Simultaneous Localization And Mapping for Wireless Networks

SLAMWiN

Distributed Real Time Systems 2006-2007

Master thesis by

Lars Jessen Roost
Michael Østergaard

Fall 2006 - Spring 2007
Aalborg University

Faculty of Engineering and Science

Aalborg University

Distributed Real time Systems

THEME:
Distributed Real Time Systems

SUBJECT:
Enhancing localization in indoor scenarios by utilizing Simultaneous Localization And Mapping

TITLE:
Simultaneous Localization And Mapping for Wireless Networks

PROJECT PERIOD:
Sep. 4th 2006 - Jun. 07th 2007

GROUP:
06gr938 / 07gr1038

GROUP MEMBERS:
Lars Jessen Roost
Michael Østergaard

SUPERVISORS:
João Figueras
Henrik Schiøler
Hans-Peter Schwefel

DUPLICATES: 7.

PAGES IN REPORT: 209.

PAGES IN APPENDIX: 41.

TOTAL OF PAGES: 250.

Abstract
This report covers the development of a approach for enhancing localization in indoor scenarios based on a wireless short-range communication technology by combining it with the Simultaneous Localization And Mapping (SLAM) approach known from robotics.
The developed method uses a Switching Extended Kalman Filter (SEKF) with five movement models to track MUs in an environment based on RSSI measurements. To handle non-LoS situations caused by walls in the environment, the approach monitors changes in the wireless channel caused by MUs moving from Line Of Sight (LOS) to non-LoS, or vice versa. Based on these changes, a knowledge base is iteratively build. This makes it possible to compensate for the negative effect on the wireless channel caused by walls in the environment.
The approach furthermore reduces the need for human interaction in the setup phase considerably, since the method designed to be applicable in already established wireless networks, e.g. Bluetooth or WLAN networks, and does not need knowledge regarding locations of APs and walls in the environment as these will be estimated in an iterative process. A by-product of the approach is a map of the environment which can be used for further enhancement of the localization accuracy, or for visual inspection of the environment estimate.
The project concludes that it is possible to build an almost PnP capable localization system that is able to estimate and take advantage the location of walls, the position of APs and the initial location of MUs based on very little human interaction in the setup phase.

Preface

This report documents the project work concerning simultaneous localization and mapping in wireless networks. The report is prepared at the *Faculty of Engineering and Science* at *Aalborg University* and is part of a Master thesis in Distributed Real Time Systems. The project spans from the 4st of September 2006 to the 7st of June 2007 and is based on the project proposal *Simultaneous Localization and Assisted Mapping.*

Instructions for reading

Some practical guidelines about how to read this report.

- Citations are referred to with square brackets e.g. SLAM[ea01]. The keyword in the square brackets can be looked up in the back of this report under "Bibliography".
- Footnotes will be used to deepen sentences e.g. This needs an additional comment[1].
- Abbreviations are written in full length the first time they are used e.g. Please Shorten This (PST) and a complete list of all abbreviations is available in the back of the report under "Abbreviations".
- All source materials are available on the enclosed disc found on the rear page of the report.

Report Structure

The report is structured as stated below. It is divided into three parts and an appendix.

Part I: Background: This part gives a short introduction to issues regarding localization in indoor scenarios. First different localization methods are described, after which a choice is made on which wireless technology that shall be used further in the project. The chosen technology is described and supplemented by description of the wireless channel with focus on propagation in indoor environments. Mobility model theory is introduced and related to localization. Finally the basic theory behind SLAM is described and the problem domain is defined.

[1]Footnotes deepen the sentence.

- **Part II: Analysis of SLAMWiN:** This part concerns the SLAMWiN approach where the overall approach and system architecture is described first. Hereafter the different components that make up the SLAMWiN system are described with respect to the chosen approach for the specific component, the implemented design and a performed evaluation.
- **Part III: Project evaluation:** This part evaluates the SLAMWiN approach and implementation on system level. It will to some extend work as an integration test of the SLAMWiN system and furthermore validate the chosen approach by illustrating how it can be applied in a scenario.
- **Part IV: Appendix:** This part contains the appendices. The appendices provide more detailed information on test cases, theory and a list of abbreviations.

The enclosed CD contains the program source-code, a copy of the report in portable document format (PDF), as well as measurement data and references.

The project group members:

Lars Jessen Roost

Michael Østergaard

Contents

Chapter 1

Introduction

This project will cover the integration of the Simultaneous Localization And Mapping (SLAM) technique known from robotics in the task of determining the position of Mobile Users (MU) in wireless networks. The purpose of integrating SLAM in the positioning process is to enhance the capabilities of the localization system.

1.1 Background

The diffusion of wireless mobile devices nowadays has reached a point where almost everyone carries such a device at almost any given time. This includes situations such as at work, in the supermarket, or on the family weekend trip to the zoo. In these situations, the mobile device can be used for purposes well beyond the ones, which it was originally designed for. Especially the introduction of wireless short-range technologies in the devices, such as Bluetooth or IEEE 802.11 WLAN, entails new possibilities. In this report a mobile device containing a wireless short-range technology will be refereed to as a Mobile User (MU).

The short-range wireless aspect of MUs lets them communicate with the environment without influencing or limiting the user's mobility and the pure presents of the devices can be enough for providing additional features to the user. Because a short-range wireless device has to communicate with some sort of Access Point (AP) or Base Station (BS) close by, it will always be possible to estimate the MU's location with some degree of precision. One can imagine a situation at work where an employee's computer locks as soon as he moves away from the workstation, or a situation at the supermarket where advertisements are streamed to any java-enabled phone in the vicinity of a special offer product.

However there are also more complex scenarios, where information about a MU being in the vicinity of an AP or BS is not precise enough. In these situations, more advanced techniques such as triangulation between several APs have to be used and furthermore, the propagation channel and the user mobility have to be considered. Such techniques and considerations can meet the need for higher accuracy in the process of determining the position of MUs.

An example of a more complex scenario can be seen in Figure 1.1. Here a MU is located in one of four rooms and has to be located through triangulation between the five AP. While triangulating a MU, one has to consider that a scenario may include walls. Such walls would affect the localization in two ways: On the one hand, the walls can have a negative effect on the precision of the localization, but on the other hand, the walls can also limit the mobility of the user so that it becomes easier to predict his movement.

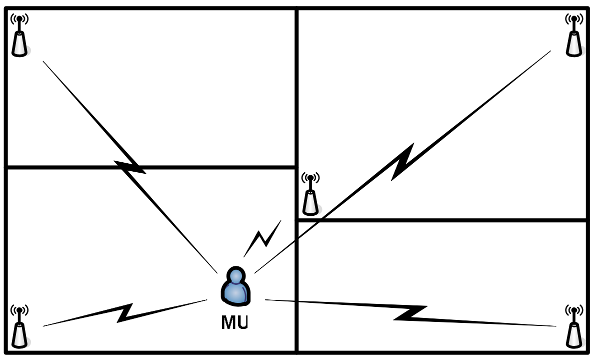


Figure 1.1: An example of a scenario where triangulation, user mobility and the propagation channel have to be considered in the process of locating a MU.

1.2 Motivation

The motivation for this project is to investigate a new method for the localization of MUs that can consider walls when triangulating the location of a MU without needing human interaction. In order to do this, a technique from the area of robotics is sought applied in the localization process. The technique is called Simultaneous Localization And Mapping (SLAM) and has aroused from research in the field of autonomous robots.

The problem SLAM seeks to solve is the situation where a mobile vehicle is placed in an unknown environment and then simultaneously has to generate a map of the environment while using the map to determine its position. In order to do this, the vehicle will choose a set of uniquely identifiable landmarks in the environment and start to move around while measuring its distance to the landmarks[ea01].

This process is shown in Figure 1.2 where a moving vehicle measures the distances to a set of chosen landmarks at different points in time. Since each measurement is only an estimation of the distance to a given landmark, the vehicle seeks to make a more precise estimation for each new measurement through filtering of the measurements in a filter.

The above described situation is not unlike a situation where a MU enters an unknown environment with a set of APs in communication range. Here the APs could be used as the unique landmarks. However, there are some differences between the two scenarios: The MU scenario does not provide the same possibilities for a choice of a convenient distance measurement technique such as lasers or sonar, which are developed for this kind

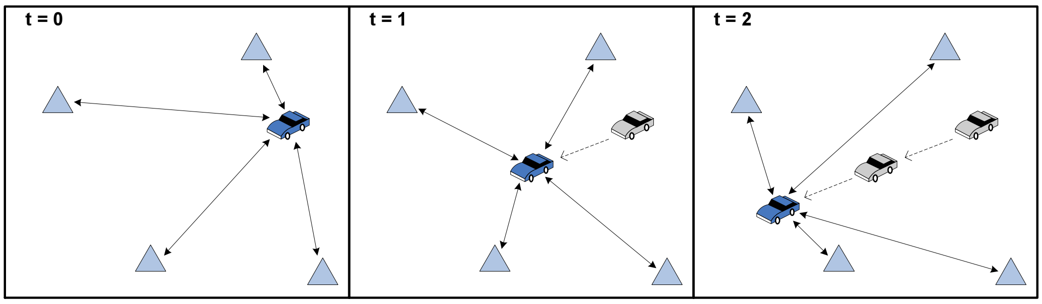


Figure 1.2: A moving vehicle determines the distance to a set of landmarks.

of tasks. The MU has to relay on the technology already present. This makes the MU scenario considerably more difficult in the way that all distance estimates are more affected by noise than in the traditional SLAM scenario. On the other hand, the use of AP's as landmarks provides the possibility for obtaining distance estimates between the landmarks. This additional information could be used to compensate for some of the lack of precision in the measurement process.

It is therefore interesting to investigate if it is possible to enhance localization of MUs in wireless scenarios using short-range technologies utilizing the SLAM approach.

1.3 Initial project description

In order to investigate if it is possible to enhance localization of MUs in wireless scenarios using short-range technologies utilizing the SLAM technique, this project will first have to discuss relevant scenarios and the prerequisites for these. Furthermore, a choice of communication technology has to be made and relevant parameters such as localization techniques, wireless propagation theory and mobility models have to be considered. Naturally, the project also will have to investigate the details of SLAM as it is applied in autonomous robots in order to understand the advantages and disadvantages of the technique. This combined background should provide a clearer definition of the problem domain and make it possible to formulate a final problem description.

The localization process is initially thought to have the input-output relation shown in Figure 1.3. The main input data to the localization process are untreated distance measurements obtained from all possible combinations of APs and the MU with ancillary noise. The distance measurements are continuously recorded and entered into the localization process. Additionally, different levels of prior knowledge and assumption about the scenario are at this point expected to be introduced in the localization process. The levels are introduced, because it initially is unknown wether the chosen localization approach is possible at all, if no prior knowledge exists and no assumptions can be made regarding the scenario in which the localization is performed. The levels will therefore go from situations where a broad prior knowledge is available and a large number of assumptions regarding the scenario can be made, to situations where no prior knowledge is available and only few

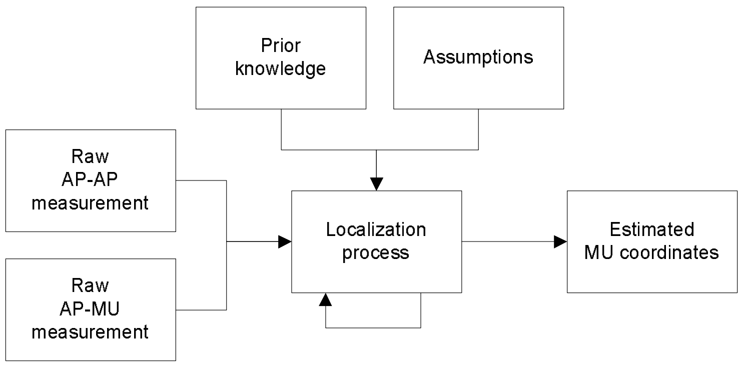


Figure 1.3: Simple input-output model of how the system initially is expected to be.

assumptions regarding the scenario are made.

The additional knowledge and the assumptions may be used in the initial phase of the localization process, or during the ongoing localization process. The additional information could be the precise location of the APs and assumptions could be made about the MU mobility behaviour or general architectonic characteristics of the buildings. In general, the prior knowledge and the assumptions should assist the localization process and provide a better starting point for a precise estimation of the MU location.

A more detailed model of the localization process could look like the model illustrated in Figure 1.4 where the boxes 'Localization process', 'Assumptions' and 'Prior knowledge' from Figure 1.3 are expanded. The localization process will very likely have recursive elements. In the shown figure, two elements are recursive: (1) the use of continuously updated and treated AP-AP measurements and (2) the map estimation process and the interaction between the map estimation, the mobility model and the continuously updated and treated AP-MU measurements. The details of the recursive elements will have to be analysed and described thoroughly in this project. In the following the different expected parts of the system model are shortly described.

Prior knowledge: This is known information that can be implemented in the system and be used to make a better map estimate and thereby improving the accuracy when estimating the position of MUs. This could be the position of the AP as either absolute coordinates or their coordinates in relation to each other.

Assumptions: These are assumptions made regarding the scenario the localization is performed in. An example could be assumptions about the typical MU behaviour or about room dimensions e.g. rooms are always larger than $1m^2$.

Raw AP-AP measurement: This is a measurement taken from an AP that is in contact with another AP. This measurement gives a raw estimate of the placement of the APs in relation to each other.

Raw AP-MU measurement: This is a measurement taken from AP that is in contact

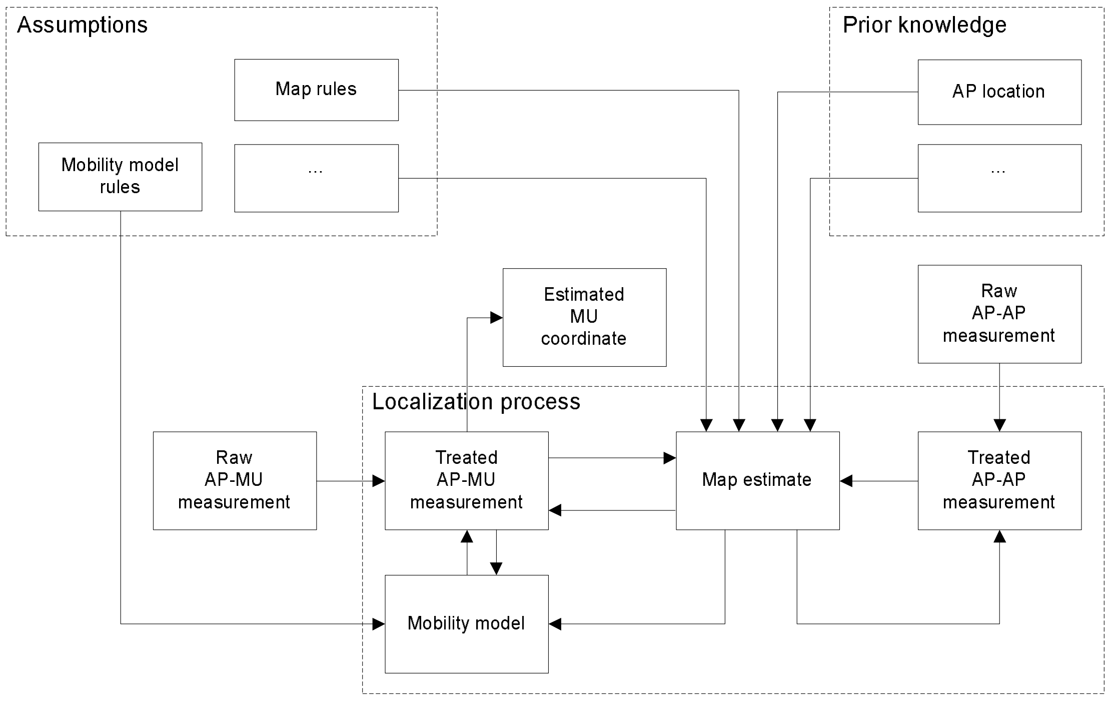


Figure 1.4: Detailed model of how the system initially is expected to be.

with a given MU. This measurement gives a raw estimates of the position of the MU in relation to the APs.

Treated AP-AP measurement: This is a raw AP-AP measurement where the ancillary noise is reduced, based on information from the map estimate and noise cancellation algorithms.

Treated AP-MU measurement: This is a raw AP-MU measurement where the ancillary noise is reduced, based on information from the mobility model, the map estimate and noise cancellation algorithms.

Map estimate: This is the virtual map of the environment e.g. placement of walls in the building. The map estimate is an iterative process that based on prior knowledge should develop a precise model of the scenario.

Mobility model: A model that describes the movement of a MU, e.g. the speed or usually used paths. The mobility model is an iterative process that is based on the map estimate and treated MU measurements should develop an increasingly precise model of the MU mobility behaviour.

Estimated MU coordinate: The coordinates of a given MU where the ancillary noise in reference to a raw AP-MU measurement is reduced.

Chapter 2

Scenarios

This chapter will discuss applications of localization services and describe a realistic scenario where localization could be used to provide a service to a MU. The purpose of the scenario is to clarify which issues, regarding localization that have to be examined before a method to enhance localization can be investigated. Furthermore this chapter is used to illustrate the applicability of a localization system.

2.1 Localization scenarios

As described in the introduction, the goal of this project is to develop a method to enhance localization of MUs in wireless scenarios using short-range technologies utilizing the SLAM technique. This subject is very interesting, because short-range technologies such as Bluetooth or IEEE 802.11 WLAN are already widely used and by using these technologies it is possible to perform localization in indoor scenarios as opposed to Global Positioning System (GPS).

GPS is a very reliable localization technology that makes it possible for people to pinpoint their geographic location all over the world. However, because localization with GPS is based on triangulation of signals from satellites, nowadays, handheld GPS devices cannot unconditionally be used indoor, due to the fact that they may not be powerful enough to receive signals through roofs or used underground in e.g. subways.

Because the goal of this project is to enhance localization of MUs in wireless scenarios using short-range technologies, already implemented wireless networks in e.g. airports, companies and a large number of public places can be utilized to perform localization. This reduces the cost of implementing a localization system considerately, making it more attractive to use. Since the localization method utilizes the SLAM technique, information about the building architecture and placement of APs should be superfluous in the localization process, because the technique relies solely on measurements taken by the MU as described shortly in the introduction. A system utilizing SLAM for localization will therefore theoretically be able to become a Plug and Play (PnP) system, needing almost no initial set up.

The following will describe a realistic scenario, which utilizes a localization service based on a short-range technology.

2.2 Project scenario

The scenario used in this project will take place in a large museum, e.g. the Louvre in Paris, where a short-range technology is implemented and accessible for visitors at any place in the museum. All visitors in the museum are thought to carry a mobile device with a headphone. The device is able to communicate with the local museum network by using the short-range technology. From a localization engine each MU knows its relative location to an arbitrary global reference point for the scenario. This knowledge, combined with context information about the museum, is used to provide special services for the visitors. These services are available through a menu system on the mobile device.

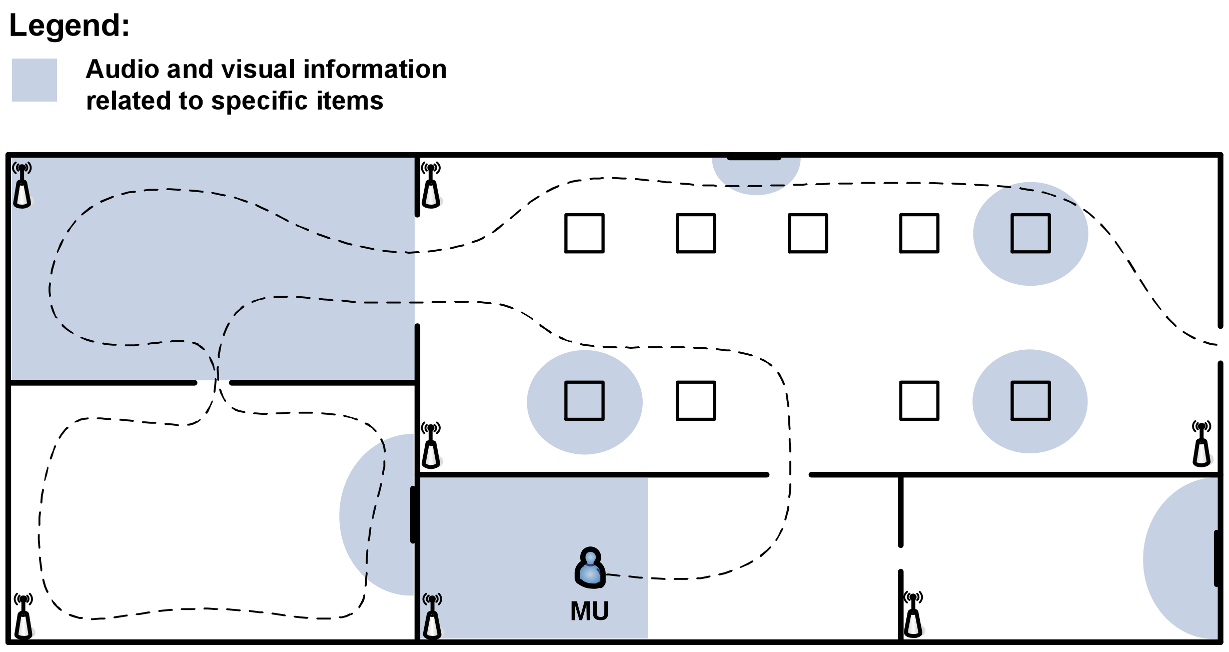


Figure 2.1: Illustration of the Project scenario.

The first service is a tour-guide service, where audio and visual information related to a specific item is presented to the visitor based on his location. This is illustrated in Figure 2.1. When the visitor enters a new room or is located in the immediate vicinity of a specific item, relevant information, e.g. the history of the item, is presented audible or visually to the visitor.

A second service is a navigation service, where a search mechanism is provided through the mobile device, which makes it possible for the visitors to search for a specific place or item in the museum. This could be a painting, e.g. Mona Lisa, a cafeteria or the exit doors. By selecting the desired item or place the visitor would then be lead to it by the mobile device. Since a localization engine using a SLAM like approach will generate a virtual map of the building, the shortest path can easily be computed. This navigation service could also used if an emergency situation should occur. Then all mobile devices are notified and visitors will be guided to the closest exit.

A system as the one described above clearly consists of more than a localization engine. Before this scenario can take place it is necessary for the system to obtain information about the location of items and places and context information associated to each of them. Obtaining this information is not part of the localization engine. This information could be provided by a central context information server working together the localization engine. On this server a person, carrying a mobile device, could previously have stored all relevant locations by placing the device in the immediate vicinity of an specific item and telling the sever which item it is and which context to assign to it. This procedure, repeated for all items and places in the museum, will not be treated in this project as it focuses on the localization engine.

Based on the described scenario different issues have to be considered when analyzing and developing a method to enhance localization of MUs in wireless scenarios using SLAM.

Localization: Pinpointing the location of a MU using a short-range technology requires some sort of a localization method e.g. triangulation. Relevant localization methods will have to be investigated.

Wireless technology: Before the localization method can be implemented it is necessary first to examine the properties of the short-range technology that will be used.

Wireless channels: Because the scenario is taken place indoor and short-range technology is used e.g. walls, people and furniture in the building might have an effect on the localization precision. Therefore it is necessary to examine what impact the environment has on the short-range technology.

Mobility models: To enhance the localization, information about human movement behaviour can be considered. Because humans usually use the same paths when walking around in a building, this information can be used to predict the movement of a person and thereby making more precise localization estimation.

SLAM: Because the initial problem is to enhance localization of MUs in wireless scenarios utilizing the SLAM technique, it is necessary to understand SLAM and examine which components it consists of.

In the following chapters the above mentioned issues will be examined with focus on the subject of this project. Together, these chapters will constitute the background of this report and form the basis knowledge before developing a method to enhance localization by utilizing SLAM.

Part I

Background

Chapter 3

Localization methods

This chapter will describe some commonly used localization methods. The aim with the chapter is to get an overview of the different methods that can be used to make localization and find the most appropriate localization method for this project.

3.1 Introduction to localization

The chapter will in general be based on [Fig04]. Supplementary references are used where needed.

Localization is the determination of the position of an object. Information about the position, called location information, can be represented in infinitely many ways depending on which reference and system of coordinates that is used. Without a reference or knowing the used system of coordinates it makes no sense to talk about a position. E.g. 1 unit to the north will not reveal any exact position. However, 1 km to the north of the Eiffel tower is a position that can be found by knowing the reference point and the system of coordinates that is used. When describing the location of non-static objects it is further necessary also to consider the time, because the location of the object can change and therefore is time dependent.

Different methods exists to perform localization. For all localization methods applies that some sort of a raw sensor values are measured and then converted to a location. Similar to the OSI-layer model for network protocol design a six-layer localization stack can be made[JHB02]. This general framework for localization systems is illustrated in Figure 3.1 and each layer of the stack is described in the following.

Sensors: Measurements from physical localization sensors.

Measurements: Algorithms to transcribe raw sensor data into the canonical measurement types. The output could be a stream of e.g distance, angle or proximity.

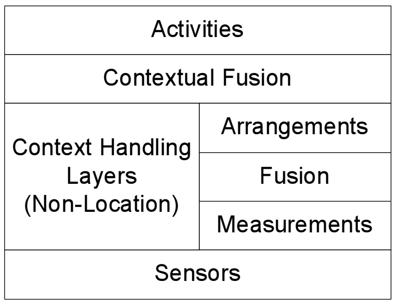


Figure 3.1: This figure shows a six-layer localization stack.

Fusion: Convert the stream into a time-stamped probabilistic representation of the positions and orientations of objects. This could include information about e.g object speed, acceleration, positional history, object names or unique IDs.

Arrangements: Relate positions of objects to a map and to the position of other objects.

Contextual Fusion: Add semantics of places and of users' situations or routines.

Activities: Infer activities of users based on location and other contextual information.

3.2 Localization methods

Localization methods can be divided into three groups, as illustrated in Figure 3.2: proximity, environment analysis and radio localization. In each group different localization methods exists. In the following each group with their respective localization method will be described. Each method will be evaluated with respect to be used to enhance localization of MUs in wireless scenarios using short-range technologies utilizing the SLAM method.

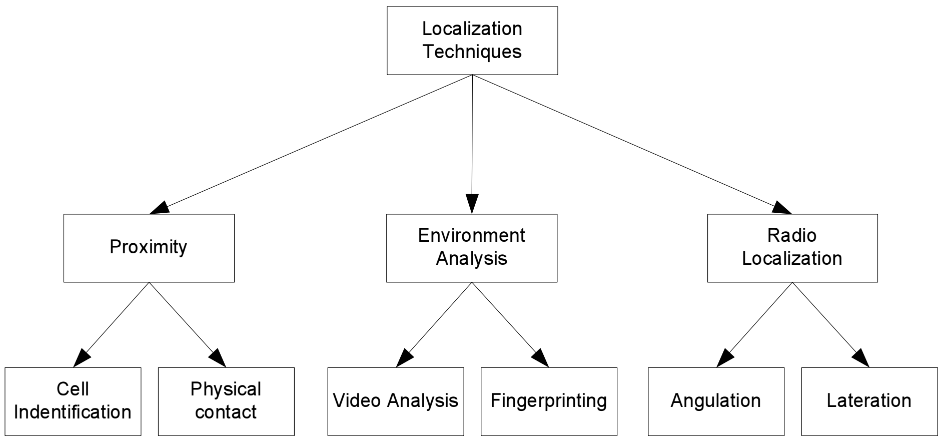


Figure 3.2: This figure shows a division of localization methods.

