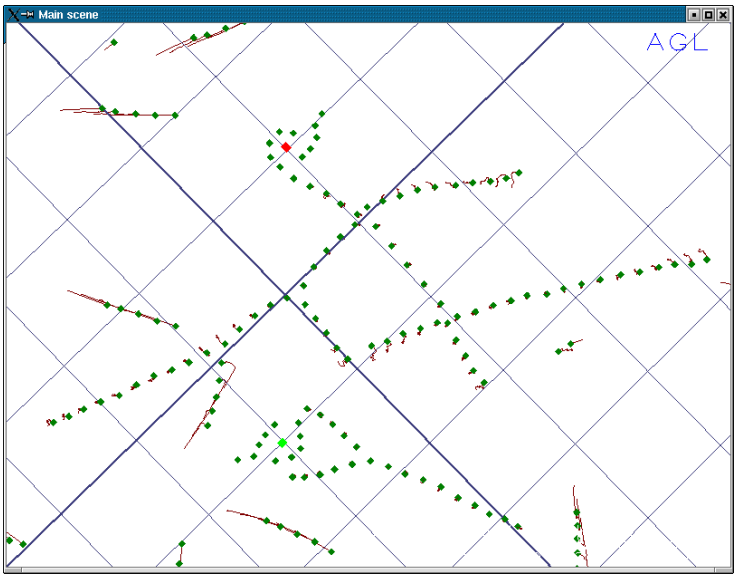


(a)

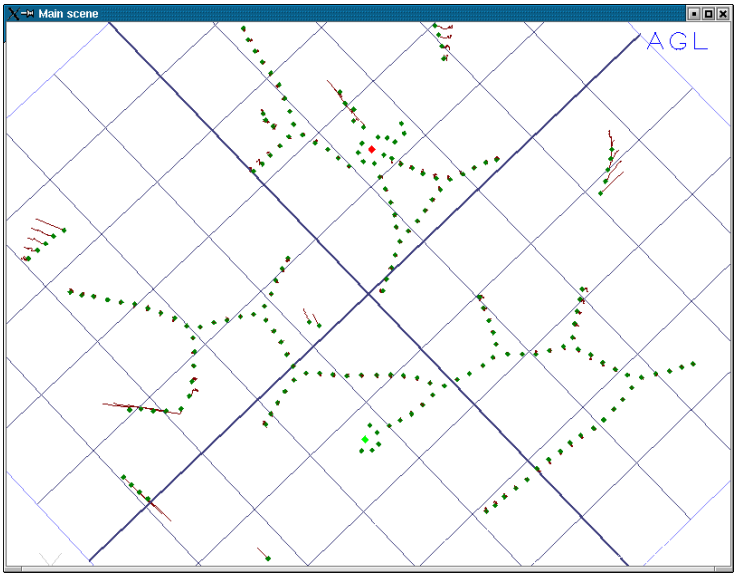


(b)

Fig. 5.6: Screen-shots of four different runs showing the tree formation phase.



(c)



(d)

Fig. 5.6: Screen-shots of four different runs showing the tree formation phase (cont.)

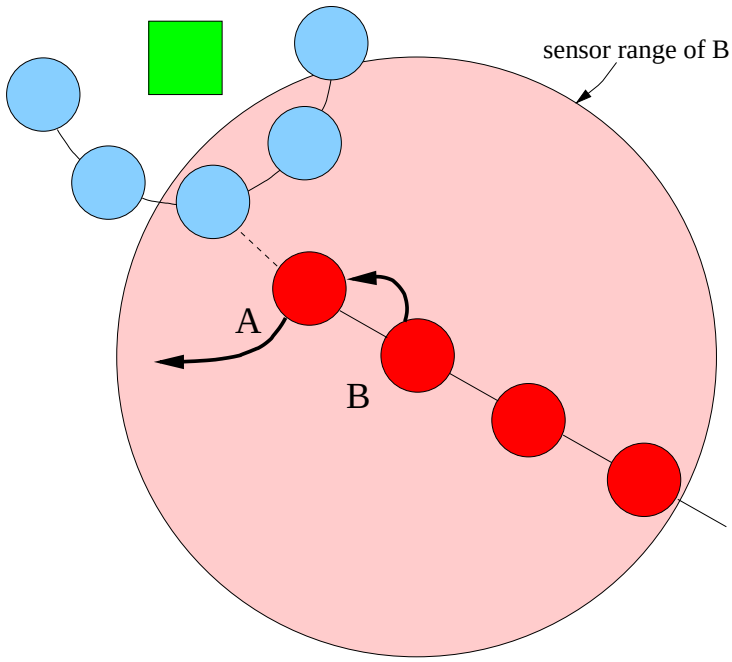
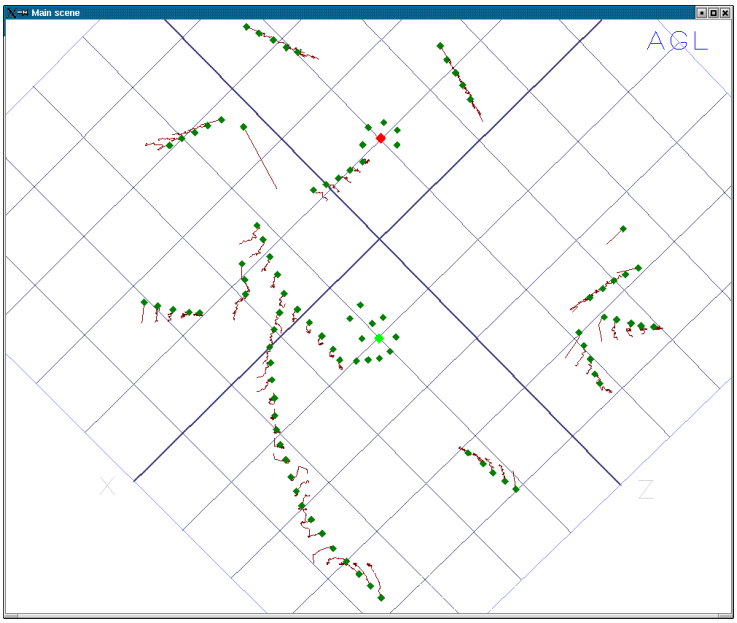


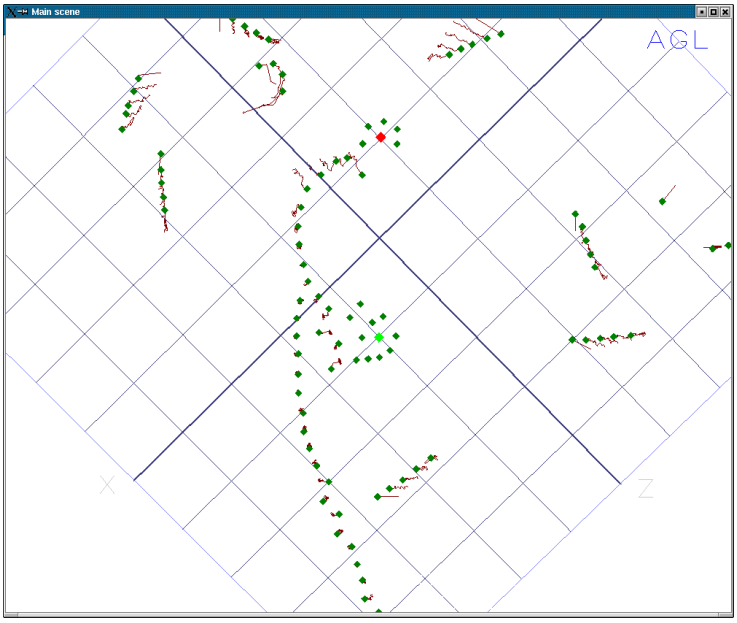
Fig. 5.7: Mechanism of reducing the number of robots in the line connecting the objects.

this gap. This leads to bigger distances between robots in the line connecting the objects.

To realize the first method we need to propagate the information through the tree that all objects are connected. This is already done by the component ID. We simply have to check whether the ID indicates that all objects are connected or not. If so the leaf transitions to *AVOIDING*. Its former neighbor becomes leaf then and in the following chain reaction all unnecessary robots push off the tree (see figure 5.8).
The second method is implemented by a more complex algorithm. Say, robot A is the first robot in the line and thus connected with the circle of robots around the object (see figure 5.7). Its neighbor in the line is called robot B. If a member of the circle around the object is within the sensor range of robot B, then B tells A that it can leave and B will take A's position. This method aligns the line straight and increases the gaps between the robots in the line to reduce the number of robots needed.

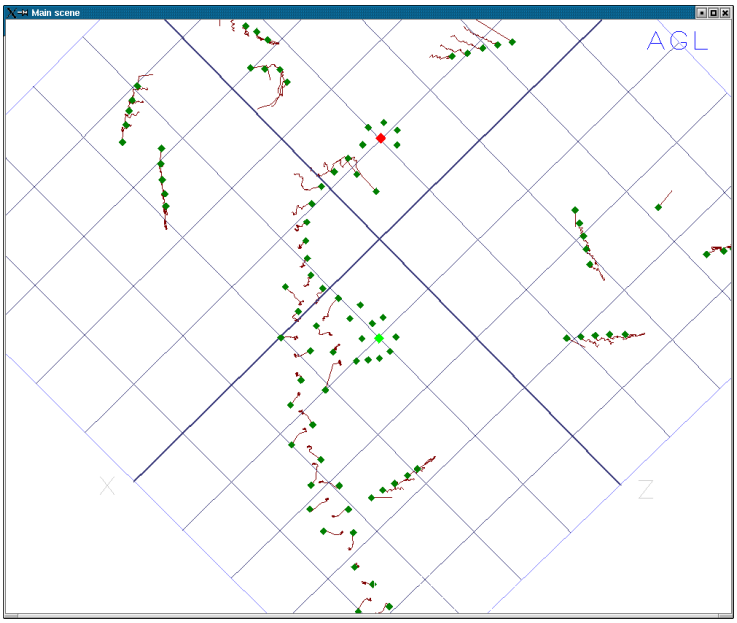


(a) Tree formation phase.

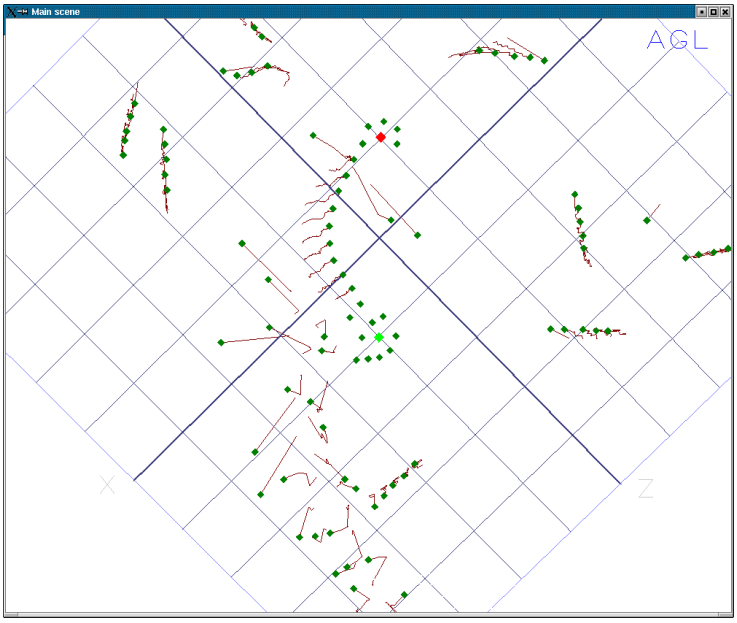


(b) Completion of the connection between the two objects.

Fig. 5.8: Sequence of screen-shots of a simulation run showing the tree reduction phase.

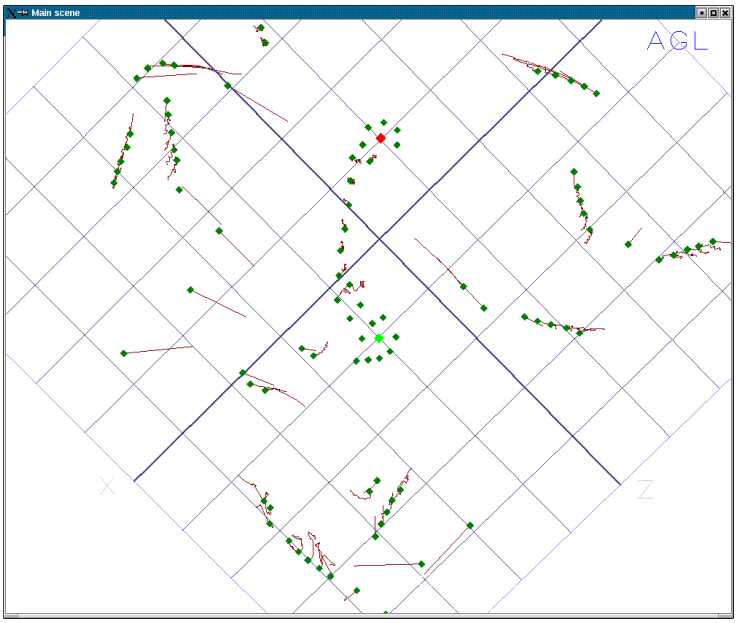


(c) Reduction of the tree begins.

