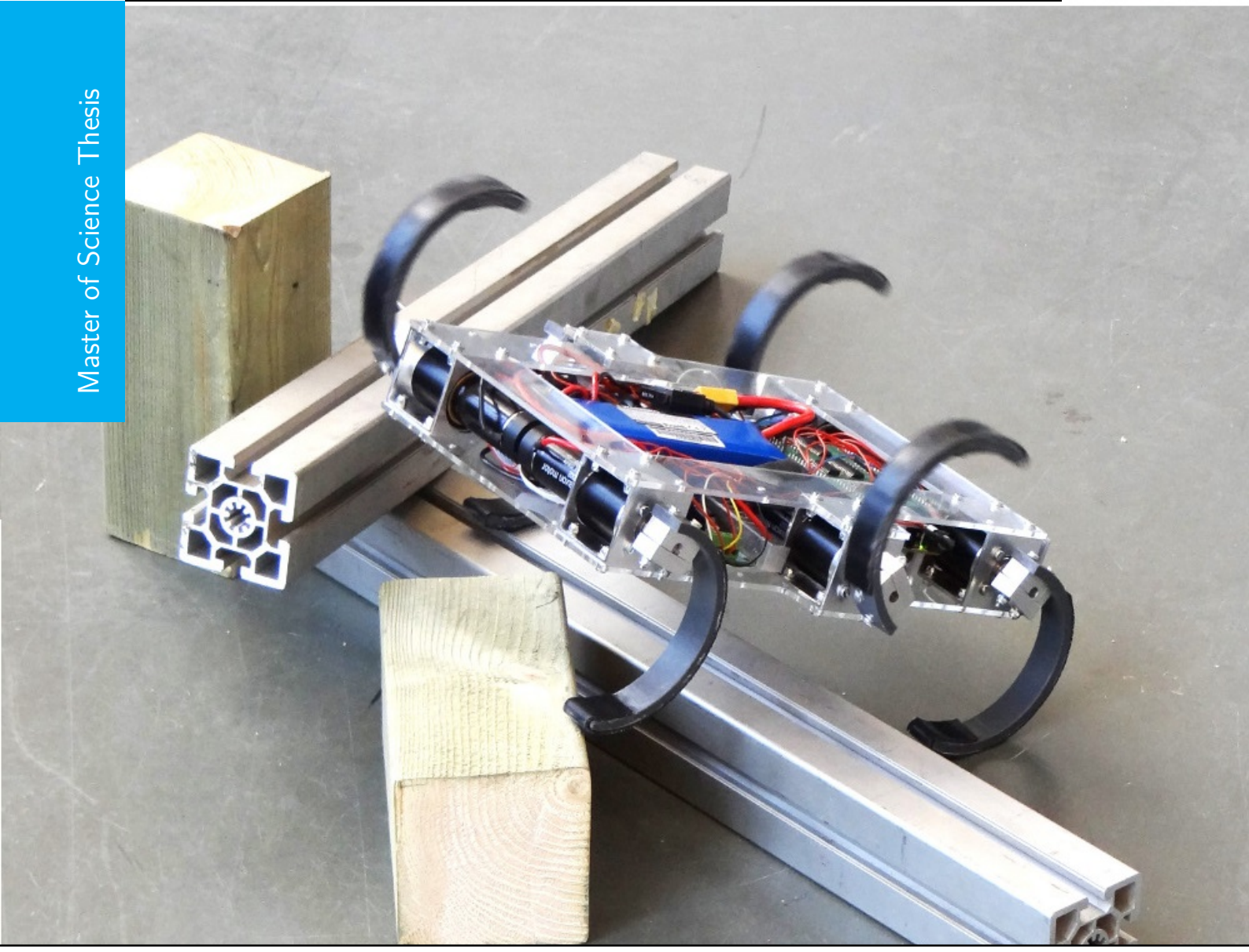
Mobile robot swarming using radio signal strength measurements and dead-reckoning

M.C.R. van der Klauw

Master of Science Thesis



TU Delft Delft University of Technology

Delft Center for Systems and Control

Mobile robot swarming using radio signal strength measurements and dead-reckoning

Master of Science Thesis

For the degree of Master of Science in Mechanical Engineering at Delft University of Technology

M.C.R. van der Klauw

October 7, 2015

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

The work in this thesis was supported by the Cyberzoo and Zebro team at the TU Delft.

Abstract

Different methods exist to create a distributed controller for swarms of mobile robots. The mobile robots considered in this thesis are six legged "Zebro" robots. These mobile robots will use Radio Signal Strength (RSS) measurements to determine the distances towards other mobile robots and radio beacons placed in the surroundings. A combination of distance measurements and dead-reckoning is used to perform a localization of the relative positions of the other mobile robots in the neighbourhood. The focus of the localization algorithm is to deal with a bad performance of the distance estimation, because this will result in uncertain position estimations. With knowledge about the bad performance of the localization an according suitable swarm algorithm is designed. This swarm algorithm is will also be used to test how valuable the position estimations can be as an addition to already existing swarm algorithms.

The localization method proposed uses equations from the Relative Pose Estimation (RPE) [1] in combination with a trilateration of distance measurements between all swarm members, so-called Multi-Robot Trilateration (MRT). An Unscented Kalman filter (UKF) is used to perform the localization using both RPE and MRT. Given the estimated positions of the other mobile robots a Virtual Potential Field (VPF) controller is used to control the heading direction of a mobile robot. These VPF controllers use streaming functions [2, 3] to avoid local minima. The first VPF controller will create cohesion of the robot swarm without letting the mobile robots collide. From simulations it was investigated how much effect uncertain position estimations can have on the cohesion of the swarm. A second VPF controller will steer the robot swarm away from repulsive obstacles represented by radio beacons. Again from simulations tests conclusions were drawn to see how much effect the uncertain positions have on avoiding obstacles and which navigation algorithm is most suitable to navigate the swarm.

Table of Contents

Chapter 1

Introduction

Today's society requires an increased level of automation to improve daily lives [4] and processes [5]. One subject for automation is the use of robots for examining or exploring a given environment. The use of robots can be helpful in cases that the environment is for example to dangerous or inhabitable for humans. It is better to have multiple robots than one robot for such tasks [6], because using a monolithic sophisticated robot would represent a single point of failure and might also be much slower in accomplishing the task. Instead, a so-called robotic swarm consists of simpler robots with less advanced and cheaper equipment. Another advantage of a mobile robot swarm is that it is scalable and can work parallel to each other, so multiple tasks can be performed simultaneously. Furthermore swarming robots are cheaper and therefore can be mass-produced.

Last few decades multiple researchers have worked on robotic swarming technology. Today robot swarms are a hot topic. In [7] a thousand little robots create complex two dimensional shapes and in [8] an example is given about very small microbots. These microbots operate with simple measurements, which result in a primitive way of swarming. One of the most primitive swarms is presented in [9], these robots do not have any sensors at all. These robots create a swarm which looks like a group of bacteria with only doing one particular movement and colliding with each other. In this thesis project also a robotic swarm with a low amount of available information is considered, because the robot swarm should work under various circumstances. Therefore one of the requirements is the use of a minimal set of sensors and equipment. This also allows reducing the amount of software and hardware faults on site and a minimal amount of equipment uses less energy and therefore longer operation time is possible. Letting the robot swarm function under various circumstances also includes situations where external sensor systems such as GPS are not even available, for example in the underground or outer space. Having an indication of the positions of the other mobile robots antenna's onboard each mobile robot are used to measure the distance to one another. The distance

is measured using Radio Signal Strength (RSS) measurements of received messages from the other mobile robots. Distance measurements by measuring the RSS are very uncertain, dealing with this will be the major challenge of this thesis. In conclusion the research question can be formulated as:

How can a distributed controller for a robot swarm containing Zebro robots be designed using uncertain distance measurements and minimal additional sensors?

An important question for creating such robotic swarm is: What kind of swarming behaviour should be achieved? In nature there are multiple examples of swarming behaviour which each use a different level of intelligence. For example whales can communicate over very large distances, sometimes thousands of kilometers, to stay in contact. They also have a very complex way of communication with high amounts of information [10]. In contrast, an ant only communicates on short distances with his antennas and at larger distances only by odor or pheromone trails left by other ants [11]. Using mainly these measurements an ant even can stay close to the group, explore the surroundings and together accomplish large tasks as constructing an entire an ant nest. This philosophy of the ant colony is also the basis for the robot swarm. A similarity between the mobile robots and ants (or other insects) is that they also only know about their direct surroundings visualized by (virtual) odors. In this thesis a mobile robot uses the Radio Signal Strength (RSS) measurement of messages from radio beacons on site in stead of real odors. Every mobile robot itself also transmits messages (odors), which will also be used to communicate between each other. The description of the Zebro robot as the considered mobile robot along with a description of its onboard sensors is given in Chapter 2.

Many swarm algorithms already exist that can deal with virtual odors presented by distance measurements from obstacles or attractive goal beacons. In the first concept for a swarming algorithm using the Zebro robot the robot only knew the distances towards objects. The swarming algorithm used a genetic learning algorithm to converge to an optimal heading direction. Having a notion of the positions of these objects will give a better heading direction. Therefore in this thesis the goal is to prove that a localization algorithm can estimate the positions of the other mobile robots and if so that it can be a valuable addition to the swarm algorithms that only use distances. A localization algorithm to give a notion of the positions of the other mobile robots is designed in Chapter 3. The major challenge of this localization algorithm is to deal with very uncertain distance measurements from the RSS measurements, which also results in uncertain position estimations.

The first step is to establish a distributed mobile robot swarm controller that to let the swarm stay together while not colliding into each other. The Virtual Potential Field (VPF) controller is one of the most used controllers that uses the position estimations to do so. A downside of the VPF controller is that most researchers use a simulation environment with known positions of all the swarm members. The effects of more uncertain and realistic position estimations of the swarm members on VPF swarm controllers was not found in literature. Therefore these effects will be researched in Chapter 4.

In the last chapter multiple algorithms are tested to let the complete mobile robot swarm

interact with the environment. In this case only the distances towards the obstacles and attractive goals are known. The goal is to combine the VPF swarm controller with an algorithm that lets the swarm interact with its environment. A point of research will be again the effects of uncertain position estimations and noisy distance measurements on the heading direction of the swarm members.

In the end conclusions are drawn to see how valuable the uncertain position estimations from the localization are and what effect they have on the existing swarm algorithms and algorithms to interact with the environment. Based on the simulation results also multiple topic of improvement are handled for future implementation and a better working swarm algorithm.

Chapter 2

Motion and measurement models

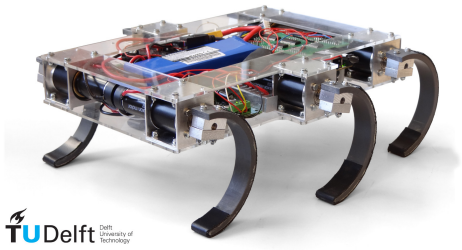


Figure 2-1: The Zebro robot.

The first task to create a mobile robot swarm containing Zebro robots is to know the characteristic of the Zebro robot. In this thesis these characteristics of Zebro robot are given by the motion models of the Zebro robot. These motion models consist of a motion model of the Zebro robot itself and a relative motion model to describe the relative motions of other mobile robots from the perspective of a single Zebro robot. A relative motion model is needed, because the mobile robots do not have a position reference to know their absolute coordinates. In this thesis all mobile robots with the same motion model as the Zebro robot are considered. Therefore the name Zebro robot will not be used, but the general name "mobile robot" that represent all mobile robots with the same motion model as the Zebro robot.

Further on the sensors that are placed on board the mobile robot are handled. These sensors have a goal to perform a localization to estimate the position of the other mobile robots and obstacles. The first sensor is an antenna which measures the Radio Signal Strength (RSS) between two antenna's onboard two different mobile robots (or beacons). The measured RSS will be converted to a measured distance between the two mobile robots. The second sensor

tracks the walked direction and distance of the mobile robot at each time sample, so called dead-reckoning. The combination of the robot-to-robot distance and tracking its own movement will be used to perform a localization of the neighbouring Zebro robots, which is handled in the next chapter.

2-1 Motion model of the Zebro robot

This chapter starts with the motion model of a single Zebro robot. The Zebro robot is a six legged robot designed by the Robotics Institute at the Delft University of Technology [12] and looks like the mobile robot in Figure 2-1. The Zebro robot can be modelled as so-called two-wheeled robots, unicycles or mono-cycle robots [13–15]. In this thesis all mobile robots that can be modelled with the same motion model as the Zebro robot are considered. The motion model is given by the non-linear state-space model given by equation (2-1) and Figure A-3.

$$\bar{x}(k+1) = \begin{bmatrix} x(k+1) \\ y(k+1) \\ \theta(k+1) \end{bmatrix} = f(\bar{x}(k), \bar{u}(k)) = \begin{bmatrix} x(k) + u_v(k)\cos(\theta + u_\theta(k)) \\ y(k) + u_v(k)\sin(\theta + u_\theta(k)) \\ \theta(k) + u_\theta(k) \end{bmatrix}, \tag{2-1}$$

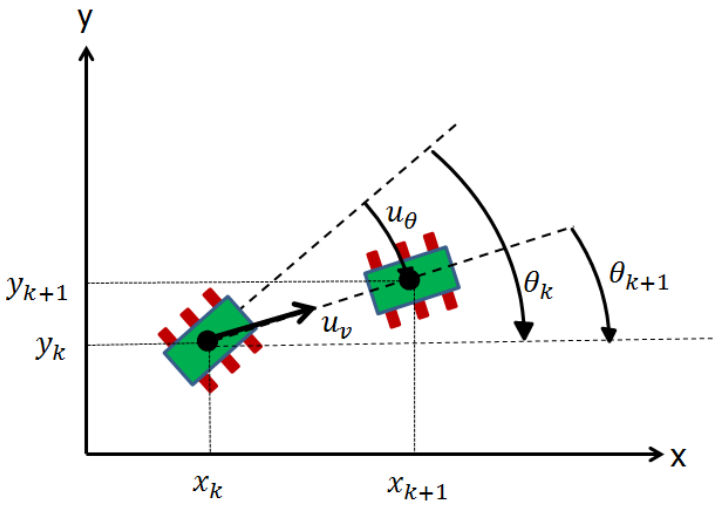


Figure 2-2: The motion model of the mobile robot considered given in an absolute coordinate system.

In Figure A-3 $\theta(k)$ is the pose at time k and is relative to the x-axis. The control inputs of the motion model are the heading direction u_θ and the travelled distance u_v.

In general $\theta(k+1)$ depends only on the previous heading direction u_θ and pose $\theta(k)$. So one might see that this motion model does not deal yet with situations where the new pose $\theta(k+1)$

is not equal to $\theta(k) + u_\theta$. An example when this happens is given in Figure 2-3. In practise this could happen often, due to rough surfaces and the jerky movement of the Zebro robot.

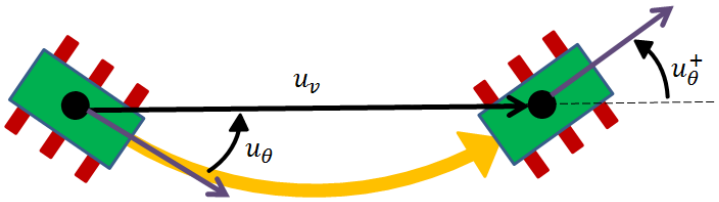


Figure 2-3: A curved walked path, where u_θ^+ is a difference between addition of the previous pose and the heading direction u_θ.

In Figure 2-3 the motion of the mobile robot is not straight, but with a curve. In that case $\theta(k+1) \neq \theta(k) + u_\theta$ holds and the new angle $\theta(k+1)$ needs an addition of u_θ^+, given as

$$\theta(k+1) = \theta(k) + u_\theta + u_\theta^+ \tag{2-2}$$

The presented non-linear motion model is sufficient to describe the absolute heading θ and absolute position (x, y) at each new time sample k with the control inputs u_θ and u_v, but this model is not sufficient to model the movement of the mobile robot in a coordinate frame without reference coordinates. In that case the robot does not know it absolute coordinates. So a relative motion model also needs to be used. The relative motion model let the mobile robot see the movement of the surrounding from its own perspective and is the topic of the next section.

2-2 Relative motion model

In the previous section the motion model of the mobile robot was given. In this section a relative motion model is presented, where the motions of the surroundings are seen from the perspective of a mobile robot. This is necessary, because the mobile robots do not have a reference point to know their absolute coordinates in the workspace. Therefore the mobile robot uses a local coordinate system for the relative positions of the other mobile robots, where the mobile robot itself is always placed in the origin headed along the x-axis, see Figure 2-4.

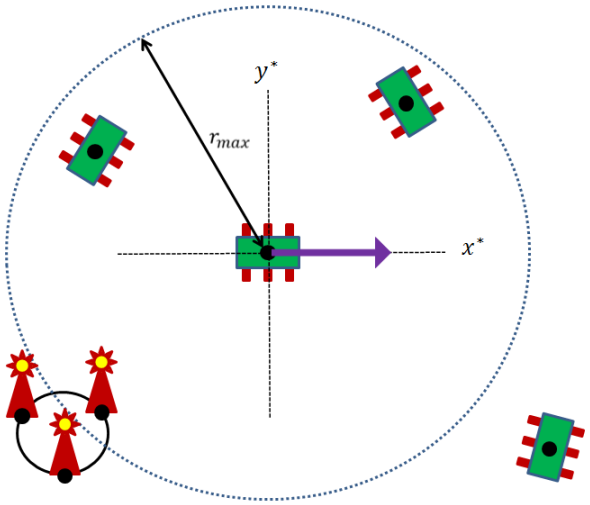


Figure 2-4: The surroundings seen from the perspective of a single mobile Zebro robot. The local coordinate system is given by (x^*, y^*) and the heading direction (purple arrow) is always aligned with the x-axis. The red symbols represent obstacles, where two of them lie outside the maximum range r_{max} of what a mobile robot can perceive.

Doing so creates a motion where the surroundings translates and rotates around the mobile robot. Figure 3-11 gives an overview on of the relative motion of mobile robot Z_i seen from mobile robot Z_0 with both robots having the same motion models of the previous section.

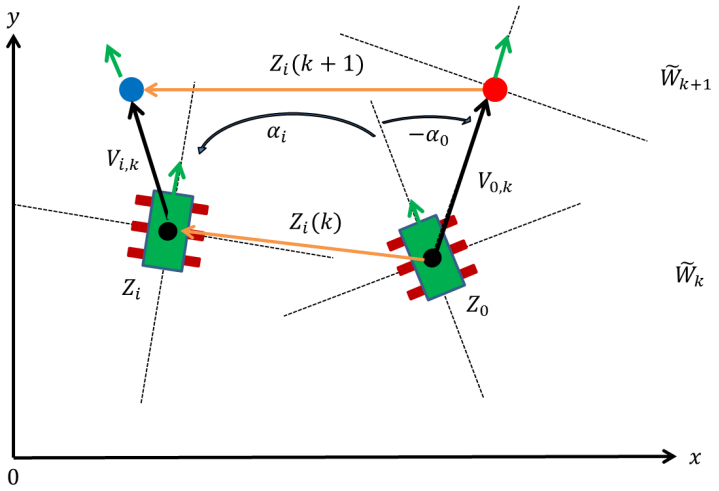


Figure 2-5: The relative motion between mobile robots Z_0 and Z_i. The mobile robots Z_0 and Z_i move with the vector $V_{0,k}$ and $V_{i,k}$ respectively.

In Figure 3-11 two mobile robots Z_0 and Z_i have moved with vectors $V_{0,k} = \begin{bmatrix} V_{x,0}(k) \\ V_{y,0}(k) \end{bmatrix}$ and

$V_{i,k} = \begin{bmatrix} V_{x,i}(k) \\ V_{y,i}(k) \end{bmatrix}$ respectively. The heading direction (pose) Z_i from the perspective of robot Z_0 is α_i and α_0 is the headed direction of Z_0 itself, where $\alpha_0 = u_{\theta,0}$. If the previous position of robot Z_i is the vector $Z_i(k)$ inside the local system $\tilde{W}_k$ then the following equation gives the updated next position $Z_i(k+1)$ inside the new local system $\tilde{W}_{k+1}$ [1].

$$\begin{bmatrix} x_i(k+1) \\ y_i(k+1) \\ \alpha_i(k+1) \end{bmatrix} = \begin{bmatrix} Z_i(k+1) \\ \alpha_i(k+1) \end{bmatrix} = \begin{bmatrix} R(-\alpha_0(k))[Z_i(k) + R(\alpha_i(k))V_{i,k} - V_{0,k}] \\ \alpha_i(k) - \alpha_0(k) \end{bmatrix}, \tag{2-3}$$

where $R(\alpha)$ is the rotation transformation matrix that rotates with the angle α. Like with the absolute motion model the pose angle α_0 can also be not equal to the headed direction u_θ, because of curved movements. As a compensation the following function for α_0 can be used.

$$\alpha_0 = u_{\theta,0} + u^+_{\theta,0} + u^+_{\theta,i}, \tag{2-4}$$

where $u^+_{\theta,i}$ is the angle between the headed direction $u_{\theta,i}$ and final pose of mobile robot Z_i, which was already explained by Figure 2-3.

The relative motion model is basically the motion model of Section 2-1 with an adaptation, where one looks from the perspective of mobile robot Z_0. One should keep in mind that the heading direction vectors $V_{0,k}$ and $V_{i,k}$ are Cartesian vectors and not polar control input vectors u_θ and u_v. Therefore the control input needs first to be converted to Cartesian coordinates to be used for $V_{0,k}$ and $V_{i,k}$. Also the relative coordinates of $Z_i(x_i, y_i)$ are Cartesian coordinates, but in this thesis a polar coordinate system will be used for the local coordinate system $\tilde{W}$. Why using polar coordinates is explained in Section 3-3. The only change to be made is a conversion from Cartesian coordinates $Z_i(x_i, y_i)$ to polar coordinates using the function $\begin{bmatrix} x_i \\ y_i \end{bmatrix} = f_{c\to p}\left(\begin{bmatrix} \theta_i \\ v_i \end{bmatrix}\right)$. The back-conversion is given by the function $\begin{bmatrix} x_i \\ y_i \end{bmatrix} = f_{p\to c}\left(\begin{bmatrix} u_{\theta,i} \\ u_{v,i} \end{bmatrix}\right)$.
With the relative motion model the relative motion of another mobile robot from the perspective of a mobile robot can be described. Which is useful, because the mobile robot do not know their absolute coordinates in the environment. The next section describes the model where the RSS between two antenna's is measured to find the distance between them. A discussion about the sensors for dead-reckoning is given in Appendix B.

2-3 Distance measurement model using radio signal strength

All the mobile robots will have the same sensors to localize the other mobile robots and/or antenna beacons. Distance measurements are performed using radio transceivers. The antenna's of the transceivers are placed on the mobile robots or as beacon placed somewhere in the surroundings. The RSS measured of a transmitted message send from another antenna is used to measure the distance r between the two antenna's. When two antenna's are further away from each other the RSS will drop and therefore an indication of the distance can be given. A

RSS-to-distance r model is known not to be very accurate, therefore in this section the main topic is to find the characteristics of the distance uncertainty. Knowing these characteristics of the distance measurements a suitable localization algorithm can be chosen or needs to be created.

The model chosen is the path loss Line-of-Sight model (LOS), which gives a good general indication of the distance in non-complex environments [16, 17]. The definition of the complexity of an environment is based on the amount of radio signal blocking/reflecting objects such as walls, humans and metal cabinets. There are models that give a better RSS-to-distance estimation in more complex environments. Such two models are the Non-Line-of-Sight (NLOS) models and fingerprint models [18, 19]. The downside is that these models need additional a priory measurements to create a map or model. This is undesired in emergency cases, or if the mission site is to dangerous for people or in remote places.

The chosen LOS model is given by the following equation

$$I(r) = I_0 - 10\beta \log_{10}(r) + \nu_I, \tag{2-5}$$

where I is the RSS in decibel Watt and r the distance in meters. I_0 is a constant also in decibel Watt based on the power at one meter and β is a path-loss parameter. The parameter β gives the declination of the RSS over the distance r and depends on the environment. A higher value for β is used when the environment is more complex. The last parameter in Equation (2-5) ν_I is an additional Gaussian white noise with a power σ_I^2. From experimental data given in Appendix the values for I_0, β and σ_I were found to be

$$I_0 = -53.71 \quad \beta = 3.424 \quad \sigma_I = 3.487$$

with a maximum distance r_{max} of 10 meters. When inverting Equation (2-5) the distance r can be calculated from I.

$$r(I) = 10^{\frac{-1}{10\beta}(I - I_0 + \nu_I)} \tag{2-6}$$

Important is that the noise ν_I is in the power ten part, which indicates a probability distribution of the distance r to be a log-normal distribution, see Figure 2-6 [13].

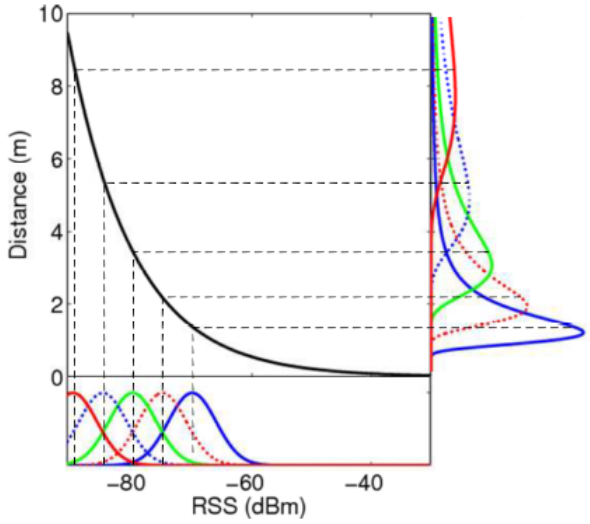


Figure 2-6: The mapping of a Gaussian noise distribution of the intensity to a log-normal distribution of the distance. The dotted lines are the mean of the distributions.

Still a normal distribution noise variance can be approximated, because the log-normal still looks quite similar to a normal distribution. For the approximation of a Gaussian noise standard deviation σ_r for the distance r a linear approximation is used. In Appendix C-1 it is proven that σ_r using the linear approximation is given as

$$\sigma_r = r \log(10) \frac{1}{10\beta} \sigma_I \tag{2-7}$$

Using this equation one should realise that the standard deviation of the intensity noise as $\sigma_I = 3.487$ dB is equal to $\sigma_r = 25\%$ of the distance r! Therefore a main topic in this thesis is how to deal with this given large amount of distance measurement noise. Also important to see is that the noise variance σ_r of the distance r is relative to the distance itself r. In some later functions the squared distance r^2 is used as the distance measurement and as proven in Appendix C-1 the following equation hold for the noise standard deviation σ_r^* of the squared distance r^2

$$\sigma_r^* = r^2 \log(10) \frac{1}{5\beta} \sigma_I \tag{2-8}$$

Using the linear approximation of σ_r and distance r in stead of the intensity I will result in biased distances estimations later on. A solution for elimination of the offset could be to rewrite all measured distances $\bar{r}$ back to a RSS $\bar{I}$ "distance", because the measurement I has an unbiased normal distribution. Later on the squared distance r^2 is used and the following function is used to map this to the RSS I.

$$I = I_0 - 10\beta \log(\sqrt{r^2}) = I_0 - 5\beta \log(r^2) \tag{2-9}$$

Rewriting all distances r to RSS distances I in the localization algorithms resulted in worse and wrong position estimations, because highly non-convex cost functions of the estimated positions were created.

In conclusion, the Zebro robot can be generalized as a one-wheeled robot with a relative motion model to model the motion of the surroundings from the perspective of a single mobile robot. Additionally to know this relative position of the other mobile robot one of the measurements used is a distance measurement. The distance is measured by measuring the RSS of the messages send between two transceivers each placed on another mobile robot. The results were as expected very noisy, $\sigma_r = 25\%$ as the standard deviation of the distance noise. The measured headed direction u_θ and travelled distance u_v are used to perform dead-reckoning. Both distance and dead-reckoning will be used for localization in the next chapter.