Proximity

Proximity is a localization method that uses different kind of sensors to registers if a MU is near by. Two approaches exists: cell identification and physical contact.

Cell identification

Cell identification is a very simple location method that only utilizes the information about which AP a MU is connected to. If the coverage range for an AP is known and a MU is connected to it, the MU has to be located within the coverage area as illustrated in Figure 3.3. The accuracy of the determined location depends on the AP coverage range. A larger coverage area will therefor naturally result in a larger inaccuracy of the determined location. For short-range technologies such as Bluetooth Class 2 the coverage area can be as large as 10 meters in radius which therefore will be the worse case localization accuracy. For WLAN the accuracy would be even worse as it has a larger range.

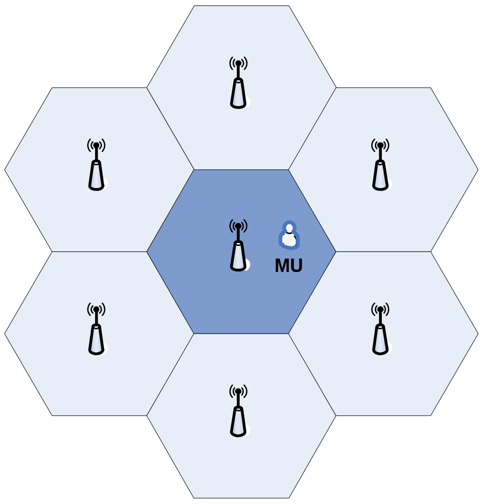


Figure 3.3: This figure shows localization with cell identification. The MU is located to be within the dark blue hexagon.

Physical contact

Physical contact is a localization method that, as the name implies, uses physical contact to perform localization. This could be switches in the floor that could be activated when a person is placed on it and hereby reveal the position. However, by only using this method it becomes difficult to distinguish between multiple MUs if their paths cross.

Environment analysis

Environment analysis is a localization method that utilizes the fact that an environment will look different from different angles to perform localization. Two approaches exists:

video analysis and fingerprinting.

Video analysis

The idea of video or radar analysis is to continuously take pictures of the environment from a MUs point of view. The process is then to extract features from the picture e.g. a mountain or a house and compare these features with information stored in a database. The database should contain a map of the environment with information about the location of selected features and by comparing these with the features obtained from the taken picture, the location of the MU is estimated. The prerequisite for using this method is therefore to have some sort of picture of the environment from a given position and a database with a map of the environment.

Fingerprinting

Fingerprinting is a similar method to the video or radar analysis. The idea with this method is that a MU takes e.g. Received Signal Strength Indicator (RSSI) measurements from all accessible APs. Because the RSSI value from each AP will vary depending of the MUs location, this can be used as a fingerprint for a given location. By comparing these RSSI values with information in a database, the location can be determined. However, before this method can be used it is necessary to take RSSI measurements from the environment, coupled with information about the location where the measurements are taken and store them into a database. The complete procedure is illustrated in Figure 3.4 with the setup phase to the left and the localization based on fingerprinting to the right.

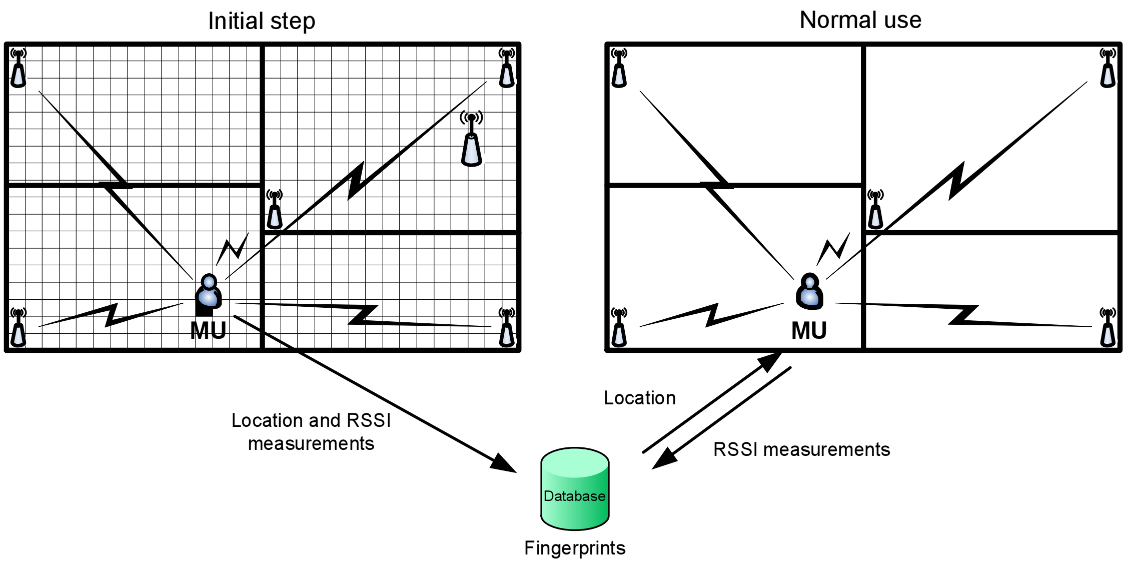


Figure 3.4: This figure illustrated how the fingerprinting method works.

Radio localization

Radio localization is a method that utilizes the radio channel to make localization. Several approaches to radio localization exists and these are described in the following.

Angle of Arrival (AoA)

AoA is a method that determines the direction of an incoming radio signal. By using an antenna array the radio channel direction can be calculated by measuring the differences in time between the signal is received on each individual element of the antenna array. By using at least two antenna arrays this method can be used for localization, because the intersection area will be the position of the MU. This is illustrated in Figure 3.5. The prerequisite for using this technology is to have special antennas, which are not used in normal operation of most wireless networks, and knowing the location of these. Furthermore this method requires to have line of sight between the MU and the antenna arrays, because multipath propagation and shadowing will introduces a big lack of precision.

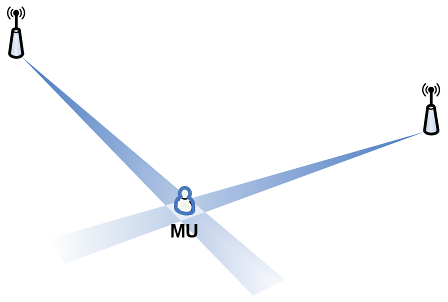


Figure 3.5: This figure shows localization based on AoA.

Triangulation based on RSSI measurements

This localization method is based on the knowledge that the electromagnetic strength of a radio signal is proportional to $\frac{1}{d^2}$ in free-space propagation, where d is the distance from the source. By measuring the received signal strength from an AP and by knowing the AP's transmission signal strength, this decrease in signal strength can be used to calculate a distance from the AP to the MU. By knowing the distance to one AP and nothing more, the MU may now be located in a sphere around the AP. By measuring the signal strength from additional APs and also knowing their location, the MU possible locations are reduced to the number of intersections of the spheres around each AP. For each additional AP that is introduced, the location of the MU is reduced with one degree of freedom. This means that locating the MU in a plane can be determined by measuring the signal strengths from at least three AP. This is illustrated in Figure 3.6.

The MUs location can be estimated in the following way[Jør06].

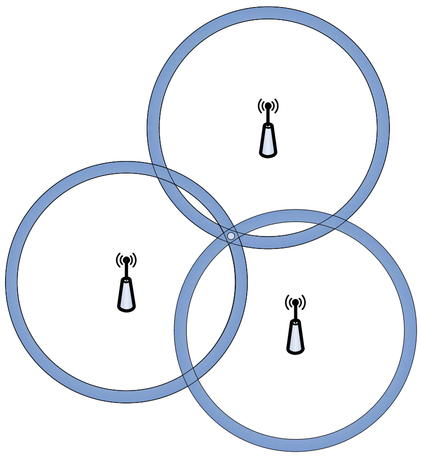


Figure 3.6: This figure shows a triangulation with three APs.

Let (x_i, y_i) denote the location of the ith AP and let $(\hat{x}, \hat{y})$ be the estimated location of the MU. d_i is the distance from the ith AP to the MU. By using Pythagoras' theorem Equation 3.1 can be expressed, which gives a set of nonlinear equations.

$$d_i^2 = (x_i - \hat{x})^2 + (y_i - \hat{y})^2, \qquad i = 1, \ldots, N \tag{3.1}$$

Here N denotes the number of used AP for this estimation. By subtracting one equation from the other the set of non-linear equations can be transformed into a set of linear equations. This is shown in Equation 3.2.

$$d_i^2 - d_j^2 = 2(x_j - x_i)\hat{x} + 2(y_j - y_i)\hat{y} + x_i^2 + y_i^2 - x_j^2 - y_j^2, \qquad i = 1, \ldots, j-1, j01, \cdots, N \tag{3.2}$$

This will give an equation system with $N - 1$ linear equations which can be stated as a matrix on the form $Ax = b$ where A,b and x are shown in 3.3, 3.4 and 3.5 respectively.

$$\mathbf{A} = \begin{bmatrix} 2(x_1 - x_2) & 2(y_1 - y_2) \\ 2(x_1 - x_3) & 2(y_1 - y_3) \\ \vdots & \vdots \\ 2(x_1 - x_N) & 2(y_1 - y_N) \end{bmatrix} \tag{3.3}$$

$$\mathbf{b} = \begin{bmatrix} d_2^2 - d_1^2 - x_2^2 - y_2^2 + x_1^2 + y_1^2 \\ d_3^2 - d_1^2 - x_3^2 - y_3^2 + x_1^2 + y_1^2 \\ \vdots \\ d_N^2 - d_1^2 - x_N^2 - y_N^2 + x_1^2 + y_1^2 \end{bmatrix} \tag{3.4}$$

$$\hat{\mathbf{x}} = \begin{bmatrix} \hat{x} \\ \hat{y} \end{bmatrix} \tag{3.5}$$

Due to lack of accuracy when measuring the distances d_i there might not be a solution to the equation system. This problem is solved by approximating the position of the MU by applying the least squares approximation. This is done by solving the equation $\mathbf{e} = \mathbf{b} - \mathbf{Ax}$ and finding a solution to the equation system where the length of $\mathbf{e}$ is as small as possible, $\hat{x}$ is then a least squares solution. This can be found with Equation 3.6.

$$\mathbf{A}^T\mathbf{A}\hat{\mathbf{x}} = \mathbf{A}^T\mathbf{b} \Rightarrow \hat{\mathbf{x}} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b} \tag{3.6}$$

Since this method is based on signal strengths, elements that affect this will have a negative impact on the location. This could be a wall between the AP and MU. The wall will absorb some effect from the signal which will result in an erroneous distance estimate, because the signal reduction caused by the wall is larger than for free space. Ideal conditions for using this technology is therefore to use totally omnidirectional antennas and free-space propagation conditions. Another element that has a big impact on the localization accuracy is the distance between AP and MU. As illustrated in Figure 3.7 it can be seen that a small distance between AP and MU results in a higher accuracy of the distance estimate if the inaccuracy of the signal strength measurement is constant.

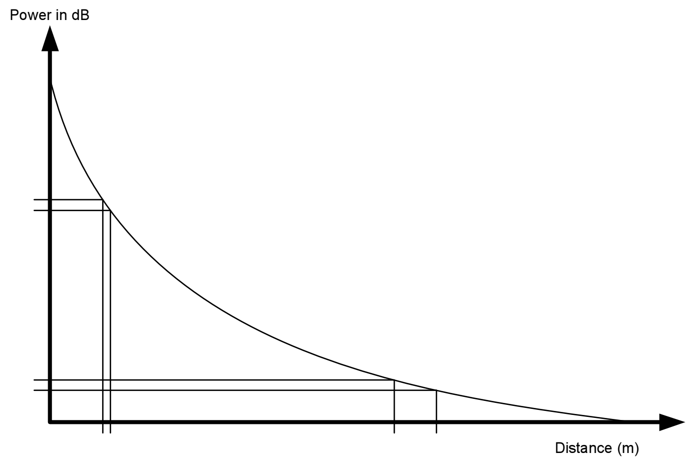


Figure 3.7: This figure shows how the accuracy of the distance estimate decreases when the distance between the AP and MU increases.

Triangulation based on Time Of Arrival (ToA)

This localization method is very similar to the previous signal strength localization method. Instead of converting a signal strength to a distance, a ToA is converted to a distance. ToA is the time it takes for a signal to propagate from the sender to a receiver. By knowing the propagation time, the ToA can be converted to a distance and triangulation can be used to do localization as described in previous section. Prerequisites for this method is that each AP has to be synchronized with the MU to be able to measure the ToA. The localization precision is therefore limited by how precise the ToA can be calculated.

Triangulation based on Time Difference Of Arrival (TDoA)

In difference to ToA, TDoA is a location method that does not require all APs to be synchronized with the MU. Instead of using ToA, this method uses the TDoA. TDoA is based on the difference in time, multiple nodes are receiving the same signal. The principle is that a MU is transmitting a signal that after some time is received by multiple APs. Because the transmission distance between the MU and each AP can vary, the time each AP is receiving the signal will vary. By measuring the TDoA between two APs the possible location of the MU is forming a hyperboloid. By introducing one more APs a new TDoA can be calculated between this AP and one of the two other APs. This will give one more hyperboloid and the intersection between these will be the location of the MU. This is illustrated in Figure 3.8. The localization precision is therefore limited by how precise the TDoA can be calculated.

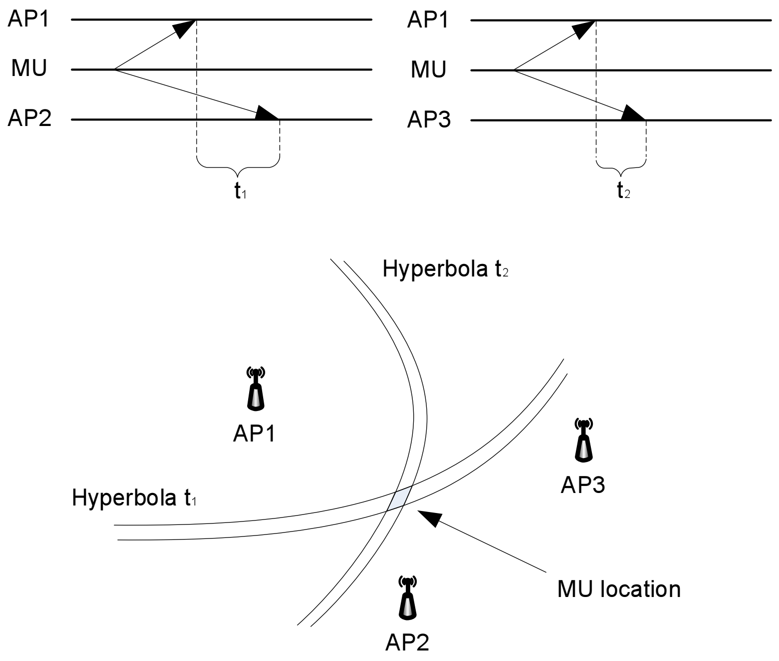


Figure 3.8: This figure shows a visualization of the hyperboloids that is calculated by TDoAs. The intersections of the hyperboloids is the MU location.

One way to measure the TDoA is to synchronize all APs and having a central unit where all time stamps from each AP are collected. The TDoAs can then be calculated and the localization can be estimated. This is illustrated in Figure 3.9.

Summary

This chapter has covered different methods for localization and found possible candidates for the project.

The first group of localization methods was Proximity where two different methods exist: Cell identification and physical contact. Caused by the low localization precision of

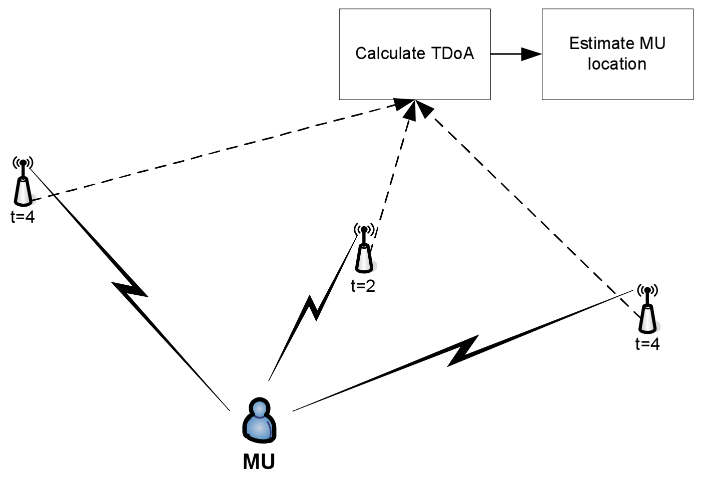


Figure 3.9: This figure illustrates localization by using Time Difference Of Arrival (TDoA). The MU is transmitting a signal at time 0. Each AP is receiving the signal at time 4, 2 and 4. The time the signal is received for each AP is sent to a central place where the TDoAs is calculated. Based on this the MU location is estimated.

cell identification this method can not solely be used in order to obtain location estimates that live up to the requirements of the project scenario. Furthermore, physical contact that is based on e.g. switches in a building floor requires a large amount of work in the installation process, which also does not agree with the PnP idea described in the project scenario. This group of localization methods can therefore not by used in this project.

The second group of localization methods was the environment analysis. Here two approaches were described: video analysis and fingerprinting. The video analysis is based on some sort of a picture of the environment. Because this project has to utilize a short-range wireless communication technology for localization, and no picture can be extracted, this method cannot be used in this project. Furthermore, as mentioned above, fingerprinting requires a comprehensive setup phase where RSSI measurements and positions are coupled and collected in a database. Because the setup phase requires a lot of human effort in training the system, which does not conform to the PnP idea described in the project scenario, this method also cannot directly be used in this project. Hence, this group of localization methods is, like the first, not useable in this project.

Lastly, the radio localization group can be utilized to do localization. Four different methods were described: Angulation and triangulation based on either RSSI, ToA and TDoA. Angulation is a method that uses antenna arrays to do localization. These antennas are not used in normally wireless installations and replacing existing antennas with antenna arrays will contradict with the PnP idea which are the reason why this method is not used. Triangulation based on ToA is a localization method that requires that the MU, that has to be located, is synchronized with APs before a ToA can be measured and localization can be done. Because synchronization can be hard to achieve and since similar method exists, such as TDoA, that does not require synchronization between MU and AP this method will also not be used for this project.

This leaves triangulation based on either TDoA or RSSI measurements. Triangulation based on TDoA is a method where the MU and AP does not need to be synchronized. To measure the TDoA on an AP it is necessary to measure the exact time a given signal is received. The more precise this time is measured, the more precise the localization can be. To measure this time as accurate as possible it can be necessary to modify the firmware in each AP. Because the idea with this project is to utilize eventual existing wireless short-range communication networks, multiple firmwares have to be developed which will not be an easy task. Furthermore, if all APs have to be upgraded, this would contradicts the PnP idea. This will therefore not be used. Triangulation based on RSSI measurements complies with the use of a short-range technology. The localization precision is good under good propagation conditions. However, because the scenario is taking place indoor this will lead to multipath fading and shadowing which will have a negative effect on the localization precision. This negative effect can possibly be reduced or even removed by using filters, mobility models and SLAM. Triangulation using RSSI measurements will therefore be the localization method used in the further analysis.

The next step will be to investigate the wireless technology on which the localization will be based.

Chapter 4

Wireless technology

This chapter will describe the wireless technology chosen for the implementation of the localization method analysed and developed in this project. It will first account for the considerations which have lead to the choice and will then describe the parts of the technology, which are interesting in relation to the subject of this project.

The aim with this chapter is to account for the reasons, which have lead to the choice of Bluetooth as the technology used in this project as well as to present the aspects of Bluetooth, which will become relevant later in the project. Here especially Bluetooth used for localization with RSSI measurements will be a central aspect.

4.1 The choice of technology

So far the idea of using a localization technique from robotics, named SLAM, has been introduced and in the Scenario chapter, chapter 2 on page 12, a scenario has been described, where using this technique may prove advantageous. Furthermore, the previous chapter has presented different localization methods, which all are applicable to the described scenario. Having this in mind, this section will discuss the choice of a suitable technology.

Only a limited set of wireless technologies will be considered as possible candidates in this project. The technologies are IEEE 802.11 Wireless Local Area Network (WLAN) and Bluetooth. These technologies are considered because they both are already widely deployed short-range technologies. This ensures that the performed research will have a bigger interest group and a larger impact.

SLAM does not set any requirements to the technology used for measuring distances other than that it can do exactly that; measure a distance. Naturally, SLAM will profit from higher accuracy of the distance measurements and from additional supporting information, such as the bearing of measurements. This is shown in Figure 4.1, where three situations with different levels of measurement accuracy and supporting information are illustrated. As it can be seen in the figure, the expected location of a MU based on a RSSI measurement, becomes more precise as better measurements that are used and additional supporting information is introduced.

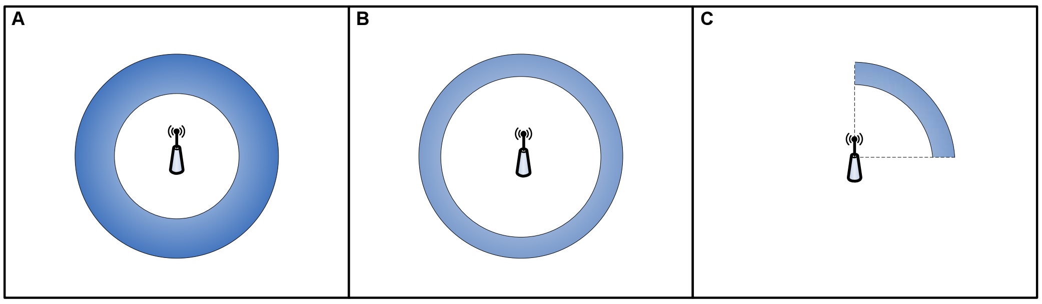


Figure 4.1: This figure shows three levels of measurement accuracy and additional information. In A only a inaccurate RSSI measurement is available. In B the RSSI measurement is more accurate and in C the RSSI measurement is supported by additional information about the bearing of the measurement.

However, since neither of the two chosen technology candidates, WLAN and Bluetooth, has a considerable advantage over the other with respect to the accuracy of RSSI measurements [Ye05] [EEM05] and non of them can provide any additional information such as the bearing of a measurement without the use of a directional antennas, other criteria will be used in order to chose one of the technologies over the other.

Instead of the RSSI accuracy, the criteria for the choice of technology will be today's diffusion of the technologies. Here Bluetooth has a clear advantage over WLAN, since it already is build-in in almost all mobile phones and PDAs sold today, while WLAN only is beginning to gain a footing in the phone industry. For instance, the number of Bluetooth units world wide was 500 million in 2005[nBS06].

The increasing number of WLAN hot-spots is here recognized, but since the use of these hot-spots mainly is characterized by laptops used while users are stationary e.g. are sitting at a table in a fast-food restaurant or at the airport, the primary use-case for these kind of scenarios does not represent the typical scenario with need for localization and tracking of MUs.

Based on these considerations Bluetooth is chosen as the technology, which later in this project will be used for the implementation of a proof-of-concept system. Appendix A on page 211 will provide an outline of the Bluetooth standard that gives some background information about the technology.

4.2 Bluetooth and localization

In the previous chapter about localization techniques, see Chapter 3 on page 16, triangulation through Received Signal Strength Indicator (RSSI) was chosen as the preferred localization technique. In order to use Bluetooth RSSI measurements for triangulation, this section will investigate how RSSI measurements can be obtained.

The section will, in addition to the sources used in the previous part, also be based on [Fig04].

RSSI measurements can be obtained only if a radio link exists between two devices. In Bluetooth networks, two situations, which provide this prerequisite, are interesting: A situation where an already established link between two devices is used to obtain the RSSI measurement and a situation where two devices are not yet connected via a radio link, but are searching for a connection. The last situation is known as the Bluetooth device discovery state. The main difference between the two situations is whether the devices are in a connection or standby state.

Obtaining RSSI measurements based on ongoing transmissions

In the first situation the devices can simply measure the power level of the received data packets and calculate the RSSI based on this. However, this technique for measuring the RSSI is not as straight forward as it sounds. Bluetooth provides a mechanism that continuously regulates the transmit power in order not to transmit with more power than necessary to reach the receiver. This way the transmitting Bluetooth device saves power. Hence, the signal strength of a signal received at a random point during an ongoing transmission cannot be predicted. This entails the problem: In order for a receiver of a signal $S(t)$ to use a RSSI measurements at $S(1)$ for localization purposes, it is necessary for him also to know the signal strength at the transmitter at $S(1)$. If the signal strength changes over time the receiver requires knowledge about these changes in order to be able to calculate the signal strength at the transmitter for any given $S(t)$. The problem could be solved by tracking control packets send from the receiver to regulate the transmit power on the transmitter. These packets contain all necessary information about the transmit power changes and could be stored on the receiver. This way the signal strength at the transmitter for any given $S(t)$ could be calculated by the receiver.

The technique of using a transmission strength regulated ongoing transmission is may prove to have advantageous properties with respect to the availability of RSSI measurements, since these can be obtained for any received packet and do not need to wait for special packets. Furthermore, the transmission signal strength regulation is a optional feature on some Bluetooth devices e.g. Bluetooth APs. On such devices, an ongoing transmission could be used for localization without having to track control packets.

Obtaining RSSI measurements based on device discovery

In the second situation the Bluetooth devices are not yet connected, but are trying to do so. The behaviour necessary to establish a connection differs whether it is a master or a slave. The master will send out an inquiry for other devices to join his piconet, whereas the slave will scan for inquiries, presuming it is interested in establishing a connection.

Because of the different behaviours and Bluetooth using Frequency Hop Spread Spectrum (FHSS), there are situations in which a connection will not be established. These situations are:

- When the master is not interested in new slaves and does not send out an inquiry.

- When the slave is not interested in joining a network and therefore does not scan for inquiries.
- When the master is sending out an inquiry and the slave is scanning, but the devices are not on the same frequency at the same time.

The following will describe the details of how the master and slave respectively behave during a Bluetooth device discovery. Here the power level of the packets exchanged in the process can be used, since they always are sending with maximum power in order to discover devices in as big an area as possible.

Inquiry phase: During the inquiry phase the master will send an identity (ID) packets as inquiry on 32 out of the 79 frequencies. The hopping sequence of the inquiry is determined by the general inquiry address and the hopping phase is defined by the native clock. This procedure is repeated a number of times depending on the number of active SCO channels on the master. The reason for this is, that the SCO channels have a higher priority than inquiries. The scheduler in the baseband resource manager will therefore always grand the SCO channels access to the baseband before inquiries. Since ACL connections have a lower priority than inquiries they will not affect the inquiry procedure.