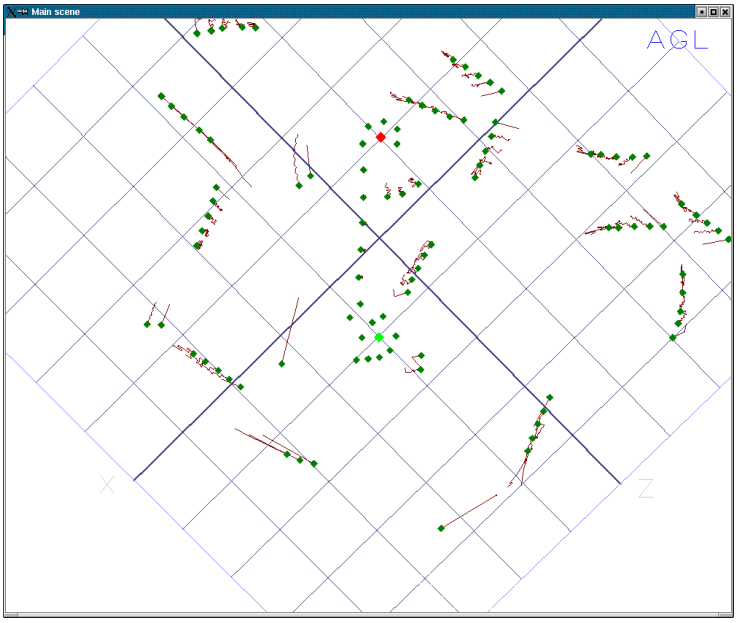


(d) Only the direct line is left and in the shortening process.

Fig. 5.8: Sequence of screen-shots of a simulation run showing the tree reduction phase (cont.).



(e) The last dispensable robots leave the line.



(f) The reduction is finished - only one straight line is left.

Fig. 5.8: Sequence of screen-shots of a simulation run showing the tree reduction phase (cont.).

Chapter 6

Random Trees

Non in favo sex angulis cella,
totidem quot habet ipsa pedes?
Quod geometrae hexagonon fieri
in orbi rutundo ostendunt,
ut plurimum loci includatur.

Does not the chamber in the comb have six angles,
the same number as the bee has feet?
The geometricians prove that this hexagon
inscribed in a circular figure
encloses the greatest amount of space.

MARCUS TERENTIUS VARRO (116-27 B.C.)
De Agri Cultura (Book III, Paragraph XVI)

There is a vast variety of applications for random trees. They are used in machine learning, planning, genealogy and ordered into many different classes: discrete and continuous trees, Galton-Watson trees, combinatorial trees, real trees and so on. The way how the random trees are constructed here and how we make use of them is most closely related to rapidly-exploring random trees [LaV98].
In this chapter we will investigate the performance of the random trees used in the base-target-scenario at large scale and compare them to simpler random trees. To do that we implemented a simple simulation software that handles robots not individually but in entire chains.

6.1 The Simulation Software

In order to make it feasible and easy to simulate huge numbers of robots the simulation is simplified compared to the original complete simulation using AnT:

1. Robots are not simulated individually but only in entire chains.
2. The arena is the unit square.
3. A chain colliding with the wall is reflected according to the rule: angle of incidence equals angle of reflection.
4. Two colliding chains turn both by 180°.
5. The sensor range is set to 1/100.
6. The length of a chain is assumed to be twice the sensor range: 2/100.
7. The speed of traveling chains is fixed to 1/100 per iteration.

It is possible to simulate trees with different maximum number of chains per joint in the tree (see section 6.2 below for a definition of x- and y-trees) and the maximization of the angles at those joints can be turned on (with alignment) and off (no alignment). At each iteration every chain of robots is moved by 1/100 except chains that are already part of the tree. The simulation stops when 95 percent of the chains are part of the tree or when the maximum of 1000 iterations is reached. It is possible to output the number of chains that are part of the tree already at every iteration. At the end of the simulation it is possible to output the geometry of the resulting tree in postscript format and the average distance of all robots to all others can be computed. This value will serve as the performance measure for our trees because we want to maximize the area covered by the robots' sensors and the all-to-all-distance is directly correlated to that. Additionally this value can be averaged over several sample runs.

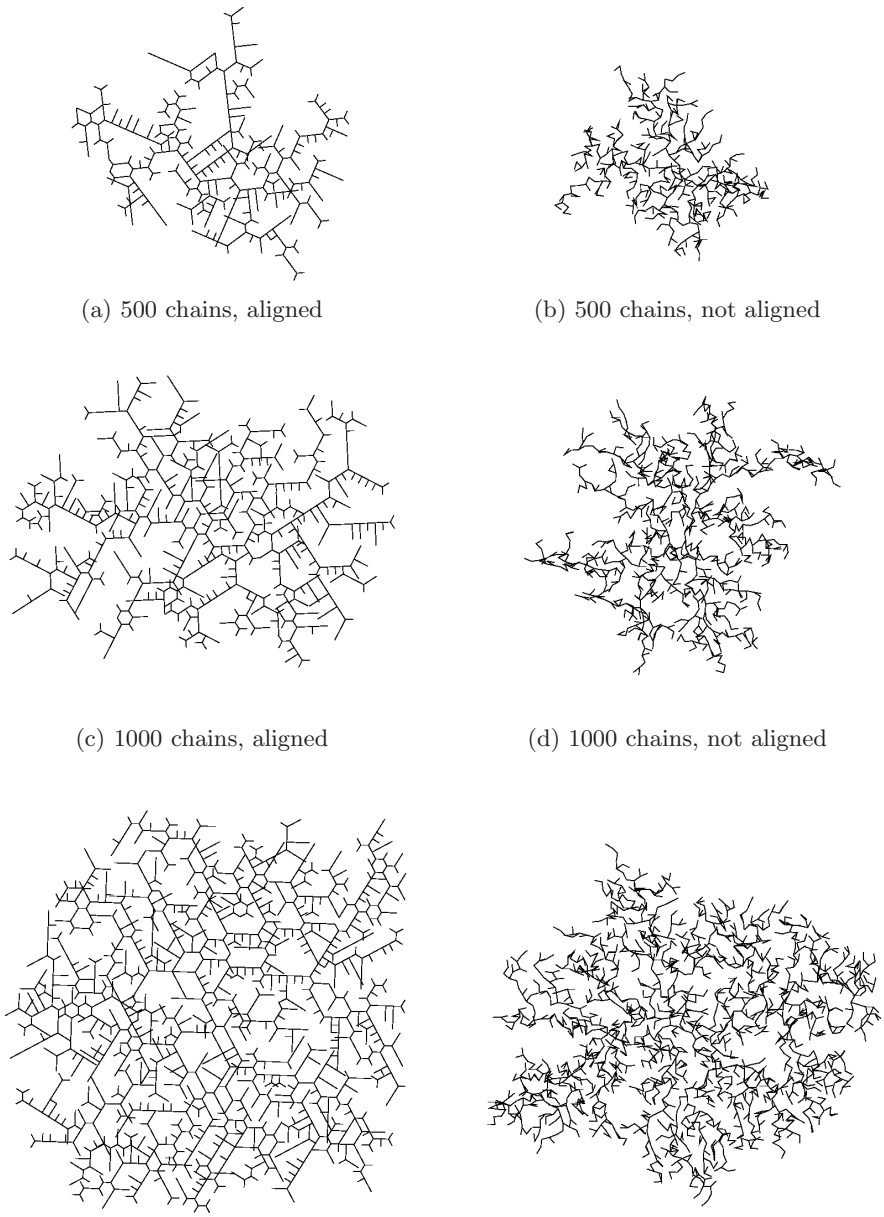
6.2 Comparison of Four Different Tree Types

In the following experiments we will vary the number of simulated chains which corresponds to the density as well as the maximum number of chains allowed per joint in the tree (three or four). At first we will allow only three chains per joint. With alignment that leads to y-shaped joints, angles

of 120° between the chains and to incomplete or complete hexagons (note that although loops are not allowed unconnected chains might happen to be close to each other). We will call this type of tree the *y-tree* in the following. In figure 6.1 examples of y-trees for different numbers of chains are presented. On the left hand side we have trees with alignment and on the right hand side trees without alignment. It can be realized already with the naked eye that the trees with alignment cover a bigger area than the other ones.

Now we will allow up to four chains per joint. Because of the right angles in the case of alignment we observe many parallel lines and crosses. But also few incomplete hexagons and y-shaped joints can be noticed. This type of tree will be called *x-tree* here. In figure 6.2 this kind of trees is compared again to trees without alignment. It seems to be tougher now to realize a bigger area covered by the trees on the left side. We will investigate that in detail later on.
Before we compare the performance of these four tree types (x- and y-tree both with and without alignment) we want to have a look at the growth of the random tree over time. Therefore we measure how the ratio of chains that are already part of the tree compared to the total number of chains develops over time. It turns out that it can nicely be described by the logistic growth (see figure 6.3 for an example). Both the shape and the speed of growth depend of course on the density. For quite low densities the behavior degenerates to linear growth (see figure 6.4). If we compare the speed of growth between the trees with alignment and the others, we notice that trees with alignment grow faster (see figure 6.5). This is true for all reasonable densities. This distinction cannot be made between x- and y-trees.

In the following we want to compare the different tree types by the performance measure we have introduced above. This is done by running the simulation and measuring the average all-to-all-distance. This value is also averaged over several sample runs and this experiment is repeated for different densities. As it can be seen in figure 6.6 the trees with alignment dominate. The very best tree type is the y-tree with alignment that is constantly better by about 0.02 compared to the x-tree with alignment. Only for the first density (50 chains in the unit square) its performance is worse than the x-tree. This is caused by its slow growth for low densities. The two tree types without alignment are far behind and interestingly there is no big difference between the x- and the y-tree. The x-tree is even a bit



(a) 500 chains, aligned

(b) 500 chains, not aligned

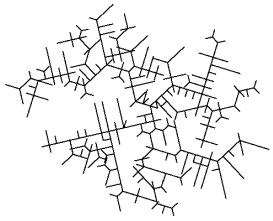
(c) 1000 chains, aligned

(d) 1000 chains, not aligned

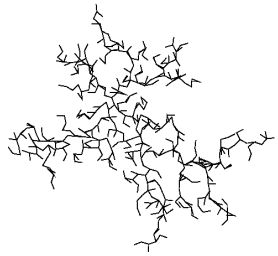
(e) 2000 chains, aligned

(f) 2000 chains, not aligned

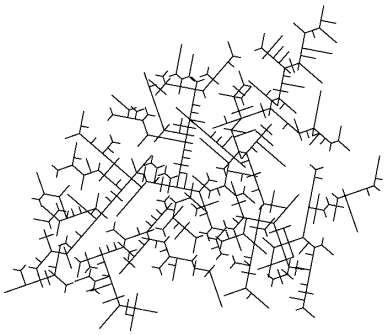
Fig. 6.1: Examples of random trees consisting of a different number of chains, max. allowed number of chains per joint is 3, with (left) and without alignment (right).



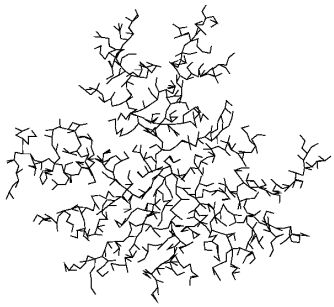
(a) 500 chains, aligned



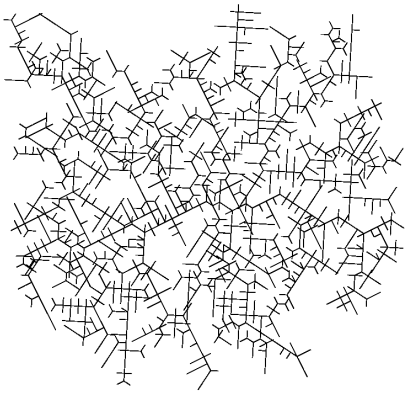
(b) 500 chains, not aligned



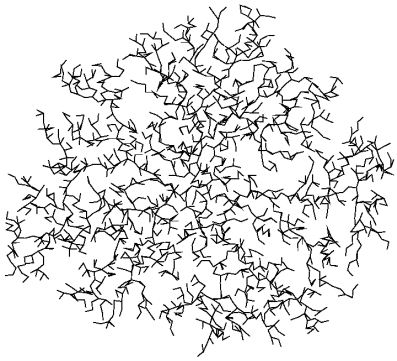
(c) 1000 chains, aligned



(d) 1000 chains, not aligned



(e) 2000 chains, aligned



(f) 2000 chains, not aligned

Fig. 6.2: Examples of random trees consisting of a different number of chains, max. allowed number of chains per joint is 4, with (left) and without alignment (right).

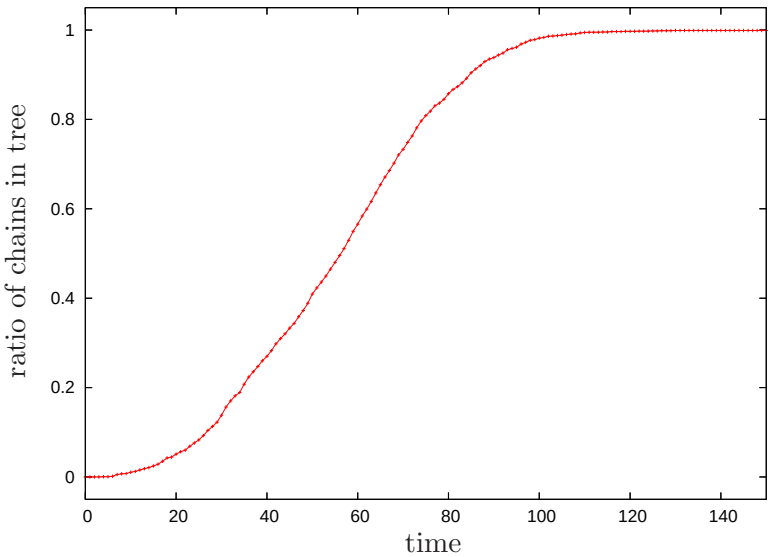


Fig. 6.3: Ratio of chains in the tree over time.