Chapter 3

Localization of other mobile robots

Knowing only the models from the previous chapter will not create a robot swarm. Each robot also needs to know in which direction they should walk to. There are algorithms that use a genetic search or a reinforcement learning approach to find the optimal heading direction. Mobile robots using these algorithms measure each time step how well they are positioned relative to other mobile robots and obstacles and learn from their "mistakes" over time to make better decisions. This can be done without a notion of the positions of the other mobile robots or obstacles, but having a notion of the location of the other mobile robots could create additional information. The additional knowledge about the positions of the other mobile robots will make it easier to decide which direction to go. Therefore in this chapter a localization algorithm is created to find the positions of the other mobile robots.

The localization algorithm uses distance measurements which were given in the previous chapter. Methods using only distance measurements are called trilaterations. In practice trilaterations are used for localization of mobile robots, cars, persons and more [20]. The most famous example is the Global Positioning System (GPS) [21]. The main difference with these implementations is that the distances are mostly measured from sensors/beacons with known positions. In this thesis the challenge is to create a localization algorithm where the position of the distance sensors also have to be estimated. In literature localization algorithms exist that do not require known positions of the distance sensors [22, 23]. The downside of these implementations is that estimation of the location of the sensors do not have a priority and still some a priory knowledge of the environment was required. Therefore a new localization method needs to be designed with the specific needs of robotic swarms and the sensors available on the Zebro robot.

First a general overview is given on how the trilateration normally is performed. As said earlier this method requires known positions of the distance sensors, but since that is not the case dead-reckoning is used as an additional measurement to perform a relative localization. The

first attempt using both distance and dead-reckoning measurements is by using a modification of weighted Multi-dimensional Scaling (MDS) [24]. Due to poor results of the position estimation of this MDS localization another attempt was necessary. A method called "relative pose estimation" (RPE) by Roumeliotis [1] is specifically designed for relative position estimation of another moving object by distance measurements and dead-reckoning. The RPE will be combined with a modification of the trilateration using a Kalman filter to improve the performance. Results and insights from simulations of this new localization algorithm are given at the end of the chapter. The newly designed localization algorithm creates estimated relative positions, which will be used as an input of the swarm controller in the next chapter.

3-1 Problem formulation

Before possible localization methods are covered the localization problem itself needs to be formulated. This relative position estimation problem is given in Figure 3-1.

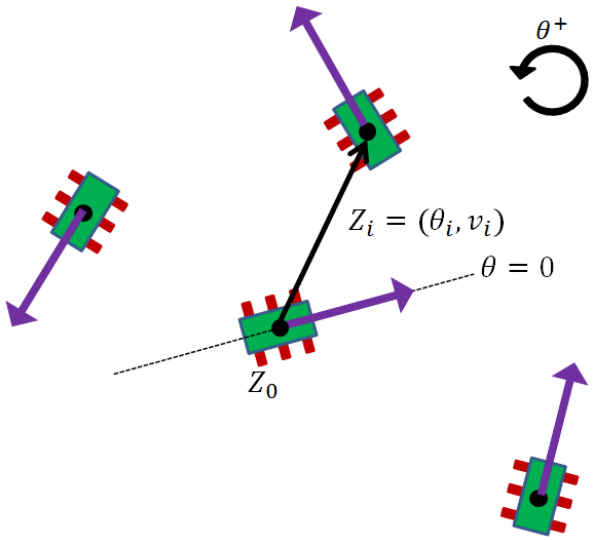


Figure 3-1: The relative position estimation problem. Here mobile robot Z_0 will try to estimate the position (θ_i, v_i) of mobile robot Z_i. Mobile robot Z_0 will always center itself in the origin of its own local coordinate system pointing to $\theta = 0$.

The main goal is to estimate the position of the other mobile robots Z_i from the perspective of a robot Z_0. Mobile robot Z_0 has its own local polar coordinate system which is called $\tilde{W}_0$. The relative position vector of Z_i is given as $\begin{bmatrix}\theta_i \\ v_i\end{bmatrix}$, with angle $\theta_i = [-\pi, \pi]$ and the distance $v_i = [0, r_{max}]$. The distance r_{max} is the maximum range of the antenna system, given in Section 2-3. The heading direction of mobile robot Z_0 always points along the axis of $\theta = 0$. As one also can see in Figure 3-1 the surrounding does not contain any static radio signal transmitting

beacons. If such a beacon with known coordinates existed the absolute coordinates of the mobile robot and others could be estimated. For such a localization problem multiple solutions and examples exist [20], but for this thesis the challenge is not having pre-knowledge about the positions of any of the robots or static beacons.

The localization algorithm will use measurements of the distance between the mobile robots. These distance measurements were discussed in Section 2-3 and are given by $\bar{r}_{i,j}$, with $i \neq j$, where $\bar{r}_{i,j}$ stand for the distance between mobile robot Z_i towards mobile robot Z_j. An important extra feature of the system is the possibility to have information about RSS measurements between two other robots Z_i and Z_j. This information can be communicated back to the mobile robot Z_0. Figure 3-2 gives an overview about which distance measurements then will be available. This method where distance measurements between other mobile robots are also used, is called "Multi-robot trilateration" (MRT).

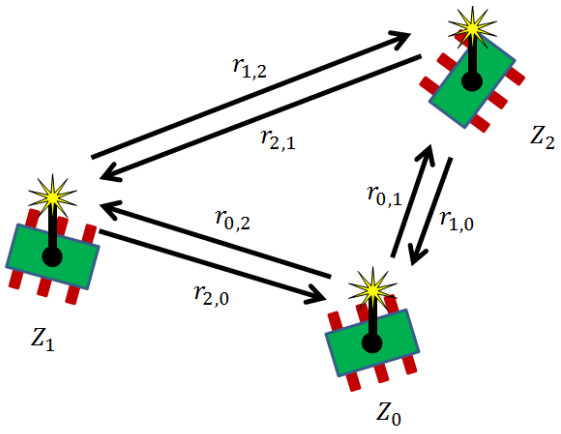


Figure 3-2: In this figure the distance $r_{i,0}$ with $i > 0$ presents the distance from mobile robot Z_i to the mobile robot Z_0. A distance measurement $\bar{r}_{i,j}$ between two other mobile robots Z_i and Z_j are also used, but only when both mobile robots are in sight of Z_0, therefore $\bar{r}_{0,i} < r_{max}$ is required for both mobile robots Z_i and Z_j.

Furthermore, the used nomenclature in this thesis used to distinguish different variables is given in Figure 3-3.

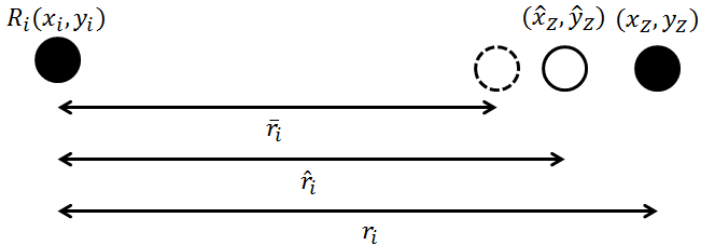


Figure 3-3: Nomenclature rules: A $\bar{}$-accent gives a measured value and a $\hat{}$-accent an estimated value. When no accent is used the true value is used.

In conclusion, the localization problem is formulated as the task where the relative position of other mobile robot needs to be estimated. The challenge is to such without having static beacons with known positions. Mainly distance measurements will be used to perform to localization, these distance measurement can also be distance measurements between two robots in sight $r_{i,j}$. Later also dead-reckoning measurements will become available as a necessity of this relative position estimation problem.

3-2 A least squares solution for trilateration

In line with the previous section a solution will be presented to perform the trilateration. The mathematical problem can be formulated in a linear trilateration problem. After a short explanation how to solve the trilateration problem a modification of this trilateration is outlined that uses the additional distances measurements so create the Multi-robot Trilateration. The MRT also deals with the constraints of unknown distance sensor positions. In the end the limitations of the MRT are given, which are solved in the next sections.

A trilateration uses only distance measurements to localize a mobile robot. Figure 3-4 presents the trilateration problem, where (x_Z, y_Z) has to be localized.

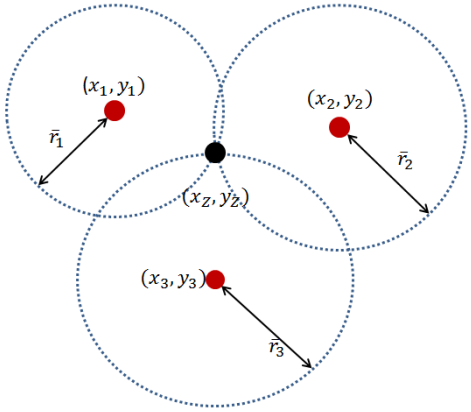


Figure 3-4: The trilateration problem. Each circle represents the possible positions of mobile robot Z according to the distance measurement $\bar{r}_i$. The goal is to find the point (x_Z, y_Z) where all circles intersect.

To solve this problem one can use the two equations below.

$$\begin{aligned} r_{1,Z}^2 - r_{2,Z}^2 &= ((x_1 - x_Z)^2 + (y_1 - y_Z)^2) - ((x_2 - x_Z)^2 + (y_2 - y_Z)^2) \\ r_{1,Z}^2 - r_{3,Z}^2 &= ((x_1 - x_Z)^2 + (y_1 - y_Z)^2) - ((x_3 - x_Z)^2 + (y_3 - y_Z)^2) \end{aligned} \tag{3-1}$$

These equations then needs to be solved for x_Z and y_Z by having at least two of such equations, which correspondents to at least three distance measurements. Solving Equation (3-1) for x_Z and y_Z can be done by a least-squares estimation. Therefore Equation (3-1) needs to be rewritten in a linear matrix form of $A\hat{x} = \bar{b}$ as

$$\begin{aligned} \bar{r}_{1,Z}^2 - \bar{r}_{2,Z}^2 &= 2\hat{x}_Z(x_2 - x_1) + 2\hat{y}_Z(y_2 - y_1) + x_1^2 - x_2^2 + y_1^2 - y_2^2 \\ \bar{r}_{1,Z}^2 - \bar{r}_{3,Z}^2 &= 2\hat{x}_Z(x_3 - x_1) + 2\hat{y}_Z(y_3 - y_1) + x_1^2 - x_3^2 + y_1^2 - y_3^2 \end{aligned}, \tag{3-2}$$

where $\bar{b}$ contains the distance measurements $\bar{r}_{i,j}$ and $\hat{x}$ is the vector with the estimated positions $(\hat{x}_Z, \hat{y}_Z)$. With the use of knowledge about the uncertainty of the distance measurements σ_r from Section 2-3 also a maximum likelihood approximation can be used to solve Equation (3-2).

Although used in multiple implementations there is a limitation why the equations from Equation 3-2 can not be used directly. In Equation (3-2) it is assumed that the positions (x_i, y_i) for $i = 1, 2, 3$ are known. As stated earlier this is not the case in this thesis, because those beacons are moving mobile robots them self. It is still possible to create equations for every measured distance $\bar{r}, i, j$ between mobile robots Z_i and Z_j, but every equation now have only unknown variables on the right hand as

$$\begin{aligned} \bar{r}_{1,Z}^2 - \bar{r}_{2,Z}^2 &= 2\hat{x}_Z(\hat{x}_2 - \hat{x}_1) + 2\hat{y}_Z(\hat{y}_2 - \hat{y}_1) + \hat{x}_1^2 - \hat{x}_2^2 + \hat{y}_1^2 - \hat{y}_2^2 \\ \bar{r}_{1,Z}^2 - \bar{r}_{3,Z}^2 &= 2\hat{x}_Z(\hat{x}_3 - \hat{x}_1) + 2\hat{y}_Z(\hat{y}_3 - \hat{y}_1) + \hat{x}_1^2 - \hat{x}_3^2 + \hat{y}_1^2 - \hat{y}_3^2 \end{aligned}, \tag{3-3}$$

This will create a non-linear equation which can not be solved by a linear state estimator any more. The set of equations now contains all non-linear equations possible. Meaning all

functions resulting from all combinations of $\bar{r}_{i,Z}^2 - \bar{r}_{j,Z}^2$, where $i \neq j$. The use of this complete set of equations is called Multi-robot trilateration.

The second part of this section is about why Equation (3-3) is not sufficient to solve the trilateration problem. As said the result of solving the multiple equations of the Multi-robot trilateration is a shape of the group of robots like a graph. The graph can only be used to estimate the distances $\hat{v}_{i,j}$ from and between robot Z_0 and robots Z_i. Still multiple solutions of the relative position of the other mobile robots exist, because the estimated orientation $\hat{\theta}_i$ is arbitrary as explained with the help of Figure 3-5.

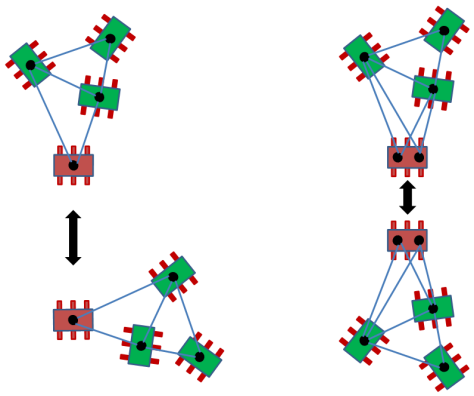


Figure 3-5: The problem of knowing only the distances between mobile robots. The graph has multiple solutions, for the orientation θ. Even with 2 sensors the graph can mirror.

Figure 3-5 shows four times the same estimated graph of the robot group with the red coloured robot as Z_0. One can see on the left picture that with only knowing the distance between all mobile robot the resulted graph can freely rotate around Z_0 giving it infinite solutions. Also when two antenna's are used on a single robot the shape of the group has two solutions which are mirrored. Therefore the distances $\hat{v}_i$ have one solution, but the orientations $\hat{\theta}_i$ are arbitrary. In the next section a method called dead-reckoning is added to find a single solution for the orientations $\hat{\theta}_i$.

3-3 Dead-reckoning as necessary additional measurement

The previous section showed a method to estimate all the distances $\hat{v}_i$ from the robot Z_0 towards and between the neighbouring robots Z_i. The remaining issue was that the estimated orientations $\hat{\theta}_i$ have infinite solutions. This section gives a solution in the form of using an additional measurement, called dead-reckoning. These measurements create an estimate of the walked path $\bar{V}_i$ of each mobile robot Z_i. Dead-reckoning in this case is performed by measuring the headed angle $\bar{u}_\theta$ and distance travelled $\bar{u}_v$ at each time step k. The sensors which can be used to measure both are discussed in Appendix B. By adding dead-reckoning a mobile robot can use the history of its walked path $\bar{V}_i$ together with the according distance measured at

the previous positions $\bar{r}_{i,j}(k)$. Doing so creates additional measurements at points with known positions in the local coordinate system. These extra known measurement points can be used as additional points from which the distance towards the target is known. By having these points a trilateration can be performed to estimate the position of the target as seen in Figure 3-6.

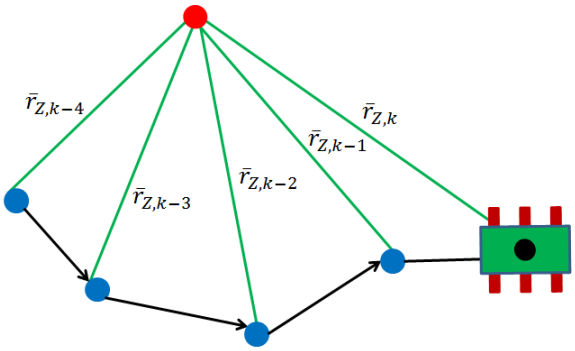


Figure 3-6: Use of additional measurements from dead-reckoning to create multiple known measurement points.

A problem is that the mobile robot to localize Z_i also moves as given by Figure 3-7. Solutions to deal with both movements of the mobile robots are discussed in Sections 3-4 and 3-5.

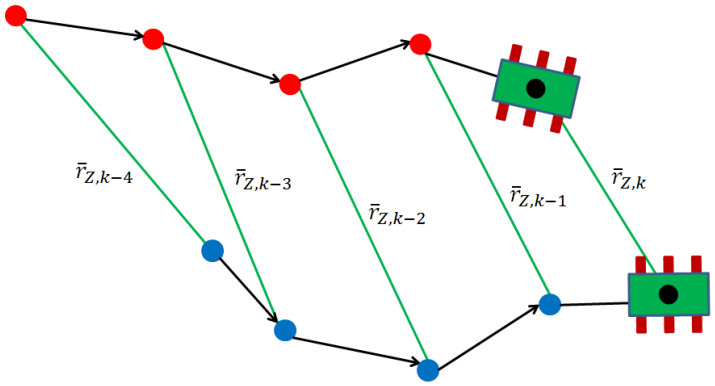


Figure 3-7: A case where two mobile robots are moving and performing dead-reckoning. Here at each time k a distance measurement was taken.

From one's intuition it can be seen that the problem about the infinite solutions of the orientation is solved, but for the mirroring problem it is also necessary not to walk in a straight line (non-collinear movement). One other necessity is to keep walking. A mathematical prove for both requirements is given in [1].

The walked path of a mobile robot Z_i is given by a matrix $\tilde{V}_i = \begin{bmatrix} V_0 & V_1 & \cdots & V_m \end{bmatrix}$, where

M is the amount of time samples ago a position is remembered. The point V_0 is the current position, which is always $(0, 0)$. The other vectors V_m are the previous positions in the local coordinate frame of the mobile robot, see Figure 3-8 for a more clear explanation.

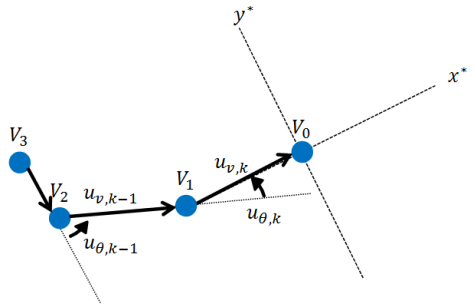


Figure 3-8: An example of the previous walked path given by the matrix $\tilde{V}_i = [V_0 \quad V_1 \quad \cdots \quad V_m]$. The point V_0 is always on the origin and position V_1 is always aligned with the x-axis, which is also the current heading of the mobile robot.

As one might sees, the dead-reckoning points V_m are vectors in a Cartesian coordinate system in stead of polar coordinates, which are normally used for the local coordinate system. Choosing a Cartesian coordinate system for $\tilde{V}$ does not have an influence on the position estimation, but is an easier way to add vectors. The complete matrix $\tilde{V}$ is updated at each new time step k by the following equation.

$$\tilde{V}_{k+1} = \left[\begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad R(-\bar{u}_{\theta,k})[\tilde{V}_{k,0\to M-1} - f_{p\to c}\left(\begin{bmatrix} \bar{u}_\theta \\ \bar{u}_v \end{bmatrix} \right) [1]^{1,M-1}] \right] \tag{3-4}$$

In Equation (3-4) $\tilde{V}_{k,0\to M-1}$ is equal to the part of the previous matrix $\tilde{V}_k$ from the first vector to the second last and the matrix $R(\theta)$ stands for the rotation transformation matrix based on the angle θ. The values for $\bar{u}_\theta$ and $\bar{u}_v$ are the measured values of control inputs u_θ and u_v. The measured values are used, because the given command might differ from the actual command by interacting with the environment and measurement noises. Think about walls or uneven ground. Furthermore the formula $f_{p\to c}()$ was given in Section 2-2 and transforms polar coordinates to Cartesian coordinates.

Like the distance measurements between two antenna's, the measured dead-reckoning points are also a bit uncertain. Dead-reckoning uncertainty is a result of measurement noises on the measured travelled path $\bar{u}_v$ and headed direction $\bar{u}_\theta$. The according noise powers σ_v and σ_θ result in uncertainty regions of the previous remembered positions, which can be seen in Figure 3-9.

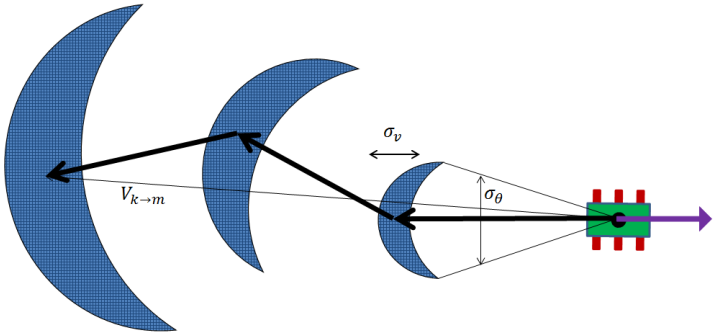


Figure 3-9: The commands for the direction of a Zebro robot, where no simultaneous forward/backward and turning movements is considered. Each time a additional tracked path is added the uncertainty region of a dead-reckoning point increases.

As one can see the uncertainty regions increase for larger m and are "banana" shaped. The banana-shape is a result of the polar like measurements $\bar{u}_v$ and $\bar{u}_\theta$ [25]. The banana-shaped uncertainty regions of the dead-reckoning will also result in banana-shaped like uncertainty regions of the position estimation of the other mobile robots. These banana-shaped uncertainty regions can be better captured by using polar states than using Cartesian states for all the positions to estimate [25].

Summarized, this section presented extra measurements to deal with the problem of the unknown orientation of the trilateration in Section 3-2. These added measurements are remembered positions of the walked path of a mobile robot in its own local coordinate system, called dead-reckoning. Dead-reckoning uses the measurements from the walked distance $\bar{u}_v$ and headed direction $\bar{u}_\theta$. Both measurements are used to define a travelled path given by positions saved in the matrix $\tilde{V}$. Due noise on $\bar{u}_v$ and $\bar{u}_\theta$ the uncertainty regions of the dead-reckoning positions in $\tilde{V}$ are 'banana'-shaped. Two methods that use these dead-reckoning data for a localization are given in the next two sections. The first makes use of Multi-dimensional scaling and the second uses a method called "relative pose estimation".

3-4 Localization with Multi-dimensional scaling

In this section the first proposed localization algorithm is presented. The localization algorithm uses Multi-dimensional scaling (MDS) to solve the unknown orientation problem of the Multi-robot trilateration (MRT). Therefore the algorithm combines the dead-reckoning data from Section 3-3 and the MRT of Section 3-2. The algorithm uses a modification of the weighted-MDS localization algorithm used in [24]. First a discription of the general MDS localization [26] is given and afterwards the modified weighted-MDS localization is explained.

Multi-dimensional scaling localization in general is a method that only uses measured distances between points X_i to create an estimated distance graph by using singular value decompositions. As given in [24] a weighted variant of the MDS gives better solutions compared to the

ordinary MDS localization algorithm. The weighted-MDS algorithm is an optimization algorithm that minimizes the difference between the measured and estimated distances, given by the so called STRESS function [26].

$$C(\Psi) = \sum_{i \neq j}^{N} w_{i,j}(\bar{r}_{i,j} - d_{i,j}(\Psi))^2 \tag{3-5}$$

In Equation (3-5) N is the amount of points to localize and the following equation describes the different variables.

$$X = \begin{bmatrix} X_1 \; X_2 \; \cdots \; X_N \end{bmatrix} \tag{3-6}$$

$$X_i = \begin{bmatrix} x_i \\ y_i \end{bmatrix} \tag{3-7}$$

$$\Psi = \begin{bmatrix} X_1^T X_1 \; X_2^T X_2 \; \cdots \; X_N^T X_N \end{bmatrix} \tag{3-8}$$

With Ψ and X known a distance matrix D holding the estimated distances between each point X_i can be created and is calculated by the following function

$$D = \Psi e^T - 2X^T X + e\Psi^T, \tag{3-9}$$

where e is a matrix of size 1 x N filled with ones. Every value $d_{i,j}$ in Equation (3-5) is a pivot in the distance matrix D at the position (i, j) presenting the distance between Z_i and Z_j. Parallel to the distance matrix D a matrix R exists that contains the measured distances $\bar{r}_{i,j}$ at the same corresponding pivots of $d_{i,j}$ in D. The third and last matrix needed is a weighting matrix W which is the same size as matrices R and D. The matrix W contains the weightings given to each distance measurement $\bar{r}_{i,j}$, where $w_{i,j} = 1/(\frac{1}{10^\beta}\sigma_I \cdot \bar{r}_{i,j})$. The weightings are chosen to be the inverse of the measurement noise power, because like the maximum likelihood estimation a high weighting is given to measurements more certain [26]. The weightings for the distances not measured are given a zero weighting. With the weighting matrix W, distance matrix D and distance measurement matrix R known a greatest descent algorithm can be used to optimize the cost function of Equation (3-5). The update function of each iteration step in the greatest descent algorithm is given in [26]. The result is a matrix $\hat{X}$ containing the estimated positions of the points X_i, with $i = 1, 2, ...N$. These points form a distance graph like the one of the linear trilateration from Section 3-2. It holds the same problem of not knowing the angle θ, therefore the following modification is proposed.

Therefore the first step of the modification is using the previous distance measurements between mobile robots, green lines/triangles in Figure 3-10. The second is to translate the dead-reckoning positions in matrix $\tilde{V}$ into distances from each other, given by the black lines in Figure 3-10.

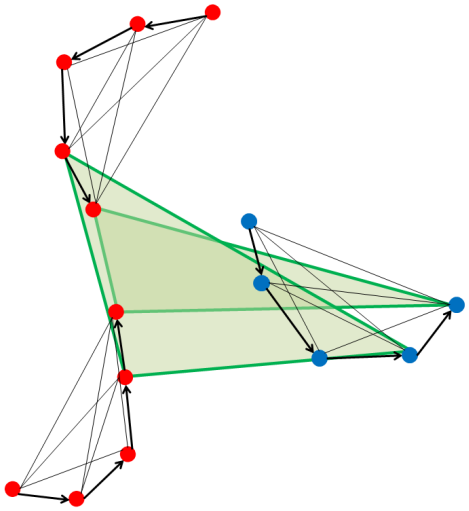


Figure 3-10: The resulted distance graph with the use of dead-reckoning data and previous measured distances. The dead-reckoning data is translated into distances between the dead-reckoning points (black lines) and the previous measured distances from the MRT create the green triangles.

Now it is like there are distance graphs for each mobile robot based on dead-reckoning. These graphs can be placed relative to each other by minimizing the distance errors between the mobile robots. If the dead-reckoning points (blue) of the mobile robot in question Z_0 are chosen to be constant the orientation θ of the other mobile robots can be found. That is, because the orientation of the dead-reckoning points of mobile robot Z_0 itself is known and the other mobile robots their positions (red) will be rotated according to those dead-reckoning points. The complete graph (red) will now rotate around the blue graph of Z_0 until it finds a best fit with the blue graph.

The use of MDS has been proven to be usable for localization of mobile robots [24, 27], but it was never used in a setting without static beacons with known positions. Therefore a modification was needed to create a localization algorithm with MDS to find positions relative to each mobile robot. These modifications result in a MDS localization algorithm which also uses dead-reckoning measurements and a weighted-MDS. A second attempt for the relative localization problem uses "relative pose estimation" and is given in the next section.

3-5 Relative pose estimation

In the previous section a localization method was presented based on Multi-dimensional scaling. In this section another proposed localization method to solve the relative position estimation is handled and is based on a localization method called "relative pose estimation" (RPE) [1]. The RPE was specifically designed to deal with the same localization problem where there are no static beacons with known positions, only distance and dead-reckoning measurements. In

this section first the equations to solve the RPE problem are handled. Next a short discussion on why the original solving method in [1] is not used. Therefore a new method is presented which uses the equations of the RPE in combination with the Multi-robot trilateration from Section 3-2. This new localization method will combine the two methods by using an Unscented Kalman filter in section 3-6-2.

The relative pose estimation uses the travelled path captured by dead-reckoning from Section 3-3 and the distance measurements from Section 2-3. The basic idea of this method is made clear with the use of Figure 3-11.