If during the inquiry phase any Frequency Hop Synchronization (FHS) packet is received as response, the master reads it without sending any acknowledgement. It can then either return to the inquiry procedure to discover more devices, or it can proceed to page the discovered device, which is the procedure that inducts the discovered device as a slave in the piconet. The inquiry procedure continues until the master has collected enough information about new slaves in the vicinity and stops the process by itself, until a timeout is reached, or until the host on the master cancels the process.

Scan phase: If a Bluetooth device is interested in joining a Bluetooth network it will start a scan procedure which makes it discoverable by devices in the inquiry phase. This is possible both for Bluetooth devices that already belong to a piconet and for devices that are not associated to any piconet. The scanning procedure is as default repeated every $1.28s$, but the device scans actively only during the first $11.25ms$ of this period. For the rest of the duration the device maintains already established connections, if such exist, or goes into standby to save power.

As in the inquiry, the scan undergoes frequency hopping. The hopping sequence is again determined by the general inquiry address, while the phase is defined by the native clock. Also, like the inquiry, scanning has a lower priority than SCO channels and a higher priority than ACL channels. Because of the static scanning duration, this means that the probability of a successful scan procedure is affected by the number of active SCO channels. Since the baseband may be occupied by a synchronous transmission during the scan, the total time spend on scanning is smaller. To counteract this problem, the scan duration is multiplied by $n + 1$, where n is the number of SCO channels.

If, during the scan procedure, an ID packet is received, the scanning device will enter the response phase.

Response phase: After reception of the ID packet, the device will wait exactly one time slot (625μ) and then it responds with a FHS packet with its address and native clock. By sending the FHS packet the scanning device states its interest to join the piconet.

Although unlikely, multiple devices may respond at the same time on the same channel. In order to avoid multiple devices to repeatedly respond to an ID packet simultaneously and thereby create repeated collisions, the device enters a random back-off interval after having transmitted the FHS packet. The interval is uniformly distributed between 0 and 1023 time slots, which means that the device will back-off $639.375ms$ ($1023 \cdot 625\mu$) at the most, during which time the device will return to connection or standby state. Upon ending the back-off, the device will start to scan for page packets from the master device.

An overview of the three steps in the Bluetooth device discovery are shown in Figure 4.2.

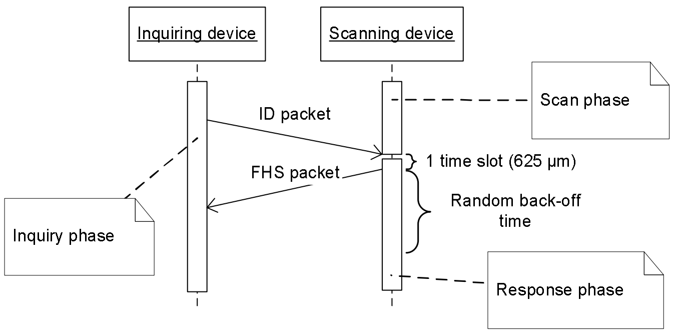


Figure 4.2: This figure shows the three steps of the Bluetooth discovery.

Choosing this second technique for localization ensures that the RSSI measurements are based on packets transmitted with a known transmit power, since packets in the described phases are not affected by the transmit power regulation mechanism.

However, obtaining RSSI measurements through this technique is only possible during the Bluetooth device discovery. This leads to limitations on both when and how often RSSI measurements are available. Previous research has shown that under optimal conditions with a noise-less channel 99% of all scanning devices are detected by a Bluetooth device after $3.84s$ in the default inquiry state[1][BSPK06].

[1]The state values are for Bluetooth version 1.2, the default $11.25ms$ inquiry scan window that opens every $1.28s$ and the standard inquiry scan mode. No values for Bluetooth version 2.0 were available.

This means that RSSI measurements can be expected to be available more often than every 3.84s in worst case scenarios using a optimal channel. This worst case interval will of cause increase as noise and ongoing transmissions on one or both of the device involved in the process is introduced. The average expected number of RSSI measurements will be $3 - 4$ measurements per second.

Choosing a RSSI measurement technique

The ideal prerequisites for localization using RSSI are measurements with a high accuracy that are provided at any time. However, this has been shown not to be possible. Thus, the choice between the two presented techniques for obtaining RSSI measurements depends on the requirements to the accuracy and timely availability of the measurements. The choice will therefore necessarily be based on an scenario dependent assessment of the needs that will have to be performed. This assessment will be performed in the in the analysis part of this project.

Although it now is clear which localization method to use and which packets the RSSI measurements potentially can be obtained from, there are still problems to be discussed. The scenario introduced in Chapter 2 on page 12 describes an environment with walls, which undoubtedly will have an effect on wireless signals. The next chapter therefore will investigate the wireless channel and its effects on the RSSI measurements.

Chapter 5

Wireless channel

This chapter will describe the basic properties of a radio channel between a sender and a receiver in indoor environments. Furthermore it will investigate what influences radio propagation has on localization when using triangulation based on RSSI measurements.

The purpose of this chapter is to determine how the wireless channel affects the localization of MU through triangulation with RSSI measurements.

5.1 Introduction to propagation

The chapter will in general be based on [ARY95] and [Par92]. Supplementary references are used where needed.

The most basic radio propagation model is propagation in "free space" that describes an environments where there are no obstacles, no gases to absorb energy and nothing to scatter the radio waves. In this model the wireless channel will emanate from the transmitter in all directions in a straight line and the energy strength will decay with $\frac{1}{d^2}$, where d is the distance between the transmitter and the measurement point. This propagation model, however, is not satisfactory in most cases, because propagation scenarios rarely takes place in totally "free space". Mechanisms which govern radio propagation can be attributed to four basic propagation mechanisms: penetration, reflection, diffraction and scattering which will be described in the following.

Penetration: In some cases waves are capable of penetrating obstacles. Depending on the material and the used radio frequency, part the energy will be absorbed by the obstacle. Shadowing is a phenomenon related to penetration and describes the effect where obstacles such as walls and fixtures absorb some, or all, energy from the signal depending on the obstacle. The obstacle then throughs a "shadow" in which the signal is weakened. In worst case this will create "dead spots" where no signal can be received.

Diffraction: Diffraction takes place when radio waves encounter an obstacle that has a sharp edge. Secondary waves resulting from the edge will in such cases be bent around the edge of the obstacle. This is illustrated in Figure 5.1.

Reflection: Reflection describes the effect when a radio wave is impinging upon an obstruction with considerably larger dimensions than the radio wave's wavelength. This could be the floor or the walls in a building. Examples of reflections is illustrated in Figure 5.2.

Scattering: A radio waves is scattered when it encounters obstacles that have small dimensions compared to the radio wave's wavelength. Scattering follows the same physical principles as diffraction, which causes the signal to be reradiated in many different directions when the signal encounters these small objects or rough surfaces.

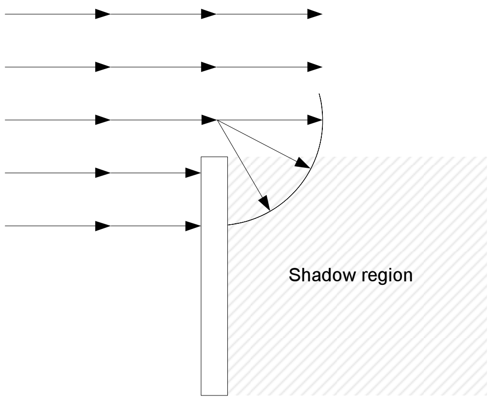


Figure 5.1: Diffraction at the edge of an obstacle.

Caused by these propagation mechanisms the radio wave will penetrate or be reflected, diffracted and scattered by objects in the environment, which means that the received signal will probably consist of multiple signals. These different paths by which the signal travels from the transmitter to the receiver is called multipath propagation and is illustrated in Figure 5.2. Because of these mechanisms it is also possible to receive a signal without having LOS to the transmitter.

As the MU moves over small distances, the received signal will fluctuate rapidly in signal strength because the received signal will be a sum of contributions coming from many different directions. Since the phases appear random due to the difference in the length of different propagation paths and therefore also transmission time, constructive and destructive addition of multipath signals will cause the received signal strength to fluctuate. At one locations there can be a constructive addition which will give a high signal strength, whereas another, close by, location can have destructive addition that causes an almost complete cancellation of the signal strength. An example of constructive and destructive addition of two signals is illustrated in Figure 5.3. This fluctuation over small distances caused by multipath propagation is called small-scale fading.

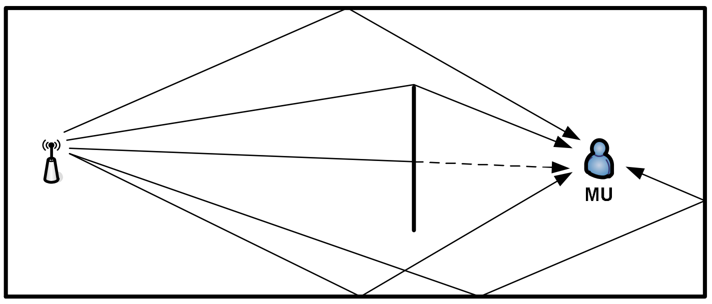


Figure 5.2: Example of multipath propagation. The signal propagates through five paths: three reflections on the outer walls, one diffraction on the inner wall and one that penetrates the wall.

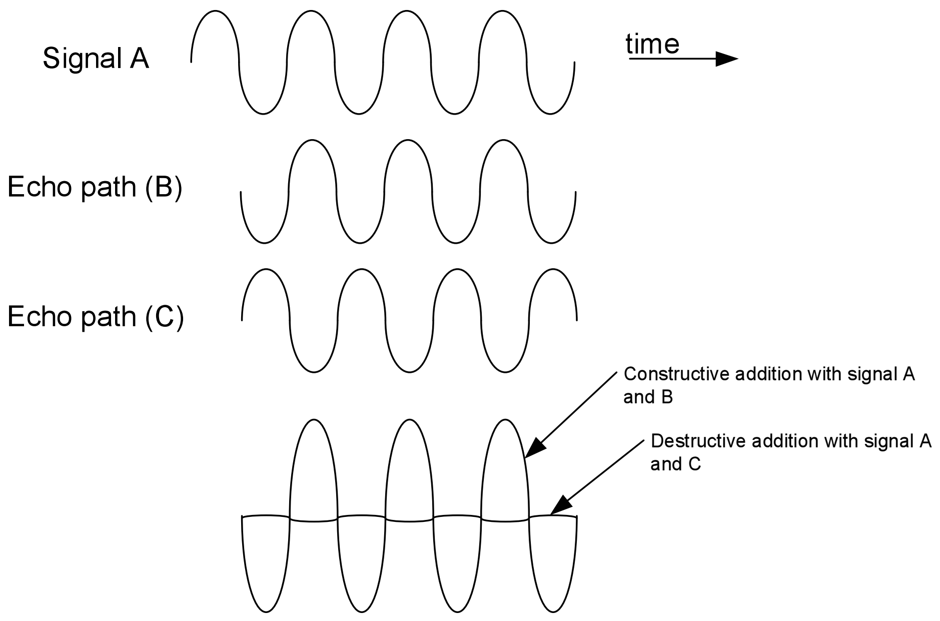

Signal A
time
Echo path (B)
Echo path (C)
Constructive addition with signal A and B
Destructive addition with signal A and C

Figure 5.3: Illustration of constructive and destructive addition of two transmission signals.

Another characteristic of radio signals is large-scale fading. This is the average attenuation in signal strength over a large area caused by the distance to the transmitter and due to shadowing.

Propagation in indoor scenarios

In indoor scenarios obstacles have a big impact on the received signal. Obstacles can be divided into two groups: soft and hard partitions. Hard partitions are all obstacles in a building which cannot be easily moved such as walls. Soft partitions are all obstacles such as furniture that are movable. Because these obstacles will absorb or reflect radio energy they have a direct impact on the wireless propagation channel and the received signal strength. Due to shadowing or destructive addition of multipath propagation a signal may be highly attenuated after propagating only a few meters e.g. through a wall, but also the opposite case exists, where a very strong signal can be received far away along a corridor.

A third group that influences the wireless channel are dynamic elements such as people moving around, doors being opened or closed and the placement of the mobile device on the MU. Because these elements are highly dynamic they can be very hard to predict.

5.2 Propagation and localization

As described in the localization chapter the method that is chosen for localization in this project is triangulation based on RSSI measurements. Because different phenomenons such as shadowing and multipath propagation have a big impact on the received signal and thereby the RSSI measurement, this impact will have a negative effect on the localization precision. To compensate for this, two aspects will be considered; mobility-models and SLAM. These aspects will be described in the following.

Chapter 6

Mobility model theory

This chapter introduces the theory of mobility models and will provide examples of the most widely used models. The chapter will first give an introduction to the subject and them describe selected examples of mobility models and discuss their properties. The chapter will end with a discussion of how mobility models can aid in the localization process.

The purpose of this chapter is to explain the advantages of combining localization with mobility models.

6.1 Introduction to mobility models

The need for mobility models aroused when it became possible to connect to networks through wireless connections. The absent need for cables provided the user with the possibility to be mobile while staying connected to the network. This mobility, however, entailed new problems such as handover between different APs or BSs and complex routing. This lead to much research in ways to handle these problems.

Some of the proposed methods to handle these problems included predicting how a given user would move in order to estimate his location in the near future. A tool for this were the mobility models. Mobility models are used to describe the movement pattern of mobile users, and how their location, velocity and acceleration changes over time[TC02].

One way to groups mobility models is to divide them into *Cell based* mobility models and *fine granularity* mobility models [Lib02]. The reason for making this distinction is that in cellular networks the field, that is the geographical area covered by the wireless network, is divided into independent cells who's size is defined by the used technology, whereas in Ad Hoc wireless networks the field can be divided into arbitrarily small cells depending on the needed resolution of the field. Hence the granularity of models in this case can be arbitrarily small. Although this initially does not seam to be a crucial parameter, the following description of the two different groups will present the reasons for the distinction between the two groups.

Cell based mobility models: In cellular networks, the field is divided into cells, each of which is serviced by some sort of fixed emplacement e.g. a BS. The movement of nodes in the field is considered only on a per-cell basis. This entails that a node's exact position becomes irrelevant; the identity of the cell that the node currently is in, is the only relevant data for the mobility model. A mobility model for this kind of networks is typically defined by the handover probability and blocking probability to each of the neighboring cells and the mobility model therefore deals with movement on per cell-level.

Fine granularity mobility models: In a mobile ad hoc network, the movement of nodes is modelled on a more detailed level. For this type of network, it is important to model node positions and movement in order to provide e.g. more stable routes through the network since the 'time to life' for a given path can be estimated. This could be done by using a data representation like shown in Equation 6.1 or 6.2 where $\vec{M}_i(t_0)$ is the start condition and $\vec{M}_i$ is the mobility metric denoting the i^{th} MU's relative location to a reference point. In the equations Δt denotes the time slot size. Equation 6.1 describes the position in cartesian coordinates where as Equation 6.2 describes the position in polar coordinates.

$$\begin{aligned} \mathbf{M}_i(t_0) &= \begin{bmatrix} x(t_0) \\ y(t_0) \end{bmatrix} \quad , \\ \mathbf{M}_i(t+1) &= \begin{bmatrix} x(t) \\ y(t) \end{bmatrix} + \begin{bmatrix} v_x(t) \\ v_y(t) \end{bmatrix} \cdot \Delta t \quad , for \quad t = 1, 2, 3, ... \end{aligned} \tag{6.1}$$

$$\begin{aligned} \mathbf{M}_i(t_0) &= \begin{bmatrix} \theta(t_0), & r(t_0) \end{bmatrix} \quad , \\ \mathbf{M}_i(t+1) &= \begin{bmatrix} \theta(t), & r(t) \end{bmatrix} + \begin{bmatrix} v_\theta(t), & v_r(t) \end{bmatrix} \cdot \Delta t \quad , for \quad t = 1, 2, 3, ... \end{aligned} \tag{6.2}$$

Since a cell based approach does not match the requirements of the chosen project scenario where a more detailed localization of MUs is needed, this chapter will henceforward only focus on fine granularity mobility models. A series of these models are introduces in Appendix B on page 218.

6.2 Mobility models and localization

Appendix B on page 218 introduces several mobility models which each consider different aspects of the MU characteristics and its surroundings. In order to use mobility models for localization they have to fulfil one purpose: They have to support the prediction of the MU's location. This support to the localization process is provided in the form of (1) additional knowledge regarding likely locations at time $t + \Delta t$ based on measurements obtained in the interval from 0 to t and (2) a delimitation of possible locations at time $t + \Delta t$ based on knowledge about 'impossible' locations.

In the project scenario several of the aspects considered by different mobility models apply. The movement of MUs will definitely be restricted by obstacles in the scenario (walls etc.) and there is a chance that certain MUs have correlated movement (guided group tours). It will therefore be necessary to consider a selection of the described mobility models that fit different aspects of the project scenario and combine these in order to develop a more complex mobility model that reflects the behaviour of MUs in the project scenario. The mobility models that will be considered are:

Weigthed Random Walk Model: This mobility model will be considered, since it provides the possibility to model MUs that follow paths that are more likely than others. In the project scenario this could reflect MU movement along guided paths or movement between certain points of interest e.g. the displayed objects of the museum.

MUs in the project scenario can change direction at any given time and may choose any speed within a certain range. This can reasonably be modelled with the Random Walk Model. However, small changes can make the model even more suitable for the project scenario. By choosing a new direction and speed not randomly, but according to some location depended distribution it can be modified to be more aware of the environment. This way the model may dictate a certain direction of movement for MUs located in a given region, or may limit the speed range of MUs in others. Thus the model can be expended to a far more complex description of MU behavior.

It may furthermore be possible to enhance this model by introducing preferred levels of speed as described in the Smooth Random Model.

Obstacle Model: This mobility model will be considered, since it provides the possibility to model obstacles in the scenario which prevent the MU from moving along any given path. In the project scenario this could reflect walls or other obstacles that prevent a MU from moving in a direct line from one point to another.

It may be possible to enhance this model by defining specific actions for situations where a MU encounters an obstacle e.g. a MU will move along the obstacle until it can resume moving towards its destination in a straight line.

Another aspect of mobility models is that all of the above considered mobility models are based on knowledge about the scenario gathered prior to the employment of the mobility model. This may be difficult or even not possible for a mobility model that is applied in a system that seeks to remove the need for an initial configuration. It should therefore be investigated if it is possible to apply a mobility model in this project that is able to change dynamically as more information about the actual movement of MU is gathered. Such a learning ability would be immensely useful.

Chapter 7

SLAM

This chapter will describe how SLAM is applied in scenarios with localization of autonomous vehicles and how it can be modified for localization scenarios based on short-range wireless technologies. The purpose of the chapter is first to describe the SLAM technique in detail and then to present the changes necessary for SLAM to be used for localization of MUs.

7.1 Introduction to SLAM

Originally Simultaneous Localization And Mapping (SLAM) represents a solution to the problem of determining the relative location of an autonomous vehicle starting in an unknown location in an unknown environment. SLAM does this by incrementally building a map of the environment while simultaneously using this map to compute the absolute vehicle location relative to a set of chosen landmarks in the environment. Thus the main advantage of SLAM is that it eliminates the need for artificial infrastructures or a priori topological knowledge of the environment[ea01].

Substantial research has been conducted into finding a solution to the general SLAM problem. Overall, the research can be divided into approaches with three different angles on the problem.

Kalman filter based: The first is a Kalman filter based approach that uses estimations of the relative locations of the vehicle and landmarks[ea01] [RSC90].

Qualitative knowledge based: The second angle uses qualitative knowledge of the relative location of the vehicle and landmarks[Bro86] [KYT91]. This could be a vehicle using the series of actions as the description of relation between the two locations e.g. "to get from landmark A to landmark B move for t seconds in direction θ at speed v.

Video analysis based: The third does away with the strict Kalman filter or statistical formalism while retaining an essentially numerical or computational approach to describe the uncertainty of observations. Examples of this are iconic landmark matching[Zha94], as shown in Figure 7.1, or global map registration[CK97].

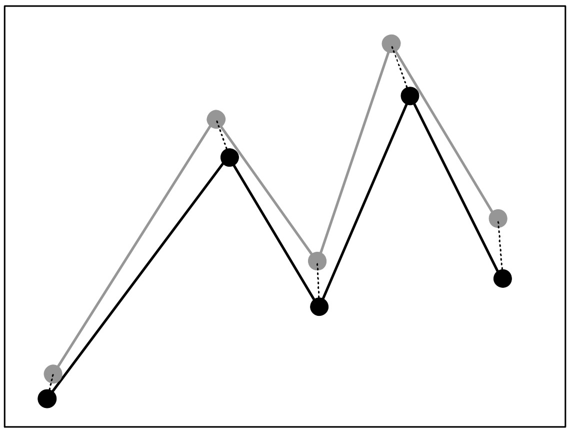


Figure 7.1: Matching of the observed curve (grey) to the known shape of the environment (black).

This project will use the first, Kalman filter based, approach because it provides the possibility to estimate the location of a MU based on statistical models for vehicle movement and relative landmark estimations.

7.2 Kalman filter based SLAM

The Kalman filter based approach relies on continuous observations of the relative location of landmarks to a moving vehicle and a known kinematic model of the vehicle. This information is maintained in a state vector consisting of all states in the vehicle model and all states of every observed landmark. It then uses the fact that the estimates of all landmark locations are correlated with each other because of the common error in estimated vehicle location[RSC90].

The following subsections will describe first a general model for a vehicle and landmarks, then a general model for the observations and lastly the estimation process. The description of the models will be similar to the on used in [ea01].

Vehicle and landmark model

As described earlier, the Kalman filter based approach to SLAM relies on observations of landmarks from a moving vehicle. An example of this is shown in Figure 7.2. As the Figure indicates some mathematical representation of both landmarks and vehicle is needed that can reflect their timely behaviour and their relation to each other. Thus a model for both the landmarks and the vehicle is needed. Furthermore these models should be based on a common spatial reference model. The state of the system consists of the position of the

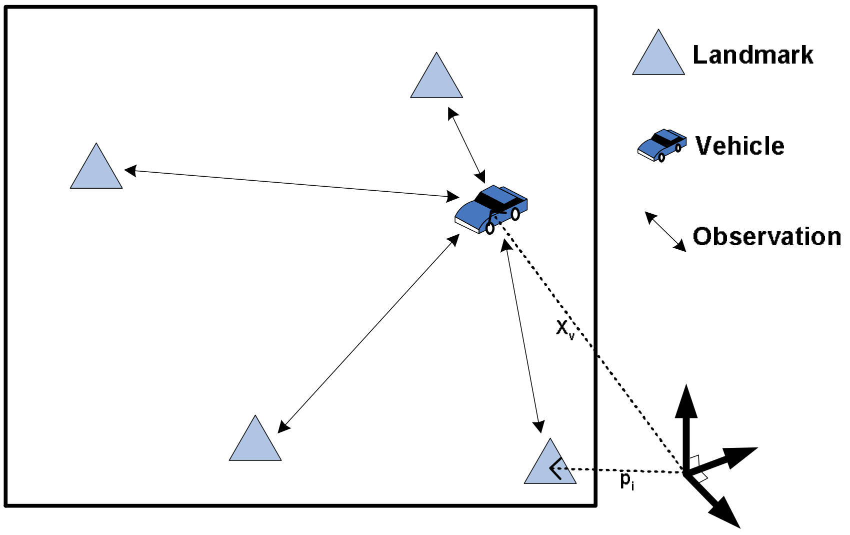


Figure 7.2: A vehicle moving with a known kinematic model while observing a set of landmarks.

vehicle and the landmarks denoted as $\mathbf{x}_v(t)$ for the state of the vehicle at time step t. The model for the vehicle, also referred to as the *process* or *motion model*, consists of the linear discrete-time transition equation between each time step. This is shown in Equation 7.1.

$$\mathbf{x}_v(t+1) = \mathbf{F}_v(t)\mathbf{x}_v(t) + \mathbf{u}_v(t+1) + \mathbf{v}_v(t+1) \tag{7.1}$$

where $F_v(t)$ is the state transition matrix, $\mathbf{u}_v(t)$ is a control vector input and $\mathbf{v}_v(t)$ a vector of temporally uncorrelated process noise with zero mean and covariance $\mathbf{Q}_v(t)$. Since the landmarks are assumed stationary the state transition equation for the i^{th} landmark will be denoted as in 7.2.

$$\mathbf{p}_i(t+1) = \mathbf{p}_i(t) = \mathbf{p}_i \tag{7.2}$$

The number of landmarks in the environment is arbitrarily set to N. A vector of all N landmarks therefore is denoted

$$\mathbf{p} = \left[\begin{array}{ccc} \mathbf{p}_1^T & \dots & \mathbf{p}_N^T \end{array}\right]^T \tag{7.3}$$

where T denotes the transpose and is used to save space. The augmented state vector containing both the vehicle state and the states of all landmarks is denoted

$$\mathbf{x}(t) = \left[\begin{array}{cccc} \mathbf{x}_v^T(t) & \mathbf{p}_1^T & \dots & \mathbf{p}_N^T \end{array}\right]^T \tag{7.4}$$

The complete augmented state transition model for the system process model as shown in Equation 7.1 may now be written as

$$\begin{bmatrix} \mathbf{x}_v(t+1) \\ \mathbf{p}_1 \\ \vdots \\ \mathbf{p}_N \end{bmatrix} = \begin{bmatrix} \mathbf{F}_v(t) & 0 & \dots & 0 \\ 0 & \mathbf{I}_{p1} & \dots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & \mathbf{I}_{p_N} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{x}_v(t) \\ \mathbf{p}_1 \\ \vdots \\ \mathbf{p}_N \end{bmatrix} + \begin{bmatrix} \mathbf{u}_v(t+1) \\ \mathbf{0}_{p_1} \\ \vdots \\ \mathbf{0}_{p_N} \end{bmatrix} + \begin{bmatrix} \mathbf{v}_v(t+1) \\ \mathbf{0}_{p_1} \\ \vdots \\ \mathbf{0}_{p_N} \end{bmatrix} \tag{7.5}$$

where $\mathbf{I}_{p_i}$ is the $dim(p_i) \times dim(p_i)$ identity matrix and $\mathbf{0}_{p_i}$ is the $dim(p_i)$ null vector.

Special cases where a landmark p_i is in stochastic motion may also be accommodated with the described model. This can be achieved by modifying Equation 7.2.

Observation model

The vehicle is equipped with a sensor that enables it to obtain observations of the location of the landmarks relative to itself. The observations are assumed to be linear and synchronous. This entails that for each step the complete transition matrix will have to be updated. The observation for the i^{th} landmark is written as

$$\begin{aligned} \mathbf{z}_i(t) &= \mathbf{H}_i x(t) + \mathbf{w}_i(t) \\ &= \mathbf{H}_{pi}\mathbf{p} - \mathbf{H}_v(t) + \mathbf{W}_i(t) \end{aligned} \tag{7.6}$$

where $\mathbf{w}_i(t)$ is a vector of temporally uncorrelated observation errors with zero mean and variance $\mathbf{R}_i(t)$. $\mathbf{H}_i$ is the observation matrix that relates the output of the sensor $\mathbf{z}_i(t)$ to the state vector $\mathbf{x}(t)$ when observing the i^{th} landmark. It is here important to note that observation model for the i^{th} landmark is written as

$$\mathbf{H}_i = \left[\ -\mathbf{H}_v, 0 \ldots 0, \mathbf{H}_{pi}, 0 \ldots 0 \ \right] \tag{7.7}$$

This structure reflects that all observations are relative between the vehicle and the landmark. Mostly this is in the form of a relative location or a relative distance and bearing.

