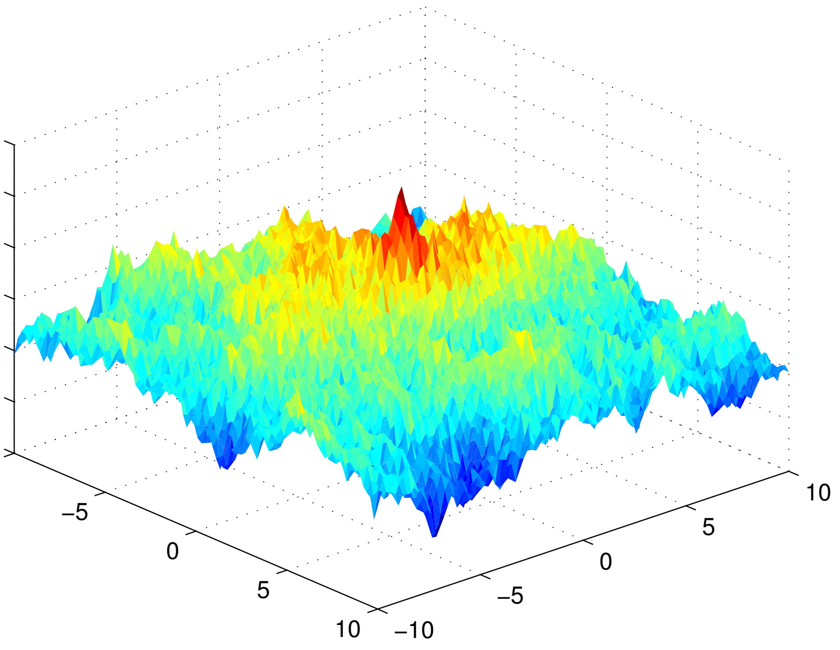
CHALMERS
UNIVERSITY OF TECHNOLOGY



Network optimisation in multi-robot systems

Implementation of mobile router positioning algorithms to minimise the end-to-end bit error rate in a multi-robot network

Master's thesis in Communication Engineering

Mikael Larsson, Georgios Tolikas

Department of Signals and Systems
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2015

Master's thesis 2015

Network optimisation in multi-robot systems

Implementation of mobile router positioning algorithms to minimise the end-to-end bit error rate in a multi-robot network

Mikael Larsson, Georgios Tolikas

Department of Signals and Systems
Chalmers University of Technology
Gothenburg, Sweden 2015

Network optimisation in multi-robot systems
Implementation of mobile router positioning algorithms to minimise the end-to-end bit error rate in a multi-robot network.
Georgios Tolikas, Mikael Larsson

© MIKAEL LARSSON, GEORGIOS TOLIKAS, 2015.

Supervisor: Markus Fröhle, PhD Student, Department of Signals and Systems
Examiner: Henk Wymeersch, Associate Professor, Department of Signals and Systems

Master's Thesis 2015
Department of Signals and Systems
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Channel visualisation constructed in Matlab showing the SNR representing the random variations of shadowing and multipath effects on top of pathloss. The plot was given for a fixed transmitter at location (0,0) and a traversing receiver.

Typeset in LaTeX
Gothenburg, Sweden 2015

Abstract

There are countless applications for teams of coordinated robots. Examples are construction, agriculture as well as search and rescue missions. The number of successful implementations of such systems is rapidly growing as well as the research put into coordinating such robots to achieve a common goal. However, the consideration on the impact of the radio link performance on the system is usually rather limited. Our thesis focuses on this problem using a previously proposed approach of measuring link performance with bit error rate and steering robots according to a maximisation problem and with the help of a channel prediction framework. The complexity of the maximisation problem and the channel predictor can be altered by including more or less of the stochastic radio channel phenomena. Compared to existing systems the approach we use considers more of the randomness of a real radio channel. We highlight the benefits of this and present simulation results of our implementation. Moreover, we extend and verify the previous work by carrying out experiments running these algorithms with physical robots.

Acknowledgements

Our master thesis work has been a journey influenced by numerous interesting, friendly and, to our work, valuable people. The department of Signals and Systems has given us the foundation from which to grow as engineers. Various researchers such as Gabriel Garcia have helped us to gather the resources needed for the completion of our work. Especially, we would like to thank our supervisor Markus for his valuable guidance, vivid interest and supply of resources. We would also like to give credit to our examiner Henk for his ability to set ambitious goals, yet combine his high expectations with support and understanding. Finally, we would like to thank each other for a fruitful collaboration which culminated in a good friendship.

Mikael Larsson, Georgios Tolikas, Gothenburg, July 2015

Contents

List of Figures

List of Tables

1 Introduction

1.1 Background

There are numerous applications that deal with cooperative robots to perform a required task [1, 2]. These tasks can achieve agriculture or construction enhancement or extend the potential of search and rescue missions. Furthermore, exploration of other planets is mainly based on employing robotic agents equipped with sensors and cameras. Most of the applications though, focus on achieving the task within the field of interest and neglect the importance of reliable communication between the task executing robots.

Consider the case of coordinated task performing mobile robots that need to exchange information while on the job. There may be a scenario according to which direct communication is lost or becomes power demanding. To elaborate, an appointed task may demand the dispersion of the robots throughout an area. Usually, the robots are armed with only low power transceivers which suffice for communication with only close neighbours. Therefore, the task collides with the connectivity maintenance prerequisite, rendering direct connection inadvisable in terms of energy saving or potentially even impossible.

This problem could then be successfully solved in an ad-hoc network of densely located robotic agents by rerouting the information via a chain of closest neighbours. A meshed network alternative would also allow the performance of the coordinated robots even in the case of one robot's potential failure. Working towards connectivity maintenance between two end-points a different chain of communicating robots could then be formed.

Another method that could be used on top of a meshed network of robots would be to employ cheaper intermediate robotic routers to form a multi-hop chain via which the expensive task performing robots could exchange vital information. The intermediate routers would then have to relocate themselves so as to optimise the link performance. In this case, one realises the importance of the radio channel theory. We focus on exploring the potentials of such an approach.

Research has been carried out for robotic router formation on a simulation level considering rough deterministic and more accurate stochastic models of a wireless channel[3]. This research however considers that only one of the two nodes on every link is able to move and does not mention the joint correlation of shadowing when both end-points of a link are able to move. Previous work has also catered for the mathematical formulation of the problems that need to be optimised according to different channel modelling.

1.2 Limitations

This thesis has some fundamental limitations in terms of the robotic router formation scheme. The subject of our interest lies at the optimisation of the total communication performance between two robots that are considered to be end-nodes in a fixed chain of communicating robots. This means that a potential failure of one intermediate router would render the communication between the two simple graphs formed impossible. Moreover, an addition of a single robotic router would demand reconfiguring the whole system.
The implemented algorithms do not guarantee that the system reaches a global maximum. Rather they cause each robot to maximise a local objective function that assumes static neighbours. For the simple channel including only pathloss it is proven that this approach does lead the system to a global maximum[3]. The same is, however, generally not true in the case of a realistic channel including random variations.

1.3 Purpose

The purpose of this thesis is to investigate, propose and implement algorithms for robotic teams to handle the radio channel variations optimally. The objective of such algorithms is to, given some constraints, position the involved robots such that the highest possible end to end throughput is achievable. The constraints may for example describe a cooperative task that the robots need to perform, avoiding obstacles while on the move or remaining static. Proposed algorithms are to be evaluated through simulations and real world experiments.

1.4 Related Work

The area of wireless router, or relay, positioning has gained considerable interest in recent years. Therefore, there are numerous publications on the subject. Most of them model the wireless channel in a simpler manner than does [3], the work this thesis is based on. However the overall objectives or the models of the relays themselves are often more refined than in this thesis. For example, in [4] the end-to-end BER of a system with one relay running the decode and forward protocol is minimised. The channel is modelled as depending only on pathloss.
In [5] a more general network topology, such as a sensor network, is considered. A method to maximise the so called Fiedler value of the network graph is presented. This work, however, relies on an even simpler disk model of the channel. With this model two nodes are either connected or not connected without considering different link qualities.

2

Theoretical Foundations and Prior Work

2.1 Theory

Employing the wireless channel to communicate information among a chain of robots, a description of various properties and challenges pertaining to it will be examined. The wireless channel theory involves factors such as:

1. Noise which can be found in any wired or wireless system, but also interference which stems from the fact that the wireless channel is a shared medium halting communication.
2. The channel constitutes a time varying medium, implying that its impediments can also vary, entailing special demands on design.

Modelling the channel to extract some knowledge towards predicting the optimum established link requires some insight on complex propagation environments. Narrating in a simplified manner, we will start from the the simplest propagation environment and gradually we will advance to more complex ones. To begin with, it is well known that in empty space Friis' law gives the received power of a wave as a function of the distance to the transmitting object, involving the free space loss factor. The relation between them is given explicitly by the equation:

$$P_{\mathrm{RX}} = P_{\mathrm{TX}} G_{\mathrm{TX}} G_{\mathrm{RX}} \left(\frac{\lambda}{4\pi d} \right)^2 , \tag{2.1}$$

where G_{TX} and G_{RX} stand for the gains of the antennas of the transmitter and the receiver respectively, λ denotes the wavelength and d the distance between transmitter and receiver.

In reality the propagation environment is not vacuum and Friis' equation must be incorporated to a more appropriate deterministic model, namely the pathloss. There are several different pathloss models that are well described in [6]. Some models are based on measurements taken. These models are called empirical and can be approximated by averaging the local power measurements to remove any spatial local variations that will be described later. Different models are used depending not only on the environment, but also the on frequencies employed by different systems.

For the case of our environment we will describe and use a simplified pathloss model, which captures the essence of signal propagation. This implies that more complex models could give better accuracy and would be more appropriate for different robot operation environments. Weighing the trade-off between complexity and accuracy though and keeping in mind that even a more advanced pathloss model would still be only an approximation of the real channel we shall examine only simplified models. In literature one simple and popular model for pathloss is[6]:

$$P_{\mathrm{r}} = P_{\mathrm{t}} K \left(\frac{d_o}{d}\right)^{\gamma}, \tag{2.2}$$

where P_r and P_t is the received and transmitted power respectively, K is a unit-less constant that depends on the antenna characteristics and the average channel attenuation, d is the distance between transmitter and receiver and d_o is a reference distance for the antenna far field. The factor γ is the pathloss exponent. Equivalently in dB we have:

$$P_{\mathrm{r}}[\mathrm{dBm}] = P_{\mathrm{t}}[\mathrm{dBm}] + K[\mathrm{dB}] - 10\gamma \log_{10}\left(\frac{d}{d_0}\right). \tag{2.3}$$

where $K = 20\log_{10}\left(\frac{\lambda}{4\pi d_o}\right)$. The value of γ in complex environments can be approximated by a minimum mean square error (MMSE) fit to empirical measurements. Another alternative is an empirical model based on frequency and antenna height. For more information the reader can refer to [6]. An illustration of a channel, simulated according to a pathloss model is shown in Fig. 2.1

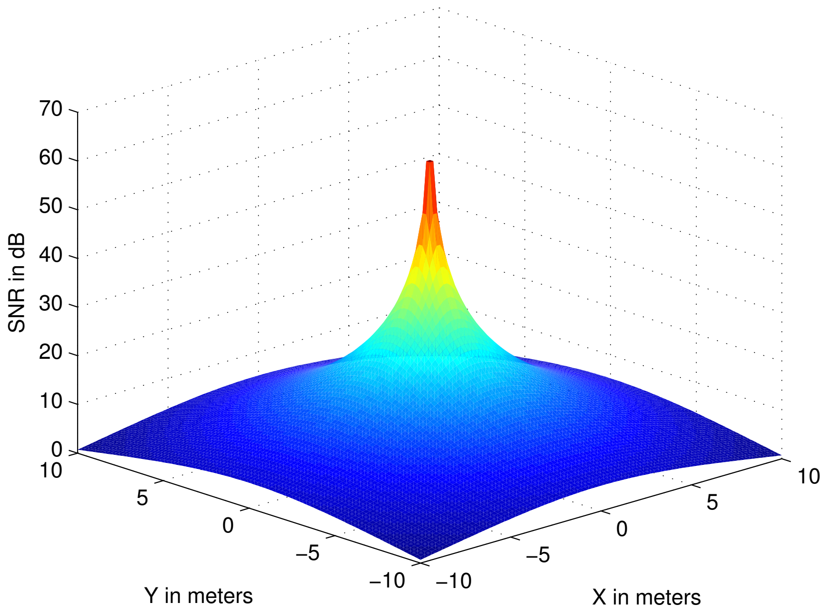


Figure 2.1: SNR received over a simulated channel according to the pathloss model for a fixed transmitter at (0,0) and a traversing receiver.

In real environments there may be obstacles that reflect, scatter, diffract or let the wave be transmitted via them depending on the location, size, surface and the dielectric properties of the material they are made of. Thus, random variations to the received signal power can be observed and communication can still be possible even in non line of sight (NLOS) scenarios. Since the characteristics of the obstacles are generally unknown, one must resort to statistical models. The most common model to describe the attenuation in received signal power is log-normal shadowing. According to this model, the ratio between transmitted to received power $\psi = \frac{P_\mathrm{t}}{P_\mathrm{r}}$ is assumed random with a probability density function:

$$p(\psi) = \frac{\xi}{\sqrt{2\pi}\sigma_{\psi_\mathrm{dB}}\psi} \exp\left(-\frac{(10\log_{10}\psi - \mu_{\psi_\mathrm{dB}})^2}{2\sigma^2_{\psi_\mathrm{dB}}}\right), \psi > 0, \tag{2.4}$$

where $\xi = 10/\ln 10, \mu_{\psi_\mathrm{dB}}$ is the mean of $\psi_\mathrm{dB} = 10\log_{10}\psi$ in decibels and $\sigma_{\psi_\mathrm{dB}}$ is the standard deviation of ψ_dB. The mean μ_{ψ_dB} can be extracted either by analytical models or empirical measurements. For the case of empirical measurements, the average attenuation due to shadowing is already taken into account. Therefore, the mean of shadowing is equal to the empirical pathloss. With some math that can be found in [6], the distribution of the dB value of transmitted to received power ψ_dB shows its Gaussian nature with mean μ_{ψ_dB} and standard deviation $\sigma_{\psi_\mathrm{dB}}$:

$$p(\psi_\mathrm{dB}) = \frac{1}{\sqrt{2\pi}\sigma_{\psi_\mathrm{dB}}} \exp\left(-\frac{(\psi_\mathrm{dB} - \mu_{\psi_\mathrm{dB}})^2}{2\sigma^2_{\psi_\mathrm{dB}}}\right). \tag{2.5}$$

To capture power fall off with distance along with the random attenuation about the pathloss caused by shadowing as in (2.5), one can combine it with (2.2) and get more accurate approximations of the channel. In this model, average pathloss in dB is the one given by the pathloss model (μ_{ψ_dB}) whereas shadowing, with a mean of 0dB, gives variations about the given pathloss. The above can be summarised in the following equation:

$$P_\mathrm{r}[\mathrm{dBm}] = P_\mathrm{t}[\mathrm{dBm}] + K[\mathrm{dB}] - 10\gamma\log_{10}\left(\frac{d}{d_0}\right) + \psi_\mathrm{dB}, \tag{2.6}$$

where ψ_dB is a Gaussian random variable with mean 0 and variance $\sigma^2_{\psi_\mathrm{dB}}$.
It is a mandate at this point to discuss another phenomenon, known as small scale fading or multipath effect, that depends on the different paths of a wave after reflections or scattering in the environment. To elaborate, a wave can split into different waves that reach the receiver at different time instances following different paths. Reaching the receiver at different times means that the received power and subsequently signal to noise ratio can largely deviate from the expected values according to the combined pathloss and shadowing model. To cater for this effect, one can model it as a random variable which adds up on top of pathloss and shadowing.
What is more the channel itself may be time varying due to mobility of obstacles, transmitter or receiver. In those cases, the envelope can have a Rayleigh or Rician fading in NLOS and line of sight (LOS) scenarios respectively. It also induces great complexity in modelling the channel. For a number of reasons that will later be discussed, we will not dig into that any further and we shall consider all spatial variations but no time variations at all.

To recapitulate, one must bear in mind the basic principle for received to transmitted signal power ratio which can be expressed in the following abstract equation:

$$P_\mathrm{r}[\mathrm{dBm}] = P_\mathrm{t}[\mathrm{dBm}] + \mathrm{pathloss}\ [\mathrm{dB}] + \mathrm{shadowing}\ [\mathrm{dB}] + \mathrm{multipath}\ [\mathrm{dB}] \tag{2.7}$$

In this section we will discuss some performance criteria of digital modulation over additive white Gaussian noise (AWGN) and fading channels.

2.1.1 Link Performance Metric

In an AWGN channel, the modulated information bearing signal $s(t)$ will have noise $n(t)$ added prior to its reception. The noise $n(t)$ is a complex white Gaussian random process with mean zero and power spectral density (PSD) $N_0/2$. The received signal is then given by the following fundamental equation:

$$r(t) = s(t) + n(t) \tag{2.8}$$

In an effort to optimise the communication performance of our robotic router formation, one could use the end-to-end bit error rate (BER). BER characterises the probability of a bit arriving in error at the receiver. To exemplify, consider two communicating nodes. Let b and b_r represent a transmitted and a received bit as a part of a communicated bit sequence in a packet. Then, BER is defined as $P_\mathrm{b} = \mathrm{Prob}\{b \neq b_\mathrm{r}\}$.
Another fundamental performance metric is the well-known signal to noise ratio (SNR). Making the assumption that the noise power is mainly due to the thermal noise at the receiver, the received SNR is defined as the ratio of the received signal power P_r to the receiver thermal noise power[3].

$$\mathrm{SNR} = \frac{P_\mathrm{r}}{N_0} \tag{2.9}$$

The instantaneous received SNR directly affects the BER and consequently the reception quality. BER relates not only the received SNR, but also modulation, channel coding and other transmission parameters with the reception quality. In an upper bound, BER of an MQAM (M-Ary Quadrature Amplitude Modulation) transmission is given by:

$$P_\mathrm{b} \leq 0.2\mathrm{exp}\left(-\frac{1.5\gamma}{M-1}\right), \tag{2.10}$$

where M is the modulation constellation size and γ is the received SNR[6, 3]. This approximation is tight for $M \geq 4$ and 0 dB $< \gamma <$ 30 dB. In the following we shall use this approximation to characterise the BER.
As already explained, in an AWGN environment the probability of error depends on the received SNR or equivalently γ. The difference between an AWGN and a fading environment like ours lies in the fact that the received signal power varies randomly over distance as a result of shadowing and multipath. In general, there are three time scales related to the spatio-temporal changes of the channel quality and, hence, the received SNR. The slowest dynamic is due to pathloss, a faster one

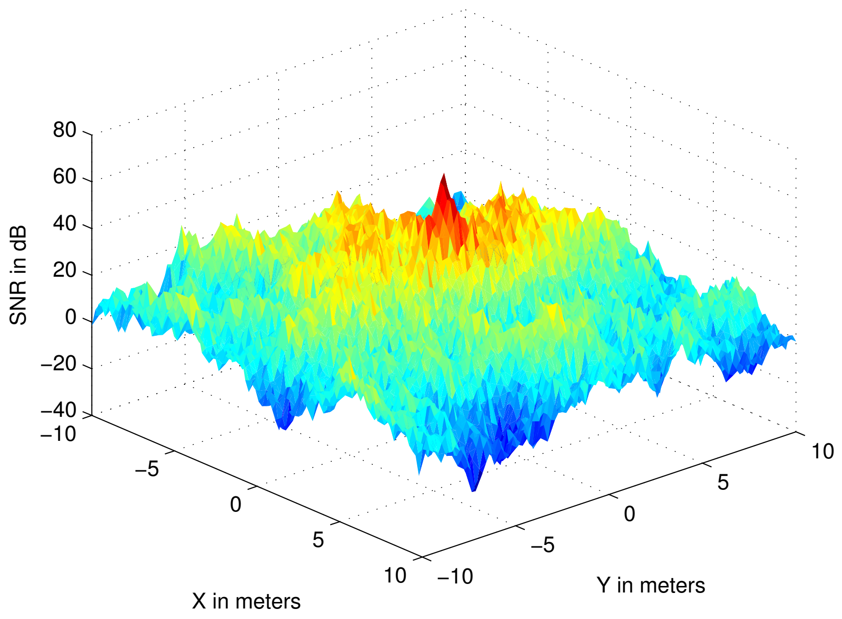


Figure 2.2: SNR received over a simulated realistic channel according to the combined model (2.7) captured for a fixed transmitter at the maximum (0,0) and a traversing receiver.

is due to shadowing and the fastest is caused by the multi-path fading. In Fig. 2.2 can be seen that motion planning only based on a pathloss model can result in severe performance degradation due to the channel variations caused by shadowing and multipath. Therefore, the motion planning of our robot routers will be designed for realistic environments described by (2.7).

2.2 Prior Work

Yuan Yan and Yasamin Mostofi in [3] take into consideration all previously mentioned spatial factors that may affect the channel quality. There, they examine the problem of robotic router formation where two nodes need to maintain their connectivity over a large area using intermediate mobile routers. They start with the simple case of modelling the channel with only deterministic pathloss and gradually build up their simulations by integrating all spatial variations into their system together with a proposed stochastic channel learning framework. Since we are to use their approach for both our simulations and actual experiments, it is a mandate to present their work in greater detail.

2.2.1 Robotic Router Optimisation Considering only Pathloss

In this section the simplest environment is considered, where fading does not exist. Therefore, the channel is assumed to be known since pathloss is deterministic. Without loss of generality, the robots in the network are labelled as follows: Transmitter is noted as node 1, the node that directly receives the information from transmitter is labelled as 2 and so on. The communication is bidirectional, meaning that node 1 also receives from 2, 2 receives from node 3 and likewise until node m-1 receives from node m. In Fig. 2.3 the possible communication steps in the robot chain are illustrated.

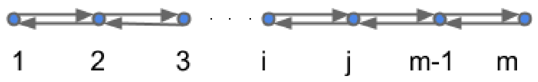


Figure 2.3: Chain of communicating robots

2.2.1.1 Objective Function

Using BER as a metric of performance according to section 2.1.1 Yan and Mostofi propose the following model for received SNR in the transmission from ith node to the jth one:

$$\gamma_{i,j} = \frac{\alpha_{i,j}}{d_{i,j}^n}, \tag{2.11}$$

where $d_{i,j}$ is the distance between robots i and j, and $\alpha_{i,j}$ is a function of system parameters including transmit power, antenna gain and frequency employed. The pathloss exponent is denoted n. The probability of correct reception at final destination node m in the communicating robot chain is given by:

$$P_c(\text{RX}) = P_c(\{m\}|\{m-1\})P_c(\{m-1\}), \tag{2.12}$$

where $P_c\{m-1\}$ stands for the probability of correct bit reception at node $m-1$ and
$P_c(\{m\}|\{m-1\})$ stands for the conditional probability of correct reception at node m given that the reception was correct at node $m-1$. Likewise, the probability of correct reception at node $m-1$ is given by:

$$P_c(\{m-1\}) = P_c(\{m-1\}|\{m-2\})P_c(\{m-2\}) \tag{2.13}$$

Solving recursively the probability of correct reception at receiver m is:

$$P_c(\text{RX}) = \prod_{i=2}^{m} P_c(\{i\}|\{i-1\}). \tag{2.14}$$

Note that the previous equation constitutes a lower bound on the probability of correct reception since it does not count as correctly received the bits that may be flipped an even number of times.
The probability of correct reception can be expressed as a function of BER as follows:

$$P_c(\text{RX}) = 1 - P_b(\text{RX}) \tag{2.15}$$

Combining equations (2.15) and (2.10) gives the objective function that the robot routers need to maximise relocating themselves for better system performance:

$$P_{\mathrm{c}}(\mathrm{RX}) = \prod_{i=2}^{m}(1 - 0.2\exp(-c\gamma_{i-1,i})), \tag{2.16}$$

where $c = \frac{1.5}{M-1}$. Note that equation (2.16) is proven as continuous and differentiable in its entire domain [3].

