Nico Winkelsträter

Virtual Pheromone for Swarm Robotic Navigation

FAKULTÄT FÜR
INFORMATIK

Intelligent Cooperative Systems
Computational Intelligence

Virtual Pheromone for Swarm Robotic Navigation

Bachelor Thesis

Nico Winkelsträter

December 5, 2019

Supervisor: Prof. Sanaz Mostaghim

Advisor: Sebastian Mai

Nico Winkelsträter: *Virtual Pheromone for Swarm Robotic Navigation*
Otto-von-Guericke Universität
Intelligent Cooperative Systems
Computational Intelligence
Magdeburg, 2019.

Abstract

This thesis aims to implement and evaluate a foraging behavior based on entirely virtual pheromones using a simulated environment with the Turtlebot 3 Burger and ROS. Implementation of the behavior is realized with multiple processes responsible for localization, pheromone map creation, pheromone sensing or navigation. As the pheromone is entirely virtual the sensing of the pheromone uses a digital pheromone map which is exchanged using wireless communications. The evaluation uses a best-case scenario in which the robots drive between home and resource uninterrupted as baseline comparison. Three different scenarios are tested which differ in environment, and number and quality of resources. It is concluded that, while the performance is worse than the baseline, the resource is reliably detected in the single resource scenario but in the multi resource scenario the best resource is not found reliably.

Contents

List of Figures

List of Tables

1 Introduction

The concept of stigmergy was first introduced by Grassé [10] in 1959 to explain how creatures that seem non-intelligent on their own can build complex structures such as termite hills when working together. Stigmergy (from ancient Greek stigma: mark, ergon: task/action) is the process in which actions of individuals change the environment in a way which can be interpreted by other individuals, thus obtaining information on the other individuals actions.

Stigmergy has also given inspiration to fields outside of biology and zoology. The Ant Colony Optimization algorithm which's first version was introduced in the 90s [7] is based on this very concept. It has also inspired the field of robotics and swarm robotics. For example, in the late 90s Kuwana and Shimoyama [14] used living moth antennae to attempt to guide a robot along a pheromone trail. Only few other researches used actual insect antennas and pheromone after that but a variety of pheromone emulation techniques and robots have been explored.

This thesis differs from most previous robotics applications of pheromones and also swarm robotics in general. The robots will not be implemented as reactive agents. Instead, a path planner is used which has knowledge of the environment Usually in swarm robotics reactive agents are used which have some sensors and actors and the sensory input directly affects the actors in every time step.

Also most previous pheromone emulations with robots used some form of physical representation of the pheromones with e.g. a projector, RFID Tags or even chemical substances. In this thesis the pheromone will be represented entirely virtual as a digital pheromone map.

1.1 Structure of this Thesis

At first an overview of similar research and robots used for this application is given. Then the details and structure of the software implementation of the behavior is presented. After that, the experiments that are conducted are described. The results from the experiments are displayed and evaluated in the chapter after that. Finally, a conclusion is drawn and possible improvements of the behavior and further research opportunities are presented.

1.2 Goals and Research Questions

Goal of the thesis is to implement a virtual pheromone map using the Turtlebot 3 Burger (see section 1.4.5) and ROS [20], simulated using the Gazebo simulation framework. Then, using this pheromone map, a foraging type behavior shall be implemented. Finally the performance of this pheromone based behavior shall be compared to a baseline behavior which uses only path planning and no pheromones.

Research Questions

- How does the pheromone based behavior perform in comparison with a best-case/baseline scenario?
- How long do the robots need to find the resource?
- Do the robots create a pheromone path and does it converge?

1.3 Previous Use of Stigmergy in Robotics

Getting inspiration from nature to create new technologies is quite common, and robotics is not an exception. Ant Colony Optimization is an example of a method that was inspired by nature. In this case by the stigmergic behavior of ants through the use of pheromones. Consequently, also the field of robotics has researched the feasibility of stigmergy as a tool for navigating robots. In this section the different types of stigmergic information propagation, specifically those that have been used to emulate pheromones similar of those used by ants, are listed. There are three categories of types of pheromone emulation.

1. Chemicals
2. Virtual Chemicals
3. Beacons / Digital Markers

Chemical emulation uses physical substances like gases or liquids which are then detected by sensors mounted on the robot. This type of emulation has the advantage that it closely resembles the properties of actual pheromone, like diffusion and evaporation. The obvious disadvantage is the technical complexity and cost of releasing and sensing these types of markers.

One example of the use of chemical emulation is by Purnamadjaja and Russell [19]. In this paper a leader-follower type behavior was implemented with the use of a gas sensor which can detect the presence of methylated spirits. The follower robots only have one of these sensors and decide their next action purely by the concentration of the chemical marker at their current position compared to their last position. This means they cannot follow a trail of pheromone but can only localize the source of a diffusing chemical.

Fujisawa et al. [9] also use chemical pheromone emulation but use ethanol instead of methylated spirits. In comparison to Purnamadjaja and Russell [19] a foraging behavior instead of a following behavior is implemented. This means that instead of just one sensor the robots have two ethanol sensors built in, in order to be able to follow a trail of the chemical. The foraging task implemented is very similar to the one implemented in this thesis. At first the robots explore the environment until the resource is found and will then return to the nest while releasing a trail of ethanol. The main difference of the implementation to this thesis is the use of chemicals and the navigation based on a reactive agent, meaning the robots do not use path planning but react directly to the sensor input.

Beacons or digital markers do not use chemical substances but rather some form of digital devices with which the robots interact in order to receive or place the stigmergic information.

Herianto et al. [12] and Khaliq and Saffiotti [13] use NFC Tags arranged in a grid from which the robots read and write the pheromone information to.

Khaliq and Saffiotti [13] use regularly spaced NFC tags to encode what they call *potential fields*. There is a goal potential field and an obstacle potential field which encode distances to obstacles and distances the goal respectively.

With this approach the environment information is permanent which is used to first store information with many small robots so that afterwards a bigger robot can perform a task using this information.

Herianto et al. [12] also use NFC tags but save information in the tags that is more similar to chemical pheromone. They describe an algorithm which allows the robots to update the NFC tags to simulate diffusion if they know the data of the surrounding tags.

Aznar et al. [6] use the robots themselves, in this case multicopters, as medium for stigmergic information in order to implement autonomous exploration of an unknown environment. When the copters stray to far from any beacon it will switch into beacon mode, meaning they will land at their current position therefore marking it as explored. Other copters will now fly further as they are in range of a beacon in that area.

Payton et al. [18] propose a scenario in which an unknown environment shall be explored and and resources be located. The robot designed to achieve this task has eight IR transceivers with which the robots can detect the distances to other robots. Depending on the distance to other robots in the proximity the robot is either attracted, repulsed or experiences no force. This causes the robots to spread out with roughly equal distances between the robots. Once a robot has located a resource it sends a message to all directions. When a robot receives such a message it will relay it with an increased hop count so that when a robot receives a resource message from multiple directions the shorter path can be deduced by choosing the message with the lower hop count.

Another approach is to forgo the physical domain entirely and only represent the pheromones virtually, as is done in this thesis.

Font Llenas et al. [8] uses a projector which projects a map representing the density of pheromone onto the floor and the small robots are equipped with light sensors to detect the pheromone. Similarly to previously named approaches the robots at first explore the environment with a random walk. Once a resource has been found the robot will turn back to the home location and lay down a pheromone trail. The consistency of the pheromone trail depends on the quality of the resource the robot just collected with the intention of a higher reinforcement of trails leading to the better resource.

Na et al. [16] also use a digital representation of the pheromones but display them on a LCD screen on which the robots drive around instead of using a

projector. These robots also use light sensors to detect the pheromone and the task for the robots is to find a circle of pheromone. Performance is measured by the cohesion of the robots while changing the diffusion, evaporation, injection and advection parameters of the pheromone.

Susnea et al. [23] use an entirely virtual pheromone, no projectors or screens are used. The robots use odometric information from encoders on their motors to localize themselves in order to instruct a pheromone server to either place pheromone onto the map or request pheromone information from the server. An answer to the request consists of the pheromone values of two points in front of the robot. One more to the left and the other more to the right. Using a fuzzy controller the velocity for the left and the right wheel of the robot are determined in a way to follow the pheromone trail.

1.4 Robots Used in Swarm Robotics

This section examines the robots used in the research named above and compares their features and sensors to the Turtlebot3 Burger, which is the robot used in this thesis.

1.4.1 ARGOS-01

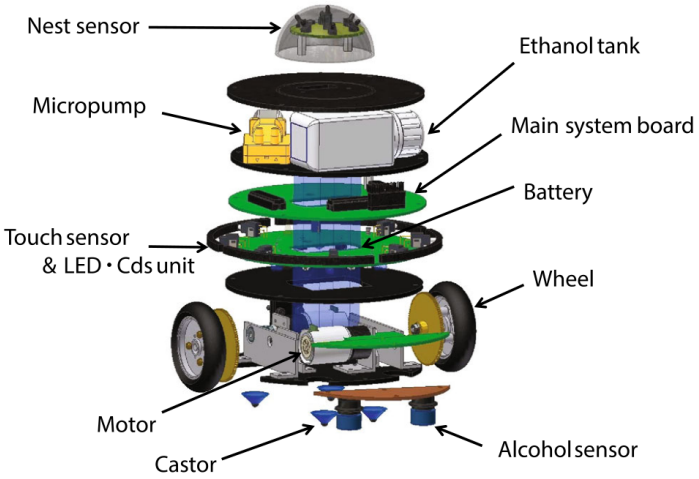


Figure 1.1: The ARGOS-01 robots as used by Fujisawa et al. [9]

The ARGOS-01 was developed by Fujisawa et al. [9] in order to implement a foraging behavior. These robots include blue LED's which can be detected by other robots. It uses infrared photo-transistors to detect the location of the nest and cadmium sulfide units to detect the resource. Fujisawa et al. [9] chose to emulate chemical pheromones by the use of ethanol as it has the properties of evaporation and diffusion. To facilitate the detection and dispersal of pheromones the robot includes two ethanol sensor at the bottom and an ethanol tank accompanied with a small pump. Other than the ethanol pump and sensor the robots have no way to communicate with each other or any central entity. Control of the robot is handled by three PSoCs (Programmable System on Chip) which communicate witch each other via a serial link Fujisawa et al. [9].