Estimation process

As described earlier, this project will use the Kalman filter based approach to SLAM. Hence, the Kalman filter is used to provide estimates of the vehicle and landmark locations. This subsection will shortly summarize the main steps of a Kalman filter. Details of the Kalman filter can be found in appendix C on page 224 and in [WB01].

The Kalman filter recursively computes estimates for a state $\mathbf{x}(t)$ which is evolving according to a process model and which is observed according to an observation model. An examples of a process model is described in Equation 7.5 and an observation model is shown in Equation 7.6. It computes an estimate which is equivalent to the conditional mean $\hat{\mathbf{x}}(\mathbf{p}|\mathbf{q}) = E[\mathbf{x}(\mathbf{p})|\mathbf{Z}^q]$ $(\mathbf{p} \geq \mathbf{q})$, where $\mathbf{Z}^q$ is the sequence of observations taken in the interval 0 to $\mathbf{q}$. The error in the estimate is denoted $\tilde{\mathbf{x}}(\mathbf{p}|\mathbf{q}) = \hat{\mathbf{x}}(\mathbf{p}|\mathbf{q}) - \mathbf{x}(\mathbf{p})$. Furthermore, the Kalman filter provides a recursive estimate of the covariance $\mathbf{P}(\mathbf{p}|\mathbf{q}) = E\left[\ \tilde{\mathbf{x}}(\mathbf{p}|\mathbf{q})\tilde{\mathbf{x}}(\mathbf{p}|\mathbf{q})^T|\mathbf{Z}^q\ \right]$ for the estimate $\hat{\mathbf{x}}(\mathbf{p}|\mathbf{q})$.

The Kalman filter algorithm proceeds recursively in three steps:

1. Prediction
2. Observation
3. Update

In the first step the Kalman filter at time t generates a prediction of the state estimate, the observation and the state estimate covariance for time $t + 1$. In the second step, at time $t + 1$, an observation of the i^{th} landmark is made and in the third step this observation is used to update the state estimate and the state estimate covariance.

7.3 SLAM in wireless networks

If Kalman filter based SLAM is to be applied to localization in wireless networks some considerations have to be made. The necessary considerations regard the changed characteristics of the landmarks and the technology used for obtaining observations.

Landmarks in wireless networks

The definition of a landmark so far is everything that an autonomous vehicle can make direct observations of. Naturally, the characteristics of such observations depend on the applied sensor and the environment. An example could be a robot with a sonar in an indoor environment where all obstacles such as chairs and walls would be directly identified as landmarks. Thus, in a scenario where a short-range wireless communication system is used as sensor by a MU this definition of landmarks will delimit the environment, of which a MU can make direct observations, to APs only.

It is therefore necessary to extend the definition of landmarks. This is done by introducing two groups of landmarks: observable landmarks and non-observable landmarks. The distinction here is that observable landmarks can be observed directly by measuring e.g. the RSSI value of a received signal, where as the non-observable landmark may only be observed indirectly through its affects on observations of observable landmarks.

Furthermore, SLAM traditionally uses landmarks for localization that are both stationary and passive. This means that the location of landmarks will not change over time and that landmarks do not have any means to make observations. This changes in wireless networks if APs are used as landmarks.

While the APs will remain stationary, they are not passive. Observations made in wireless networks will, as described in Chapter 3 on page 16, in this project be based on RSSI measurements. This entails that any AP also will be able to make observations, not only of MUs but also of other APs within its transmission range. This carries with it new possibilities.

If landmarks are able to make observations of other APs in transmission range, this may prove helpful in the localization process in the way that the relative distance between the observation can be introduced as additional knowledge in the system. However, SLAM cannot be used in a scenario consisting only of APs. This is based on the requirement of SLAM for non-stationary elements, which is a prerequisite not provided in such a scenario.

Observations in wireless networks

Another aspect that has to be considered is that observations in wireless networks are made with a technology that is not originally intended for localization. It has to be expected that the observation will be more inaccurate than in settings where an appropriate technology is used and also may not be available as often as desired.

Reservations regarding active landmarks

It should be noted that a system only can take advantages of active landmarks that is APs that are able to make observations, if it can access the measurements. If the localization system is placed on the MUs or on a dedicated unit in the network, these will have to be able to access the measurements on the APs. If this is not a default feature of the used APs, it may require modification of the AP firmware.

Although this may prove possible on most APs, it does not support the main goal with the work done in this project, as any additional work necessary to install localization in a wireless network reduces the PnP-capability of the localization system.

Chapter 8

Problem domain

This chapter will delimit the scope and define the goal of this project. First the state of the project is described, shortly summarizing the key subjects from the background part. Hereafter, a delimitation section will delimit the content of the project. Finally, the goal of the project is defined. The purpose of this chapter is to define the scope and goal of this project.

8.1 State of project

As described in the initial problem description, this project seeks to enhance localization of MUs in scenarios where a wireless short-range communication technology is used for positioning. Until now different aspect of localization utilizing a short-range technology have been investigated. Initially different localization techniques were described with the purpose to find a fitting localization technique for this project. The outcome of this was that radio based triangulation based on RSSI measurements is the most suitable localization technique for this project. Different short-range technologies were then investigated with the aim, first to chose which wireless technology that should be used and then to present different aspects of this technology that will be relevant later in the project. This analysis resulted in the choice of Bluetooth as the technology used in this project.

Next, characteristics of wireless channels were analysed to get a feeling of how they behave in indoor environments and what impact different aspects of the indoor environments have on it. Because the wireless channel affects the signal strength and thereby the localization precision it is necessary to understand its impact on the signal before localization can be done. Finally, mobility models and SLAM were analysed as possible methods to compensate for the negative impact on the signals propagation caused by the indoor environment and as methods to expand the localization capabilities.

Based on this background some general comments can be made regarding the further development of the project.

The outcome of this project should be a localization engine that is cable of locating MUs in a indoor environment similar to the one described in Chapter 2 on page 12 by utilizing

RSSI measurements from a Bluetooth network to do radio based triangulation. To expand the range of scenarios in which localization can be applied, the localization engine should furthermore utilize the described SLAM method in order to become more Plug-and-Play (PnP) capable by reducing needed effort in the system setup and installation phase.

The scenario introduced in Chapter 2 on page 12 furthermore describes that this localization engine is thought to be used in a museum. This scenario will therefore set some practical requirements to the system that have to be considered. If some of these requirements are not met, this could reduce the usability of the system. These practical requirements are listed in the following.

Localization precision: As described in the project scenario in Section 2.2 on page 13, information about a given item is presented audible or visually to the visitor when the visitor is located in the immediate vicinity of the item. Because these items can be placed close to each other it is necessary that the system can distinguish between relative close locations. It is assumed that a localization accuracy within a radius of 2 meter, as shown in Figure 8.1, is precise enough for this scenario. The method that shall be used to verify the accuracy is illustrated in Figure 8.2. The localization accuracy of the localization engine is determined in two steps. First, since no unambiguous reference between the best 'virtual map' and the 'real map' exists, the total distance of all estimated AP locations to all real AP locations is summed and a solution is found that minimizes the sum. This way the best mapping of the 'virtual map' to 'real map' is found. Then the distance between the estimated MU location and the real MU location is measured over time and the average deviation is defined as the system inaccuracy. The time frame for this procedure will depend on the scenario and is therefore not defined here.

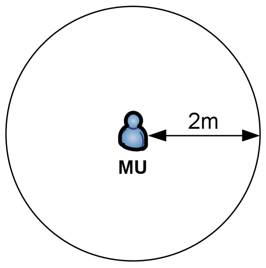


Figure 8.1: This figure shows the wanted accuracy of MU localization.

Model convergence: Since SLAM is used, a map of the environment is recursively build simultaneously to the localization process. The time this recursive procedure takes will depend on the implementation, the number of measurements, the movement patterns of MUs in the environment and the size of the environment. It is therefore difficult to set a upper limit for the time it takes for the map to converge without knowing the details of the above mentioned criteria.

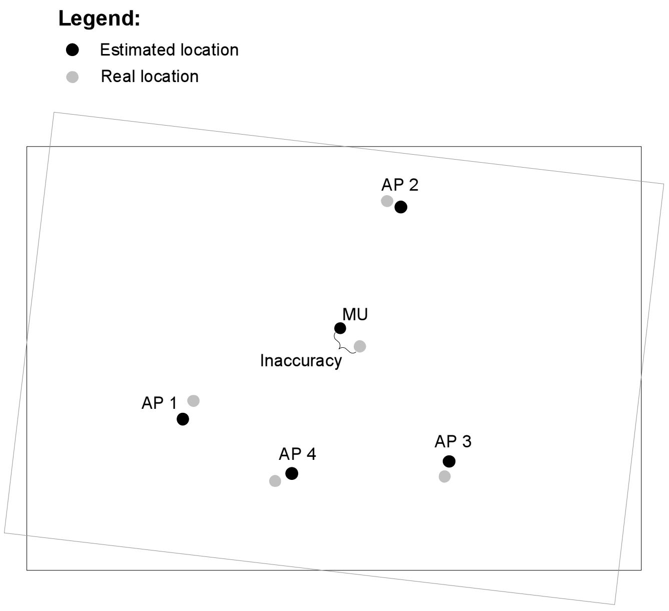


Figure 8.2: This figure illustrates how to calculate the accuracy of the localization of MUs. This is done by minimizing the total distances from estimated AP locations to real AP locations. The localization inaccuracy is then the average distance between the estimated MU location and the real location measured over.

8.2 Delimitations

Different delimitations are here made to limit the extent of this project and to keep the focus on the project goal. Furthermore different assumptions are made to make restrictions to the scenarios. Some of these restrictions are used to make the localization possible and others to simplify the complexity of the scenarios and thereby ease the localization process.

Delimitations and assumptions for this project are listed in the following supplied with a reasoning. The reason is written in *italic*.

Delimitations

1. Context service providing will not be a part of the system.
   - The main goal in this project is to enhance localization in indoor scenarios by utilizing SLAM. To limit the extend of the project and keep the focus on the main goal, context service providing will not be considered in the design and implementation of a solution in this report.

2. Fault-tolerance and security aspects will not be considered as a part of the system.
   - Because this project is a proof of concept there will be no focus on fault-tolerance in this project.

3. Only Bluetooth will be considered as short-range technologies.
   - Bluetooth is considered because it already is a widely deployed short-range technologies and has low implementation cost. This ensures that the performed research will have a bigger interest group and a larger impact.

4. There will not be developed any graphical user interface for the MU or for administrative purposes.
   - Because this project is a proof of concept there will be no focus on developing graphical user interfaces for neither MUs or for administrative purposes. Interaction with the system will take place through a basic interface that suites the requirements of the developers.

5. Localization will only be considered in the 2 dimensional (x,y) plane.
   - Because this project is a proof of concept, localization on multiple floors (e.g. different floors of a building) will not be considered, to avoid the increased complexity.

Assumptions regarding the scenario

1. Dynamic elements are not considered as a part of the scenario. Dynamic elements are defined to be: Doors, other people than one MU and soft partitions such as tables, chairs etc.
   - To reduce the complexity of the wireless channel, dynamic elements will not be considered as a part of the scenario.

2. Within the environment a MU is always able to be located by at least 4 AP.
   - Because previous research has shown that localization in a plane profits from at least three measurements to estimate the location, the coverage of at least three APs will be considered as a prerequisite for being able to perform localization at all. However, observability of the applied filter may cause problems if to few AP are available. Therefore the number of required APs within range of the MU is set to 4 in the project scenario. This number will have to be validated in the project.

3. All APs have omnidirectional antennas and are place at the same height.
   - To reduce the localization estimation complexity all AP antennas are assumed to be omnidirectional and are placed at the same hight.

8.3 Project description

A system that can provide localization of Mobile Units (MUs) using existing wireless short-range communication systems will no doubt have a huge potential in the near future. However, this localization still needs to be enhanced in order to provide a satisfactory level of service in terms of accuracy of the positioning and effort needed to establish such a system . This project will investigate if the Simultaneous Localization And Mapping (SLAM) approach can be used to solve both these problems.

The approach will be to develop a system for PnP localization of MUs in an indoor environment that is based on triangulation of RSSI measurements from a wireless short-range communication network, which in this project is Bluetooth. The system shall apply mobility models for the MUs and a Kalman filter based SLAM approach to provide a localization accuracy of 2 m according to the previous definition of the system localization accuracy.

The project will cover an implementation of the Simultaneous Localization and Mapping for Wireless Networks (SLAMWiN) system that will be evaluated based on measurements from both a environment emulator and a test bed using Bluetooth devices available in the WING-lab at Aalborg University. The measurements from the environment emulator will be used for development and refinement of the methods as well as a proof of concept, whereas the test bed is thought as a tool for demonstration purposes.

In order to make a final evaluation of the SLAMWiN system a series of evaluation scenarios will be needed. These scenarios will be developed so that they represent an increasingly difficult setting for performing localization. This is done by introducing two general parameters that describes the evaluation scenarios: Knowledge about physical characteristics of walls in the scenario and knowledge about the location of APs in the scenario. In the following knowledge about walls and APs in the scenario will be formulated as different levels of knowledge. The purpose with these levels are to describe scenarios that have characteristics which range from being highly bounded to having no restrictions. In this way an increasingly difficult setting for performing localization is described and the capability of the developed localization system may be explored in a stepwise approach moving through scenarios with continuously increasing complexity.

The levels of knowledge regarding walls and APs in the evaluation scenarios are illustrated in Figure 8.3 and are described in detail below. A level will always includes the knowledge from the levels below.

Knowledge levels regarding walls in the evaluation scenarios starting with the lowest level first:

No knowledge: At this knowledge level any combination of walls within the limits of the project delimitation is possible.

All walls are either $\|$ or $\perp$: At this level all walls will either be parallel or orthogonal. This knowledge is reflected in many real life scenarios. It provides better conditions for estimating the position of walls by setting rules for their relationship to each other.

All wall are equally thick: At this level all walls are equally thick. The conditions makes it easier to generalize knowledge regarding walls.

Known thickness of walls: At this level the thickness of walls in the scenario is assumed known. This knowledge significantly simplifies the scenarios providing better conditions for estimating the number and position of walls.

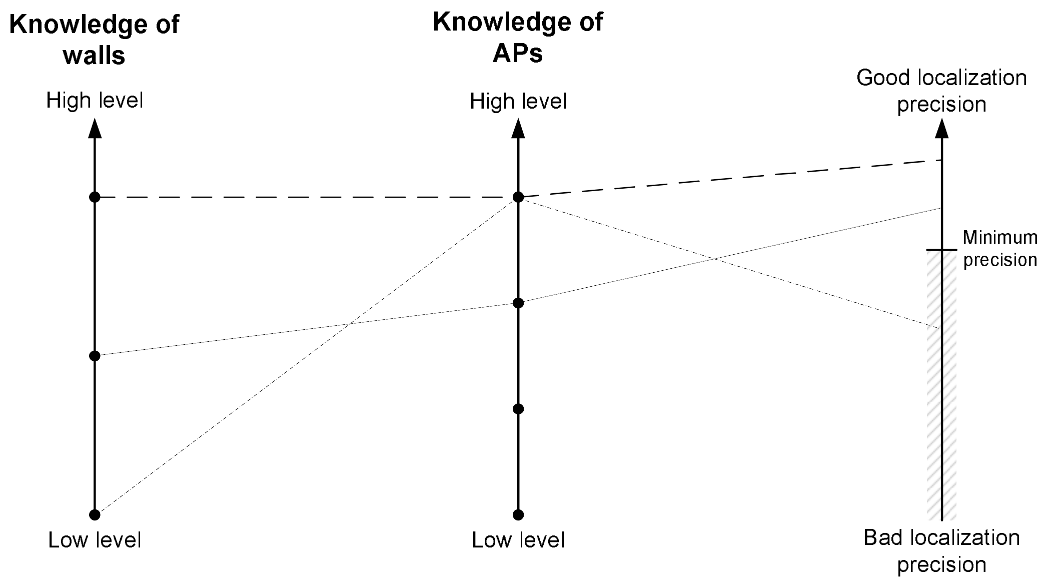


Figure 8.3: Each combination of knowledge of walls and APs will give a certain localization precision.

Knowledge levels regarding APs in the evaluation scenarios starting with the lowest level first:

No knowledge: At this level no knowledge regarding the position of APs is available.

Two AP on the same wall: At this level two known APs will have to be placed on the same wall. This will in combination with the scenario restriction regarding $\parallel$ or $\perp$ walls provide favorable conditions for estimating the position of walls.

Known number of walls between APs: At this level the number of walls between any pair of APs is known. This sets some restrictions to the PnP capability of the system, but is expected to provide good conditions for estimating the position of APs.

Known AP to AP distance: At this level the exact distance between all AP is known. This does not go well together with the project goal of developing a PnP system. However, it is included to ensure a basic start-scenario that will provide results to which results from the more difficult scenario can be compared. It should be noted that knowing the AP to AP distances corresponds to knowing the location of all APs since this knowledge can be directly derived.

The approach to exploring the capabilities of the SLAMWiN system is described in steps below. It is assumed that knowledge about the location of APs is more likely to be available than knowledge regarding the location of walls. Thus, the approach implements a policy where situations with varying level of knowledge regarding walls are evaluated for each level of knowledge regarding AP locations.

1. Start with a high level of knowledge regarding walls and the position of APs and evaluate the achieved localization accuracy against the desired minimum accuracy.

2. Decrease the level of knowledge regarding walls to the next lower level and evaluate the achieved localization accuracy against the desired minimum accuracy. This is repeated until the lowest level is reached or the achieved localization accuracy is worse than the desired minimum accuracy.

3. Set the level of knowledge regarding walls to the highest level and decrease the level of knowledge regarding the position of APs to the next lower level. Then evaluate the achieved localization accuracy against the desired minimum accuracy.

4. Repeat step 2 with the new level of knowledge regarding APs.

5. Step 3 and 4 are repeated until either the lowest level is reached for both knowledge regarding walls and the position of APs or a scenario with the highest level of knowledge regarding the position of APs fails to provide an localization accuracy that is better than the desired minimum accuracy.

The described steps will not be used in the development of the localization system. The development will be described later in the relevant chapters.

Part II

Analysis of SLAMWiN

Chapter 9

Introduction to SLAMWiN analysis

This chapter will first describe the general approach that has been chosen by the project group. This basically covers how the analysis and development is tackled and what it incorporates and why. Secondly the chapter will introduce the Analysis of SLAMWiN part by describing the purpose and the structure of the different chapters in this part. By doing so, it will provide an initial overview of the analysis of SLAMWiN and support the readability of the following chapters.

The purpose of this chapter is to introduce and present arguments for the chosen development approach.

9.1 Development approach

This section will describe the general approach that has been chosen by the project group. It accounts for a stage-based approach and will in order to do so anticipate the model for the overall system architecture deduced in the succeeding chapter. This is done in order to give a relevant illustration of the stages.

The chosen approach will make use of a development strategy where intermediate goals are defined throughout the analysis and development. This way the development is broken down into smaller parts that may be analysed iteratively. Each part, or stage, will thus be defined by its own goal, entailing its own requirements and success criteria. The development strategy also aims at identifying and describing interfaces between different components of the SLAMWiN implementation as early in the development phase as possible in order to achieve a well structured implementation that:

- Allows easier development of separate components.
- Provides the possibility to upgrade separate components later on without having to change larger parts of the SLAMWiN implementation.

It is furthermore expected that this approach will ensure useful results early in the process and that it thus will be possible continuously to evaluate the progress of the project and the quality of the developed system.

Development scope

The goal of this project is to investigate if SLAM can be used to enhanced localization solutions for indoor environments based on wireless short-range communication technologies. This investigation has so far included an analysis of several aspects of localization in general described in the Background part of this report. The next step is to perform a more specific analysis of elements that may be applied directly in the development of a system that combines SLAM and localization based on wireless short-range communication technologies into a SLAMWiN system.

A quick brainstorm indicates that the following elements to some extend will be needed in the development of a SLAMWiN system:

- A filter that is able to convert measurements into location estimates.
- A modified version of SLAM.
- Mobility models that describe the MU behaviour.
- An environment emulator that supports easy and fast testing and evaluation of iterations during the development.

This project constitutes the first investigation of the problem whether SLAM and localization based on wireless short-range communication technologies may be combined and involves as such a series of tasks that have previously been paid only little attention. Consequently, only limited research is available on the topic and thus the outcome of the project is uncertain as especially the second element introduced above carries considerable uncertainty into the development process.

To oblige these conditions, the development of a solution to SLAMWiN will follow the two development strategies that will be described in the following.

Step-by-step development

In order to take care of the uncertain outcome of the project the development is structured into steps. This entails that the problem is broken down into smaller parts that may be treated one at a time in separate development steps. The steps are chosen so that they each are defined by there own requirements and success criteria and so that they represent a clear functional enhancement to the previous step. There should thus only be a limited number of steps and each step should represent an extension to the previous step that adds new functionality on system level.

There are several reasons for applying a step-by-step development approach among which the following have been the most crucial:

Complexity increased in steps: The first reason for using steps is that they allow to treat problems one at a time and thus allow to delimit complexity in the early steps. This supports the development by keeping focus, while iteratively being able to add more functionality and complexity in an iterative manner.

Iterative evaluation of approach: The next reason for using steps is that they provide results early in the development process. This way the system performance can be measured throughout the development, which allows for continuous evaluation of the project approach. The chances for success of the complete system can thus be reevaluated for each step and design choices regarding later steps may be changed according to the gained knowledge.

Possibility for sub-conclusions: The last reason for this approach is less minded on the development process and more on the project circumstances. This project is conducted within a fixed time frame and will, since it is a student project, have to present a conclusion on the possibility for applying SLAM in wireless communication networks. Having the uncertainty of the project outcome in mind, the step based approach thus also meets the desire of the project group to be able to provide a conclusion based only on parts of the approach as these may be evaluated before the development of a entire SLAMWiN system is completed.

An example of the step-based approach is illustrated in the following where the overall system architecture deduced in the succeeding chapter is anticipated. Figure 9.1 shows simplified version a model of the complete SLAMWiN system. Here e.g. the filter is identified as a separate development step as it may be described with its own requirements and goals and provides a well-defined functionality on system level. It thus meets the definition of a development step introduced above.

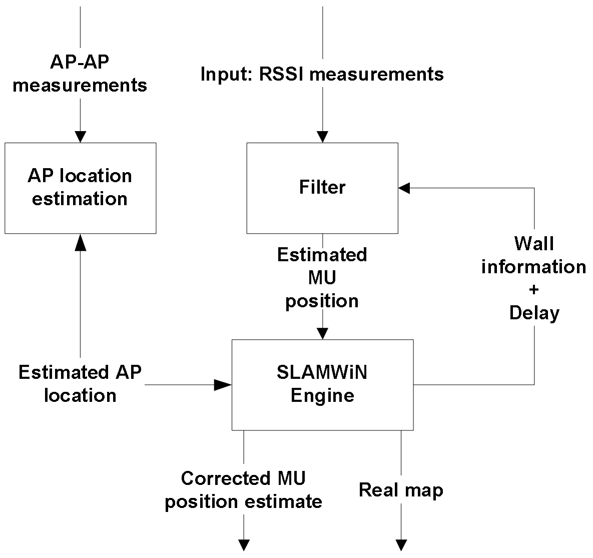


Figure 9.1: This figure shows the model for the complete SLAMWiN system. The filter represents an example of one step of the development of the system

Modular design

On the more detailed level of the development another strategy is followed: Modular design. The modular design strategy dictates that the developed SLAMWiN system should be arranged so that critical components are encapsulated within it. The purpose of this strategy is to ensure that they are interchangeable.

The advantage of the modular design strategy and the resulting interchangeability of components thus lies in the possibility to delimit complexity in certain parts of the development in favor of developing a full SLAMWiN solution.

9.2 SLAMWiN analysis structure

This section will introduce the chapters that make out the SLAMWiN analysis in order to provide an overview of the development steps.

System architecture

The first chapter will describe how the SLAMWiN problem is approached on a conceptual level. It discusses what tasks that are necessary and why and how these may best be combined into a solution to the problem at hand. Several ideas for SLAMWiN will be introduced to document the evolution from an initial idea into the final approach.

The outcome of the chapter will be a description of the overall system architecture. This description will cover the system inputs and outputs and the high level interfaces.

Filter

First step of the development of a solution to SLAMWiN will concern the filter. The chapter will shortly introduce Kalman filters in general and then document the analysis and development of a Kalman filter suited for SLAMWiN.

The analysis of the filter will, based on knowledge about the original Kalman filter, treat Extended Kalman filters and a special version of these, the switching Kalman filters. Two suited filters are selected and evaluated in order to find the best performing solution.

For the filter evaluation a set of scenarios are derived and the filters are evaluated based on their capabilities for both tracking and map building.

SLAMWiN Core

Second step will engage the core problem of a solution to SLAMWiN: How and when should SLAMWiN compensate for walls? This is done while delimiting all parts of SLAMWiN that are not vital for performing basic tracking based on RSSI measurements in environments containing APs with known locations and walls.

The chapter will document the development of a SLAMWiN Core that is limited to providing tracking capabilities of MUs using only knowledge regarding the scenario obtained by the MU itself in scenarios with known walls and AP positions.

The SLAMWiN Core and its ability to detect walls and react on the detection will be evaluated before the development of a more comprehensive implementation of SLAMWiN is approached.

SLAMWiN Knowledge sharing

The SLAMWiN Core system developed in the previous chapter presents a solution for performing SLAMWiN for a scenario based on a single MU. This forms a contrast to the project scenario, which potentially contains several MUs. This chapter will thus introduce enhancements to the approach that will extend SLAMWiN so that it can take advantage of multiple MUs in a scenario. Extending SLAMWiN in this way includes the merging of wall information obtained by different MUs passing through a covered area and the ability of MUs to use wall information from previous MUs. Furthermore, this chapter will introduce a approach to estimate the initial location of MUs.

As in the previous chapter, the SLAMWiN system will again be evaluated before proceeding to the next step. This time the storage of information regarding walls and the use of the map will be the main focus of the evaluation.

Estimation of AP positions

After having developed a SLAMWiN system that employs a map in scenarios with known AP locations, this chapter will document a further enhancement that does away with the need of known AP locations. It thus approach a central problem of developing not only a SLAMWiN system, but a fully PnP capable SLAMWiN system.

The final solution to SLAMWiN described in this chapter represents the main goal of this project and will thus be evaluated according to the parameters and the approach described in the problem domain.

Build real map

This chapter will describe how a map of the environment is build based on information from the SLAMWiN Knowledge sharing component. This map will give an estimate of where possible walls are located in the environment and it will be used as input to the advanced mobility model component to enhance localization accuracy.