2.2.1.2 Positioning the Robotic Routers to Optimise Performance

Throughout this work it is considered that the robots operate in a 2-D workspace. Denoting the workspace of robots as $W \subset \mathbb{R}^2$ and the position of robot i as $\mathbf{x}_i \in \mathbb{R}^2$ the set of robots in the system is then $\mathbf{x} = [\mathbf{x}_1^T \mathbf{x}_2^T ... \mathbf{x}_m^T]^T$ and the set of routers $\mathbf{x}_{\mathrm{r}} = [\mathbf{x}_2^T ... \mathbf{x}_{m-1}^T]^T$. Then the routers need to move so as to satisfy the following maximisation problem:

$$\begin{aligned} &\text{maximise } J(\mathbf{x}_{\mathrm{r}}) = \sum_{i=2}^{m} \ln(1 - 0.2\exp\left(-c\gamma_{i-1,i}\right)) \\ &\text{subject to } \mathbf{x}_i \in W, \forall i \in \{2, ..., m-1\} \end{aligned} \tag{2.17}$$

where $\gamma_{i-1,i} = \frac{a_{i-1,i}}{d_{i-1,i}^n}$ and $d_{i-1,i} = \|x_{i-1} - x_i\|$.
It is proven that with the premise that $n+1 \leq \min_i \left(\frac{nc\gamma_{i-1,i}}{1-0.2\exp\left(-c\gamma_{i-1,i}\right)}\right)$, 2.17 is concave for convex W. A stronger condition for this is to ensure that $\min_i(\gamma_{i-1,i}) \geq \frac{n+1}{nc} = \frac{(M-1)(n+1)}{1.5n}$, meaning that all robots need to maintain a minimum received SNR.
Denoting the move of a robotic router $i \in \{2, .., m-1\}$ as $u_i = \dot{x_i}$ [3] suggests the following control law, which for each router depends only on the two nodes that directly communicate with it (specifically the previous and next robot in the communication chain as illustrated in Fig. 2.3):

$$\begin{aligned} \mathbf{u}_i &= \kappa \ \nabla_{\mathbf{x}_i} J(\mathbf{x}_r) \\ &= \kappa \left(\frac{\partial J_i(\mathbf{x}_r)}{\partial d_{i-1,i}} \frac{\mathbf{x}_i - \mathbf{x}_{i-1}}{d_{i-1,i}} + \frac{\partial J_{i+1}(\mathbf{x}_r)}{\partial d_{i,i+1}} \frac{\mathbf{x}_i - \mathbf{x}_{i+1}}{d_{i,i+1}} \right), \end{aligned} \tag{2.18}$$

where κ is a positive constant and $J_i = \ln(1 - 0.2\mathrm{exp}\{-c\gamma_{i-1,i}\})$.
It is proven that for an environment without obstacles, the optimum positions of the routers lie on the straight line segment between the end-points 1 and m. So a maximisation problem equivalent to (2.17), but reduced to this workspace, is presented in an effort to explain and prove that the routers should be equally spaced along the line between transmitter 1 and receiver m when the transceivers are homogeneous and all links have the same underlying pathloss parameters.
Moreover, [3] proves that with transmission parameters in different robots such as $\alpha_{i-1,i} > \alpha_{j-1,j}$, it should hold that $d_{i-1,i} > d_{j-1,j}$ meaning that if a link experiences lower α (e.g due to lower transmit power) then the associated link nodes should get closer to each other to optimise performance.

2.2.1.3 Obstacle Avoidance in Motion Planning

Having presented a control law towards moving the robot routers so as to optimise communication performance, [3] incorporates obstacle avoidance in environments for which we have prior knowledge on obstacles. Here the robots are assumed to operate in a walled environment, which allows to model the obstacle avoidance as linear constraints. Denoting the outward normal vector of one side of a walled obstacle as $N_\perp$, the ith router can avoid collision with that side of the obstacle when:

$$\langle N_\perp, \dot{\mathbf{x}_i} \rangle = \langle N_\perp, \mathbf{u}_i \rangle \geq 0 \tag{2.19}$$

This can be easily understood if we keep in mind that negative dot product means that the angle between the vector of movement and the outward vector would be greater than 90 degrees.
Taking into consideration the above constraint and letting $\nabla_{\mathbf{x_r}}\mathbf{J}$ denote the vector $\left(\frac{\partial J_2(\mathbf{x}_r)}{\partial \mathbf{x}_2}, \frac{\partial J_3(\mathbf{x}_r)}{\partial \mathbf{x}_3}, \ldots, \frac{\partial J_i(\mathbf{x}_r)}{\partial \mathbf{x}_i}, \frac{\partial J_{i+1}(\mathbf{x}_r)}{\partial \mathbf{x}_{i+1}}, \ldots, \frac{\partial J_{m-2}(\mathbf{x}_r)}{\partial \mathbf{x}_{m-2}}, \frac{\partial J_{m-1}(\mathbf{x}_r)}{\partial \mathbf{x}_{m-1}}\right)$ and $\mathbf{u}$ the vector $(\dot{\mathbf{x}_2}, \dot{\mathbf{x}_3}, \ldots, \dot{\mathbf{x}_i}, \dot{\mathbf{x}_{i+1}}, \ldots, \dot{\mathbf{x}_{m-2}}, \dot{\mathbf{x}_{m-1}})$ the maximisation problem becomes:

$$\text{maximise } \langle \nabla_{\mathbf{x}_r} J, \mathbf{u} \rangle \text{ subject to } W\mathbf{u} \geq 0, \tag{2.20}$$

with $\langle \nabla_{\mathbf{x}_r} J, \mathbf{u} \rangle$ denoting the sum of the projections of the position changes on their corresponding gradients of the objective function and with W denoting the set of all constraints for close obstacles stemming from each obstacle side such as in (2.19). The maximisation of (2.20) means that for each of the intermediate routers which belong to the x_r one must first compute the movement vector $\mathbf{u}_i$ according to (2.18). Then some means of assuring the constraint in (2.20) must be imposed. One approach is presented in section 3.1.5. Infinitely many obstacles could be modelled, but the complexity of calculations rises. That is because according to the previous analysis, the routers at each step need to first compute $\nabla_{\mathbf{x}_r} J$ and W and then solve (2.20) for $\mathbf{u}$. The framework given in (2.20) drives the routers towards to the optimum position with the premise that the router will not collide with an obstacle.

2.2.2 Operating in Fading Environments

The paper [3] continues with presenting a widely accepted method of probabilistic modelling of a wireless channel in complex environments. This method takes into consideration spatial variations to predict the channel in a 2-D space accounting for pathloss, shadowing and multipath components. Then, it proposes a general framework for motion planning and develops some sub-optimal special cases.

2.2.2.1 Probabilistic Modelling of Fading Channels

In [3], the SNR from node i to node j (received at node j in dB) is denoted as $\gamma_{\mathrm{dB},i,j} = \gamma_{\mathrm{dB}}(\mathbf{x}_i, \mathbf{x}_j)$. We have already shown in (2.3) that the distance dependent pathloss has a linear decay in the dB domain. Keeping also in mind that (2.7) holds for fading environments it becomes easy to understand the following model that [3] proposes:

$$\gamma_{\mathrm{dB},i,j} = \gamma_{\mathrm{dB}}(\mathbf{x}_i, \mathbf{x}_j) = \alpha_{\mathrm{dB},i,j} - 10n\log_{10}(\| \mathbf{x}_i - \mathbf{x}_j \|) + \gamma_{\mathrm{SH}}(\mathbf{x}_i, \mathbf{x}_j) + \gamma_{\mathrm{MP}}(\mathbf{x}_i, \mathbf{x}_j), \tag{2.21}$$

where $\gamma_{\mathrm{dB}}(\mathbf{x}_i, x_j) = 10\log_{10}(\gamma(x_i, x_j))$. $\gamma_{\mathrm{SH}}(x_i, x_j)$ and $\gamma_{\mathrm{MP}}(x_i, x_j)$ are independent random variables representing the shadowing and multipath effects respectively. The effect of shadowing, as can be seen by (2.6) for the combined model, can best be described by a zero-mean Gaussian random variable with an exponential spatial correlation. For multipath effect [3] suggests a simplification of lognormal modelling so as to ease the mathematical analysis. Multipath is taken to be spatially uncorrelated, since it is well known to decorrelate very fast.
The authors use the aforementioned channel model (2.21) between two routers to deal with a channel prediction scheme based on a number of measurements. The prediction of the channel is later used in the control law of each robot.

2.2.2.2 Stochastic Channel Learning Framework Based on a Number of SNR Measurements

For ease of mathematical analysis, [3] presents the prediction of the channel in a link between a fixed node and a moving one at unvisited locations although everything is valid even when both links move.
Denoting the position of the fixed node as $\mathbf{x}_{\mathrm{b}}$, the reception quality at position $\mathbf{x}$ for the moving node is written according to (2.21) as:

$$\gamma_{\mathrm{dB}}(x) = \alpha_{\mathrm{dB}} - 10n\log_{10}(\| \mathbf{x} - \mathbf{x}_{\mathrm{b}} \|) + \gamma_{\mathrm{SH}}(\mathbf{x}) + \gamma_{\mathrm{MP}}(\mathbf{x}). \tag{2.22}$$

The positions for which measurements exist are denoted as $\mathbf{Q} = \{\mathbf{q}_1, ..., \mathbf{q}_k\}$ for $k = |\mathbf{Q}|$. The number of measurements corresponding to the positions $\mathbf{Q}$ is given in vector form by:

$$\mathbf{y} = \mathbf{G}_q\theta + \omega_{SH} + \omega_{MP}, \tag{2.23}$$

with $\mathbf{G}_q = [\mathbf{1}_k - \mathbf{F}_q], \mathbf{F}_q = [10\log_{10}(\| \mathbf{q}_1 - \mathbf{x}_b \|)...10\log_{10}(\| \mathbf{q}_k - \mathbf{x}_b \|)]^T, \theta = [\alpha_{\mathrm{dB}}\ n]^T$, $\omega_{\mathrm{SH}} = [\gamma_{\mathrm{SH}}(\mathbf{q}_1)...\gamma_{\mathrm{SH}}(\mathbf{q}_k)]^T$ and $\omega_{\mathrm{MP}} = [\gamma_{\mathrm{MP}}(\mathbf{q}_1)...\gamma_{\mathrm{MP}}(\mathbf{q}_k)]^T$. The zero mean Gaussian random vector ω_{SH} is characterized by a covariance matrix with entries $\Omega_{i,j} = \xi^2_{\mathrm{dB}}\exp\{-\frac{\|\mathbf{q}_i - \mathbf{q}_j\|}{\eta}\}$ for $i, j \in \{1, ..., k\}$, based on the lognormal distribution and the Gudmundson model for shadowing. Here, ξ^2_{dB} stands for the variance of shadowing and η the decorrelation distance. Similarly, the assumed zero mean Gaussian random vector for multipath ω_{MP} has a covariance matrix $\rho^2_{\mathrm{dB}}\mathbf{I}_k$ (since multipath is considered uncorrelated with distance). Here, ρ^2_{dB} stands for the power of multipath fading (in dB).

Estimating the Channel Parameters Towards a Prediction Framework:

In order to estimate the parameters of the channel the paper suggests a least square error approach based on the measurements according to which we have the following mathematical formulae:

$$\hat{\theta} = (\mathbf{G}_q^T\mathbf{G}_q)^{-1}\mathbf{G}_q^T\mathbf{y} \tag{2.24}$$

$$\hat{\xi}^2_{\mathrm{dB}}, \hat{\eta} = \min_{\xi^2_{\mathrm{dB}}, \eta} \sum_{l\in L(l)} \zeta(l)[\xi^2_{\mathrm{dB}}\exp(-l/\eta) - \hat{A}(l)]^2 \tag{2.25}$$

$$\hat{\rho}^2_{\mathrm{dB}} = \frac{1}{k}\mathbf{Y}^T_{\mathbf{G}_q}\mathbf{Y}_{\mathbf{G}_q} - \hat{\xi}^2_{\mathrm{dB}} \tag{2.26}$$

with $\mathbf{Y}_{\mathbf{G}_q} = (\mathbf{I}_k - \mathbf{G}_q(\mathbf{G}_q^T\mathbf{G}_q)^{-1}\mathbf{G}_q^T)\mathbf{y}$ standing for the centred version of the measurement vector which is almost free of pathloss and $\hat{A}(l) = \sum_{(i,j)\in\mathbf{S}(l)}[\mathbf{Y}_{\mathbf{G}_q}]_i[\mathbf{Y}_{\mathbf{G}_q}]_j/|\mathbf{S}(l)|$ is a numerical approximation of the spatial correlation at distance l, where $\mathbf{S}(l) = \{(i,j)|\mathbf{q}_i, \mathbf{q}_j \in \mathbf{Q}, \|\mathbf{q}_i - \mathbf{q}_j\| = l\}$. In (2.25), $\zeta(l)$ is a weight that relates to the assessment of the accuracy of the approximation of $\hat{A}(l)$ and $L(l) = \{l|0 < \hat{A}(l) < \hat{\xi}^2_{dB} + \hat{\rho}^2_{dB}\}$.

2.2.2.3 Optimisation Problem Based on Prediction Framework

The complete channel prediction, which [3] names as pathloss/Shadowing estimator, is then used to formulate the optimisation problem. The authors also suggest some sub-optimum cases when it comes to predicting the channel. Among them are the probabilistic pathloss estimator and a deterministic pathloss estimator similar to the one presented in (2.17), but with a more realistic approach of estimating the channel parameters according to (2.24). In the following we present the most complete framework of all.

Prediction Framework

It is proven that an estimation which accounts for pathloss and shadowing can be given by a Gaussian distribution. This framework can assess the channel at an unvisited location $\mathbf{x} \in W|Q$ with mean:

$$Y_{\text{dB,PL/SH}}(\mathbf{x}) = \underbrace{\mathbf{G}_x\hat{\theta}}_{\text{estimated pathloss}} + \underbrace{\Psi_x^T\Phi^{-1}(\mathbf{y} - \mathbf{G}_q\hat{\theta})}_{\text{estimated shadowing}} \tag{2.27}$$

and a variance:

$$\sigma^2_{\text{dB,PL/SH}}(\mathbf{x}) = \underbrace{\hat{\xi}^2_{\text{dB}} + \hat{\rho}^2_{\text{dB}}}_{\text{prediction var assuming only pathloss}} - \underbrace{\Psi_x^T\Phi^{-1}\Psi_x}_{\text{reduction in var when estimating shadowing}} \tag{2.28}$$

where $\mathbf{G}_x = [1 \ \ -10\log_{10}(\|\mathbf{x} - \mathbf{x}_b\|)]$, $\Phi = \hat{\Omega} + \hat{\rho}^2_{\text{dB}}\mathbf{I}_k$, $[\hat{\Omega}]_{i,j} = \hat{\xi}^2_{\text{dB}}\exp\{-\frac{\|\mathbf{q}_i - \mathbf{q}_j\|}{\eta}\}$ and $\Psi_x = [\hat{\xi}^2_{\text{dB}}\exp\{-\frac{\|\mathbf{x}-\mathbf{q}_1\|}{\eta}\}...\hat{\xi}^2_{\text{dB}}\exp\{-\frac{\|\mathbf{x}-\mathbf{q}_k\|}{\eta}\}]^{\text{T}}$.

Optimisation Problem

As already described, for a fading environment the SNR is modelled with a random variable which results in a stochastic probability of correct reception. Forming the maximisation problem for a stochastic P_c, the common practice is to maximise the average of it. Based on the previously assumed lognormal distribution for fading, they approximate the log-normal distribution with a gamma distribution having the same first and second moments as the lognormal one. Then [3] suggests the following maximisation problem:

$$\begin{aligned} &\text{maximise } \sum_{i=2}^{m} \ln(1 - 0.2(1 + \theta_{i-1,i}10^{\frac{Y_{\text{dB}_{i-1,i}}}{10}})^{-\phi_{i-1,i}}) \\ &\text{subject to } \mathbf{x}_i \in W, \forall i \in \{2, ..., m-1\}, \end{aligned} \tag{2.29}$$

where $\phi_{i,j} = (\exp\{(\alpha\sigma_{\mathrm{dB}_{i,j}})^2\} - 1)^{-1}, \theta_{i,j} = \exp\{1.5(\alpha\sigma_{\mathrm{dB}_{i,j}})^2\} - \exp\{0.5(\alpha\sigma_{\mathrm{dB}_{i,j}})^2\}$. $Y_{\mathrm{dB}_{i-1,i}}$ is the $Y_{\mathrm{dB,PL/SH}}$ of (2.27) among nodes i-1 and i and $\sigma_{\mathrm{dB}_{i-1,i}}$ is the variance $\sigma_{\mathrm{dB,PL/SH}}$ given in (2.28) for the channel between node i-1 and i. Note that the control law proposed in the pathloss only case is not sufficient to move the robots according to the prediction framework, since the complete problem above has many local maxima in conjunction with the global maxima. This problem can be tackled by directly applying (2.29) in a sufficiently large search window around each router's current position. The larger the search window, the more probable that the optimum solution is reached.

2.2.2.4 A Look Into Joint Shadowing Correlation Between two Mobile Nodes

As already mentioned the previous work examines the case of moving routers when only one of the two subsequent nodes in a communicating chain moves and the other remains static. Consequently, the spatial correlation of shadowing is the one given by covariance matrix Ω in Sec. 2.2.2.2. The authors of [7] propose a channel simulator modelling the shadowing effect and show its performance with the appropriate joint spatial correlation function between two moving end-points.

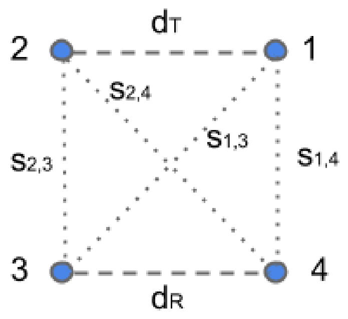


Figure 2.4: Transceivers in a hypothetical ad-hoc network. Transmitter at node 1 moves distance d_{T} to node 2 and receiver at node 3 moves distance d_{R} at node 4. We define as joint correlation function (JCF) the correlation between $s_{1,3}$ and $s_{2,4}$

[7] assumes the ad-hoc network of Fig. 2.4, where each transmitting node i, $i \in \{1, 2\}$, has a position $\mathbf{p}_{\mathrm{TX}} = (x_i, y_i)$ and each receiving node j,$j \in \{3, 4\}$ a position $\mathbf{p}_{\mathrm{RX}} = (u_j, v_j)$. Moreover, all transceivers are assumed to have the same configuration and employ the same channel. The shadowing in a channel between nodes i and j is denoted as $s_{i,j}$. As previously mentioned the shadowing is the variance seen in the expected local pathloss and is modelled having a log-normal distribution with zero mean and standard deviation σ_s. Then, the shadowing experienced in a

link between i and j should be the same when swapping transmitter and receiver locations. In other words it should hold that $s_{i,j} = s_{j,i}$, i.e the channel is reciprocal. The shadowing variance is then $\sigma_s^2 = \mathbf{E}[s_{i,j}^2]_{i,j} = \mathbf{E}[s_{j,i}^2]_{j,i}$, simply the mean of the square of shadowing effect between all possible locations for node i and node j. Assuming isotropic shadowing correlation i.e on average the same to all directions, [7] examines the case where both transmitter and receiver move and labels the functions giving the correlation between two shadowing instances. To elaborate consider that $s_{i,j}$ is the shadowing between two nodes i and j, and $s_{i',j'}$ is the shadowing experienced between the same nodes after their relocation to $\mathbf{p}_{\mathrm{TX}_{\mathrm{i}'}}$ and $\mathbf{p}_{\mathrm{RX}_{\mathrm{j}'}}$ respectively. Then the correlation between $s_{i,j}$ and $s_{i',j'}$ is given by the so called Joint shadowing spatial correlation function (JCF).

$$R_s(d_T, d_R) = E(s_{i,j} s_{i',j'})/\sigma_s^2, \tag{2.30}$$

where $d_T = |\mathbf{p}_{\mathrm{TX}_{\mathrm{i}}} - \mathbf{p}_{\mathrm{TX}_{\mathrm{i}'}}|$ is the transmitter's position shift and $d_R = |\mathbf{p}_{\mathrm{RX}_{\mathrm{j}}} - \mathbf{p}_{\mathrm{RX}_{\mathrm{j}'}}|$ is the receiver's position shift. Due to assumed isotropic shadowing spatial correlation, the JCF is a function that depends only on the position shifts as shown in (2.30).
The simple case where only one node moves (e.g. the receiver) is also examined. In that case the spatial shadowing correlation is named spatial auto-correlation function (ACF). In Fig. 2.4 consider that a fixed transmitter is located at position 1 and a moving receiver at position 3 which relocates to 4. Then the correlation between $s_{1,3}$ and $s_{1,4}$ is called ACF. Obviously, ACF is the same as JCF, when one of the two nodes either transmitter or receiver has a zero spatial shift.
Continuing with the assumption of fixed transmitter and a non-zero receiver shift of d_R, research has shown that ACF is generally given by:

$$R_s\{0, d_R\} = \exp\left(-\frac{|d_R|}{d_{cor}}\ln(2)\right), \tag{2.31}$$

with d_{cor} denoting the decorrelation distance, which is the distance where the correlation drops to 0.5.
Furthermore, [7] assumes that for two moving end-points in a link, the shadowing fluctuation pertains only to local scatterers around the nodes and that the movements of each node are uncorrelated. So, the JCF is given by:

$$R_s\{d_T, d_R\} = R_s\{d_T, 0\} R_s\{0, d_R\} = \exp\left(-\frac{|d_\mathrm{T}| + |d_\mathrm{R}|}{d_\mathrm{cor}}\ln(2)\right) \tag{2.32}$$

For the sake of completeness we should mention that in case we are talking about the correlation between the shadowing effects of two different links between different nodes (e.g. $s_{1,3}$ and $s_{2,4}$ when 1,2 are independent transmitters and 2,4 independent receivers) [7] defines it as spatial cross-correlation function (CCF). In principle it is the same as JCF where the positions of second transmitter and receiver are assumed to be the positions after relocation of a single transmitter and a single receiver.

2.2.2.5 Shadowing Simulation Model

Having presented that shadowing is modelled as a random variable with a normal (Gaussian) distribution in dB and the joint correlation function for it, [7] continues

by presenting a shadowing simulation model. This model is based on the fact that an infinite sum of sinusoids with appropriately selected frequencies and random phases can approximate a Gaussian random process. To reduce complexity, a finite number of sinusoids can be used in practice i.e. 6 to 30. To achieve this, the phases of the sinusoids must be drawn from a uniform distribution while the different frequencies and amplitudes must be selected so as that the resulting summation of sinusoids has the same power spectral density (PSD) with a Gaussian process.
Stemming from the fact that shadowing $s_{i,j}$ was previously shown to be a function of transmitter $\mathbf{p}_{\mathrm{TX}} = (x, y)$ and receiver $\mathbf{p}_{\mathrm{RX}} = (u, v)$ locations or in other words a 4-D function $\mathbf{p}_{\mathrm{4D}} = (x, y, u, v)$, the set of spatial frequencies used in the sum of sinusoids is defined as a 4-element vector of spatial frequencies with one element for each of the coordinates of transmitter and receiver. Then taking the Fourier transform of the JCF given in (2.32) , [7] shows that it is equal to the product of the two separate 2-D PSDs for transmitter and receiver movement.
Finally, the authors propose the Monte Carlo method to sample the desired PSD and determine the spatial frequency set and coefficients for the sum of sinusoids model, which approximates the desired Gaussian distribution of the shadowing effect. The shadowing is then generated so as it is the same when transmitter and receiver positions are swapped.

3
Methods

Algorithms to implement the approach for robot positioning previously proposed in [3] were initially designed and computer simulated. Eventually these core algorithms were combined with supporting mechanisms and implemented on a physical robot. These two parts are described in Sec. 3.1, Algorithms, and Sec. 3.3, Physical Implementation, respectively.