1.4.2 E-Puck

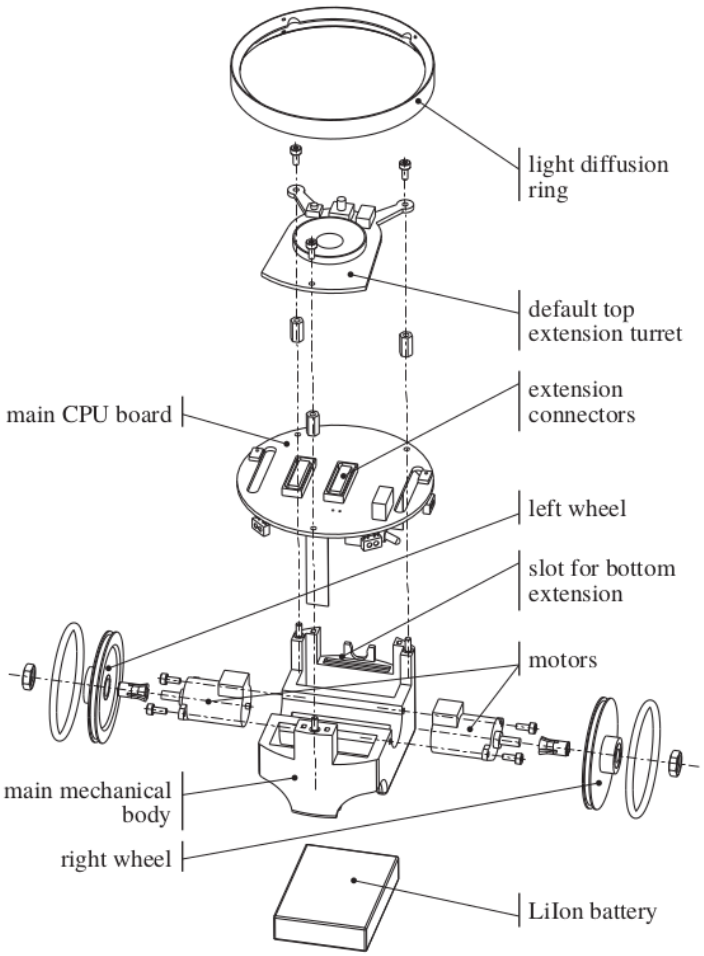


Figure 1.2: Explosion Drawing of the E-Puck as described in [15]

The E-Puck is a robot designed for educational purposes with a focus on low cost, open information and extensibility. It's a differential drive design using stepper motors with a PIC microcontroller as the main processor which handles communication with the sensors and actors. As it includes extension connectors it is relatively easy to deploy custom sensors to the robot. This has been used by Herianto et al. [12] and Khaliq and Saffiotti [13] to extend the E-Puck with RFID/NFC functionality in order to read and write to an RFID grid on the floor. The default configuration of the E-Puck includes wireless

communication in the form of a bluetooth module, but neither Herianto et al. [12] nor Khaliq and Saffiotti [13] make use of it.

1.4.3 Colias Microbot

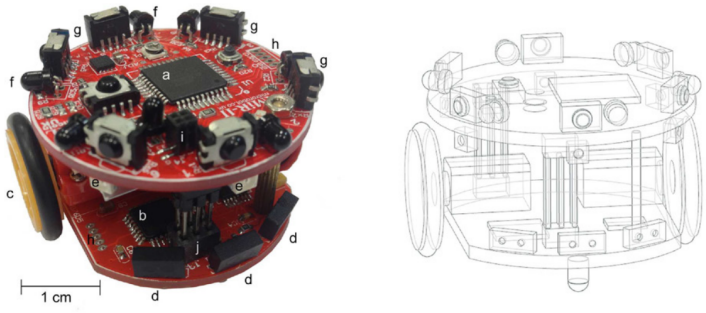


Figure 1.3: Image of the Colias Microbot as described in [5]

The Colias Microbot was also designed for educational purposes but especially with swarm robotics in mind. Therefore there was a focus on low cost (estimated cost is 25$) and small size in order to make it easy to deploy many of them. Two Atmel microprocessor work in conjunction to process sensor information and control the motors. In the default configuration the Colias only uses IR transmitters and receivers intended for obstacle and robot detection. This makes it very similar to the robot needed for the exploration task proposed by Payton et al. [18]. An actual example of the use of the Colias is given by Na et al. [16] who extended the Colias by two light intensity sensors which are used to detect the pheromone. The pheromone is displayed on a LCD screen on which the robots drive on.

1.4.4 Kilobot

Figure 1.4: Image of the Kilobot as described in [22]

The Kilobot was also designed with swarm robotics in mind which shows in its very small size compared to all other robots mentioned in this section. With a part cost of about 14$ it is one of the most affordable robots in this selection. Other than its small size its unconventional locomotion strategy sticks out. Instead of the classical two wheel differential drive system it uses vibration motors to propel itself via the slip-stick principle [25]. Inter-Robot communication is possible by a IR transmitter receiver pair pointed at the ground which results in a communication radius of about 10cm with a data rate of about 30kb/s. The intensity of the incoming signal can be used to determine distance to nearby robots. Additionaly the robot includes an ambient light sensor which was utilised by Font Llenas et al. [8] in conjunction with ARK (Augmented Reality for Kilobots [21]) as means of displaying the pheromone representation with a projector.

1.4.5 Turtlebot3

For this thesis the Turtlebot3 Burger was used.

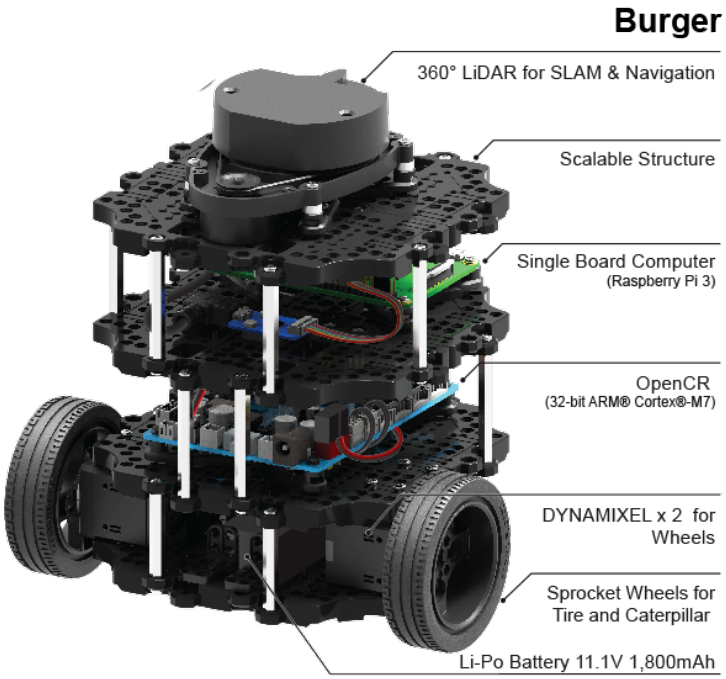


Figure 1.5: Overview of the Turtlebot3 Burger [2]

Most notably it includes a 360° Laser scanner and Raspberry Pi single board computer, which enables the Turtlebot3 to perform Simultaneous Localization a Mapping (SLAM) and other Point Cloud based techniques for localization like Adaptive Monte Carlo Localization (AMCL). All other robots named in this section are based on microcontrollers and therefore lack the computing power needed to perform such tasks, also none of those robots include a laser scanner. With a footprint of 138 mm by 178 mm the Turtlebot3 Burger is a relatively small robot which is based on a differential drive motor assembly, allowing it to rotate in place. The used Dynamixel Motors include rotary encoders which supplement the odometry information from the on-board gyroscope, accelerometer and laser scanner. With a price point of about 550$ and the ability to run the Robot Operating System (ROS) it is a good candidate for the study of small scale driving robots in swarm intelligence in scenarios where laser scanner assisted localization is wanted.

The Turtlebot project also provides a complete simulation environment based on Gazebo [17] which makes testing, especially with more than just a few robots, easier.

1.5 ROS

The Robot Operating System (ROS) is a meta operating system or robotics middleware. ROS "provides a structured communications layer above the host operating systems of a heterogenous compute cluster" [20]. " ROS is designed to be modular at a fine-grained scale" [20], meaning that every node (process) performs a single task and communicates with other nodes using messages, services or actions which are defined in a language agnostic way. This makes it easy to exchange parts of the software or add additional nodes, even at runtime. All the communications between the nodes can easily be recorded and played back at a later time which is very convenient for evaluating experiments.

The Turtlebot project already provides a variety of ROS packages for the Turtlebot3 which makes it easy to extend the robots capabilities with further packages. Additionally, the modularity of ROS makes code very reusable and therefore there are a lot of software packages available which can be made to work with the Turtlebot. For example there are various SLAM implementations, a navigation stack, a simulation environment and localization nodes.

2 Implementation

In this chapter the implementation of the behavior using ROS is explained. First the implemented behavior, then the way the pheromone maps are created and lastly all used ROS-nodes and their functions is described.

2.1 Behavior

A simple foraging type behavior is implemented by using a state machine. For the transition graph of the state machine refer to fig. 2.1

At first the robot randomly walks around the arena using a Lévy Walk. Once the robots detects resources it drives to the home location using a path planner. While doing so it emits a trail of pheromones. When it reaches the home locations it switches back to randomly walking. If the robot detects pheromone while randomly walking it switches to following the trail of pheromone until it has found a resource at which point it also starts driving to the home location while emitting pheromones.

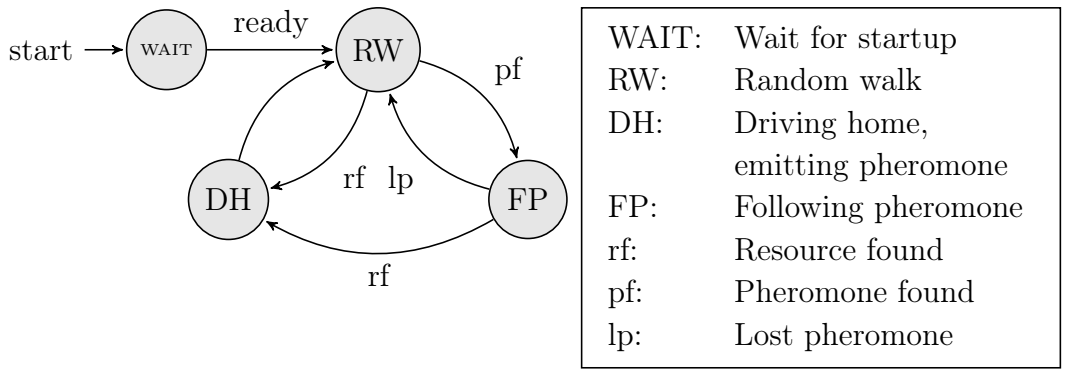


Figure 2.1: Transition graph of the controller node FSM.

2.2 Creation of Pheromone Maps

As described in section 2.3, the robots use the *amcl* node to localize themselves in the map provided by the map server. This localization information can be used to place virtual pheromones onto a new map which will contain only pheromone information. In order for this information to be useful, the transformation between the pheromone map and the environment map has to be known. The simplest way to do this, is to create the pheromone map with the same parameters (size, resolution and origin) as the environment map. Doing this ensures there is a direct mapping between grid cells in the pheromone map and the environment map.

There are two ways to create a complete pheromone map. A map is complete if it contains pheromone information from all robots. Either the robots exchange pheromone or localization information with each other and every robot builds a complete map from that Information, or the robots communicate with a central node responsible for combining all the information and redistributing it to all robots. These two concepts are illustrated in fig. 2.2. The one with the fully connected graph will be called 'decentral', the one with the star configuration will be called 'central'.