This is a typical example for logistic growth. At the beginning the tree is small and growth is slow although there are many free chains. Later it is bigger and covers enough area to get in contact with many free chains. In the end the tree is big but the resource of chains is getting low.

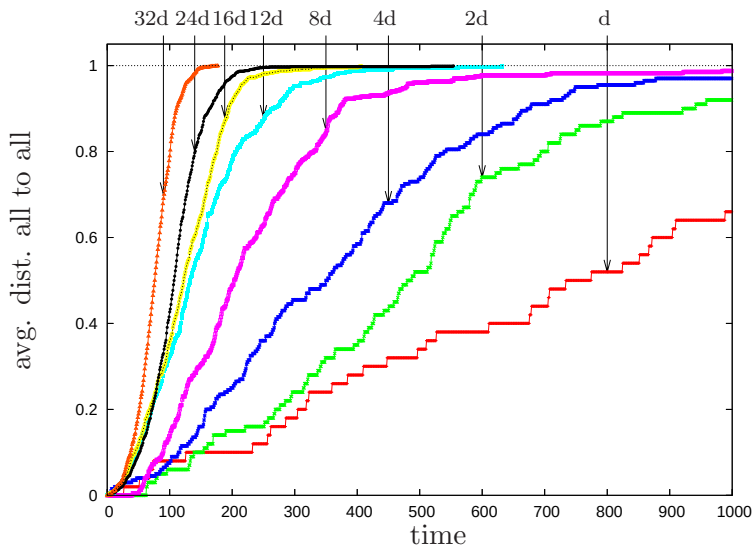


Fig. 6.4: Number of chains in the tree over time with different densities.

The density of the runs gets bigger from bottom to top. Per unit square we have $d = 50$, $2d = 100$, $4d = 200$, $8d = 400$, $12d = 600$, $16d = 800$, $24d = 1200$, *and* $32d = 1600$ *chains.*

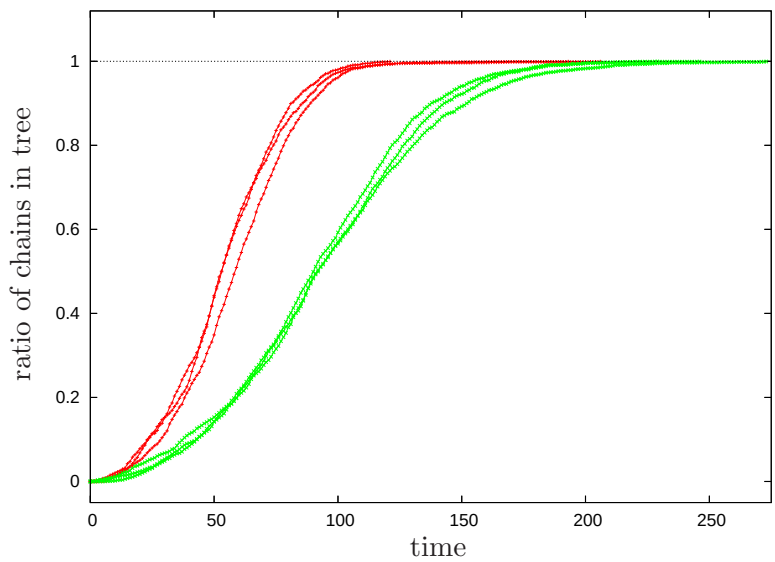


Fig. 6.5: Six sample runs of tree formations with (red) and without alignment (green).

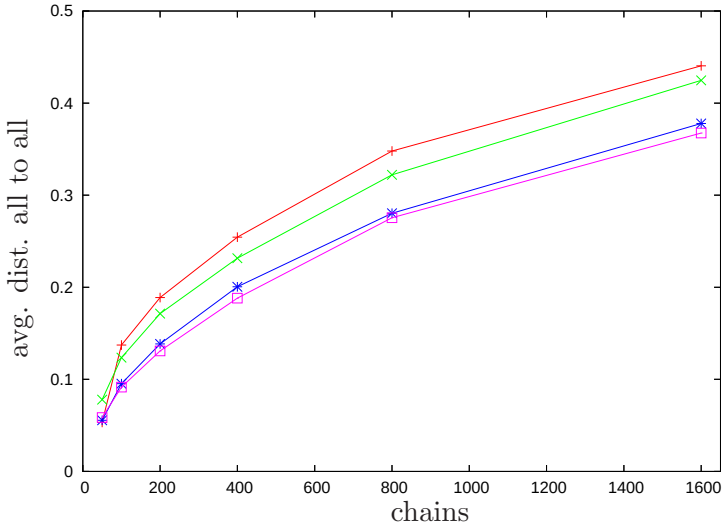


Fig. 6.6: Average distance from all to all with increasing number of chains, comparison between four tree types: y-trees with (red) and without alignment (purple), x-trees with (green) and without alignment (blue).

One can see that trees with alignment are the best choice. While with alignment the y-tree is better than the x-tree this is not true for the case of no alignment.

better than the y-tree. Looking at the results it is obviously a good idea to use angle maximization as well as y-trees instead of x-trees.

Although we have almost only incomplete hexagons and other geometric forms that are made out of parts of hexagons like y's and long lines this result is connected to the classical honeycomb conjecture: any partition of the plane into regions of equal area has perimeter at least that of the regular hexagonal honeycomb tiling (for the proof see [Hal01]). It was already stated around 36 B.C. by Marcus Terentius Varro or possibly even before by Zenodurus in "Isometric Figures" around 180 B.C. While honeybees try to minimize the wax they need by minimizing the perimeter of the geometric form they use for their bin for the honey we want to minimize the number of robots we need to cover an as big as possible area with our random tree. It looks like the bee's decision is also the right one for our artificial swarm here although we have only fragments of hexagons.

Chapter 7

Alternative Ways of Solving the Base-Target-Scenario

If I have a thousand ideas
and only one
turns out to be good,
I am satisfied.

ALFRED BERNHARD NOBEL (1833-1896)

Both the general basic features as well as the specialized one of this scenario could possibly be improved. However, these improvements are almost always a trade-off between better performance in the scenario and higher complexity of the control software. The best change to the software would be a better or the same performance with the same or even less complexity. This seems to be a hard task since efficient simplicity was already one focus of the design and development.

7.1 Possible Improvements

7.1.1 General Improvements

There are many small changes in the strategy possible. We present a small selection here:

- Insertion of robots and chains in the middle of a chain: The improvement in terms of performance is obvious. However, while the insertion of single robots might be a quite easy task the insertion of whole chains is a complex and communication intensive task. This procedure would

need to be very reliable because chains that fall apart in case of a unsuccessful insertion hurt the performance heavily.

- Chains that approach a joint without a free spot could be lead to an open end of the tree by a special virtual force. A lack of reliability would be no problem here but if the needed complexity pays off in terms of performance, is questionable.
- More than one tree per object could be allowed. This is a small and simple change. If it improves the performance would be needed to be investigated experimentally.
- The open ends of a tree could be moving instead of being static. This could possibly be done by some parameter changes. Again if this could be an improvement to the performance is open.
- Simpler geometries: Two chains could always connect orthogonally to each other or in angles of 120°. This could be a reduction in complexity compared to the angle maximization method used here. It should be investigated experimentally whether the performance stays on the same level. An improvement of performance seems to be unlikely.
- The robots of exploring chains could travel side by side instead of following the head of the chain. Then they would explore more room at a time.

7.1.2 Avoidance and Requirements of Communication in the Formation of Chains

Since every single exchange of information costs resources and the reliability of the simple communication devices used here is relatively low we want to avoid communication in all situations that don't make it absolutely necessary. However, communication is essential for a robot swarm that has to cooperate to solve a problem collectively. In the following we want to focus on the situations that make communication necessary as well as those that can be mastered with little communication or where we can reduce the amount of communication in a trade-off.

There are basically two situations in the formation of chains in which communication is essential:

1. Meeting of two incomplete chains (especially one robot meets an end of a chain)

2. Gathering information for the computation of the virtual forces of a member of a chain

Examining the first situation from the point of view of a single robot that meets a chain we can identify two distinct questions that need to be answered: Is it allowed to join? If yes: Which state transition should it perform and who will be its neighbor in the chain? Whether he is allowed to join or not is determined by checking the length of the current chain. The single robot announces that it would like to join and the addressed member of the chain initiates a counting. This is done by sending a message to the neighbor requesting the other end of the chain to enumerate. After this message has reached the other end of the chain by being handed over through every member the robot that represents the other end of the chain sends a '1' to its neighbor that increases it and sends it to the next one and so on. This intense amount of communication could be reduced by the cost of some memory by storing the current length of the chain in the ends of it. After the chain has counted its members successfully the single robot is informed whether it is allowed to join depending on the maximum allowed number of robots per chain. If so it is also informed whether it becomes the tail or the head of the chain, which is stored in its state, and who its neighbor is. The latter information needs to be stored and makes some kind of IDs necessary. These don't have to be unique over the whole swarm but could be random numbers of appropriate size for example. Another solution could be to let the robots define their IDs self-organized at the beginning of the scenario if all of them start in the same region of the arena.
The second situation is tolerant to reducing the amount of communication almost at will by the cost of overall performance. Consider as an example the situation where a robot is in the middle of a chain an he perceives only two other robots while one of them is say $\pm 45^o$ around is heading. To compute its virtual forces it needs to know who is the one in front and who the one behind him if these two forces are asymmetrically defined. Just to make sure that the forces are computed correctly this could be clarified by communication in every cycle. However, that is definitely not necessary and we could for example define to check the states every n cycles. This n could be optimized to get a good trade-off between hurting the performance and the need of communication. In other situations the robot might perceive three or more other robots that would make it more difficult to distinguish. The cost of classifying a pair of robots in the wrong way might be quite high because robots that are not members of the chain have in general a repelling effect while chain members mainly attract. Although it might be

too much overhead and might need too many resources one could think of training a learner that classifies those robots and that decides if we should communicate.

7.1.3 Avoidance and Requirements of Communication in the Formation of Trees

Communication is needed in three different situations in the formation of trees:

1. Joining of a chain with the tree.
2. Exchanging the information of the relative angles between chains and bearings of connected robots in state *TREEEND*.
3. Updating the IDs of the connected components in the tree.

If the head or the tail A_1 of a chain meets a robot B_1 in state *TREEEND* (see figure 7.1), these two robots communicate at first to determine whether the new chain is allowed to join (only up to four connected chains are allowed depending on the tree type). If so B_1 tells A_1 the IDs of the up to two other robots C_1 and D_1 representing the chains that are also connected and B_1 receives the ID of A_1. This information about the new connected robot needs to be propagated to the other up to two robots. Typically they are in the communication range of B_1 so that B_1 simply sends them a message. If for some reason it doesn't see them, it has to try it again in the next cycle and needs some variable to remember that. This could also be implemented as a permanent check for an update. See the next paragraph for a discussion of that.
To compute the angle maximizing force connected robots in state *TREEEND* need to update the angle information (that is the angle between the bearing of B_1 seen from A_1 and the chain of B_1, B_2, and so on) about the other connected chains. A simple way of implementing that could be as a permanent update. For example we could initiate an update in every cycle if two connected robots see each other. That would mean that they have to communicate almost in every cycle. To avoid that much communication overhead we could define some criteria when a robot should send his new angle information. This could be for example a fixed number of cycles or some threshold in the change of the angle (for example communicate only if the present angle differs from the old one by more than 4°). Again this is a trade-off between communication and memory overhead.
Every tree or connected component of robots has an ID. This ID changes

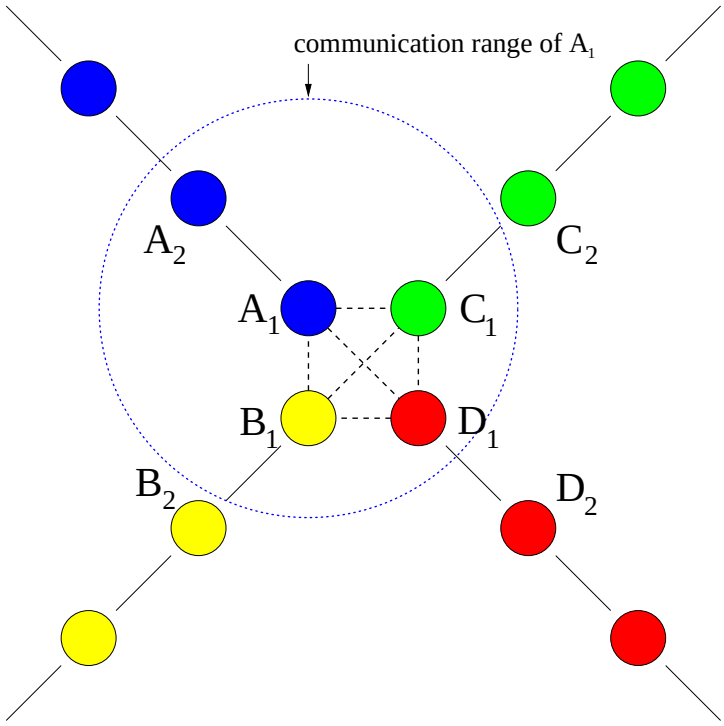


Fig. 7.1: Example of four connected chains in a tree.