3.1 Algorithms

In this section the robot on which the algorithm of discussion is executed on is referred simply as the robot. Any adjacent robot is referred to as a neighbour.

3.1.1 Measurement Collection

Since the overall purpose of this thesis includes gathering knowledge about a local radio channel, collection and storage of measurements is central. The main reason for taking measurements is that we want to be able to predict the SNR, as in Sec. 2.2.2.3, for a radio link between two robots at two arbitrary positions. This predicted SNR is used to evaluate different relocation alternatives as described in Sec. 2.2.2.3. Assuming a two dimensional workspace we can view one measurement of the channel as a line between two points on the workspace plane. Each such measurement is then characterised by one point in 4-D space $\mathbf{p}_{\text{4D}} = (x, y, u, v)$, where $\mathbf{p}_{\text{TX}} = (x, y)$ and $\mathbf{p}_{\text{RX}} = (u, v)$ are the positions of the transmitter and receiver respectively. Such a point is referred to as a 4-D point in the following. Note however that the distance between two 4-D points is not said to be the 4 dimensional Euclidean distance but rather the sum of two distances as in (3.1). This is because of the decorrelation characteristics of shadowing(see (2.32) reproduced from [7]). The distance is:

$$d(\langle x_1, y_1, u_1, v_1\rangle, \langle x_2, y_2, u_2, v_2\rangle) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} + \sqrt{(u_2 - u_1)^2 + (v_2 - v_1)^2} \tag{3.1}$$

We go one step further to simplify the discussion on distance and define the optimistic distance between two 4-D points to be

$$d_{\text{opt}}(\langle x_1, y_1, u_1, v_1\rangle, \langle x_2, y_2, u_2, v_2\rangle) = \min \begin{cases} \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} + \sqrt{(u_2 - u_1)^2 + (v_2 - v_1)^2} \\ \sqrt{(x_2 - u_1)^2 + (y_2 - v_1)^2} + \sqrt{(u_2 - x_1)^2 + (v_2 - y_1)^2} \end{cases} \tag{3.2}$$

In order to visualise the previous claim, consider the case shown in Fig. 3.1 where the transmitter moves from the point with coordinates (x_1, y_1) to the point (x_2, y_2). Similarly, the receiver moves from (u_1, v_1) to (u_2, v_2). Then, according to (3.2) the distance taken into account for the correlation between the pair of measurements (x_1, y_1, u_1, v_1) and (x_2, y_2, u_2, v_2) should be $d_T + d_R$. However, since the channel experienced is the same even when swapping transmitter's and receiver's positions if we swap receiver at (u_2, v_2) with transmitter at (x_2, y_2) the distance would then be a+b. So, (3.2) holds.

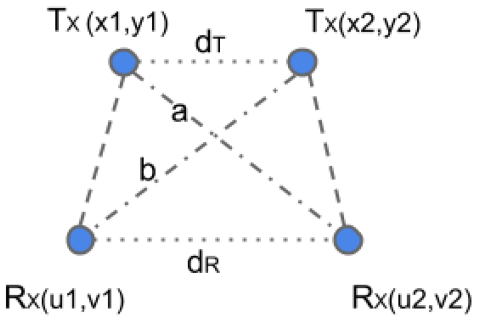


Figure 3.1: A case of moving transmitter $\mathrm{T_x}$ and receiver $\mathrm{R_x}$, where transmitter traverses distance d_T and receiver distance d_R. The reciprocal channel assumption influences the distance accounted in the shadowing correlation to be the minimum of the two sums $a + b$ and $d_\mathrm{T} + d_\mathrm{R}$.

Ideally the channel from all positions to all other positions in the workspace should be measured. This, however, results in an infinite number of measurements and is obviously not achievable. Hence we need to choose a suboptimal measuring strategy that limits the amount of measurements stored at any time. Since the effect of path loss can be calculated deterministically from the distance between the communicating robots, only the random shadowing needs to be captured by the measurements. Shadowing is considered spatially correlated which means that for a prior measurement to be valuable it does not need to have been taken at the exact 4-D point we want to predict the radio channel. Measurements that are close enough to the point of prediction are also valuable. What should be considered close enough relates to the decorrelation distance described in Sec. 2.2.2.4. Our proposed measurement collection strategy should aim at achieving an even density of measurements in 4-D space to enable shadowing prediction between any two points in the workspace.
The second requirement is that we need to obtain a large number of measurements with small distances between them according to (3.2) . This is required for a good decorrelation distance and variance estimation as in Sec. 2.2.2.3.
As will be explained in Sec. 3.1.2, it is beneficial for the prediction time to limit the number of measurements on which the prediction is based. The limited set then needs to contain only the most relevant measurements, that are the ones closest to the 4-D point of prediction. Calculating the distances between the point of prediction and all available measurements before each prediction is a time consuming task. To avoid this our strategy should also store measurements taken in the same physical

area close to each other in memory. In other words the algorithm should, given a prediction point, quickly be able to isolate the memory addresses of interest for that particular prediction.
The fourth requirement on our collection strategy is that it should not interfere with the task of the robots. That is it should not alter the robots' trajectories but rather allow them to collect measurements as they perform their main task. To allow for this we do not predefine exact 4-D points to be visited but rather areas that are considered measured as soon as any point within the area is measured. To elaborate, the probability that a robot visits any out of a finite set of points while moving in a continuous room is equal to zero. Hence, the strategy must rely on the spatial correlation property of shadowing to interpret a measurement as holding information about an area rather than just a point.
To summarise, the way we collect and store measurements must

1. Lead to a spatially even density of measurements
2. Provide measurements with small optimistic distances between them
3. Enable quick access to a limited set of relevant measurements
4. Not require altering of robots' task

To fulfil the above requirements the workspace is divided into squares. The size of each square should be chosen to be on the same order of magnitude as the decorrelation distance defined in Sec. 2.2.2.4. A reasonable value for our indoor experiments and that is used in the following is 1 meter. This is on the same order of magnitude as what was measured in experiments by Mostofi and Yan [3]. Our goal is to have measurements from each square to each other square in the workspace. Each such combination of two squares forms a hypercube in 4-D space. At this stage it is possible to calculate the maximum number of measurements to keep and to preallocate memory for. For example, a 10x10 m^2 workspace results in 100 squares and 10 000 hypercubes. However, we do not take measurements from and to the same square. Also, the radio channel is assumed to be reciprocal. This means that the number of hypercubes m we consider can be found using (3.3)

$$m = \sum_{i=1}^{n} n - i = \frac{n^2 - n}{2} \tag{3.3}$$

where n equals the number of squares in the workspace. In our example case this reduces the number to $m = 4950$.
The second requirement has some special implications for the measurement collection strategy and might be seen as conflicting with the first and the third. That is, the combination of an overall even density and small optimistic distances leads to an everywhere high spatial resolution that counteracts the limiting ambition of the third requirement. However, dissecting further into the underlying motives of these requirements the collection goal illustrated in the simplified two dimensional Fig. 3.2 can be justified. In this figure the measurements are certainly not evenly spaced. Nonetheless the groups of them are which ensures that there exist measurements

in the proximity of any point in the workspace. At the same time the second requirement is fulfilled. We call groups of dense measurements bursts and preallocate memory to save one such burst per hypercube in the workspace. During the burst collection the robot measures the channel at every step it takes. One such measuring operation could include taking the average of a number of measurements depending on the underlying hardware. This strategy results in measurements spaced one step size assuming a static neighbour. Each burst consists of a small number of measurements. In our experiments the number 7 was found to give a good parameter estimation. Multiplying $m = 4950$ calculated in the previous paragraph with 7 gives 34650 which corresponds to the maximum number of measurements to keep in memory.

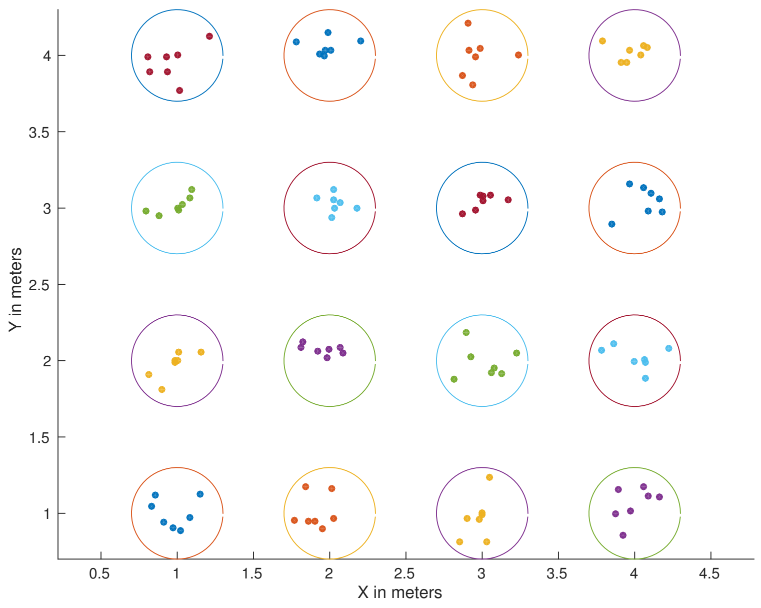


Figure 3.2: An example of measurement groups. Within each group the measurement density is higher than the general global density.

In practice the collected measurements are stored in a 6-dimensional MATLAB data structure. The size of this structure is for our example (10,10,10,10,7,5) where the first four indices indicate in which hypercube the measurements belong. The fifth index indicates the number in the burst. The final "5" stems from the fact that each measurement has 5 attributes, namely x,y,u and v coordinates and the measured signal level. When a new measurement burst containing a number of measurements has been collected, the average coordinates of these measurements decide to which hypercube the burst belongs. The corresponding indices are calculated and the measured levels and the exact coordinates are written to the corresponding memory addresses. We refer to the correlation between memory address and physical coordinate of a measurement as the data closeness property.
A consequence of the reciprocal channel assumption is that what is considered the same measurement, without caution, could be stored in two different locations in the data structure. For example a measurement burst with average coordinates (1,2,3,4)

would be written to an address different from the one of (3,4,1,2) even though they are both measurements of the same channel. The worst outcome of this would be that acquired measurements are not used because the predictor searches only for measurements with the alternative coordinate order. To prevent such issues a convention for the ordering of coordinates is adopted. It claims the following

1. $u \geq x$
2. If $u = x$ then $v \geq y$

If any of these does not hold, the coordinates are changed $x \to u, y \to v, u \to x, v \to y$ which will cause the coordinates to follow the convention. The coordinate check is automatically performed on all read and write operations on the data structure. Collection of a measurement burst by a robot is triggered when itself or its neighbour has moved a certain distance from the point of the start of the last measurement. More specifically, at each iteration the optimistic distance between the current positions of the robot and its neighbour and the position of the previous measurement is calculated. If this distance is greater than a threshold a new burst collection is initiated. The threshold should be chosen such that it gives a high probability that a burst is written to any hypercube the robots have crossed. In practice a threshold equal to or smaller than the length of one side of a workspace square gives adequate such probability. When a burst has been collected the corresponding hypercube is calculated and the burst is written to the data structure. Depending on where the robot is located the new burst might overwrite an old one or be put in previously empty memory. With this strategy the collection of measurements is kept updated over time without limiting the ability of the robots to perform their main task.

3.1.2 Parameter Estimation

To minimise the end-to-end bit error rate in our wireless network we rely on knowledge about the radio channel. One global such piece of information is the one of the channel parameters. We model the radio channel the same way as did Y. Mostofi and Y. Yan in [3] and as generally described in Sec. 2.1. That is, we consider the effect of pathloss, shadowing and multipath. Hence, the parameters of interest for a proper prediction are the ones given in table 3.1.

Parameter name	**Symbol for estimate**
Alpha	$\hat{\alpha}$
Pathloss exponent	$\hat{n}$
Decorrelation distance	$\hat{\eta}$
Shadowing variance	$\hat{\xi}^2_{\text{dB}}$
Multipath variance	$\hat{\rho}^2_{\text{dB}}$

Table 3.1: Estimated parameters

Note that in contrast to (2.32) although in accordance with [3], we include the factor of ln(2) in η and hence model shadowing correlation as

$$C_{\mathrm{SH}}(d_{\mathrm{opt}}) = \xi_{\mathrm{dB}}^{2} \exp\left(-\frac{d_{\mathrm{opt}}}{\eta}\right). \tag{3.4}$$

The process of estimating the path loss related parameters based on measurements of the channel is straightforward. It uses the standard least squares approach stated in (2.24) and returns a vector $\hat{\theta} = [\alpha \quad \hat{\eta}]$.

The estimation of the remaining parameters relies heavily on the spatial correlation property of shadowing and will require a more detailed explanation. The mathematical relation between the measurements and these parameters is readily defined in (2.25). Note that the vector $\mathbf{Y}_{\mathbf{G}q}$ contains the original measurements modified to remove the effect of pathloss. An element of this vector is in the following referred to as just a measurement and corresponds to an estimate of only shadowing at that point. An SNR product is the product of two elements of this vector like

$$\text{SNR product} = [\mathbf{Y}_{\mathbf{Gq}}]_j [\mathbf{Y}_{\mathbf{Gq}}]_k \tag{3.5}$$

where j and k belong to the set of integers ranging from one to the number of available measurements.

(2.25) in its original setting of [3] assumes measurements taken with a static transmitter. In [7] it is claimed, however, that the expected correlation of shadowing between two position pairs depends only on the sum of distances defining our optimistic distance in (3.2). Hence, (2.25) can still be considered valid if the distance l is replaced by the optimistic distance d_{opt}.

The process of finding the shadowing parameters can be viewed as the fitting of a curve to measured data. According to (2.25), with our distance modification, the average product of measurements at a certain optimistic distance from each other should be calculated. Each such distance corresponds to one term of the sum to be minimised. However, since the positions of our measurements are free to take on any value, representable by 64 binary bits, most of the calculated optimistic distances are likely to be unique. A consequence is that the number of terms comes close to the number of measurement combinations. This is generally a large number that results in an unfeasible computation time. To improve performance we discretise the range of optimistic distances between measurements. That is, each product of measured SNR's is added to one bin out of a finite set of bins based on its optimistic distance. The concept is illustrated in Fig. 3.3.

After binning of the products of all combinations has been performed, the squares of the differences between these correlation estimates and the model are calculated. We let highly optimised MATLAB functions find the model parameters that minimises the sum of these squares. The model fitting problem is visualised in Fig. 3.4. The small dots correspond to one average of the SNR products each. Hence, the square of the deviation of this SNR product average from the line of the model makes up one term in the minimisation problem. Note carefully in Fig. 3.4 that the average product at distance zero is a special case that is not included in the minimisation. This is because the average product at distance zero, that is the estimate of the channel variance, is affected by both shadowing and multipath. Since multipath is modelled as spatially uncorrelated it does not contribute to the correlation estimates at distances other than zero. Hence, and according to (2.26), we can estimate

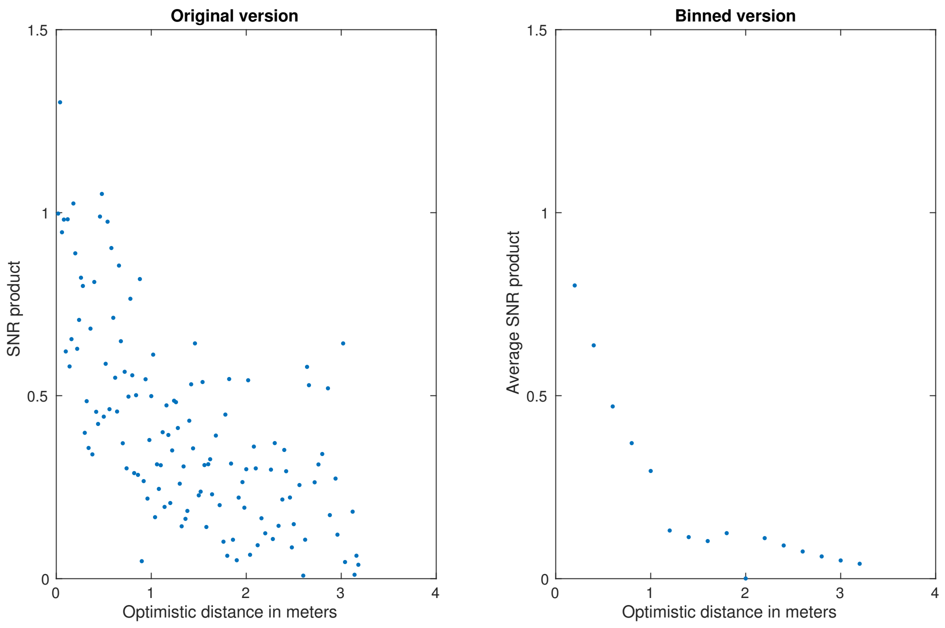


Figure 3.3: A comparison of an unbinned and a binned shadowing correlation plot

multipath variance as the difference between the total variance and the intersection of the model and the y-axis in Fig. 3.4.
Since the model employed for shadowing correlation states an exponential decay of correlation with distance, the correlation of measurements with large optimistic distance between them is modelled as close to zero. This also means that a certain amount of sampling error E_s degrades the parameter estimation more the larger the distance at which it occurs is. The phenomenon is illustrated in Fig. 3.5. In our numerical experiment we define the error of a shadowing correlation model to be the integral from distance zero to ten of the square of the difference between the model and a reference.

$$E = \int_0^{10} \Bigg(\underbrace{\hat{\xi^2_{\mathrm{dB}}} \exp\Big(- \frac{d}{\hat{\eta}} \Big)}_{\text{model}(d)} - \underbrace{\exp\Big(- d \Big)}_{\text{ref}(d)} \Bigg)^2 \mathrm{d}d \tag{3.6}$$

In equation (3.6) $\hat{\xi^2_{\mathrm{dB}}}$ and $\hat{\eta}$ are model parameters that stem from available measurements. In our experiment they should both be equal to one for zero error. However, to characterise this resulting model error as a function of the optimistic distance at which a sampling error occurs, the ideal measurements were distorted in a controlled manner. The question to answer is the following. Given that the best possible model parameters are used and if the model is forced to pass through one erroneous point, how does the resulting error of (3.6) depend on where the erroneous point is located. Fig. 3.5 shows the minimum possible error as a function of where a fixed difference occurs. We generate the figure by creating one curve referred to as ref(d) that is considered correct and noiseless with realistic shadowing parameters of $\xi^2_{\mathrm{dB}} = 1$ and $\eta = 1$, like in equation (3.6). Another curve, constituting our model, that passes

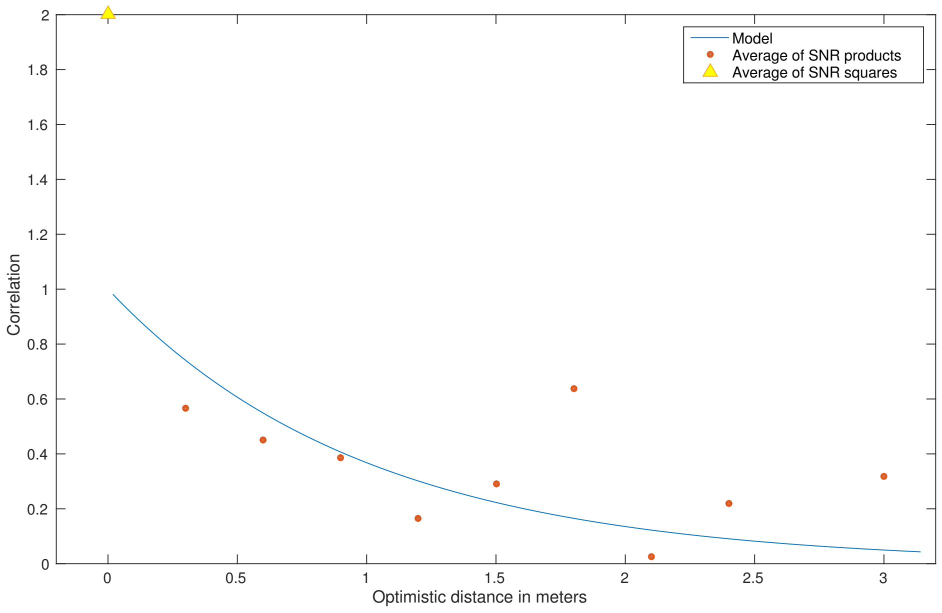


Figure 3.4: Fitting of the shadowing correlation model to measurement based correlation estimates.

through the point $(d_E, \mathrm{ref}(d_E) + 0.1)$ is generated where $E_\mathrm{s} = 0.1$ in one point, namely at d_E. The $\hat{\xi^2_{\mathrm{dB}}}$ and $\hat{\eta}$ that minimise the error are found with numerical methods. The error is then calculated and added to the left plot of Fig. 3.5. The procedure is repeated for numerous values for d_E in the range $[0, 10]$.
The left of Fig. 3.5 clearly shows that, for the investigated case, the error in the resulting model grows considerably with the distance at which the fixed error occurs. Another plot, to the right in the figure, shows the lines of the resulting models described in the previous paragraph. It can be seen that the 0.1 difference in the estimated correlation occurs at distances 2.1 m and 4.6 m for the two lines respectively and that the former deviates considerably less from the true correlation. Note that noise in the position estimates of measurements is not considered. However, this noise is in practice several orders of magnitude lower than the sampling error of spatial correlation and can hence be excluded from this reasoning. Also note that, when performing the real parameter estimation, a certain amount of error in the estimate at one single distance, in general, does not cause the resulting model to pass directly through this point. This is because the algorithm searches for a least squares solution involving more equations than only the one in error. A consequence is that the model errors, in practice, rarely render so obvious as in Fig. 3.5. Nonetheless, the emphasised relationship between sampling error and distance holds also for this case. Without mathematical proof we observe and conclude that for all reasonable parameter values the prediction is more sensitive to sampling error at larger distances.
As briefly mentioned in Sec. 3.1.1 the number of measurements stored for a limited indoor workspace is generally on the order of thousands. The optimistic distances

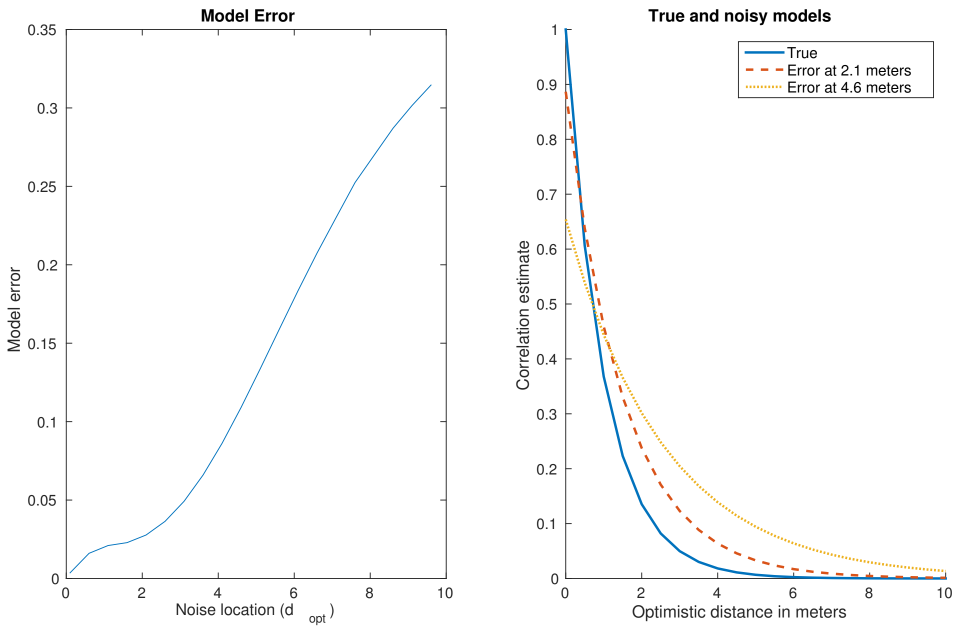


Figure 3.5: Left: The error in the resulting correlation model as a function of the optimistic distance at which correlation sampling error occurs. Right: Examples of true and corresponding erroneous models. The models differ since the underlying measurements were distorted to the same degree but at different optimistic distances. It is seen that sampling error at a small distance gives smaller model error than the same sampling error at a larger distance.

and products of all combinations of such measurements could, in theory, be included in the shadowing parameter estimation. However, the computational time for such an operation is unfeasible for an embedded implementation. Also, justified in the previous paragraphs, small amounts of sampling error in the product of SNR's at large distances could severely degrade the quality of the estimate. To overcome these two issues we rely on a convenient property of the measurement collection strategy of Sec. 3.1.1, namely the data closeness. This enables the estimation algorithm to efficiently combine each measurement only with others belonging to the same hypercube. With this approach a limited number of, and only the most valuable, measurement combinations are included in the shadowing parameter estimation process.