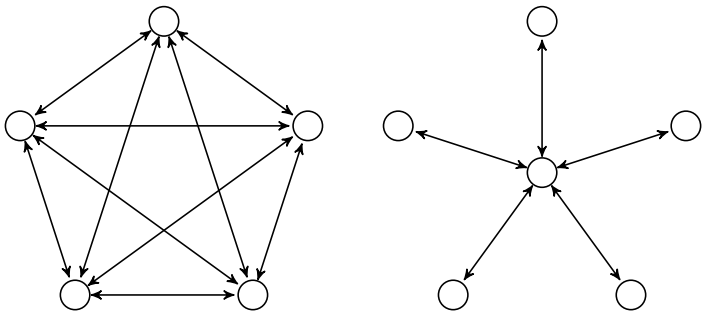
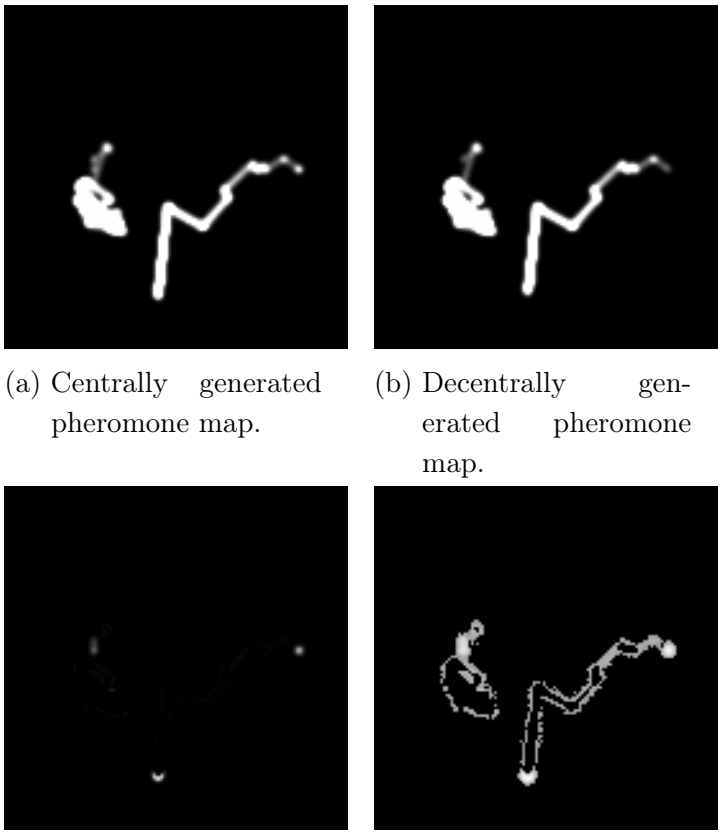


Figure 2.2: Possible configurations for creating pheromone maps. Either fully connected or star configuration.

In the decentral option every robot needs to send and receive from every other robot. This means that the network configuration is a fully connected graph. A fully connected graph has $\frac{n(n-1)}{2}$ edges. As the communication is bidirectional there are $n(n-1) = \mathcal{O}(n^2)$ communication paths.
The decentral option requires one bidirectional communication path for each new robot added. One direction is sending position or pheromone information

to the central node, the other direction is receiving the complete pheromone map. The amount of communications paths is therefore $2n = \mathcal{O}(n)$.
The bandwidth cost of the decentral option increases very rapidly while bandwidth cost of the central option increases linearly with the amount of robots. Also, the central option only requires an additional computation node as extra cost, which could be eliminated by simply choosing an existing robot as central node. This are the main reasons that the decentral version was chosen in this work. However, there are some drawbacks. Mainly, the decentral option has less resemblance to the way a natural swarm communicates and is more susceptible to communication errors. When one communication path fails the information from one robot is missing for all other robots. If a single communication path fails in the decentral option, pheromone information is only missing from robot and could perhaps be transmitted by another robot, making it more robust. Due to the centralized network topology used in this thesis (Wireless LAN) the possible error tolerance of the decentral implementation are not important.



(a) Centrally generated pheromone map.

(b) Decentrally generated pheromone map.

(c) Difference between Central and Decentral map.

(d) Difference between Central and Decentral map with enhanced contrast.

Figure 2.3: Comparison of pheromone map generation techniques.

A comparison of the resulting pheromone maps can be seen in fig. 2.3. For this comparison both methods were active simultaneously while two robots were driving in a random pattern. The difference of the maps is very small and is only present at the edges and start- and endpoints of the pheromone track. During the comparison pheromone evaporation was being simulated. Thus the difference between the two maps can entirely be explained by fact that the nodes creating the maps were not synchronized and the output might be out of sync by a single step. So on one map the pheromone has undergone one more step of evaporation and therefore there is less pheromone.

The pheromone evaporates and diffuses over time (like in [8]). This process is implemented as a 2d convolution and with evaporation rate ε, diffusion rate γ and time since the last update δt the convolution kernel K is the following:

$$K = \begin{bmatrix} 0 & \gamma \cdot \delta t & 0 \\ \gamma \cdot \delta t & \varepsilon \cdot \delta t & \gamma \cdot \delta t \\ 0 & \gamma \cdot \delta t & 0 \end{bmatrix}$$

And with the current pheromone map P the update map P' is calculated as follows:

$$P' = P * K$$

2.3 Node Architecture

This section details the software implementation of the behavior. It is explained which ROS-nodes are used, what jobs they have and what other nodes they communicate with. First, all the nodes are named and an overview of the communication structure is given. After that, each node is described in detail.

A single robot requires a variety of nodes, accomplishing different tasks, in order to exhibit the desired behavior. The following nodes are used in this implementation:

- Pheromone sensor
- Pheromone follower
- Resource sensor
- Random walker
- Pheromone map publisher
- Central pheromone map
- Controller Node
- Map Server (from ROS)
- AMCL (from ROS)

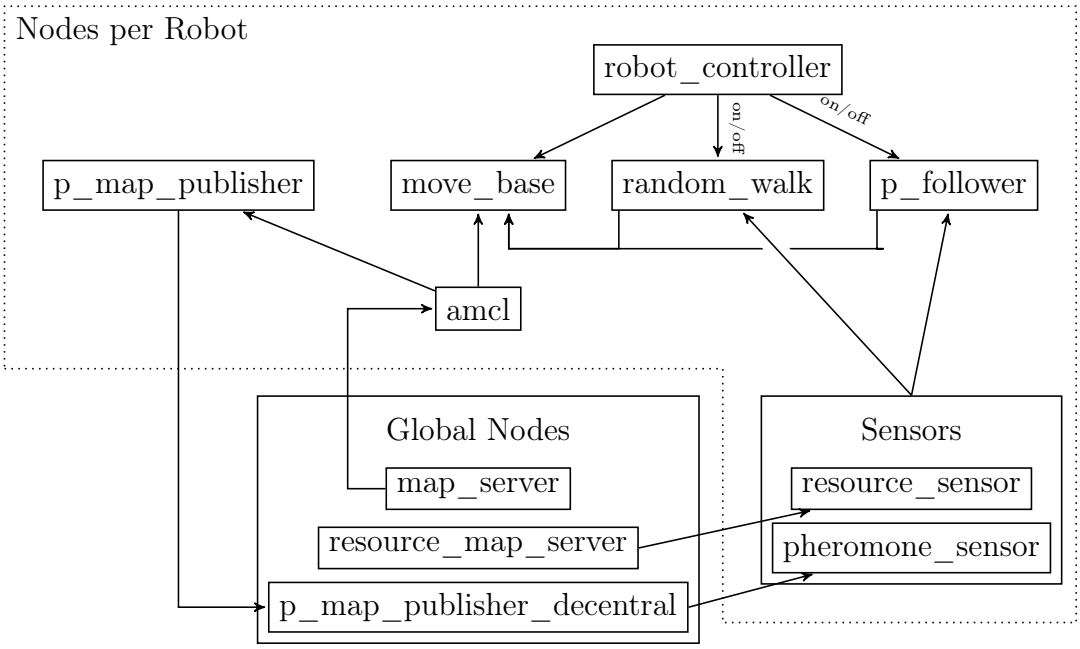


Figure 2.4: Graph of major ROS-Nodes responsible for the robot behavior.

The fig. 2.4 shows the communications paths between the nodes. The robot_controller nodes implements a finite state machine [1] which's transition graph is depicted in fig. 2.1. Depending on the current state exactly one of *random_walk* and *pheromone_follower* is enabled. The nodes *random_walk* and *pheromone_follower* are enabled and disabled using ROS-Actions. The sensor nodes *resource_sensor* and *pheromone_sensor* receive information from the global resource and pheromone map nodes and send the data for the current robot position to the *random_walk* and *pheromone_follower* nodes. This sensor information is used by the nodes to determine whether the action is finished, i.e. when a resource is reached, or in case of *pheromone_follower* to determine the direction to move to. Both *random_walk* and *pheromone_follower* send navigation goals to *move_base*. The *robot_controller* also sends a navigation goal to *move_base* when it reached the driving home state. Finally, the *p_map_publisher* uses the position information from *amcl* to construct a robot-local pheromone map which is sent to the global *p_map_publisher* to construct a global pheromone map from all sub maps.

[1] Implemented using smach[1]

2.3.1 Pheromone Nodes

This section describes the pheromone sensor and the pheromone publisher which are nodes nodes that are processing pheromone information.

pheromone_sensor The task of the pheromone sensor node is to supply information about the current pheromone situation in the robot's proximity. It uses the current position estimate of the robot and the most current available pheromone map to measure the amount of pheromone present in front of the robot. A circular section in front of the robot is divided into sectors of the same size and all pheromone present in those sectors is summed up so that a measure of pheromone is assigned to each sector. For this thesis a sector amount of five with a total viewing angle of 180° and a radius of $0.5m$ was chosen.

Let n be the number of sectors, ϕ_r the heading of the robot, ϕ the viewing angle of the robot and r the radius of the pheromone sensor. Then the starting angles of the sensor sectors are $\{\phi_{l_s} = \phi_r - \frac{\phi}{2} + l \cdot \frac{\phi}{n} | l \in [0..n-1]\}$ and the ending angles are $\{\phi_{l_e} = \phi_r - \frac{\phi}{2} + (l+1) \cdot \frac{\phi}{n} | l \in [0..n-1]\}$.