Up to four connected chains in a tree are allowed. The robots A_1, B_1, C_1, and D_1 are in state TREEEND. Each of them knows the IDs of each other as well as the angles between their chains.

if two components join. In this case the new component ID is negotiated by the two robots that have connected to each other and were parts of two different connected components before. This new ID has to be propagated through the whole component. This can simply be done by sending messages to the connected robots informing them of the new ID. This message would make the receivers send new messages in turn to the robots they are connected to while of course omitting the robot who sent them the message. Using some kind of permanent update for the connected components ID would obviously be too much unnecessary communication overhead.

7.2 Other Strategies

The number of possible strategies is unlimited as the number of algorithms one can define for a given complex problem. We state here some of our alternative ideas that would be interesting to be pursued.

7.2.1 Simple Strategies

We start with some strategies that cannot stand alone and that could be used easily together with the strategy that was implemented in this work.

- At the beginning we try to achieve a uniform distribution of the swarm in the hope that they find the object faster. The success of this strategy depends strongly on the density and the maximum range of the sensors.
- Trying to exploit the fact that the swarm is in a variant of the scenario initially close together at the base we let them build one big chain beginning or a tree starting at the base. Probably distant objects would be hard to localize by this method.
- The above strategy can be combined with the implemented one by dividing the swarm in two halves. One half starts to build a tree at the base while the other half explores the rest of the arena and starts to build trees at other objects. The devision could be done by every robot individually at random or depending on its environment.
- To reduce the number of incomplete chains especially single robots that walk around to find others we could define an area at the base that can only be left by complete chains. This would increase the local density and thus the probability that two robots meet is higher.

- Also chains that are not connected to an object could be allowed to connect to others. It should be taken into account that such free trees might not be able to travel bigger distances.

7.2.2 More Complex Strategies

More complex strategies are conceivable that make a more intensive use of communication and that contribute even a different philosophy. Here we present three ideas shortly:

- We start with small chains to explore the arena efficiently. After a chain has found an object it sends some robots out to spread this message. Other chains that get to know this connect with others to form bigger chains to be more efficient in the building of trees. The density of small chains needs to be high enough so that the probability of the meeting of two chains is sufficiently high.

- As an extension of the above strategy we can abandon the idea of fixed chain lengths. Instead we could allow chains of arbitrary lengths but introduce special single robots that act as *separators*: A robot that is part of a chain and meets such a separator cuts the connection to one of its neighbors in the chain. By that the length of chains would be governed by the local density of the separators. This density could be influenced by chains that find objects and make some of their members to separators. It could also be adjusted by the separators themselves. If they meet other separators regularly, they could perform a transition to a chain building state to reduce the number of separators.

- There is a relatively simple way of improving the orientation of the robots. They could notice the time since they saw the base or another object for example. When a robot A meets another robot B that tries to find the base, A tells him the time since he saw the base the last time. If that was quite recently, B has just to turn to the direction A comes from and will find the base with a higher probability.

7.3 Ideas for Robots with Different Hardware Designs

In case of a heterogeneous swarm many new possible strategies open up. For example we could assign roles to the robots according to their hardware:

- Exploration: Robots with high nominal velocity as well as many sensors with wide range to the front.
- Part of the tree or in the middle of a chain: nominal velocity is of limited importance; sensor range can be small but complete circumferential visibility would be useful.
- Chains: Each chain could be equipped with robots of different types: head and tail with many sensors, in the middle for example robots with wide communication range, high computational power, or a big memory.

Part III

Analysis

Chapter 8

Analysis of the Chain Formation Scenario

An inextensible heavy chain
Lies on a smooth horizontal plane,
An impulsive force is applied at A,
Required the initial motion of K.

JAMES CLERK MAXWELL (1831-1879)
L. Campbell and W. Garnett
The life of James Clerk Maxwell
with selections from his correspondence and occasional writings

8.1 Introduction

In this part we want to define a list of minimal demands to the configuration of the robots based on the simulation. Because of the complexity of the I-SWARM scenario we expect nontrivial correlations for example between the number of sensors and the performance of our robot swarm. While a linear correlation would just imply that more is better we would like to find critical values that separate configurations of bad performance and successful configurations clearly.

8.2 Sensors

8.2.1 Number of Sensors

According to the current hardware specification the used sensors can only report the presence of an object but not its bearing. Therefore the underlying assumption in the simulation is that the uncertainty of the angle measure is directly correlated with the number of sensors. The resolution of the angle measurement corresponds to the number of sensors. If a robot has for example only two sensors, one to the front and one to the back, then it can only report bearings of 0° or 180° depending on whether the object is in the range of the front or the back sensor. In the following simulation we assume that the robot is equipped with several sensors that are all of the same kind. All of them cover 36° (this is different from the current hardware specification) so that ten of them result in circumferential visibility and the robot can distinguish ten different angles. The scenario used here is the chain formation in a swarm of 50 robots. The nominal chain length is five and the density is set to $\rho = 0.002$. For each number of sensors between two and ten the results of ten experiments are averaged. The ratio ξ of robots that are part of a chain to all robots is measured for different numbers S of sensors. The results are shown in figure 8.1.
There is an obvious kink at $S = 6$ (see also below in sections 8.2.3 and 8.2.4). Thus it would be a bigger increase in performance if the robot would be equipped with seven or eight sensors instead of five or six. However, the current hardware specification includes only four sensors (see section 3.2). But these sensors have a broader range (60°) and with an increased density the performance can also be improved.

8.2.2 Sensor Range

The maximal range of the sensors has only little influence to the performance in the chain formation scenario (see also figure 8.7 and section 8.4). Depending on the density a bigger sensor range might help increasing the performance.
Note that by the sensor range also the distance D between two robots in the chain are defined and scale linearly. This can cause side effects for extreme values. For very small sensors ranges r these distances become negative: $D = (r - 2s)c + 2s$ is negative for

$$r < 2s - \frac{2s}{c}$$

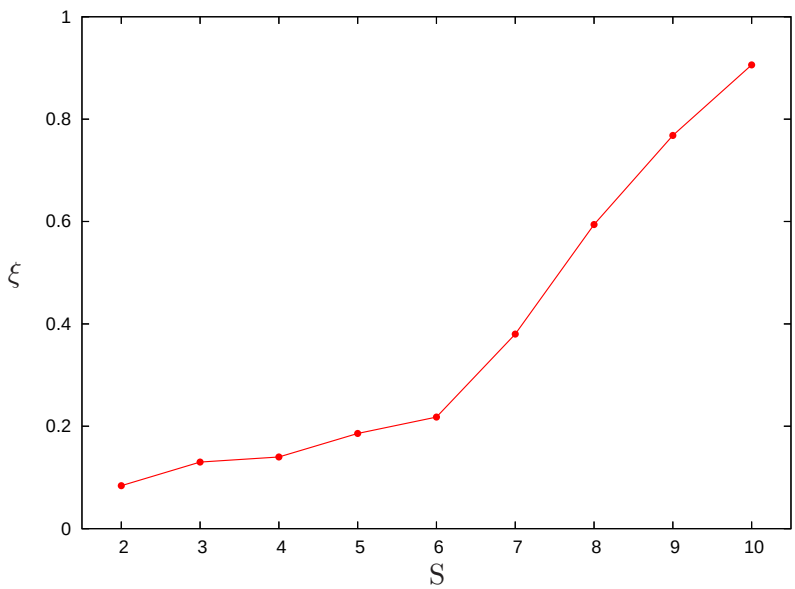


Fig. 8.1: Number of sensors over the performance in the chain formation scenario.

The ratio ξ of robots that are part of a chain to the number of robots for different numbers S of sensors is measured. One can see that there is a kink at $S = 6$.

with s robot size and some constant $0.0 < c < 1.0$. For very big values of r the density ρ is the restricting parameter here. The needed area per robot in a chain is πD^2. In the border case we set $\pi D^2 = 1/d$ and get

$$r = \frac{1}{c}\left(\sqrt{\frac{1}{d\pi}} - 2s\right) + 2s.$$

For bigger values of r the performance will decline clearly.

8.2.3 Angle Uncertainty and Partial Visibility

The arrangement of the sensors on the robot as well as their internal characteristics provide two distinctive features: the covered and uncovered area and the uncertainty in the angle measurement. Both features will typically depend on each other but are investigated individually in this and the next section to determine the more important one.
In the following simulations the number of sensors S is fixed but their range r in degrees is varied. We define the visibility ratio:

$$v = \frac{S \cdot r}{360^o}.$$

So v can range from 0 (no covered range, i.e. completely blind robot) to 1 (complete circumferential visibility). Figure 8.2 shows the performance of the chain formation (ratio of robots in chains to all robots ξ) in dependence on the visibility ratio averaged over 100 runs with a constant number of $S = 8$ sensors. The kink that was observed at $S = 6$ in figure 8.1 would correspond to $v = 0.6$. But no extreme change in the slope can be noticed here in this experiment. The slope is almost constant for $v < 0.4$ but for $v > 0.4$ an increase in the covered area causes a noticeable better performance. The I-SWARM hardware specification has a visibility ration $v \approx 0.67$ which is a quite good value. But the I-SWARM robots will have only four instead of the eight sensors in this simulation here.

8.2.4 Angle Uncertainty

In this section we will vary the number of sensors with fixed visibility ratio. This will change the uncertainty of the angle measurement only. We assume the robots have circumferential visibility and thus we set $v = 1$. The number of sensors S is varied between two and ten. In the case of $S = 2$ each sensor has to cover 180° since $v = 1$ is fixed. But the front sensor will assign the bearing of $\phi = 0^o$ (right in front of the robot) to

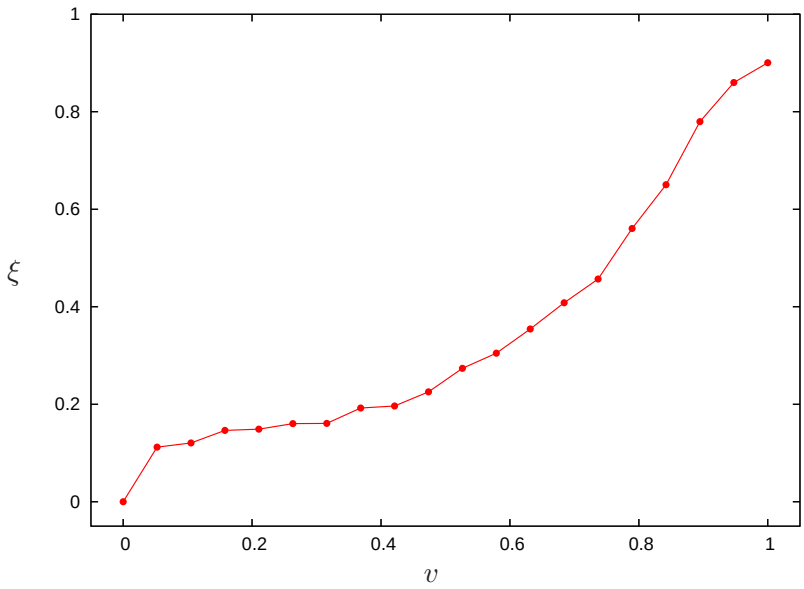


Fig. 8.2: Influence of the visibility ratio on the chain formation with fixed number of sensors.

The number of sensors is fixed to $S = 8$. The visibility ratio v describes how much area around the robot is covered by sensors and measured is the ratio ξ of robots in chains to the number of robots. Clearly for $v < 0.4$ an increase in the covered area has only little or no effect to the performance. In contrast to that the effect to the performance for $v > 0.4$ is much bigger.

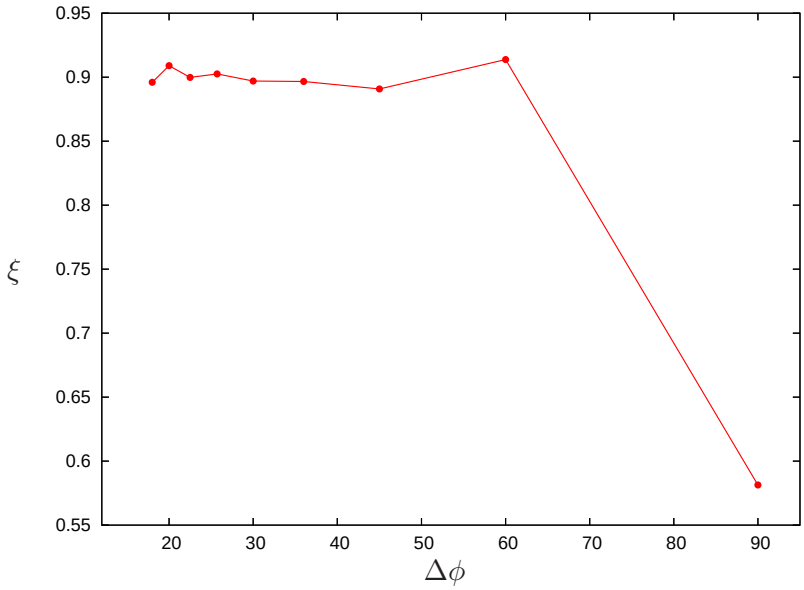


Fig. 8.3: Precision in the angle measurement over the performance of the chain formation scenario.