3.1.3 Channel Prediction

When steering robots to maximise radio link performance according to the procedure described in Sec. 3.1.4 it is necessary to be able to predict SNR at unvisited locations. Yan and Mostofi [3] provide us with a channel prediction framework as described in Sec. 2.2. While the proposed prediction equations are completely defined in Sec. 2.2.2.3, they will be further explained here.
Equations (2.27) and (2.28) provide a good overview of the prediction mechanism. Assuming transmitter and receiver position $\mathbf{x}_{\mathrm{TX}}$ and $\mathbf{x}_{\mathrm{RX}}$ respectively and either estimated or default channel parameters, the predictor can be thought of as consisting of one deterministic part that is a function only of the distance $\|\mathbf{x}_{\mathrm{TX}} - \mathbf{x}_{\mathrm{RX}}\|$. Moreover, in the presence of measurements, it adds to this deterministic prediction an adjustment that is calculated using nearby measurements of the local shadowing. Assuming the predictor is queried for the SNR of the link between $\mathbf{x}_{\mathrm{TX}}$ and $\mathbf{x}_{\mathrm{RX}}$ as before. It initially checks if the coordinates follow the convention of enumeration 3.1.1 and swaps them if necessary. For convenience we denote the 4-D point of prediction with $\mathbf{x}$. The predictor then calculates the index of the hypercube in which the prediction lies. All measurements from that and the neighbouring hypercubes are extracted and concatenated to form a matrix of measurements local to the point of prediction. We refer to the column of measured SNR values of this matrix as $\mathbf{Y}$ in accordance to (2.27).
The algorithm continues by walking through all combinations of these local measurements and calculating the optimistic distances between them. The optimistic distances along with the estimated or default shadowing and multipath parameters are used to populate a covariance matrix Φ with

$$\begin{aligned}\Omega_{i,j} = \Omega_{j,i} &= \hat{\xi}^2_{\mathrm{dB}} \exp\left(-\frac{d_{\mathrm{opt}_{i,j}}}{\hat{\eta}}\right) \\ \Phi &= \Omega + \hat{\rho}^2_{\mathrm{dB}} \mathbf{I}. \end{aligned} \tag{3.7}$$

Note that the Ω matrix is symmetric and hence the same calculated covariance is written to both $\Omega_{i,j}$ and $\Omega_{j,i}$ during one step. In (3.7), $\mathbf{I}$ is an identity matrix of the same size as Ω. The identity matrix is used to add multipath variance to the diagonal of Φ. Since multipath is considered spatially uncorrelated it appears only on the diagonal where the optimistic distance between a measurement and itself is

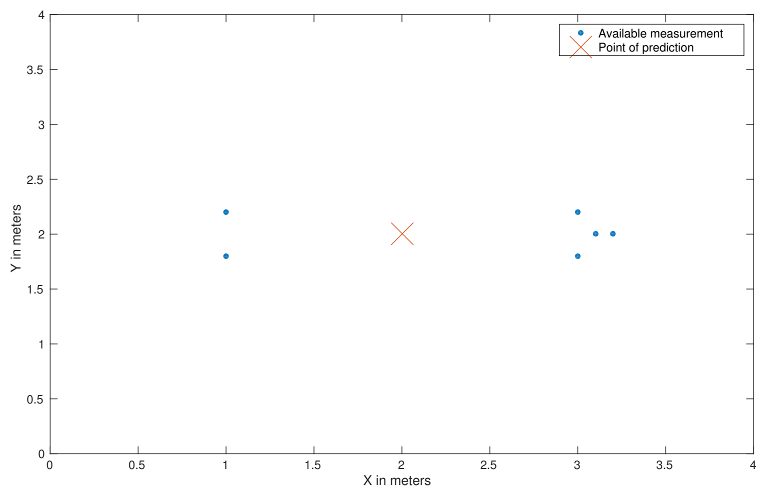


Figure 3.6: An example scenario for prediction. On one side of the point of prediction there are more highly correlated measurements than on the other.

exactly zero. See Fig. 3.4 for a visualisation of this property where the SNR product at distance zero is much higher than the correlation model.
In the next step the predictor creates a column vector Ψ of correlations between the point of prediction, $\mathbf{x}$, and all local measurements as in (3.8). The measurements must be ordered in the same way as in Φ.

$$\Psi_i = \hat{\xi}_{\mathrm{dB}}^2 \exp\left(-\frac{d_{\mathrm{opt}_{x,i}}}{\hat{\eta}}\right) \tag{3.8}$$

Finally the estimated or default pathloss parameters are included to perform the matrix multiplications of (2.27) and (2.28). The result is one predicted mean SNR and one accompanying variance for the 4-D point $\mathbf{x}$.
If the reader is looking for a more intuitive understanding of these matrix multiplications please consider the following example. In Fig. 3.6 is seen a potential scenario where previous measurements are illustrated as dots and the point of prediction as a cross. Note that to visualise such a scenario in two dimensions the opposite end of the measured link must be equal for all measurements as well as for the point of prediction. The shadowing prediction at the cross consists of a linear combination of all local measurements. That is, a weighted sum of the measured SNR values with pathloss removed $\mathbf{y} - \mathbf{G}_q\hat{\theta}$, or simply shadowing measurements, at the locations of the dots. The weights are equal to the amount of correlation between the cross and the dots according to the exponentially decaying shadowing correlation model. However, using only this, the prediction at the location of the cross in Fig. 3.6 would be more affected by the measurements to the right because of their larger number. To compensate for this $\mathbf{y} - \mathbf{G}_q\hat{\theta}$ is multiplied with the inverse of the covariance matrix Φ^{-1} to reduce the individual weights of measurements that are close together and hence have a large covariance. In a similar manner the reduction in variance is formed.

3.1.4 Control Law

Even though the probabilistic objective function of (2.29) accurately defines the optimum positions for all robots it does not provide an algorithm for the robots to reach this state. Hence, a motion control law must be derived from (2.29). In this thesis we use a similar approach to this problem as (did Y. Mostofi and Y. Yan) in [3]. This involves locally evaluating the objective function at different points of interest and steering the robot towards the best of these points.
A robot considers only the two terms of the objective function that depend on its own position, that is the terms corresponding to its left and right link. The goal is to find the position that maximises the sum of these terms. To evaluate a term of the objective function for a certain position of interest one needs to be able to predict the SNR of the reception from the corresponding neighbour at that position. For this purpose we use the channel predictor described in Sec. 2.2.2.3. The predictor is a function of two positions that returns the predicted SNR for a radio link between these points as well as the uncertainty, or variance, of the prediction. One of the positions used with the predictor is always the position of a neighbour to which the current term in the objective function corresponds. The second position is varied within a square window centred on the robot. For each evaluated position within the window, the SNR and variance of both the left and the right link is predicted. These SNRs and variances are then used to evaluate the two terms of the objective function. When all positions have been evaluated the one that maximises the objective function is found and the robot is steered towards this point. In practice the vector difference between the position of the maxima and the robots current position is formed. The resulting vector is then scaled so that its length is equal to our preferred step size. The scaled vector, referred to as the relocation vector, is then fed to the obstacle avoidance algorithm described in Sec. 3.1.5.
Ideally the size and resolution of the search window should be infinite. However, because of limited computational resources the number of evaluated points must be strictly limited. The size of the window should be large enough for the difference in path loss from one side of the window to the other to be larger than the standard deviation of shadowing. This gives a high probability that the robot moves towards the global maximum of the objective function and does not get stuck on a local maxima caused by shadowing. Moreover, the resolution of the window should preferably be close to the decorrelation distance. This ensures that measured areas are not missed in the search while computation does not become unnecessarily heavy.
The desirable behaviour of a robot is that, assuming static neighbours, it reaches an optimal position within a limited amount of time after which it remains static. Without further improving the control law, however, a particular situation illustrated in Fig. 3.7 may occur and prevent this desired behaviour.
See Fig. 3.7 for a theoretical continuous two dimensional objective function. The dots indicate points at which the control law evaluates this function during the window search and the cross is the current position of the robot. During window search a, the left part of Fig. 3.7, the current position of the robot is considered the best out of the evaluated alternatives. In this situation the robot will remain static. However, since the system needs to be self stabilising the current position needs

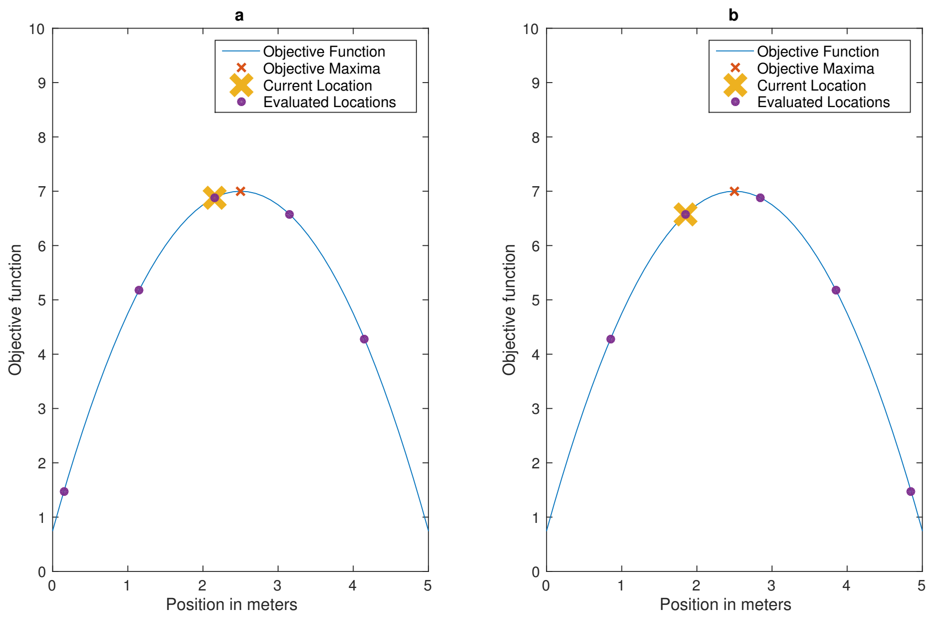


Figure 3.7: Two realisations of the search window differentiated by a small amount of position estimate noise.

to be continuously measured. Because of the non zero noise level of the position estimates the situation illustrated in the right part of Fig. 3.7 will eventually occur. The plot shows how a noisy estimate causes the estimated position of the robot and therefore all evaluated points in the window centred on the robot to be shifted to the left. This has the consequence that a point other than that of the robots current position is considered optimal and hence the robot is moved. In practice this causes the robot to move back and forth over a peak of the objective function.
To prevent this inefficient behaviour one extra step is added after the main window evaluation. If the window maxima is found at a point that is a direct neighbour of the current position of the robot, a new window with higher resolution, indicated by circles in Fig. 3.8, is formed around the robot. If the maxima in this new window is again a direct neighbour of the current position the robot is not moved.

3.1.5 Obstacle Avoidance

Included in our solution is a simple obstacle avoidance algorithm that requires known obstacles to be preprogrammed into the robot. Each obstacle is represented as a set of polygons. Each such polygon includes a vector of unit length that is perpendicular to and pointing out from its outer edge as in the left half of Fig. 3.9. Such a vector is referred to as the normal vector of that polygon. The right side of Fig. 3.9 shows a complete obstacle made up of four polygons.
The obstacle avoidance algorithm receives a relocation vector from the control law. The sum of the current position and the relocation vector is called the preliminary destination. If the preliminary destination does not lie within the area occupied by a known obstacle it is fed to the robot interface immediately. If however, the

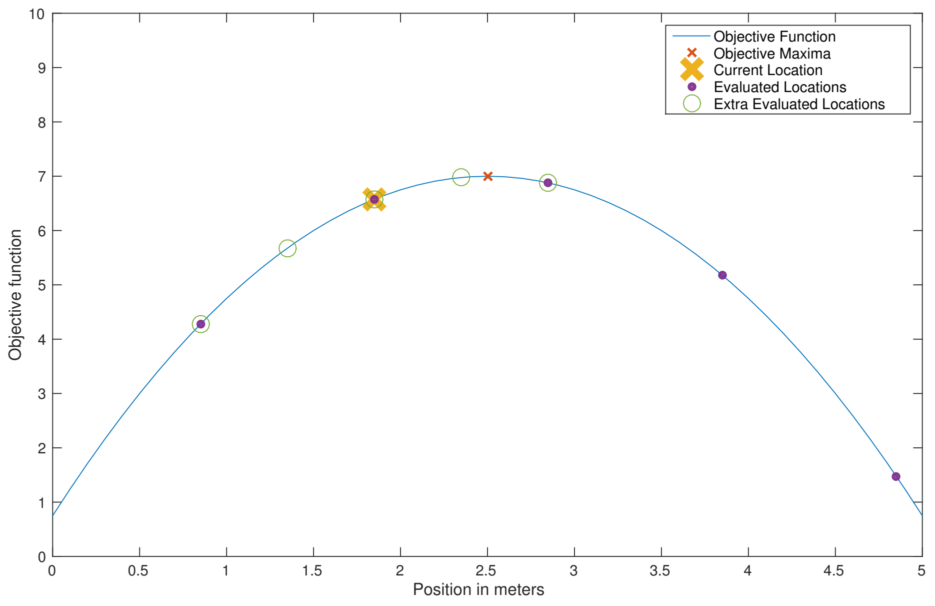


Figure 3.8: The high resolution search window formed if the original window search results in a maxima neighbouring the current position.

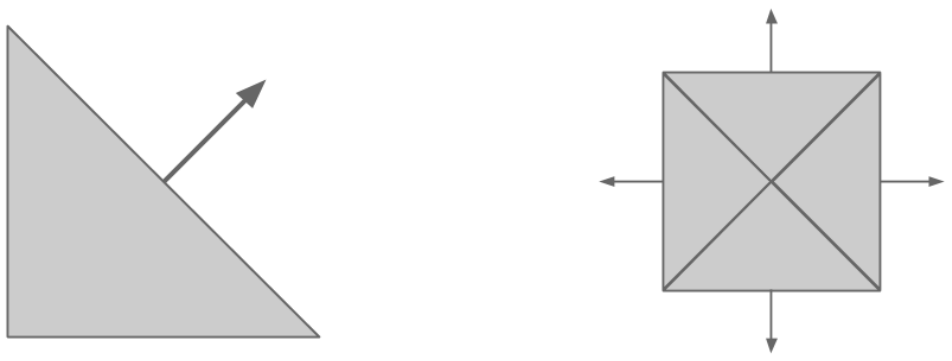


Figure 3.9: Left: One obstacle polygon with its normal vector. Right: A complete obstacle made up of four polygons.

preliminary destination does lie within such an area, a modified destination must be formed not to crash the robot into the obstacle. Such a destination is computed through a change of basis. The relocation vector **r**, defined in the basis of the global coordinate system, is transformed to a new basis. In the new basis the normal vector of the interfering polygon forms one of the basis vectors. The change of basis is shown in (3.9) where **n** corresponds to the normal vector and n_1 and n_2 to its components in the x and y direction respectively.

$$\mathbf{C} = \begin{bmatrix} n_1 & n_2 \\ n_2 & -n_1 \end{bmatrix} \tag{3.9}$$

$$\mathbf{q} = \mathbf{C}\mathbf{r}$$

In (3.9) the vector $\mathbf{q}$ now holds the same relocation information as the original vector $\mathbf{n}$ but represented using another basis. At this stage it is easy to remove the component of this vector pointing towards the obstacle. As can be seen in (3.10) any negative component parallel to the normal vector of the interfering polygon is replaced by a zero. The resulting vector is then transformed back to the original basis and scaled to the preferred length to form the final relocation vector $\mathbf{r}\prime$. This allows the robot to move only alongside the obstacle and not towards it.

$$\mathbf{q}\prime = \begin{cases} (0, q_2) & \text{if } q_1 < 0 \\ (q_1, q_2) & \text{otherwise} \end{cases} \tag{3.10}$$

In order to present the performance of the obstacle avoidance algorithm we shall employ the deterministic model of pathloss only. Consider two static end-nodes in the chain of communicating robots. Then the intermediate node tries to relocate itself in the middle of the line between the end-points according to theory. Illustrated in Fig. 3.10 is the described process.

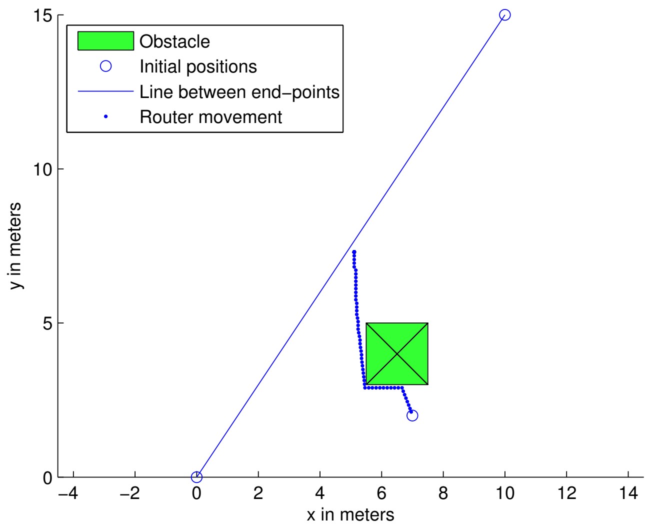


Figure 3.10: The intermediate node relocates itself according to (2.18) changing its route only to avoid an obstacle.

The algorithm as described so far does not yet guarantee obstacle avoidance. A certain situation illustrated in Fig. 3.11 might occur and lead the robot to hit the obstacle.

This matter arises when the moving step of the robot is set to be greater than the physical dimensions of the obstacle along the path of movement. Since only the preliminary destination and not the path to it is assured to lie outside all polygons, an obstacle might occupy an area somewhere between the robots position an the destination. This problem however, is easily solved by expanding the representation of all obstacles by half the length of the relocation vector.

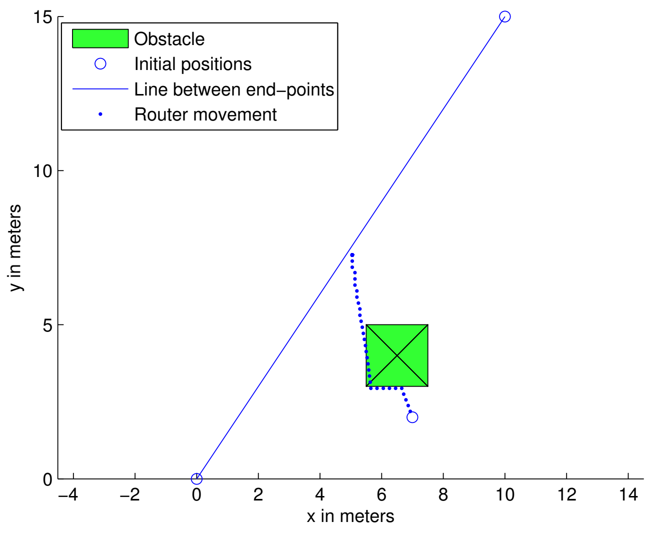


Figure 3.11: The intermediate node relocates itself according to the control law and collides with the obstacle.

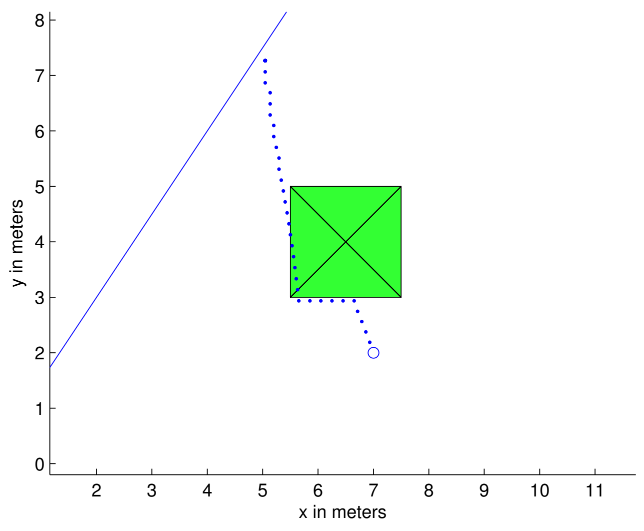


Figure 3.12: The intermediate node relocates itself according to the control law and collides with the obstacle.

3.2 Simulation

Before connecting the developed algorithms to our hardware computer simulations were performed in the MATLAB environment. Note that we take a shared memory approach in our simulations. That is we do not simulate the complexity of message passing but rather allow each robot to directly read a property of any other robot as well as the shared measurement data structure. Also we do not consider dynamics of the mechanical robot system. The reason for this is to isolate the performance of the control law and measurement collection strategy.