The pheromone map is a matrix $P \in \{0..100\}^{n \times m}$ in which each entry corresponds to a pheromone value in a given cell. The pheromone value corresponding to point $p \in \mathbb{R}^2$ is $P_{i,j}$ where $i = \left\lfloor \frac{p_x}{c} \right\rfloor$,$j = \left\lfloor \frac{p_y}{c} \right\rfloor$ and c is the side length of the cells. For each sector s_l a mask matrix $M_{s_l} \in \{0,1\}^{n \times m}$ is constructed. Let i_r, j_r be the cell indices for the current robot position p_r.

$$D(i,j) = \begin{cases} 1, & \text{if } c|(i,j) - (i_r, j_r)| \leq r \\ 0, & \text{otherwise} \end{cases} \tag{2.1}$$

$$A_l(i,j) = \begin{cases} 1, & \text{if } \phi_{l_s} \leq atan2(i - i_r, j - j_2) \leq \phi_{l_e} \\ 0, & \text{otherwise} \end{cases} \tag{2.2}$$

The function $D(i,j)$ determines whether a cell is within distance r of the robot. The function $A_l(i,j)$ determines whether a cell is within the starting and ending angle of sector l relative to the robot cell p_r.

The mask matrix can now be constructed,

$$M_{s_l} = (D(i,j) \cdot A_l(i,j) | i < m, j < n) \tag{2.3}$$

and the pheromone value P_l for each sector can be calculated.[2]

$$P_l = \sum_{i=0}^{n} \sum_{j=0}^{m} p_{ij} | p_{ij} \in M_{s_l} \circ P \tag{2.4}$$

A visual representation of the mask matrices for $r = 0.5m, \phi = 180°$ and $n = 5$ can be seen in fig. 2.5.

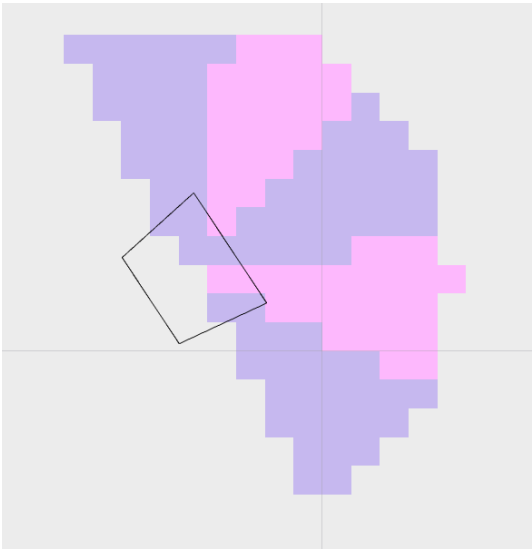


Figure 2.5: Illustration of the sectors of the pheromone sensor.

p_map_publisher It is responsible for taking the position estimate from the *amcl* node and placing a pheromone 'blob' at the according position on a separate pheromone map. The pheromone blob is in the shape of a 2D discrete Gaussian distribution with a σ of 1.2 and with a size of 11×11 grid cells. The strength of the pheromone blob, which corresponds to integer values being saved in the pheromone map, is influenced by the quality of the resource previously observed. As seen in fig. 2.4, this is done by each robot separately. This means that it has a local pheromone map only containing

[2]The $\circ$ operation used in eq. (2.4) is the entrywise matrix product, also known as the Hadamard Product.

pheromone information of this single robot. To compile all of the local maps into a complete map, a mechanism to exchange this information needs to be in place. In this case a central node is used to perform this operation.

The initial pheromone blob $B_i \in \mathbb{R}^{11\times 11}$ is at first multiplied with the current resource quality $q \in [0..100] : B = qB_i$. Then it is padded to match the size of the pheromone map $P \in \{0..100\}^{n\times m}$. With the current robot position q and the cell size c the padding is calculated.

$$\begin{aligned} p_l &= \left\lfloor \frac{q_x}{c} \right\rfloor - \left\lfloor \frac{11}{2} \right\rfloor \\ p_t &= \left\lfloor \frac{q_y}{c} \right\rfloor - \left\lfloor \frac{11}{2} \right\rfloor \end{aligned}$$

$$B_p = \begin{bmatrix} 0_{0,0} & \dots & 0_{0,p_l+1} & \dots & 0_{0,m-1} \\ \vdots & \ddots & & & \\ 0_{p_t+1,0} & \dots & B & \dots & 0_{p_t+1,m-1} \\ \vdots & & & \ddots & \\ 0_{n-1,0} & \dots & & \dots & 0_{n-1,m-1} \end{bmatrix}$$

The pheromone map P can now be update to $P' = \text{clamp}_i(P + B_p, 0, 100)$ The function clamp_i ensures that the pheromone map only contains values $\in \{0..100\}$.

$$\text{clamp}_i(x, l, u) = \begin{cases} l, & \text{if } x < l \\ u, & \text{if } x > u \\ \lfloor x \rfloor, & \text{otherwise} \end{cases} \tag{2.5}$$

central_p_map All of the robot-local pheromone maps are sent to this node. As all of the maps share the same size, resolution and origin, the combining operation is as simple as adding the values for each grid cell together while making sure to not exceed the maximum value for a given grid cell. So every time a new map P_r is received from one of the robots the pheromone map P can be updated to $P' = \text{clamp}_i(P + P_r, 0, 100)$

2.3.2 Behavior Nodes

This section describes all the nodes which directly influence the behavior of the robot.

robot_controller Controlling when which node is active is done by this node. It is implemented as a finite state machine, similar to the one used by Font Llenas et al. [8], though it omits the turning back state for simplicity. Therefore, it has three states: following pheromone, driving home and random walk. It starts in the random walk state in which the random walker node is activated. It remains in this state until either a resource or pheromone is found. If a resource is found, it transitions to the driving home state. If pheromone is found, it transitions to the following pheromone state. Should resources and pheromone be present at the same time the resource takes precedence and it transitions to the driving home state. The transition graph is shown in fig. 2.1.

random_walk Similarly Font Llenas et al. [8], the robots uses an isotropic random walk for exploration. The robot chooses a random direction and drives a distance, which is determined by a Lévy distribution, in that direction. Once it arrives it rotates to point to a random direction and the cycle repeats. This also uses ROS's navigation stack and therefore only a limited time is allocated to reach the goal in order to prevent the robot from getting stuck trying to reach an unreachable position.

p_follower The pheromone follower node uses the information provided by the pheromone sensor to follow a trail of pheromone. There are two different ways this was achieved. The first directly sets the speed of the motors, whereas the second one leverages the powerful navigation stack included in ROS.

The first, more simple approach sets the forward speed of the motors proportionally to how close the sector with the most pheromone is to the current direction the robot is facing and the rotational speed inversely so. In an example with four sectors the rightmost sector has the most pheromone of the four. The rightmost sector is far from the direction the robot is facing and therefore the forward speed will be low but the rotational speed will be high because the sector is far from center. This will cause the robot to turn right.

Being based on the already present navigation stack provided by ROS, the implementation of the second approach is actually quite simple. It chooses the sector with the most pheromone and orders the navigation stack to move the

robot a set distance in the direction of that sector. As the navigation goal sent could possibly be unreachable, e.g. in a wall, it is aborted after a set time to prevent the robot from getting stuck.

The first approach is closer the classic swarm intelligence concept of a reactive agent. However, in this approach the robots do not avoid each other and with multiple robots driving on the same path this could lead to the robots colliding and getting stuck. The second approach, while straying away from the classic reactive agent, has the advantage that the the ROS navigation has collision avoidance built in and is also able to detect dynamic obstacle detection built in. Thus with this approach the robots should be able to avoid obstacles as well as other robots.

resource_sensor This node simply takes the current position p_r of the robot and looks at the resource map and returns the value at that position. Using this, the robot can determine whether it has found a resource and of which quality it is.

3 Experiments

This chapter explains what experiments where conducted and how they were set up. All experiments were conducted in simulation. For every scenario, including baseline runs, 31 different simulation runs were conducted. In total three experiments were conducted, increasing in complexity. Starting with a single room and single resource, continuing with two rooms and a single resource and finishing with two rooms and two resources of differing quality.

3.1 Common Setup

All experiments follow the same principle, everything that does not differ between the experiments is explained in this section.

The Robots will be placed exactly at the home location which will cause the simulation engine to spread the robots apart as the robots are colliding with each other. Then, after a short delay, the robots will start moving.

Variables that differ between experiments include the arena, the location of the resource the initial position of the robots and the amount of time the simulation is run for.

3.2 Arena

The arena consists of one continuous, closed off area and it is has been made sure that there are no spots in the arena where all walls are more than 2.5 m away. This has been done to ensure proper localization as the laser scanner used by the turtlebot has a maximum range of 3.5 m [4]. A wall at a distance of 3.5 m would result in only a few or no measurements which do not allow

the detection of the wall orientation. Additionally, in order to avoid localisation errors, the arenas have been created to reduce ambiguities. For example rotational symmetry and similarities between regions of the map have been avoided, e.g. in the two room arena the two rooms have different shapes. The initial position of the robot is provided to the amcl node but as it takes some time for the pose estimation to become more accurate it was helpful to design the arenas this way. Otherwise amcl sometimes converged to the wrong location. This happens in the beginning when the certainty of the position estimate is still low.

Map creation

As the robots are not performing SLAM during the experiment, a prerecorded map of the arena is required. The dimensions of the arena are known, so a perfect map could be created with image processing software. But to keep more similarity to a setup where SLAM is used actively by the robots the map is also prerecorded using SLAM. This creates an imperfect map ensuring that during the experiments the robots will have to cope with slight location inaccuracies which is closer to reality.

The map is created by setting a single robot into the arena and starting the ROS nodes necessary for SLAM, in this case gmapping [3][11] is used, and teleoperation are started. Now the robot is driven around the arena until a complete map is created and then the map is saved[1] for use in the experiment.

Baseline Run

In order to be able to evaluate the performance a baseline run is performed for each scenario. During a baseline run the robots will not make use of any pheromone related features. They will continually drive between home location and resource using the path planner. Thus this emulates a best case scenario where all robots instantly find the resource and the pheromone path stays established forever. All other parameters like amount of robots and environment are kept the same. Because every scenario uses three robots the baseline runs also use three robots. This makes it easy to compare the

[1] Using the map_saver node from the map_server package

total collected resources and resource collection rate without any conversion calculation.

Scenario One

The first experiment is the simplest one and uses a single room as an arena with dimensions which allow it to be replicated in a typical room. A single resource is placed opposite to the home location of the robots. For this experiment three robots are used and the simulation is run for 1800s (30 minutes),

Figure 3.1: Map of single room arena created using SLAM gmapping.The red area marks the location where a resource is present. The red x marks the home location of the robots.

Scenario Two

The second experiment uses an arena comprised of two rooms connected by a corridor. Like in the first experiment three robots and a single resource have been used. The robots have their home location in the lower left of the lower room and the resource is placed in the upper right of the upper room, see fig. 3.2. This experiment is run for 2400s (40 minutes).

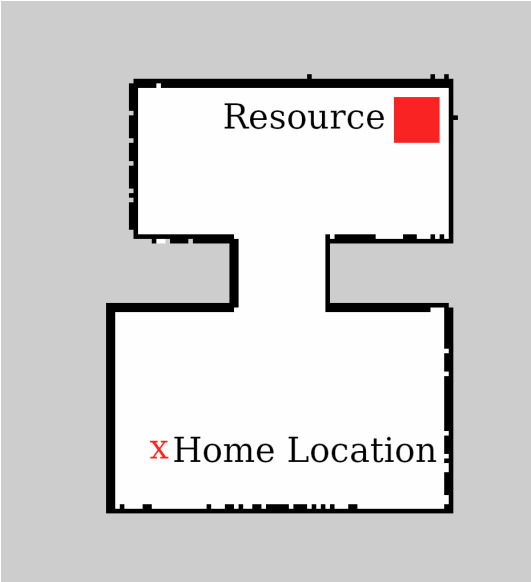


Figure 3.2: Map of two room arena created using SLAM gmapping. The red x marks the home location of the robots.