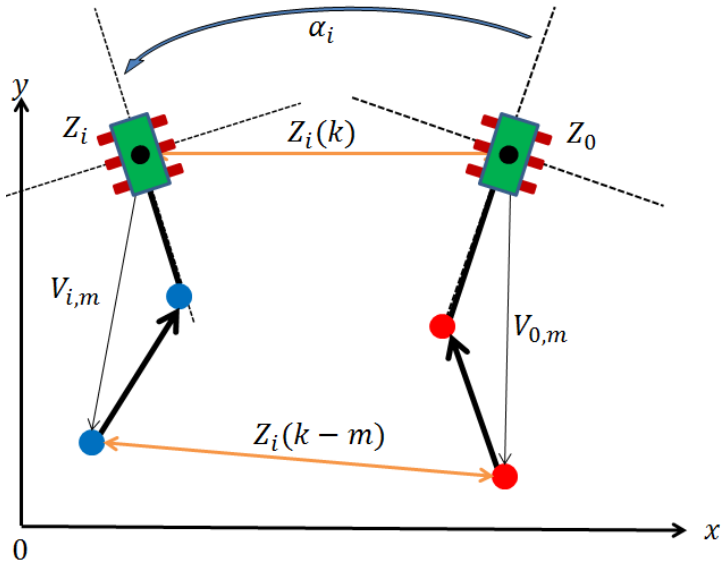


Figure 3-11: The relative position and movement of mobile robot Z_i seen from mobile robot Z_0, where $Z_i(k-m)$ presents the relative position of Z_i from m time samples ago. The vectors $V_{0,m}$ and $V_{i,m}$ are the positions of Z_0 and Z_i respectively from m time samples ago.

In this figure two mobile robots walk relative to one another. Mobile robot Z_0 is the robot that tries to localize mobile robot Z_i. Robot Z_0 knows the relative positions of the walked path $\tilde{V}_{0,m}$ of itself up till time m ago and the same for $\tilde{V}_i$ of robot Z_i. How to create and how the matrices with the tracked positions $\tilde{V}_0$ and $\tilde{V}_i$ are constructed is explained in Section 3-3. The angle α_i is the pose angle of Z_i towards Z_0. The next equation holds for all times samples m ago.

$$Z_i(k-m) = Z_i(k) + R(\alpha_i)V_{i,m} - V_{0,m}, \tag{3-10}$$

with $R(\alpha_i)$ as the rotation transformation matrix to rotate with angle α_i. This equation looks much like the relative motion equation from Equation (2-3), which is no coincidence. The main difference is the elimination of the rotation transformation of the rotation Z_0 has made itself. Equation (3-10) can be converted into Equation (3-11) [1], where also the distance measurements $\bar{r}_{0,i}$ are implemented. The value $\bar{r}_{0,i,m}$ stands for the measured distance between

mobile robot Z_0 and Z_i from m time samples ago.

$$(Z_i(k) - V_{0,m})R(\alpha_i)V_{i,m} - Z_i(k)^T V_{0,m} = 0.5(\bar{r}^2_{0,i,m} - \bar{r}^2_{0,i,0} - V^T_{i,m}V_{i,m} - V^T_{0,m}V_{0,m}) \quad (3\text{-}11)$$

Equation (3-11) holds on the right side all distance measurements and the dead-reckoning data from both mobile robots from the present k to time $k-m$. On the left side the states position Z_i and pose embedded $R(\alpha_i)$ are positioned. Also on the left hand side measured dead-reckoning points are placed. For now this is not considered a problem, because the largest measurement noises do not come from dead-reckoning. Later simulations are done on how much dead-reckoning noise has an influence, see Section 3-7-3. Roumeliotis proved in [1] that at least five of these equations need to be solved to have a single solution.

Roumeliotis used a numerical exact solver to solve the problem created by Equations (3-11), which will not be the solver in this thesis. The first argument is that the algorithm with the exact solver used by Roumeliotis is inaccurate for relative distance noise as large the ones in this thesis. In this thesis one should think about relative distance noise variance σ_r of more than 10%, see Section 2-3. The simulation in [1] uses distance measurements around 1.5 meter. A distance noise variance of $\sigma_r = 10\%$ results in an distance measurement noise variance of at least $\sigma_r = 15cm$. From results in [1] $\sigma_r = 15cm$ creates an RMS orientation θ error of one radian or more, which is already a lot. A case of $\sigma_r > 20\%$ is not even considered.

A solution to let the RPE have a more accurate orientation θ estimation is to make use of the other mobile robots of the swarm. As in Section 3-2 mentioned one can create more accurate distance estimations v by using the Multi-robot trilateration (MRT). A method using extra distance measurements between multiple robots is not mentioned by Roumeliotis. A solution was found in combining the relative motion model of Section 2-2, Multi-robot trilateration equations of Section 3-2 and the RPE equations (3-11) in to a Kalman filter setting. How and why this is done will be the topic of the next Section.

3-6 A Kalman Filter design to use the relative pose estimation

In the previous section it was stated that the relative pose estimation needs the Multi-robot trilateration to create a more precise estimation of the distances $v_{i.j}$ between the mobile robots to improve the orientation θ estimation. Estimate the relative positions using the RPE, MRT is done with a recursive state estimator. In the end of this section a unscented Kalman Filter is chosen as the state estimator. First the non-linear discrete state-space representation of the system is handled and in the second part the choice on why using the unscented Kalman filter is discussed.

3-6-1 A non-linear state-space representation

The state-space representation of the system is discrete, non-linear and can be given as follows.

$$x(k+1) = f(x(k), \bar{u}(k+1)) + \nu_x \tag{3-12}$$
$$y(k+1) = h(x(k+1)) + \nu_y \tag{3-13}$$

The main difference between an ordinary state-space representation is that Equation (3-15) uses measured control inputs $\bar{u}$ from time step $k+1$ as the control input. The control input noise of the measured control input $\bar{u}$ is added to the noise vector ν_x. The noise models for ν_x and ν_y are captured by the noise variance matrices Q and R formulated by the following relation.

$$E\left[\begin{bmatrix}\nu_x(k)\\ \nu_y(k)\end{bmatrix}\begin{bmatrix}\nu_x(k)^T & \nu_y(k)^T\end{bmatrix}\right] = \begin{bmatrix}Q(k) & 0\\ 0 & R(k)\end{bmatrix} \tag{3-14}$$

Considered that there is no correlation between the state and control input noise and the measurement noises. Also ν_x and ν_y are considered to be Gaussian white noise. As one might have seen the state-update function $f(x(k), \bar{u}(k+1))$ is non-linear and is based on the relative motion model given in Section 2-2.

$$\begin{bmatrix}\theta_i(k+1)\\ v_i(k+1)\\ \alpha_i(k+1)\\ u_{\theta,0}(k+1)\\ u_{v,0}(k+1)\\ u_{\theta,i}(k+1)\\ u_{v,i}(k+1)\end{bmatrix} = \begin{bmatrix} f_{c\to p}\left(R(-u_{0,\theta})[f_{p\to c}\left(\begin{bmatrix}\theta_i(k)\\ v_i(k)\end{bmatrix}\right) + R(\alpha_i) f_{p\to c}\left(\begin{bmatrix}\bar{u}_{i,\theta}\\ \bar{u}_{i,v}\end{bmatrix}\right) - f_{p\to c}\left(\begin{bmatrix}\bar{u}_{0,\theta}\\ \bar{u}_{0,v}\end{bmatrix}\right)]\right)\\ \alpha_i(k) - \bar{u}_{0,\theta}(k)\\ \bar{u}_{\theta,0}(k+1)\\ \bar{u}_{v,0}(k+1)\\ \bar{u}_{\theta,i}(k+1)\\ \bar{u}_{v,i}(k+1)\end{bmatrix} \tag{3-15}$$

The position states of Z_i are θ_i and v_i, which are polar coordinates. The functions $f_{c\to p}()$ and $f_{p\to c}()$ are used to convert between the polar and Cartesian coordinate system. Arguments why a polar coordinate system is used were discussed in Section 3-3 and in [25]. The third state α_i is the relative pose of robot Z_i, see Figure 3-11. As one also can see, the control inputs (walked distance $u_{i,v}$ and direction $u_{i,\theta}$) are present in the state vector. Doing this one can add noise on the control inputs, where the state-noise matrix Q contains information about this control input noise [28] given in Equation 3-16

$$Q(k) = \begin{bmatrix}\epsilon & 0 & 0 & & & & \\ 0 & \epsilon & 0 & & & 0 & \\ 0 & 0 & \epsilon & & & & \\ & & & \sigma_u & 0 & 0 & 0\\ & 0 & & 0 & \sigma_{v,rel}u_{v,0}(k) & 0 & 0\\ & & & 0 & 0 & \sigma_u & 0\\ & & & 0 & 0 & 0 & \sigma_{v,rel}u_{v,i}(k)\end{bmatrix}, \tag{3-16}$$

where $\sigma_{v,rel}$ is the relative noise power of the travelled distance. To ensure positive definiteness of the covariance matrix P_{xx} in the unscented Kalman filter later on, at least positive nonzero numbers for ϵ should be chosen.

The previous part outlined the state-update equation $f(x(k), \bar{u}(k+1))$. The next part will present the equations for the measurement update of Equation (3-13). The measurements are divided into two types: y^m represents the measurements to perform the Multi-robot trilateration (MRT) from Section 3-2 and y^r the relative pose estimation (RPE) of Roumeliotis in Section 3-5.

The equations for the MRT are based on Equation (3-1), where the right hand side is y^m as the updated measurement and left hand side is $\bar{y}^m$ as the measured part. The measurement prediction part of the MRT y^m is given as follows.

$$\begin{aligned} y^m_{i,j,k}(k+1) = \quad & ((x_i(k+1) - x_j(k+1))^2 + (y_i(k+1) - y_j(k+1))^2) \\ & -((x_i(k+1) - x_k(k+1))^2 + (y_i(k+1) - y_k(k+1))^2)' \end{aligned} \tag{3-17}$$

where the positions $x_i(k+1)$ and $y_i(k+1)$ are the Cartesian coordinates from the updated position state vector $x_i(k+1)$. The updated measurement $y^m_{i,j,k}(k+1)$ uses the position differences between robots $Z_i \to Z_j$ and $Z_i \to Z_k$. Therefore Equation (3-17) is used for all combinations of i,j and k, where $i \neq j \neq k$. In total this creates $\binom{Z+1}{3}$ amount of equations, with Z as total amount of the robots used for the MRT.

The measured part of the MRT $\bar{y}^m$ is given below.

$$\bar{y}^m_{i,j,k}(k+1) = \bar{r}_{i,j}(k+1)^2 - \bar{r}_{i,k}(k+1)^2 \tag{3-18}$$

In equation (6-1) $\bar{r}_{i,j}(k+1)$ and $\bar{r}_{i,k}(k+1)$ stand for the measured distances between robots $Z_i \to Z_j$ and $Z_i \to Z_k$ respectively. Information of the noise power of each measured $\bar{y}^m_{i,j,k}(k+1)$ is preserved in the measurement noise variance matrix R^m where values $R^m_{i,j,k}$ are variables on the diagonal.

$$R^m_{i,j,k} = (\bar{r}^2_{i,j} \log(10) \frac{1}{5\beta} \sigma_I)^2 + (\bar{r}^2_{i,k} \log(10) \frac{1}{5\beta} \sigma_I)^2, \tag{3-19}$$

with $\sigma^*_r = \bar{r}^2 \log(10) \frac{1}{5\beta} \sigma_I$ from Section 2-3. Also discussed in Section 2-3 the use of the distances $\bar{r}$ in stead of RSSs $\bar{I}$ leads to a biased state estimator. It was possible to rewrite Equations (3-17) and (6-1) into an equation that uses $\bar{I}$ in stead of $\bar{r}^2$, but it creates worse position estimations than using the distance $\bar{r}^2$.

Secondly, the updated RPE measurement y^r and measured RPE values $\bar{y}^r$ are outlined, using (3-11).

$$y^r_{i,m}(k+1) = (Z_i(k) - V_{0,m})^T R(\alpha_i) V_{i,m} - Z_i(k)^T V_{0,m} \tag{3-20}$$

$$\bar{y}^r_{i,m}(k+1) = 0.5(\bar{r}^2_{0,i,m} - \bar{r}^2_{0,i,0} - V^T_{i,m} V_{i,m} - V^T_{0,m} V_{0,m}), \tag{3-21}$$

where i is the identity of the other mobile robot Z_i. The vector $V_{i,m}$ is the m-th vector in the tracked path matrix $\tilde{V}_i$ of Z_i. The matrix $\tilde{V}_i$ contains the positions of the walked path by performing dead-reckoning, discussed in Section 3-3. The amount of equations needed is $Z \cdot M$, with Z as all the mobile robots in the neighbourhood and M as the amount of remembered

dead-reckoning positions. The according diagonals of the measurement noise variance matrix R^r are given by

$$R^r_{i,m} = 0.5 \left((\bar{r}^2_{i,j} \log(10) \frac{1}{5\beta} \sigma_I)^2 + (\bar{r}^2_{i,k} \log(10) \frac{1}{5\beta} \sigma_I)^2 + 2(\sigma_v \sum_{m=0}^{M-1} |V^{vec}_{i,m}|)^4 + 2(\sigma_v \sum_{m=0}^{M-1} |V^{vec}_{0,m}|)^4 \right) \tag{3-22}$$

The summation is used to calculate the total distance travelled after m time samples, where the value $|V^{vec}_{i,m}|$ refers to the length of the path taken between the time $k-m$ and $k-m+1$.

Now equations for the Multi-robot trilateration and relative pose estimation measurements are given both can be stacked in a single vector as follows.

$$y(k+1) = \begin{bmatrix} y^m_{0,0,0}(k+1) \\ \vdots \\ y^m_{i,j,k}(k+1) \\ y^r_{0,0}(k+1) \\ \vdots \\ y^r_{i,m}(k+1) \end{bmatrix} \quad \bar{y}(k+1) = \begin{bmatrix} \bar{y}^m_{0,0,0}(k+1) \\ \vdots \\ \bar{y}^m_{i,j,k}(k+1) \\ \bar{y}^r_{0,0}(k+1) \\ \vdots \\ \bar{y}^r_{i,m}(k+1) \end{bmatrix} \quad R = \begin{bmatrix} R^m & 0 \\ 0 & R^r \end{bmatrix} \tag{3-23}$$

An important aspect to have a decent state estimator is to have one optimal global solution of the states. One global optimum for the states in this case also refers to one optimal solution of the difference between the updated measurement $y(k+1)$ and measured $\bar{y}(k+1)$.

$$\min_{x,y,\alpha} \quad C = \|y(k+1) - \bar{y}(k+1)\| \tag{3-24}$$

The cost function should be preferably convex, but it should have at least has no local minima (unimodal). Local minima could indicate that more solutions exist for the position x, y and pose α. The cost function of Equation (3-24) is a combination of the measurements used for MRT and RPE. In case of the MRT the combination of Equations (3-17) and (6-1) results in a quadratic cost function, which has a single solution of the distances between the mobile robots. The solution for the distances always converges towards this minimum and the RPE part will then use this solution of the distances to estimate an optimal pose. An empirical analysis was used to prove unimodality of the RPE cost function given optimal distances measurements. This was done by constructing the cost function with at least five measurements, which is according to [1] the minimal amount of measurements needed to have a single solution. The cost function value C was visualized by taking cut planes of the three dimensional cost function with the pose angle constraint $\alpha = [-\pi, \pi]$. It can be concluded empirical that the RPE part of the cost function (3-24) is not convex, but unimodal. This only counts for observing one other mobile robot. An analysis to see if local minima exist with multiple robots is not done, because of the high amount of states to be analysed is $Z * 3$. From simulation results it seems that the total localization algorithm has some local minima solutions, which will be discussed in Section 3-7-4. Based on arguments given earlier polar state coordinates are used in stead of the Cartesian coordinates. The conversion of Cartesian coordinates to polar coordinates will not

have an effect on the convexity and/or global minima characteristics, if the polar coordinate is bounded by $\theta = [-\pi, \pi]$.

With all the equations from this section a non-linear discrete state-space representation is created. The system now contains a state vector with the positions of all the neighbour mobile robots Z_i in polar coordinates. It also contains the control inputs $u_{i,v}$ and ui, θ of robot Z_0 and neighbour robots Z_i. The state vector has an according state noise variance matrix Q given in Equation (3-16) containing information about the input noise power of $u_{i,v}$ and $u_{i,\theta}$. The output vector $y(k+1)$ contains the state-updated measurements based on MRT and RPE. The equations also show that there is one optimal solution if one mobile robot has to localized, but not proven for localizing multiple robots. Later results show that with multiple mobile robots the position estimation still converges towards the global minimum. So a state estimator can be used to create the best approximation of the mobile robot Z_i its position θ_i and v_i. The choice on what state estimator is used and a short description of this the chosen Unscented Kalman filter is discussed in the next section.

3-6-2 The Unscented Kalman Filter

In the previous section the Multi-robot trilateration and relative pose estimation are combined into a single state-space representation. For estimating the positions of the neighbouring mobile robots a state estimator will be used. Why a unscented Kalman filter for the state estimator is chosen will be the topic of this section.

The system given by Equation 3-15 in the previous section shows a non-linear state-update function. Due to the non-linearity a linear Kalman filter can not be used. Using a sequential Monte Carlo filter like the particle filter is also not preferred, because of the high amount of states ($Z * 5 + 2$) to estimate. Mainly two Kalman filters are used to cooperate with defined non-linear state-space systems with assumed Gaussian white noise: The Extended Kalman filter (EKF) [29] and the unscented Kalman filter (UKF) [30]. The EKF linearizes the system around the newly predicted state to create a linear state-space representation. The UKF does not uses a linearization and has a better non-linear state estimation of the system than the EKF [30]. Also the UKF has the same computational complexity as the EKF. A short summary with the use of Figure 3-12 will explain the principles of the UKF.

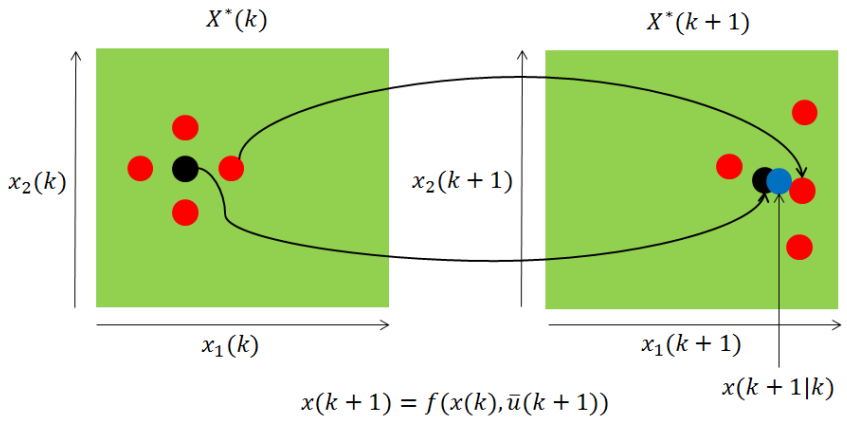


Figure 3-12: A picture to give a notion of the update and filtering steps of the UKF.

If $x(k)$ (black dot) is the previous estimated state then based on the spread of the previous estimated state covariance matrix $P_{xx}(k)$ additional states $X^*(k)$ are created (red dots). Doing so creates a cloud of states $X^*(k)$ which represents the uncertainty of the previous prediction $x(k)$. For every state in $X^*(k)$ the next state is predicted by the state-update function $f(x(k), \bar{u}(k+1))$. The unscented transform calculates the mean predicted state $x(k+1|k)$ using all predicted states $X^*(k+1)$. The same is done with the measurement update $y(k+1|k)$, where the state cloud $X^*(k)$ creates a cloud of predicted measurements $Y^*(k+1|k)$ measurement update using equation $h(x(k))$. The cloud of predicted measurements $Y^*(k+1|k)$ also use the unscented transform to calculate the mean predicted measurement $y(k+1|k)$. In the end the Kalman gain is calculated by

$$K = \frac{P_{xy}}{P_{yy}}, \tag{3-25}$$

where P_{xy} is the cross-correlation matrix between the states in $X^*(k+1)$ and $Y^*(k+1)$ and P_{yy} is the auto-covariance matrix of $Y^*(k+1|k)$. The Kalman gain K will be used to update the previous states estimation and state-covariance matrix like an ordinary Kalman filter. For a more detailed explanation of the UKF reading the paper of E.A. Wan and R. Van Der Merwe [30] is suggested.

A downside of the UKF is that it has a problem with the possibility of creating a non-positive definite matrix for P. Therefore a "scaled UKF" is used [31]. Furthermore in [32] a typical problem of using Kalman filters for localization was outlined:" In the application of passive target tracking, however, because of the large initial error and weak observability of the system, the standard UKF also shows its weakness in robustness, convergence speed, and tracking accuracy. In this correspondence, we propose an iterated UKF to address these problems." This was also witnessed in this thesis with in the next section a mirrored position estimation as a result.

In this section the best state estimator for the non-linear state-space system of the previous section was discussed. The result is the scaled-UKF. The completed localization algorithm

is a combination of a scaled UKF, the relative pose estimation and Multi-robot trilateration and will be called the Multi-Robot Relative Pose Localization (MRRPL). The MRRPL will be compared with the modified weighted-MDS localization in the next section.

3-7 Localization algorithm results

Previously in this chapter two localization methods were presented, the weighted-MDS and MRRPL localization. These methods will be compared and analysed in this section. As criteria which method is better the estimation error of the orientation θ and relative error of the distance v are used. In the simulation a suitable walked path is chosen, which will be the same every simulation run. Later more detailed characteristics of the MRRPL are given by tuning the state noise variance matrix Q and measurement noise variance matrix R.

For the simulations a suitable movement of the mobile robots is used, which will be the same for every test. So no swarm controller is used yet, because the swarm controller its outcome depends on the performance of the position estimation and vice versa. This will result in an undesired correlation between the localization and the swarm controller when only the localization needs to be tested. The second requirement is not walking in a straight line as outlined in Section 3-5. Another requirement is that every mobile robot does not make the same movement parallel towards the other mobile robot, because that could result in multiple results due to a mirror effect. A control input that is the same every run and full fills the given requirements is given below

$$\begin{bmatrix} u_{\theta,i} \\ u_{v,i} \end{bmatrix} = \begin{bmatrix} \sin(k/(1+i\cdot 0.1)) \\ 0.2 \end{bmatrix}, \tag{3-26}$$

where i is the identity number $i = 1, 2, 3\ldots$ of mobile robot Z_i. This creates a movement as in Figure 3-13.

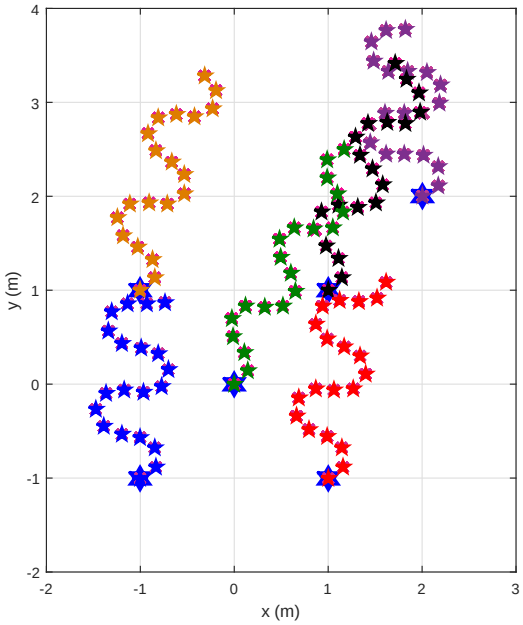


Figure 3-13: The constant path the mobile robot will take for the localization test. Blue large stars are the initial positions.

In the simulation the initial positions are known, this will still give a first indication of the performance of the different localization methods. At the end of this section is concluded that no clear data can be retrieved with unknown initial conditions in combination with the robot movement in Figure 3-13.

The performance of the localization is measured by the error of the position estimation of the orientation θ and distance v. The error of the orientation is given as the root-mean-squares orientation estimation error of all the mobile robots calculated by all the mobile robots themselves.

$$RMS_\theta = \sqrt{\frac{1}{N}\sum^{N}(\theta_{ij} - \hat{\theta}_{ij})^2}, \tag{3-27}$$

where $\theta_{ij} - \hat{\theta}_{ij}$ stands for the orientation error between mobile robot Z_i and Z_j. The value $N = Z(Z-1) \cdot T \cdot R_{\text{runs}}$ is the total amount of estimations, where T is the length of the simulation run and R_{runs} is the amount of simulation runs taken. For the distance a relative RMS error is used, because the same absolute error is better at larger distances than at shorter distances.

$$RMS_v = \sqrt{\frac{1}{N}\sum^{N}\left(\frac{\hat{v}_{ij} - v_{ij}}{v_{ij}}\right)^2} \tag{3-28}$$

In the next section the first simulation test is performed to compare the presented localization

algorithms, weighted-MDS and MRRPL.

3-7-1 Position estimation error distributions

With the simulation model of the previous section the two designed localization algorithms weighted-MDS and MRRPL are compared. The first topic of discussion is to see how the errors of the estimations are distributed for both localization algorithms. Both methods are tested with the use of $M = 10$ remembered positions by dead-reckoning. The result of the position estimation error of the orientation θ and relative distance v error is given in Figure 3-14.

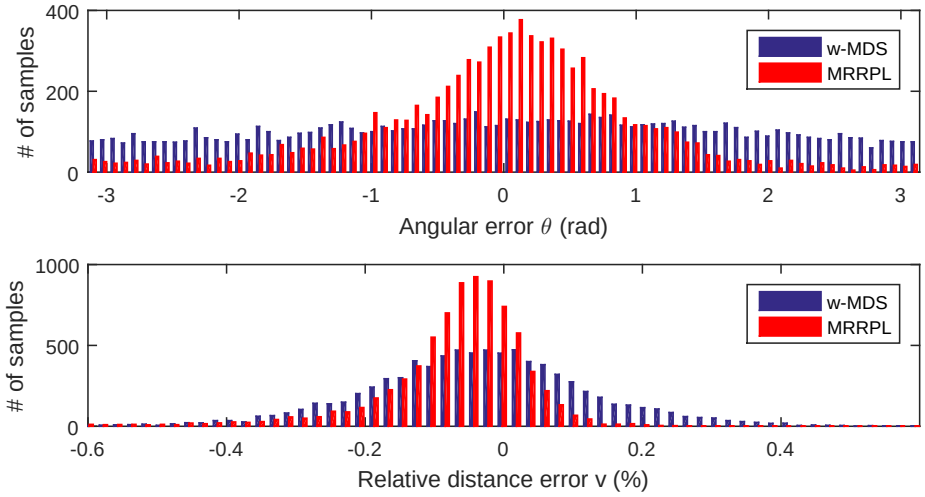


Figure 3-14: Error distribution of the weighted-MDS and MRRPL localization algorithm of $N_{runs} = 20$ runs of $T = 50$ long and 5 mobile robots. The noise parameters used are: $\sigma_\theta = 5°$, $\sigma_v = 10\%$ and $\sigma_I = 1$dB

The result shows an orientation estimation θ which is way worse using the weighted-MDS localization, because there is no sign of estimating an angular measurement. The reason might lie in the fact that the weighted-MDS is a non-convex optimization problem with possible wide minima for the orientation. The weighted-MDS uses a greatest descent optimization solver, which might have a problem dealing with such local minima. To solve this it might be smart to look into adaptations to create a steeper global minimum by relaxation of the optimization cost function [33] or make it more suitable for trilateration [27]. Although it might create some improvement is seems that the large orientation error is a more fundamental problem. The amount of time possible for this thesis was not enough to look more into the problems of the weighted-MDS localization. Therefore the focus lies on the MRRPL which already created a result that contains no irregularities.

With the localization algorithm chosen to be MRRPL the second topic of this section is the distribution of the distance error. In Figure 3-14 one can see that the error distribution of the

distance v is somewhat biased, meaning that most of the distance estimations is biased. Why this is the case was explained in Section 2-3. It was possible to modify the MRRPL to one with unbiased distance estimations, but as said in Section 2-3, this will create a worse position estimation.