Advanced mobility models

The last chapter of this part will discuss possible methods to enhance the location accuracy provided by the SLAMWiN implementation developed in the previous chapters. The introduced methods will be advanced mobility models that can be used to enhance the systems ability to estimate future locations.

Chapter 10

System architecture

This chapter will introduce the chosen approach on a conceptual level and describe the development on the system level. It will discuss what tasks that are necessary and why and how these may best be combined into a solution to SLAMWiN. Several ideas for SLAMWiN will be introduced which together document the evolution from an initial idea into the final approach.

The purpose of the chapter is to describe the overall system architecture covering the system inputs and outputs and the high level interfaces.

10.1 SLAMWiN considerations

The background part of this report has introduced several aspects of localization based on wireless short-range communication technologies. These have been used as an inspiration for the approach to SLAMWiN and the following will describe the process that have lead to the final approach.

However, before going into describing the design process that has lead to the final approach, some issues regarding SLAM and necessary modifications of it in order to make it work in wireless networks will be stated. Moving SLAM into wireless networks is not trivial, since several of the fundamental principles of traditional SLAM do not apply for, or can not be directly transferred, to wireless networks. A list of the key issues will be described here in order to illustrate the problems that will have to be considered in the development of SLAMWiN.

The observer: In traditional SLAM, observations are made from the moving part, e.g. a robot, which also is the part that is interested in the information gained through SLAM. The robot makes observations, which it needs in order to determine its own relative location in the environment.

This may not necessarily be the case in a SLAMWiN system. APs may just as well make observations of a MU moving through the environment, as the MU can make observations of APs in the environment. Considering factors such as power consumption on the MU and scalability of the SLAMWiN solution, a design choice could very likely end up favoring a centralized SLAMWiN solution placed on a server in the network.

This would add another dimension to SLAM as the number of observers increases, since all APs may function as observers. Observations could in that case not only be made of the relation between the moving part and the environment, but also of the relation between different points in the environment as APs may observe each other.

Observations: Also the kind of observations that are made will not be the same in traditional SLAM and SLAMWiN. In the traditional case observations are made directly of the environment. A robot can directly observe landmarks in the environment to which it wishes to determine its relative location.

Although a MU moving through a SLAMWiN scenario is able to determine its relative location based on observations between the MU and the APs, it cannot directly observe walls in the environment. This represents a problem, as walls are an equally important part of the environment and because they furthermore influence on the observations used to make the MU location estimate.

Known motion: A fundamental part of traditional SLAM is that the robot that moves through the unknown environment has knowledge of its own motion, which e.g. comes from sensors on its wheels. Observation of the environment made by the robot at different times can thus be linked together since the robot knows the relation between the different positions where the observations are made from. This enables the robot iteratively to eliminate a large part of the uncertainty of its own relative location estimate as more observations become available.

Unfortunately this knowledge of the MU's motion is not available in SLAMWiN and the system will have to compensate for this.

Available technology: The last issue described here is that SLAMWiN will have to work with significantly noisier measurements. SLAM has the advantage of typically being applied in scenarios where technologies are used for making observations that are designed for that specific task. It will thus be able to rely on a technology such as sonar or radar that is designed for making distance estimates and following produces better and less noisy observations.

Since SLAMWiN is developed as an extension to an already existing system that is designed for an altogether different purpose, the available technology in most cases will not be designed for making distance estimates. Observations made by such technologies will most likely be more inaccurate.

SLAMWiN placement in existing systems

As described in the list of key issues introduced by migrating SLAM into SLAMWiN, SLAMWiN provides more possibilities for making observation. The first step of designing a solution to SLAMWiN should thus be to consider the function of a SLAMWiN system and where this function may best be placed in an existing system. This should then also include considering the optimal position of the observer, or the observers.

As described in the Scenario chapter (Chapter 2 on page 12) SLAMWiN is thought to be integrated into an existing wireless communication system. As such, it may be integrated in two different ways. It can either be integrated as a centralized solution placed in the network, or as a decentralized solution placed on every MU. Both approaches have advantages and disadvantages and these will have to be weighted against each other. The following presents four different approaches that have been considered. Two for each solution with the observer placed at either the MUs or the APs respectively.

Decentralized SLAMWiN solution: The decentralized solution for integrating SLAMWiN into an existing wireless communication system is based on running a SLAMWiN client at every MU that wishes to make use of SLAMWiN. This can be done in two different ways depending on the desired position of the observer. Figure 10.1 illustrates the case where the observer is placed on the MU and Figure 10.2 illustrates the case where the observer is placed at the APs in the scenario.

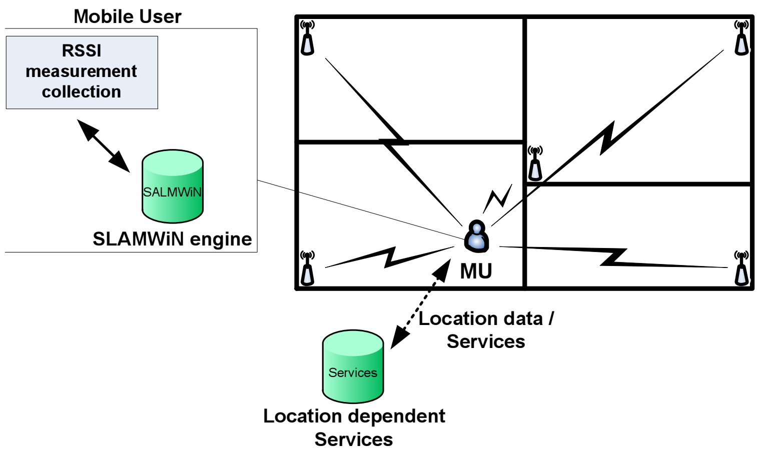


Figure 10.1: This figure illustrates an example of a decentralized SLAMWiN solution with the MU as observer.

Depending on the chosen setup, the decentralized SLAMWiN solutions has different advantages and disadvantages. These are described separately in the following:

Observer placed at the MU :

Advantages :

- Use of the wireless channel is minimized as no measurements or location data has to be exchanged. This avoids congestion in the network due to SLAMWiN.
- Power on the MU is saved as the wireless interface is only used for taking measurements. This ensure a longer lifetime of MUs which typically run on batteries and thus only have limited power available.

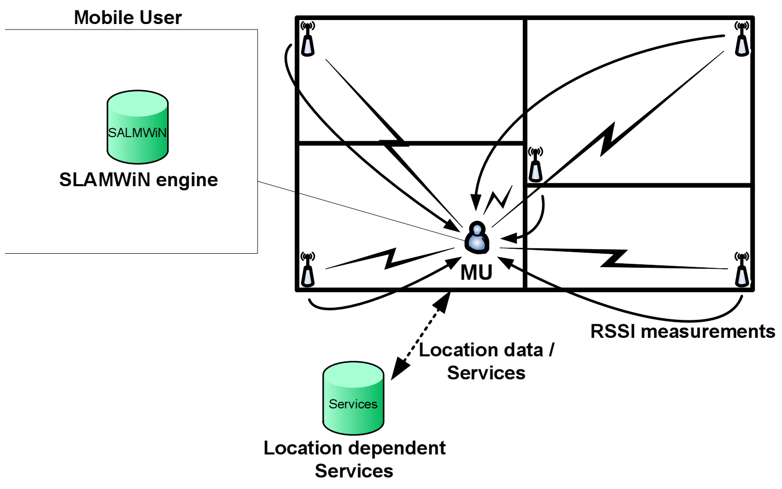


Figure 10.2: This figure illustrates an example of a decentralized SLAMWiN solution with the APs as observers.

- Location estimation on the MUs is independent of other devices in the network. Only requirement are the APs of which it can take RSSI measurements.

Disadvantages :

- Only MU-to-AP measurements are available as only the MU may make observations. This cuts the SLAMWiN engine off from AP-to-AP measurements which can provide additional information about the environment.
- Much computational power is needed for the SLAMWiN engine running on the MU. This additional computational power may not be available and upgrades to new hardware are expensive and thus unwanted.
- The increased use of computational power by the SLAMWiN engine entails an increase in the power consumption of MUs. This may cancel out the advantage of saved power through the reduced use of the wireless interface.
- As each MU calculates its own location and since this location only is made available on the MU itself, no advantages can be made of shared information regarding the environment between different MUs.
- Even though the estimated location of a MU provided by the SLAMWiN engine represents a service in itself, it will in many cases be used as context information by other services. As such the information will have to be made available to these services which may be located in the network. It may therefore be necessary to transmit the location information from the MU to other entities in the network entailing a increased use of the wireless channel.

Observer placed at the APs :

Advantages :

- Both AP-to-MU and AP-to-AP measurements are available as the APs which makes the observations. This provides a better knowledge base in the initial phase of the location estimation process as measurements of a larger part of the environment are available. Furthermore AP-to-AP measurements are extremely useful in scenarios where the location of APs are unknown and has to be estimated by SLAMWiN.

Disadvantages :

- The wireless channel is used for both taking measurements and for periodically transmitting these to the MUs. As it is preferable that the SLAMWiN engine on the MUs can use the newest RSSI measurements for making location estimates the frequency of transmitting RSSI measurements will be relatively high. Furthermore, each MU requires its own measurements and RSSI measurements will thus have to be unicasted to each MU. This solution does not scale very well with respect to needed bandwidth on the wireless channel.
- As in the previous setup, the SLAMWiN engine is located on the MUs and will thus increase the power consumption and needed computational power on the device.
- Since this setup also is based on a decentralized approach to SLAMWiN it also cannot take advantage of shared information regarding the environment.
- Furthermore, location information about the MUs may also in this setup be needed elsewhere in the network than on the MU where it is estimated.

Centralized SLAMWiN solution: The centralized SLAMWiN solution applies a single SLAMWiN engine located on a device in the network e.g. a dedicated SLAMWiN server as illustrated in the following figures. Again the observer may be placed at either the MUs or the APs. This is illustrated in Figure 10.3 and 10.4.

The advantages and disadvantages of the two setups are listed in the following.

Observer placed at the MU :

Advantages :

- All computations needed in the SLAMWiN engine is performed on a centralized server. This solves the problems with limited computational power on the MUs and furthermore does not drain the batteries on the MUs.
- Information regarding the environment obtained from a MU may easily be used in the location estimation of another MU. This can be done without burdening the wireless channel and provides better conditions for location estimation for all MUs.

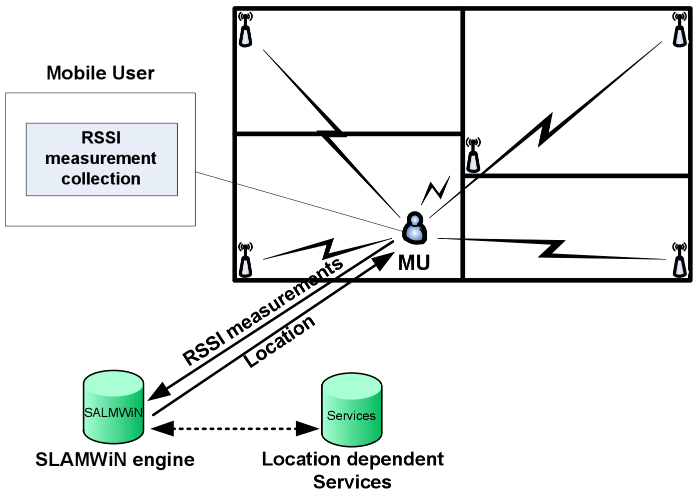


Figure 10.3: This figure illustrates an example of a centralized SLAMWiN solution with the MU as observers.

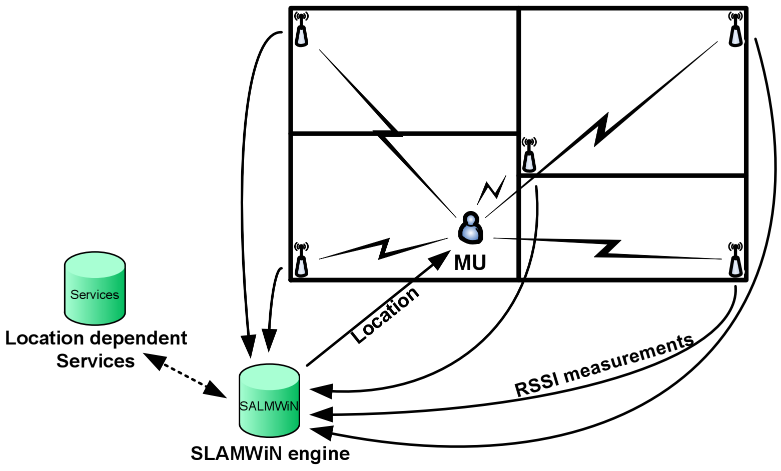


Figure 10.4: This figure illustrates an example of a centralized SLAMWiN solution with the APs as observers.

- Network services that can make use of the MU location can request these from a device in the network which reduces the load on the wireless channel.

Disadvantages :

- RSSI measurements and location information must be exchanged between the MU and the SLAMWiN engine in the network. This will burden the wireless channel.
- Furthermore the frequent use of the wireless interface will increase power consumption on the MU.
- Since the SLAMWiN engine is located at one server only and initially

is designed without any redundancy, this introduces a reliability problem as the SLAMWiN engine is vital for making location information available to MUs in the scenario.

Observer placed at the APs :

Advantages :

- Both AP-to-MU and AP-to-AP measurements are available as the APs are observers. This provides a better knowledge base in the initial phase of the location estimation process as measurements of a larger part of the environment are available. Furthermore AP-to-AP measurements are extremely useful in scenarios where the location of APs has to be estimated by SLAMWiN.
- As the SLAMWiN engine is located on a device in the network and the APs are used to make measurements no transmission of RSSI measurements are transmitted over the wireless channel. Either only the estimated location of each MU has to be transmitted from the SLAMWiN engine to the MUs, or locatino estimates can be handed directly to location dependend services in the network. This way no location estimates need to be transmitted.
- All computations needed in the SLAMWiN engine is performed on a centralized server. This solves problems with limited computational power on the MUs and furthermore does not drain the batteries on the MUs.
- Information regarding the environment obtained from a MU may easily be used in the location estimation of another MU. This can be done without burdening the wireless channel and provides better conditions for location estimation for all MUs.

Disadvantages :

- Since the SLAMWiN engine is located at one server only and initially is designed without any redundancy, this introduces a reliability problem as the SLAMWiN engine is vital for making location information available to MUs in the scenario.

Central factors in the choice of the best approach are scalability, life time of the MU and performance of the SLAMWiN system.

With respect to scalability especially the use of the wireless channel should be limited. The wireless channel is already used for making the RSSI measurements and should not be burdened further as this resource most likely also will be needed for other services that are introduced together with SLAMWiN such as video and information services.

Also the time a MU can survive before it has to recharge its batteries must be considered. Reducing the use of the wireless interface and the computational effort needed from the MU have a large positive influence on the life time.

Finally, the performance of the SLAMWiN system also will depend on its ability to merge the measurement history of several MUs into one knowledge base.

Based on these consideration the centralized solution with the observer placed at the APs is chosen as it provides the best conditions for a SLAMWiN system.

10.2 SLAMWiN development history

This subsection will introduce the idea that forms the base of the SLAMWiN solution developed in this project. Since the idea throughout the project has been re-evaluated several times, various iterations of it exist. In order to provide a better understanding of the final approach, selected iterations will be described here starting with the initial idea. For each of the following iterations the changes to the previous will be described and arguments for the changes will be stated.

Since the design of SLAMWiN was conducted in parallel to prototype implementations of smaller parts of SLAMWiN, several other reasons for changing the approach, beyond the major ones described in the following, exist. These have not been included as they mainly regard more expedient implementation approaches.

Initial SLAMWiN prototype

Initially the solution to SLAMWiN was build around a virtual map that should merge available knowledge of AP locations and the estimated current MU location and location history in order to make guesses of the position of walls in the environment. The virtual map was thought to be divided into cells and the MU location should only be registered on a per cell basis. The influence of walls on the mobility behaviour of MUs was not initially a part of the virtual map.

The following bullets state general features that the virtual map was thought to provide:

- The domain used in the virtual map should apply Cartesian coordinates in two dimensions (x, y).
- The map should have consisted of a grid that was put over the entire simulation/test environment and was limited to the area that was covered by the APs. A maximum range of 100 m was assumed. The grid-size should have been based on the desired localization precision. Initially a cell size of 33 time 33 cm was proposed that was allowed to change during development if it proved useful, or necessary.
- One predetermined AP should have been chosen as reference point $((0, 0))$ for the grid and the localization. Another predetermined AP should have been used to define the $x - axis$ and a third AP should define the side of the x-axis.
- Knowledge about walls was thought to be stored in the cells. Each cell should have contained information about which APs that were visible from its location and

whether there was one or more walls between the cells position and the position of each of the visible APs, or not.

- Knowledge about which APs that were visible from a cell should have been provided by passing MUs. The list of APs that a given MU was covered by should have been transferred to the cell while passing it.

- It was assumed that the environment which the virtual map described initially did not have any walls and thus all cells should have assumed line of sight to all APs if not told otherwise. Knowledge of estimated wall positions should have been added during the map building process.

The first step of building the virtual map should have been to place the APs in the map. If the location of the APs was known, they could simply have been added according to the coordinate (x, y) that described their location. If the locations were unknown they would have to be estimated. This should have been done by using AP-to-AP measurements to place the APs in a way that best fitted the estimated distances between each pair of APs. A best fit should have been used, since it would not necessarily have been possible to place the APs so that the distances between all pairs of APs equals the estimated distance. This problem was caused by the initial assumption of no walls in the environment.

After that the APs would have been placed, a MU should have been introduced into the virtual map. The MU should have been tracked using a filter and there should initially have been made the assumption that no walls existed in the environment.

The central idea was to introduce 'shadows' or 'shaded areas'. These should have been defined as areas behind obstructions that caused non-LoS to a given AP. Based on knowledge about the wireless radio channel it was expected that walls could be detected through 'jumps' in the mean received signal strength from the MU on a given AP. The idea was based on two observations:

- Walls typically have a 'shadow' on the opposite site of a signal source in which the signal either is less strong or non existent. The shadow, or shaded area, is defined as the area behind walls where there is no LoS to the AP.

- When a MU moves through a door, an AP will experience a noticeable change in the mean received signal strength from the MU. Depending on whether the MU moves towards or away from the side of the wall where the AP is located, the mean signal strength will either rise or fall.

As the MU would move through the environment it should have been observed by the APs. The estimated location of the MU should then have been matched to a cell in the virtual map and knowledge about which APs were visible from the cell should have been stored in it. As the MU moved along, one of the APs observing it could then at some point have experienced a noticeable change in the Mean Received Signal Strength (MRSS). This event could then have been assumed to be the end, or beginning, of a wall somewhere between the AP and the MU. If the MRSS had risen the MU had moved out of a shadow behind a wall and if the MRSS had dropped the MU had moved into a shadow.

In the first case where the MU moved out of a shadow all previous measurements from the AP that experiences the MRSS rise would have had to be modified according to the new knowledge. This means that the measurement history of the MU would have to be changed and a new path should have been computed according to the newly found knowledge of a wall. An example of how this was thought is illustrated in Figure 10.5. The new path would thus have been closer to the AP, since the effects of the wall on the signal strength would have been considered.

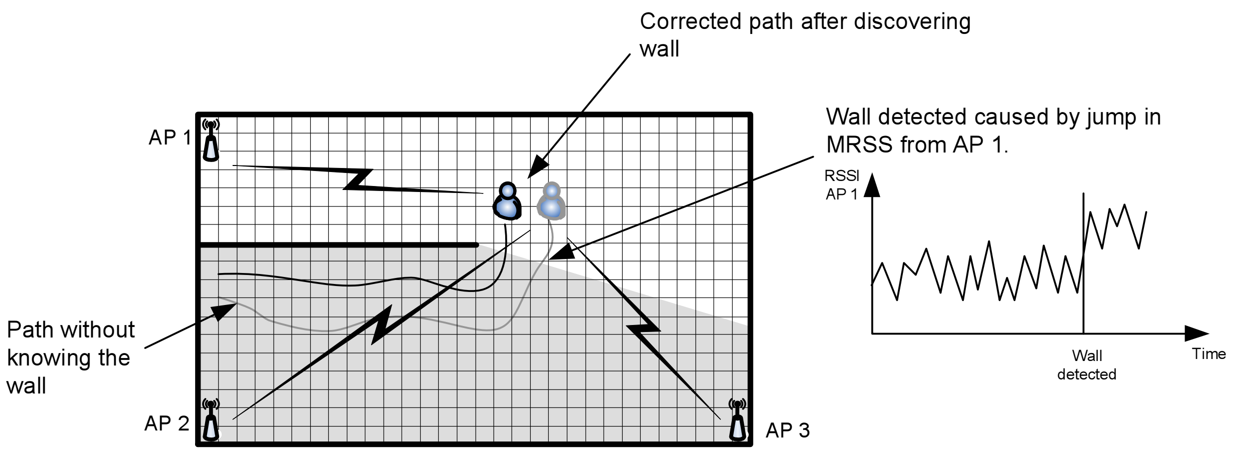

Corrected path after discovering wall
Wall detected caused by jump in MRSS from AP 1.
AP 1
AP 2
AP 3
Path without knowing the wall
RSSI AP 1
Wall detected
Time

Figure 10.5: A new corrected path should have been computed according to the newly found knowledge of a wall.

In the opposite case where a MU would have moved into a shadow, future measurements from the AP that experienced the drop in the MRSS would have to have respected the possible existence of a wall between the AP and the MU. Again, the effects of the wall would have to be considered when the MU path had been estimated.

In order for the MU not to have to move through every cell in the map before all cells would have gained knowledge of possible walls, knowledge of possible walls was thought to be radiated outwards from a MU path. This effect was thought to work much like a shadow in which knowledge of walls would have been distributed to all cells in the shadow. In Figure 10.6 the previous example is modified according to this.

Figure 10.7 gives an overview of the first iteration of the SLAMWiN system, showing the main components and their interaction. It shows that the *Virtual Map* was designed as the central part of the system containing and handling all information about MU locations as well as environmental information. A *Filter* was thought to be responsible for the process of estimating the MU position, however, the modification of measurements based on possible knowledge of walls was chosen to be handled outside the filter in a *RSSI manipulator* for increased simplicity. The *SLAM engine* was at this point limited to a component that made modifications on the virtual map updating it with new position estimates of MUs. A further component was introduced that monitored transitions between cells in the map

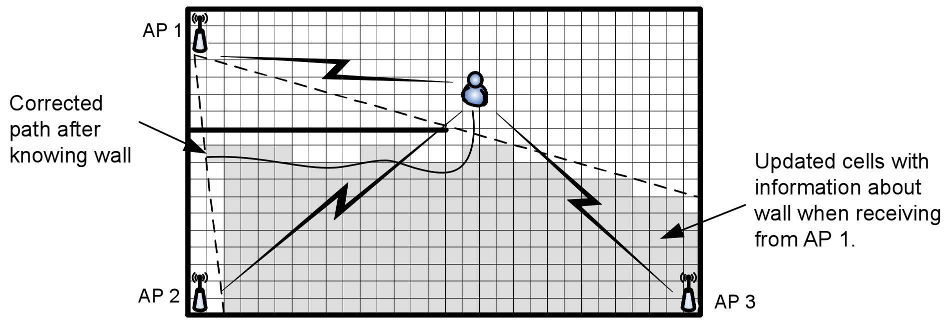

AP 1
Corrected path after knowing wall
Updated cells with information about wall when receiving from AP 1.
AP 2
AP 3

Figure 10.6: All cells in the shadow should have been updated with information about a shadowing wall and which AP the measurement was taken from.

in order to be able to provide transition probabilities between the cells that should have been used in an extension to SLAMWiN. This extension was *Advanced mobility models* that should have applied more advanced mobility models to the estimated position of MUs in order to increase the location accuracy of the system. The last two components *AP location estimation* and *Build real map* were introduced for estimating initial AP locations as previously described and for providing a real map based on the information stored in the virtual map. The *Build real map* component was introduced mostly as a 'nice to have' feature, since it was not a necessary component for making position estimates of MUs, but instead utilized information already available in the system.

The above described method should have been extended in the way that one registration of a possible wall would not have been enough to make a MU consider the effects of it e.g. recalculate its measurement history in order to take the wall into consideration for the already moved path. A cell should have stored the number of times MUs told it about possible walls between it and a certain AP. It should then furthermore have stored the number of time MU passed it and while not telling it about possible wall between it and the same AP. This would have given some indication of the likelihood of a wall in fact being present.

The described initial method was also thought to work for scenarios where the initial location of APs in the scenario was unknown. Here the initial location of the APs should have been estimated as described above and this location should then have been used until likely walls were identified with a certain confidence. It would then have been necessary to re-estimate the location of the APs according to the new knowledge of walls in the scenario. A location change of APs would furthermore have influenced all previously estimated MU paths and information that had been radiated outwards to other cells would have become obsolete.

The re-calculation of shadows was thought to either take place in the same virtual map, or to be spawned in a parallel virtual map that was initiated when the location change of APs had been registered. This way the original virtual map could have continued to be used for tracking until the new virtual map had converged.

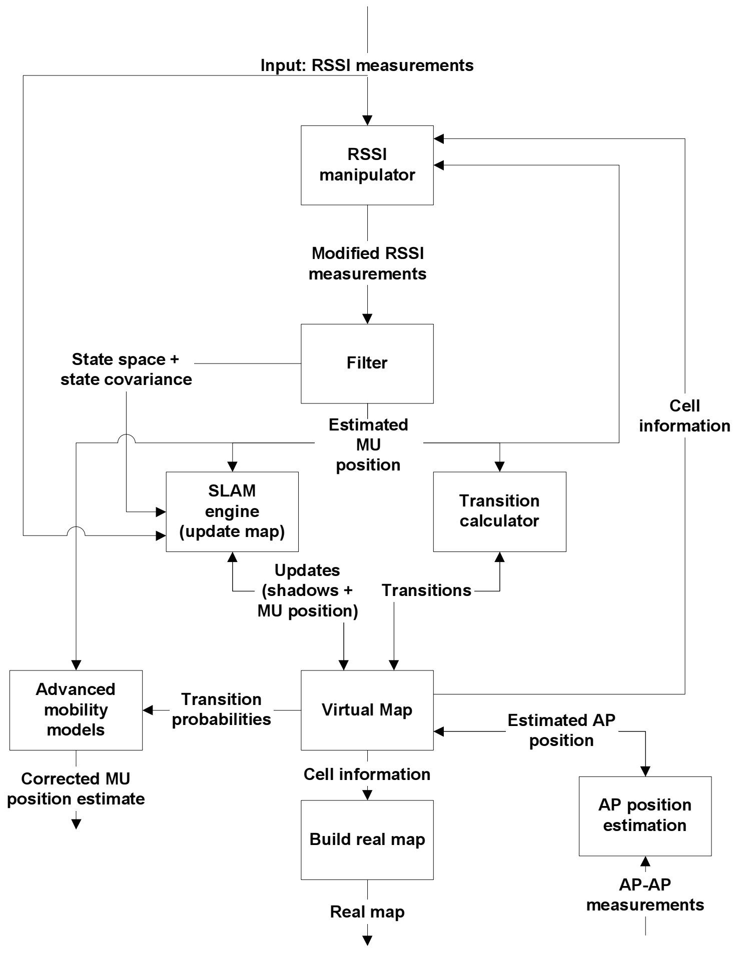


Figure 10.7: This figure illustrates the component and their interactions in the first iteration of SLAMWiN.