Scenario Three

The third and final experiment makes use of the same arena and home location as in the second experiment but this time two resources are used. Both resources are placed in the upper room and resource A is of significantly lower quality than resource B. The quality of resource A equals 15% of the value of resource B. This means that when a robot encounters resource A it will place weaker pheromone causing it to evaporate quicker. This experiment also uses three robots and it is run for 3000s (50 minutes).

The baseline run used for this experiment is the same as for experiment two as the best case scenario would be that the robots instantly find the better of the two resources which is identical to experiment two. But in the baseline run for experiment two all recorded values after 2400s are discarded.

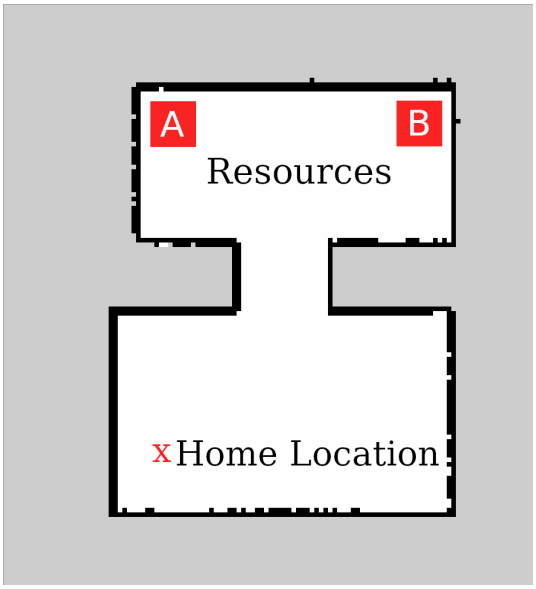


Figure 3.3: Map of two room arena created using SLAM gmapping. The red x marks the home location of the robots. Resource B has a significantly better quality than Resource A

4 Evaluation

This section is structured by the metrics that are evaluated. For every scenario the metric is compared with the baseline run. Following metrics are evaluated:

- Total resources collected
- Time until resource is detected
- Resource collection rate
- Heatmap of robot positions and pheromone concentration

4.1 Total Resources Collected

Every time a robot reaches a resource and then drives home the quality of the resource is added to the total. As in scenario one and two the only resource has a quality of 100, the scores in those scenarios are always a multiple of 100. In the third scenario the resource with the lower quality has a value of 15 so scores other than multiples of 100 appear. As 31 was chosen for the number of simulations for each scenario the lower quartile, the median and the upper quartile correspond to the 8^{th} best, the 16^{th} best and the 24^{th} best run respectively.

Table 4.1: Total resources collected in the simulation

Scenario	Median	Median baseline	% of baseline
One	1900	8300	22.892%
Two	2100	6800	30.882%
Three	825	8300	9.940%

As the table 4.1 and fig. 4.1 show, the median, the lower quartile and the upper quartile are consistently outperformed by the baseline run. This is to

be expected as the baseline represents the best case scenario. To find out where the shortcomings of the pheromone based approach are there are a few metrics one might look at. The most obvious might be the time until a robot first found the/a resource, as in the baseline run the robots do not need to explore the environment and locate the resource. Another interesting metric to compare is the rate at which resources get collected as this is basically a measure for how well the robots can follow the pheromone trail they previously laid down.

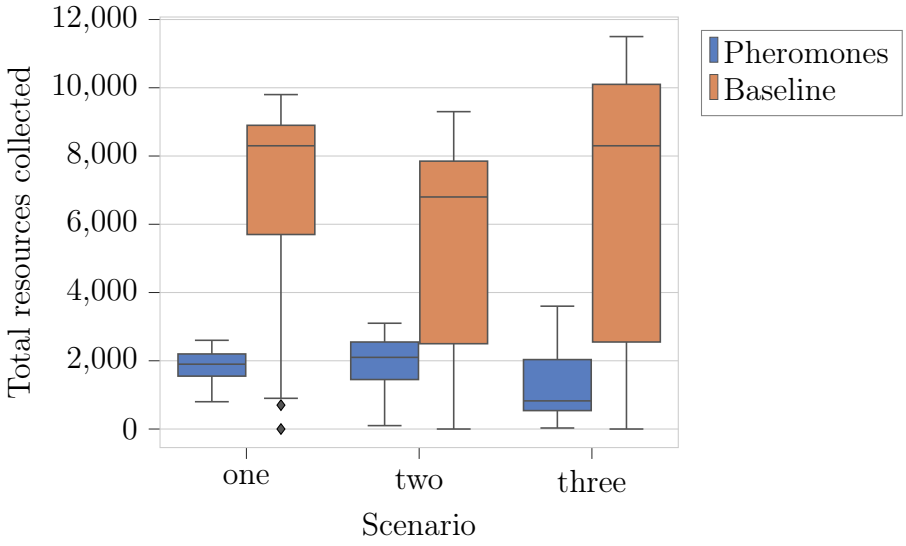


Figure 4.1: Boxplot of the total amount of resources the robots collected compared to the baseline runs.

4.2 Time Until Resource is Detected

As can be seen in table 4.2 in scenario one and two the robots spend about one fifth of the total experiment time searching for resources. In scenario three only five percent of the time is spent searching for resources. This is probably caused by the extra resource being available in scenario three. But the discrepancy between the very poor performance in total collected resources (table 4.1) and the very short time spent searching for resources (table 4.2) in scenario three indicates that the exploration time is not a good indicator for performance./ That is also seen in scenario one and two. The median run only scored about 20% to 30% as much resources as the baseline run but the

time that was available was about 80% that of the baseline run. The time it takes the robots to find the resource is a function of the environment (the arena shape, pheromone placement and size) and the random walk, and is of no further interest for the evaluation of the pheromone system.

Table 4.2: Table of the time it takes the robots to detect the resource for the first time. tuR is the time until a resource was first detected.

Scenario	Median tuR in s	Percentage of experiment
One pheromones	378.950	21.053%
One baseline	35.736	1.986%
Two pheromones	443.120	18.464%
Two baseline	40.534	1.689%
Three pheromones	152.370	5.079%
Three baseline	40.534	1.351%

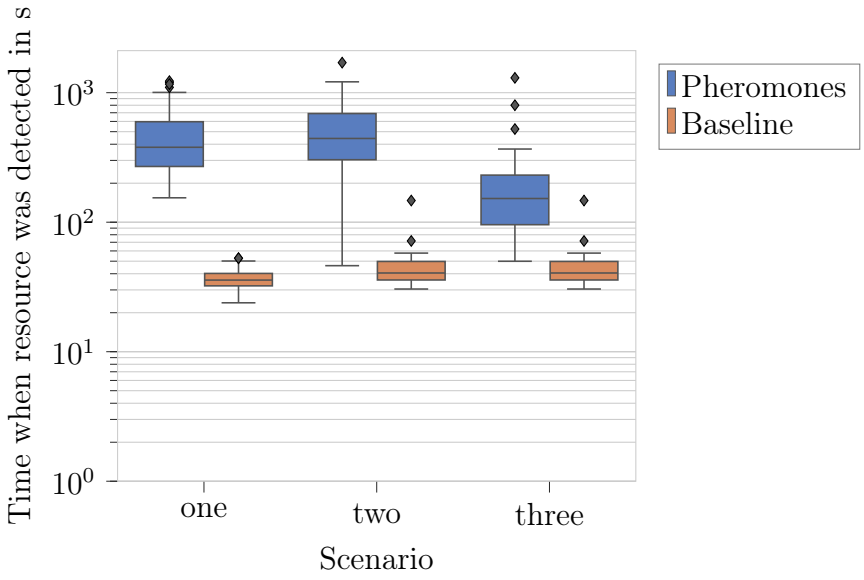


Figure 4.2: Boxplot of the time it takes the robots to detect the resource for the first time.

4.3 Resource Collection Rate

Once a single robot has detected a resource it will lay down a pheromone trail from the resource to the home location. Now the resource collection is dependent on how fast the robots can locate the pheromone trail and how well they can follow it. Thus the rate at which the robots collect resources, specifically the rate of resource collection *after* the first time a robot has detected a resource, is a better measure of performance for the pheromone system than the time until a resource is first found.

As can be seen in table 4.3 the resource collection rate is significantly lower than the baseline run for all scenarios. It sticks out that performance in scenario three is only about half that of scenario two even though the used arena is the same. This indicates that a significant portion of the time the robots encountered the lower quality resource. Confirmation thereof is found in fig. 4.4. For scenario three there is a bundle of poorly performing runs with a low resource collection rate which consists of about half of all runs. The other half of the runs perform way better but are less consistent with the total resources collected ranging from about 1000 up to about 4000 while the lower half consistently scores below 1000. The high amount of poorly performing runs probably results from the robots finding resource A and then sticking to it, even though it has lower quality. Confirmation that the robots use resource A more often is presented in the next section.

Table 4.3: Table of the average resource collection rate *after* the first time a resource was detected

Scenario	Mean resource collection rate in $\frac{resource}{s}$		% of baseline
	Normal	Baseline	
One	1.467	4.575	32.066%
Two	1.157	3.098	37.347%
Three	0.541	3.080	17.565%

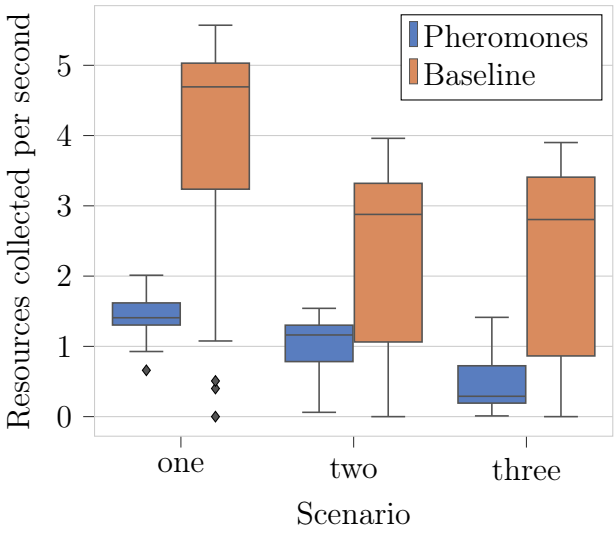


Figure 4.3: Boxplot of the resources collected per second *after* the first time a resource was found.