From left to right the points correspond to 10,9,...,2 sensors and complete circumferential visibility ($v = 1$) for all numbers of sensors is assumed in this simulation. Already the resolution of three different angles (three sensors, $\Delta\phi = 60^o$) is sufficient.

each object in its range and the back sensor will always assign $\phi = 180^o$. Therefore we get a maximal uncertainty in the angle measurement of $\Delta\phi = 90^o$. For $S = 3$ we get $\Delta\phi = 60^o$, for $S = 4$ we get $\Delta\phi = 45^o$, and so on. Again we measured the performance in the chain formation scenario depending on $\Delta\phi$ averaged over 100 runs for $S = 2, 3, ..., 10$ and thus $\Delta\phi \in \{18^o, 20^o, 22.5^o, \sim 25.7^o, 30^o, 36^o, 45^o, 60^o, 90^o\}$ (see figure 8.3). Obviously already the large angle uncertainty of $\Delta\phi = 60^o$ is sufficient for the chain formation scenario.
Figure 8.4 shows the results of the same simulation but additionally with varied visibility ratio. For high angle measurement precision $\Delta\phi < 20^o$ the performance seems to depend linearly on the visibility ratio while for less accurate angle measurements $\Delta\phi > 40^o$ we notice a drop for $0.75 < v < 0.85$. In the current I-SWARM hardware specification we have $v \approx 0.67$ and $\Delta\phi \approx 30^o$. An increase in the visibility ratio seems to be more attractive to aspire here compared to an increase in the angle measurement precision.

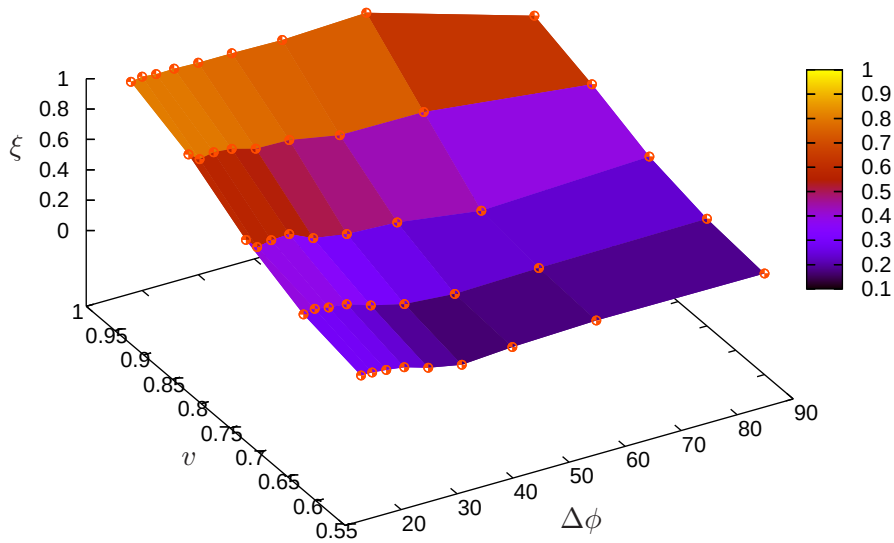


Fig. 8.4: Performance over visibility ratio and precision of the angle measurement.

The ratio of robots in chains ξ is measured over the visibility ratio v and the angle uncertainty $\Delta\phi$. The importance of a good angle measurement declines with an increasing visibility ratio. But for a fixed small visibility ratio high precision in the angle measurement results in a good improvement.

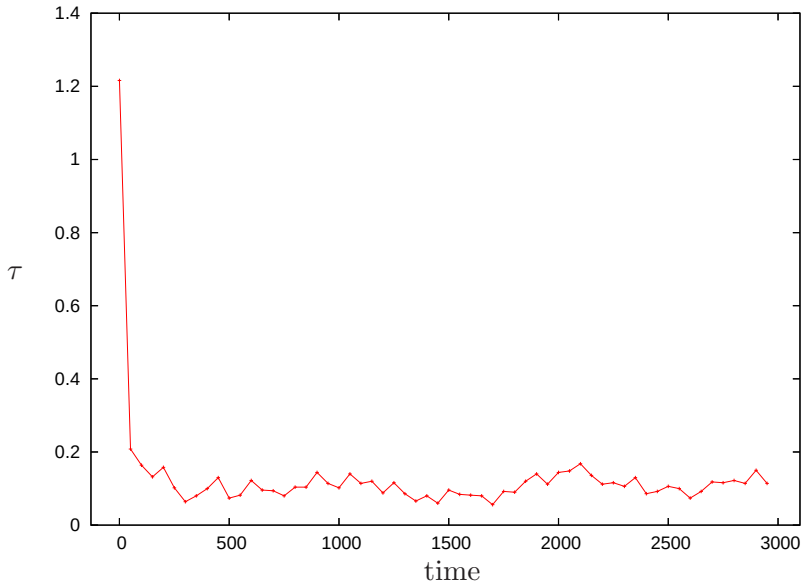


Fig. 8.5: Ratio of transitioning robots within 50 iterations over time

The ratio τ of robots that perform a transition within 50 iterations (bins of size 50) is measured over 3000 iterations. The results presented are averaged over ten runs. The chains are completely formed within some hundreds of iterations. Thus only the noise caused by robots loosing the connection to their chain accidentally and rejoining again can be seen here.

8.3 Communication Intensity over Time

There are some actions of robots that are caused by communication. The most significant is the state transition of a robot. Most of the transitions involve communication with nearby robots. Thus we can use the number of transitions in the swarm as an indicator for the intensity of the communication. Over periods of 50 iterations we will measure the ratio of robots τ that have performed a transition within this interval. See figure 8.5 for the results that are averaged over ten runs. Since the formation of almost the maximal possible number of chains is done within some hundreds of iterations in this scenario rather the maintaining is measured than the actual obtaining of chains. This fact is expressed in the noise that can be seen in figure 8.5. It is caused by robots that loose the connection to their chain accidentally and rejoin again.

8.4 Density

The density of the robots in a scenario can be changed by varying the number of robots or the size of the arena. In a typically application the size of the arena will be fixed and only the number of robots will be variable. Thus it is easy to decrease the density by removing some robots but it will be a problem to increase the density since the number of robots will always be limited. Therefore the feasibility of a scenario will in general be restricted by a minimal density.
In this section we want to determine the minimal density for the chain formation scenario so that the performance is satisfactory. This is done by placing a fixed number of $N = 50$ robots uniformly distributed in arena of different sizes. By N and a fixed density ρ the length of the quadratic arena is defined by $\rho = \frac{N}{a^2}$. Since ρ is inversely proportional to a^2 for a fixed N we measure the performance over a logarithmic scale for the densities. See figure 8.6 showing the results for ten different densities measured in $\frac{robots}{mm^2}$ and averaged over 50 runs. For the critical value we detect $\rho_c \approx 10^{-4} \frac{robots}{mm^2}$. That corresponds to an arena with the area of $500,000mm^2 \approx 0.5m^2$ or a square of length $\sim 707.1mm \approx 70.71cm$. While the diagram shows almost perfect performance for big densities ($\rho > 0.1$) it should be mentioned that in such dense scenarios the successful formation of a chain cannot be noticed by the bare eye. This formations are indeed built but only in an informational way, i.e. not geometrically but in the internal variables of the robots.

Since the robots are uniformly distributed over the whole arena and at the beginning of the scenario they are traveling in an arbitrarily direction obviously at some density the probability that two robots meet becomes very low. This low probability can only be compensated by an increasing cost of time but in the simulation here the time is fixed. However, depending on the application it might be very unlikely that the robots are uniformly distributed over the whole arena. Therefore we performed the same experiment as above but placed all 50 robots uniformly distributed in a square of length $160mm$ if that does not exceed the size of the arena. This results in a local density of $\rho \approx 0.002 \frac{robots}{mm^2}$. The measured performances for this experiment can be compared to the above in figure 8.6. The performance does not differ for $0.002 \leq \rho \leq 1$ since the two experiments are exactly the same for these values. But for further decreasing densities the performance of the experiment with concentrated robots declines only a bit and stays constant for much smaller densities. Even for the lowest density of $\rho = 10^{-7} \frac{robots}{mm^2}$

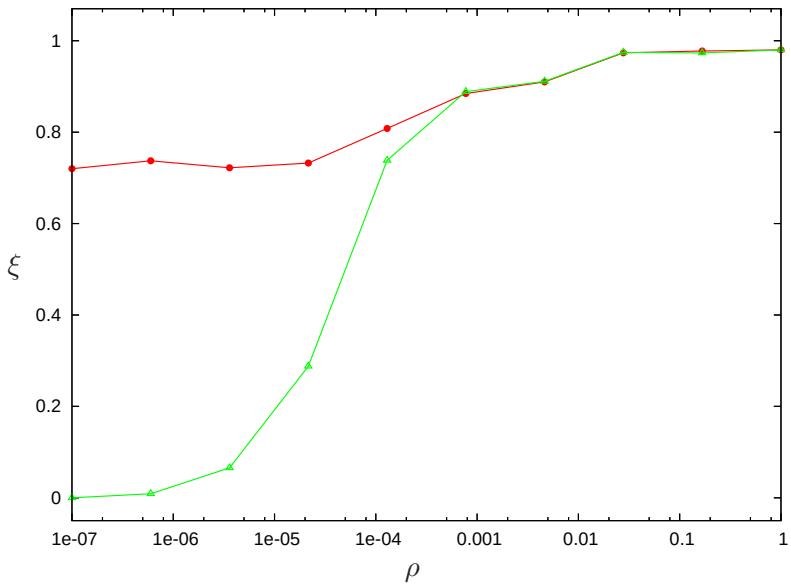


Fig. 8.6: Density over performance with uniformly distributed (green) and with in one place concentrated (red) robots.

The performance (ratio ξ of robots in chains) in the experiment with uniformly distributed robots declines noticeably for decreasing densities as one would expect while in the experiment with initially concentrated robots it declines only a bit and stays constant after that.

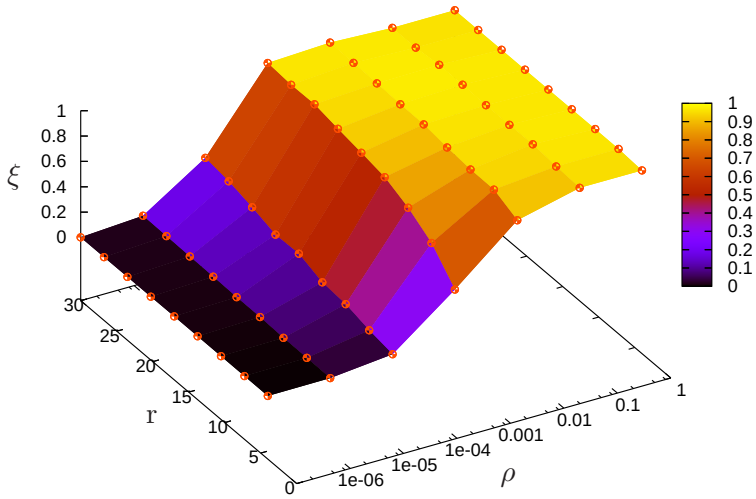


Fig. 8.7: Performance over density and sensor range.

The sensor range r has only little influence to the overall performance (ratio ξ of robots in chains). Only for critical densities $10^{-6} < \rho < 0.001$ a bigger sensor range is definitive an improvement.

that corresponds to huge arena of $500m^2$. Apparently the size of the arena has almost no influence to this scenario with initially concentrated robots. Finally we want to investigate the connection between the maximal range of the sensors and the density. We measure the performance in the chain formation scenario over two parameters: the density and the maximal view range. Figure 8.7 shows the results averaged over 50 runs. Big sensor ranges seem never to hurt the performance in the chain formation scenario but help at critical densities $10^{-6} < \rho < 0.005$. But see also section 8.2.2 for the discussion of extreme sensor ranges.

Chapter 9

Analysis of the Base-Target-Scenario

A forest is the triumph
of the organisation
of mutually dependent species.

ALFRED NORTH WHITEHEAD (1861-1947)
Science and the Modern World (p. 297)

9.1 Introduction

As in the chapter above we want to find critical parameters but now in the more complex base-target-scenario. We want to do this following roughly the structure of the above chapter. But we will focus on the special characteristics of this scenario and have two new sections at the end of this chapter focusing on the difference between x- and y-trees and the distance between the base and the target object.

9.2 Sensors

9.2.1 Number of Sensors

As in chapter 8.2 we begin with the analysis of the number of sensors. For each simulation we fix the range a single sensor covers (here: 60°, 36°, and 10°) and measure the performance in the base-target-scenario for different numbers of such sensors averaged over 100 simulation runs. The performance is expressed in the probability P that the scenario is completed successfully.

An additional sensor increases the visibility ratio v and the robot can identify an additional bearing. With three sensors that cover 60° each for example a robot has the visibility ratio $v = 0.5$ and can distinguish three different bearings of objects: $\phi_0 = 0^o$, $\phi_1 = 120^o$, and $\phi_2 = 240^o$. An additional sensor increases the visibility ratio to $v \approx 0.67$ and the following bearings are possible now: $\phi_0 = 0^o$, $\phi_1 = 90^o$, $\phi_2 = 180^o$, and $\phi_3 = 270^o$.
Figure 9.1 shows the results. It is obvious that the covered range of a single sensor is more important than the absolute number of sensors, i.e. the number of distinguishable bearings. If the visibility ratio gets above a critical value of about $v_c \approx 0.7$ that only depends a little on the number of possible bearings, then the performance increases suddenly.
Thus a useful and inexpensive way to equip the robot with sensors would be to provide it with few sensors of a wide range so that the visibility ratio is close to one.