3.2.1 Channel Simulation

In order to evaluate the performance of our algorithm we simulate a channel in the 2-D workspace. The channel seen between transmitter and receiver at every moment is a function of the corresponding locations they are placed. As it has already been described the channel consists of three spatio-temporal components. Namely, the pathloss, the shadowing and the multipath. In the following we note transmitter's coordinates with (x, y) and receiver's with (u, v).
To simulate the linear decay of pathloss with distance in the dB scale we model it as in (2.21).
For the shadowing, we follow the approach found in [7]. More specifically, consider the 1-D function of shadowing in time. Then, as we have previously described a sum of N sinusoids $\hat{s}$ can approximate the shadowing effect s described in Sec. 2.2.2.4.

$$\hat{s} = \sum_{n=1}^{N} c_n \cos(2\pi f_n t + \theta_n), \tag{3.11}$$

where f_n is a vector containing carefully selected spatial frequencies, c_n the amplitude with $c_i = c_j$ for $i, j \in \{1, ..N\}$ and θ_n the corresponding phase from a number of N phases drawn from a uniform distribution. The goal is to select the appropriate f_n, c_n and θ_n in such a way that the PSD of the sum in (3.11) resembles the one of the desired Gaussian process representing shadowing.
In an analogy with the time dependent shadowing, shadowing which depends on transmitter's and receiver's locations s(x,y,u,v) can be expressed as a function of time for moving end points. Then the sum of sinusoids estimating the shadowing is:

$$\begin{aligned}\hat{s}(x, y, u, v) &= \sum_{n=1}^{N} c_n \cos[2\pi \mathbf{f}_n [x\ y\ u\ v]^T) + \theta_n] \\ &= \sum_{n=1}^{N} c_n \cos[2\pi(f_{x,n}x + f_{y,n}y + f_{u,n}u + f_{v,n}v) + \theta_n],\end{aligned} \tag{3.12}$$

with $\mathbf{f}$ denoting the vector of spatial frequencies $[f_x\ f_y\ f_u\ f_v]$. The vectors $\mathbf{f}_\mathrm{T} = [f_x\ f_y]$ and $\mathbf{f}_\mathrm{R} = [f_u\ f_v]$ can represent the spatial frequencies of transmitter's and receiver's movement in time. The joint spatial correlation is a 4-D function that can be expressed as the product of two 2-D functions depending on the transmitter's and receiver's positions according to (2.32) with $d_\mathrm{T} = [\Delta x, \Delta y]$ and $d_\mathrm{R} = [\Delta u, \Delta v]$.
The appropriate values of amplitude, spatial frequencies and phases have already been derived so as that the PSD of the sum resembles a Gaussian process. These values are given in [7] as follows:

$$\mathbf{f}_\mathrm{T} = \mathbf{f}_\mathrm{R} = \frac{\alpha}{2\pi}\sqrt{\frac{1}{(1-\beta)^2} - 1}, \tag{3.13}$$

where $\alpha = \frac{\ln 2}{d_\mathrm{corr}}$ and d_corr is the decorrelation distance, β is a uniformly distributed random variable . Note that $\mathbf{f}_\mathrm{T} = \mathbf{f}_\mathrm{R}$ since the ACF of (2.31) is the same if instead of a moving receiver we assume a moving transmitter with the same position shift. Equivalently, it means that $R_\mathrm{s}\{0, d_\mathrm{R}\} = R_\mathrm{s}\{d_\mathrm{T}, 0\}$ when $d_\mathrm{T} = d_\mathrm{R}$. Then,

$$\mathbf{f}_x = \mathbf{f}_u = \mathbf{f}_\mathrm{T}\cos(\phi) \text{ and } \mathbf{f}_y = \mathbf{f}_v = \mathbf{f}_\mathrm{T}\sin(\phi), \tag{3.14}$$

where ϕ is drawn from a uniform distribution in $[0, 2\pi)$. θ_n of the sum (3.12) is drawn from a uniform distribution as follows:

$$\theta_n = U(0, 2\pi) \tag{3.15}$$

In order to simulate the reciprocality of shadowing, we need the spatial frequencies of transmitter and receiver to form symmetrical vectors. Thus, we consider an even number of sinusoids N. Then, we sample the random variables presenting in (3.13) to (3.15) for N/2 values. We append to the right side of the spatial frequency vectors the result of their product with the matrix $\begin{bmatrix} 0 & I \\ I & 0 \end{bmatrix}$ as shown in the following equation:

$$[f_x \; f_y \; f_u \; f_v]_{n+N/2} = [f_x \; f_y \; f_u \; f_v]_n \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}, \quad \text{where } n = 1, ..., N/2 \tag{3.16}$$

In other words it holds that $f_{\mathrm{T}_n} = f_{\mathrm{R}_{n+N/2}}$ and $f_{\mathrm{R}_n} = f_{\mathrm{T}_{n+N/2}}$, which makes the frequency set symmetric with respect to $f_\mathrm{T} = f_\mathrm{R}$ resulting in same shadowing experienced for uplink and downlink. The phases of the sinusoids θ_n are formed as follows:

$$\theta_{n+N/2} = \theta_n, \tag{3.17}$$

where n=1,..,N/2.
For the amplitudes of the sinusoids we have described that they must be chosen so that the resulting PSD of the sum is similar to the PSD of the Gaussian process of shadowing. Recall that in time domain a sinusoid with amplitude A is given in frequency domain by two spikes of magnitude A/2. This is due to the fact that:

$$X(t) = A\sin(2\pi f_0 t) = A\frac{e^{j2\pi f_0 t} - e^{-j2\pi f_0 t}}{2}.$$

The Fourier transform of the previous equation supports our claim. Then the PSD is given in frequency domain with two spikes of magnitude $(A/2)^2 = A^2/4$. The root mean square (rms) power of a continuous time sine wave can be also calculated as follows:

$$P_\mathrm{rms} = \frac{1}{T_0}\int_0^{T_0} (A\sin(2\pi f_0 t))^2 dt = \frac{A^2}{2},$$

which is equal to the sum of the two spikes $A^2/4$ that give the PSD in frequency domain. This observation comes due to Parseval's theorem. You can clearly see that P_rms is calculated by squaring the signal and then averaging over a single period. In other words the PSD of a time series $x(t)$ describes how the variance of the data $x(t)$ is distributed over the frequency components into which $x(t)$ may be decomposed. Returning to our sum of sinusoids the PSD will be calculated for the sum of all different frequency components meaning that we need to find the variance of the sum. Keep in mind that we need to produce shadowing with a specific standard deviation, say σ_{std} but also that $\text{var}(X + Y) = \text{var}(X) + \text{var}(Y)$ and that $\text{var}(aX) = a^2\text{var}(X)$. Then the amplitudes of the sinusoids c_n can be found as follows:

$$c_n = \sqrt{\frac{2}{N}}\sigma_{std}, \tag{3.18}$$

which means that the total sum will have a PSD equal to $c_n^2/2 = \sigma_{\text{std}}^2/N = var(s)/N$. The variance will be distributed equally to all spatial frequency components.
The reader can find in the appendix A a code to generate spatially correlated shadowing as explained above, as well as code to simulate received SNR via a channel according to the combined model comprising pathloss, shadowing and multipath.

3.2.1.1 Multipath

Most real radio channels are severely affected by the multipath phenomenon described in Sec. 2.1. Hence this needs also to be included in the simulated channel. Compared to shadowing, the level of multipath decorrelates very quickly. A rule of thumb is that two points have completely uncorrelated multipath when their distance is greater than or equal to 0.4λ where λ equals the radio wavelength[6]. In our case this critical distance is approximately 5 centimetres.
5 centimetres is much less than both the step length when the robot relocates and the resolution of the search window, that is the system is unaware of the spatial correlation. Because of this a simplified way of simulating multipath is employed. When one realisation of the random channel is generated, so is also a 4-D grid of 10 centimetre resolution. Each point in the grid is assigned a multipath value drawn from a distribution with the desired variance. When the channel object is requested to evaluate the channel between two points, it adds to the return value the multipath of the closest point in the grid. With this technique the simulated channel has a constant multipath within 10 centimetre squares.

3.2.2 Program Execution

The simulation script creates one robot object per simulated robot and sets up their initial positions. It also creates one realisation of the random shadowing as in Sec. 3.2.1 and makes it available to the robots. Potentially obstacles are created and a reference to these is given to the robots as well. The simulation loop then iteratively invokes each robot's main method in sequence. The main method is illustrated in Fig. 3.13 and each process is briefly described below.
Initially, the control law and obstacle avoidance processes described in sections 3.1.4 and 3.1.5 are executed. The coordinates of the resulting new position are written to the robot's properties. Next a measure condition is checked. It causes the robot to measure the channel if either it has moved a certain distance since the last measurement, or it is currently in a measurement burst initiated during a previous iteration. If the robot is to measure the channel it reads out the current position of its left neighbour and uses the simulated channel to evaluate shadowing between that position and its own. After appending the new measurement to a temporary burst

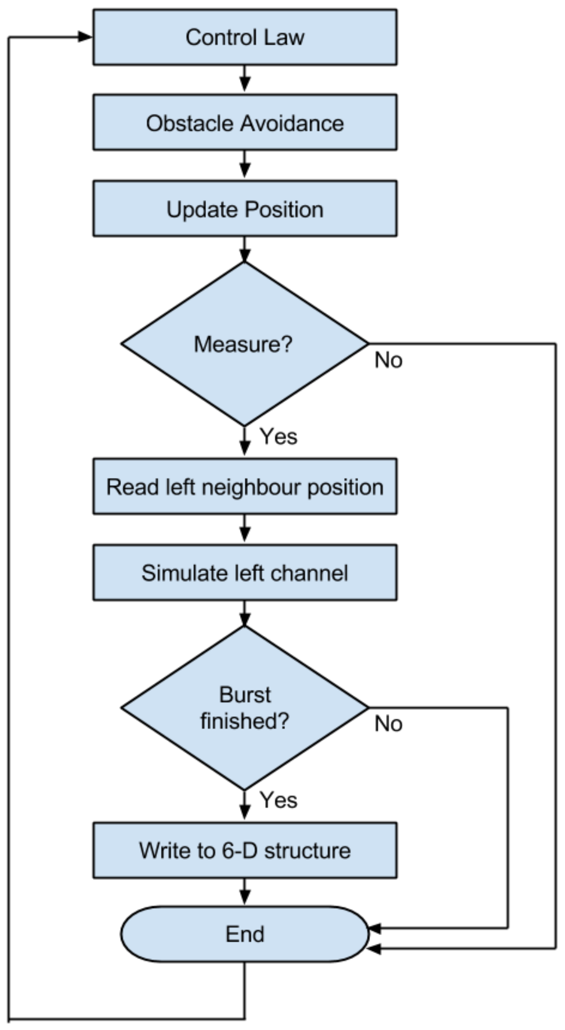


Figure 3.13: The program execution of a single robot during simulation

matrix it checks if that matrix has reached the preferred size, that is the number of measurements per burst. If the size is adequate this complete burst is written to the shared 6-D measurement data structure.
The end to end bit error rate (BER) is calculated using two different models at every iteration in the simulation loop. First, the deterministic BER is calculated using SNRs of the pathloss only model of (2.11). Second, the simulated channel is used to draw random SNRs between the positions of the robots. Then the corresponding BERs are calculated as well. The resulting probabilities of correct reception, P_c according to (2.16), are plotted at the end of each simulation.

3.3 Physical Implementation

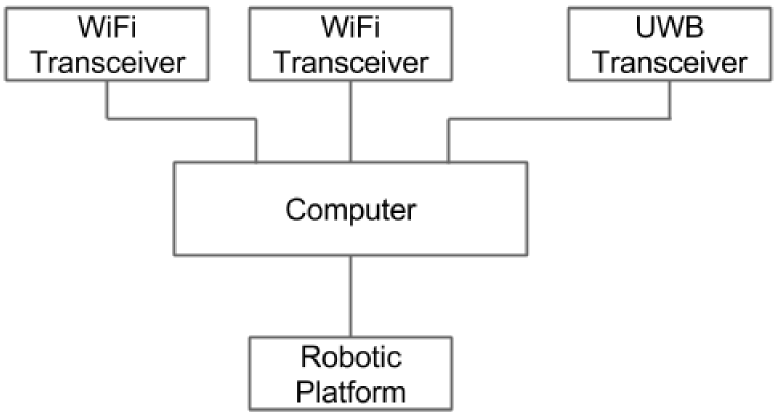


Figure 3.14: Schematic including the main components of a complete physical robot unit.

3.3.1 Robotic Platform

3.3.1.1 Pioneer 3-DX

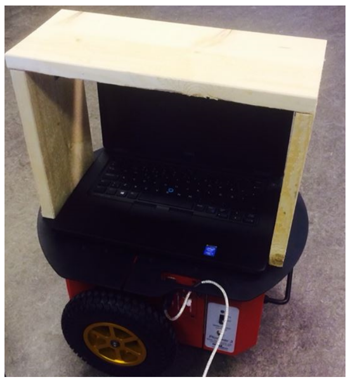


Figure 3.15: One of the robots with the computer on top. The table over the computer was used to place the tranceivers.

The mobile robotic platform used for our implementation is the Pioneer 3-DX from Adept[8]. This platform was chosen since it is fully programmable. The robot acts as a server which simply waits for commands. It is also equipped with sonars, which can be employed to avoid collisions with obstacles that were not mapped in simulation.

3.3.1.2 Software Interface

To control the robot from software a serial communication protocol needs to be implemented. The protocol provided by the manufacturer allows for commands to

be sent to the robot and for information to be sent back to the computer. The robot sends information packets to computer several times per second by default.
The commands sent to the robot consist of packets including a fixed header, a command number, command arguments and a checksum. The information packets sent from the robot are constructed in a similar manner and contain for example sonar readings and velocity of the robot. All communication with the robot is handled from inside MATLAB but in a separate script called the robot interface run in parallel with the main program. The reason is the need to constantly monitor sonar readings and if necessary send emergency stop commands to avoid collision. Also, the robot expects frequent communication to keep the established connection alive. The robot interface listens for commands on an IP network socket. Any other program should use this IP address as an interface to the robot. In our case where both the robot interface and the main program run on the same computer the data is looped backed in the Linux kernel. Note however that, in general, this approach makes it easy to control the robot remotely.

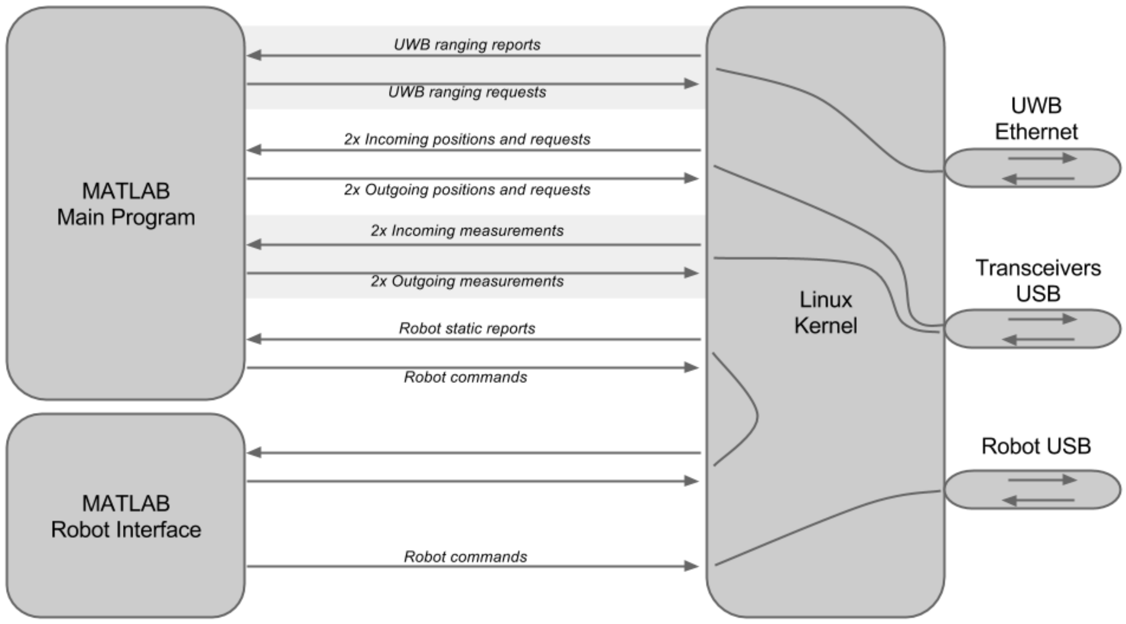


Figure 3.16: The main software modules and their interconnects.

The standalone robot interface can be thought of as a gatekeeper between the robot and any other program and has the responsibility to:

1. Prevent initialisation of movement towards obstacle
2. Interrupt movement towards obstacle
3. Report static state
4. Keep connection alive

The first point is handled by inspecting all incoming packets on the IP socket. If it contains a movement command the latest received sonar readings are checked to

make sure that there is no obstacle within a certain distance in the desired direction of movement. Only when this check has been passed the command is sent to the robot.
Secondly, at the reception of an information packet from the robot the speed of each wheel is read to determine the current direction of movement. The readings from the corresponding sonars are then checked similarly to as described earlier. If a close obstacle is detected in the direction of movement the robot interface immediately sends a stop command to the robot.
Any program trying to control the robot might depend on knowledge about when a previous movement command has finished, that is when the robot has reached its destination. Because of this the robot interface maintains a Boolean variable indicating if the previous information packet reported a non zero velocity or not. If, at arrival of a new packet, the previous packet did report velocity and the current does not, a message is sent out from the IP socket notifying any other connected program that the robot is now static.
Finally, at every iteration of the infinite loop making up the robot interface, a special command with the sole purpose of keeping the connection alive is sent to the robot.

3.3.2 Radio Transceivers

3.3.2.1 WiFi Dongles

Each mobile router in the communication chain is equipped with two identical radio transceivers, one to communicate with each of its two neighbours. These are the off-the-shelf WiFi USB dongles of type WNA1100 from Netgear [9]. The wireless standard used by these transceivers is the widely spread IEEE 802.11n [10] that in our case operates at frequencies around 2.4 GHz.
There are two purposes of these transceivers. First, they allow collected measurements as well as position estimates to be shared between robots. This is important for the function of the control law. Second, in our research setting, they provide the ability to measure the received signal power and evaluate link performance. This means that they can act as representative placeholders and easily be replaced for most wireless technologies that could utilise our algorithms in the future.
The measurement collection strategy as well as the channel predictor assume that these two transceivers see identical channels and have identical positions at any time. This, however, is obviously not possible to achieve although effort has been put into coming close. The two transceivers are mounted on a wooden construction and connected to the computer using USB extension cables as in Fig. 3.14.

3.3.2.2 Network Configuration

To allow the measuring of the received signal power from each neighbour individually the hardware drivers require the system to operate in an hierarchical so called infrastructure mode. This implies that the left and the right side of each link as in Fig. 3.17 must take on different roles. Specifically, the left side of each link is configured to act as an access point. That is an owner of the network that broadcast its presence and network parameters and allow clients to connect to it. The right side of each link then acts as such a client and connects to its left hand network upon detection of it.

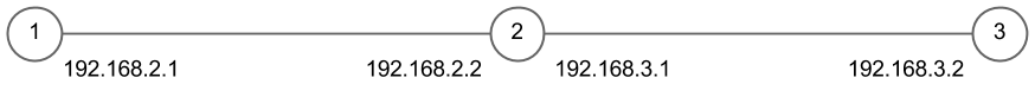


Figure 3.17: The network topology and IP addresses for robots 1, 2 and 3.

Running a WiFi transceiver as an access point is accomplished using the open source hostapd software available for Linux[11]. The client side is connected to the access point using the Network Manager shipped with Ubuntu Linux. Static IP addresses are assigned to each side according to Fig. 3.17.

3.3.2.3 Network Software Interface

Regarding the information exchange between routers, the transceivers are interfaced using the hardware drivers as well as the TCP/IP stack included in the Linux kernel. By Fig. 3.16 is provided a schematic view of the software modules involved in data exchange and their interconnects. Focusing on the data passing through the transceivers, there are in total four two-way sockets used by the main MATLAB session. For each neighbour there is one socket used for measurements communication. After collection of a measurement burst, the measured signal levels and the corresponding positions are sent using this socket. Also, it is used to receive such measurements from its neighbours. If measurements are received from one neighbour, they are written to memory as well as relayed to the other neighbour. For each neighbour there is also one socket used for position communication. When required by some algorithm of the main program, a position request is sent to the desired neighbour using this socket. The response, or a potential broadcast, including a robot id and position coordinates is also received in this way.
When sending IP traffic in computer networks of this kind there are mainly two types of transport layer protocols one can choose from. These are, somewhat simplified, reliable or unreliable protocols. Reliable protocols implement some technique to be able to confirm correct foreign reception of an information packet. These protocols could for example include, error checking, retransmission and rate control. Unreliable protocols however, work according to a best effort principle. They could include techniques that improve the probability of correct reception but they do not retransmit nor are they able to notify above layers if an error occurred.
In this thesis work the unreliable UDP transport layer protocol is used. The main reason for this is to avoid having to handle the establishment of a stateful connection with each neighbour. A temporary malfunction of one robot should preferably not

require any special action of the neighbours for the system to return to a functioning state. Secondly, even though modern reliable protocols are considered efficient they introduce some overhead that could reduce performance in systems of many robots and make them less scalable. To adequately discard the use of a reliable protocol one should consider not only its drawbacks but also why its benefits are less valuable in the current setting. Hence we argue that the majority of data communicated in the robot network is perishable. That is a received position renders any previously received position from that robot completely useless. Because of this, retransmission of data is not valuable and could be replaced by a relatively infrequent broadcast as a safeguard against a system freeze. Note that the above considers only positions and not the distribution of measurements. However, the fact that one lost measurement, at worst, only degrades the system to a pathloss predictor for a limited amount of time in combination with the overall low probability of packet loss leads us to confirm our choice of UDP.

3.3.2.4 Signal to Noise Ratio Measurement

One crucial function provided by the WiFi transceivers is the ability to measure the radio channel between neighbouring robots. The utilised hardware driver provides user space processes with an estimate of the received signal power from the access point it is currently connected to. Reading out signal power estimates from a transceiver acting as an access point is not supported. However, because of the reciprocal assumption of the radio channel, it suffice to measure the link in one of the two directions.
Making power readings accessible to the main MATLAB program is accomplished by modifying another piece of software included in Ubuntu Linux. The iwconfig is an open source software module originally designed to print various information, such as received signal power, reported by hardware drivers in terminal. Since all of the low level driver interaction is already in place, it is convenient to use this software as a foundation. Minor modifications to the source code of iwconfig makes it return the reading of the received signal power to the parent process instead of printing it in a terminal. The modified version is compiled and called from inside the MATLAB program.
Note that, at this stage, the measurement consists of a total received power estimate rather than the signal to noise ratio required by the algorithms. Power estimates are returned with unit dBm and our desired SNR is given, using unit Watt, by

$$\mathrm{SNR}_{\mathrm{dB}} = 10\log_{10}\left(\frac{P-N}{N}\right). \tag{3.19}$$

In the equation, P corresponds to the total measured power and the N term consists of the noise power which is currently unknown. There is no possibility to measure this noise with the available hardware so it has to be estimated using models. The main noise source in wireless systems is thermal noise which can be calculated by

$$N = KTB. \tag{3.20}$$

Here, K is Boltzmann's constant, T is the temperature surrounding the receiver given in Kelvin and B is the bandwidth of the radio channel. In our case $T =$

298.16 K and $B = 22\ 10^6$ Hz are used which corresponds to a surrounding temperature of 25 degrees Celsius and the 802.11n standard bandwidth of 22 MHz.
The conversion of the dBm value returned from iwconfig to a power in Watts is

$$P = 10^{\frac{P_{\mathrm{dBm}}}{10}}\ 10^{-3}. \tag{3.21}$$

Finally, (3.20) and (3.21) can be combined with (3.19) to find the SNR value required by the algorithms.

3.3.3 Radio-Based Positioning

There are countless applications that need sufficiently accurate positioning in indoor environments. Traditional positioning techniques that give satisfactory outdoor accuracy such as the Global Positioning System (GPS) face significant challenge to operate indoors or more generally in NLOS environments such as in narrow streets between buildings due to the signal properties. Thus, alternative solutions have already been given by researchers that employ existing radio technologies.
The main idea behind positioning lies in combining knowledge of ranges between the location of interest where the agent lies and fixed known locations called anchors with a multilateration technique. Among ranging techniques are measuring the time of arrival (TOA), the time difference of arrival (TDOA) and the two way time of arrival (TW-TOA). TOA requires synchronised clocks at every end-point, whereas in TDOA position is estimated by the anchors, thus requiring that the clocks of all anchors are synchronised. TW-TOA requires only one sufficiently accurate clock at the agent. All previous methods suffice with the premise that the times measured are corresponding to the time needed for a wave to propagate in the shortest possible path or in other words the LOS path.
Ultra-Wideband (UWB), unlike narrowband radio technologies, seems not to suffer severely from multipath enabling it to track the LOS path easily[12]. To elaborate, due to high bandwidth the pulses are shorter in time and path overlap occurs less. Hence, the individual paths can be resolved, allowing tracing of the LOS path. Thus, UWB is considered ideal for indoor positioning or more generally positioning in cluttered environments. For more information about UWB positioning one can refer to [13].

3.3.3.1 UWB Model

In order to operate safely our robots need to be aware of their locations. Moreover, it is a requirement to map the SNR measurements taken at the corresponding positions in the 2-D workspace. To cater for the above requirements, we employ the PulsON 410 (P410) platform of Time Domain that provides ranging information using the TW-TOA ranging technique [14]. To elaborate, each robotic agent carries a P410 and three more identical devices are placed in fixed known locations as anchors. These locations must have a good geometry to allow for more precise positioning. The agent sends a ranging request to each one of the anchors, whenever it wishes to identify its location. Then, the agent has three available pseudo-ranges, one for every anchor.

Each pseudo-range gives a hyperbola of possible locations around the corresponding anchor. The position of the agent is then the point where the hyperbolas meet as in Fig. 3.18.

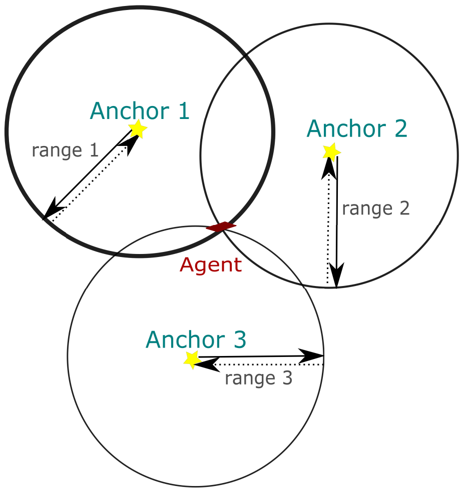


Figure 3.18: Position estimation via ranging.