From the results in Figure 3-14 it is concluded that the weighted-MDS is not the best localization method for now. Therefore the MRRPL is used for the swarm localization. In the next section some characteristics and insides from simulations of the MRRPL are given to improve the position estimation even more.

3-7-2 Results from tuning noise variance matrices Q and R

In the previous section the conclusion was that the MRRPL does a better position estimation than the weighted-MDS. In this section a more detailed look of the characteristics of the MRRPL is given. This is done by varying and tuning the noise variance matrices R and Q to improve the position estimation. When tuning the matrices R and Q, these matrices are considered to act like weighting matrices. An optimal tuning of R and Q is important especially in cases of non-linear systems like the one used for the MRRPL. The optimal R and Q are not simply found as proven in this section, because R and Q are heavily dependent on the amplitudes of distance measurement noise variance σ_I, see Equations (3-19) and (3-22). Therefore in this section also simulations are done to see what differences occur when the distance noise variance σ_I is varied.

Before tuning R and Q remember that the state noise variance matrix Q contained a control input noise part Q_u and state noise part Q_x. For the state noise variance small values were given for $Q_x = 0.001$. Now the matrix Q can be kept constant. A gain K_R will be placed on the noise variance matrix R, which becomes the varied parameter to tune the ratio between Q and R. The result for different K_R at different noise powers σ_I is given in Figure 3-15 below.

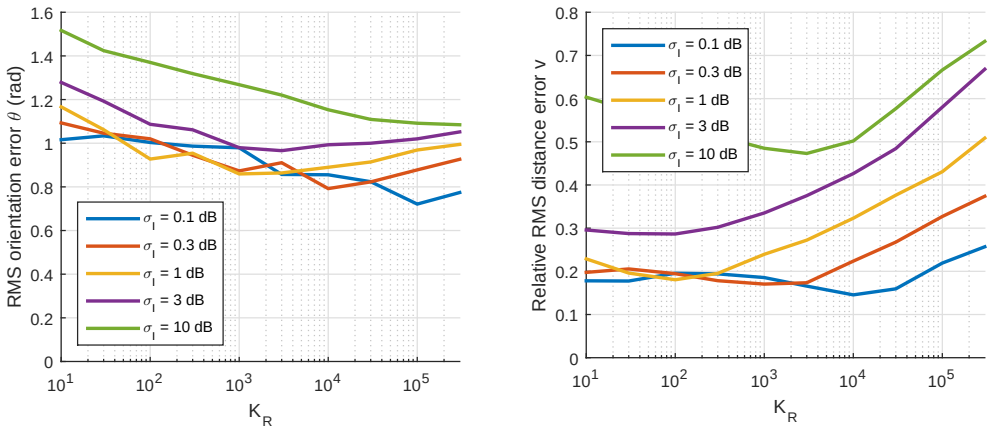


Figure 3-15: Different gains K_R on the noise variance matrix R tested for different distance measurement noises σ_I. The simulation ran for 50 samples 20 times with a group of 5 mobile robots with dead-reckoning noise parameters: $\sigma_\theta = 5°$,$\sigma_v = 0.1\%$.

As one can see there are different optimal gains for R for different distance measurement noises σ_I. The main surprise is that there is no clear trend between different σ_I and the optimal value for K_R. For example, for a low noise value $\sigma_I = 0.1$dB and a high noise value $\sigma_I = 10$dB the optimum of K_R is for both higher than with $\sigma_I = 1$dB. It will make it hard to find a exact definition for the optimal noise variances matrices Q and R. Solutions for a live tuning of the matrices R and Q will be discussed later.

Another ratio for the MRRPL that is important is the priority ratio between the MRT and RPE. The MRT has the function to create a better estimate of the distances v , which in return improves the orientation θ estimation of the RPE. The ratio is given by the parameter K_{ratio} and defined as follows.

$$R = \begin{bmatrix} \frac{1}{K_{\text{ratio}}} R^m & 0 \\ 0 & K_{\text{ratio}} R^r \end{bmatrix}, \tag{3-29}$$

The results are given in Figure 3-16 below.

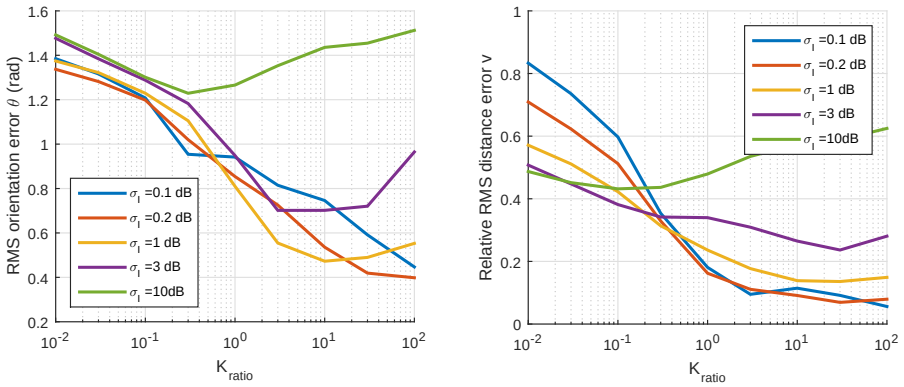


Figure 3-16: A relation between the importance of RPE and MRT at different amounts of distance measurement noise σ_I. The simulation ran for 50 samples 20 times with a group of 5 mobile robots with dead-reckoning noise parameters: $\sigma_\theta = 5°$,$\sigma_v = 0.1$. $K_Q = 0.001$,$K_R = 1$

In Figure 3-16 a relation between σ_I and the optimal K_{ratio} is shown. When the distance noise σ_I increases the optimal for K_{ratio} shifts to the left. This shift to the left means a higher priority of using the MRT than the RPE. Intuitively, this is correct, because a higher distance noise requires more help of the Multi-robot trilateration to improve the distance measurement v. Which is needed to perform a better relative pose estimation for the orientation estimation θ.

From the previous results it can be concluded that it is necessary to find live optimal noise variance matrices R and Q, because the optimal solutions are highly dependent on changes of distance noise standard variance σ_I. The value for σ_I will also change for changes in environment. A solution can be an Adaptive Kalman Filter (AKF) which also updates both noise variance matrices R and Q [34]. The disadvantage is that fact that an AKF modification for a UKF was not found in literature. Also the UKF was not specifically designed to deal with a state dependent noise parameter

$$\sigma_r^* = \bar{r}_{i,j}^2 \log(10) \frac{1}{5\beta} \sigma_I \tag{3-30}$$

Rewriting this equation into one that does not dependent on the distance only worsens the results, see Section 3-3. A solution might be another state-estimator for handling state-dependent noise variance matrices, such as given in [35].

Using the chosen parameters $K_R = 1000$ and $K_{ratio} = 10$ for a distance measurement noise of $\sigma_I = 3$dB a simulation is done to compare the MRRPL with only a relative pose estimation part (UKF-RPE) of the localization algorithm, which is basically the MRRPL without the Multi-robot trilateration. Doing this could conclude that the multi-robot trilateration does give better distance estimations v and therefore also better angle θ estimations. No K_{ratio} can be used for the UKF-RPE, therefore the same K_R will be used as for the MRRPL. The results are given in the table below.

Table 3-1: A comparison between MRRPL and a localization with only the RPE, which can tell if the added Multi-robot trilateration does indeed creates a better overall position estimation.

	UKF-RPE	MRRPL
RMS error θ (rad)	1.0943	0.7116
Relative RMS error v (%)	75.49	31.00

In Table 3-1 it can be seen that the addition of the Multi-robot trilateration reduces the distance error and therefore also the angular error. Compared with the original relative pose estimation algorithm of Roumeliotis in [1] this is an improvement, because relative distance errors above 20% are not even considered by Roumeliotis in [1] and if so that would create an angular error larger than 1 rad.

3-7-3 Dead-reckoning noise influences

In this section the influence of the all the measurement noises is analysed. To do so, if not varied, the following values will hold

$$K_R = 1000, \quad K_{\text{ratio}} = 10, \quad \sigma_\theta = 5^\circ, \quad \sigma_v = 0.1, \quad \sigma_I = 3, \quad N = 10, \quad \text{and} \quad \beta = 3.424 \quad Z = 5,$$

where N is the amount of measurements used to perform dead-reckoning. This is also the first parameter to change, because by intuition it sounds logic that a higher amount of measurements can improve the position estimation. The results are given below

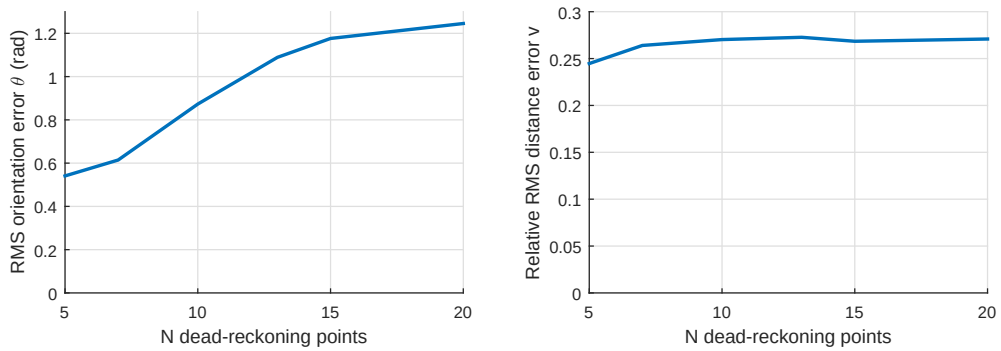


Figure 3-17: Localization performance with different amount of dead-reckoning points N.

Surprisingly the higher amount of dead-reckoning points worsens the position estimation of the angle θ. One or two causes might be involved. One is that the optimal ratios and amplitudes K_R and K_{ratio} also dependent on N. Therefore a new optimal also has to be found for different N. The second cause might be that the localization problem becomes too large such that the UKF of the MRRPL convergences too slow and an iterative UKF is needed [32], as discussed in Section 3-6-2.

Next an analysis of the influence by the dead-reckoning noise parameters σ_θ and σ_v is given. The results are given in Figures D-1 and D-4.

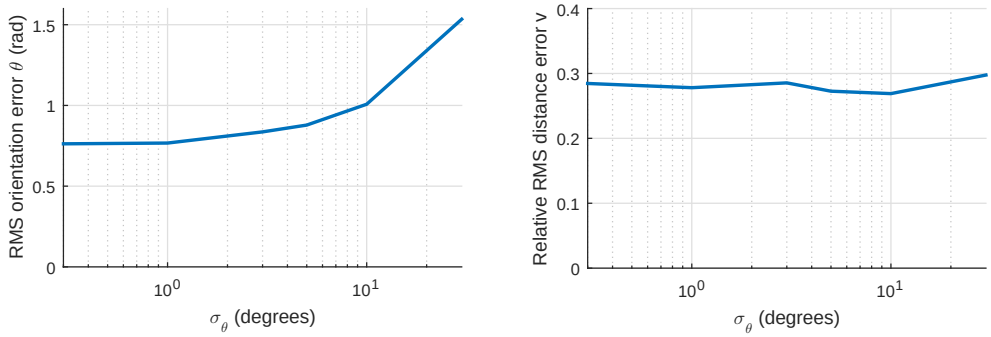


Figure 3-18: Localization performance with different amount of heading direction σ_θ noise for dead-reckoning.

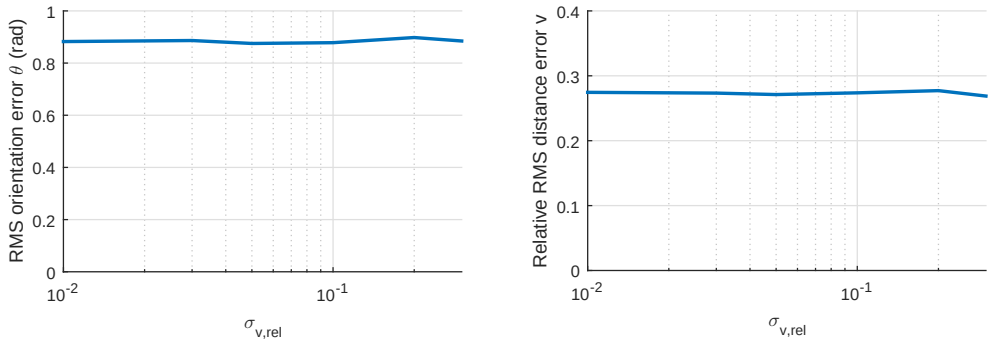


Figure 3-19: Localization performance with different amount of the travelled relative distance noise σ_v for dead-reckoning

Clearly seen from both simulations is that the only major differences are caused by a higher orientation noise σ_θ. This is from intuition quite straightforward, because if the walked angles are not accurately known the relative angle towards the others is also not accurate. What is surprisingly is that the noise on the travelled distance seems of no importance even at $\sigma_v = 30\%$.

In this section it is concluded that the amount of dead-reckoning points is surprisingly better than with less points. Also the performance of the dead-reckoning is almost of no importance. The important factor on the performance of the localization is therefore the distance noise variance σ_I of the distance measured by taking the RSS between the two antennas on different mobile robots.

All the simulations in the previous sections were performed with known initial positions. In

practice this might not be the case. For example if the communication between mobile robots fails or a new robot enters the neighbourhood of another mobile robot. Therefore in the next section the influence of unknown initial positions is discussed.

3-7-4 Unknown initial positions

In the previous simulations the initial state was known. This will not always be the case and a study needs to be performed to prove robustness if the state might become unknown during the mission. The results to compare will not be given by the estimated RMS error of θ and v, but looking if the estimated state always converges towards the real values. In Figure 3-20 a picture of the estimated positions and real positions from the perspective of each mobile robot is given. This picture is generated after a relatively long time, where the estimated positions were already quite converged.

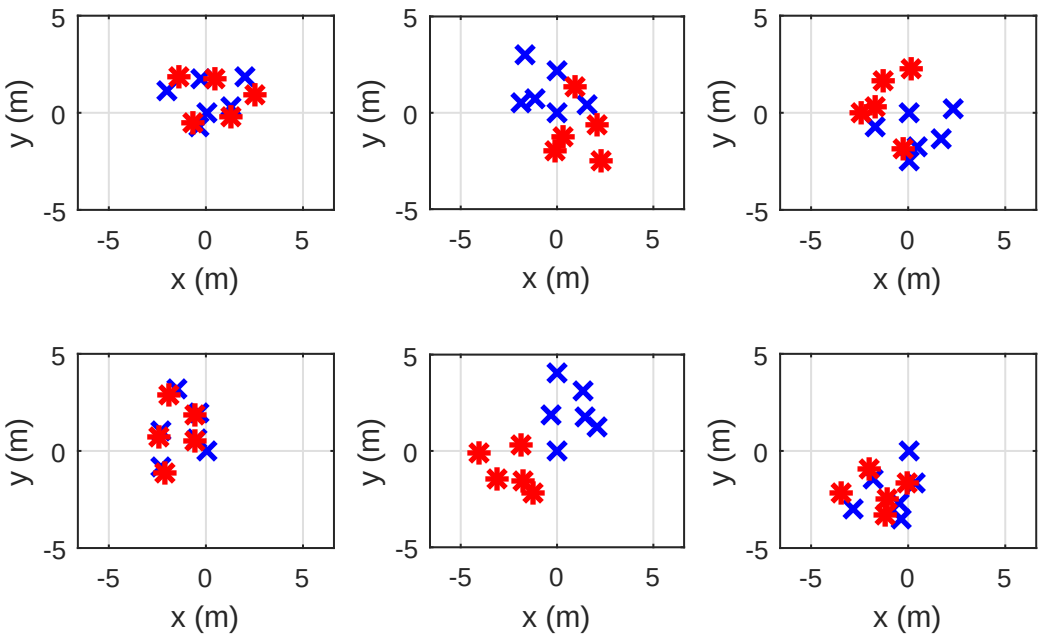


Figure 3-20: A picture of each robot and it estimated states using a random unknown initial position. The mobile robot in question is always at the origin and the other blue crosses are the real positions. The estimated positions of the other mobile robots are given by the red dots. The main problem here is that half of the mobile robots have estimated states that are mirrored.

As can be seen three of the six mobile robot have converged quite well to the real states, but the other three did not. When looked more closely one might see that the states are mirrored

with a symmetry line crossing the mobile robot itself. It looks like the estimated states are trapped in a mirrored local minimum. A disadvantage of the UKF is proven here, because the non-linear cost function $C = \|y(k+1) - \bar{y}(k+1)\|$ from Section 3-6-1 seems to have local minima. From multiple simulations it seems that this "mirroring" solution is a typical result of the local minima. Later in this thesis it seems that it is possible to get to the global minimum by walking in a more directions than in this simulation. In the next chapter the swarm controller is designed, which should create such movements that prevents local minima.

3-8 Conclusions

In this chapter a localization algorithm was designed to perform a position estimation of the other mobile robots by using the motion models and measurement model given in the first chapter. The specific task is to create a localization algorithm that estimates the relative position of each mobile robot in the neighbourhood without static beacons. Two solutions were given, first a modified weighted Multi-dimensional scaling localization (weighted-MDS) approach. The second attempt uses the relative pose estimation combined with the Multi-robot trilateration in a Unscented Kalman filter setting (MRRPL). Based on simulations it was concluded that the weighted-MDS did not converge to the correct orientation θ of the other mobile robots. Therefore the MRRPL is chosen to use as the localization algorithm in this thesis. From simulation results where the RSS noise was varied it was seen that tuning the noise variance matrices Q and R was difficult. Also it seems that σ_I is the major factor that needs to be improved to have better position estimations. It is still possible to do a well position estimation with the MRRPL even when the distance measurement noise σ_I is quite large.

Although it has been proven that the problem given in Section 3-6-1 can be solved with a Unscented Kalman Filter (UKF), it might be better to use other state estimators or adaptations of the UKF. A hopeful adaptation seems to be an iterated UKF, because of the large measurement noises and non-linear relations [32]. Another improvement can be gained by updating the noise variance matrices Q and R every time step, in that case the localization can become more robust to environment changes and/or improves the position estimation. An Adaptive Kalman Filter [34] could perform such Q and R optimization, but an adaptive version of the UKF was not found in literature. The last problem was that the estimated positions could get trapped in a mirrored local minima by the UKF, because the cost function $C = \|y(k+1) - \bar{y}(k+1)\|$ for the UKF of the MRRPL has local minima. In the next chapter the solution is given by a less parallel walking behaviour of the mobile robots. This is a result of the Virtual potential field swarm controller.

Chapter 4

Virtual potential field control for swarming behavior of mobile robots

In the following two chapters a controller is designed which let the mobile robot Z_0 react to other mobile robots and its environment. The goal is to stay away from dangerous places, avoiding collisions with each other or objects and at the same time going to a possible place of interest. Although a human or animal could do this with ease, the computational power and amount of sensors in a small mobile robot is not very high therefore a less intelligent but effective approach is needed. For example, think about the primitive mind of ants. With much lower intelligence, but in large numbers they can accomplice complex tasks like building and maintaining a complete ant nest. For doing so they only use simple rules for interacting with an environment filth with odours left by other ants. Every odour has a function like "stay away", "something interesting is around here" or as used in some path search algorithms "someone was here". Like this ant the mobile robot will also react to its environment, but in this case the odours are replaced by antenna beacons sending messages about what function the beacon has.

The first part of the chapter only considers other mobile robots to be present and a controller is designed to create a swarming behaviour of all the mobile robots together. For this controller the localization algorithm in the previous chapter is used to estimate the relative positions of other mobile robots in the neighbourhood. With a notion of the positions of the other mobile robots a controller can be designed to give the next direction a mobile robot should go. This direction is given as the control inputs u_θ and u_v for the heading angle and distance respectively. A Virtual Potential Field controller (VPF controller) is chosen to let the mobile robots react as the ants to avoid collisions with each other. The VPF controller has a very intuitive way on how to react to the environment and other mobile robots. Also a lot of research is already done on how to do so, but a combination of a localization algorithm with noisy measurements and

the VPF controller was not yet found in literature. Therefore this part of the thesis researches the effects of noisy position estimations on a VPF swarm controller is researched. Based on these conclusions a VPF controller is designed that will deal with these effects.

First an explanation on how the VPF controller works is handled. Next a VPF swarm controller for only reacting on other mobile robots is designed to avoid collisions and let them stay together. The VPF swarm controller results in the swarming behaviour of the mobile robots together. The results of the VPF swarm controller are given in the last section. How to use the VPF controller to control the complete swarm in combination with the environment is discussed in the next chapter.

4-1 The virtual potential field controller

In this section the Virtual Potential Field (VPF) controller is explained and which set of functions are chosen to create the VPFs. Before going into detail first an intuitive explanation of the VPF controller is given. One should imagine an environment like a field with obstacles to avoid, but at the same time a location of interest (goal). If the obstacles are seen as small hills and the goal is the lowest point of a valley one can create for example the following virtual environment.

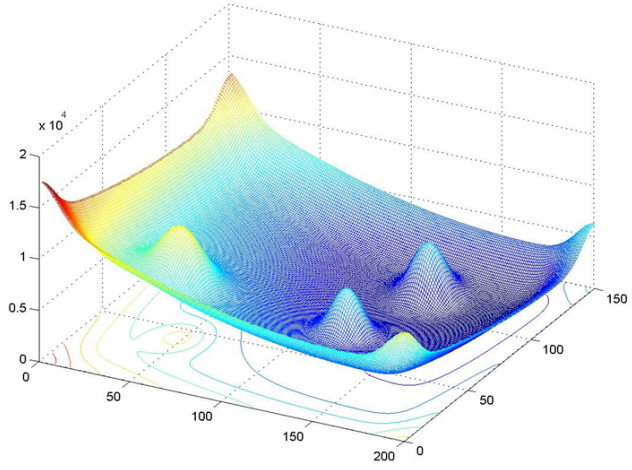


Figure 4-1: An example of a Virtual Potential Field, where the "hills" represent obstacles and the bottom of the "valley" is the point of interest. When a ball is released it will move around the obstacles towards the lowest point.

If a ball is placed on the left in this virtual environment it will move from a higher point (high potential energy) towards the valley (goal, with low potential) while moving around the obstacles (hills). If a mobile robot uses the same path of the ball it will also avoid the

obstacles and go towards the goal. For the direction of the mobile robot (ball) the greatest descent direction of this VPF is used. This type of controller was first created by Khatib [36], where he uses an electrical potential field instead of a height potential field as inspiration. It is also possible to add the second derivative (Hessian) [37], which has some advantages regarding undesired oscillations and faster convergence to the global minimum. The downside is that the used Hessian can become very computational intensive.

The greatest descent direction is calculated by using of all the directions $\vec{d_i}$ and distances $|d_i|$ towards objects, given in Figure 4-2.

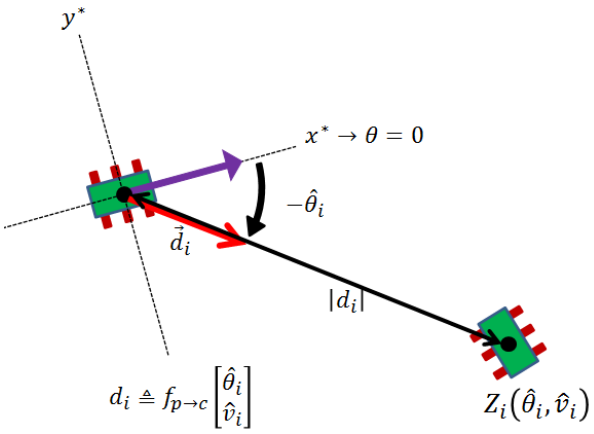


Figure 4-2: Direction and distance towards obstacles, goals and other mobile robots. The heading (purple arrow) is always pointed along the x-axis of the local coordinate system. The position of Z_i was estimated in polar coordinates $(\hat{\theta}, \hat{v})$ and transformed to a Cartesian vector d_i, where $\vec{d_i}$ is the normalized directional vector and $|d_i|$ the length.

In Figure 4-2 the other mobile robot Z_i is at the estimated relative position $(\hat{\theta}_i, \hat{v}_i)$ inside the local coordinate system of the mobile robot Z_0 in question. The use of these polar coordinates is not always desired. For example, when adding vectors as needed for the VPF controller it is more beneficial to use Cartesian vectors. Therefore the polar estimated position $(\hat{\theta}_i, \hat{v}_i)$ is converted to a Cartesian vector called d_i, where the vector $\vec{d_i}$ stands for the normalized vector used for directional purposes and $|d_i|$ is the length of the vector d_i used to indicate the distance between mobile robots. The vector $\vec{d_i}$ and value $|d_i|$ have a strong relation with the values θ_i and v_i respectively, but are more easy to use in vector calculus.

For every object in the environment a function can be given that calculates the heading u_θ and step size u_v the mobile robot should perform according to its position towards the object. The set of objects can contain obstacles (repulsive), goals (attractive) and other mobile robots (both attractive and repulsive). The following equations are the general formulations to create a VPF with "height" P and greatest descent direction ∇P for every type of object i.

$$P_i = g_r(|d_i|) + g_a(|d_i|), \tag{4-1}$$

where the repulsive part $g_r(|d_i|)$ is a decreasing function and the attractive part $g_a(|d_i|)$ increases for larger $|d_i|$. In case of an obstacle only the repulsive part $g_r(|d_i|)$ is used and for a goal only the attractive function $g_a(|d_i|)$ is used. For a mobile robot the attractive and repulsive part are both used, see the next section. The general greatest descent direction is given by

$$-\nabla P_i = -\vec{d_i}(\frac{\partial g_r(|d_i|)}{\partial |d_i|} + \frac{\partial g_a(|d_i|)}{\partial |d_i|}), \tag{4-2}$$

where $(\frac{\partial g_r(|d_i|)}{\partial |d_i|} + \frac{\partial g_a(|d_i|)}{\partial |d_i|})$ stands for the heading amplitude u_v and the direction $\vec{d_i}$ is the same as the heading control input u_θ.

The combination of all the virtual potential fields P_i of every object create the complete VPF of the environment P. Two main approaches exist to combine all the potential fields P_i. One used by Khatib [36] is to sum up all derivatives ∇P_i of P_i as

$$-\nabla P = \sum_i^N -\nabla P_i = \sum_i^N -\vec{d_i}(\frac{\partial g_r(|d_i|)}{\partial |d_i|} + \frac{\partial g_a(|d_i|)}{\partial |d_i|}), \tag{4-3}$$

with N as the total amount of objects. The other approach is first proposed by Rimon and Koditschek [38] and is based on multiplication. It multiplies all the attractor functions $g_a(|d_i|)$ to create $\gamma(|d_i|)$ and also multiplies all the repulsive $g_r(|d_i|)$ functions into $\beta(|d_i|)$. The multiplication functions $\gamma(|d_i|)$ and $\beta(|d_i|)$ are then used in the following function

$$P(|d_i|) = \frac{\gamma(|d_i|)}{\lambda\beta(|d_i|) + \gamma(|d_i|)} \tag{4-4}$$

which forms the complete VPF P, where λ is a shape parameter. With a numerical approach it is possible to derive the greatest descent direction $-\nabla P$ of P, which can be used as the input direction u_θ and u_v of the mobile robot. The restriction of this method is that the functions $g_a(|d_i|)$ and $g_r(|d_i|)$ have to be Morse functions [39].

An important aspect of creating a suitable VPF is that it contains no local minima. If a local minima exists it is possible that with the greatest descent direction the mobile robot navigates towards that local minimum, but not to the global minimum (goal). The method used by Rimon and Koditschek is said to have less chance of a local minima than that of the summation method by Khatib [38]. An important search was to find a method that has no local minima at all. Such a method was found in a paper by Kim [2]. Proven in [2] there exists a set of functions for VPFs P_i that can be summed up without the creation of local minima. The result is a motion like a leaf in a water stream avoiding stones and the functions used are called streaming functions [3].

$$P_i = \lambda \log(|d_i|) \tag{4-5}$$

$$-\nabla P_i = \lambda \frac{\vec{d_i}}{|d_i|}, \tag{4-6}$$

where $\lambda > 0$ is used for attractive goals and $\lambda < 0$ for repulsive obstacles. The function $\lambda \log(|d_i|)$ is a harmonic function with no local minima and also a summation of harmonic

functions is a harmonic function, therefore the summation of these streaming functions has no local minima [3].