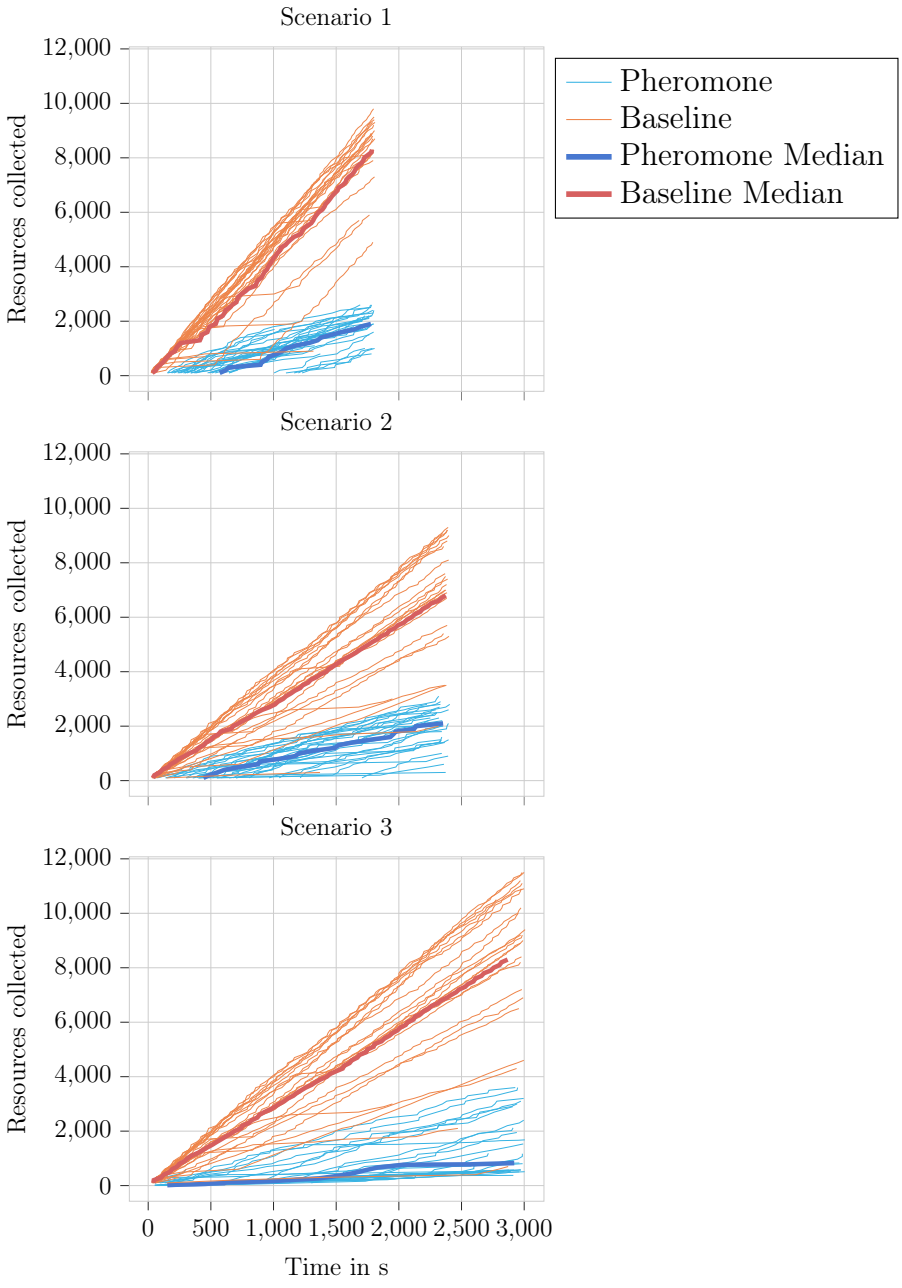
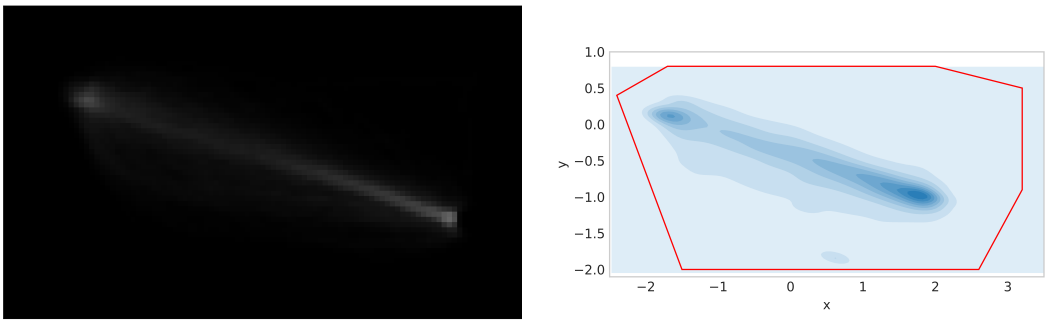


Figure 4.4: Resource collection over time for each of the scenarios compared to the baseline run

4.4 Position Heatmaps and Pheromone Concentration

In fig. 4.5b the kernel density estimation of the robots positions over all 31 runs of scenario one, sampled every second, is displayed. Figure 4.5a [1] shows the relative pheromone density of all runs. This was calculated by summing up all complete pheromone maps that were sent during the runs and dividing it by the maximum value. This shows that the robots spend a lot of time at the home location and the resource. Also, there is a high density on the straight path between home location and resource, indicating that a pheromone trail has been established and that this is the path the robots took most often. However there is an area of higher density on this path which gravitates towards the home location. There is two probable causes for this. The first is that the robots might be turning around while following the pheromone path and return to the home location. There is no mechanism in place to prevent the robot following the pheromone traul in the wrong direction. While examining some sample runs this behavior could certainly be observed but no quantitative analysis was conducted as this is not easily done programmatically and it is not feasible to manually analyse this amount of data. The other cause might be the path planner which doesn't always choose the straight path to the home location causing the pheromone trails to fan out from the resource as can be seen in fig. 4.5a



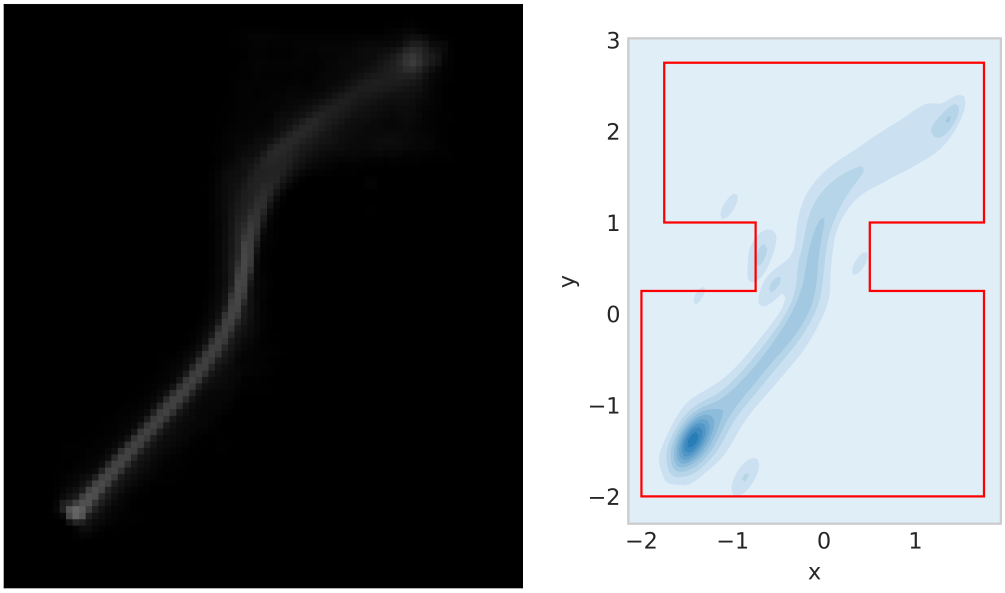
(a) Pheromone density

(b) Position density

Figure 4.5: Density of pheromone and robot position over all runs in scenario one

[1] The pheromone density maps have been create with the help of GNU Parallel [24]

The same visualization for scenario two can be seen in fig. 4.6. Similarly to scenario one there are areas of higher density at the home location and a direct path between home location and resource can be seen. Also like in scenario one the density on that path increases towards the home location. Possible reasons for this have been stated above.



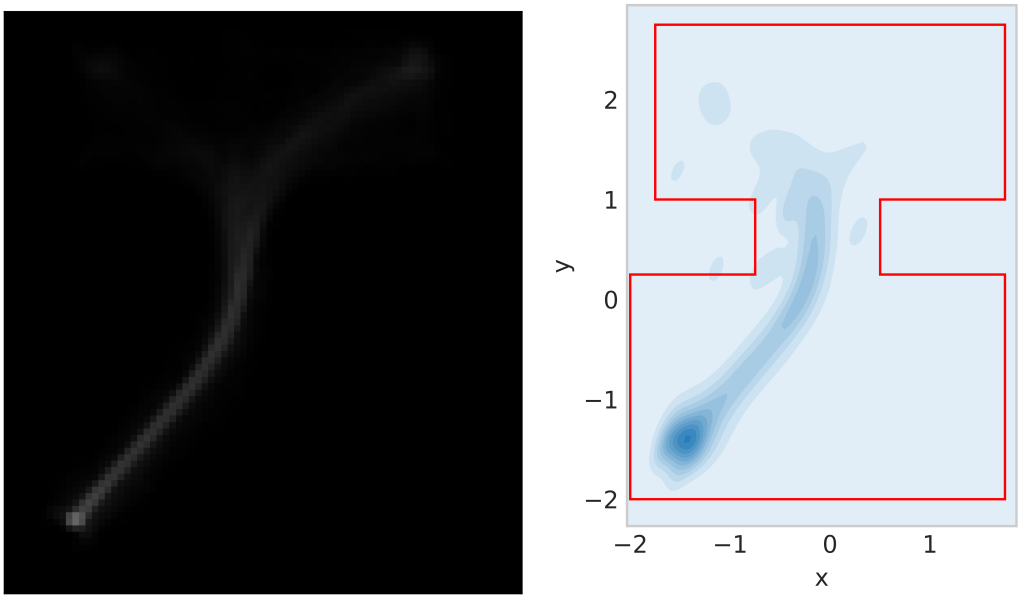
(a) Pheromone density

(b) Position density

Figure 4.6: Density of pheromone and robot position over all runs in scenario two

The pattern formed in the kernel density estimation of the robots positions for scenario three (see fig. 4.7b is different from the one formed in scenario one and two. In the lower half of the arena the density looks very similar to scenario two. There is also a higher density at the home location and a distinct path up until the end of the corridor. But after that the path fades out and breaks up into two parts. A small spike is visible that points toward resource B (the higher quality one) and a visibly larger protrusion that points toward resource A. Additionally a patch of slightly higher density is visible at the location of resource A. So clearly there is a tendency of the robots to collect from resource A. At first glance the pheromone density map in fig. 4.7a seems to contradict this observation but this is not the case. Because of the lower quality of resource A the robots lay down pheromone with less strength and

therefore the density of the pheromone trail towards resource A can be lower than that to resource B even though it was traveled to more often.