Since the ranges provided are not precise and carry some noise(e.g. due to limitations of agent clock), mathematically the position estimate is given by solving the following least squares problem:

$$(\hat{x}, \hat{y}) = \min \sum_{i=1}^{3} (\text{range}_i - \sqrt{(x - x_{\text{anchor}_i})^2 + (y - y_{\text{anchor}_i})^2})^2, \tag{3.22}$$

where $i \in \{1, 2, 3\}$ stands for the anchor's id, range_i the pseudo-range measurement between the agent and anchor_i and (x, y) is the agent's unknown location. Note that the position estimate will always be noisy and can have centimetre accuracy.

3.3.3.2 Filtering the Noisy Position Estimate

An algorithm to improve a noisy estimate of one single measurement based on a series of previous measurements is known as Kalman filtering. This method is heavily used in various applications such as navigation and control of vehicles. More information about Kalman filtering can be found in [15]. Taking this factor into account, our function to estimate position is based on correcting the noisy UWB measurements. More precisely, our approach on steering the robot involves knowledge extracted from both the UWB measurements and the mechanical motors of our robotic agent. To accomplish this, we keep a finite history of the last position estimates taken. These positions can give an estimate of the current heading of the robot, which determines the angle formed between the front side of the robot and the reference axis. Note that we assume the y-axis to be the reference and positive angle is considered an angle formed clockwise in the unit circle starting from the reference axis. In order to estimate the current heading, we create the vectors of movement between successive positions in the history. To do so we maintain a history of the last turning angles that were taken by the robot. That is due to the fact that the

robot movement always involves first an on the spot turn (if needed) followed by a straight line trajectory. Adding up the vectors of movement to a total vector can give us the final current heading estimation by evaluating the angle it forms with the y-axis.
If $\mathbf{x}_i$ is the filtered position of the robot, $\mathbf{x}_{\mathrm{UWB}_i}$ is the noisy position estimate given by (3.22) and $\mathrm{d}\mathbf{x}_{\mathrm{control}}$ is the change in position according to the control signals sent to the robot since the last estimate was formed, then the following equation holds

$$\mathbf{x}_i = \alpha \mathbf{x}_{\mathrm{UWB}_i} + \beta(\mathbf{x}_{i-1} + \mathrm{d}\mathbf{x}_{\mathrm{control}_{i-1}}), \tag{3.23}$$

where α and β stand for the weights assigned to the UWB estimate and the position estimate according to the robot movement with $\alpha + \beta = 1$. The noise in the $\mathbf{x}_{\mathrm{UWB}}$ is assumed to have a zero mean. Potential biasing could be handled by a fixed compensation of the involved ranges.
The filtered position estimate is always given by a weighted sum of the UWB measurement taken after every move and the expected position according to the robot movement. The expected position according to the robot is calculated by adding to the previous position a vector of length equal to the expected forward movement and direction of the previous heading, rotated by the most recent saved turning angle in the history. Thus, it should be clear that each combined position estimate has some influence from all previous estimates. Note that the algorithm works only when it has initial guess for the history of positions and mechanical turns. This guess can be arbitrary and the algorithm will still converge. However, with better initial guess, the recursive algorithm stabilises faster.

3.3.3.3 Software Interface

The ranging software that is incorporated in our main program connects to the UWB radio via IP network. Each anchor is identified by a unique ID allowing the agent-attached UWB radio to send a ranging request to a specific anchor. Then the ranges measured are associated with the corresponding anchors. To do so we keep a vector with the IDs of the UWB devices used as anchors in a global configuration file that is the same for every robot. Each MATLAB main program sees a local configuration file containing the IP of the agent UWB connected to the robot. Then the ranges are the result of 20 averaged requests for every anchor. All the robot functions pertaining to movement, position estimation via solving a least square problem as well as the Kalman filtering are implemented in a class named Pioneer since it is related only to the physical implementation employing the Pioneer robot.

3.3.4 Program Execution

This section describes the execution of the main program. That is how the previously described parts interact and in what order they are executed.
The outermost script that is first executed when starting a robot sets up communication sockets, preallocates memory and initialises some variables. This is mainly done by creating objects of the types illustrated in Fig. 3.19. The central object is of a type called Robot the code for which can be found in the appendix B. A Robot

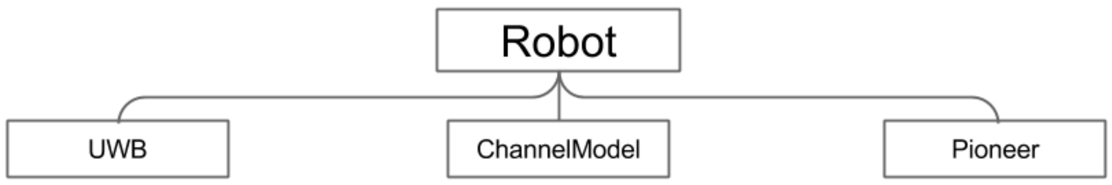


Figure 3.19: The software classes used on each robot and their relations.

object is responsible for the coordination of algorithms, communication and movements as well as for maintaining information regarding a robots state. Each Robot object creates one instance of a ChannelModel. This object contains all gathered information about the radio channel as well as methods to utilise this data. For example, it includes the 6-dimensional data structure holding all measurements as well as methods to add new measurements, send and receive measurements over the network, calculate channel parameters and the channel predictor. Moreover, each Robot object owns one object of type Pioneer. The Pioneer object holds all information regarding the robots current physical position and posture. It also includes a method that takes a desired coordinate as input and sends the appropriate commands to reach there to the robot interface. To estimate the position of the robot, the Pioneer object keeps a reference to one UWB object responsible for the connection with the UWB radio on board the platform.
After all objects have been created the main program enters an infinite loop. A somewhat simplified visualisation of this loop is provided in Fig. 3.20.

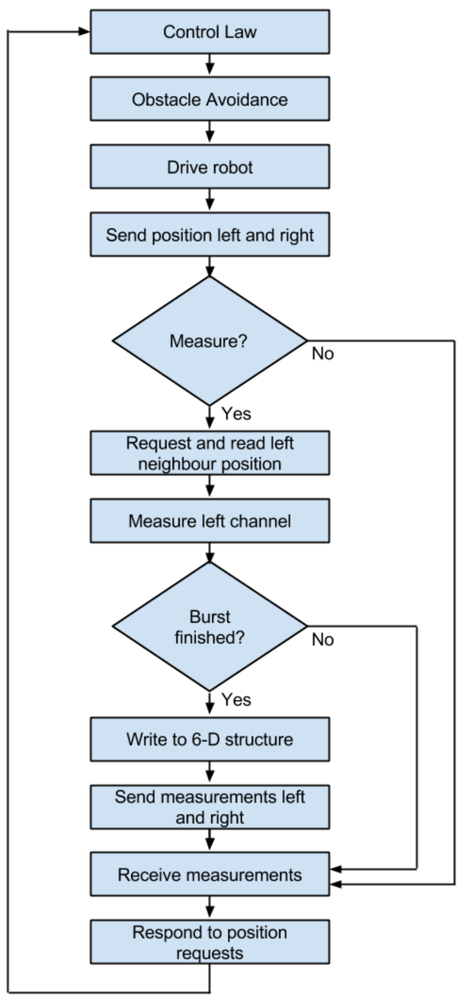


Figure 3.20: The program execution of a single physical robot

4

Results

The algorithms presented in the Methods chapter are used in a system that was both computer simulated and implemented on physical robots as described in sections 3.2 and 3.3 respectively. Various operation scenarios were examined and system behaviour was documented. The goal was to compare the performance of two approaches to steering the robots. One approach considers only the effect of pathloss on bit error rate whereas the other also aims towards optimally including knowledge about shadowing in the control law. Observed outcomes of these experiments are presented and commented in this chapter.

4.1 Channel Predictor Simulation

In this section we present the results of a channel simulation scenario, where we evaluate the performance of the prediction framework proposed by Y. Mostofi and Y. Yan. In Fig. 4.1 our simulated channel drawn by employing a fixed transmitter at position (0,0) and a receiver traversing through all possible locations in our finite window resolution is shown. It is easy to observe that it is a simulation of a fading channel accounting for small scale fading, namely the multipath. The parameters of the simulation model include transmission parameter $\alpha = 1000$, decorrelation distance of 3, pathloss exponent 2.32, shadowing variance 3.1607 and multipath variance 3.1607.
We keep a sufficiently large number of measurements drawn randomly by the previously presented channel. It is noteworthy that we follow the approach of formed hypercubes when storing these measurements and each measurement is stored to the hypercube it belongs. Then when employing the proposed prediction framework the values in neighbouring hypercubes are taken into account to estimate a mean predicted SNR with a variance adjusted by exploiting the shadowing correlation. Since we examine only the neighbouring hypercubes the computation time of the system is optimised. In Fig. 4.2 the predicted mean according to (2.27) is plotted. The closeness of the predicted to the real channel can be observed. This also proves that the data closeness property of our proposed data structure works as intended. Note that since the very fast fading multipath cannot be predicted, the resulting plot is smoother than the real channel's in Fig. 4.1.

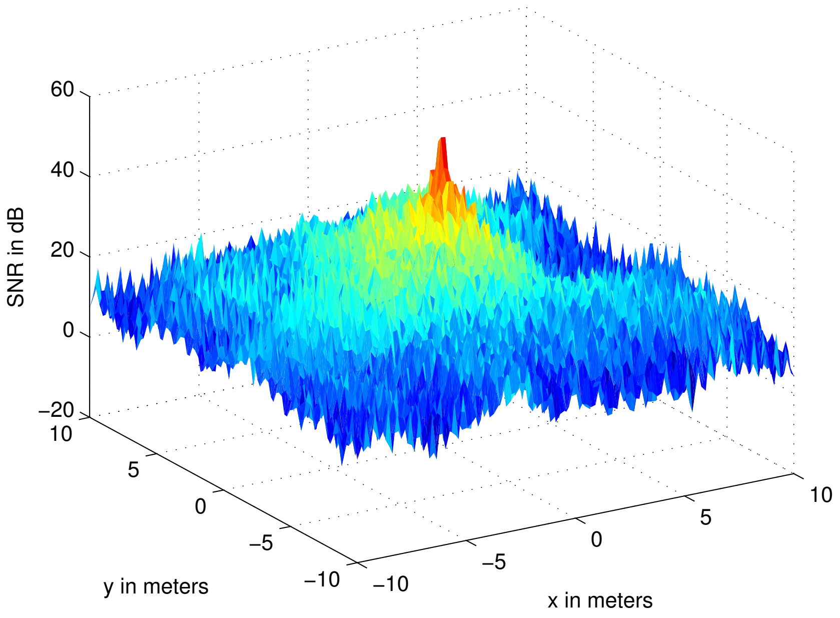


Figure 4.1: A simulated realistic channel with fixed transmitter at (0,0) and receiver traversing through the workspace.

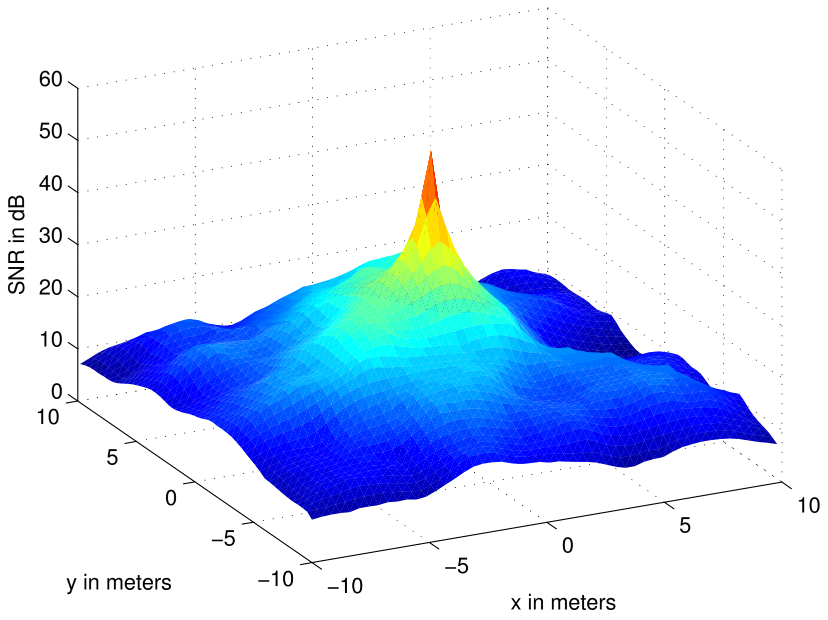


Figure 4.2: An estimation of the realistic channel seen in Fig. 4.1 using (2.27).

Finally, the variance according to the prediction framework of equation (2.28) is given in the two-dimensional Fig. 4.3. The variance shows how accurate the predicted SNR at every location is. The difference in magnitude of variance is due to the observed smoothing of fast fading meaning that multipath cannot be captured with certainty by the prediction framework.

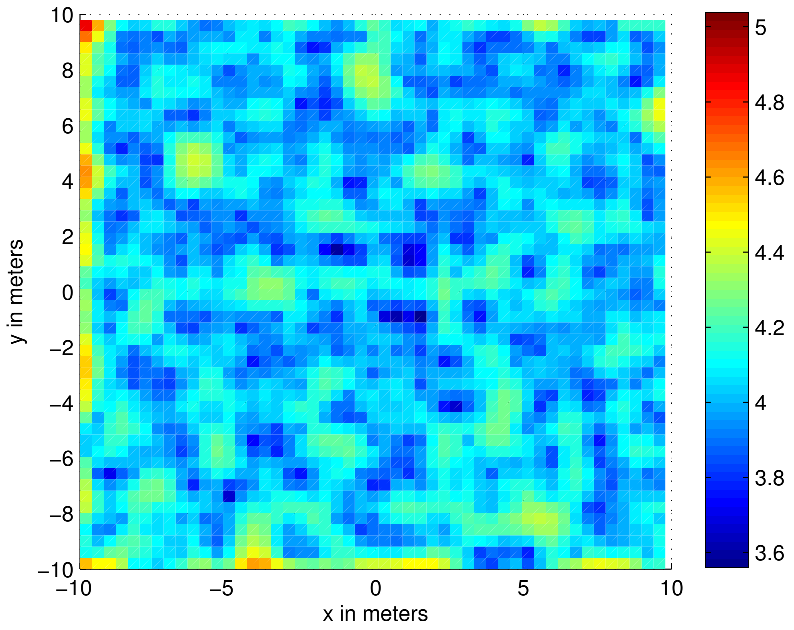


Figure 4.3: A quantification of the uncertainty of the predicted mean with fixed transmitter and traversing receiver. Dark blue tint signifies low variance and dark red higher.

4.2 System Simulation

A multi-robot system was simulated in accordance with Sec. 3.2. A description of the properties and initial states of the particular systems simulated as well as the outcome of these simulations is presented in this section.

4.2.1 Simulated Scenario

All simulated systems are configured as the chain of robots described and illustrated in Sec. 2.2.1. They assume static endpoints between which the probability of correct reception is to be maximised. Note that because of the assumption of Sec. 2.2.2.4 that the radio channel is reciprocal, the direction of communication is insignificant. Also all simulated systems consist of a total of four robots, that is two static endpoints and two mobile routers. The endpoints are located at positions $(-5, -5)$ and $(15, 22)$ respectively. The simulated channels have a shadowing variance of $\xi^2_{\text{dB}} = 10$, decorrelation distance $\eta = 3$ and a path loss exponent $n = 2.32$. The parameter Alpha that includes the transmit power and antenna gains is set to $\alpha_{\text{dB}} = 30$.
The two mobile robots are started in positions a few meters from their, according to the pathloss control law, final destinations of equal distances between all nodes.

Their data structures for storing measurements are empty at the start of the simulation but the correct channel parameters are hard coded in the algorithms.

4.2.2 Simulation Outcome

Outcomes from running the pathloss only control law and outcomes from running the measurement aided version are presented in separate sections. Comparison and comments are also given in the end.

4.2.2.1 Pathloss Only Control Law

The trajectories of the two mobile robots as they execute their program considering only pathloss are illustrated in Fig. 4.4 . It can be seen that robot 2 has stayed approximately on the centre point between the left endpoint and robot 3 during the final half of the simulation. Eventually, both mobile robots reach the positions that divide the communication chain into three links of equal distance.

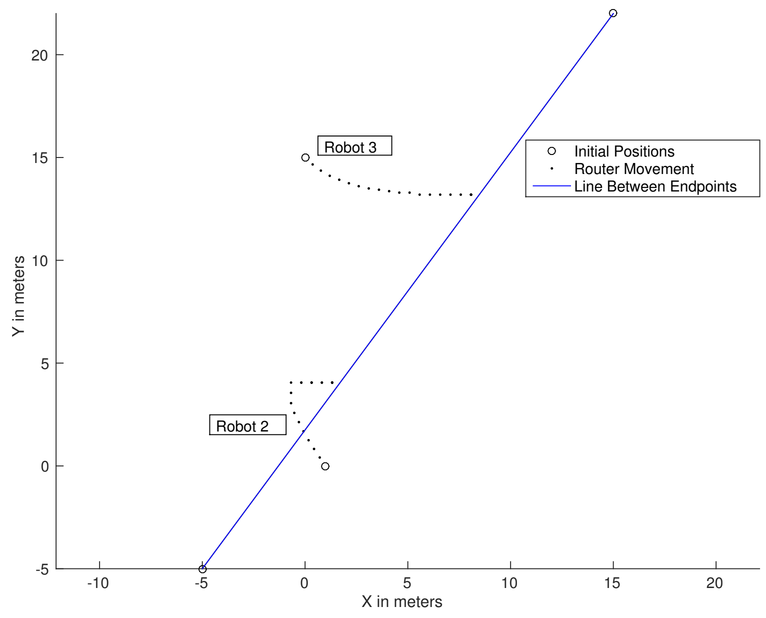


Figure 4.4: The trajectories of two mobile routers when steered by a pathloss control law

Note in Fig. 4.5 that the P_c according to pathloss is increased for every step the robots take. However, the more realistic random channel gives a much more dynamic and eventually worse result.

4.2.2.2 Measurement Aided Control Law

The trajectories of the two mobile robots as they execute their program considering both pathloss and measurements of shadowing are shown in Fig. 4.6.

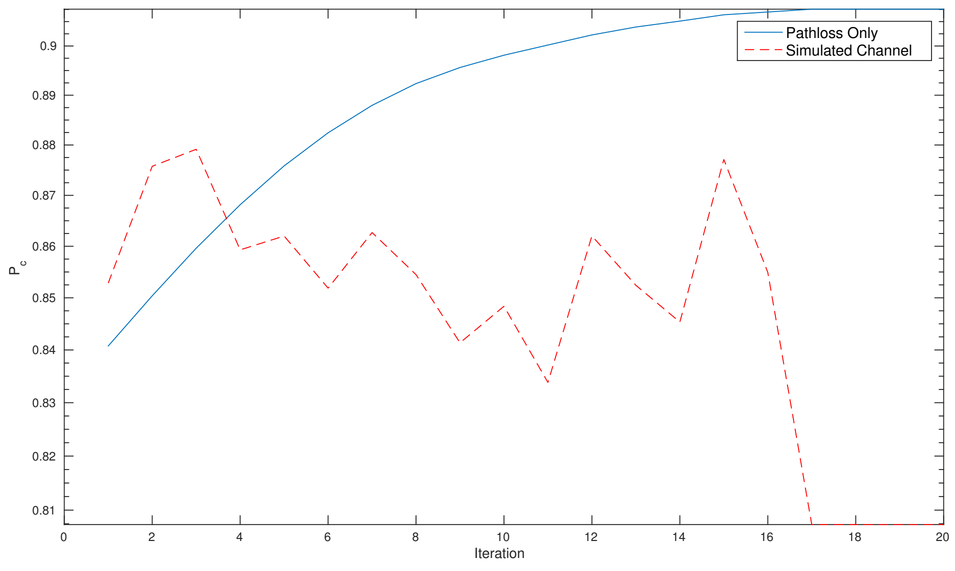


Figure 4.5: The lower bound on the probability of correct reception when two mobile routers are steered by a pathloss control law

The BER's according to the two models during the execution of the measurement aided control law are presented in Fig. 4.7.

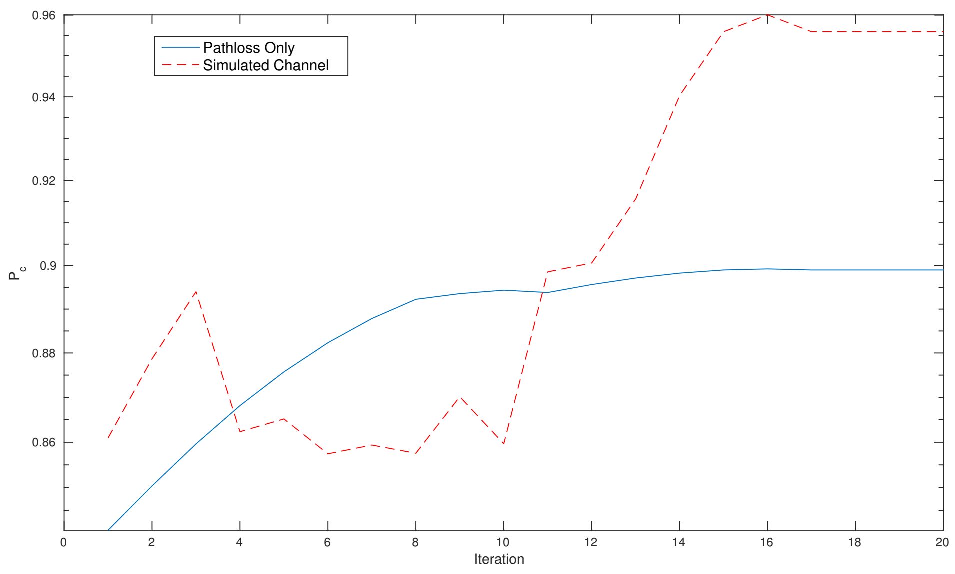


Figure 4.7: The lower bound on the probability of correct reception when two mobile routers are steered by a measurement aided pathloss and shadowing control law

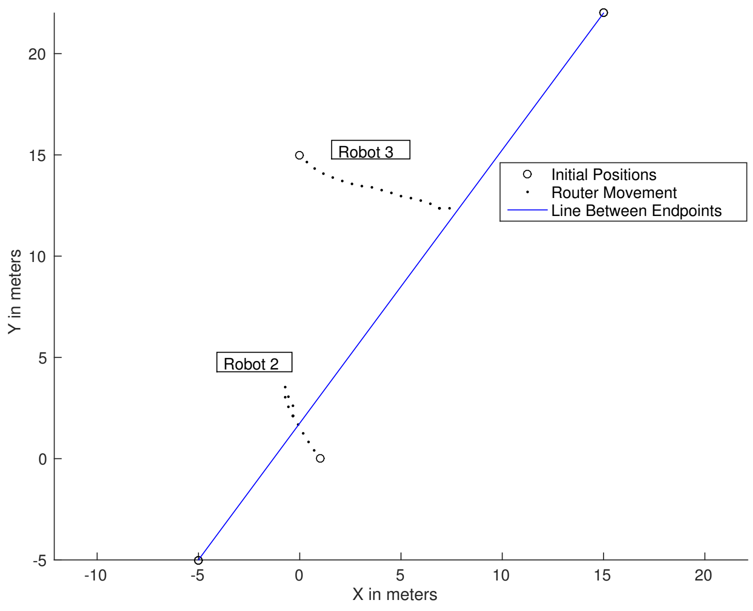


Figure 4.6: The trajectories of two mobile routers when steered by a measurement aided pathloss and shadowing control law

4.2.2.3 Comparison and Comments

As expected the two versions of control law steer the robots approximately identically in the beginning of each simulation. This behaviour is expected since no prior measurements are available to the robots and the measurement aided control law reverts back to a deterministic one. The small differences during the first few meters are due to rounding of the close values of the objective function. Theoretically and in many scenarios the functions give exactly identical results.
The deterministic control behaves as it should and keeps robot 2 close to the centre point between its neighbours while robot 3 is moving to its centre point.
In the measurements aided approach, after submission of the first measurement burst, robot two decides to return to a previously visited area. Robot three aligns itself accordingly with the line between robot two and the right endpoint.