Two disadvantages where found using these streaming functions. One is that $-\nabla P$ at the global minima of a goal is not zero, which will create no asymptotic stability of the system. Secondly the set of streaming functions does not contain functions representing U-shape obstacles. Like the leaf in a stream of water, on the edges of a stream there are non flowing pools created mostly by U-shaped like obstacles, like in Figure 4-3.

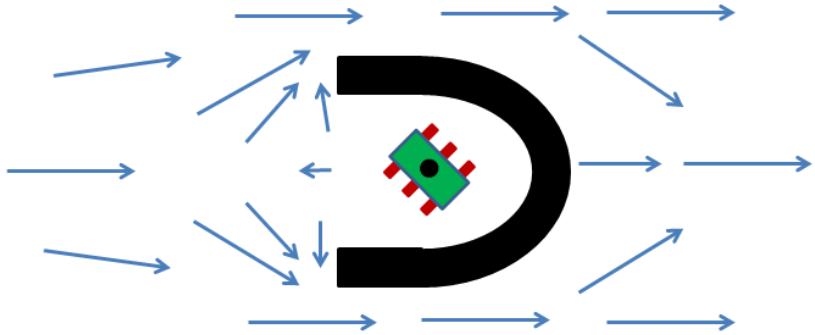


Figure 4-3: An example of a local minimum when using streaming function. In this situation the mobile robot got trapped in side the U-shaped obstacle.

Another observation made is that if one wants a global minimum at the goal, then the sum of all the amplitudes for obstacles λ_o needs to be smaller or equal to the sum of all the amplitudes of the goals λ_g, for the prove see Appendix C-3.

$$\sum_{o}^{O} \lambda_o \leq \sum_{g}^{G} \lambda_g, \tag{4-7}$$

where O is the total amount of obstacles and G is the total amount of goal points. This constraint can easily be explained by the fact that for a stream of water "incoming flow $\leq$ outgoing capacity" holds. The question remaining now is: Can the streaming functions still be used?. The answer is yes, because if these functions are only used for interaction with other mobile robots the disadvantages do not exist. That is ,because other mobile robots are not U-shaped obstacle and the goal part of the VPF of a mobile robot will never be reached, see next section. Creation of a VPF to deal with obstacles and goals is the subject of the next chapter.

4-2 Virtual potential field functions to create a swarm

Previously the VPF controller was explained. The VPF controller uses attractive and repulsive functions to navigate a mobile robot by using the greatest descent direction $-\nabla P$ of the VPF P.

The set of functions for attractive and repulsive forces were chosen to be streaming functions, which were given by

$$P_i = \lambda \log(|d_i|) \tag{4-8}$$

$$-\nabla P_i = \lambda \frac{\vec{d_i}}{|d_i|}, \tag{4-9}$$

The goal of this section is to use these equations to create a VPF that is used for interacting with other mobile robots. The result needs to be a controller that creates a swarming behaviour of the robot swarm. For a swarming behaviour the following three requirements are given.

- No collisions between mobile robots
- Stable cohesion of the swarm. Meaning no separation of the swarm at all times.
- A random moving behaviour for exploration between the boundaries given by the previous two points.

The third requirement is partially already achieved by the fact that the position estimation from the previous section is quite noisy. Therefore the VPFs are also quite noisy, which results in a control input given by the greatest descent direction that is also more random. Although this is not particularly a solution for exploration, it adds a little randomness to the movement of the swarm. Although the noisy position estimation is beneficial for exploration it is a disadvantage for the other two requirements, because by intuition if one wants to avoid collision or stay with the swarm, one wants to know where the other members of the swarm are. How to deal with this and more problems due to uncertain positions will be pointed out and tried to be solved during this section.

First a general explanation of how the swarming VPF should look like. The goal was no collisions between mobile robots and stable cohesion by not walking to far from each other. A picture of such a VPF is given in Figure 4-4

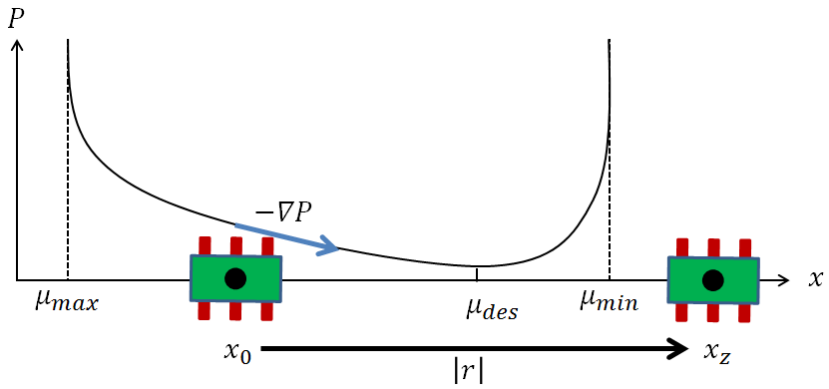


Figure 4-4: The VPF that represents another mobile robot. This VPF has the goal of staying close $|r| < \mu_{max}$ to the other mobile robot, while not colliding into it $|r| > \mu_{min}$. The minimum at μ_{des} is the desired distance. The control input direction is then given by the derivative $-\nabla P$ of the VPF P.

In Figure 4-4 three important parameters are given. Firstly, μ_{min} gives the minimal distance between two mobile robots to avoid collision. If a mobile robot has a significantly large inertia, than a larger minimal distance would be required [40]. In case of the Zebro robot this is not necessary, because the Zebro robot can almost stand still instantaneously. Second, the distance μ_{max} stands for the maximum distance two mobile robots should be apart from each other. The maximum distance could depend on multiple factors, but in this thesis it depends on the maximum connectivity range, which was 10 meters, see Section 2-3. With a large robot swarm it is not necessary to stay in contact with all the mobile robots. In that case a mobile robot can chose to only have a maximum distance if a minimal amount of mobile robots in range is reached. The last parameter is μ_{des} which is the desired distance towards another mobile robot. The chosen functions for the VPF to represent another mobile robot is given by the following equations.

$$P_i = -\log(|d_i| - \mu_{min}) - \frac{\mu_{max} - \mu_{des}}{\mu_{des} - \mu_{min}} \log(\mu_{max} - |d_i|) \tag{4-10}$$

$$-\nabla P_i = \vec{d_i} \left(-\frac{1}{|d_i| - \mu_{min}} + \frac{\mu_{max} - \mu_{des}}{\mu_{des} - \mu_{min}} \frac{1}{\mu_{max} - |d_i|} \right), \tag{4-11}$$

This equation is constructed as follows. First the potential field function is chosen a form which contains a part that attracts and a part that repulses with a balance parameter C between both.

$$P_i = -\log(|d_i| - \mu_{min}) - C \log(\mu_{max} - |d_i|) \tag{4-12}$$

If the distance $|d_i|$ is μ_{des} then the amplitude of the gradient of $-\nabla P_i$ should be zero. This leads to the following equation.

$$\| - \nabla P_i \| = -\frac{1}{\mu_{des} - \mu_{min}} + C \frac{1}{\mu_{max} - \mu_{des}} = 0 \tag{4-13}$$

The solution for C is given as

$$C = \frac{\mu_{max} - \mu_{des}}{\mu_{des} - \mu_{min}} \tag{4-14}$$

Before Equation (4-10) can be applied non-imaginary values should be avoided when a mobile robot is located outside the boundaries of μ_{min} and μ_{max}. This is necessary, because with the noisy position estimations there is a possibility of suddenly having a distance outside the boundaries of μ_{min} and μ_{max}.

$$\begin{array}{llll} \text{if} & |d_i| < \mu_{min}(1+\epsilon) & \text{then} & |d_i| = \mu_{min}(1+\epsilon) \\ \text{elseif} & |d_i| > \mu_{max}(1-\epsilon) & \text{then} & |d_i| = \mu_{min}(1-\epsilon)\text{'} \end{array} \tag{4-15}$$

where ϵ is a small value to avoid having the result where $-\infty = \log(0)$. The resulted combined vector for the direction and speed in case there are multiple mobile robots Z_i is now given by

$$-\nabla P = \sum_{i}^{Z} -\nabla P_i, \tag{4-16}$$

where Z is the amount of mobile robots in the neighbourhood. A mobile robot always has a maximum speed v_{max}, therefore a transformation of $-\nabla P$ is used to set this speed limit. This transformation needs to maintain the characteristics of $-\nabla P$ of Equation (4-11), such as maintaining the no local minima characteristic. A function that does so is a Gaussian function given by

$$-\nabla P^* = \frac{-\nabla P}{|-\nabla P|}\left(v_{max} - v_{max}\exp^{-(2/\alpha\sqrt{-\log(0.5)}|-\nabla P|)^2}\right) \tag{4-17}$$

Here α is used as a smoothing factor, explained later. In Figure 4-5 a summary of the different functions for P, $-\nabla P$, the smoothing transformation and resulted $-\nabla P^*$ is given when the distance to one other mobile robot is given by $|d|$.

The influence of the smoothing factor α is given in Figure 4-6.

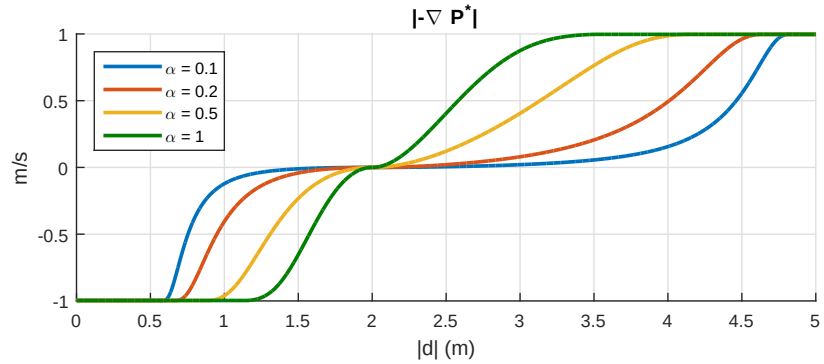


Figure 4-6: The influence of the smoothing parameter α on the smoothed gradient $-\nabla P^*$. With a lower value for α a mobile robot will only have a large control input reaction when the limits $\mu_{max} = 5$ and $\mu_{min} = 1$ are reached. With larger α large control input reactions are also present at short difference from the desired distance $\mu_{des} = 2$.

As shown in Figure 4-6 the smoothing parameter α is of much influence on the behaviour of the mobile robot. If α is large then a mobile robot will react with large speed amplitudes even when the optimal desired position is almost reached. The opposite occurs for low values of α. In that case the mobile robot has more freedom to move around without acting to much on other individuals of the swarm. This might be a suitable solution if an exploration mission is required. A downside of a low or high smoothing parameter α is larger speed differences due to steeper walls of the VPF. This will be a probem when having uncertain estimated positions of the other mobile robots. In that case the control input speed is also noisy and non-smooth. For example, if in Figure 4-6 an $\alpha = 1$ is chosen and the true distance $|d|$ is 2 meters. A given distance noise of 0.5 meter (or 25 percent) creates undesired back-and-forth oscillations of the robot. As told in Chapter 2-3 within a more complex environment an estimated distance noise of 25 percent is a realistic. If such oscillations needs to be avoided $\alpha = 0.5$ in Figure 4-6 would be around optimal, because it has less steep slopes. Summarized, a low smoothing factor α is suitable for more freedom to explore and a large α creates a strong force to keep the mobile

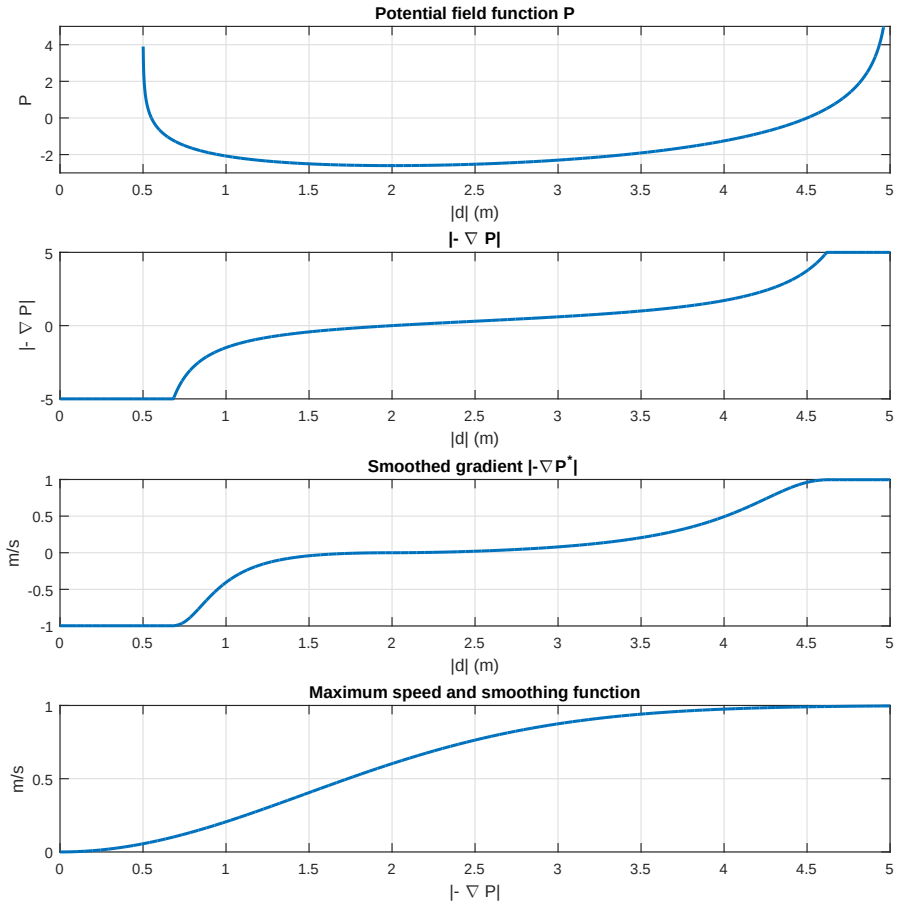


Figure 4-5: Front top to bottom: The potential field function for P from Equation $(4\text{-}10)$. The second graph presents the gradient $|-\nabla P|$, see Equation $(4\text{-}11)$. The third graph is the smoothed gradient $-\nabla P^*$ transformed by the smoothing function in the last graph with $\alpha = 0.2$ and $v_{max} = 1$

robots at the desired distance from each other. Both large and small values of α result in jerky or oscillating movements, therefore a more averaged value $\alpha = 0.5$ is desired.

An important topic is if the expected heading direction with noisy localization is the same as with a perfect localization. The distribution of the angular error is like a normal distribution and $E[\hat{\theta} - \theta] = 0$ holds, which will lead to the following result for every mobile robot i.

$$\theta_i = \vec{d_i} = \angle \nabla P_i \tag{4-18}$$

$$\hat{\theta}_i = \hat{\vec{d}}_i = \angle \nabla \hat{P}_i \tag{4-19}$$

$$\Rightarrow E[\angle \nabla \hat{P}_i - \angle \nabla P_i] = 0 \tag{4-20}$$

where $\angle \nabla P_i$ is the heading angle and the the same as the estimated angle θ_i of mobile robot Z_i. So the expected error on the heading $\angle \nabla P_i$ of the mobile robot is zero. Also when Z amount of multiple robots are used, see below.

$$E[\angle \nabla \hat{P} - \angle \nabla P] = E\left[\sum_i^Z \angle \nabla \hat{P}_i - \sum_i^Z \angle \nabla P_i\right] = \sum_i^Z E[\angle \nabla \hat{P}_i - \angle \nabla P_i] = 0 \tag{4-21}$$

So the expected error of the estimated heading direction $\hat{u_\theta}$ is u_θ. In that case the robot swarm has an "expected swarm behaviour".

4-3 Simulation results

In this section the previous swarming VPF controller with an smoothing factor of $\alpha = 0.5$ is combined with the tuned MRRPL algorithm for the localization of the previous chapter. The localization will create the estimated positions as the input of VPF swarm controller. Simulations are done to find out if the VPF gives the desired result and also how the movements of the VPF swarm controller have a correlation with the localization algorithm. In all the cases the mobile robots do not know the initial relative positions of the other mobile robots and therefore use a random initial estimated position close to zero.

The first analysis looks if the mobile robots are indeed moving like the VPF swarm controller was designed for. A measure of cohesion is taken by taking the maximum over all distances towards the center of the swarm δ_{max} over all time for all mobile robots.

$$\delta_{max} = \max\left(X_1(t) - \frac{\sum_i^Z X_1(t)}{Z}, \ldots, X_i(t) - \frac{\sum_i^Z X_i(t)}{Z}\right) \ \forall\ i = 1, 2, \ldots Z \text{ and } t = 1, 2, \ldots T, \tag{4-22}$$

where Z is the amount of mobile robots and X_i the position of Z_i. When using this maximum as a measure of the cohesion of the swarm a larger simulation time T is necessary to prove if one of the robots really walks away. In this case a simulation time of $T = 200$ is chosen.

During this analysis it was already quickly found that the behaviour of the swarm was largely dependent on the tuning parameters K_R and K_{ratio} of the UKF in the MRRPL localization

algorithm. A more accurate tuning of the parameters K_R and K_{ratio} is needed to deal with this relation. The same approach was taken as the tuning of the MRRPL in the previous section, but in this case also in combination with the swarm controller and a measure of cohesion δ_{max}. For the tuning in this chapter with the VPF swarm controller the distance measurement noise σ_I is kept constant at a value of 3dB from Chapter 2. In Appendix D the results are given for different K_R and K_{ratio}. The new tuning parameters of the noise variances matrices Q and R with the swarm controller are

$$K_R = 1 \quad K_{\text{ratio}} = 1,$$

where K_{ratio} should not be larger than 1 for all times. Compared to the results of the previous chapter which were

$$K_R = 1000 \quad K_{\text{ratio}} = 10$$

It seem not much of a difference, but the difference in values for K_{ratio} makes a lot of difference in the swarm its behaviour.

The next part of this section will give a feeling of the behaviour of the mobile robots individually. The optimal localization parameters $K_R = 1$ and $K_{ratio} = 1$ will be used to estimate the positions. The simulation run for $T = 200$ time samples, with 5 swarm members starting at $X_0 = \begin{bmatrix} x_1 & x_2 & \cdots & x_5 \\ y_1 & y_2 & \cdots & y_5 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 1 & -1 & -1 \\ 0 & 1 & -1 & 1 & -1 \end{bmatrix}$. In Figure 4-7 the absolute movements of the swarm members of a simulation run are given. Each member has its own colour.

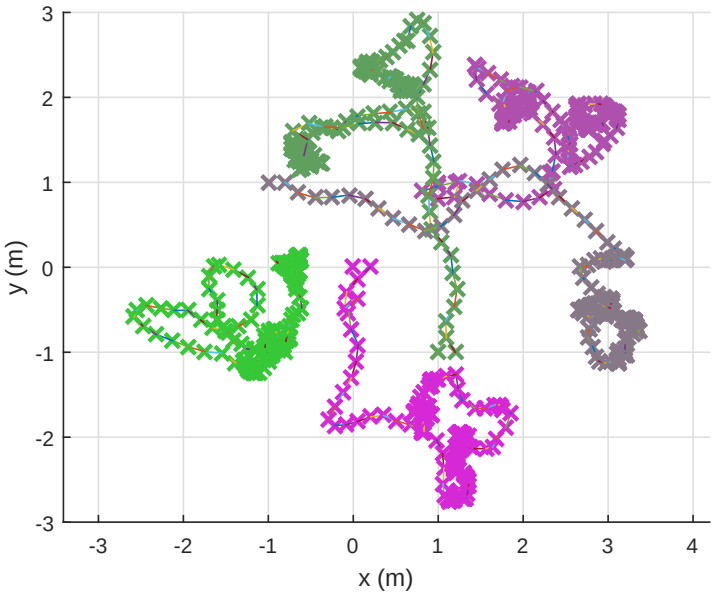


Figure 4-7: An example of the route walked by the mobile robots using the VPF swarm controller.

From Figure 4-7 it can be seen that all the members stay in a certain boundary, which shows the wanted cohesive behaviour of the swarm. Also it shows that after a short while the swarm members have created a well estimation of the position of the other members. After the positions were estimated the members walk according to the VPF swarm controller to a position where they are not to far away, but also not too close to each other. This equilibrium point for each member create an according equilibrium graph shaped by the swarm members. In this case of 5 swarm member a pentagram, more clearly in Figure 4-8. The behaviour after this equilibrium is reached might be difficult to see. Some swarm members will sometime walk away from the equilibrium point, because when standing still the localization gets worse. This is due to the loss of the localization its observability requirement from Section 3-5 to keep walking. By walking the position estimation gets better again and the swarm member goes back to its new equilibrium point in the swarm. This "loop" of walk around → better position estimation → reach an equilibrium → stop walking → position estimation worsens → walk around, is self containing. One should watch out that there is always a slight chance that this loop is broken, the chance doing so increases especially when changing the parameters K_R and K_{ratio}. The next two figures give an indication on how well the position estimation is performed. In Figure 4-8 it is show that the mirroring effect did not occur. In the previous chapter the more random walk was said to be the solution of this mirroring problem and as it was predicted the VPF swarm controller solved this problem. This was especially shown in the animated version of Figure 4-8, which is the final position estimation of the simulation in Figure 4-7.

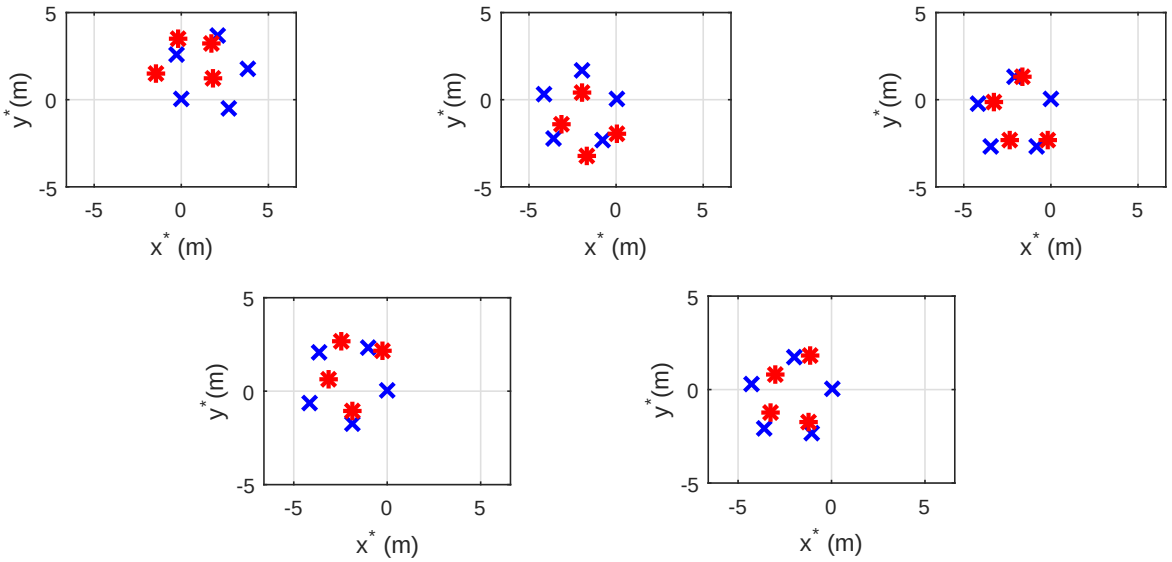


Figure 4-8: Individual relative position estimations, where blue is the true value and the red stars the estimated values. One can see that compared to the position estimations of the previous chapter the mirroring problem is solved by implementing the VPF swarm controller.

The last figure shows the error distribution of all the position estimations done in the single simulation of Figure 4-7.

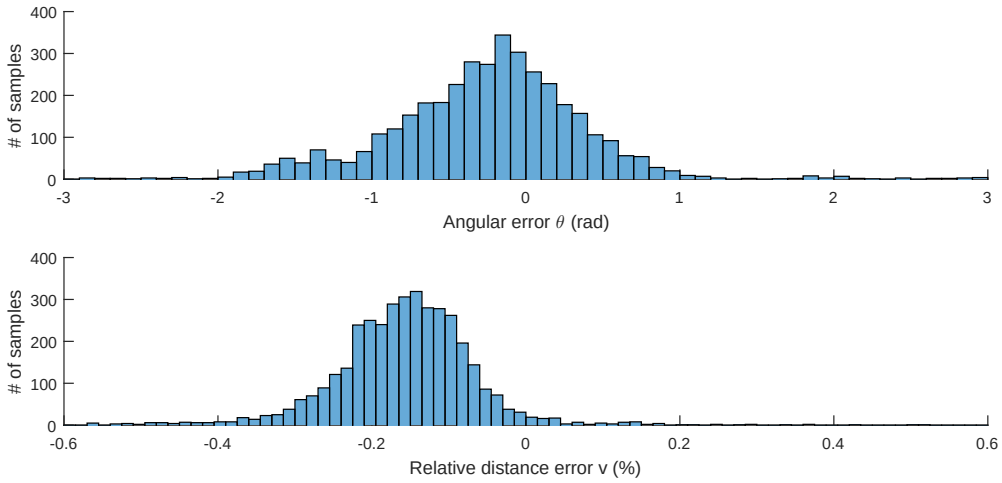


Figure 4-9: The error distibution of the estimated relative angle and distance of the relative localization.

Compared with the results from the localization simulations in the previous chapter the localization performance did even improve.

	MRRPL with localization test setup	MRRPL with VPF swarm controller
RMS error θ (rad)	1.2925	0.8976
Relative RMS error v (%)	0.5219	0.1309

Table 4-1: A comparison of the localization error between the localization algorithm (MRRPL) with and without the VPF swarm controller. For this 20 runs were taken for both simulations with five swarm members. A simulation length $T = 50$ for the localization test setup and $T = 200$ for the setup with the VPF swarm controller.

In the end the VPF swarm controller in combination with the MRRPL with relative large distance measurement creates a swarm of mobile robots that shows cohesive behaviour. The results also show a balanced behaviour between the position estimation and movement from the VPF swarm controller. The resulted behaviour "loop" works as follows: walk around $\rightarrow$ better position estimation $\rightarrow$ reach an equilibrium $\rightarrow$ stop walking $\rightarrow$ position estimation worsens $\rightarrow$ walk around. It was necessary to minimize the chance of instability and was done by a well done tuning of the variance noise matrices R and Q. From simulations it was seen that changes of the noise variance gain parameters K_R and K_{ratio} the chance of instability rapidly increases, especially sensitive to K_{ratio}. In the end the optimal gain on the matrices R and Q

seem to lie around one, which could indicate that the matrices R and Q are formulated well. It is still possible that other optimal values for R and Q needs to be found when the distance noise variance σ_I does change by changes of the environment. Therefore as discussed in the end of the previous chapter another state estimator or modification of the UKF needs to be used.

4-4 Conclusions

In this chapter a Virtual Potential Field (VPF) swarm controller was designed. This controller is based on VPFs which create a virtual version of the real surrounding. All objects to avoid have a repulsive field around them and a point of interest has an attractive field. The greatest descent direction of such a VPF is the heading direction a mobile robot. For the VPF swarm controller another mobile robot is seen as a combination of a repulsive obstacle and an attractive goal, because the mobile robots should stay close and not collide into the other mobile robots. The VPFs used to define the function for a VPF of another mobile robot are based on a summation of streaming functions. The summation of streaming functions do not contain no local minima and the summation makes calculating the greatest descent direction computational easy. The disadvantages of the streaming function were given in Section 4-1 and were discarded when only used to represent another mobile robot.