(a) Pheromone density

(b) Position density

Figure 4.7: Density of pheromone and robot position over all runs in scenario three

4.5 Paths

This section includes all paths of non-baseline runs. Every 2 by 2 block shows one run with each quarter showing one quarter of the run. So for experiment one the top left shows the first 7.5 minutes, the top right from 7.5 minutes to 15 minutes, the bottom left from 15 to 22.5 minutes and the bottom right from 22.5 to 30 minutes. The three different colors correspond to the path of one of the three robots.

These figures show that the robots establish a path between home location and resource in most of the runs. But they also reveal some issues. There are some cases where some of the quadrants are empty or some paths are missing, indicating that the robots got stuck(e.g. 4.8 Run 11 and 4.9 Run 13). Also there are some cases where a path seems to be established but in a timestep

after that the paths cover the whole arena, which means that the robots lost the path and fell back into the exploration stage(e.g. 4.9 Run 10 and 15, 4.10 Run 6 and 19).

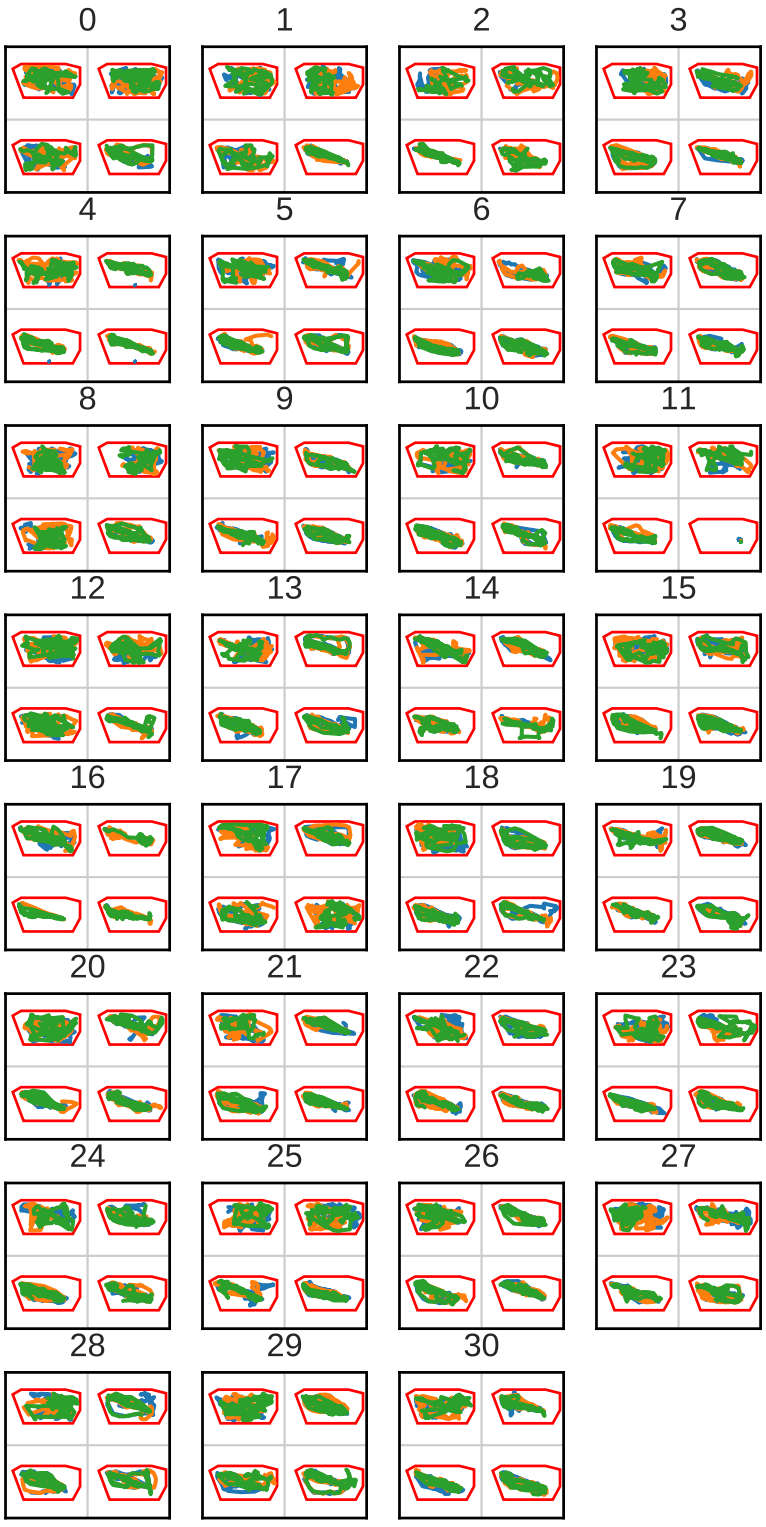


Figure 4.8: Paths for all experiment runs from scenario one

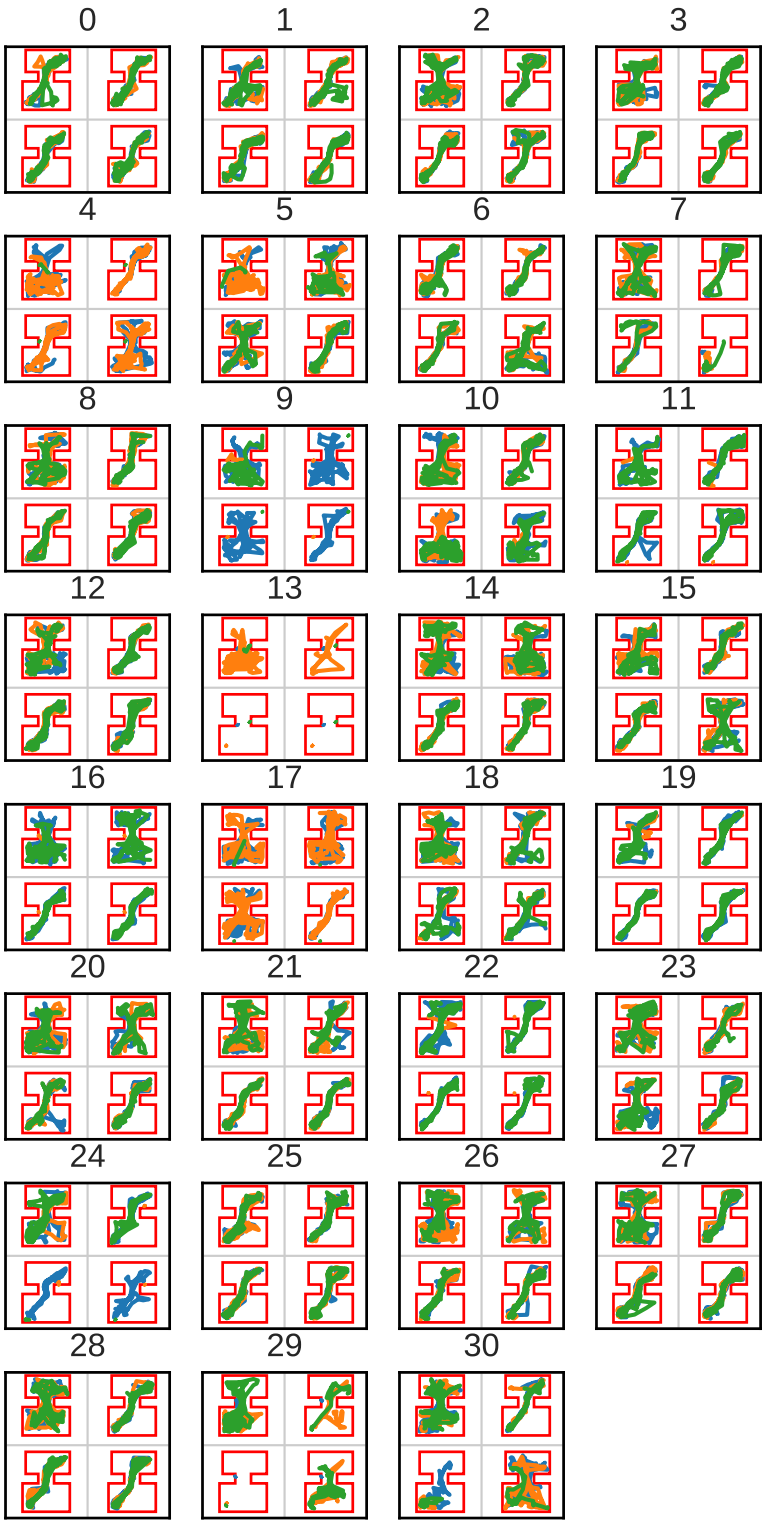


Figure 4.9: Paths for all experiment runs from scenario two

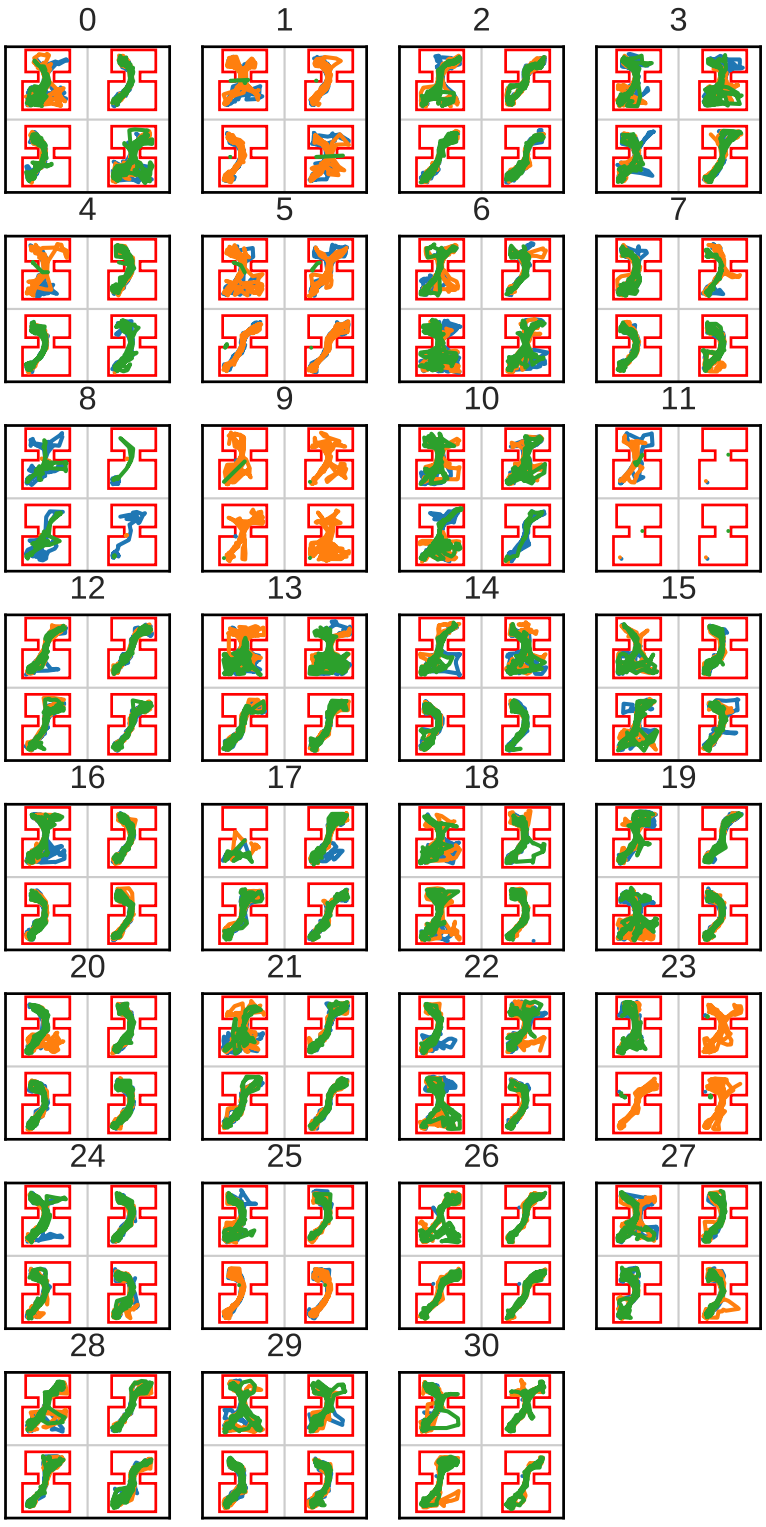


Figure 4.10: Paths for all experiment runs from scenario two

4.6 Summary

To summarize, the total amount of resources collected, the time until resources are detected, the resource collection rate and the pheromone and position densities were evaluated. The total amount of resources collected is about 23% of the baseline in scenario one, about 31% of the baseline in scenario two and about 10% of the baseline in scenario three. In scenario two the robots take about 20% of the total experiment duration to locate the food and only 5% in scenario three. The resource collection rate in scenario one and two is between 30% and 40% of the baseline and only about 18% of the baseline in scenario three. The pheromone and position density maps for scenario one and two show a clearly developed path between home location and resource. For scenario three a bias towards resource A, the lower quality one, is visible.

5 Conclusion and Future Work

The goals defined in section 1.2 have been achieved. In this thesis a pheromone map using the Turtlebot 3 Burger has been implemented. It uses ROS and it can be simulated using Gazebo. Utilizing this pheromone map a foraging behavior has been implemented and its performance has been compared to a baseline behavior.

The implemented behavior, while performing worse than the best-case scenario, has been shown to work reliably in single resource scenarios. The individuals locate the resource and establish a pheromone trail between it and the home location. In multi resource and multi resource quality scenarios however, the performance is lower and the individuals do not find the best resource reliably.

All the experiments in this thesis used three individuals and different swarm sizes have not been evaluated because of the significant computation time these experiments require. But because swarm size can have a major impact on the exhibited behavior different swarm sizes should be evaluated.

In bee and ant colonies individuals assume different roles. Applying this concept to this pheromone based foraging approach could mean assigning *explorer* and *forager* roles. This has the potential to mitigate the issue that the individuals only forage from one resource once it is found even though a better resource is available. Also dynamic reassignment of these roles could be of interest, e.g. when it is very certain that the best resource has been found or the whole environment has been explored.

The way in which the robots follow a pheromone trail uses a path planner mainly because it offers collision avoidance. Adding collision detection and avoidance, and adding it to the reactive implementation of the pheromone follower should be tested and evaluated as this would lead to an overall behavior implementation that is more close to the classic reactive agent and also relies less on a previously known environment. Another shortcoming of the

pheromone follower is that it has no way of preventing itself from following the pheromone trail in the wrong direction. This could be implemented by adding home sensor and using its output when determine which pheromone sensor sector to choose. So that when there are two sectors with very similar amounts of pheromone the one further from the home location is chosen.

Currently the implementation of the foraging behavior relies on an environment that was previously explored and is therefore known. A more interesting and more close to reality approach would involve an unknown environment as an already known environment, doesn't require a random walk for exploration. What could be done in the future is running SLAM on each robot and then in some way exchange the gathered environment information between the robots. When combining the environment maps each robot created additional localization systems could be used to find the transformation between all the robots reference frames.

The pheromone map creation currently works in a centralized fashion. All robots report their local pheromone map to a central node which combines them. When no centralized network like wireless LAN is available, like a partially connected mesh network, this might not be feasible. With no pairwise direct link between all individuals and the central node a decentral approach would probably work better.