9.2.2 Sensor Range

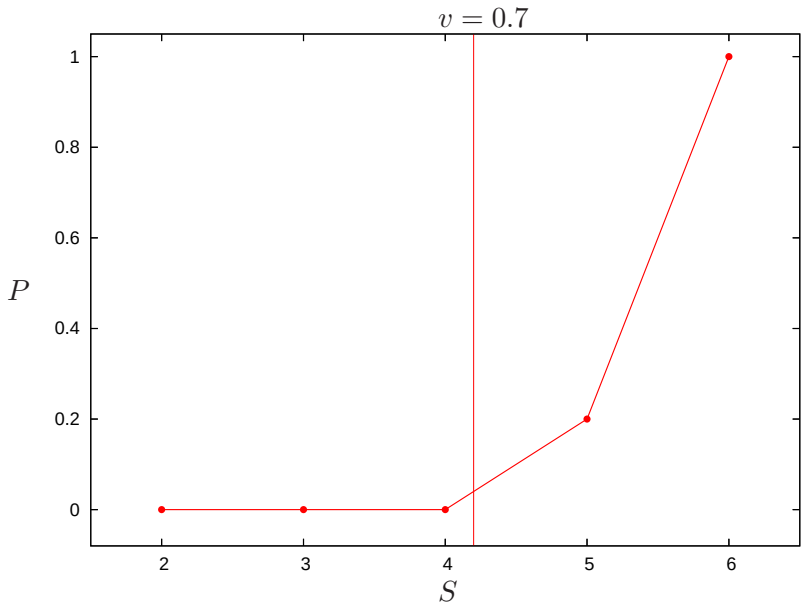
The influence of the sensor range r to the performance in the base-target-scenario is investigated in figure 9.2. The performance of 100 robots for 14 different sensor ranges from $4mm$ to $30mm$ were tested averaged over 100 runs. The declining performance for $r > 20mm$ can be explained by the fixed density as done in section 8.2.2: The distances between the robots in the trees are set to $D = r/2$. Thus each robot is satisfied by a minimum area of about $\pi D^2 = \pi(\frac{r}{2})^2$. In the border case we set

$$\pi(\frac{r}{2})^2 = \frac{1}{d}. \tag{9.1}$$

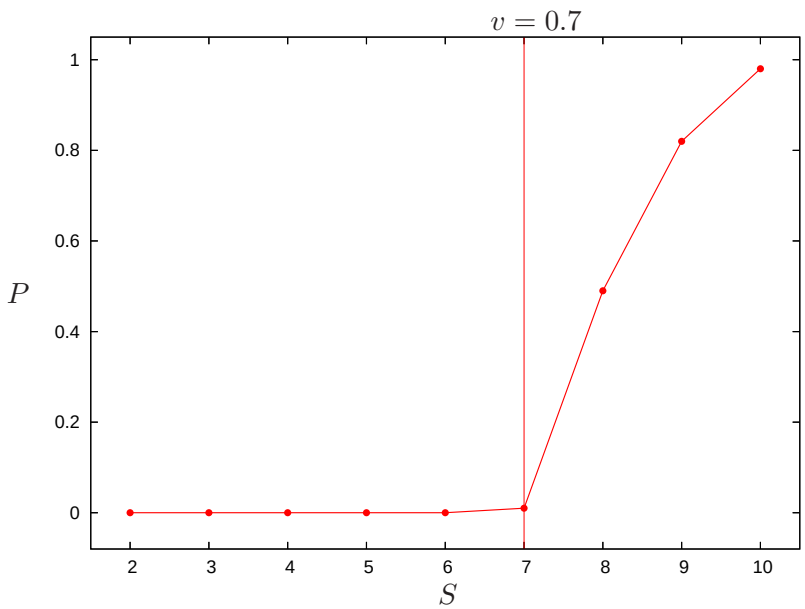
Note that with the difference of a factor of four ($\pi r^2 = \frac{1}{d}$) we get the case that the robots cover exactly the arena with their sensors.
With the density $\rho = 0.003\frac{robots}{mm^2}$ that was used in this experiment here we get

$$r = \sqrt{\frac{4}{d\pi}} = \sqrt{\frac{4}{0.003\pi}} \approx 20.6(mm)$$

Thus for $r > 20.6$ the area per robot that is available becomes too small. That results in worse maneuverability, more movements in the tree, and therefore in worse performance.
Note the peak around $r \approx 8$ and the sudden drop at $r \approx 10$ that cannot be explained yet. Actually one would await an increasing performance with increasing sensor range or a constantly high performance for $r \ll 20.6$ and $r \gg 0$. See also section 9.4.

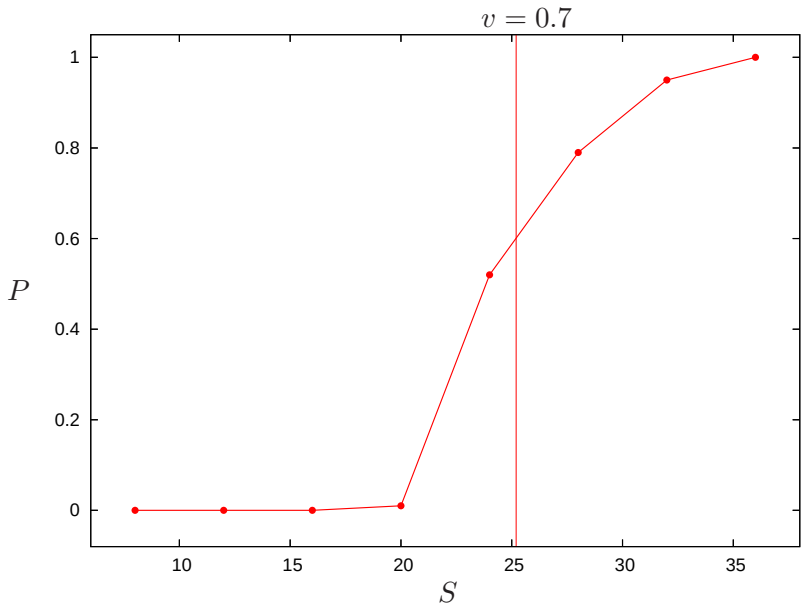


(a) Each sensor covers 60°.



(b) Each sensor covers 36°.

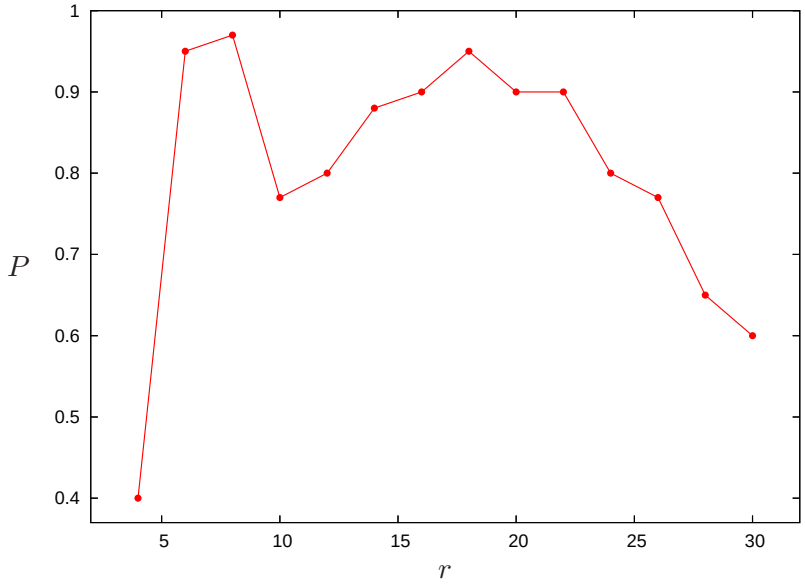
Fig. 9.1: Performance over the number of sensors.



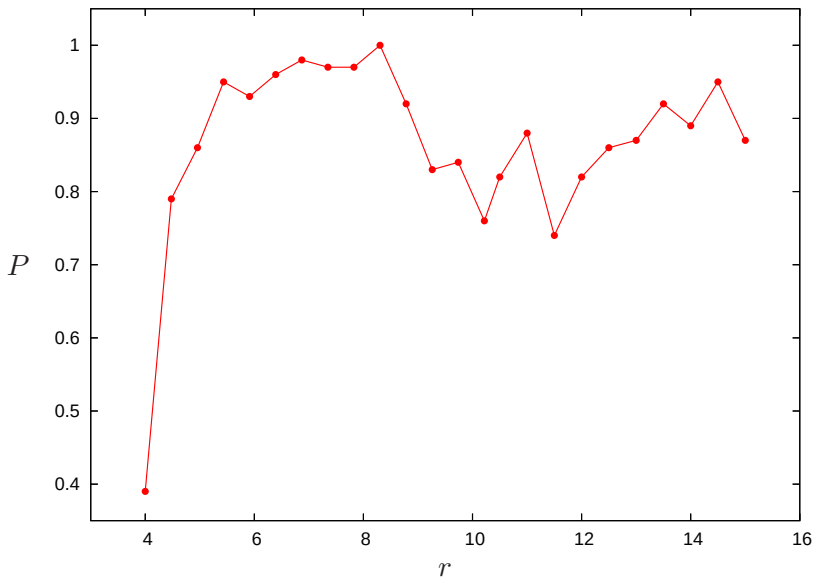
(c) Each sensor covers 10°.

Fig. 9.1: Performance over the number of sensors (cont.).

One can see, that the visibility ratio v is more important than the absolute number of sensors. Above the critical value of $v_c \approx 0.7$ the performance increases suddenly at least for sensors covering big areas.



(a) $r \in [4, 30]$



(b) $r \in [4, 15]$

Fig. 9.2: Performance over the sensor range.

Note the peak for $r \approx 8$ and the drop for $r \approx 10$.

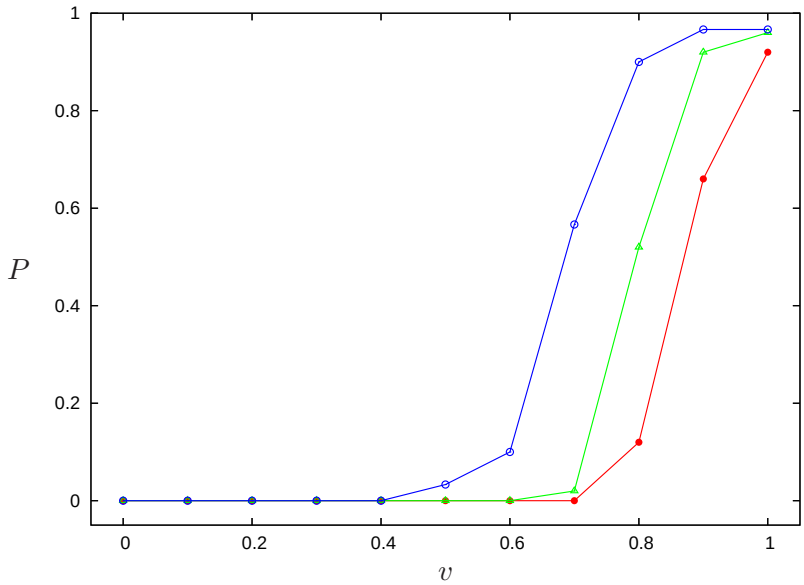


Fig. 9.3: Performance over the visibility ratio for three different number of sensors and hence different angle uncertainties ($S = 4$ in red, $S = 8$ in green, $S = 30$ in blue).

The angle uncertainty for $S = 4$ is $\Delta\phi = \frac{360^o v}{8}$, for $S = 8$: $\Delta\phi = \frac{360^o v}{16}$, and for $S = 30$: $\Delta\phi = \frac{360^o v}{60}$. One can see, that an increase in the visibility ratio v results in much bigger improvements in performance than increasing the angle measurement precision $\Delta\phi$.

9.2.3 Angle Uncertainty and Partial Visibility

In the next experiment the performance over the visibility ratio v is measured for three different angle measurement precisions $\Delta\phi = \{6^o, 22.5^o, 45^o\}$. The results in figure 9.3 are averaged over 100 runs. Compared to this results the curve in figure 8.2 was comparatively smooth. Here we have a clear jump in the performance around $v = 0.7$ depending on the angle uncertainty. Therefore the target-base-scenario cannot be solved with robots that have a visibility ratio of $v < 0.7$ and angle measurement precision of $\Delta\phi \geq 22.5^o$. That would include the robots of the current I-SWARM hardware specification ($v \approx 0.67$ and $\Delta\phi \approx 30^o$. But an increase in the visibility ratio to $v \approx 0.8$ could already result in feasibility and should be preferred to an increase in the angle measurement which would need to be quite intensive to reach the same performance.

9.3 Communication Intensity over Time

As discussed in section 8.3 while analyzing the chain formation scenario the ratio τ of transitioning robots has been measured over time for a swarm of 100 robots in the base-target-scenario. See figure 9.4 for the results that are averaged over ten sample runs. In comparison to section 8.3 one can notice that here τ drops even below the range that has been realized as the typical noise of the chain formation scenario: $0.1 < \mu < 0.2$. Since the robots in the tree barely move the risk of loosing the connection to a neighbor is less than the risk for robots in fast moving chains in the moment when they approach the borders of the arena or other chains.
Another way of measuring the communication intensity is to check the precondition for communication. This is that two robots need to be within the sensor range of each other to communicate. The number of robots μ that are within the sensor range averaged per robot is measured in figure 9.5. With increasing number of complete chains and with increasing sizes of trees μ increases about linearly and stays constant and saturated after almost all robots are part of the tree.

9.4 Density

In the following we will investigate the role of the density in the base-target-scenario at first isolated and later together with the sensor range. All following experiments start with the same initial state: All robots are equally distributed over the whole arena at the beginning.
At first we will investigate the influence of the density alone by varying the size of the arena with a fixed number of robots $N = 100$. One would await worst performance for little densities because the robots need to find each other by random walk and if the arena is too big, the robots will very seldom meet each other. For bigger densities the performance will increase to its best value for an optimal density and for even bigger densities we await bad performance because the robots are forced to operate in a too small arena that reduces their moving abilities and increases useless pacing around.