The results of Virtual Potential Field (VPF) swarm controller in combination with the MRRPL position estimation were showing general cohesion of the swarm. From simulation a balanced behaviour between the position estimation and movement from the VPF swarm controller was shown. This expected balance or "loop" works as follows: walk around $\rightarrow$ better position estimation $\rightarrow$ reach an equilibrium $\rightarrow$ stop walking $\rightarrow$ position estimation worsens $\rightarrow$ walk around. It was necessary to minimize the chance of instability and was done by a well done tuning of the variance noise matrices R and Q. Summarized, this chapter only creates a swarm inside a completely empty environment. Therefore in the next chapter an overview of additions for the VPF is given to let the swarm have interactions with a more real environment.

Chapter 5

Obstacle avoidance and goal searching using radio beacons

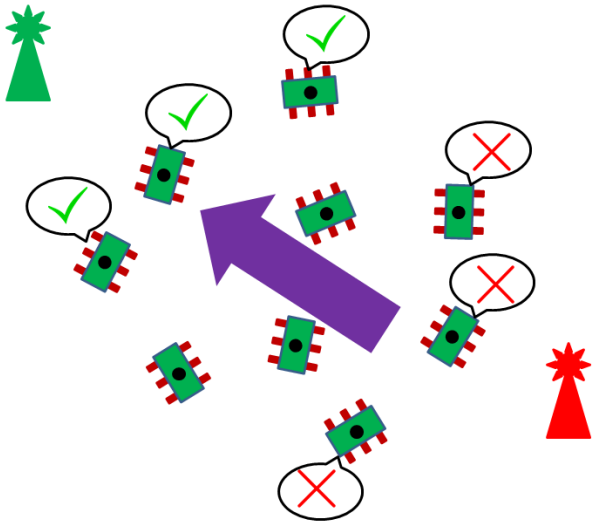


Figure 5-1: A visualization of a swarm heading towards a point of interest which position is unknown. It does that by letting each member of the swarm walk towards the best positioned members. In that case the complete swarm will move towards the attractive goal and walk away from obstacles.

Knowing the location of the other robots and creating a swarm that sticks together is not the only goal of a mobile robot swarm. The next and final step of this thesis is to give the

swarm a function to full fill. There are multiple types of missions a mobile robot swarm can be used for. Most of these can be summarized into two types: Exploration and goal search. An exploring swarm explores or examines the surrounding while avoiding obstacles. In case of a goal search there is a attractive goal which needs to be reached while avoiding obstacles to get there. Both goal and obstacles are presented in this thesis by antenna beacons, where the distance is given by the RSS measurements from Section 2-3. In this chapter both swarm functions will be tested with different simulation scenarios. In all these scenarios the locations of the obstacles and goal beacons will be unknown, because a real swarm individual does also not know the exact location of obstacles or goal. It therefore uses information and movement of the complete swarm to derive its individual heading direction, see Figure 5-1.

There are multiple ways to navigate the swarm without knowing the exact location of the beacons. Three navigation algorithms that does do such are presented and tested in this chapter and also to see which can deal with noisy distance measurements the best.

5-1 Virtual potential fields for obstacle and goal beacons

Like with the VPF swarm controller also a VPF of the surrounding P_{surr} can be created. The main difference is that only the distance $|d_i|$ towards the obstacles (and goal) are known, but not the orientation $\vec{d_i}$. So only the value or height of the surrounding VPF P_{surr} of every mobile robot is known, see Figure 5-2.

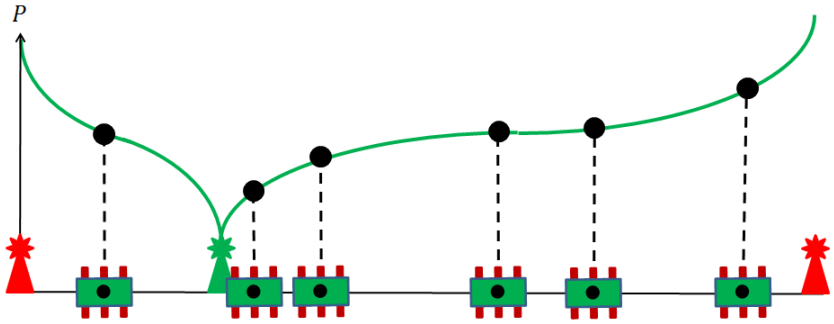


Figure 5-2: An one dimensional example where every mobile robot only knows the value/height of the surrounding VPF.

This "height" represents how well the mobile robot is positioned in the surrounding. A low value means the mobile robot is at a suitable position away from obstacles or a position closer towards a goal point. The opposite, a higher value tells that the mobile robot is at a less suitable position or away from a goal beacon. In literature this value of the VPF is mainly visualized as the concentration of toxic odor and the mobile robots are like insects avoiding these toxic odors. In this chapter also the notion of toxic concentration will be used. The toxic concentration level is given by the value OG (Obstacle-Goal value). If the OG values of all mobile robots in the swarm are known a local impression of the surrounding can be given,

see Figure 5-1, where the set of all the positions of the mobile robots Z_i and their OG values will be create a local VPF P_{local}. The complete robot swarm will react according to this local VPF P_{local}, which results in a joint movement of the complete swarm. The result is a swarm that acts like a single "organism". This single organism should navigate around the obstacles and if needed goes towards a goal beacon.

For the surrounding VPF P_{surr} the same requirements as for the VPF swarm controller controller VPF P_{swarm} are considered, which were given in Section 4-2. The most important requirement in case of the surrounding VPF P_{surr} is to avoid the existence of local minima. If there is a local minima the complete swarm could get trapped in it and does not go toward a global minimum. To avoid this the streaming VPF functions from Section 4-2 are used again. The VPF created by Equation (5-1) can be used for point obstacles and goals.

$$P_{\text{surr},i} = \lambda \log(|d_i|), \tag{5-1}$$

where for an attractive goal $\lambda > 0$ is used and for obstacles $\lambda < 0$. Also the same constraint to let the goal be the global minima holds.

$$\sum_{o}^{O} \lambda_o < \sum_{g}^{G} \lambda_g, \tag{5-2}$$

In this case the function $\lambda_g = 1 + 2\sum_i^O \lambda_o$ is chosen to full fill the constraint. The sum of all the values $P_{\text{surr},i}$ from each obstacle or goal is the total value of the toxic concentration OG at the position of the mobile robot, see below.

$$OG = P_{surr} = \sum^{O} \lambda_o \log(|d_{i,o}|) + \sum^{G} \lambda_g \log(|d_{i,g}|) \tag{5-3}$$

The next part of this section tells about possible modifications of the obstacle VPF functions $P_O = \lambda_o \log(|d_o|)$ to create forbidden regions and line obstacles. First the VPF function P_O with a forbidden circular region is presented. Like with the VPF swarm controller obstacles the forbidden circular region uses the following equation.

$$P_O = -\lambda_o \log(|d_O| - \mu_{min}), \tag{5-4}$$

where $|d_O|$ is the distance towards the obstacle and μ_{min} the radius of the forbidden region. It is necessary to assure a high positive toxic values OG in the forbidden region and therefore the following constraint is given.

$$\text{if} \quad |d_O| - \mu_{min} < \epsilon \quad \text{then} \quad P_O = -\lambda \log(\epsilon), \tag{5-5}$$

This constraint also assures real values for $|d_O| - \mu_{min} < \epsilon$ and no infinite values for P_O when $|d_O| - \mu_{min} = 0$ as explained in the figure below.

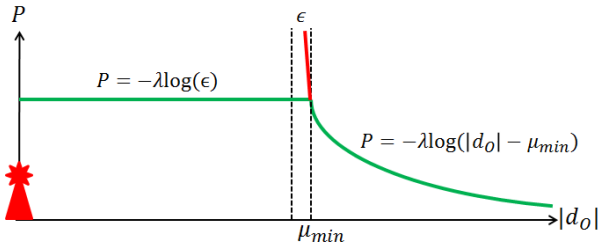


Figure 5-3: The commands for the direction of a Zebro robot, where no simultaneous forward/backward and turning movements is considered.

It is also possible to create a wall-like obstacle. This can be done if the distance towards two points A and B representing the outer points of the wall are known.

$$P_{O,line} = -\lambda \log(|d_{O,A}| + |d_{O,B}| - |d_{line}|), \tag{5-6}$$

where $|d_{O,A}|$ and $|d_{O,B}|$ are the distances from the mobile robot towards point A and point B respectively and $|d_{line}|$ is the distance between point A and B. For the line obstacles also a forbidden region can be created. This is given by

$$P_{O,line} = -\lambda_O \log(|d_{O,A}| + |d_{O,B}| + \mu_{min} - |d_{line}|), \tag{5-7}$$

where μ_{min} is a distance added to $|d_{O,A}| + |d_{O,B}|$ that lets the wall become wider. Like with the circular obstacle the following constraint assures existing solutions

$$\text{if} \quad |d_{O,A}| + |d_{O,B}| + \mu_{min} - d_{line} < \epsilon \quad \text{then} \quad P_{line} = -\lambda \log(\epsilon), \tag{5-8}$$

where ϵ is a small positive number. One need to remember that no greatest descent directions $-\nabla P_O$ can be given, because $\vec{d_O}$ is unknown. Therefore solutions are given in this chapter to deal with obstacles position whose orientations are unknown.

Summarized the swarm has no knowledge about the location of obstacles or goal beacons in the surrounding. Only the distance $|d_i|$ towards an obstacle or goal is measured. The obstacles can be a point, circle or line obstacle. A VPF P_{surr} represents the surrounding filled with obstacles and a possible goal point. Each mobile robot gets a value of the surroundings VPF P_{surr} called OG representing a "toxic concentration". The combination of all OG values of each mobile robot creates a local impression of the surrounding VPF P_{surr}. The heading of a mobile robot depends on this impression and the result is a swarm which acts as a single "organism". In the next section three methods are presented which let the swarm act based on all OG concentration values of each member of the swarm.

5-2 Obstacle avoidance and goal search algorithms

In the previous section a VPF P_{surr} that describes the surrounding is given. This VPF P_{surr} is unknown, but each mobile robot measures its toxic concentration value OG based on it value

inside the surrounding VPF P_{surr}. Knowing all the OG values of the other mobile robots and their relative positions gives an impression of the local surrounding. A individual heading direction F_{surr} is based on the OG concentration values and will be added to the VPF swarm controller direction $-\nabla P_{\text{swarm}}$ by the following function.

$$F_{tot} = v_{max} \frac{-\nabla P_{\text{swarm}} + \eta F_{surr}}{\| -\nabla P_{\text{swarm}} \| + \| \eta F_{surr} \|}, \tag{5-9}$$

where $-\nabla P_{\text{swarm}}$ is the smoothed VPF swarm control heading vector and F_{surr} is the heading vector from interacting with the surrounding. This equation has the advantage that it creates a balance between the control input influenced by the surrounding and cohesion of the swarm. When a mobile robot is in a good position relative to the swarm ($-\nabla P_{\text{swarm}} = 0$) then the mobile robot will move with maximum speed v_{max} in the direction of F_{surr}. If the swarm has a good position towards the environment ($F_{surr} = 0$) then it will focus on maintaining the cohesion of the swarm. The balance between $-\nabla P_{\text{swarm}}$ and F_{surr} is given by a priority parameter $\eta > 0$ and results in a speed which is always v_{max}. The following algorithms use the concentration values OG to create the additional heading direction F_{surr}. In the next section these algorithms are simulated with some surrounding test cases.

The first algorithm is based on Particle Swarm Optimization (PSO) [41] and will be called m-PSO. The main idea is to walk towards the "best" mobile robot at position $\vec{d}_{high}$ and from the "worst" mobile robot at position $\vec{d}_{low}$, presented by the following equation.

$$F_{surr} = \vec{d}_{high} - \vec{d}_{low}, \tag{5-10}$$

If the mobile robot itself has the highest or lowest value it only uses the lowest or highest value respectively.

The second algorithm let every mobile robot follow the gradient of the toxic concentration values OG_i of all the swarm members Z_i [42], which in literature is called chemotaxis. A least-squares approximation is used to find this first order (plane) gradient by using all the OG_i values at the estimated positions X_i of the corresponding mobile robots Z_i. It is also possible to approximate the gradient of higher orders. This might give more detailed information of the local surrounding, but is not confirmed by literature and not tested.

The third method is based on the combination of the previous two methods. It uses all the OG values of all the mobile robots and compares it to its own OG value to decide to walk from or towards the other mobile robot. One can say that another mobile robot with a higher OG value is an repulsive obstacle and a mobile robot with a lower OG is an attractive goal. Then the function will be

$$P_{Z,i} = (OG_i - OG_0) \log(\|d_i\|) \tag{5-11}$$

$$\sum_{i}^{Z-1} -\nabla P_{Z,i} = F_{surr} = \sum_{i}^{Z-1} (OG_i - OG_0) \vec{d}_i \tag{5-12}$$

where OG_i is the OG concentration value of the other mobile robot Z_i and OG_0 of itself.

5-3 Obstacle avoidance and goal search simulation results

In this section simulation tests are done to compare the three different navigation algorithms which interact with the surroundings. For each of the algorithms three types are considered: Goal search, Goal search with obstacles and staying inside a virtual cage. Multiple other cases are possible, but these three cases represent a basis of all the possibilities.

For the simulation tests the following settings from the previous chapters are used:

- Localization algorithm: MRRPL with optimal parameters: $N = 10$, $Q_x = .01$, $K_R = 1$ and $K_{\text{ratio}} = 1$
- VPF swarm controller: $\mu_{min} = 1\text{m}$, $\mu_{des} = 2\text{m}$, $\mu_{max} = 4\text{m}$ and $\alpha = 0.5$.
- Maximum distance per time sample: $v_{max} = 0.2\text{m}$
- Amount of mobile robots: $Z = 5$
- Initial positions: Unknown, $X_0 = \begin{bmatrix} x_1 & x_2 & \cdots & x_5 \\ y_1 & y_2 & \cdots & y_5 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 1 & -1 & -1 \\ 0 & 1 & -1 & 1 & -1 \end{bmatrix}$.
- Measurement noise standard variances: $\sigma_I = 3\text{dB}$, $\sigma_\theta = 5^\circ$ and $\sigma_v = 10\%$
- Used heading priority ratios: m-PSO $\rightarrow \eta = 3$, Chemotaxis $\rightarrow \eta = 2$ and Compare $\rightarrow \eta = 3$.

The parameters η for the different navigation algorithms are found empirical by doing multiple simulations. The first simulation case consists of only of a goal beacon. The next three figures show the result of each navigation algorithm, where the swarm tries to find the goal at position $(0, 10)$. If all measurements were (near) perfect a precise optimal path would be taken every time. In this thesis it was the challenge to deal with highly uncertain measurements. Therefore a density of the walked path of the swarm members is analysed to see if the localization algorithm creates positions estimation well enough to guide a mobile robot swarm. This density grid is created by counting how many times a mobile robot was in a specific area. These areas are given by squares of 0.25 meter. For the goal search test 50 runs of $T = 300$ time samples long were taken. The first result is given below, where the m-PSO navigation algorithm is used.

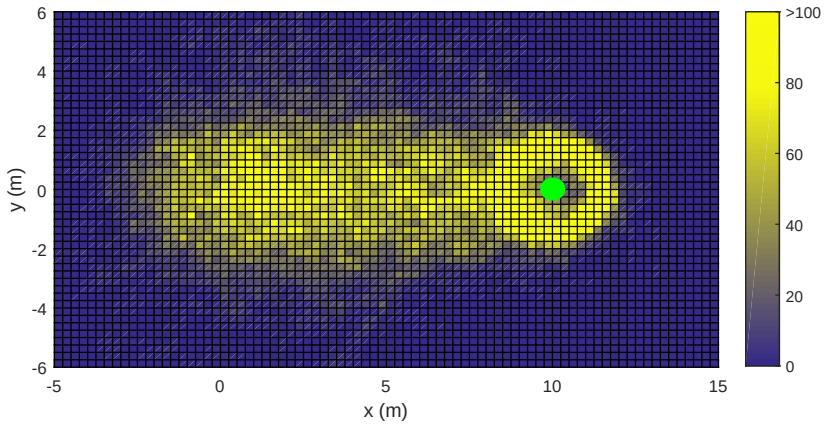


Figure 5-4: m-PSO: A two dimensional density plot, where the density tells how many times a mobile robot was inside that area of 0.25x0.25 meter. Areas with a counting higher than 100 are all yellow. The case: Goal search with the m-PSO algorithm, where the swarm starts around $(0, 0)$ and the goal is positioned at $(0, 10)$.

Figure 5-4 shows a quite satisfied behaviour of the swarm members going towards the goal with the use of the m-PSO algorithm. The result proves that even with the most simple thinking a swarm can be navigated. The large donut shaped density around the goal is caused by the convergence of the swarm members around the goal and while maintaining the desired distance $\mu_{des} = 2$m from each other. This behaviour will also been shown by other simulations. The next simulation figure shows the result of the Chemotaxis algorithm.

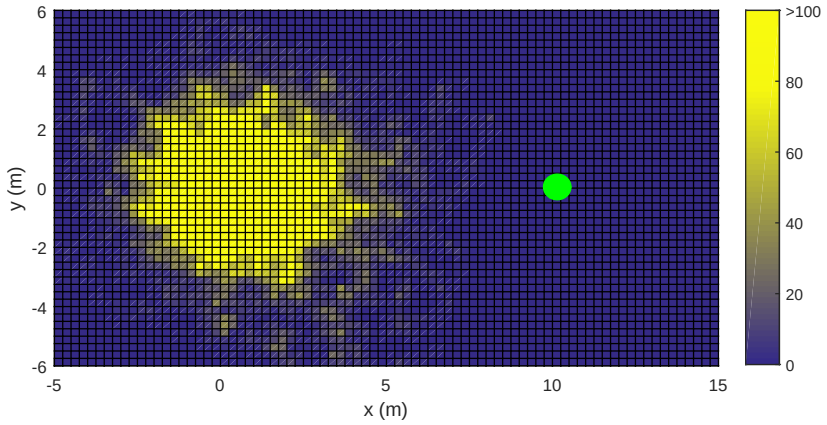


Figure 5-5: Chemotaxis: A two dimensional density plot, where the density tells how many times a mobile robot was inside that area of 0.25x0.25 meter. Areas with a counting higher than 100 are all yellow. The case: Goal search with the Chemotaxis algorithm, where the swarm starts around $(0, 0)$ and the goal is positioned at $(0, 10)$.

The result is rather dissatisfying. All the members of the swarm stay on a desired position from each other, but there is no clear sign of a movement towards the goal beacon. The problem might lie in the fact that the gradient of the plane constructed from OG values is to small. If the gradient and in that case also the heading direction F_{surr} is very noisy then the expected heading $E[F_{surr}]$ is almost the zero vector. When this happens all the members of the swarm stay around the same position. More priority to act on the surrounding by increasing η gave only worse results, because the priority to stay together became to small. This resulted in a less cohesive behaviour of the swarm. The last test to find a goal beacon is taken with the Compare algorithm.

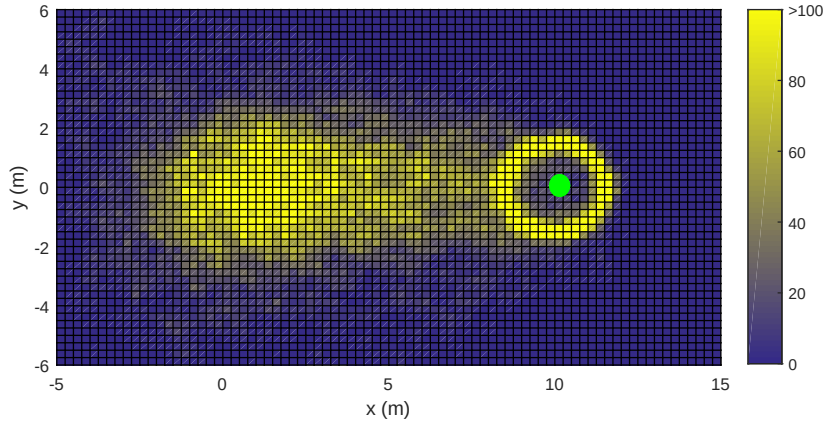


Figure 5-6: Compare: A two dimensional density plot, where the density tells how many times a mobile robot was inside that area of 0.25x0.25 meter. Areas with a counting higher than 100 are all yellow. The case: Goal search with the Comparealgorithm, where the swarm starts around $(0, 0)$ and the goal is positioned at $(0, 10)$.

The result of the compare algorithm as showed in Figure 5-6 shows also a quite satisfying behaviour of the swarm compared to the Chemotaxis algorithm. Although there is not much difference between the Compare (Figure 5-6) and m-PSO algorithm (Figure 5-4) some things can be said. The Compare algorithm gains some momentum towards the goal and the m-PSO has a constant movement towards the goal. This behavior is of most likely correlated with the shape of the VPF, because the surrounding VPF P_{surr} its gradient increases when getting closer towards the goal. With the Compare algorithm this will result in larger amplitudes of the heading direction F_{surr} near the goal, where in the case of the m-PSO the amplitude stays constant over all time.

From the previous results it can be concluded that the Chemotaxis is less suitable to navigate through the VPF of the surrounding. Therefore the next tests are taken with only the use of the m-PSO and Compare algorithm. The setup of the test makes use of the same goal point, but with an obstacle beacon between the initial position and the goal. This is a classic case to create a local minima in front of the obstacle. By deciding to use streaming functions these local minima will not occur, but now a saddle point is created. It is now interesting to see

what the results will be to find out if the streaming functions are indeed a way to deal with local minima. For the tests $N = 50$ simulation runs of $T = 500$ time samples long were taken. The first result of the the m-PSO is given below.

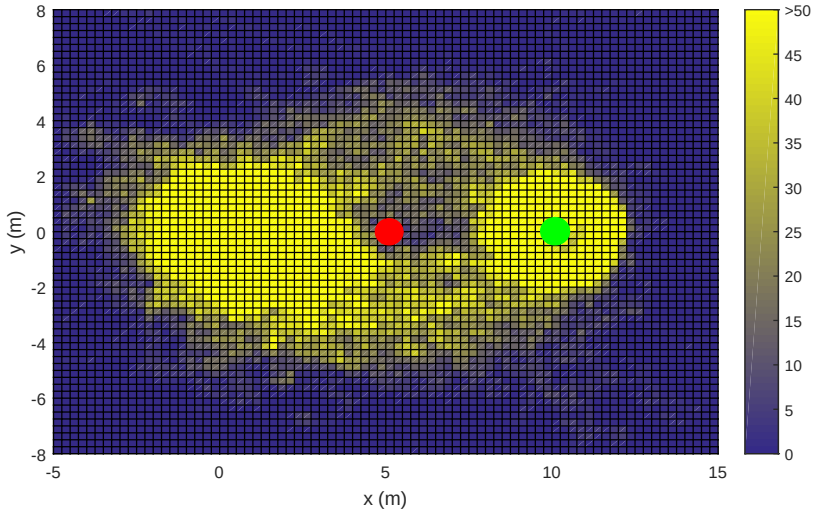


Figure 5-7: Case: An obstacle between the initial position of the swarm and the goal using the m-PSO algorithm. A two dimensional density plot, where the density tells how many times a mobile robot was inside that area of 0.25x0.25 meter. Areas with a counting higher than 50 are all yellow.

The result is a desired behaviour that guides the swarm around the obstacle towards the goal, which proves that the swarm does not get stuck inside a local minima. Due to a saddle point as the replacement of the local minima it is hard to find a clear direction, therefore a large cloud exists in front of the obstacle. This large cloud also includes the initial process of locating each other and positioning each other at desired distances before dealing with the surrounding. Furthermore there is a slight chance that a swarm member walks very near/through the obstacle, which is caused by worse localization results due to noisy measurements. Moreover the choice of route to go around the obstacle is chosen randomly and depends on a sudden better OG concentration measurement of one of the swarm members. With the same setup the Compare algorithm is used for comparison with the modified PSO algorithm, see Figure 5-8.

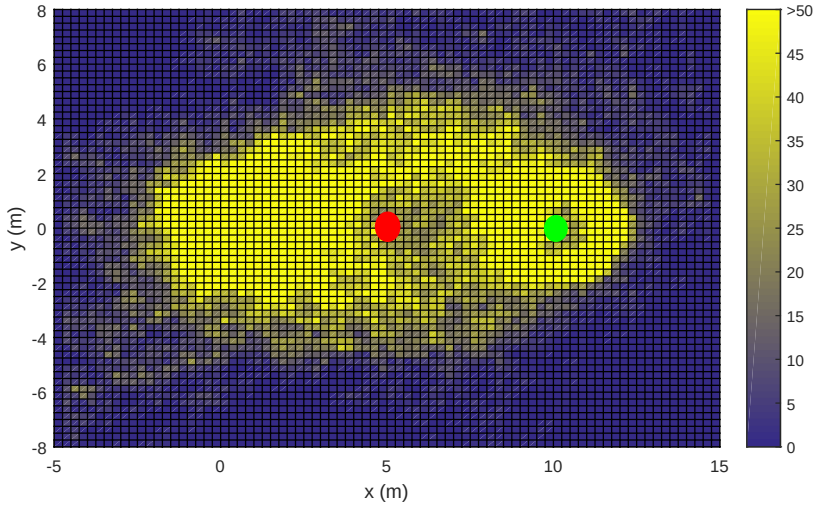


Figure 5-8: Case: An obstacle between the initial position of the swarm and the goal using the Compare method. A two dimensional density plot, where the density tells how many times a mobile robot was inside that area of 0.25x0.25 meter. Areas with a counting higher than 50 are all yellow.

The result of the Compare method shows a similar behaviour as with the m-PSO algorithm. A slight difference can be seen in the spread of the walk paths. In case of the Compare algorithm the density of the walked path is a little more widely spread. An increase of the priority to stay together did not decrease the spread of the walk path. In conclusion, the m-PSO algorithm shows some better results.

The next simulation is taken to show and test the behaviour of the swarm in a bounded workspace and in particular the case of the Cyberzoo at the TU Delft is taken. This requires a system where all the mobile robot stay inside a virtual cage, see Figure 5-9.

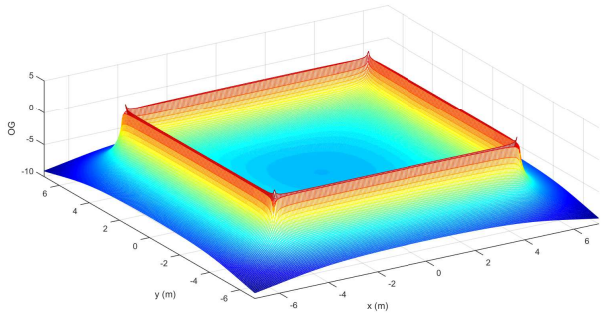


Figure 5-9: A representation of the virtual cage created by four line obstacles of each 10 meters long.

This cage is constructed by four beacons placed on the corners of a 5x5 meter square and connected as a line obstacle from the previous section. The main requirement it that the mobile robots stay inside this virtual cage, therefore the analysis will mainly focus on the chance of swarm members walking outside the virtual cage. A test will be done using 50 runs with the chosen PSO algorithm of each $T = 500$ time samples long. 50 runs are taken to assure enough different cases of initial localization and convergence sequences.