4.3 Experimental Results

The developed system described in Sec. 3.3 was also operated in a physical environment. Specific parameters used, the environment of operation as well as observed outcomes of these experiments are presented here.

4.3.1 Experiment Setting and Method

Like in the simulated scenario, the experiment setting consists of a chain of robots with endpoints that do not automatically relocate themselves. However, to simplify performance evaluation and to limit the amount of space needed for experiments the

Left Endpoint		Right Endpoint		Centre Point	
x	**y**	**x**	**y**	**x**	**y**
20.90	1.71	2.50	0.64	11.70	1.18
15.68	3.33	2.50	0.64	9.90	1.99
4.28	2.22	2.50	0.64	3.39	1.43
12.86	0.86	2.50	0.64	7.68	0.75
25.80	3.50	2.50	0.64	14.15	2.07

Table 4.1: Table of the endpoints positions and centre points for the experiment, given in meters.

number of mobile robots is reduced to one. The site of experiments is a common indoor area at Chalmers University of Technology schematically illustrated in Fig. 4.8. On one side of the room the wall is made out of mainly wood and similar materials whereas on the opposite side there are glass windows from roof to ceiling.

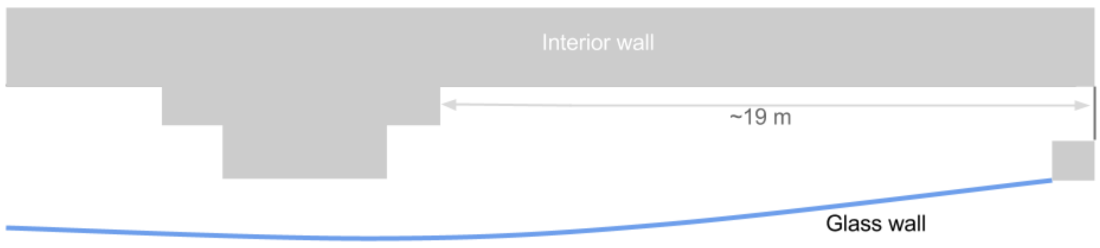


Figure 4.8: A schematic illustration of the area in which the physical experiments were performed

As in the simulations, the memory allocated for measurements is initially empty at system start. The initial position of the mobile robot is approximately $x = 18$m and $y = 1.4$m. Since, at this time, no measurements are available and the control law considers only pathloss, this is guaranteed to initiate a movement of the robot towards the centre point.
Since a goal of the experiments is to evaluate the performance of the measurement aided control law, a strategy to trigger measurement collection is used. Specifically, one of the two endpoints remains static throughout the experiment whereas the other is moved in a controlled manner. The moving endpoint is kept static as long as the robot is travelling towards its calculated maximum of the objective function. As soon as the router has stopped, or potentially moves around the same point due to noisy position estimates, the endpoint is moved. When changing the position of one endpoint the shape of the intermediate robots objective function is altered, causing it to relocate. A total of five positions presented in table 4.1 are visited by the moving endpoint during the experiment. The static position of the other endpoint as well as the positions of the resulting centre points are also presented.

4.3.2 Experiment Outcome

Outcomes from running the pathloss only control law and outcomes from running the measurement aided version are presented in separate sections. Comparison and comments are also given in the end.

4.3.2.1 Pathloss Only Control Law

In Fig. 4.9 the trajectory of the mobile router is shown when steered by the control law considering only pathloss. The trajectory spans a distance of approximately 14 meters. Careful investigation of this figure shows that the robot during some occasions moves seemingly randomly in proximity of a central point. These central points are close to the centre points of table 4.1.

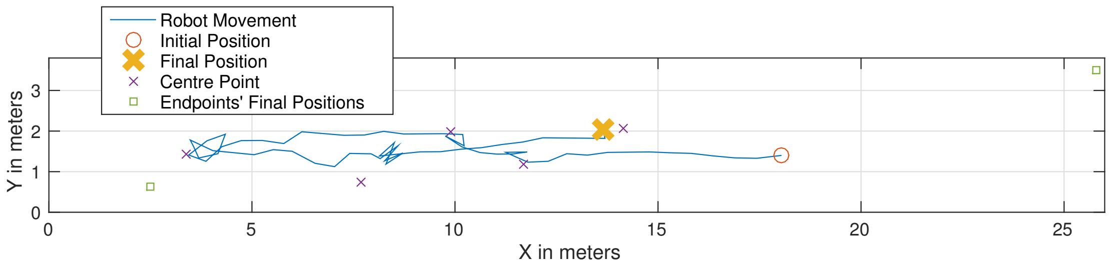


Figure 4.9: Trajectory of one physical robot under pathloss only control law.

In Fig. 4.10 the end to end bit error rate for each iteration of the robots program is shown, calculated using the measured signal power and SNR according to Sec. 3.3.2.4 and shown in Fig. 4.11. The figure shows a randomly fluctuating probability with a final value of 0.9806 when the router has converged to its final position. Bear in mind that one endpoint is moved several times during this experiment and finally to a position further away from the other endpoint than when the experiment started. This explains the higher BER in the end than in the beginning despite an optimal control.

4.3.2.2 Measurement Aided Control Law

The trajectory of the mobile router when steered by a measurement aided control law is shown in Fig. 4.12. The trajectory spans a total of approximately 10 meters. Similar to the observation in Sec. 4.3.2.1, there are points in the workspace around which the router stays for a few iterations. Note that these are not any of the centre points of table 4.1.
Our experiment is conducted in an environment similar to that used in [3]. Note that this paper presents a decorrelation distance of $\hat{\eta} = 1.2$m. Also note that during the experiment the robot drives by and collects measurements within one meter from the final centre point between endpoints. Combining this with the assumed decorrelation distance of more than one metre we consider the system to have measured the shadowing at the centre point.

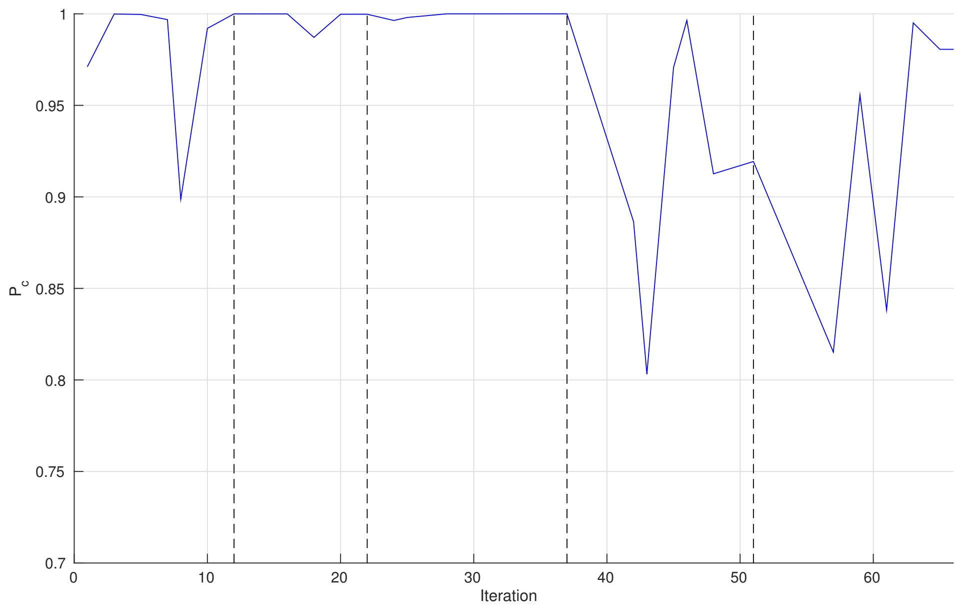


Figure 4.10: Calculated end to end probability of correct reception (P_c) per iteration of the mobile router under pathloss control. Dashed lines indicate times when left endpoint was moved.

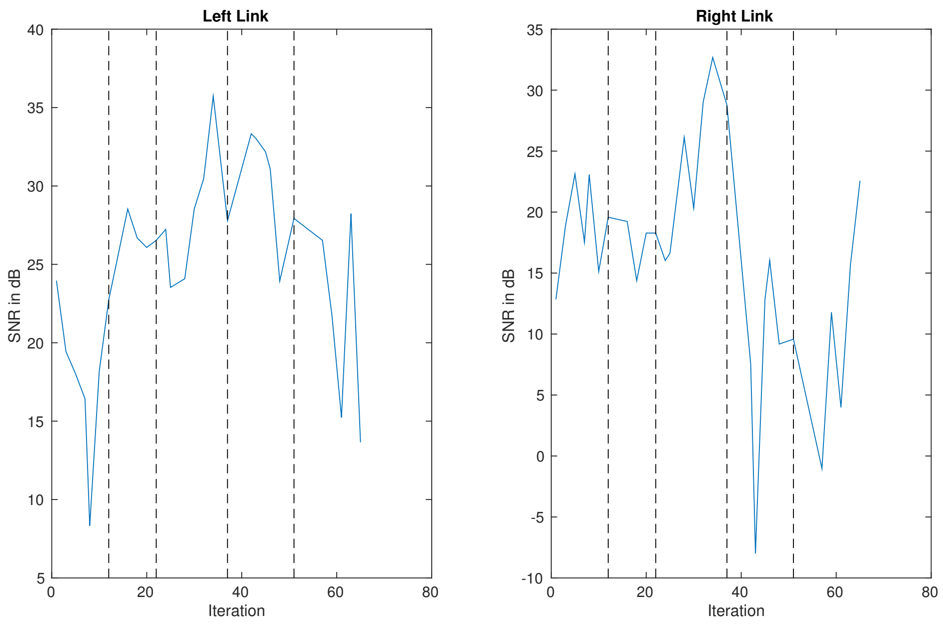


Figure 4.11: Estimated signal to noise ratio based on received power for the left and the right link during experiment with pathloss control. Dashed lines indicate times when left endpoint was moved.

The end to end bit error rate for each iteration of the robots program, calculated using the measured signal power and SNR according to Sec. 3.3.2.4 is shown in Fig.

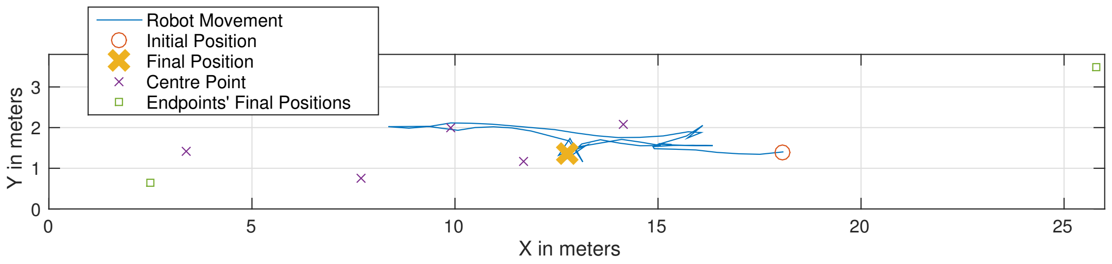


Figure 4.12: Trajectory of one physical robot when steered by a measurement aided control law.

4.13. A randomly fluctuating probability with a final bit error rate of less than 10^{-4} can be observed, when the router has converged to its final position. The calculated BER is based on the SNR measurements presented in Fig. 4.14.

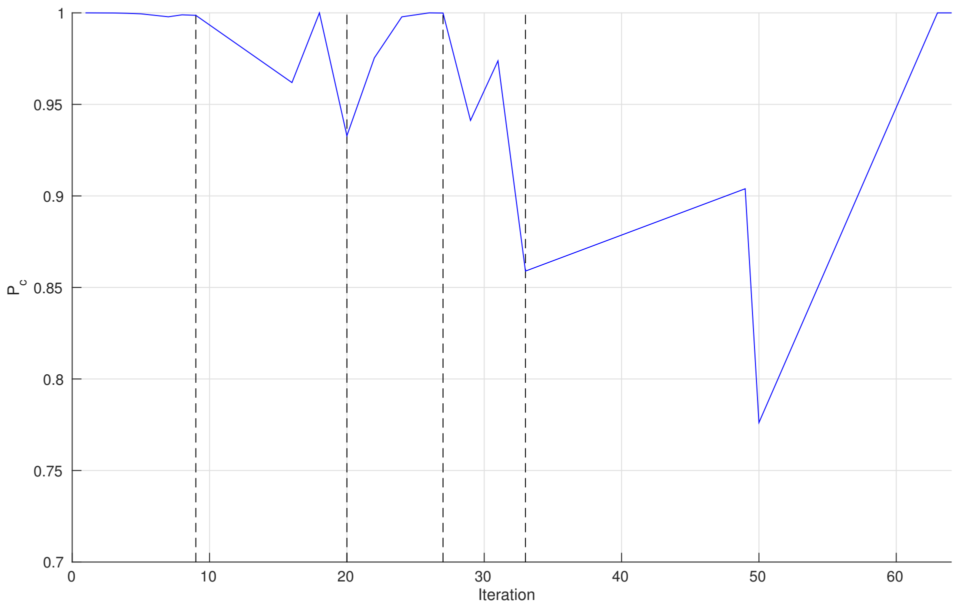


Figure 4.13: Calculated end to end probability of correct reception (P_c) per iteration of the mobile router under measurement aided control. Dashed lines indicate times when left endpoint was moved.

4.3.2.3 Comparison and Comments

The trajectories of the mobile robot for the two versions of control law are initially approximately equal. This is in accordance with theory since measurement aided control, without any available measurements, should steer the robot towards the centre point between its neighbours.
Minor variations between the two observed trajectories and calculated bit error rates can be explained by:

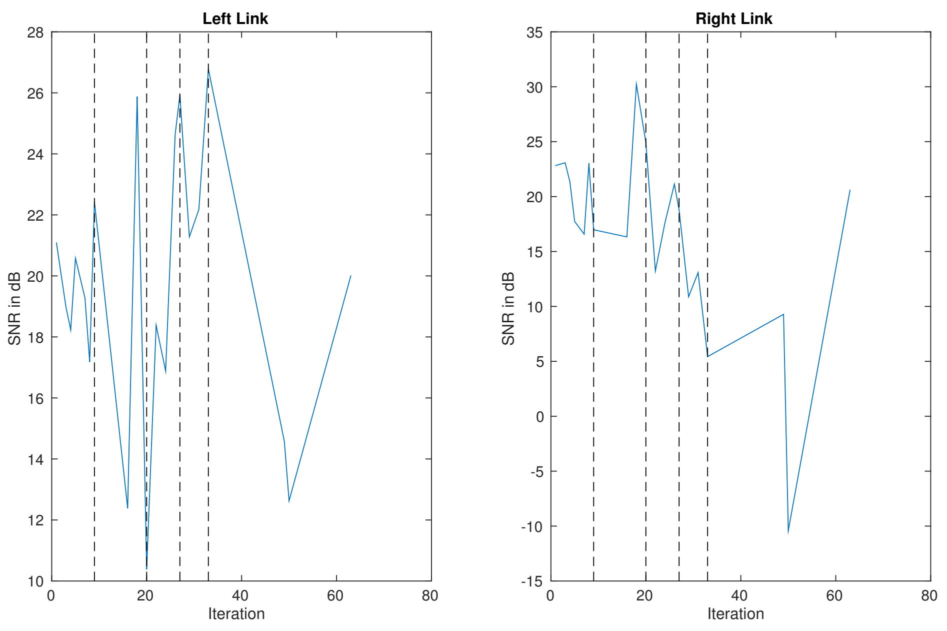


Figure 4.14: Estimated signal to noise ratio based on received power for the left and the right link during experiment with measurement aided control. Dashed lines indicate times when left endpoint was moved.

Variation in the initial position

Before each experiment the robot is positioned by hand close to the hard coded initial position. Also a heading of zero degrees is desirable. Deviations from these nominal values induces a nonzero convergence time for the estimates of position and heading described in Sec. 3.3.3.2.

Noisy UWB measurements

After the system is started the robot iteratively utilises its UWB radio to estimate its current position, and ultimately its heading. The UWB ranging requests of Sec. 3.3.3.3 obviously return measurements affected by noise that degrades the quality of the position estimate.

Non ideal robot control

The actual path the robot traverses is affected by potential imperfections in how the movement commands change the physical position of the robot. For example, a command to turn the robot 90 degrees never accomplish this exactly. A part of this will always have to be modelled as random whereas other parts probably can be derived from variations in surface, temperature et cetera.

Noisy power measurements

Even though the estimate of the received signal power consists of an average of many measurements, time varying interference from other radio transmitters may affect the estimate. Hence, the two experiments, even if assuming identical positions, could yield different calculated BER's.

The three first of the above directly affects the position estimates plotted in Fig. 4.9 but also the actual path traversed by the robot. Hence, the difference between the calculated bit error rates is a function of all of these imperfections. Acknowledge that due to the presence of multipath even small differences in position can severely affect the experienced bit error rate. Refer to [16] for a rigorous investigation of the impact of localisation errors on channel prediction and BER.

As expected the pathloss controlled robot visits the proximity of all the centre points of table 4.1 with approximately straight lines between them. With one of the centre points the robot performs significantly worse and misses it by around one metre. Most probably the LOS between the UWB antenna and one or more of the anchors was blocked by a WIFI transceiver during this time. As previously discussed one erroneous position estimate degrades also future estimates due to the Kalman filtering. The seemingly random movement around the centre points is again due to noisy position estimates. The technique of increasing search window resolution described in Sec. 3.1.4 succeeds in keeping the robot static for a few iterations at a time but does not completely solve the problem.

The more complex measurement aided control law initially steers the robot towards the first centre point. When it has reached an x-coordinate of approximately 15 meters it turns back towards a previously visited area. Note that the distance after which it turns corresponds to the distance needed to collect one measurement burst, that is $d_{\text{turn}} = d_{\text{step}} \cdot \text{MPB} = 0.5 \cdot 7 = 3.5\text{m}$. The interpretation is that up until the point of the first turn, the robot is steered without the aid of any measurements. At the submission of the first measurement burst, the control law finds a new optima at a previously visited location, a phenomenon further explained in the Discussion chapter and illustrated in Fig. 5.1. The left endpoint is eventually moved, forcing the intermediate router to relocate and measure new areas. This behaviour is repeated for some of the other centre points as well. Finally the robot uses the previous measurements to locate itself at a position approximately one metre from the final position of the previous control law. Since the environment in which the experiments are conducted have good radio wave propagation properties and the distances are relatively small, the theoretical bit error rate for the assumed 16 QAM modulation is practically zero. To still be able to visualise the performance over time, 30 dB of SNR is subtracted from all measurements before calculating and plotting the probability of correct reception. Such an operation is equal to running the system in a larger environment where each distance is increased by approximately 1 kilometre. The resulting end to end bit error rate is 0.02 and less than 10^{-4} for the deterministic and measurement aided control respectively.

5 Conclusion

This thesis addressed the optimisation of communication in robotic router formation focusing on performance in realistic environments. We do this by validating the solutions proposed by previous research using two different experimental methods. In the first method, we perform simulations to capture the performance of the proposed mathematical formulation of the optimisation problem compared to a simpler deterministic model. The second method required comprehensive work towards evaluating performance of the model in a physical implementation with real robots. The experiments gave some encouraging results that can be subject of further investigation. A by-product of this method is a functional distributed multi-robotic system. The developed algorithms achieve a substantial computational complexity reduction without sacrificing a lot in terms of communication performance. Furthermore, our implementation is able to operate in a decentralised manner, where each robot processes the collected data independently without the need of any central processing node. We conclude that a major impediment on real time performance optimisation is linked with the stochastic nature of the communication channel. In order to work, the prediction framework of the SNR needs a training set readily available so as to exploit information extracted by a sufficient number of measurements and get some insight of the channel parameters. In other words predicting the channel without any previously acquired knowledge is impossible. Our approach of collecting the training set is done on the fly. However, this approach could prove beneficial only in cases where both robots in each formed link move in a rather limited area.

5.1 Main Findings

The deterministic approach, be it the case where the control law of (2.18) is employed or the complete optimisation problem without any available measurements, demands some reasonable guess on the parameter α which affects the SNR and subsequently the probability of correct reception. During the experiments it was observed that setting high values would not allow the robot to move. The reader can easily confirm this claim by checking that the probability of correct reception approaches 100 percent.

The core of the algorithm steering the robot when accounting for path-loss and shadowing is based on the prediction model Sec. 2.2.2.3 established by Y. Mostofi. The accuracy of the predicted mean and variance for the expected SNR is a factor of how good the parameter estimation of Sec. 2.2.2.2 is. During the experiments with the robot we saw that a channel parameter estimation was a prerequisite to

actually achieve some logical steering for the robot. An important notion here is also the fact that a prediction for position x (assuming a fixed transmitter) with high average SNR drives the robot towards that particular point. On the contrary, the same optimisation problem in Sec. 2.2.2.3 shows that points with low variance are preferred.
As described in section 3.1.1 our approach lies in spreading the number of measurements into a 6-D structure. Having already discussed the channel predictor behaviour in section 3.1.3, one must be aware that the estimated shadowing mean and variance in the right part of (2.27) and (2.28) are computed based on the spatial correlation with only the measurements which belong to the same and the closest hypercubes. Similarly, when estimating the channel parameters employed in the left part of the same equations, one tries to fit unknowns to a least squares model accounting for the same measurements. It should feel intuitive that denser measurement bursts imply better channel fitting parameters but higher computational load.
In practice, when the measured SNR at close bursts of measurements is steadily high, the variance of shadowing and multipath is approximated to be low. This means that the left part of predicted SNR variance for a point x close to this good burst of measurements will be low and it will further be reduced by the estimated shadowing correlation. Steadily high local SNR measurements imply a high predicted mean as well. According to the discussion of the first paragraph, the robot would try to steer towards these measurements. This was also confirmed by the robot's movement during our experiments.
When reading mathematical proofs of optimality presented in [3] one could be misled to believe that the measurement aided approach always is superior to the pathloss one. We encountered, however, several practical situations, some of which are mentioned in the Results chapter, in which this assumption proved to be naive. Consider, for example, the one dimensional objective function of the left side of Fig. 5.1. Such an objective function is seen by a router around the centre point between its two neighbours before any measurements are taken. Since no measurements are involved, the values of the objective function depend only on default pathloss parameters and the distances to the two neighbours. That is, it is based on a prediction of the SNR for the left and right link with an overall constant and high variance. Furthermore, assume that this intermediate router moves from the left towards the maxima while at the same time measuring the channel. After a period of time, when the first measurements have been submitted and included in the SNR prediction, the objective function could resemble that of the right of Fig. 5.1. Measured shadowing causes the objective function to deviate from the previous. As soon as a part of the objective function in the newly measured area reaches the level of the centre peak, the robot will no longer move but consider itself to have reached the maximum. On average, that is what gives the highest average probability of correct reception, this is the correct prediction. However, in at least 25 percent of the cases, that is at least when the shadowing in both links are positive, the bit error rate would be even lower at the centre point. In any such case, the pathloss only control law would perform better. Even though the measurement aided control law as implemented does steer the robot to minimise the average bit error rate, it is tempting to visualise a control

law that is more exploratory and invests in the quality of future predictions.