Figure 9.6 shows the results that are averaged over 100 runs. For densities close to zero the performance is as expected very bad. For increasing densities we notice a jump in the performance caused by the fact that the area of the arena decreases quadratic with increasing density. After having

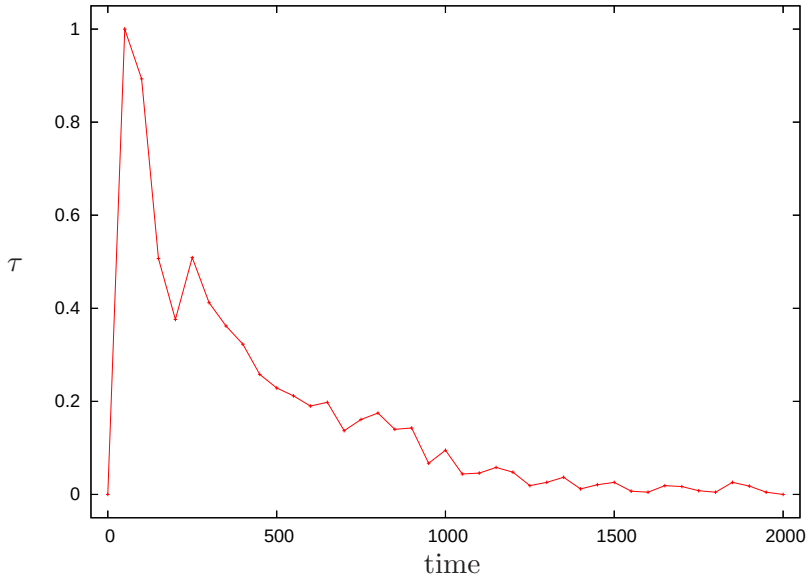


Fig. 9.4: Ratio of transitioning robots within 50 iterations over time

The ratio of robots that perform a transition within 50 iterations (bins of size 50) is measured over 2000 iterations. The results presented are averaged over ten runs.

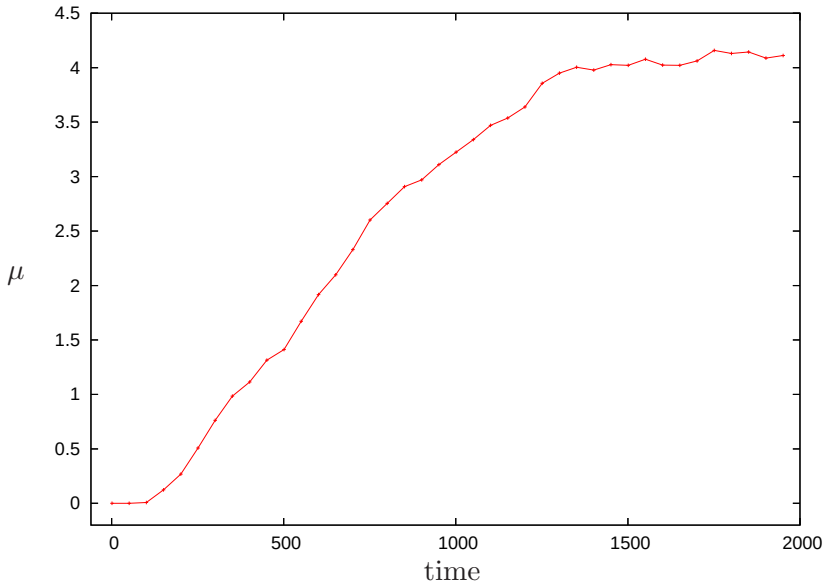


Fig. 9.5: Average number of robots that are within communication range per robot and iteration over time.

reached some optimum (at $\rho \approx 0.0125$) the performance suddenly drops at $\rho = 0.02$. From $\rho \approx 0.025$ on there seems to be a linear dependency between the density and the performance. This diagram corresponds to our general expectations.

Now we want to investigate the interplay between the density and the sensor range. That is focusing on the question: What is the optimal sensor range for a given density? In a first experiment the performance was measured linearly over the density against the sensor range on the intervals $0.0011 < \rho < 0.05$ and $4 < r < 30$ (see figure 9.7 for the results). We notice that wide sensor ranges are good only for small densities while small sensor ranges perform always very good. A clear edge in the range of $10 < r < 12.5$ and $0.028 < \rho < 0.045$ and along a line from $(\rho = 0.028, r = 10)$ to $(\rho = 0.005, r = 30)$ can also be noticed. This edge is correlated to the border case of equation 9.1 that is shown by a black curve. In the case of the area below this curve there is more space available per robot than it is needed in the tree. For the area above the curve the opposite is true, there is less space available than the robots need in the

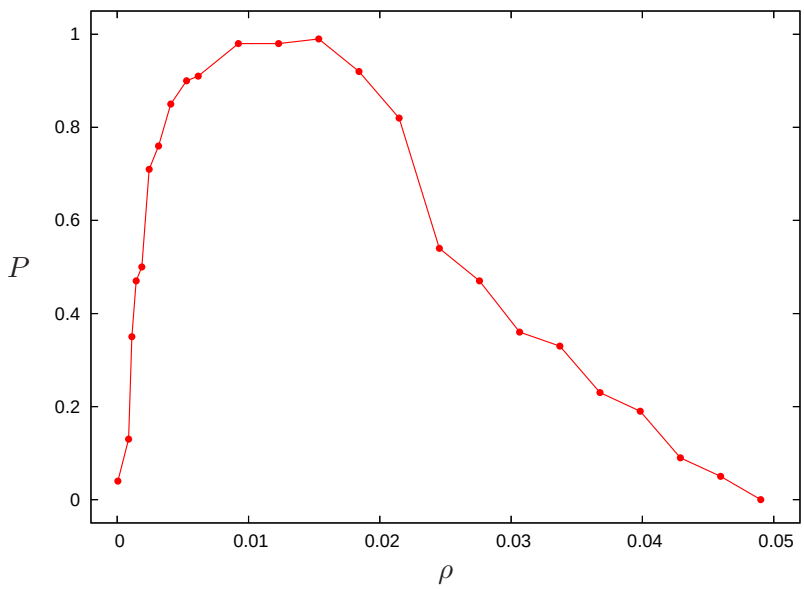


Fig. 9.6: Performance over the density.

The steep slope for $0 < \rho < 0.03$ is due to the fact that the area of the arena decreases quadratic with increasing density ρ. As one can see, there is an optimum in the range of $0.009 < \rho < 0.017$. For $0.025 < \rho < 0.05$ there seems to be a linear dependency between the density and the performance.

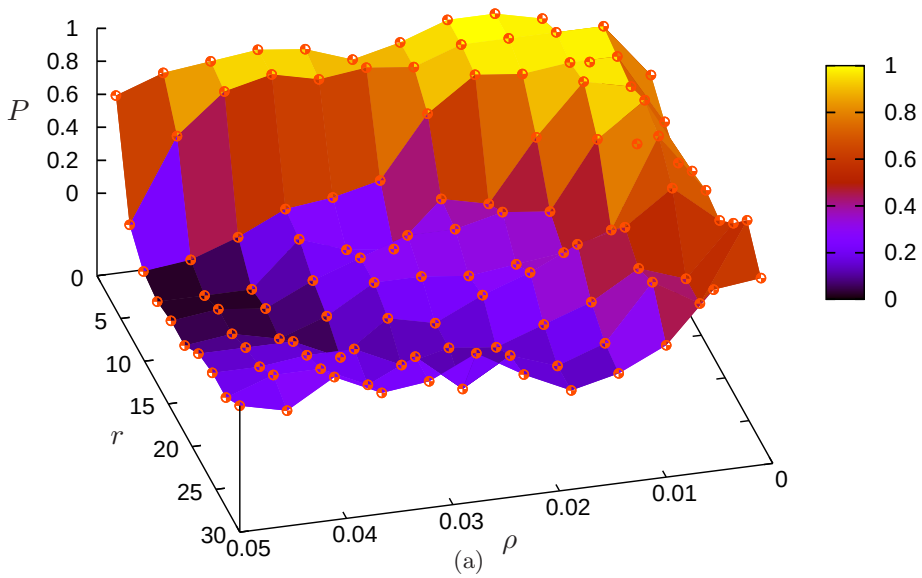
tree. In this area we notice a region of worst performance along the red line defined by $r(\rho) = -429r + 36.4$. For a given density ρ the performance of smaller or bigger sensor ranges is better than the performance of the sensor range at the red line.

In the following we want to investigate the swarm behavior in the base-target-scenario for small densities. Figure 9.8 shows the results of a scan that is logarithmic over the density in the range of $1 \cdot 10^{-4} < \rho < 5 \cdot 10^{-3}$ and linear over the sensor range in the range of $4 < r < 30$. For too small densities the performance is bad and gets better for bigger densities. Big sensor ranges perform better than small sensor ranges in the range of smaller densities. We also notice again that the performance is directly correlated to the border case of the area available per robot indicated again by the black line. Furthermore a peak for $r \approx 8$ and $\rho > 0.001$ is clearly developed. This region is investigated in the next diagram.

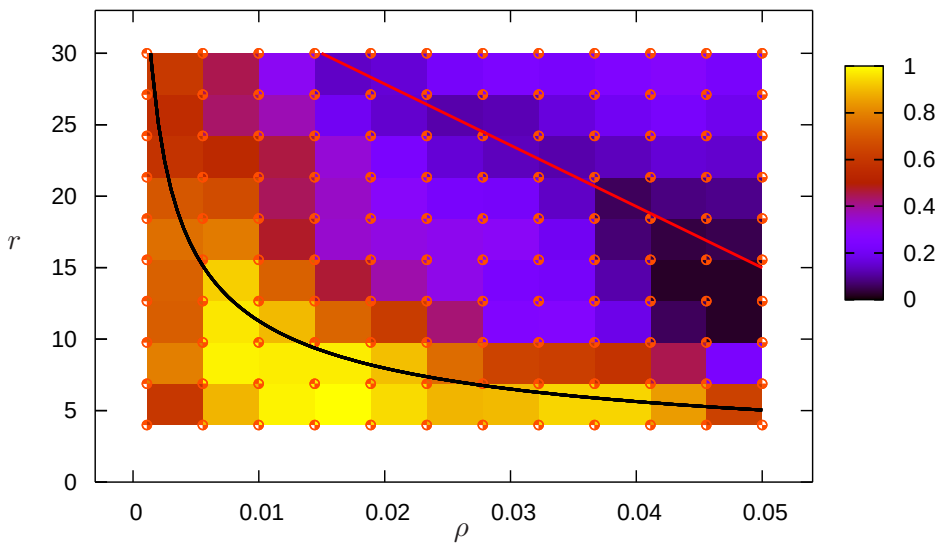
See figure 9.9 that focuses on the peak for $r \approx 8$ that can clearly be seen at two positions: $\rho < 0.004$ and $\rho > 0.015$. This phenomenon cannot be explained yet and might be the interplay of several parameters. See also section 9.2.2 for a short discussion of the same problem.

9.5 Y-Trees and X-Trees

In chapter 6 we introduced y-trees and x-trees and noticed that y-trees cover more area than x-trees in scenarios of 100 or more chains. Sine the computations of the complete simulation (in contrast to the simplified simulation used in chapter 6) are very intensive and increase quadratic in the number of robots we are not able to simulate 100 chains. With a chain length of five we would have to simulate 500 robots, which cannot be done yet with a reasonable effort. Instead a comparatively small scenario with only 100 robots and 20 chains was simulated 100 times for both tree types. However, no difference could be determined neither in the probability of success nor in the time needed to connect the two objects. In conclusion we note that the number of robots and the distance of the objects needs to be bigger so that the y-trees display their better qualities.



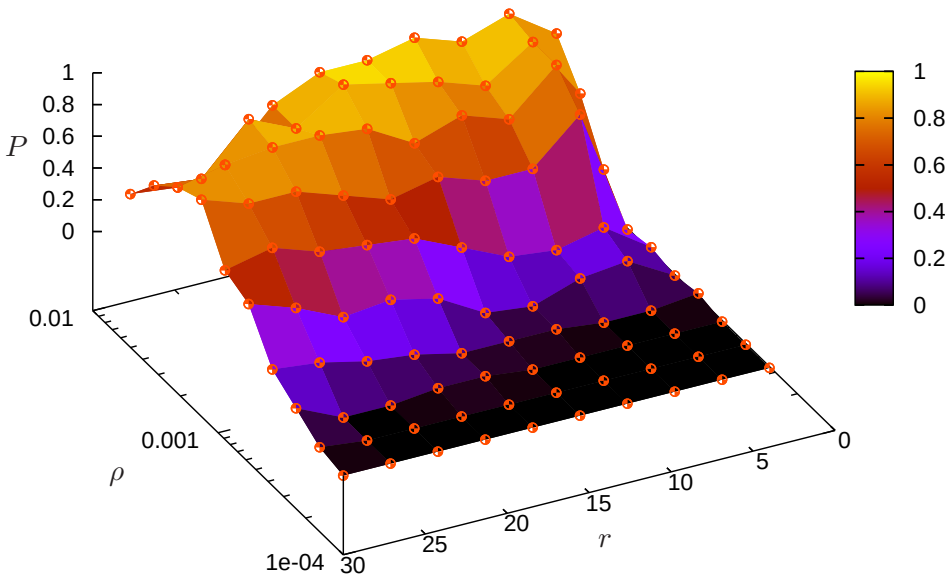
(a)



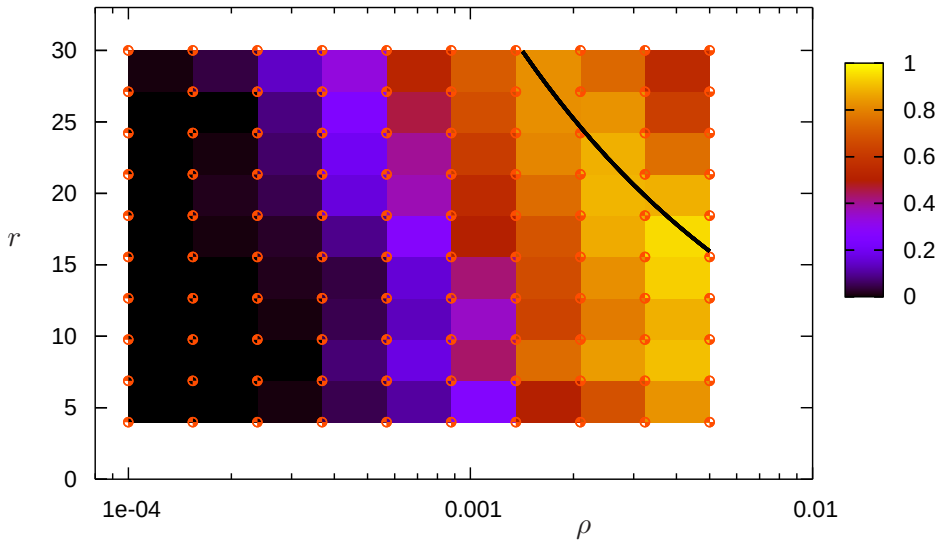
(b)

Fig. 9.7: Performance over the sensor range and the density (linear).