SLAMWiN prototype applying shadows

The main change from the first iteration to the second iteration of SLAMWiN was that it did away with the fixed cells used in the virtual map. This change was motivated by the large amount of information that had to be stored in the cells. As much of the information stored in the cells was identical for all cells affected by the same wall, a new kind of cells were introduced: shadows.

The shadows were not defined by a grid created prior to the tracking, as cells had been before. Instead the shadows were defined as areas with identical information regarding a certain AP. A shadow was thus towards the AP bounded by the path travelled by the MU and to the sides bounded by the line that was span by the AP and the two points were a jump had been detected. An example of this is shown in Figure 10.8. The two points where a jump was detected were marked with flags. The purpose of these flags was to have a structure for storing information about the jump points. Examples of this information is the location and size of the jump. The size indicated how much the MRSS was reduced

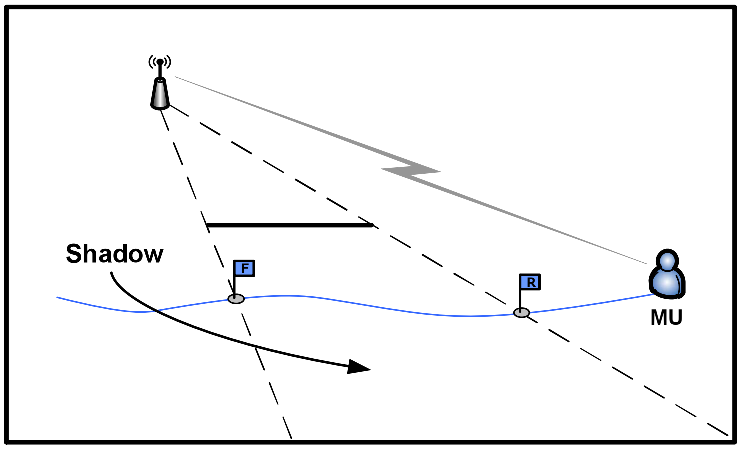


Figure 10.8: The shadows used in the 2^{nd} iteration of SLAMWiN are defined as a area with identical information regarding a certain AP.

due to the wall. This information was valid for the complete shadow. Shadows were not considered having a maximum range. Any point further away than the MU path generating a given shadow between two flags that it has experienced was thus considered being in the shadow and had to consider the effect of the wall.

The new definition of shadows simplified the SLAMWiN approach and had several advantages over the first definition:

1. The relation between shadows and the area obstructed by a wall felt more natural and thus easier to work with while discussing the system design.

2. The number of shadows and thus also the amount of information that needed to be stored was reduced considerably.

3. The description of shadow areas became more precise, since the smallest possible granularity was not limited by the fixed cell size in the virtual map.

Figure 10.9 shows the changes in the component diagram of the SLAMWiN system entailed by the new definition of shadows. It should be noted that since the new appraoch relies on shadows and not on a grid of cells, the calculation of transition probabilities has not been included in the 2^{nd} iteration.

SLAMWiN prototype with polar coordinates

The third and last iteration of the SLAMWiN idea represented a restructuring of the components into a more intuitive order based the step-based design approaches. The components were thus redefined so that they live up to the definition of a step defined in the previous chapter: Steps are chosen so that they are defined by there own requirements and success criteria and so that they represent a clear functional enhancement to the previous step. Figure 10.10 illustrates the new component structure.

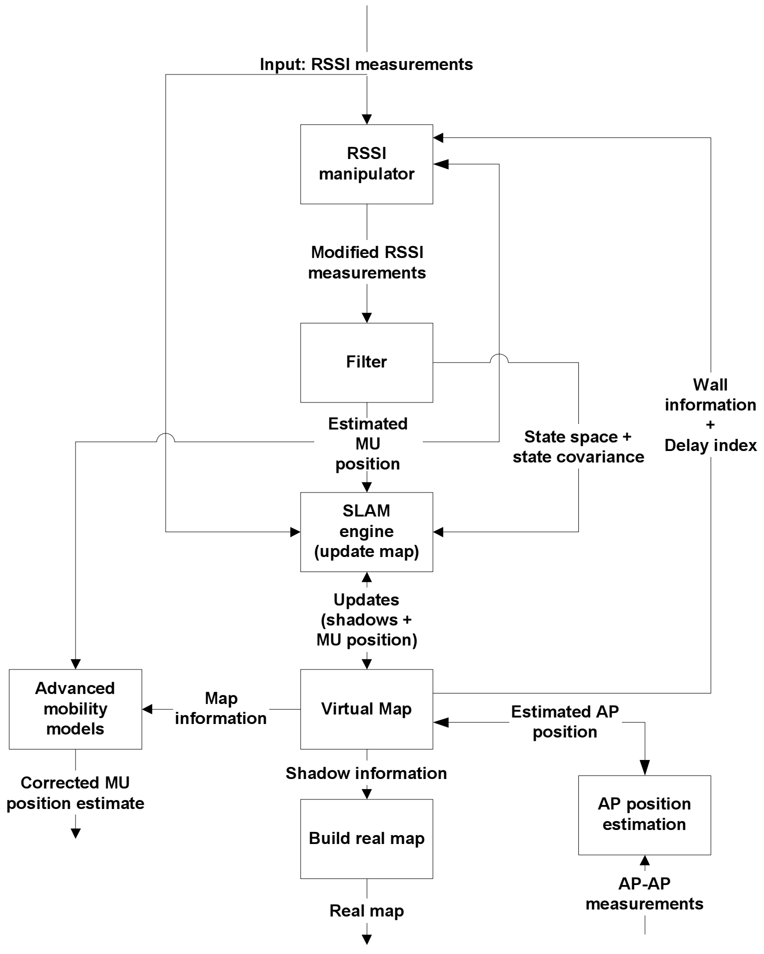


Figure 10.9: This figure illustrates the component and their interactions in the second iteration of SLAMWiN.

The re-structuring has mainly taken place in the central part of the design as the components *RSSI manipulation*, *Filter*, *Advanced mobility models*, *Build real map* and *AP location estimation* already lived up to the step definition. The *SLAM engine* and the *Virtual Map* have been split up so that both *SLAMWiN Core* and *SLAMWiN Shadow merging* contain parts from both of these former components.

Besides the restructuring of the components the third iteration used polar coordinates instead of cartesian coordinates. This change was motivated by the more intuitive description of locations in the map as these easier could be described relative to an AP. Furthermore, the use of polar coordinates reduced computational complexity in central parts of SLAMWiN as well as it simplified the storage of information regarding flags and shadows.

The following section will introduce the system design of the final iteration of SLAMWiN in more detail describing both the components and their interactions.

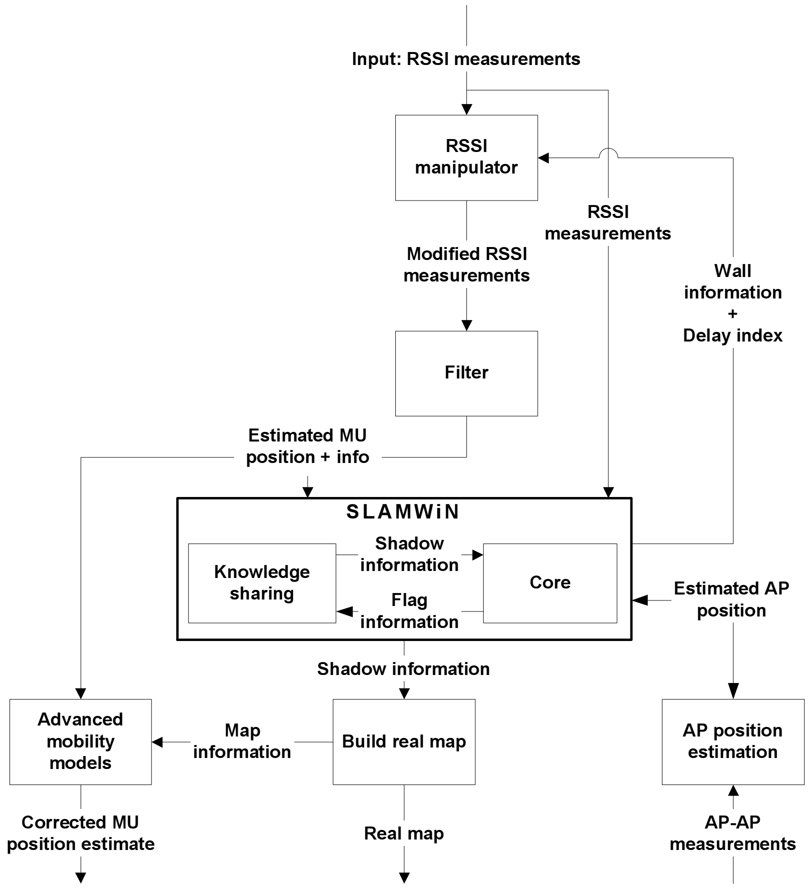


Figure 10.10: This figure illustrates the component and their interactions in the third iteration of SLAMWiN.

10.3 SLAMWiN architecture

This section will describe the SLAMWiN architecture design. It will shortly introduce the functionalities that are necessary and then focus on how these may be taken care of by components in the SLAMWiN system. It will furthermore introduce the system inputs and outputs.

The functionalities that are necessary in order to perform SLAMWiN have been identified using the localization stack introduced in Chapter 3 on page 16 as inspiration. Each of the lower layers of this stack have been analysed using the information introduced in the background section which has resulted in the following functionalities:

- Wireless radio transmitters forming the sensor layer of the localization stack.
- A log file or data structure containing measurements (measurement layer).
- Mobility models that describe the MU behaviour (fusion layer).
- A filter that is able to convert measurements into location estimates (fusion layer).

- A modified version of SLAM (arrangement layer).
- A environment emulator that supports easy and fast testing and evaluation of iterations during the development.

In the following the design of a solution to SLAMWiN will be described on a conceptual level and in the order is was invented. The environment emulator will be described in Appendix D on page 227.

SLAMWiN environment interaction

The first consideration regarding the design of the SLAMWiN system architecture treated the system input and output.

Naturally, the most important input will be RSSI measurements from each AP that is able to make an observation of the MU. These observations have been described in Chapter 3 on page 16 and may originate from two sources: Either a test bed is used that produces real life measurements, or an environment emulator is used that approximates the wireless channel of a given environment. The SLAMWiN system should apply an interface that may use measurements from both sources.

The system is thus designed to take a log file as input that may be generated by both measurement sources. The log file will have the structure shown in Figure 10.11 that has been adopted from the BLIP System Location Engine.

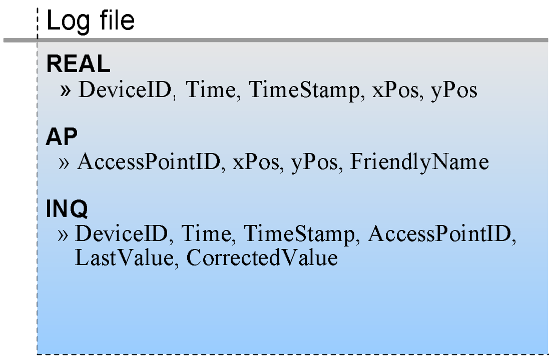

Log file
REAL
» DeviceID, Time, TimeStamp, xPos, yPos
AP
» AccessPointID, xPos, yPos, FriendlyName
INQ
» DeviceID, Time, TimeStamp, AccessPointID, LastValue, CorrectedValue

Figure 10.11: This figure shows the log file structure used by the SLAMWiN implementation.

The log file specifies the real position of a MU at a given time ($REAL$), the real position of APs in the environment (AP) and the RSSI measurements of a specific MU at a given time (INQ). The RSSI measurement is stated both as the current measured value ($LastValue$) and as the average value for the last 10 measurements ($CorrectedValue$). The $CorrectedValue$ will not be used in the SLAMWiN implementation as it only needs the current, unbiased RSSI measurement.

The SLAMWiN implementation will be conducted in Matlab and the log file will have to be parsed into a more expedient structure. This is done by a parser that has been adopted from previous research. The parser provides the struct shown in Figure 10.12 that is used in the Matlab implementation of SLAMWiN.

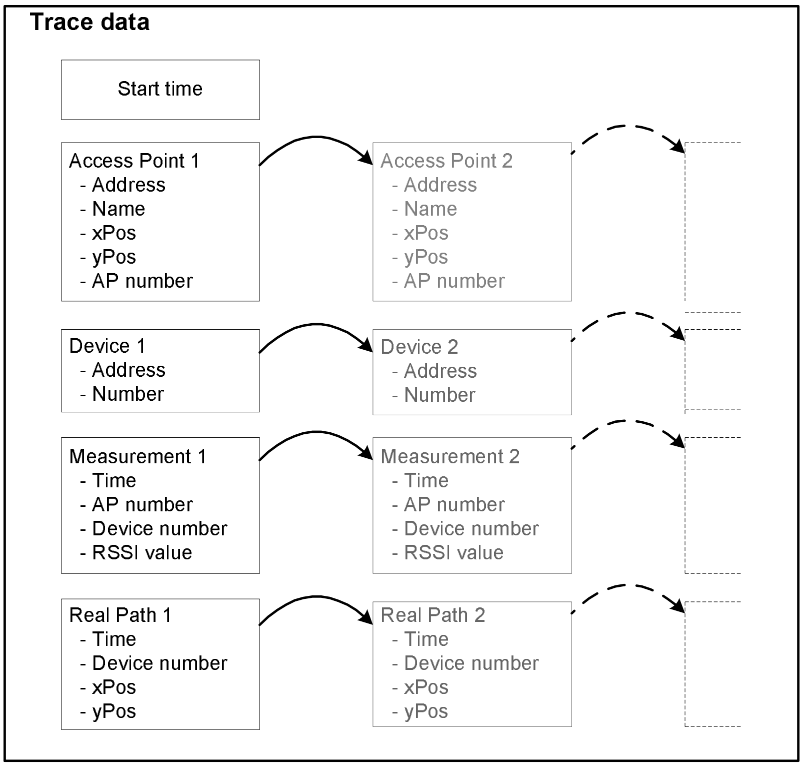


Figure 10.12: This figure shows the structure for all information used in the SLAMWiN implementation.

It has already been stated that a filter will be part of the SLAMWiN implementation. This filter will need a set of parameters that specify its initial state. These parameters can be divided into two groups: The first group describes a guess of the initial position ($\mathbf{x}_0$) of the MU and the certainty of this guess ($\mathbf{P}_0$), whereas the second group makes out a set of environment dependent constants describing some basic properties of the environment (α and β). Both groups of parameters will be described further in the Filter chapter.

The output of the system will be the estimated position of the MU defined by a set of coordinates and the time the MU was at the given position. Furthermore, the obtained information about the environment should make it possible to generate a map with likely walls in the environment.

The following list sums up the system inputs and outputs:

System input :

- Raw RSSI measurements from either environment emulator or test bed.
- $\mathbf{x}_0$ and $\mathbf{P}_0$ which are the initial co-ordinate and estimation error.
- α and β which are the initial values for the joint estimation part of the filter.

System output: :

- Estimated MU position (coordinates + time).
- Map of the environment (walls only).

The following subsections will describe the SLAMWiN components that have been introduced above. Each subsection will shortly describe the main purpose of the component and then present its inputs and outputs.

RSSI manipulator

The first component of the SLAMWiN system is the RSSI manipulator component. Its purpose is to manipulate RSSI measurements based on information provided by the SLAMWiN Core. Manipulation of RSSI measurements is performed by subtracting a value from specific measurements within a specific time interval. The subtracted value corresponds to the estimated damping of the measurement by possible walls.

Component inputs

It takes the RSSI measurement log file provided by either the environment emulator, or the BLIP Systems Location Engine. Furthermore, the component relies on information from the SLAMWiN Core about:

- ID of the AP from which the measurements should be modified.
- Estimated wall absorption value which should be subtracted from the real RSSI measurement.
- The time interval in which the manipulations of RSSI measurements should be applied.

Component outputs

The component provides a Matlab struct as the one described above in the *SLAMWiN environment interaction* section.

Filter

As described in Chapter 3 on page 16 regarding localization methods, the method of choice for performing localization in this project is triangulation based on RSSI measurements from different APs. Such measurements have to be interpreted into location estimates considering that they are affected by noise. The first consideration on the path towards a solution to SLAMWiN thus concerns filtering of the available measurements. Both filtering of noisy measurements and the interpretation of measurements as location estimates may be performed in one step. SLAMWiN thus applies a filter that performs this task.

Component inputs

As illustrated in Figure 10.10 the filter takes modified RSSI measurements as input. These are passed from the *RSSI manipulator* as a Matlab struct as the one described above in the *SLAMWiN/environment interaction* section.

Component outputs

The *Filter* component provides a matrix that contains the state space plus additional information for each tracked MU. The information provided for each measurement is:

- Position on $x - axis$.
- Position on $y - axis$.
- α
- β
- Time stamp for the location.
- Co-variance of the location error.

SLAMWiN Core

The SLAMWiN Core component will make out one half of the central part of the SLAMWiN system. It processes location estimates provided by the filter and estimates the effect of possible walls in the environment on a MU. This is done by monitoring both RSSI values and the estimated position. Situations where MUs pass from a LoS to a non-LoS situation, or vice versa, can through this be observed and this information is treated in order to correct the estimate position of a MU. The SLAMWiN Core informs the RSSI manipulator about previous and future measurements that are influenced by a wall and thus will have to be manipulated.

Furthermore the SLAMWiN Core handles the merging of estimated AP locations with the received location estimates from the filter.

Component inputs

The SLAMWiN Core component requires two kind of information which are provided by the filter:

- Current location estimate.
- The un-manipulated RSSI measurements.

Beyond that, the SLAMWiN Core will get information from a second SLAMWiN component that iteratively builds a virtual map of the environment that can be used for better estimates of walls. This second SLAMWiN component will provide the SLAMWiN Core with 'shadows' that define areas that have no LoS to specific APs.

Furthermore it will require information about the position of APs in the environment. These can either be provided as fixed positions that are known, or as position estimates that may change over time.

Component outputs

The output of the SLAMWiN Core component are information about the estimated MU paths through the environment. This information is specified as:

- Information about how much walls influences the measurements from a specific MU.
- Information about which measurements from a specific AP have been influenced by a given wall. This is done through a delay index that marks the first measurement that was influenced by the wall.

Both these information are stored in flags.

SLAMWiN Knowledge sharing

This component uses flag information gained from all previous MUs in the SLAMWiN Core in order to build a virtual map of the environment. This way it iteratively learns more accurately to predict how future MUs will be affected by walls in the environment. This information may then be used in the SLAMWiN Core to compensate for erroneous measurements or in order to enhance the accuracy of the wall position estimation process.

Since the SLAMWiN Knowledge sharing component is very closely linked to the SLAMWiN Core, it may be seen more as an extension of the functionality provided by the SLAMWiN Core, than as a separate component. The reason for describing it as a independent component anyway is that although it relies on the Core it offers new functionality to the SLAMWiN system that cannot be provided by the SLAMWiN Core.

Component inputs

The SLAMWiN Knowledge sharing component accesses all available information regarding flags in the SLAMWiN Core. It furthermore needs information regarding APs in the scenario and MU locations.

Component outputs

The output of this component is information about walls in the form of 'shadows' used by the SLAMWiN Core.

AP position estimation

This component is used in scenarios where the position of APs in the environment is unknown. It estimates the position of the APs based on initial AP to AP measurements before the SLAMWiN system is used to estimate MU positions. It may also be used later in order to re-estimate the AP positions.

Component inputs

Used for the initial AP position estimation, this component takes a set of AP to AP measurements as input as well as an indicator of the number of walls between each pair of APs. If it is used later for re-estimation of AP positions, it also takes a set of AP to AP measurements and estimations of wall absorptions as input, but this time these inputs will be based on more measurements and will thus provide a better estimate of the environment.

Component outputs

This component provides an estimated position of all APs in the environment.

Build real map

Much information about the environment is gained throughout the process of making better MU position estimates. This information can be used to build a map of the environment as a by-product to the localization. This map may either be used as a visualization of the estimated environment or in order to provide estimated wall locations to the advanced mobility model component. These tasks are handled by this component.

Component inputs

Input to this component is all information regarding likely location of walls acquired in the SLAMWiN Core.

Component outputs

The output should be a map of the environment that details the position of walls in it. Furthermore, it should pass information regarding the estimated map of the environment to the advanced mobility models component.

Advanced mobility models

The idea of this component is to correct the estimated MU positions from the filter based on mobility models that apply for the scenario. This is hoped to enhance the accuracy of the position estimate.

Component inputs

The input of this component should mainly be the estimated position provided by the filter. In addition to this also information regarding the estimated map of the environment could be used as an input, since it may be necessary to have this information in order to derive valid mobility models.

Component outputs

The output of this component should be a corrected MU position estimate that has a higher accuracy than the estimate provided by the filter.

Chapter 11

Filter

This chapter will describe the analysis of the filter used in SLAMWiN. First it will describe the purpose of the filter in this project and introduce two different types of filters. It will then analyze each filter and, based on a set of evaluation scenarios, perform an evaluation on them with respect to needed characteristics.

The purpose of this chapter is to account for the analysis that has lead to the choice of the filter used in the SLAMWiN implementation.

11.1 Introduction

In this project the purpose of the filter is to reduce the noise on the RSSI measurements obtained from either the environment emulator or the testbed. Several types of filters can be used to achieve this goal. All filters that will be investigated in this chapter are Kalman filters. The reason for this is that they represent a computational efficient approach to filtering[WB01] and because the SLAM approach also has been chosen to be Kalman based (see Section 7.1).

Particle filters will not be considered in this project because of two reasons. One, they do not fit as well as Kalman filters with the SLAM approach described in the background part and two, they are delimited in order to keep focus on SLAM and not on an evaluation of different filter types.

The filters that will be analysed are a Extended Kalman Filter (EKF) from [Fig04] and a Switching Extended Kalman Filter (SEKF). Both filters will be described in detail later in this chapter. The original Kalman filter is not analysed since it cannot handle non-linear systems which will be a required feature in this project. However, as an introduction to Kalman filters in general and to the notation used for describing them, a description of the original Kalman filter is included in Appendix C on page 224. The implementation of the respective filters will be based on the Recursive Bayesian Estimation Library (ReBEL) Toolkit[vdMW] which provides implementations of multiple types of filters and a general state space framework for these and will not be discussed in this chapter.

The following section will first describe the process model and the observation model used in this project. The process and observation models will be identical for both analysed filters and the description of the models will furthermore provide insight into the MU movement behaviour. Then the EKF and the SEKF will be introduced and after that a smoother extension to the Kalman filter and another extension for joint estimation of statespace parameters will be presented.

This chapter will close with a set of simple evaluation scenarios that are introduced with the purpose of providing measurements for the evaluation of the filters. The last section will compare the performance of the two filters and choose the preferred filter for this project.

11.2 Filter characteristics

This section describes the process and observation models. The process model describes the movement of the MU by defining a state transition matrix and the observation model describes the relation between the MU and e.g. a AP which makes an observation of it by measuring the RSSI value. Both models are essential for the filter and must be carefully derived. Each model is described in a separate subsection.

Observation model

In general the observation model describes the relation between an observer and a observed object. The observation will, as described in the background part of this report, be based on the RSSI value of the MU received at different APs. The APs are thus the observer while the MU is the observed object. This will be identical for both the EKF and the SEKF.

The observation model will describe the path loss of the signal from the MU to a given AP in dBm. The path loss may be calculated by knowing the transmission power at the MU and the RSSI value at the AP and can then be converted into a distance in m. This is done with the following equation:

$$\text{Path Loss (in dBm)} = \alpha + \beta \cdot [\log(D \text{ in meters})] \tag{11.1}$$

Here α is a measurement offset that is introduced to compensate for hardware and environment related influence on the measurement and β is a constant that depends on the propagation probabilities of the environment. D represents the distance between the MU and the AP which makes the observation and may be calculated using triangulation.

The position of the MU is then represented by an estimated distance to the given AP. Obviously, it is not enough to use a single observation when using this observation model, since the distance to a point in a 2-dimensional plane represents a circle. Thus, at least three observations from different APs are necessary in order to provide a useful location estimate. And even in this situation the location estimate may not be unambiguous, since the estimated circles around the three APs may not intersect in the same point.

The estimation of the MU location is thus an interpretation of the distance to different APs that considers that each distance is only an estimate and will be inaccurate to some degree.

α and β can be estimated dynamically for each measurement in order to optimize the performance of the filter. This is done through joint estimation in the Kalman filter which is described after the introduction of the Kalman filters (see Subsection *Joint estimation* in Section 11.3 on page 88).

Process model

In its essence filtering is the removal of noise from a 'tainted' signal based on information of the process that generates the signal and information about the noise. A process model is the description of the true process and therefore it has to reflect the actual behaviour as good as possible in order for the filter to work as desired. Thus, the process model applied in the filters used in this project should describe the actual movement of the MUs it tries to track. Since each MU represent a human being with a portable wireless device, the derived process model is based on observations of general human movement behaviour where a short extract of it is listed here:

Movement with preferred speeds: Humans will mostly walk at a speed of approximately 5 km/h.

No preferred movement direction: Humans do not in general have a preferred direction they move in. Neither are they restricted in which direction they may chose to go in. Of cause this changes when obstacles are introduced into the environment, but this is not considered in the process model.

A description of the process model noise used in this project is included after the filters and their state spaces have been introduced (see Subsection *Process model noise* in Section 11.3 on page 88).

11.3 Evaluated filters

As previously described two different Kalman filters will be evaluated. These filters have a fundamental difference with respect to the process model. While the EKF applies a general process model that describes the MU movement at all times, the SEKF makes use of several process models. In the following both filters are first introduced and then there process model is described.

Extended Kalman Filter

The Kalman filter described in Appendix C on page 224 addresses the general problem of estimating states of a discrete-time process that is governed by a *linear* stochastic difference equation. However, many applications have a *non-linear* relationship between the process to be estimated and the measurements. If a Kalman filter is being used in such applications it has to perform a linearization about the current mean and covariance and is then referred to as an Extended Kalman Filter (EKF)[Sal01].

In order to compute a state estimate in spite of the non-linearity, an EKF, in something similar to a Taylor series, linearizes the estimation around the current estimate by using the partial derivatives of the process and the measurements functions. The *process model* is now governed by the non-linear stochastic difference equation

$$\mathbf{x}(k) = f(\mathbf{x}(k-1), \mathbf{u}(k), \mathbf{w}(k-1)) \tag{11.2}$$

and the *observation model* by

$$\mathbf{z}(k) = h(\mathbf{x}(k), \mathbf{v}(k)) \tag{11.3}$$

where the random variables $\mathbf{w}(k)$ and $\mathbf{v}(k)$ again represent the process and observation noise respectively as in the Kalman filter case. The non-linear function f relates the previous state to the current state and includes a deriving function $\mathbf{u}(k)$ and h is the non-linear function that relates the state $\mathbf{x}(k)$ to the measurement $\mathbf{z}(k)$.