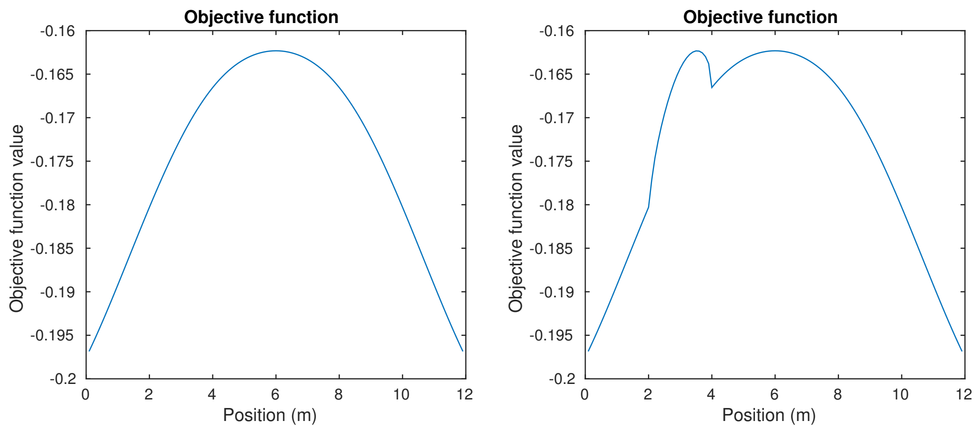


Figure 5.1: Examples of two objective functions in one dimension

In summary there is one fundamental limitation of the implemented approach in a real robotic setting. The objective function and the control law lead the robot to the optimum position according to the information already available. However, they do not consider that the choice of robot destination affects the quality of future predictions. More specifically, the ability of a robot to measure the channel is not considered in the previous work by Mostofi and Yan[3].

As has been emphasised in the previous paragraphs, the implemented algorithm tends to limit the robots' exploration of the workspace. There are, however, environments and tasks in which such an approach may be successful without modifications of the algorithms. A task that involves movement of one or both endpoints over large but finite areas will force the intermediate routers to move. If random movement of these endpoints is allowed to continue for a long period of time, enough measurements will be collected for the control law to perform consistently better than the one considering only pathloss. Examples of such applications are those where the system is enclosed in a finite area such as if it performed logistics tasks in a warehouse.

Another important application for teams of coordinated robots is search and rescue missions. In such an application robots are frequently entering previously unvisited areas where no measurements are available. A modification of the control law that forces robots to explore areas around centre points before discarding them as potential maxima could be beneficial. Another important aspect is that routers located far from the centre point between its neighbours risk finding themselves at relatively extreme distances from the new optimum if one neighbour relocates. Such aspects, like the average distance moved per relocation of a neighbour, must also be considered in time critical applications.

A final alternative for a functioning system that does not require algorithm modification is to manually measure the channel of the workspace and load this information into the system before it starts to perform its task. Manually measuring the channel can involve use of the robots or other equipment. This approach is obviously suitable in the warehouse but not in the search and rescue case.

5.2 Summary

In chapter 1 we establish the context of our work by briefly describing important aspects of radio channel theory and presenting related work. In chapter 2, we dig further into the related work and propose the optimistic distance as a factor in the exponential correlation of shadowing for moving transceivers based on the reciprocal property assumption. A contribution here is also the binned approach on the collected training set of measurements. Moreover, we present the algorithms employed to support our methods. Here, we take a different path for developing an obstacle avoidance algorithm. Finally, we present the overall functionality of both the simulator and the multi-robotic physical implementation. In chapter 3, we present the results obtained according to both simulation and physical experiment. In chapter 4 we discuss various aspects pertaining to the implementation.

5.3 Future Work

This thesis has been aimed at implementing previously proposed algorithms and creating a complete functioning system for bit error rate minimisation. Because of this rather ambitious goal, necessary limitations have been imposed on the work and the removal of these suggest some alternatives for future work. Moreover, due to the nature of an implementation project, many previously unconsidered problems have arisen that need further investigation .

- **General Graph:** One of the most obvious limitations of our work is that it assumes a fixed and simple communication graph. A generalisation of the optimisation problem to allow for any number of neighbours in a general graph would extend the set of possible applications drastically. Also, the integration of already mature methods of dynamically forming such graphs and routing information between nodes could be included in the research. The performance of the routing algorithms could then potentially further be improved by incorporation of channel measurements.

- **More accurate modelling of SNR:** Both deterministic and probabilistic approaches are plagued by one side-effect of the oversimplified SNR model when accounting for the path-loss component. More specifically, the parameters of antenna gain, channel frequency and transmit power are modelled as one fixed parameter. This does not take into account complex scenarios such as when the transmitter performs some power control. On the bright side, the work accomplished due to the generality of the model allows for further improvements.

- **Exploratory Control Law:** As mentioned in the Discussion chapter, the currently implemented control law tends to limit the robots exploration of the workspace. Our discussion points out some implications of this but completely lacks mathematical formality. Future research could preferably concretise the trade-off between convergence time and final performance when robots measure the channel during relocation. Based on this, one or more modified control laws could be presented.

- **Global Maximisation:** As mentioned in our Introduction, the control law each robot executes maximises only the performance of the to links connecting to that particular robot. That is, the neighbours are assumed to be static and only one variable, the position of the centre robot is varied when searching for a maxima. A global maximisation requires coordination and agreement regarding relocation of robots. Such research would improve the performance of the system.

- **Obstacle Aware Route Planning:** Most practical environments where robots might be deployed include some sort of obstacles. Our implementation includes a simple obstacle avoidance algorithm. However, it does only prevent the robot from hitting the obstacle. What the implementation lacks is an algorithm to calculate an obstacle free route from source to destination. Implementation of such an algorithm is absolutely necessary for a practical deployment.

- **Channel Aware Route Planning:** Apart from avoiding obstacles, there are other considerations that could be made when planning the relocation of a robot. In our current implementation the position that maximises objective function within a window is found and the robot is steered in a straight line towards this point. However, in realistic applications, not only the performance at the destination but also while the robot is moving might be of interest. Hence, performance metrics of different routes could be calculated and the optimal route could be selected.

Bibliography

[1] Y Uny Cao, Alex S Fukunaga, and Andrew Kahng. "Cooperative mobile robotics: Antecedents and directions". In: *Autonomous robots* 4.1 (1997), pp. 7–27.

[2] Balazs Janko et al. "A multiple robot solution for urban reconnaissance". In: *International Journal of Intelligent Defence Support Systems* 4.1 (2011), pp. 70–86.

[3] Yuan Yan and Yasamin Mostofi. "Robotic router formation in realistic communication environments". In: *Robotics, IEEE Transactions on* 28.4 (2012), pp. 810–827.

[4] Hazem Mohammed, Taha Khalaf, et al. "Optimal positioning of relay node in wireless cooperative communication networks". In: *Computer Engineering Conference (ICENCO), 2013 9th International*. IEEE. 2013, pp. 122–127.

[5] Ahmed S Ibrahim, Karim G Seddik, and KJ Ray Liu. "Improving connectivity via relays deployment in wireless sensor networks". In: *Global Telecommunications Conference, 2007. GLOBECOM'07. IEEE*. IEEE. 2007, pp. 1159–1163.

[6] Andrea Goldsmith. *Wireless communications*. Cambridge university press, 2005.

[7] Zhenyu Wang, Eustace K Tameh, and Andrew R Nix. "Joint shadowing process in urban peer-to-peer radio channels". In: *Vehicular Technology, IEEE Transactions on* 57.1 (2008), pp. 52–64.

[8] *Pioneer 3-DX*. Datasheet. Adept Technology, Inc. 2011.

[9] *Netgear WNA1100*. datasheet Accessed:2015-5-10.

[10] IEEE 802.11 n Working Group et al. "Draft amendment to standard for information technology-Telecommunications and information exchange between systems-Local and metropolitan area networks-Specific requirements-Part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications: Enhancements for higher throughput". In: *IEEE P802. 11n D* 1 (2006).

[11] *Linux Wireless About hostapd*. `https://wireless.wiki.kernel.org/en/users/documentation/hostapd#wireless_interface`. Accessed: 2015-06-30.

[12] Moe Z Win and Robert A Scholtz. "On the robustness of ultra-wide bandwidth signals in dense multipath environments". In: *IEEE Communications Letters* 2.2 (1998), pp. 51–53.

[13] Giovanni Bellusci. “Ultra-wideband ranging for low-complexity indoor positioning applications”. PhD thesis. 2011.

[14] *PulsON 410.* datasheet Accessed:2015-5-10.

[15] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents).* The MIT Press, 2005. ISBN: 0262201623.

[16] Yuan Yan and Yasamin Mostofi. “Impact of localization errors on wireless channel prediction in mobile robotic networks”. In: *Globecom Workshops (GC Wkshps), 2013 IEEE.* IEEE. 2013, pp. 1374–1379.

[17] Gilbert Strang. *Linear algebra and its applications.* Belmont, CA: Thomson, Brooks/Cole, 2006.

A

Code to generate shadowing according to sec. 3.2.1

```
classdef channel < handle
properties
    f1;
    f2;
    f3;
    f4;
    c;
    theta;
    n;
    a;
    dcorr;
    N;
    MP_grid;
end
methods
    function obj = channel(n, N, dcorr, std, MP_std)
        obj.n=n;             %Pathloss exponent
        obj.N=N;             %Number of terms
        obj.MP_grid=MP_std*randn([10 10]);      %Random multipath grid
        obj.dcorr=dcorr;         %Shadowing decorrelation distance
        obj.c=sqrt(2/N)*std;     %For correct shadowing power
        obj.a=log(2)/dcorr;      %Convenience constant

        %Base frequency set with a PSD according to the dcorr
        f=obj.a/(2*pi).*sqrt(1./(1-rand([N/2 1])).^2-1);


        %Generation of spatial frequencies and phases according to
        %section 3.2.1
        angle=2*pi*rand([N/2 1]);
        f1=f.*cos(angle);
        f2=f.*sin(angle);
        f=obj.a/(2*pi).*sqrt(1./(1-rand([N/2 1])).^2-1);
        angle=2*pi*rand([N/2 1]);
        f3=f.*cos(angle);
        f4=f.*sin(angle);
        matrix=[0 0 1 0;0 0 0 1;1 0 0 0;0 1 0 0];
        f1st=[f1 f2 f3 f4];
        f2nd=f1st*matrix;
        f=[f1st;f2nd];
```

```
            obj.f1=f(:,1);
            obj.f2=f(:,2);
            obj.f3=f(:,3);
            obj.f4=f(:,4);
            theta=2*pi*rand([N/2 1]);
            obj.theta=[theta;theta];

        end
        function [ s ] = evaluate(obj,x,y,u,v, alfa )
            %Method invoked to return the SNR between
            %any two points. Return value includes pathloss, shadowing and
            %multipath.
            shadowing=sum(obj.c*cos(2*pi*(obj.f1*x+obj.f2*y+...
                obj.f3*u+obj.f4*v)+obj.theta));

            %Simple multipath model as in section 3.2.1.1
            m=rem(round((x+u)*sign(x+u)*10),10)+1;
            n=rem(round((y+v)*sign(y+v)*10),10)+1;
            multipath=obj.MP_grid(m,n);
            s=10*log10(alfa)-10*obj.n*log10(sqrt((x-u)^2+(y-v)^2))+...
                shadowing+multipath; %in dB
        end
    end

end
```

B

The Robot MATLAB class

```
%In this file some lines in the "relocate" method handling the special case
%of this robot being a manually controlled endpoint have been removed for
%simplicity.
classdef robot < handle
    properties
       id;
       dist;
       model;
       MPB;
       alfa;
       c;
       n;
       last_burst_x;
       last_burst_y;
       measurement_index;
       burst_condition;
       burst_matrix_left;
       burst_matrix_right;
       real_channel;
       xmin;
       xmax;
       ymin;
       ymax;
       ids;
       left_sock;
       right_sock;
       left_pos;
       right_pos;
       UWB_radio;
       right_endpoint;
       left_endpoint;
       pioneer;
       init_x;
       init_y;
    end

    methods
        function obj=robot(id, measure_size, c, smallest_x, smallest_y,...
                burst_spacing, ids)

            obj.pioneer=Pioneer();
            obj.id=id;
```

```
%Creates ChannelModel that preallocates measurement memory
%of the preferred size.
obj.model=ChannelModel(measure_size(1), measure_size(2),...
    measure_size(3),measure_size(4),measure_size(5),...
    smallest_x, smallest_y, burst_spacing, id, ids);

obj.MPB=measure_size(5);    %Measurements per burst
obj.c=c;                    % 1.5/(M-1), M=constellation size
obj.last_burst_x=0;         %Coordinates of last burst
obj.last_burst_y=0;
obj.measurement_index=0;    %Current state, method "relocate"
obj.burst_condition=0;      %Distance to trigger new collection
obj.burst_matrix_left=zeros([obj.MPB 5]);
obj.burst_matrix_right=zeros([obj.MPB 5]);

%The four corners of the workspace
obj.xmin=smallest_x;
obj.xmax=smallest_x+measure_size(1)*burst_spacing;
obj.ymin=smallest_y;
obj.ymax=smallest_y+measure_size(2)*burst_spacing;

%Array with id's for all robots in the system
obj.ids=ids;

obj.left_pos=struct('x',0,'y',0);   %Left neighbour's position
obj.right_pos=struct('x',0,'y',0);  %Right neighbour's position

%Checks if this robot is the endpoint of our chain. In that ...
%case we don't communicate with left/right.
obj.left_endpoint=false;
obj.right_endpoint=false;
if id==obj.ids(1)                   %First in chain
    obj.left_endpoint=true;
end
if id==obj.ids(end)                 %Last in chain
    obj.right_endpoint=true;
end

global UWB_IP;
obj.UWB_radio=UWB(UWB_IP);          %Ultra Wideband Radio obj.

%Create sockets for position requests and responses
if ~obj.left_endpoint
    obj.left_sock=udp(get_ip(obj.id-1, 'right'), 2015,...
        'LocalPort', 2013, 'timeout', 300);
    fopen(obj.left_sock);
    readasync(obj.left_sock)
end
if ~obj.right_endpoint
    obj.right_sock=udp(get_ip(obj.id+1, 'left'), 2013,...
        'LocalPort', 2015, 'timeout', 300);
    fopen(obj.right_sock);
    readasync(obj.right_sock)
end
```

```
end

function update_channel_parameters(obj)
    %Updates channel parameters in the ChannelModel object "model"
    %based on measurements.
    obj.model.estimate_PL_parameters();
    obj.model.estimate_SH_parameters();
end

function relocate(obj, move_step, obstacles)
    %Evaluates the objective function and, if necessary, moves the
    %robot. Also measures the channel under some conditions.

    %Send position requests to the two neighbours
    request_new_pos=true;
    left_pos=obj.get_position(obj.id-1, request_new_pos);
    right_pos=obj.get_position(obj.id+1, request_new_pos);

    %Get own position from Pioneer object and evaluate the
    %objective function in a window around the robot.
    my_pos=obj.pioneer.get_position();
    [xm, ym]=findMax(left_pos.x, left_pos.y, my_pos(1), my_pos(2),...
        right_pos.x, right_pos.y, obj.c, obj.c, 20, obj.model,...
        obj.xmin, obj.xmax, obj.ymin, obj.ymax)
    change_x=xm-my_pos(1);
    change_y=ym-my_pos(2);

    %Make sure that any non-zero change is according to the
    %preferred step size.
    change=norm([change_x change_y]);
    if change>0
        step_scaler=move_step/change;
    else
        step_scaler=0;
    end

    %Check if the preliminary destination lies within an obstacle
    dest_x=my_pos(1)+step_scaler*change_x;
    dest_y=my_pos(2)+step_scaler*change_y;
    for i=1:length(obstacles)
        inside=inpolygon(dest_x, dest_y, obstacles(i).vertices_x,...
            obstacles(i).vertices_y);
        if(inside==1)
            disp('Preliminary destination inside an obstacle')
            normal=obstacles(i).normal;

            %Perform a change of basis and set one component to
            %at least zero.
            C=[normal(1) normal(2); normal(2) -normal(1)];
            new_coord=C*[change_x change_y]';
            k=new_coord(1);
            l=new_coord(2);
            k(k<0)=0;

            %Change back to standard basis
        standard_coord=C\[k l]';
        change_x=standard_coord(1);
        change_y=standard_coord(2);
    end
end

%Again, make sure that any non-zero change is according to the
%preferred step size.
change=norm([change_x change_y]);
if change>0
    step_scaler=move_step/change;
else
    step_scaler=0;
end
dest_x=my_pos(1)+step_scaler*change_x;
dest_y=my_pos(2)+step_scaler*change_y;

%Drive the robot to the desired destination
obj.pioneer.move_to(dest_x, dest_y);

pos=obj.pioneer.get_position();
fprintf('Estimated position; x:%d y:%d \n', pos(1), pos(2));


if obj.measurement_index==0
    %If we are in state zero we check for the distance
    %condition to be met before starting to measure.
    if (sqrt((obj.last_burst_x-pos(1))^2+(obj.last_burst_y-...
            pos(2))^2) > obj.burst_condition)
        obj.measurement_index=obj.measurement_index+1;
        obj.last_burst_x=pos(1);
        obj.last_burst_y=pos(2);
        if ~obj.left_endpoint
            obj.burst_matrix_left(obj.measurement_index,:)=...
                obj.measure_the_channel('left');
        end
    end
elseif obj.measurement_index==obj.MPB
    %Eventually we will have taken enough measurements and end
    %up here. We reset the state and then add the taken
    %measurements to the ChannelModel which also triggers
    %distribution to the neighbours.
    obj.measurement_index=0;
    if ~obj.left_endpoint
        obj.model.add_measurement_burst(obj.burst_matrix_left);
    end
    obj.update_channel_parameters();
else
    %We are in a measurement burst and should continue to
    %measure.
    obj.measurement_index=obj.measurement_index+1;
    if ~obj.left_endpoint
        obj.burst_matrix_left(obj.measurement_index,:)=...
            obj.measure_the_channel('left');
    end
end
```

```
end
function handle_requests(obj)
   %This method should be called frequently to respond to requests
   %from neigbours.
   if ~obj.left_endpoint
      fprintf('Looking for requests from the left\n');
      obj.read_buffer(obj.left_sock, 'left', false);
   end
   if ~obj.right_endpoint
      fprintf('Looking for requests from the right\n');
      obj.read_buffer(obj.right_sock, 'right', false);
   end
end
function measurement=measure_the_channel(obj, neighbour)
    %Reads out the received signal power from the transceiver
    %associated with the left or right neighbour. The readings are
    %averaged and used to estimate an SNR that the method returns.
    my_pos=obj.pioneer.get_position();
    global LEFT_INTERFACE RIGHT_INTERFACE SIGNAL_AVG;
    switch neighbour
        case 'left'
            neigh_pos=obj.get_position(obj.id-1, true);
            val=0;
            for i=1:SIGNAL_AVG
               val=val+signal_level(LEFT_INTERFACE);
               pause(0.05);
            end
            val=val/SIGNAL_AVG;
            val=level_to_SNR(val);
            fprintf('Averaged SNR:%.2f dB\n', val);
            measurement=[neigh_pos.x neigh_pos.y my_pos(1) ...
                my_pos(2) val];
        case 'right'
            neigh_pos=obj.get_position(obj.id+1, true);
            val=0;
            for i=1:SIGNAL_AVG
               val=val+signal_level(RIGHT_INTERFACE);
               pause(0.05);
            end
            val=val/SIGNAL_AVG;
            val=level_to_SNR(val);
            fprintf('Averaged SNR:%.2f dB\n', val);
            measurement=[neigh_pos.x neigh_pos.y my_pos(1) ...
                my_pos(2) val];
    end
end

function pos=get_position(obj, robot_id, force_UWB)
    %This method returns the position of either this robot or one
    %of its neighbours. The "force_UWB" flag determines whether the
    %last recorded position or a new UWB estimate should be
    %returned.
    global ANCHORS ANCHORS_X ANCHORS_Y;
    if robot_id==obj.id
        if force_UWB
```

```
                %20 ranging operations are performed for each anchor.
                %The averages are used with a least squares algorithm
                %to estimate this robot's position.
                range=zeros([1 length(ANCHORS)]);
                for i=1:length(ANCHORS)
                    range(i)=obj.UWB_radio.rangingTest(ANCHORS(i),...
                        20, false);
                end
                error=@(input) range'-sqrt((input(1)-ANCHORS_X').^2+...
                    (input(2)-ANCHORS_Y').^2);
                sol=lsqnonlin(error, [obj.x obj.y], [], [],...
                    optimoptions('lsqnonlin','Display', 'off'));
                pos.x=sol(1);
                pos.y=sol(2);
            else
                pos=obj.pioneer.get_position();
                pos.x=pos(1);
                pos.y=pos(2);
            end
        elseif robot_id==0
            %Special case for experiments
            pos.x=0;
            pos.y=0;
        elseif robot_id==obj.id-1
            %Return the latest received position of left neighbour or
            %potentially demand a new one
            obj.read_buffer(obj.left_sock, 'left', false)
            if force_UWB
                fprintf(2, 'Sending position request to the left\n');
                to_be_sent=[obj.id 9999];
                fwrite(obj.left_sock, to_be_sent, 'float');
                obj.read_buffer(obj.left_sock, 'left', true)
            end
            pos.x=obj.left_pos.x;
            pos.y=obj.left_pos.y;

        elseif robot_id==obj.id+1
            %Return the latest received position of right neighbour or
            %potentially demand a new one
            obj.read_buffer(obj.right_sock, 'right', false)
            if force_UWB
                fprintf(2, 'Sending position request to the right\n');
                to_be_sent=[obj.id 9999];
                fwrite(obj.right_sock, to_be_sent, 'float');
                obj.read_buffer(obj.right_sock, 'right', true)
            end
            pos.x=obj.right_pos.x;
            pos.y=obj.right_pos.y;
        end
    end


    function read_buffer(obj, sock, neigh, force_new)
        %For all non-empty sockets, this method always retreives the
        %latest position information in the input buffer and stores it
        %in a Robot object property using the "receive_pos" method.
```

```
            %If the "force_new" flag is true, the method first empties the
            %buffer and then blocks until a new position is received. This
            %way the topicality of the information is guaranteed.
            if ~isempty(sock)
                available=sock.bytesAvailable;
                while available>0
                    obj.receive_pos(sock, neigh);
                    available=sock.bytesAvailable;
                end
                if force_new
                    k=0;
                    while ~obj.receive_pos(sock, neigh)
                        if k>30
                            %If the method blocks for a long time, this
                            %robot's position must be broadcasted while
                            %waiting.
                            obj.broadcast_position();
                            k=0;
                        end
                        k=k+1;
                        pause(0.1);
                    end
                end
            end
        end


        function received_pos=receive_pos(obj, sock, neigh)
            %This method reads out any data available on the provided
            %socket. It returns true or false depending on whether there
            %was data to read or not. If the received data is a neighbours
            %position, this is stored in the Robot object properties. If
            %the received data is a position request, this robot's position
            %is returned to the querying neighbour.
            received_pos=false;
            if sock.bytesAvailable>1
                rec=fread(sock, 3, 'float');
                if length(rec)>1
                    id=rec(1);
                    if rec(2)==9999
                        fprintf(2,['Received position request from '...
                            'robot %d\n'], id);
                        obj.send_position(sock);
                    else
                        switch neigh
                            case 'left'
                                obj.left_pos.x=rec(2);
                                obj.left_pos.y=rec(3);
                                fprintf(2,['Received position x:%f y:%f '...
                                    'from robot %d\n'], obj.left_pos.x,...
                                    obj.left_pos.y, id);
                            case 'right'
                                obj.right_pos.x=rec(2);
                                obj.right_pos.y=rec(3);
                                fprintf(2,['Received position x:%f y:%f '...
                                    'from robot %d\n'], obj.right_pos.x,...
                                    obj.right_pos.y, id);
                        end
                        received_pos=true;
                    end
                end
            end
        end
        function send_position(obj, sock)
            %Sends a packet with this robot's id and position on the
            %provided socket.
            my_pos=obj.pioneer.get_position();
            to_be_sent=[obj.id my_pos(1) my_pos(2)];
            fwrite(sock, to_be_sent, 'float');
        end
        function broadcast_position(obj)
            %Sends this robot's position to all available neighbours.
            if ~obj.left_endpoint
                obj.send_position(obj.left_sock);
            end
            if ~obj.right_endpoint
               obj.send_position(obj.right_sock);
            end
        end
    end
end
```