The black line shows the border case (space needed per robot in the tree is equal to the actual space available) defined by equation 9.1. The red line $r(\rho) = -429r + 36.4$ *shows the region of worst performance.*



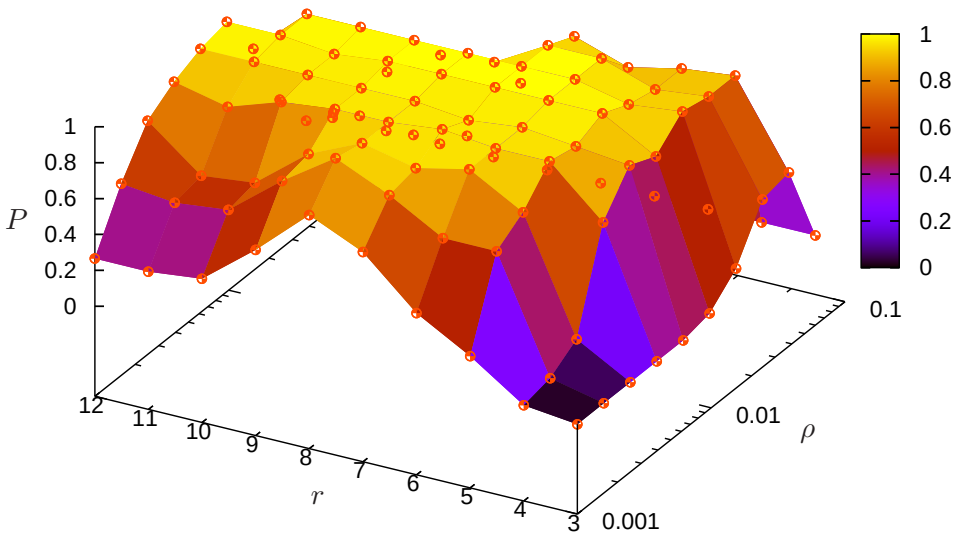
(a)



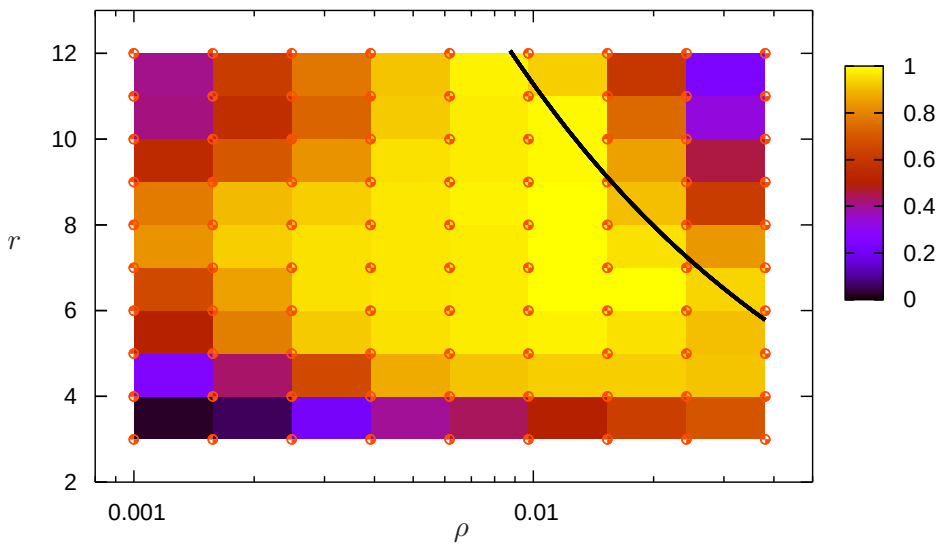
(b)

Fig. 9.8: Performance over the sensor range and small densities (logarithmic).

Notice the correlation between the border case of the area available per robot (black curve) and the performance, the advantage of big sensor ranges for small densities and the disadvantage for bigger densities, and the peak for $r \approx 8$ and $\rho > 0.001$.



(a)



(b)

Fig. 9.9: Performance over small sensor ranges and small densities (logarithmic).

A peak at $r \approx 8$ can clearly be seen for both $\rho < 0.004$ and $\rho > 0.015$.

9.6 Distance between Base and Target

As well as the density of robots the distance between the base and the target object is an environmental characteristic that highly influences the probability of success. If the distance is bigger than the longest possible tree (defined by the number of robots and sensor range), the probability will be zero. For technical reasons the simulation software will have problems to detect the successful connection of the two objects without the help of a tree if the distance is very small. For an increasing distance we would expect a decreasing probability of success.
We simulated the base-target-scenario with distances d_{bt} between the base and the target between 5mm and 45mm with 100 robots and measured the probability that the scenario is solved over 100 sample runs. The density was set to $\rho = 0.003\frac{robots}{mm^2}$ and the sensor range to $r = 10mm$. See figure 9.10 for the results. Because of the above mentioned technical reasons we get bad performance for very small distances (here for $d_{bt} < 9$). For a bit bigger but still small ($d_{bt} \approx r$) distances in the range of $9 < d_{bt} < 14$ we get the expected high success rates. At $d_{bt} \approx 14$ there is a drop and for bigger distances the probability is still relatively high at $P \approx 0.35$. For even bigger distances we would observe a success probability of $P = 0$ for distances that cannot be bridged by 100 robots with a sensor range of $10mm$.

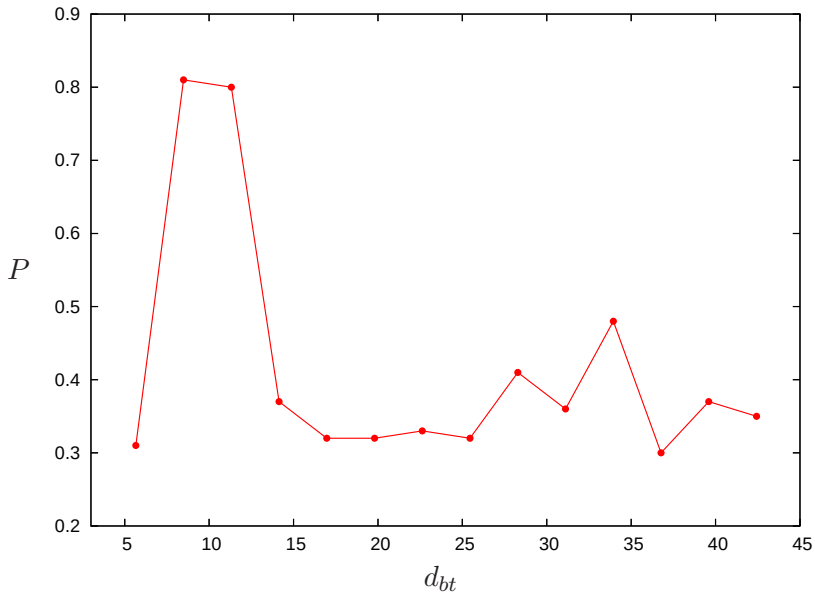


Fig. 9.10: Probability that the base-target-scenario is solved in dependence on the distance between base and target.

Small but not too small distances between base and target are optimal for the performance. For bigger distances d_{bt} the probability of a success stays constantly on a relatively high value of $P \approx 0.35$.

Chapter 10

Conclusion and Future Prospects

Books ought to have good endings.

Bilbo Baggins
JOHN RONALD REUEL TOLKIEN (1892-1973)
The Fellowship of the Ring (p. 359)

10.1 Conclusion

This work began with the development of a more realistic locomotion and sensor model, which has successfully be done and presented in the first part. In the second part the complex base-target-scenario has been designed, implemented and explained. This scenario is appropriate to show both the capabilities and the boundaries of this artificial swarm. In the final third part the simple chain formation scenario and the complex base-target-scenario have been analyzed concerning many different parameters.
Some important critical parameters have been found. For example the uncertainty in the angle measurement for the chain formation scenario of $\Delta\phi_c \approx 60^o$, the critical visibility ratio of $v_c \approx 0.7$ in the base-target-scenario, or the critical relation between the area per robot in the tree defined by the sensor range and the density (see equation 9.1).
Also the good performance of the swarm for very low densities in the chain formation scenario, if the robots are initially concentrated in one place, should be mentioned as well as the counterintuitive dependence on the sensor range of the swarm robots in the base-target-scenario.

These results show that the hardware specification of the I-SWARM robots is right on the edge to very good performance of the swarm and the results demonstrate that the general concept of very simple robots forming a cooperative swarm using self-organization is purposeful and most likely feasible.

10.2 Future Prospects

A first step of a follow-up project could be to investigate the suggested improvements of the control software stated in section 7.1.

Also a good working point is the interesting and counterintuitive connection between the performance in the base-target-scenario and the sensor range. The distance between two robots in the tree could be defined independently of the sensor range. Combined with more intensive investigations of the connection between density and sensor range this could help to find an optimal sensor range in dependence of the density. Additionally the idea of concentrating the robots initially at one spot should be followed up especially in the case of the base-target-scenario.

The third possibility could be to increase the complexity of the base-target-scenario, which could be combined with a more theoretical investigation. For example the scenario could be extended so that it could be seen as a Steiner tree problem [FHW92, Hau04, PS02]. That is finding a minimum-weight subgraph that connects certain nodes. A common variant is the Euclidean Steiner tree; here, the input is a set of points in space that are to be connected by a tree of minimum length. This corresponds almost directly to the base-target-scenario, which could also be extended by defining the metric not only as the length but additionally by gradients – for example the light intensity at this area of the arena, one could think of an uneven arena and so on. This would be a possibility to investigate the computational abilities of artificial swarms in connection with an NP-hard problem.

List of Figures

Bibliography

[Avr04] V. Avrutin. *Zum Verhalten dynamischer Systeme mit einer stückweise-glatten Systemfunktion*. PhD thesis, Universität Stuttgart, Stuttgart, 2004.

[FHW92] D. Richards F. Hwang and P. Winter. *The Steiner Tree Problem*. Elsevier, Amsterdam, 1992.

[Hak83a] H. Haken. *Advanced Synergetics*. Springer-Verlag, 1983.

[Hak83b] H. Haken. *Synergetics. An introduction*. Springer-Verlag, 1983.

[Hal01] T.C. Hales. The honeycomb conjecture. *Discrete & Computatuinal Geometry*, 25(1):1–22, 2001.

[Hau04] M. Hauptmann. *Approximation Complexity of Optimization Problems: Structural Foundations and Steiner Tree Problems*. PhD thesis, Rheinische Friedrich-Wilhelms-Universität Bonn, Bonn, 2004.

[IS06] I-SWARM. Intelligent small world autonomous robots for micromanipulation project, February 2006. http://www.i-swarm.org.

[ISW05] I-SWARM, periodic activity report no 2, December 2005. Internal paper.

[Koc05] A.T. Koch. Design and investigation of formations in robot swarms. Student research project, Universität Stuttgart, 2005.

[LaV98] S.M. LaValle. Rapidly-exploring random trees: A new tool for path planning. Technical report, Computer Science Dept., Iowa State University, 1998.

[ND04] S. Nouyan and M. Dorigo. Chain formation in a swarm of robots. Technical Report TR/IRIDIA/2004-18, IRIDIA, Université Libre de Bruxelles, March 2004.

[NP77] G. Nicolis and I. Prigogine. *Self-Organization in Non-Equilibrium Systems*. Wiley, New York, 1977.

[PS02] H. Prömel and A. Steger. *The Steiner Tree Problem. A Tour through Graphs, Alorithms, and Complexity*. Vieweg, Wiesbaden, 2002.

[SG99] W.M. Spears and D.F Gordon. Using artificial physics to control agents. In *IEEE International Conference on Information, Intelligence, and Systems*, 1999.

[TND05] V. Trianni, S. Nolfi, and M. Dorigo. Cooperative hole avoidance in a *swarm-bot*. *Robotics and Autonomous Systems*, 2005. to appear.

Index

Declaration

All the work contained within this thesis, except where otherwise acknowledged, was solely the effort of the author. At no stage was any collaboration entered into with any other party.

(Heiko Hamann)