Bibliography

[1] smach - ros wiki. URL `https://wiki.ros.org/smach`. accessed: 2019-09-25.

[2] PLATFORM - TurtleBot 3 - ROBOTIS. URL `http://www.robotis.us/turtlebot-3/`. accessed: 2019-09-18.

[3] gmapping - ros wiki, 2019. URL `https://wiki.ros.org/gmapping`.

[4] 360 laser distance sensor lds-01 (lidar) - robotis, 2019. URL `http://www.robotis.us/360-laser-distance-sensor-lds-01-lidar/`.

[5] Farshad Arvin, John Murray, Chun Zhang, and Shigang Yue. Colias: An Autonomous Micro Robot for Swarm Robotic Applications. *International Journal of Advanced Robotic Systems*, 11(7):113, July 2014. ISSN 1729-8814. doi: 10.5772/58730. URL `https://doi.org/10.5772/58730`.

[6] Fidel Aznar, Mar Pujol, Ramón Rizo, and Carlos Rizo. Modelling multirotor UAVs swarm deployment using virtual pheromones. *PLOS ONE*, 13 (1):e0190692, January 2018. ISSN 1932-6203. doi: 10.1371/journal.pone.0190692.

[7] Marco Dorigo, Vittorio Maniezzo, Alberto Colorni, et al. Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, man, and cybernetics, Part B: Cybernetics*, 26(1):29–41, 1996.

[8] Anna Font Llenas, Mohamed S. Talamali, Xu Xu, James A. R. Marshall, and Andreagiovanni Reina. Quality-Sensitive Foraging by a Robot Swarm Through Virtual Pheromone Trails. In Marco Dorigo, Mauro Birattari, Christian Blum, Anders L. Christensen, Andreagiovanni Reina, and Vito Trianni, editors, *Swarm Intelligence*, Lecture Notes in Computer Science, pages 135–149. Springer International Publishing, 2018. ISBN 978-3-030-00533-7.

[9] Ryusuke Fujisawa, Shigeto Dobata, Ken Sugawara, and Fumitoshi Matsuno. Designing pheromone communication in swarm robotics: Group foraging behavior mediated by chemical substance. 8(3):227–246. ISSN 1935-3812, 1935-3820. doi: 10.1007/s11721-014-0097-z. URL `http://link.springer.com/10.1007/s11721-014-0097-z`.

[10] Plerre-P. Grassé. La reconstruction du nid et les coordinations interindividuelles chezBellicositermes natalensis etCubitermes sp. la théorie de la stigmergie: Essai d'interprétation du comportement des termites constructeurs. 6(1):41–80. ISSN 1420-9098. doi: 10.1007/BF02223791. URL `https://doi.org/10.1007/BF02223791`.

[11] Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard. Improved techniques for grid mapping with rao-blackwellized particle filters. 23(1): 34–46. ISSN 1552-3098. doi: 10.1109/TRO.2006.889486. URL `http://ieeexplore.ieee.org/document/4084563/`.

[12] Herianto, Toshiki Sakakibara, and Daisuke Kurabayashi. Artificial pheromone system using RFID for navigation of autonomous robots. *Journal of Bionic Engineering*, 4(4):245–253, December 2007. ISSN 1672-6529, 2543-2141. doi: 10.1016/S1672-6529(07)60038-9.

[13] Ali Abdul Khaliq and Alessandro Saffiotti. Stigmergy at work: Planning and navigation for a service robot on an RFID floor. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1085–1092, Seattle, WA, USA, May 2015. IEEE. ISBN 978-1-4799-6923-4. doi: 10.1109/ICRA.2015.7139311.

[14] Yoshihiko Kuwana and Isao Shimoyama. A pheromone-guided mobile robot that behaves like a silkworm moth with living antennae as pheromone sensors. *The international Journal of Robotics research*, 17 (9):924–933, 1998.

[15] Francesco Mondada, Michael Bonani, Xavier Raemy, James Pugh, Christopher Cianci, Adam Klaptocz, Stephane Magnenat, Jean-Christophe Zufferey, Dario Floreano, and Alcherio Martinoli. The e-puck, a Robot Designed for Education in Engineering. page 7.

[16] Seongin Na, Mohsen Raoufi, Ali Emre Turgut, Tomas Krajnk, and Farshad Arvin. Extended Artificial Pheromone System for Swarm Robotic Applications. page 8.

[17] OSRF. Gazebo, 2019. URL `http://gazebosim.org/`. accessed: 2019-09-18.

[18] David W. Payton, Michael J. Daily, Bruce Hoff, Michael D. Howard, and Craig L. Lee. Pheromone robotics. pages 67–75. doi: 10.1117/12.417331. URL `http://proceedings.spiedigitallibrary.org/proceeding.aspx?articleid=927316`.

[19] Anies Hannawati Purnamadjaja and R. Andrew Russell. Guiding robots' behaviors using pheromone communication. *Autonomous Robots*, 23 (2):113–130, August 2007. ISSN 0929-5593, 1573-7527. doi: 10.1007/s10514-007-9035-x.

[20] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, Japan, 2009.

[21] Andreagiovanni Reina, Alex J. Cope, Eleftherios Nikolaidis, James A. R. Marshall, and Chelsea Sabo. ARK: Augmented reality for kilobots. 2(3): 1755–1761. ISSN 2377-3774. doi: 10.1109/LRA.2017.2700059.

[22] Michael Rubenstein, Christian Ahler, Nick Hoff, Adrian Cabrera, and Radhika Nagpal. Kilobot: A low cost robot with scalable operations designed for collective behaviors. *Robotics and Autonomous Systems*, 62 (7):966–975, July 2014. ISSN 09218890. doi: 10.1016/j.robot.2013.08.006.

[23] Ioan Susnea, Grigore Vasiliu, Adrian Filipescu, and Adrian Radaschin. Virtual Pheromones for Real-Time Control of Autonomous Mobile Robots. *Studies in Informatics and Control*, 18(3):8, 2009.

[24] O. Tange. Gnu parallel - the command-line power tool. *;login: The USENIX Magazine*, 36(1):42–47, Feb 2011. URL `http://www.gnu.org/s/parallel`.

[25] P. Vartholomeos and E. Papadopoulos. Analysis, design and control of a planar micro-robot driven by two centripetal-force actuators. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 649–654. doi: 10.1109/ROBOT.2006.1641784. ISSN: 1050-4729.

Declaration of Authorship

I hereby declare that this thesis was created by me and me alone using only the stated sources and tools.

Nico Winkelsträter Magdeburg,