In practice the random variables $\mathbf{w}(k)$ and $\mathbf{v}(k)$ for each time step are not known. The *process model* and *observation model* are therefore approximated as

$$\tilde{\mathbf{x}}(k) = f(\hat{\mathbf{x}}(k-1), \mathbf{u}(k), 0) \tag{11.4}$$

$$\tilde{\mathbf{z}}(k) = h(\tilde{\mathbf{x}}(k), 0) \tag{11.5}$$

where $\hat{\mathbf{x}}(k)$ is some *a posteriori* estimate of the state.

It should at this time be noted that a fundamental flaw of the EKF is that the distributions after undergoing their respective nonlinear transformations are no longer normal. The EKF state estimator only performs an approximation of the optimality of Bayes' rule by linearization.

If a similar procedure to the one used in the previous section with the simple Kalman filter is applied to the EKF the following *time update* (11.6 and 11.7) and *measurement update* (11.8 to 11.10) equations are found.

$$\hat{\mathbf{x}}(k|k-1) = f(\hat{\mathbf{x}}(k-1), \mathbf{u}(k), 0) \tag{11.6}$$

$$\mathbf{P}(k|k-1) = \mathbf{A}(k)\mathbf{P}(k-1|k-1)\mathbf{A}^T(k) + \mathbf{W}(k)\mathbf{Q}(k-1)\mathbf{W}^T(k) \tag{11.7}$$

$$\mathbf{k}(k) = \mathbf{P}(k|k-1)\mathbf{H}^T(k)(\mathbf{H}(k)\mathbf{P}(k|k-1)\mathbf{H}^T(k) + \mathbf{V}(k)\mathbf{R}(k)\mathbf{V}^T(k))^{-1} \tag{11.8}$$

$$\hat{\mathbf{x}}(k|k) = \hat{\mathbf{x}}(k|k-1) + k(k)(\mathbf{z}(k) - h(\hat{\mathbf{x}}(k|k-1), 0)) \tag{11.9}$$

$$\mathbf{P}(k|k) = (\mathbf{I} - k(k)\mathbf{H}(k))\mathbf{P}(k|k-1) \tag{11.10}$$

In the EKF a single process model is used which describes the relation between the previous state and the current state as shown in Equation 11.4.

$$\mathbf{state}(k) = \begin{bmatrix} x(k) \\ y(k) \\ v_x(k) \\ v_y(k) \end{bmatrix} \tag{11.11}$$

In this project this relation is described using the list of parameters shown Equation 11.11. The state k is thus defined by the position on the x and y axis at k $(x(k), y(k))$ and the speed parallel to the x and y axis at k $(v_x(k), v_y(k))$. Jointly these parameters are known as the state space of the filter.

The approximated linear process model that may be used for the EKF is defined by the following update step

$$\mathbf{state}(k) = \mathbf{A} \cdot \mathbf{state}(k-1) \tag{11.12}$$

where $\mathbf{A}$ is the state transition matrix

$$\begin{bmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{11.13}$$

Switching Extended Kalman Filter

As previously mentioned the difference between the EKF and the SEKF is that the SEKF may choose between several process models instead of being restricted to using the same model at all times. This property is the consequence of the desire to be able to handle the following conflict that arises when a more complex process shall be filtered.

- The more precise a process model describes the actual process, in this case the MU movement, the better the filtering is going to be.
- The properties of a filtered signal may be complex in the way that the signal may be described by one function in one time interval and by other functions at other time intervals. This results in that the process model either has to be formulated very broad to be valid for all time intervals, which conflicts with the desire to have a precise model, or simply will not describe the actual process satisfactory at all times.

This conflict can be illustrated in the following example which could have been taken from on of the evaluation scenarios later in this chapter. In Figure 11.1 the real movement path of a MU is shown. As it can be seen in the figure the MU in the first part moves in a

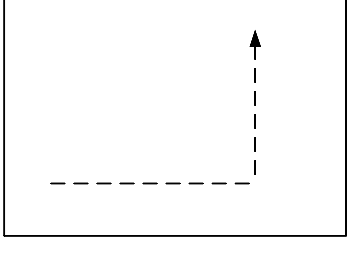


Figure 11.1: This figure shows an example of a MU movement path.

straight line towards right and then starts moving up in the second part of the path. A filter that should treat noisy observations of the MU movement may have the following process model which is equal to the previously described model for the EKF:

$$\mathbf{state}(k) = \mathbf{A} \cdot \mathbf{state}(k-1) \tag{11.14}$$

where $\mathbf{state}(k)$ has the state space $\begin{bmatrix} x(x) & y(k) & v_x(k) & v_y(k) \end{bmatrix}^T$ and where $\mathbf{A}$ is the state transition matrix

$$\begin{bmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{11.15}$$

This process model describes a MU that may move in all directions depending on the distributions from which $v_x(k)$ and $v_y(k)$ are drawn and it can therefore be used for both parts of the MU path. However, if the two parts of the track are investigated separately, a more precise process model could be applied. The general process model is sub-optimal and the filter will provide sub-optimal filtering.

The alternative is thus to apply a process model with the state transition matrix shown in Equation 11.16 to the first part and another state transition matrix shown in Equation 11.17 to the last part were the MU moves up.

$$\begin{bmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{11.16}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{11.17}$$

These last two state transition matrix's in Equation 11.16 and 11.17 assume movement in the direction the MU is actually moving towards and thus they are a more precise representation of the MU movement and will, according to the assumption that a more precise process model will provide better filtering, be able to perform a better tracking of the MU.

The SEKF uses a similar approach applying five different process models but goes one step further. As described in the subsection regarding the observation model, the α and β parameters may also be included in the state space. The SEKF does this and is furthermore designed without the speed parameter. Instead of having constant values for α and β, the SEKF is designed to use a constant speed. This is possible in the SEKF as opposed to the EKF, because the movement direction in the SEKF doe not depend on the speed. In the EKF the movement direction depends on the parameter that defines the speed along the $x-$ and $y-$axis.

The reason for this change in the state space is that the model assumes a MU movement pattern with an approximated constant speed for all MUs and therefore does not need to estimate it in the state space. Instead α and β are included in the state space instead. The reason for this is that these parameters may change during tracking and a dynamic estimate of these parameters could provide a better localization accuracy than a fixed value. This is discussed in the *Joint Extended Kalman Filter* section later in this chapter. The dimension of the state space will thus not change and is illustrated in Equation 11.18.

$$\mathbf{state}(k) = \begin{bmatrix} x(k) \\ y(k) \\ \alpha \\ \beta \end{bmatrix} \tag{11.18}$$

The five different models describes movement along the x-axis, the y-axis and no movement and the movements are called *North*, *South*, *East*, *West* and *Stationary* respectively. All five filters apply the same update step shown in Equation 11.19 and the same transition matrix $\mathbf{A}$ shown in Equation 11.20.

$$\mathbf{state}(k) = \mathbf{A} \cdot \mathbf{state}(k-1) + \begin{bmatrix} T \cdot cos(angle) \cdot speed \cdot move \\ T \cdot sin(angle) \cdot speed \cdot move \\ 0 \\ 0 \end{bmatrix} \tag{11.19}$$

where *angle* is the direction of movement chosen among the values $[0, \frac{\pi}{2}, \pi, \frac{3 \cdot \pi}{2}]$. *Speed* is the movement speed with may be set to an arbitrary constant and *move* is set to 1 for the *North*, *South*, *East* and *West* models describing movement and to 0 for the *Stationary* model. Thus, *move* defines if a MU is moving or not.

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{11.20}$$

Solving one problem, the SEKF introduces a new problem. This problem consists of when to choose which model. Different approaches exists which try to solve this. Some of these apply a Hidden Markov Model (HMM) as a 'switch' that chooses the right process model[TL00] [BT02]. This approach was implemented in early versions of the SLAMWiN implementation and some preliminary tests were performed. The HMM approach may very well have lead to a successful implementation of the SEKF, but it was replaced by another approach based on comparison of errors from the different process models. The chosen approach is described below and was chosen since it was based on a conceptually simpler approach and fast lead to good localization results.

The chosen approach lets all models run simultaneously in five separate filters. For each new measurement the error is calculated as the trace of the state covariance matrix. The switch then selects the filter with the smallest error, provided, that one of the filters has a smaller error than the others. In cases were two or more filters have the same error, the last previously selected filter will be used again.

Process model noise

On both filters noise is introduced on the process model. For the EKF the process noise covariance ($\mathbf{Q}$ matrix) will be derived according to [Sta01]. This has lead to the noise covariance matrix

$$\begin{bmatrix} s_x \cdot T^{2/3} & 0 & s_x \cdot T/2 & 0 \\ 0 & s_y \cdot T^{2/3} & 0 & s_y \cdot T/2 \\ s_x \cdot T/2 & 0 & s_x & 0 \\ 0 & s_y \cdot T/2 & 0 & s_y \end{bmatrix} \tag{11.21}$$

where T is the size of each step in time and s_x and s_y are the standard deviation of the noise parallel to the x and y axis respectively. The values for s_x and s_y are based on experiences from earlier measurements and are set to 0.001.

For the SEKF noise will only be introduced on the x and y position and there will not be any correlated noise between these parameters. This reflects the independence between these two parameters that stems from the fixed speed. The standard deviation of the process noise will not be derived for the SEKF, but set based on experiences from a set of evaluation scenarios. This has lead to the noise covariance matrix

$$\begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{11.22}$$

where s_x and s_y are set to 0.0095. These values are again found based on experiences from earlier measurements.

Joint Extended Kalman Filter

The Joint Extended Kalman Filter approach is commonly used when parameter estimates have to be derived sequentially as new observations are being collected[CFN]. It allows for joint estimation of both the state space and parameters included in the state space as these are combined into a joint state vector: $\mathbf{j}(k) = [\mathbf{x}(k)^T \mathbf{p}(k)^T]^T$.

The previously introduced parameters α and β from Equation 11.18 are such parameters that can be dynamically estimated for each measurement. The process and observation model introduced in Equation 11.2 and 11.3 will thus be modified accordingly[WvdMN00] in the following equations.

$$\mathbf{j}(k) = f(\mathbf{j}(k-1), \mathbf{u}, \mathbf{w}(k-1)) \tag{11.23}$$

$$\mathbf{z}(k) = h(\mathbf{j}(k), \mathbf{v}(k-1)) \tag{11.24}$$

where the parameters $\mathbf{p}$ (α and β) from a discrete-time nonlinear dynamic system are included in the joint state space $\mathbf{j}$. The estimation on $\mathbf{j}(k)$ can be done recursively by the following state space equation:

$$\mathbf{j}(k) = \begin{bmatrix} f(\mathbf{x}(k-1), \mathbf{p}(k-1)) \\ \mathbf{I} \cdot \mathbf{p}(k-1) \end{bmatrix} + \begin{bmatrix} \mathbf{B} \cdot \mathbf{v}(k) \\ \mathbf{u}(k) \end{bmatrix} \tag{11.25}$$

Kalman smoother

The Kalman smoother[BSLK01], which also is called the Rauch-Tung-Streibel Smoother[RTS65] uses a forward-backward algorithm to compute the estimate of a state at step k based on the measurement data $y_{1:T}$. This is done by two passes of a Kalman-like recursion where, as opposed to the Kalman filter that only uses prior and current measurements ($y_{1:k}$); also future measurements ($y_{k:T}$ for $T > k$) are used. This delimits the Kalman smoother from being used in an 'online' situation where the estimated state at step k is required instantaneously, but increases the estimation precision in situations where a complete set of measurement data $y_{1:T}$ is available for all steps k for $1 \leq k \leq T$.

In this project a *fixed-lag* Kalman smoother is used. A fixed-lag Kalman smoother estimates the state k based on the measurement data $y_{1:k+L}$ where L is a fixed lag. In this way the Kalman smoother may start estimating the state at step k when measurement number $k + L$ becomes available. This will typically be before the complete measurement data $y_{1:T}$ is available since the lag often is chosen so that $L << T$.

For the last L number of measurements in the measurement data $y_{1:T}$ the lag will be limited by the number of available measurements in $y_{k:T}$. The lag will thus decrease by one for each step until it becomes 0 for measurement y_T.

The calculation of the smoother gain, the smoothed state and the covariance of the smoothed state are performed iteratively and much like in the Kalman filter[BSLK01]. Equation 11.26 shows the smoother gain, while Equation 11.27 and 11.28 show the smoother state and state covariance respectively.

$$\mathbf{C}(k) = \mathbf{P}(k|k)\mathbf{A}(k)^T\mathbf{P}(k+1|k)^{-1} \tag{11.26}$$

where $\mathbf{A}$ is the state transition matrix.

$$\hat{\mathbf{x}}(k|\mathbf{N}) = \hat{\mathbf{x}}(k|k) + \mathbf{C}(k)[\hat{\mathbf{x}}(k+1|\mathbf{N}) - \hat{\mathbf{x}}(k+1|k)] \tag{11.27}$$

$$\mathbf{P}(k|\mathbf{N}) = \mathbf{P}(k|k) + \mathbf{C}(k)[\mathbf{P}(k+1|\mathbf{N}) - \mathbf{P}(k+1|k)]\mathbf{C}(k)^T, k = \mathbf{N}-1, \ldots, 0 \tag{11.28}$$

11.4 Filter evaluation

In order to find the filter that will provide the best localization the EKF and SEKF are evaluated using six evaluation scenarios. The following subsections will define these evaluation scenarios and two evaluation metrics. Thereafter the results from both filters are listed and discussed and one of the filters is chosen.

Evaluation metrics

Two different evaluation metrics will be used in the following evaluation scenarios. The metrics are designed to evaluate two different aspects of the filters. The first metric shall evaluate the filters capabilities in a tracking situation where the error from the real position to the estimated position is calculated for every measurement. This metric is called the

Track-metric. The second metric shall evaluate the filters ability to estimate the path of a MU in situations where the position of the MU on the path is not considered. This metric is called the *Path-metric*.The metrics should not be mistaken as a filtering and smoothing metric, since both filtered and smoothed position estimates can be used in both metrics. In the following both metrics are described in more detail.

The Track-metric: The Track-metric is calculated based on comparison of the estimated position (found by either filtering or smoothing) and the real position for every measurement. The real position of the MU is calculated by defining way points along the real path, finding the rest of the path via interpolarization. The distance from the real position to the estimated position is calculated by finding the length of the vector that spans from the real position to the estimated position for each measurement. This gives an evaluation metric that produces a single result in m representing the average localization error of all measurements weighted by the sample times.

The Path-metric: The Path-metric is calculated based on the shortest distance from the estimated path to the real path. This is done by using an algorithm that first divides the estimated path into segments that correspond to segments on the real path by estimating the corners on the path and then efficiently calculates the shortest distance between a point (the estimated position) and the corresponding line segment (the real path). This is illustrated in Figure 11.2.

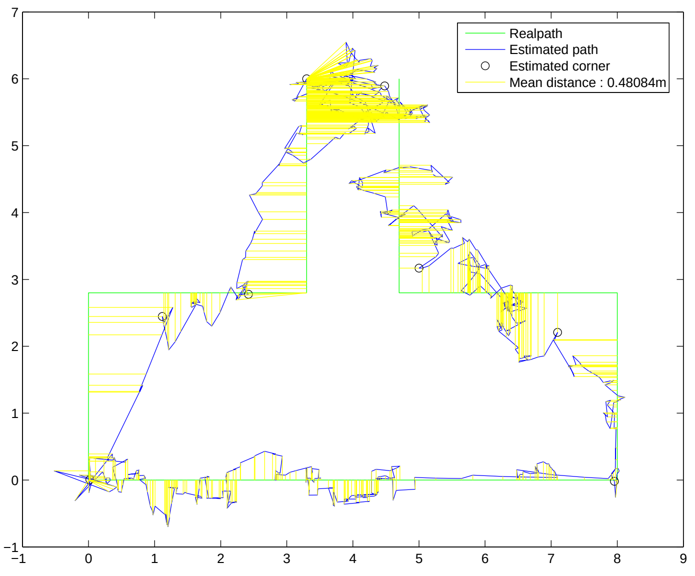


Figure 11.2: The figure shows an example of how the Path-metric is calculated.

Evaluation scenario

During the development of the SEKF, measurements from previous research[Fig04] were used in an iterative evaluation process of the filter. Each step of the development was evaluated before proceeding to the next step. The used measurements were obtained from the scenario shown in Figure 11.3. The environment in this scenario is a stable in Foulum and the figure details the MU path and the location of 15 APs.

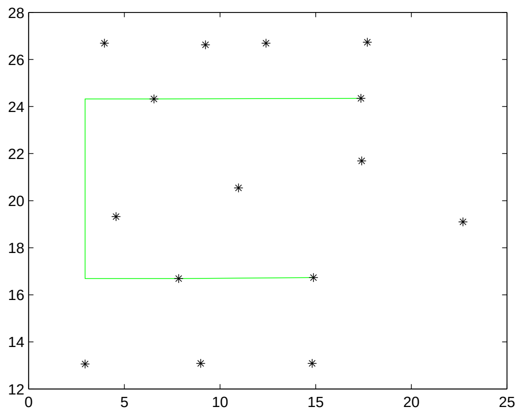
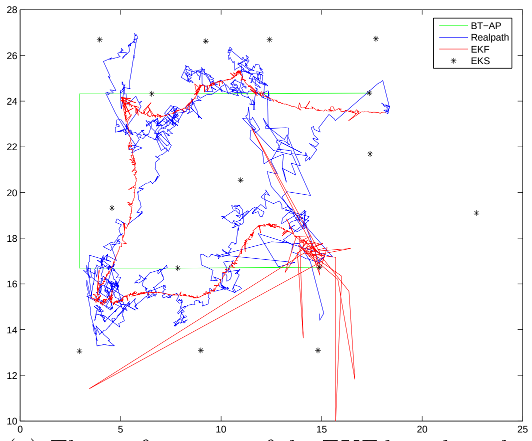


Figure 11.3: The figure shows the MU path (green line) and the AP locations ($*$) of the evaluation scenario used during development of the SEKF.

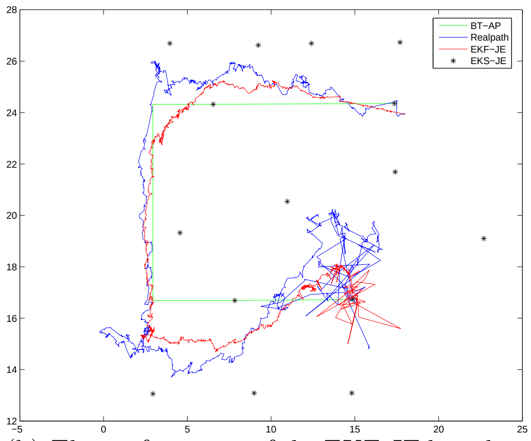
The performance of the EKF applied in [Fig04] is shown in (a) in Figure 11.4 and was used as goal for the performance of the developed SEKF. In order for the SEKF to succeed, it had to perform better than the EKF. This would either be due to a better performance, or due to equal performance in which case the simplest filter, the EKF, would be preferable. The performance of the SEKF is shown in (c) in Figure 11.4 and it appears to perform better than the EKF.

The filters have been evaluated by both metrics and the results are shown in Table 11.1. Another aspect that was considered during the development was the size of the state space. The Extended Kalman Filter with Joint Estimation (EKF-JE) shown in (b) in Figure 11.4 from [Fig04] was the filter that provided the best results, but it was also using a state space consisting of six parameters ($\begin{bmatrix} pos_x & pos_y & v_x & v_y & \alpha & \beta \end{bmatrix}^T$). In order to be observable, this filter requires six APs, as it is linear and the observability may be determined by looking at the rank of the state transition matrix. This has the rank of six. This is not a problem in the scenario illustrated in Figure 11.3 where enough APs are available, but it is desirable not to be restricted to scenarios with a large number of APs.

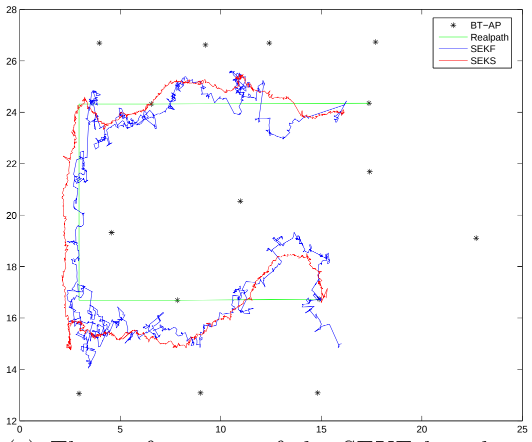
The developed SEKF thus uses a smaller state space consisting of only four parameters ($\begin{bmatrix} x & y & \alpha & \beta \end{bmatrix}^T$) in order to work in scenarios with less APs. It is, however, hard to determine precisely how many APs the SEKF requires as there is no rule for determining the observability of a general non-linear system.[SBM98]. Since both the state transition matrix and the observation model are non-linear in the SEKF, the observability conditions of the SEKF have thus only been verified through tests. Figure 11.5 shows that the filter



(a) The performance of the EKF based on the Foulum scenario.



(b) The performance of the EKF-JE based on the Foulum scenario.



(c) The performance of the SEKF based on the Foulum scenario.

Figure 11.4: This figure shows the performance of the EKF and SEKF respectively.

is able to perform localization in a scenario where the MU is covered with down to four APs.

In order to perform a more comprehensive evaluation of the filters, five different scenarios have been selected in order to provide measurements for the filter evaluation. The scenarios have been designed in order to provide measurements that cover certain interesting aspects of localization and according to available environments in which the measurements could be made. The selected evaluation scenarios are:

1. Square movement with favorable AP locations.

2. Square movement.

Filter :	EKF	EKF-JE	SEKF
Start position :	$(16, 15)$	$(16, 15)$	$(16, 15)$
Alpha is set to :	-52	-52	-52
Beta is set to :	2.5	2.5	2.5
Speed is set to :	dynamic	dynamic	0.07m/s
Track-Metric (no smoother):	3.6777 m	3.4166 m	2.9202 m
Track-Metric (smoother) :	3.2911 m	2.7843 m	2.9368 m
Path-Metric (no smoother):	1.5483 m	1.3972 m	0.8605 m
Path-Metric (smoother) :	1.1208 m	0.7685 m	0.8768 m

Table 11.1: This Table shows the details of the evaluation scenario and the evaluation metric for each of the filters.

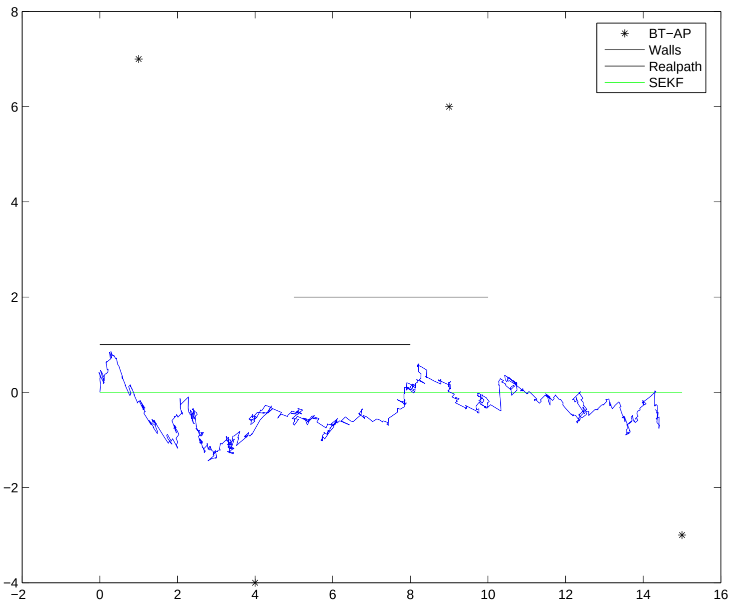


Figure 11.5: This figure shows a scenario where a MU is located by using only four APs.

3. Movement with several direction changes.
4. Triangular movement.
5. Movement with temporary stop.

In order to reflect the PnP capability of the implementation, the filters will not be tuned for each scenario, but will apply the same settings in all scenarios. This means that the same α, β, noise and speed parameters are used in all evaluation scenarios. However, one exception will be made. In order to provide more insight into the SEKFs dependence on a good approximation of the speed parameter, the SEKF additionally will be evaluated using two different speeds. The first speed will be the average speed calculated for each scenario and this speed will thus vary from scenario to scenario. The exact values of the parameter is shown in the *Evaluation results* subsection. The second speed will be fixed for all scenarios and will represent a guess of how fast the MU will most like move in the scenarios. This is equivalent to a real life situation where an initial guess about the typical MU speed has to be made before the localization system can be used.

It should be noted that the scenarios, due to limitations off the available equipment, do not use more than five APs. This is considerably less than in the scenario used in [Fig04] and unfortunately the number of APs is not large enough for the best performing EKF from that research. That EKF required at least 6 APs to be observable. Instead another of the EKFs from [Fig04] will be used as a reference in the following evaluation of the developed SEKF. The details of each evaluation scenario are described below. The following parameters shown in Table 11.2 are the same for all scenarios:

Filter :	SEKF (fixed speed)	SEKF (true speed)	EKF
Alpha is set to :	-67	-67	-55
Beta is set to :	2.1	2.1	2.1
Observation noise covariance:	0.0095	0.0095	0.001
Lag size for smoothing :	100 measurements	100 measurements	100 measurements

Table 11.2: This Table shows the parameters that all evaluation scenarios have in common.

Square movement with favorable AP locations: This evaluation scenario will take place within a single room with one MU. The details of the room are shown in Figure 11.6 which details the room size and all soft partitions within the room which in this case mainly are chairs and tables. Furthermore, the position of the used APs

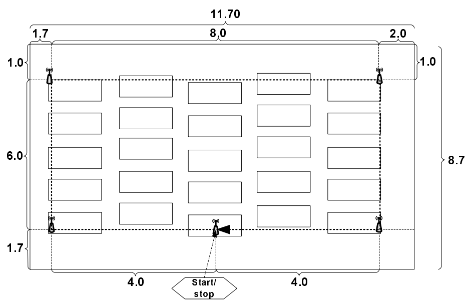


Figure 11.6: This figure shows the environmental details of the first evaluation scenario and the MU path. All distances are in meters.

and the path, which the MU travels while the evaluation data is collected, is shown. The path has been placed close to each AP in order to provide measurements with as high an accuracy as possible (see Figure 3.7 on page 22 in chapter 3).

Square movement: The second evaluation scenario takes place within the same room with the same setup of APs. The difference to the previous evaluation scenario is that the path which the MU travels has changed. In this scenario the path will not come close to all AP and the direction changes are thus expected to be harder to detect. The path is illustrated in Figure 11.7.