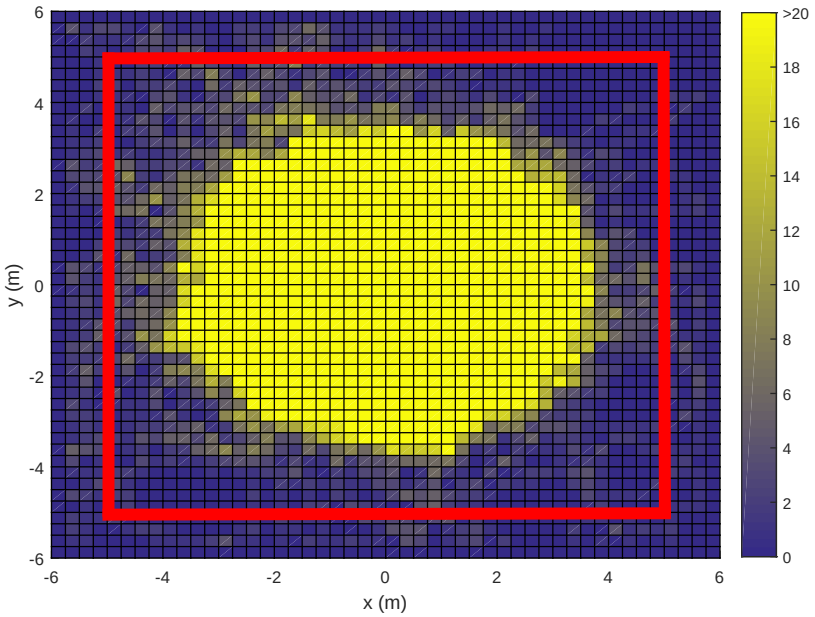


Figure 5-10: Case: A virtual cage with the m-PSO algorithm for 50 runs of $T = 500$. Only 0.65% of the positions of mobile robots were outside the cage.

From results given in the position density figure above it can be seen that most of the swarm members stay inside the virtual cage. Only 0.65% of the positions of the mobile robots were outside the cage. From single simulation runs it can be seen that if a mobile robot walks outside the cage it convergence back most of the time. In this simulation a downside of using a m-PSO, Chemtaxis or Compare algorithm can be seen, this has to do with not knowing exactly the position of the obstacle beacons. If a mobile robot walks outside the cage it can not distinguish if it is still inside the cage or outside the cage, because it only sees differences between OG concentration values of itself and the others. Only knowing these it can not know if there is a large peak of concentration between itself and the rest of the swarm. In the future this should be solved by also looking at how the OG values of a single mobile robot changes over time. Moreover in the chapter about future work.

More details about the swarm behaviour and it interaction with the surrounding can be obtained by looking at figures like Figure 5-11.

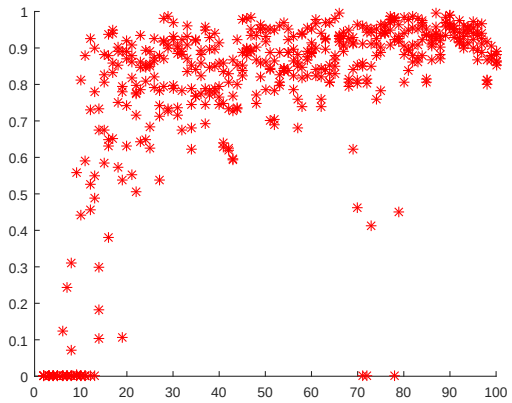


Figure 5-11: The priority ratio between swarm behaviour and interacting with the environment over time. A larger value means a higher priority to interact with the environment.

Figure 5-11 shows what priority is given between the VPF swarm controller and the VPF surrounding controller at each time sample k of each swarm member Z_i, given by

$$\text{Priority ratio} = \frac{\|\eta F_{surr,i}\|}{\| - \nabla P_{swarm,i}\| + \|\eta F_{surr,i}\|} \tag{5-13}$$

As one can see, the priority over time changes from a high swarm controller priority to one where the surrounding controller is dominant. The reason for that is that the swarm first needs to be established before trying to interact with the surrounding. From Figure 5-11 and 5-12 it is shown that the average priority ratio after the swarm is established is around 0.9. From multiple experiments the ratio of 0.9 results in a desired behaviour, therefore a guideline is to find a priority ratio near 0.9 by changing the priority parameter η. In the next figure another example of the balance between the VPF swarm controller and VPF surrounding controller is given.

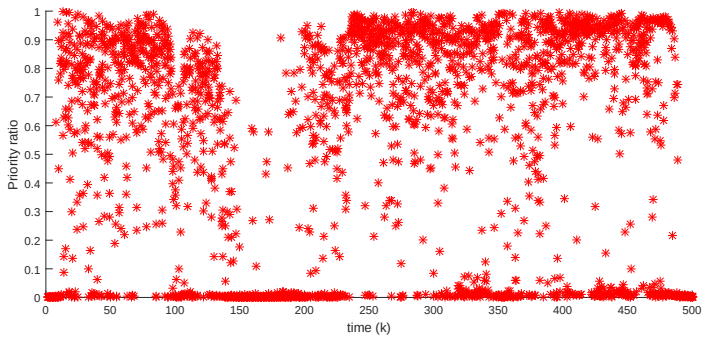


Figure 5-12: The priority graph over time corresponding to the swarm movement of Figure 5-13

with the according swarm movement given in Figure 5-13

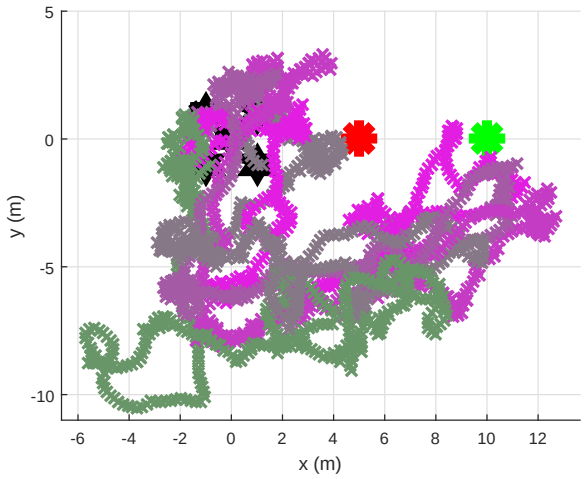


Figure 5-13: Movement of each swarm member for a single run of the Obstacle-goal setup using the m-PSO and the initial positions are given by the black spots.

If Figure 5-12 a large dip in the graph is seen. This dip corresponds with the moment when a swarm member (green) walks to far from the other members and therefore a high priority has been given to keep the swarm together.

After all the simulation test with different navigation algorithms some conclusions can be given. First is that the Chemotaxis algorithm has a to small gradient to give a clear navigation direction. Increasing the priority of the VPF surrounding controller only created worse results. Moreover, the m-PSO algorithm showed an average constant speed of the swarm and the speed of the swarm using the Compare algorithm depended on the gradient of the VPF of the surrounding. Both m-PSO and Compare algorithms succeeded to guide the swarm around an obstacle towards a goal. The m-PSO had a slightly better results than the Compare algorithm, because the spread of the walk path towards to goal was less. A larger spread of the walked path also a relates with a larger chance of colliding with the obstacle, therefore the m-PSO was chosen to be better. In the last simulation test a virtual cage was constructed and the m-PSO algorithm was used a the navigation algorithm. The swarm stayed quite well inside the cage, only 0.65% of the positions of the swarm members were outside the box and if so most of the time converged back to the inside of the cage.

5-4 Conclusions

This chapter had the goal to give the swarm a purpose to full fill. This purpose was split in to two types of missions: Exploration and goal search. The surrounding for exploration and goal search contains obstacles and possible goal beacons given by radio transmitting beacons. For navigation with a environment filled with obstacles and possible goals also a VPF is created, where the obstacles are like hills and the goal is a valley. In this case only the distances

towards the beacons are known and no localization is performed to find their exact position. In that case every mobile robot only knows it height/concentration value OG of the VPF. Three navigation algorithm were presented to navigate using only these values. First a modification of the Particle swarm optimization and second a method called Chemotaxis which uses an estimated gradient of the field. The last one compare let the mobile robots compare their own OG values with the others to decide which direction to go.

Simulation tests were taken to test these three navigation algorithms. The results were given by a probability density like graph giving an indication of the probability of the navigation route. For the first simulation test only a goal was searched. It was seen that the Chemotaxis was less suitable than the m-PSO and Compare algorithm. The cause might have been a to small and noisy gradient which could not give a clear heading direction F_{surr}. Furthermore the streaming function used for the VPF to represent the surrounding indeed showed no sign of local minima. Remaining saddlepoints at the local minima were overcome by the m-PSO and Compare algorithms. These simulation tests used a classical case to create local minima by placing an obstacle between the swarm and goal. Still there is a slight chance that a mobile robot walks near/through an obstacle, but this is due to worse position estimations caused by large measurement noises. In conclusion the m-PSO algorithm was chosen to be a better navigation algorithm, because it has less spread of the walk path towards the goal beacon. Using the m-PSO algorithm a last simulation test was performed wereby the swarm had to stay inside a virtual cage. The result was a swarm that mostly stayed inside this virtual cage, only 0.65% of the mobile robot positions were outside the cage. A particular problem was seen in this test, which is that a mobile robot does not know if it is in the cage or not. When only the OG values are known one can not say if there are large peaks or walls between them. Solutions to solve this are given in the chapter Future work.

The last subject to analyse was to look in to the "mind" of swarm member on how to make the decide between stay with the swarm or to act based on the surrounding. This balance was presented by the priority graph, which showed the difference in amplitude between the vectors of the VFP swarm controller $-\nabla P_{swarm}$ and the VPF surrounding controller F_{surr}. The result was that when the swarm falls apart or the swarm members are not at the desired distance from each other the VPF swarm controller indeed got more priority. From empirical results it was concluded that a priority ratio between 0.6 and 0.8 is desired.

In the end the simulation results showed that with large measurement noises a mobile robot and its swarm are able to navigate around an environment placed with obstacle and/or goal beacons.

Chapter 6

Conclusions and future work

6-1 Conclusions

From results in thesis multiple conclusions can be drawn, which will be presented in this section. These conclusions are devided into four topics that lead to the design of the swarm controller that uses Radio Signal Strength (RSS) measurements to have an indication of the distance between mobile robots. The first chapter described the Zebro robot and its movements using a general model and also a model that gives the relation between RSS measurements and distance. Secondly, a localization algorithm was designed to estimate the relative positions of the other mobile robots. With the use of these estimated relative positions of the other mobile robots a distributed Virtual Potential Field (VPF) controller is designed. The VPF swarm controller has as goal to create a swarm with stable cohesion and no collisions between swarm members. In continuation of the design of the VPF swarm controller it was investigated to see what the effects of uncertain position estimations is on the cohesion of the swarm. In the end another VPF controller is used to let the swarm interact with its environment. In this thesis the goal is avoiding obstacles and staying in a bounded area. Also the effects of uncertain distance measurements and position estimations on the walked path of the swarm are studied. These topics all had there own specific challenges and problems to deal with, which will be shortly outlined with their corresponding solutions.

The swarm considered in this thesis consist of mobile robots called Zebros. These Zebro robots are six-legged "cockroach" like walking robots, whose motion model in general is the same as a so-called two-wheeled robot. Besides the movement model of the mobile robot itself a relative motion model is necessary. The relative motion model gives the movement of the other mobile robots from the perspective of a mobile robot in question. The necessity of the relative motion model comes from the fact that all the mobile robots do not know their absolute positions

in the working space, because there is no reference available to create an absolute coordinate frame.

Next a localization algorithm is designed to find the relative position of the other mobile robots. Having information about the relative positions of the other mobile robots can be a valuable addition to the already existing distributed swarm controllers. In literature a localization with the use of only distance measurements is called trilaterations. Most trilateration algorithms consider well measured distances, which in this thesis is not the case. That is, because the distance measured uses a receiving antenna to measure the Radio Signal Strength (RSS) of a message coming from a transmitting antenna on another mobile robot. As model for the RSS-to-distance a Line-Of-Sight (LOS) model is used that has a RSS noise standard variance of $\sigma_I = 3.487$dB, which corresponds to a distance standard variance noise around $\sigma_r = 25\%$ of the measured distance $\bar{r}$. A large distance measurement noise was expected and considered to be an important topic to keep in mind for the creation of a localization algorithm.

The localization of the relative positions was proven not to be possible with only distance measurements, therefore dead-reckoning was introduced. Dead-reckoning is a method that tracks the walk path of a mobile robot with the use of only sensors onboard the mobile robot. Using both distance and dead-reckoning measurements a localization tries to find the polar coordinates of another mobile robot given by an orientation θ and a distance v. Polar coordinates are used, because this will lead to better approximations of the position variances which also leads to improvement of the position estimation. Two algorithms were designed to do this localization: A modified weighted Multi-dimensional Scaling localization (weighted-MDS) and the Multi-Robot Relative Pose Localization (MRRPL). Both methods use a feature of the system to communicate the distances between two other mobile robots. For the weighted-MDS the tracked path of a mobile robot is converted into distances. Combined with the measured distances between all the mobile robots an optimization algorithm tries to find the best fit over all these measured distances. The other localization method MRRPL uses the functions of the relative pose estimation and multi-robot localization in an Unscented Kalman Filter (UKF) setting. The result of the performance of both localization algorithms it was found out that the weighted-MDS had problems to estimate the orientation θ. The MRRPL proved to be a better candidate, because the MRRPL showed better results and importantly also signs of being a converging position estimator. By finding better values of the noise variance matrices R and Q of the UKF the MRRPL could even improve more.

In the end an orientation estimation RMS error of 0.8976 rad and a relative distance estimation RMS error of 13 % was established at a RSS noise standard deviation of $\sigma_I = 3$dB. Simulation tests also showed that the distance measurement noise σ_I has a great influence on estimation of the orientation θ and distance v. Additionally the amplitude of the noise of heading direction σ_v had only a clear influence of the estimation position angle θ. So to improve the position estimation the distance and heading direction measurements are the main factors that needs to be improved.

When having a estimated position of the other mobile robots using the MRRPL a distributed VPF swarm controller can be designed. The VPF swarm controller is based on using a VPF P

which can be imagined as a mountain site, where obstacles represent mountains and a valley corresponds to an attractive point of interest (goal). The greatest descent vector $-\nabla P$ of this VPF is the heading direction of the mobile robot. Streaming functions were used to formulate the VPFs, because they eliminate the local minimum problem. A VPF to describe another member of the swarm is a combination of the mobile robot being a repulsive obstacle and having a circular wall around it that let the other mobile robots stay close. Doing so will lead to a cohesive behaviour of the swarm by staying close to each other with no collisions between each other. From early results it was seen that a correlation exists between the cohesion of the swarm and the noise variance matrices R and Q of the MRRPL. By a fine tuning which also took the cohesion of the swarm into consideration a more optimal choice of the noise variance matrices R and Q was found. With the retuned matrices R and Q a VPF swarm controller was established. Simulation tests showed that with high noise on the distance and dead-reckoning measurements it is still possible to use a VPF swarm controller.

Next a task was given to the robot swarm. Another distributed VPF controller was created to let the swarm interact with the surrounding. The surroundings contains radio beacons with unknown positions presenting virtual obstacles or walls. Also the surrounding can contain attractive goal points. The VPF swarm controller is used to perform cohesion of the swarm while the VPF surrounding controller let the swarm interact with the surroundings. Each mobile robot only knows a value OG that represents the "height" in the VPF of the surroundings. Larger OG values indicate an obstacle is near. Knowing the positions and OG values of each mobile robot three methods were found to give a heading vector F_{surr} to add to the VPF swarm control heading $-\nabla P_{\text{swarm}}$. These three methods are: a modified Particle Swarm Optimization (m-PSO), Chemotaxis and a method that compares a mobile robot its OG value with the OG values of the other swarm members (Compare). These three methods were tested with multiple cases and concluded that the Chemotaxis approach is not suitable, but the m-PSO and Compare algorithms are. Also from all simulations is was seen that the localization gives position information that is valuable enough as an addition to swarm algorithms that do not use the exact positions. Still the high noise of the distance measurements give a slight chance of wrong position estimations. These wrong position estimations let a mobile robot going near/through obstacles or it walk to far away from the swarm. In almost all these cases the heading of the mobile robot converged back to the desired heading direction as a result of a converging localization.

In the end it is proven that a localization algorithm can estimate the positions of the other mobile robots and it can be a valuable addition to the already existing distributed VPF swarm controllers that only uses RSS distance measurements and optimization techniques. The localization can therefore be used to generate better heading directions by knowing the positions of the other swarm members and knowing information gained as a the swarm in total. Considering the challenge of dealing with the large amount of measurement noises on the RSS-to-distance and dead-reckoning this is a beneficial result.

6-2 Future work

Although the results of this thesis are quite satisfying some problems occurred and there are some points that need some extra attention in the future of the robot swarm project. The first topic that needs some additional attention is the RSS-to-distance model. The experiment data used to create the RSS-to-distance model showed some unexplainable results. It was as if the RSS measurements $\bar{I}$ of the messages from the transmitting antenna had two values, because two distinct RSS levels were measured at each distance r.

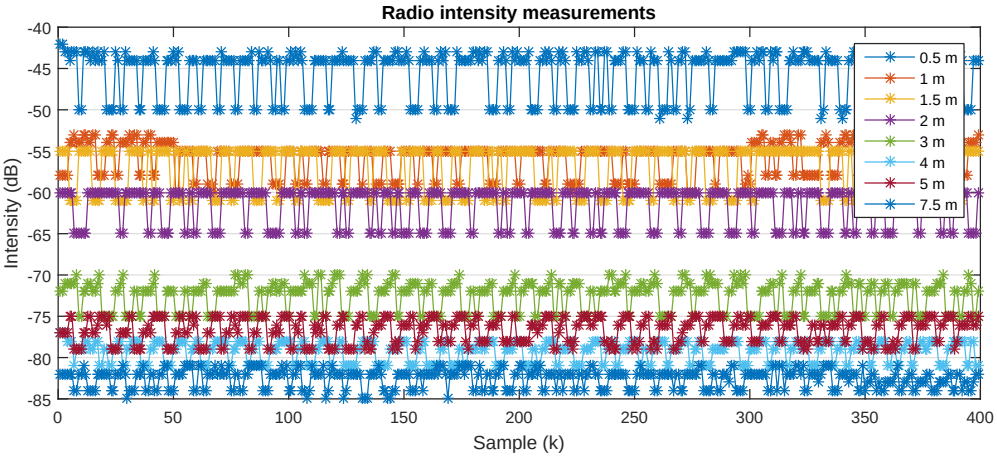


Figure 6-1: Test results of the RSS-to-Distance experiment from one of the two tests. Each colour represents the RSS data at a specific distance. The experiment shows unexplainable two distinct levels of RSS for each distance.

With improvements to solve the unexplainable two RSS levels the noise σ_I could become less, which would increase the distance estimation. Although it will become better it is still necessary to create a localization algorithm that can deal with larger noises, because in real implementations the distance measurement noise can be much larger in more complex environments. Although it is still preferred to let the mobile robot swarm operate in an environment that is not too complex, because a not very complex environment is needed to preserve the validity of the Line-Of-Sight (LOS) model. The boundary on how complex an environment needs to be such that a LOS model is still accurate enough is unknown.

Another topic of improvement is to find better noise variances matrices Q and R. The optimal Q and especially R matrices are highly depend on measurement noise variances. It is still useful to know the measurement noise variances, but in this case not the distance measurement noise variance σ_I and dead-reckoning noise variances σ_θ and σ_v. Why these are not the noise variances to use will be explained.

Remember that the Unscented Kalman Filter (UKF) uses the measurements of the multi-robot trilateration (MRT) and relative pose estimation (RPE) given by the following equations, more

details in Section 3-6-1.

$$\bar{y}^m_{i,j,k}(k+1) = \bar{r}_{i,j}(k+1)^2 - \bar{r}_{i,k}(k+1)^2 \tag{6-1}$$

and

$$\bar{y}^r_{i,m}(k+1) = 0.5(\bar{r}^2_{0,i,m} - \bar{r}^2_{0,i,0} - V^T_{i,m}V_{i,m} - V^T_{0,m}V_{0,m}), \tag{6-2}$$

The noise variances of the two measurements $\bar{y}^m_{i,j,k}$ and$\bar{y}^r_{i,m}$ are now the noise variances to be used to provide the optimal noise variance matrices Q and R. The noise variances for $\bar{y}^m_{i,j,k}$ and$\bar{y}^r_{i,m}$ where given in 3-6-1, but as a combination of σ_I, σ_θ and σ_v. It is better to find the noise variances of $\bar{y}^m_{i,j,k}$ and$\bar{y}^r_{i,m}$ themselves. Especially when it varies with the time and surroundings. Finding an accurate value of these noise powers is hard. Already Kalman Filter adaptations exist to estimate the measurement noises on $\bar{y}^m_{i,j,k}$ and $\bar{y}^r_{i,m}$ on-line. One is the Adaptive Kalman Filter (AKF) [34]. The downside of the AKF is that it is a modification of a linear Kalman Filter and not of the non-linear Unscented Kalman Filter (UKF). In literature an adaptive modification for the UKF was not found, but would certainly be interesting to test. In that case the localization algorithm can deal with changing environments over time. As said before in Section 3-6-2 a particle filter can used, but the use of the particle filter might be to computational complex.

Next a small issue was found in the use of polar coordinates as the position states, especially with the orientation θ. Inside the UKF multiple state predictions $\hat{X}(k|k-1)$ are created which form a cloud of predicted states. The mean predicted state $\hat{\bar{x}}(k|k-1)$ is the average of all these predicted states. If an average of the angle θ is taken of a specific set of predicted angles a problem occurs, see Figure 6-2.

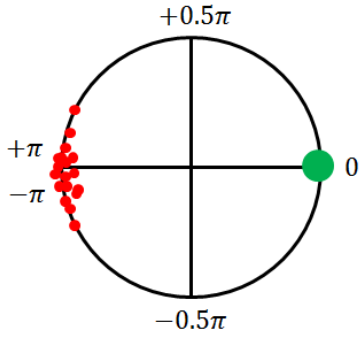


Figure 6-2: A possible situation of taking the average angle (green) of multiple predicted angles (red)

The estimated average angle $\hat{\theta}$ is now 180 degrees off and will result in a heading direction 180 degrees off, which also results in a worse behaviour of the swarm member. Sometimes this happened, but with enough extra measurements and walking directions it converged back towards the real orientation. Occasionally it did not converge back, because due to the resulting wrong walking direction the swarm member walked too far from the swarm to have measurements that are well enough to let the estimate angle $\hat{\theta}$ converge back to the real angle θ. This can be

solved by converting the states to Cartesian states, then take the average and convert them back to polar coordinates. Due to the large amount of estimated state vectors $\hat{X}$ that have to be averaged this method will increase the computation time. If another simpler solution is found to take the average of the angles θ this would improve the localization algorithm.

A major topic for improvement comes from the problem that in the final results it was possible for a mobile robot to walk through point and line obstacles. The reason this happens comes from the fact that the complete swarm only knows the OG-values at the positions of the mobile robots and therefore no information about the surrounding VPF between mobile robots is known as in Figure 6-4.

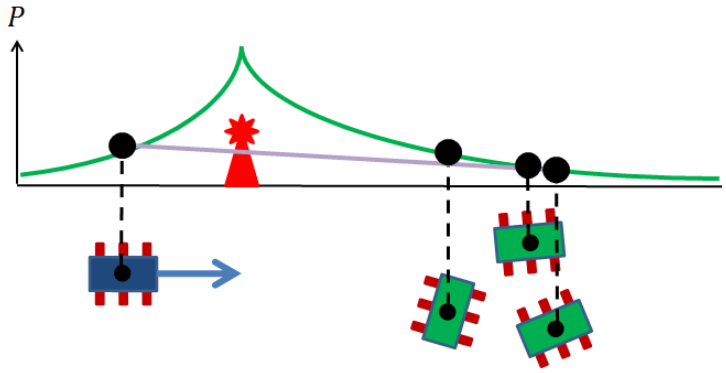


Figure 6-3: A case where the OG-values give an estimation of the surrounding VPF which is not true. In this case the blue mobile robot will walk through the obstacle, because the mobile robots on the other side of the wall have lower OG-values.

The solution lies in the fact that the mobile robot also needs to make individual decisions based on OG-values it had measured earlier. Doing so a mobile robot can have an indication of it going the wrong way, see the figure below.

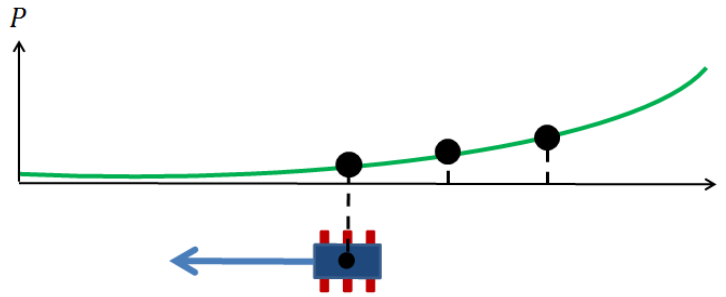


Figure 6-4: A representation where the mobile robot remembers the OG-values of the past.

One measurement available to the mobile robot not used yet for the VPF surrounding controller is the measured walked path by performing dead-reckoning. An option to act based

on this knowledge can be an additional individual VPF, which will be constructed by placing virtual obstacle points on previously tracked coordinates with higher OG-values. These tracked coordinates were saved in the dead-reckoning position matrix $\tilde{V}$. Doing so the heading direction will possibly be pointed away from obstacles.

Based on the main topic of this thesis the best solution should be found in using the localization algorithm as an additional measurement to provide better heading directions. In the future the resulted heading direction of the complete swarm controller F_{tot} of this thesis will be added to swarm optimization algorithms that only uses distance estimations. Therefore in the future the localization algorithm needs to be combined with the existing swarming algorithms. This should be done in a way that the existing swarm algorithm always can be performed and the localization algorithm will provide additional information to improve the choice of the heading direction. Doing so a mobile robot can make decisions based on both position estimation and distance measurements and the mobile robot makes decisions based on individual and group knowledge.

The complete algorithm to create a swarm of mobile robots which is a real wish of most researchers working on this subject. This thesis will provide extra insights about how to give more information about the position of the swarm members and how the resulting noisy position estimations effects the already existing VPF swarm and VPF surrounding controllers.

Appendix A

Radio signal strength to distance measurements

This appendix gives an overview on how the model parameters for the RSS to distance r are found by experimental measurements. The measurements are taken with a transmitter which sends constant open messages and a receiver which receives the messages and measures the RSS from the Radio Signal Strength Indicator (RSSI). The transmitter and receiver used are both *Bluegiga BLE113 Bluetooth Smart Modules* and the attached circuit board is designed by the *Electronics Research Laboratory, TU Delft*. The communication between the antenna's is based on the 2.4GHz Bluetooth IEEE 802.15.1 protocol. The transmitter had a omni-directional antenna attached to it, therefore the transmitter can be placed with every angle relative to the receiver. The receiver antenna is not omni-directional, therefore the angle towards the transmitters stayed constant. The computer board has a microSD-card reader connected to it to save the data received from the RSSI. All the documents about this setup are attached to this thesis.

The measurements where taken as follows. Place the transmitter and receiver from a chosen distance to each other. Push the "start" button and walk away from the receiver within 10 seconds. This is necessary, because the human body blocks 2.4 GHz signal significantly. The recording indicator LED lights up, and wait until the recording indicator light goes off. Now 400 measurements were taken. Repeat the whole sequence for every distance measured. A picture of the surrounding where the measurement took place is given in Figure A-1, which is clearly a non-complex environment.

Figure A-1: The environment of the RSS-to-distance experiments.

The next figure gives the result of the measurements taken at different distances. The maximum distance seemed to be around 10 meters.

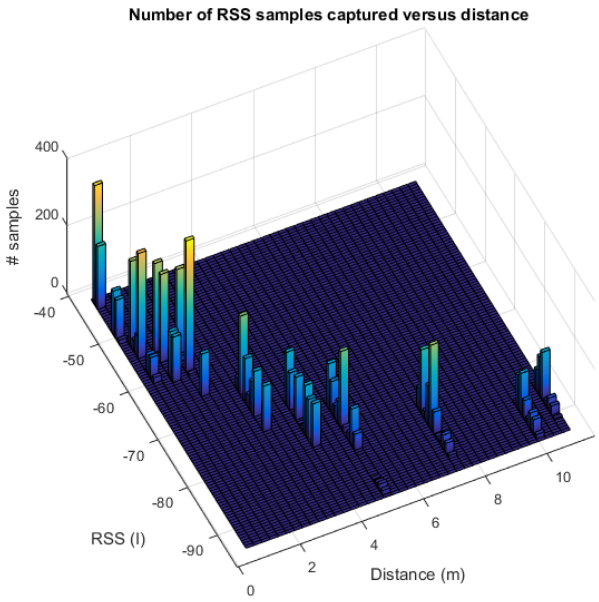


Figure A-2: A histogram of the experimental measurements. The height indicates how many of the packages received had a certain RSS. This was done for every distance r

The second graph follows the previous graph, but the height representing the amount of measured samples is removed. This graph shows the result of the RSS-to-distance model fitting. The chosen model is

$$I(r) = I_0 - 10\beta \log_{10}(r) + \nu_I, \tag{A-1}$$

where ν_I has a noise variance of σ_I. Then the fitted parameters where

$$I_0 = -53.71\text{dB} \quad \beta = 3.424 \quad \sigma_I = 3.487\text{dB}$$

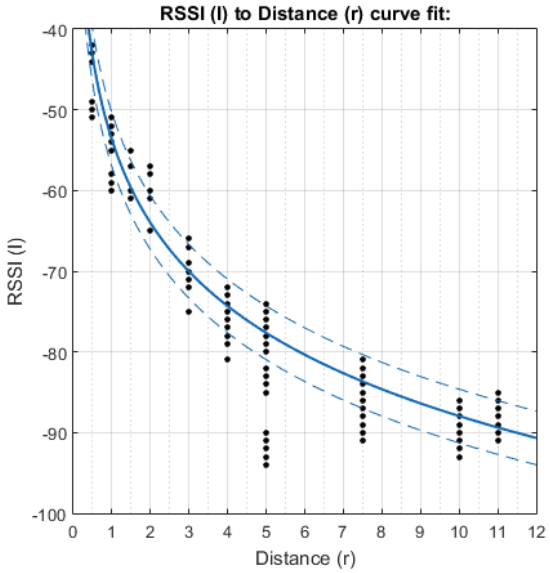


Figure A-3: The result of the RSS-to-distance model fitting.

Appendix B

Usable sensors to perform dead-reckoning

In this section an overview about sensors to use for dead-reckoning is given. Dead-reckoning is the method to measure the walked path of a mobile robot with sensors that are placed on the mobile robot itself. Performing dead-reckoning is necessary, because distance measurements were not enough to localize neighbouring mobile robots. For dead-reckoning in this thesis two measurements are used: the covered distance $\bar{v}$ and the corresponding heading angle $\bar{\theta}$ per time step. Polar coordinate are used, because most sensors are based on angles and covered distance.

To perform dead-reckoning by measuring the heading direction and covered distance multiple sensors can be used. A requirement is that these sensors are relatively cheap, do not use a lot of energy, low computational power and can work on all kinds of surfaces. A short overview of sensors according to these requirements is given:

- Accelerometer: A sensor to measures accelerations and when integrated twice a distance can be derived. Due of the necessity of a double integrator this sensor is very sensitive to noise and jerky movements of a mobile robot.
- Gyroscope: A sensor that measures all three rotational speeds. This sensor needs only one integrator to measure the angle of the heading direction $\bar{\theta}$. This sensor is also sensitive to jerky movements
- Magnetometer: The magnetometer is a sensor that measures the three angles towards the earth's magnetic north and therefore can be used to measure the relative angle displacement. The downside is that this sensor is sensitive to for example metal holding objects nearby or even the metal components in the robot itself.

- Counting steps: One can count the steps to know what the travelled distance is if the travelled distance per step is known. In case of the Zebro the travelled distance per step is not constant due to a large chance of slip.
- Light sensors: When multiple light sensors are placed in a circle they can detect a light shift through the sensors resulting in knowing the relative changed angle.
- Optical sensor: An optical sensor can be used to scan the ground and measure the difference between frames and by doing so measuring a displacement. Most of these sensors have a focal point, which is undesired due to non-flat surfaces.
- Rolling wheel: A rolling wheel mounted for example on a carriage can be used to measure the travelled distance. Due to jerky movements and rough surface this will not make enough contact with the surface and therefore distance measurements become inaccurate.

In conclusion to perform dead-reckoning there is not a single sensor that could measure the travelled distance v or heading direction θ for a Zebro robot well enough. Therefore the use of multiple sensors are required. The minimal requirements for the accuracy of the combined dead-reckoning systems are given in Section 3-7-3.

Dead-reckoning has measurement noises that create uncertainties on the path travelled. An impression of the uncertainty of the travelled path is given in Figure B-1.

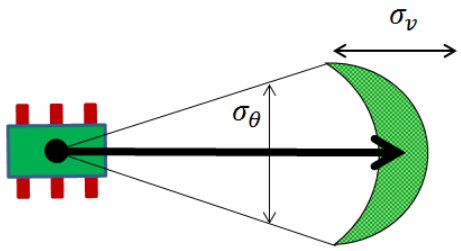


Figure B-1: An indication of the measured walked path uncertainty. Where σ_v is the noise variance of the covered distance σ_v and σ_θ the noise variance of the heading direction.

Appendix C

Mathematics

C-1 Obtaining σ_r and σ_r^*

First the expression for the distance noise variance σ_r based on the intensity noise variance σ_I is given and secondly the expression for noise standard deviation of r^2. Both are obtained by using a linear approximation. A second order approximation is not required, because the linear approximation was already near enough the real value and the distance measurements are already to noisy too give a clear indication.

$$r = g(I) \qquad \sigma_r = \sigma_I \frac{\partial g(I)}{\partial I}\Big|_{I=\mu_I} \tag{C-1}$$

$$g(I) = 10^{-(I-I_0)/(10\beta)} \tag{C-2}$$

$$\frac{\partial g(I)}{\partial I} = 10^{-(I-I_0)/(10\beta)} \log(10) \frac{1}{10\beta} \tag{C-3}$$

$$= r \log(10) \frac{1}{10\beta} \tag{C-4}$$

$$\sigma_r = r \log(10) \frac{1}{10\beta} \sigma_I \tag{C-5}$$

The next calculations create an expression for the standard deviation σ_r^* of the squared distance r^2 based on σ_I.

$$r^2 = g(I)^2 \qquad \sigma_r^* = \sigma_I \frac{\partial g(I)^2}{\partial I}\Big|_{I=\mu_I} \tag{C-6}$$

$$g(I)^2 = 10^{-(I-I_0)/(10\beta)\cdot 2} \tag{C-7}$$

$$\frac{\partial g(I)^2}{\partial I} = 10^{-(I-I_0)/(10\beta)\cdot 2} 2\log(10)\frac{1}{10\beta} \tag{C-8}$$

$$= r^2 \log(10)\frac{1}{5\beta} \tag{C-9}$$

$$\sigma_r^* = r^2 \log(10)\frac{1}{5\beta}\sigma_I \tag{C-10}$$

C-2 Variance of a squared normal distribution

When $X \sim \mathcal{N}(0, \sigma^2)$, what is the variance of $Y = X^2$? Using the Chi-squared distribution it follows that

$$X^2 \sim \sigma^2 \chi_1^2, \tag{C-11}$$

with χ_1^2 as the one degree of freedom Chi-distribution. Since $E[\chi_1^2] = 1$ and $Var[\chi_1^2] = 2$ the result is

$$E[X^2] = \sigma^2 \tag{C-12}$$

$$Var[X^2] = 2\sigma^4 \tag{C-13}$$

C-3 Prove of streaming function constraint on λ, Equation (4-7)

It needs to be proven that when a streaming function for a VPF is used the attractive goal is always the global minimum. Therefore the attraction of the goal is always larger than the repulsive forces at infinity distances. For proving the requirements of λ_g and λ_O point, circular and wall obstacles are considered. First the requirements for λ_g and $\lambda_{O.i}$ are given if only point

obstacles based on $\lambda_i \log(|X_Z - X_i|)$ is considered.

$$P = -\lambda_g \log(|X_Z - X_g|) + \sum_i^O \lambda_i \log(|X_Z - X_i|) \quad \text{(C-14)}$$

$$\lim_{X_Z \to \infty} \lambda_g \log(|X_Z - X_g|) > \lim_{X_Z \to \infty} \sum_i^O \lambda_i \log(|X_Z - X_i|) \quad \text{(C-15)}$$

$$\lim_{X_Z \to \infty} \frac{\lambda_g \log(|X_Z - X_g|)}{\sum_i^O \lambda_i \log(|X_Z - X_i|)} > 1 \quad \text{(C-16)}$$

$$\lim_{X_Z \to \infty} \log(|X_Z - X_g|) = \lim_{X_Z \to \infty} \log(|X_Z - X_i|) \quad \text{(C-17)}$$

$$\frac{\lambda_g}{\sum_i^O \lambda_i} > 1 \quad \text{(C-18)}$$

$$\lambda_g > \sum_i^O \lambda_i \quad \text{(C-19)}$$

Next the same in the case of circular obstacles $\lambda_i \log(|X_Z - X_i| - \mu_{min})$

$$\lim_{X_Z \to \infty} \frac{\lambda_g \log(|X_Z - X_g|)}{\sum_i^O \lambda_i \log(|X_Z - X_i| - \mu_{min})} > 1 \quad \text{(C-20)}$$

$$\lim_{X_Z \to \infty} \log(|X_Z - X_g|) = \lim_{X_Z \to \infty} \log(|X_Z - X_i| - \mu_{min}) \quad \text{(C-21)}$$

$$\frac{\lambda_g}{\sum_i^O \lambda_i} > 1 \quad \text{(C-22)}$$

$$\lambda_g > \sum_i^O \lambda_i \quad \text{(C-23)}$$

and last the wall obstacles given by $\log(|X_Z - X_i| + |X_Z - X_j| - \mu_{min})$

$$\lim_{X_Z \to \infty} \frac{\lambda_g \log(|X_Z - X_g|)}{\sum_i^L \lambda_i \log(|X_Z - X_i| + |X_Z - X_j| - \mu_{min})} > 1 \quad \text{(C-24)}$$

$$\lim_{X_Z \to \infty} \frac{|X_Z - X_i|}{|X_Z - X_j|} = 1 \quad \text{(C-25)}$$

$$\lim_{X_Z \to \infty} \frac{\lambda_g \log(|X_Z - X_g|)}{\sum_i^L \lambda_i \log(2|X_Z - X_i| - \mu_{min})} > 1 \quad \text{(C-26)}$$

$$\lim_{X_Z \to \infty} \log(2|X_Z - X_i| - \mu_{min}) = \quad \text{(C-27)}$$

$$\lim_{X_Z \to \infty} \log(2|X_Z - X_i|) = \quad \text{(C-28)}$$

$$\lim_{X_Z \to \infty} \log(2) + \log(|X_Z - X_i|)) = \lim_{X_Z \to \infty} \log(|X_Z - X_g|) \quad \text{(C-29)}$$

$$\frac{\lambda_g}{\sum_i^L \lambda_i} > 1 \quad \text{(C-30)}$$

$$\lambda_g > \sum_i^L \lambda_i \quad \text{(C-31)}$$

Important is that these results do not prove that there are no local minima in the VPF with circular or wall obstacles.

Appendix D

K_R and K_{ratio} estimation with the VPF swarm controller

The following graphs show the RMS errors of the estimated orientation θ and distance v using different gains for the noise variances matrices Q and R.

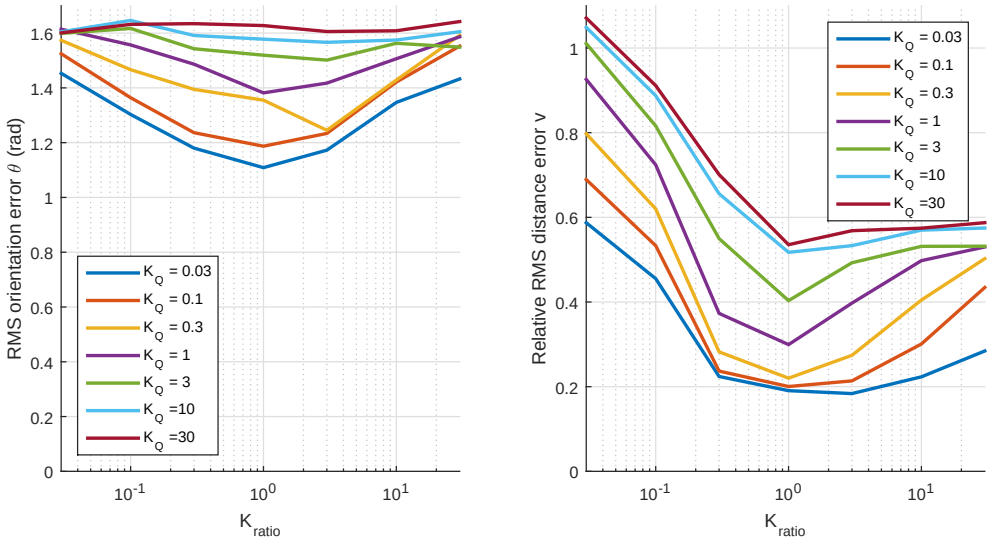


Figure D-1: $K_Q = 1$,$\sigma_\theta = 5°$,$\sigma_v = 0.1$

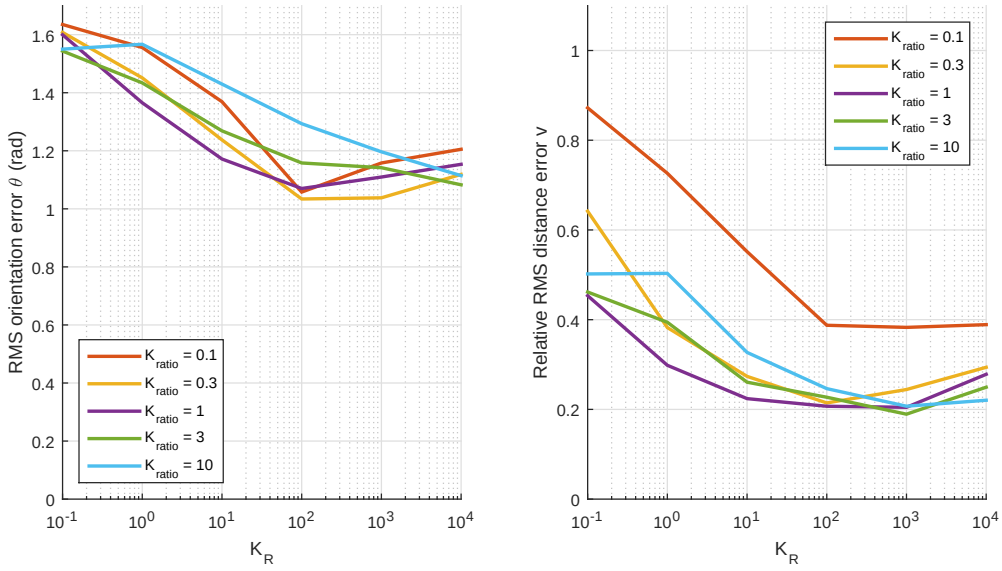


Figure D-2: $K_R = 1$,$\sigma_\theta = 5°$,$\sigma_v = 0.1$

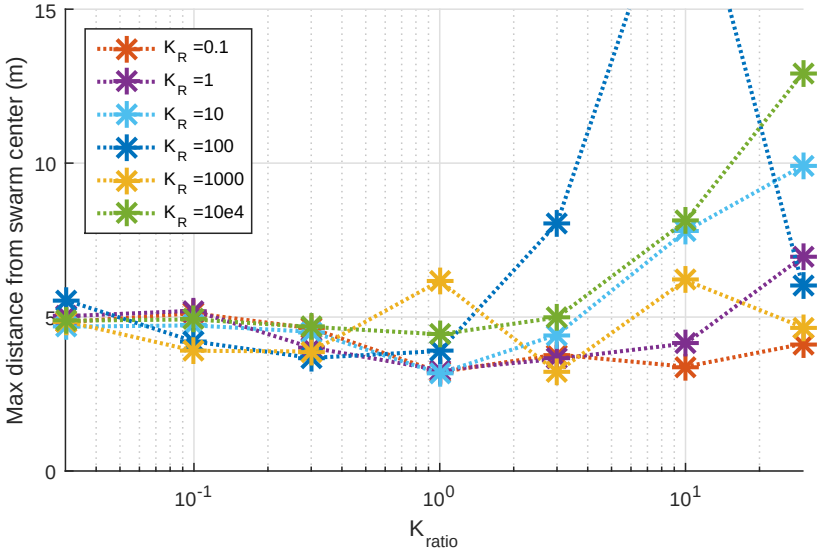


Figure D-3: $K_Q = 1$, $\sigma_\theta = 5°$,$\sigma_v = 0.1$

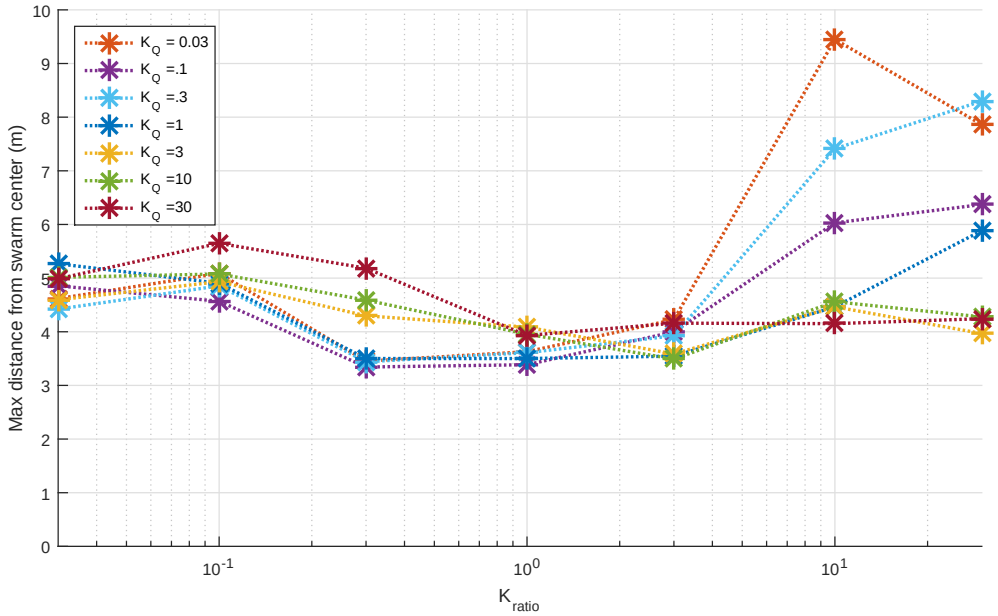


Figure D-4: $K_R = 1$,$\sigma_\theta = 5°$,$\sigma_v = 0.1$

Bibliography

[1] X. Zhou and S. Roumeliotis, "Robot-to-robot relative pose estimation from range measurements," *Robotics, IEEE Transactions on*, vol. 24, pp. 1379–1393, Dec 2008.

[2] J.-O. Kim and P. K. Khosla, "Real-time obstacle avoidance using harmonic potential functions," *Robotics and Automation, IEEE Transactions on*, vol. 8, no. 3, pp. 338–349, 1992.

[3] S. Waydo and R. M. Murray, "Vehicle motion planning using stream functions," in *Robotics and Automation. Proceedings. ICRA.*, vol. 2, pp. 2484–2491, IEEE, 2003.

[4] D. H. Stefanov, Z. Bien, and W.-C. Bang, "The smart house for older persons and persons with physical disabilities: structure, technology arrangements, and perspectives," *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, vol. 12, no. 2, pp. 228–250, 2004.

[5] M. P. Groover, *Automation, production systems, and computer-integrated manufacturing*. Prentice Hall Press, 2007.

[6] Y. Tan and Z. yang Zheng, "Research advance in swarm robotics," *Defence Technology*, vol. 9, no. 1, pp. 18 – 39, 2013.

[7] "A thousand kilobots self-assemble into complex shapes." http://spectrum.ieee.org/automaton/robotics/robotics-hardware/a-thousand-kilobots-self-assemble. Accessed: 3-12-2014.

[8] "I swarm you swarm we all swarm." http://spectrum.ieee.org/tech-talk/semiconductors/devices/i_swarm_you_swarm_we_all_swarm. Accessed: 3-12-2014.

[9] "Even brainless robots can show swarm behavior." http://spectrum.ieee.org/automaton/robotics/artificial-intelligence/even-brainless-robots-can-show-swarm-behavior. Accessed: 3-12-2014.

[10] C. W. Clark, "The acoustic repertoire of the southern right whale, a quantitative analysis," *Animal Behaviour*, vol. 30, no. 4, pp. 1060 – 1071, 1982.

[11] T. D. Wyatt, *Pheromones and animal behaviour: communication by smell and taste*. Cambridge University Press, 2003.

[12] "Zebro six-legged robot." http://www.robotics.tudelft.nl/?q=research/zebro-six-legged-robot. Accessed: 25-1-2014.

[13] R. Negenborn, *Robot localization and kalman filters*. PhD thesis, Utrecht University, 2003.

[14] J. Borenstein and L. Feng, *A method for measuring, comparing, and correcting dead-reckoning errors in mobile robots*. 1994.

[15] T.-C. Lee, K.-T. Song, C.-H. Lee, and C.-C. Teng, "Tracking control of unicycle-modeled mobile robots using a saturation feedback controller," *Control Systems Technology, IEEE Transactions on*, vol. 9, pp. 305–318, Mar 2001.

[16] B. Roberts and K. Pahlavan, "Site-specific RSS signature modeling for WiFi localization," in *Global Telecommunications Conference.*, pp. 1–6, IEEE, Nov 2009.

[17] P. Barry and A. Wiliamson, "Statistical model for UHF radio-wave signals within externally illuminated multistorey buildings," *Communications, Speech and Vision, IEE Proceedings I*, vol. 138, pp. 307–318, Aug 1991.

[18] A. De Toledo, A. Turkmani, and J. Parsons, "Estimating coverage of radio transmission into and within buildings at 900, 1800, and 2300 mhz," *Personal Communications, IEEE*, vol. 5, pp. 40–47, Apr 1998.

[19] M. Rahman and L. Kleeman, "Paired measurement localization: A robust approach for wireless localization," *Mobile Computing, IEEE Transactions on*, vol. 8, pp. 1087–1102, Aug 2009.

[20] G. Mao and B. Fidan, *Localization Algorithms and Strategies for Wireless Sensor Networks: Monitoring and Surveillance Techniques for Target Tracking*. Information Science Reference, 2009.

[21] N. Bulusu, J. Heidemann, and D. Estrin, "Gps-less low-cost outdoor localization for very small devices," *Personal Communications, IEEE*, vol. 7, no. 5, pp. 28–34, 2000.

[22] K. Chintalapudi, A. Padmanabha Iyer, and V. N. Padmanabhan, "Indoor localization without the pain," in *Proceedings of the sixteenth annual international conference on Mobile computing and networking*, pp. 173–184, ACM, 2010.

[23] A. Rai, K. K. Chintalapudi, V. N. Padmanabhan, and R. Sen, "Zee: zero-effort crowdsourcing for indoor localization," in *Proceedings of the 18th annual international conference on Mobile computing and networking*, pp. 293–304, ACM, 2012.

[24] Z.-X. Chen, H.-W. Wei, Q. Wan, S.-F. Ye, and W.-L. Yang, "A supplement to multidimensional scaling framework for mobile location: A unified view," *Signal Processing*, vol. 57, no. 5, pp. 2030–2034, 2009.

[25] J. Djugash, S. Singh, and B. Grocholsky, "Modeling mobile robot motion with polar representations," in *Intelligent Robots and Systems,. IEEE/RSJ International Conference on*, pp. 2096–2101, IEEE, 2009.

[26] J. A. Costa, N. Patwari, and A. O. Hero III, "Distributed weighted-multidimensional scaling for node localization in sensor networks," *ACM Transactions on Sensor Networks (TOSN)*, vol. 2, no. 1, pp. 39–64, 2006.

[27] J. Saloranta, D. Macagnano, and G. Abreu, "Interval-scaling for multitarget localization," in *Positioning Navigation and Communication (WPNC)*, pp. 59–64, March 2012.

[28] N. Sunderhauf, S. Lange, and P. Protzel, "Using the unscented kalman filter in mono-slam with inverse depth parametrization for autonomous airship control," in *Safety, Security and Rescue Robotics. IEEE International Workshop on*, pp. 1–6, Sept 2007.

[29] S. J. Julier and J. K. Uhlmann, "New extension of the kalman filter to nonlinear systems," in *AeroSense'97*, pp. 182–193, International Society for Optics and Photonics, 1997.

[30] E. Wan and R. Van Der Merwe, "The unscented kalman filter for nonlinear estimation," in *Adaptive Systems for Signal Processing, Communications, and Control Symposium.*, pp. 153–158, 2000.

[31] S. Julier, "The scaled unscented transformation," in *American Control Conference, 2002. Proceedings of the 2002*, vol. 6, pp. 4555–4559 vol.6, 2002.

[32] R. Zhan and J. Wan, "Iterated unscented kalman filter for passive target tracking," *Aerospace and Electronic Systems, IEEE Transactions on*, vol. 43, pp. 1155–1163, July 2007.

[33] S. Srirangarajan, A. Tewfik, and Z.-Q. Luo, "Distributed sensor network localization using socp relaxation," *Wireless Communications, IEEE Transactions on*, vol. 7, pp. 4886–4895, December 2008.

[34] R. Mehra, "On the identification of variances and adaptive kalman filtering," *Automatic Control, IEEE Transactions on*, vol. 15, pp. 175–184, Apr 1970.

[35] G. Agamennoni and E. M. Nebot, "Robust estimation in non-linear state-space models with state-dependent noise," *Signal Processing, IEEE Transactions on*, vol. 62, no. 8, pp. 2165–2175, 2014.

[36] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Robotics and Automation. Proceedings.*, vol. 2, pp. 500–505, IEEE, Mar 1985.

[37] J. Ren, K. McIsaac, and R. Patel, "Modified newton's method applied to potential field-based navigation for mobile robots," *Robotics, IEEE Transactions on*, vol. 22, pp. 384–391, April 2006.

[38] E. Rimon and D. Koditschek, "Exact robot navigation using artificial potential functions," *Robotics and Automation, IEEE Transactions on*, vol. 8, pp. 501–518, Oct 1992.

[39] "Morse theory." http://www.math.harvard.edu/history/bott/bottbio/node9.html. Accessed: 9-1-2015.

[40] S. Ge and Y. Cui, "Dynamic motion planning for mobile robots using potential field method," *Autonomous Robots*, vol. 13, no. 3, pp. 207–222, 2002.

[41] S. B. Akat, V. Gazi, and L. Marques, "Asynchronous particle swarm optimization-based search with a multi-robot system: simulation and implementation on a real robotic system," *Turkish Journal of Electrical Engineering and Computer Sciences*, vol. 18, no. 5, pp. 749–764, 2010.

[42] D. Zarzhitsky, D. Spears, and W. Spears, "Distributed robotics approach to chemical plume tracing," in *Intelligent Robots and Systems. IROS. IEEE/RSJ International Conference on*, pp. 4034–4039, Aug 2005.

Glossary

List of Acronyms

3mE	Mechanical, Maritime and Materials Engineering faculty
AKF	Adaptive Kalman Filter
DCSC	Delft Center for Systems and Control
EKF	Extended Kalman Filter
GPS	Global Positioning system
IEEE	Institute of Electrical and Electronics Engineers
LED	Light emitting diode
LOS	Line-of-sight
m-PSO	modified Particle Swarm Optimization
MDS	Multi-dimensional scaling
MRRPL	Multi-robot relative pose localization
MRT	Multi-robot trilateration
NLOS	non-Line-of-sight
OG	Obstacle Goal value
PSO	Particle Swarm Optimization
RMS	Root mean squared
RPE	Relative Pose Estimation
RSS	Radio Signal Strength

RSSI Radio Signal Strength Indicator

TU Delft Delft University of Technology

UKF Unscented Kalman Filter

UKF-RPE Unscented Kalman filter with only the relative pose estimation

VPF Virtual Potential Field

Zebro "Zes-benige robot", six legged robot