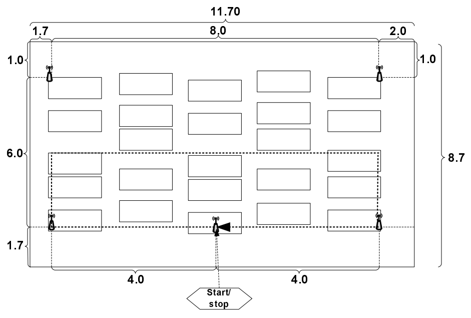


Figure 11.7: This figure shows the environmental details of the second evaluation scenario and the MU path. All distances are in meters.

Movement with several direction changes: The third evaluation scenario takes place within the same room. The details of the room are the same as in the previous evaluation scenario and are shown in Figure 11.8. The difference to the previous evaluation scenario is again that a different MU path is used. In this scenario the path will go by close to all AP, but will also have several more direction changes. This scenario shall evaluate if the filters are able to distinguish between direction changes that are closer to each other.

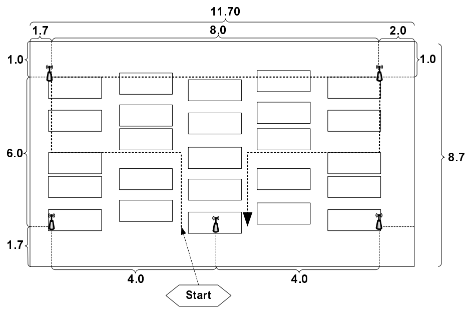


Figure 11.8: This figure shows the environmental details of the third evaluation scenario and the MU path. All distances are in meters.

Triangular movement: The fourth evaluation scenario takes again place in the same room. The new MU path is shown in Figure 11.9 as are the position of the used APs and the soft partitions. This time the MU will travel along a path which not directly follows the movement models *North*, *South*, *East*, *West* and *Stationary* in the SEKF. It is thus evaluated how the SEKF is able to estimate movement that does not fit any of its process models.

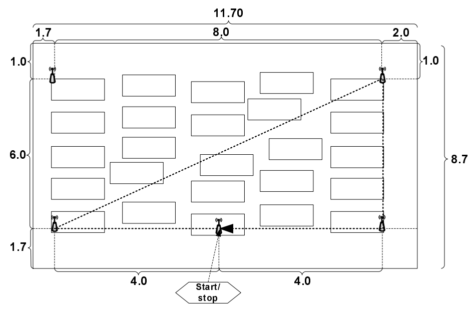


Figure 11.9: This figure shows the environmental details of the fourth evaluation scenario and the MU path. All distances are in meters.

Movement with temporary stop: The last evaluation scenario, is still taking place in the same room and uses a very simple movement with a temporary stop lasting for 60 seconds. The path and the location where the stop takes place is shown in Figure 11.10. The position of the used APs is the same. This scenario shall evaluate how the filters handle a non-moving MU.

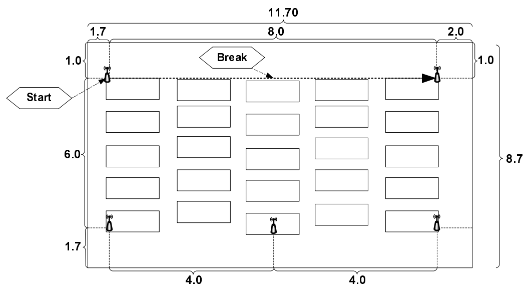


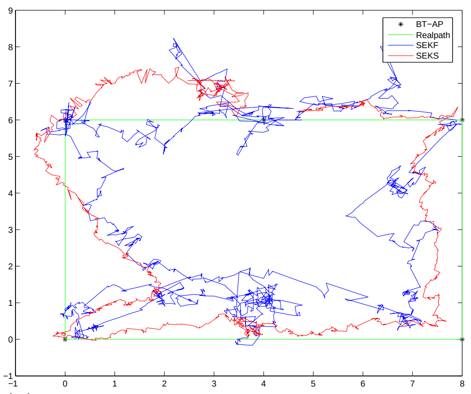
Figure 11.10: This figure shows the environmental details of the fifth evaluation scenario and the MU path. All distances are in meters.

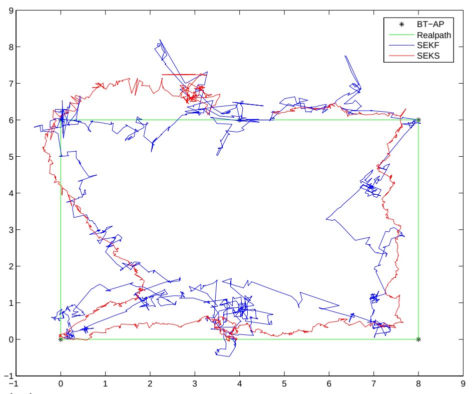
Evaluation results

This subsection lists the results from the filter evaluation scenarios. A figure will illustrate both the real path and two estimated paths. The estimated paths are with and without a smoother respectively. Furthermore, tables will be included which show the details of each evaluation scenario.

Square movement with favorable AP location: The first two entries in Table 11.3 show the values of the relevant filter parameters used in this evaluation scenario for each of the filters and for the SEKF with both fixed speed and the actual speed used in the scenario.

The estimated paths from the two filters are shown as graphs in Figure 11.11 and the results from the evaluation metrics are shown in Table 11.3:


(a) The performance of the SEKF with fixed speed based on the square scenario.


(b) The performance of the SEKF with true speed based on the square scenario.

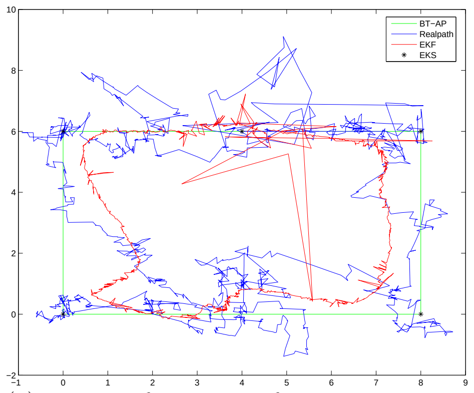

(c) The performance of the EKF based on the square scenario.

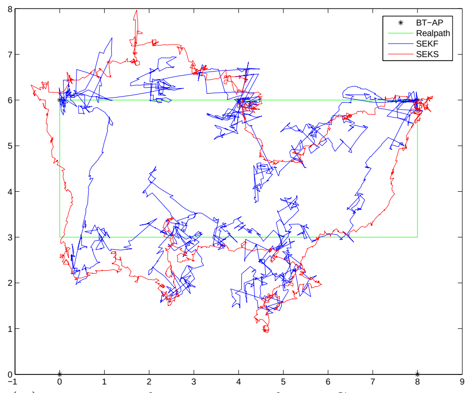
Figure 11.11: This figure shows the performance of the SEKF with fixed and true speed and the performance of the EKF. Both results with (red) and without (blue) smoothing are shown.

Filter :	SEKF (fixed speed)	SEKF (true speed)	EKF
Start position :	$(4, 6)$	$(4, 6)$	$(4, 6)$
Speed is set to :	0.08 m/s	0.095 m/s	dynamic
Track-Metric (no smoother) :	2.9602 m	2.8988 m	2.8536 m
Track-Metric (smoother) :	2.4163 m	2.3649 m	2.4591 m
Path-Metric (no smoother) :	0.8525 m	0.7397 m	0.7216 m
Path-Metric (smoother) :	0.5747 m	0.5382 m	0.5667 m

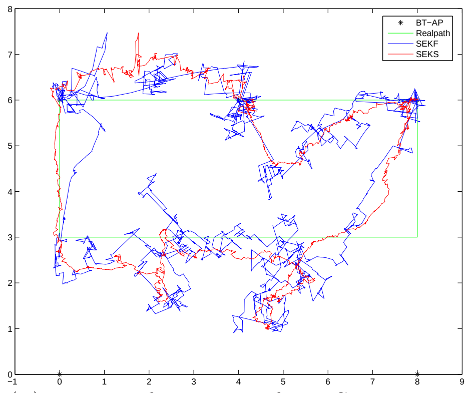
Table 11.3: This Table shows the parameters and results of the evaluation metrics for the first evaluation scenario.

Square movement: The first two entries in Table 11.4 show the values of the relevant filter parameters used in this evaluation scenario for each of the filters and for the SEKF with both fixed speed and the actual speed used in the scenario.

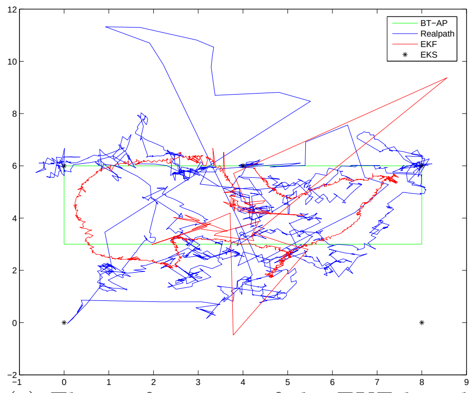
The estimated paths from the two filters are shown as graphs in Figure 11.12 and the results from the evaluation metrics are shown in Table 11.4:



(a) The performance of the SEKF with fixed speed based on the half square scenario.



(b) The performance of the SEKF with true speed based on the half square scenario.



(c) The performance of the EKF based on the half square scenario.

Figure 11.12: This figure shows the performance of the SEKF with fixed and true speed and the performance of the EKF. Both results with (red) and without (blue) smoothing are shown.

Filter :	SEKF (fixed speed)	SEKF (true speed)	EKF
Start position :	$(4, 6)$	$(4, 6)$	$(4, 6)$
Speed is set to :	0.08 m/s	0.032943 m/s	dynamic
Track-Metric (no smoother) :	2.9842 m	2.9636 m	3.4254 m
Track-Metric (smoother) :	2.8192 m	2.7493 m	3.1658 m
Path-Metric (no smoother) :	0.56763 m	0.59882 m	0.90018 m
Path-Metric (smoother) :	0.67674 m	0.62888 m	0.57013 m

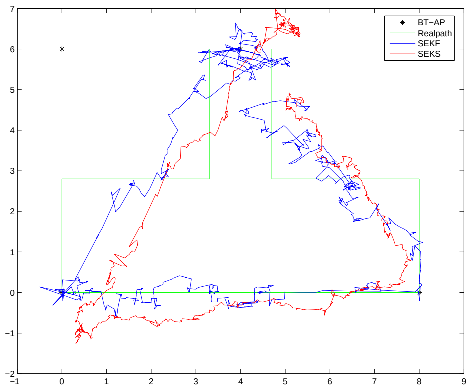
Table 11.4: This Table shows the parameters and results of the evaluation metrics for the second evaluation scenario.

Movement with several direction changes: The first two entries in Table 11.5 show the values of the relevant filter parameters used in this evaluation scenario for each of the filters and for the SEKF with both fixed speed and the actual speed used in the scenario.

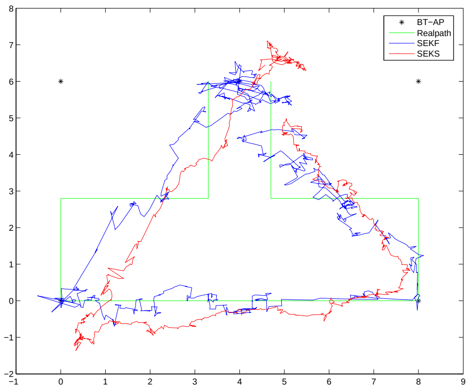
The estimated paths from the two filters are shown as graphs in Figure 11.13 and the results from the evaluation metrics are shown in Table 11.5:

Filter :	SEKF (fixed speed)	SEKF (true speed)	EKF
Start position :	$(3.5, 6)$	$(3.5, 6)$	$(3.5, 6)$
Speed is set to :	0.08 m/s	0.086939 m/s	dynamic
Track-Metric (no smoother) :	3.1819 m	3.1728 m	2.9199 m
Track-Metric (smoother) :	3.0119 m	3.0157 m	2.9189 m
Path-Metric (no smoother) :	0.4797 m	0.4808 m	0.8769 m
Path-Metric (smoother) :	0.7828 m	0.7720 m	0.6736 m

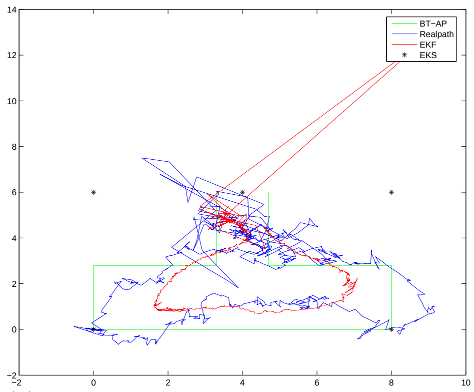
Table 11.5: This Table shows the parameters and results of the evaluation metrics for the third evaluation scenario.



(a) The performance of the SEKF with fixed speed based on the tetris scenario.



(b) The performance of the SEKF with true speed based on the tetris scenario.

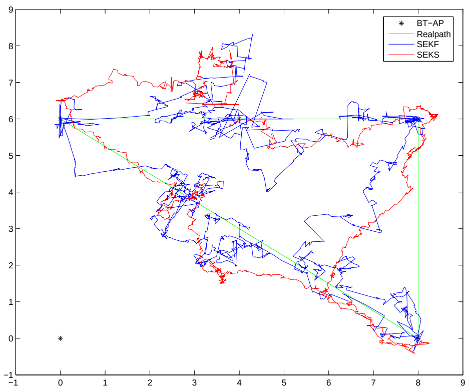


(c) The performance of the EKF based on the tetris scenario.

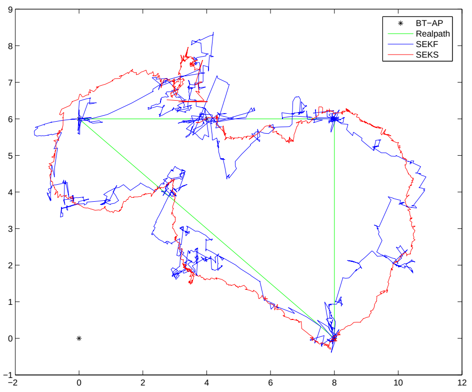
Figure 11.13: This figure shows the performance of the SEKF with fixed and true speed and the performance of the EKF. Both results with (red) and without (blue) smoothing are shown.

Triangular movement: The first two entries in Table 11.6 show the values of the relevant filter parameters used in this evaluation scenario for each of the filters and for the SEKF with both fixed speed and the actual speed used in the scenario.

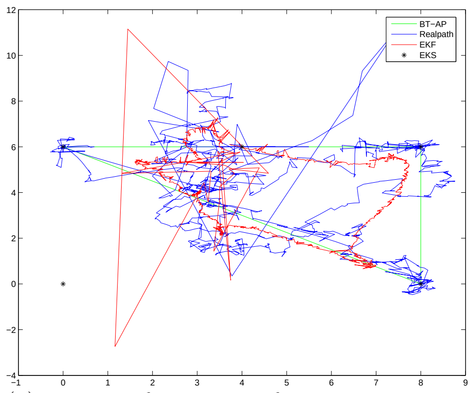
The estimated paths from the two filters are shown as graphs in Figure 11.14 and the results from the evaluation metrics are shown in Table 11.6:



(a) The performance of the SEKF with fixed speed based on the triangle scenario.



(b) The performance of the SEKF with true speed based on the triangle scenario.



(c) The performance of the EKF based on the triangle scenario.

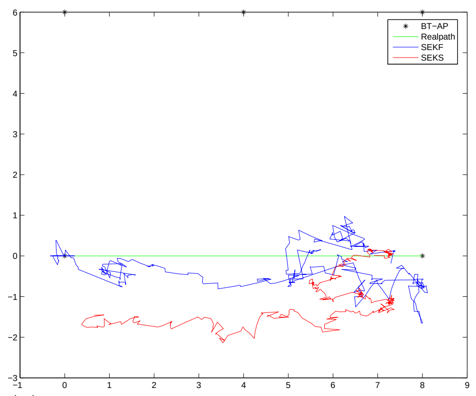
Figure 11.14: This figure shows the performance of the SEKF with fixed and true speed and the performance of the EKF. Both results with (red) and without (blue) smoothing are shown.

Filter :	SEKF (fixed speed)	SEKF (true speed)	EKF
Start position :	$(4, 6)$	$(4, 6)$	$(4, 6)$
Speed is set to :	0.08 m/s	0.042457 m/s	dynamic
Track-Metric (no smoother) :	2.9796 m	3.1464 m	3.0246 m
Track-Metric (smoother) :	2.7997 m	2.9624 m	2.835 m
Path-Metric (no smoother) :	0.5992 m	0.8197 m	0.7193 m
Path-Metric (smoother) :	0.7117 m	0.9919 m	0.5267 m

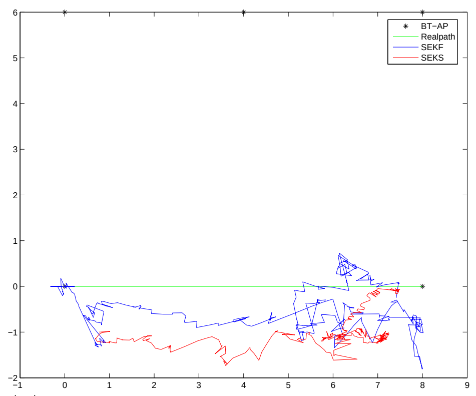
Table 11.6: This Table shows the parameters and results of the evaluation metrics for the fourth evaluation scenario.

Movement with temporary stop: The first two entries in Table 11.7 show the values of the relevant filter parameters used in this evaluation scenario for each of the filters and for the SEKF with both fixed speed and the actual speed used in the scenario.

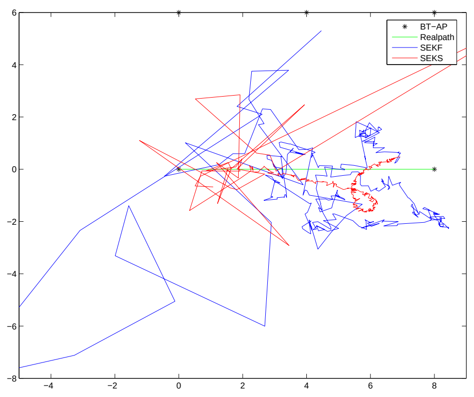
The estimated paths from the two filters are shown as graphs in Figure 11.15 and the results from the evaluation metrics are shown in Table 11.7:



(a) The performance of the SEKF with fixed speed based on the stop test scenario.



(b) The performance of the SEKF with true speed based on the stop test scenario.



(c) The performance of the EKF based on the stop test scenario.

Figure 11.15: This figure shows the performance of the SEKF with fixed and true speed and the performance of the EKF. Both results with (red) and without (blue) smoothing are shown.

Discussion and choice of filter

Overall the evaluation scenarios have shown that the SEKF with neither of the used settings performs significantly better than the EKF if the applied Track-metric is used for

Filter :	SEKF (fixed speed)	SEKF (true speed)	EKF
Start position :	$(4, 6)$	$(4, 6)$	$(4, 6)$
Speed is set to :	0.08 m/s	0.046003 m/s	dynamic
Track-Metric (no smoother) :	3.4433 m	3.5616 m	13.2189 m
Track-Metric (smoother) :	3.1464 m	3.1478 m	2.9395 m
Path-Metric (no smoother) :	0.4727 m	0.6123 m	1.413 m
Path-Metric (smoother) :	1.0916 m	0.9461 m	0.7634 m

Table 11.7: This Table shows the parameters and results of the evaluation metrics for the fifth evaluation scenario.

evaluation. The difference in performance of the two filters for the most part lies below 30 cm with changing filters having the lowest estimation error. The reason for the SEKF performing less good in the Track-Metric could be due to localization errors in space that are parallel to the real path. This property could be caused by the basic design of the SEKF. If the largest error indicates movement along the y-axis, but the MU in fact moves along x-axis, the SEKF will not allow much movement parallel to the x-axis and will therefore have problems keeping up with the MU. This would directly be reflected in the tracking evaluation metric, since the filter would estimate the MU location to be further back along the real path. Such an error would most likely also affect the following location estimates in a negative way and thus have a large impact on the localization accuracy. No choice of filter can be made based only on this result.

The Path-metric revealed that the SEKF performs significantly better than the EKF. Only in the first evaluation scenario does the EKF perform better, but only by 2 cm. The first evaluation scenario is also the only scenario where the SEKF performs better when using the actual speed.

The evaluation of the filters reveals that smoothing enhances the performance of the EKF significantly whereas it does not have the same effect on the SEKF. This is surprising and was thus investigated further and a flaw in the SEKF smoother implementation was found. As described earlier the SEKF chooses between different filters for the forward pass through the measurements. Naturally, the SEKF smoother should do the same on the backwards pass. However, the current implementation forces the SEKF smoother to choose the same filter for each measurement on the backwards pass as the SEKF used on the forward pass. This makes the SEKF smoother biased by the choices of the SEKF. Since it is not trivial to correct this fault in the implementation and since the SEKF already provides satisfactory results the fault will not be corrected.

The choice of which filter performs best with respect to the Path-metric stands thus between the SEKF and the EKF with smoothing. If these two are compared for each evaluation scenario the SEKF performs significantly better in two scenarios and marginally better in two others. Only in the fourth scenario with triangular movement does the EKF with smoother perform best. Considering that the EKF performance is enhanced with smoothing while the SEKF is not, the evaluation based on the Path-metric thus points at the SEKF as the best choice of filter in this project.

Another general tendency is that the SEKF does not perform significantly better if the real speed is used as compared to the fixed speed of 0.08 m. This may be due to relative small differences between the real speed and the fixed speed, but also in evaluation scenarios where the fixed speed is twice as high as the real speed does the localization error not deviate considerably. This is considered a good property of the filter, since it supports the PnP capability of the filter.

A specific remark should be made regarding the last evaluation scenario where the MU temporarily stops. Neither of the evaluated filters are able to handle this situation as desired. Although the graphs indicate that something takes place in the scenario, further enhancement of the filter implementation would be needed in order to enable the filter to handle this situation as desired. However, this enhancement has been delimited due to the project priority being the development of SLAM for wireless networks and not filter design.

Finally a note should be made regarding the basic design of the SEKF. Because the SEKF applies multiple filters it will be computationally heavier than EKF. This is not considered as a problem, since the localization engine is thought to run on centralized localization server where enough computational capacity should be available.

Based on the consideration outlined above this project will apply the SEKF with fixed speed in the development of a SLAM for wireless networks implementation.

Status of SLAMWiN

After having implemented the filter, SLAMWiN is able to produce a location estimate based on triangulation of RSSI measurements. However, at this early stage SLAMWiN still requires additional information regarding the scenario. The required information is the location of APs in the scenario and the initial location of MUs.

Furthermore, SLAMWiN is not able to handle walls. It lacks both a detection mechanism for walls and the means to compensate for the effects they cause.

Chapter 12

SLAMWiN Core

This chapter will describe the development of the SLAMWiN Core, which covers detection and compensation of walls in a scenario where the position of APs and the start position of the MU are assumed known. In the following the approach and design is described first, after which the implemented design will be evaluated.

12.1 SLAMWiN Core approach

As mentioned before, the objective with SLAMWiN Core is to detect possible walls in an indoor environment scenario where walls will have a influence on RSSI measurements and thereby the localization precision, and then to compensate for this effect so that a satisfactory localization accuracy can be achieved.

The principle for detecting these walls is to utilize the radio shadow effect that occurs when a MU is moving behind or out from behind a wall. Due to the penetration effect some energy will be absorbed by the obstacle and the RSSI value will decrease when moving behind a wall. By looking at the RSSI values while going from e.g. LoS into non-LoS, or reverse, the signal strength will rise, or fall. This effect is illustrated in Figure 12.1 which shows RSSI measurements over time taken from an AP of a moving MU. The MU is moving behind a wall at approximately time 40 and then out of the radio shadow caused by the wall again at time 140.

One way to compensate for the walls is therefore to detect these jumps and, either by knowing or estimating the absorption loss through the given wall, then add the offset caused by the wall to the RSSI signal. In this way the wall will be neglected with respect to the received signal strength and unbiased localization should be possible. Because the thickness and material can vary from wall to wall, also the absorption loss can vary. In Table 12.1 different building materials are listed with their associated absorption loss in dB.

To detect a jump it is necessary to have measurements before and after the jump has taken place. This will of cause affect the way the Kalman filter is running because the filter has to recalculate the estimated path given new information about the past. This

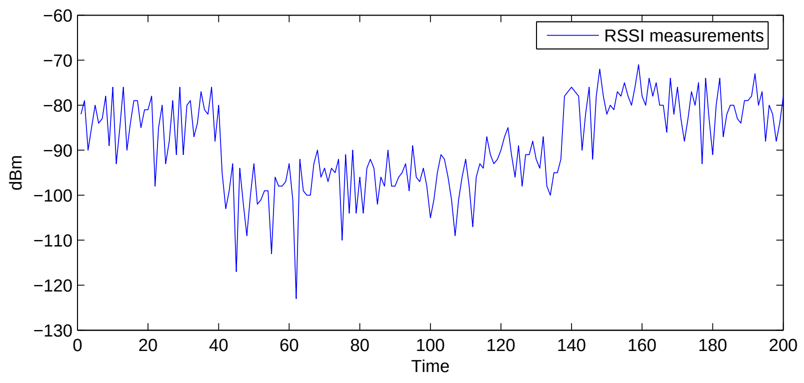


Figure 12.1: RSSI measurements over time of a MU moving into a shadow behind a wall and out of it again.

Material	Approximated absorption loss
Glass Window (non tinted)	2 dB
Wooden Door	3 dB
Cubicles	3 – 5 dB
Dry Wall	4 dB
Marble	5 dB
Brick Wall	8 dB
Concrete Wall	10 – 15 dB

Table 12.1: Typically path loss for different building materials for 2.4 GHz signals.[air]

recalculation is therefore necessary for each jump due to the delay before a jump is detected.

Another situation where recalculation is necessary is when the MU is starting behind one or multiple walls. Because no information is given about the wall(s) and the jump detector is not able to detect them at the start, the filter will start making erroneous localization. This error can be discovered when the MU is moving out from behind one of the walls. By having information about the wall, the path can be recalculated from the initial location with the new information. Therefore by keeping track of the number of signal rises which happens when moving out from a wall and signal falls when moving behind a wall, the localization system can easily detect when the path has to be recalculated from initial point, because it should be done if the number of rises is higher than the number of falls.

This approache is illustrated in Figure 12.2 which shows a flowchart diagram of the SLAMWiN Core. When a jump is detected this position will be marked with a flag which will contain information about the detected position, the estimated offset, and the state variables for the filter at the time where the jump has taken place. In that way, when it is necessary to jump back and recalculate from an earlier jump, the filter state can easily be loaded for the given time. Since it can be necessary to jump back to the initial point, this point is also marked with a flag.

As information about every estimated position is stored in the system, the flags mainly function as markers in the SLAMWiN Core system. They will, however, become more important in the SLAMWiN Knowledge sharing extension to the SLAMWiN Core. The reason for storing information for all estimated positions is that any jump detection in the current implementation requires the system to go back a number of measurements as described above. Since jumps cannot be anticipated, the system therefore always needs to know the filter state of previous measurements.

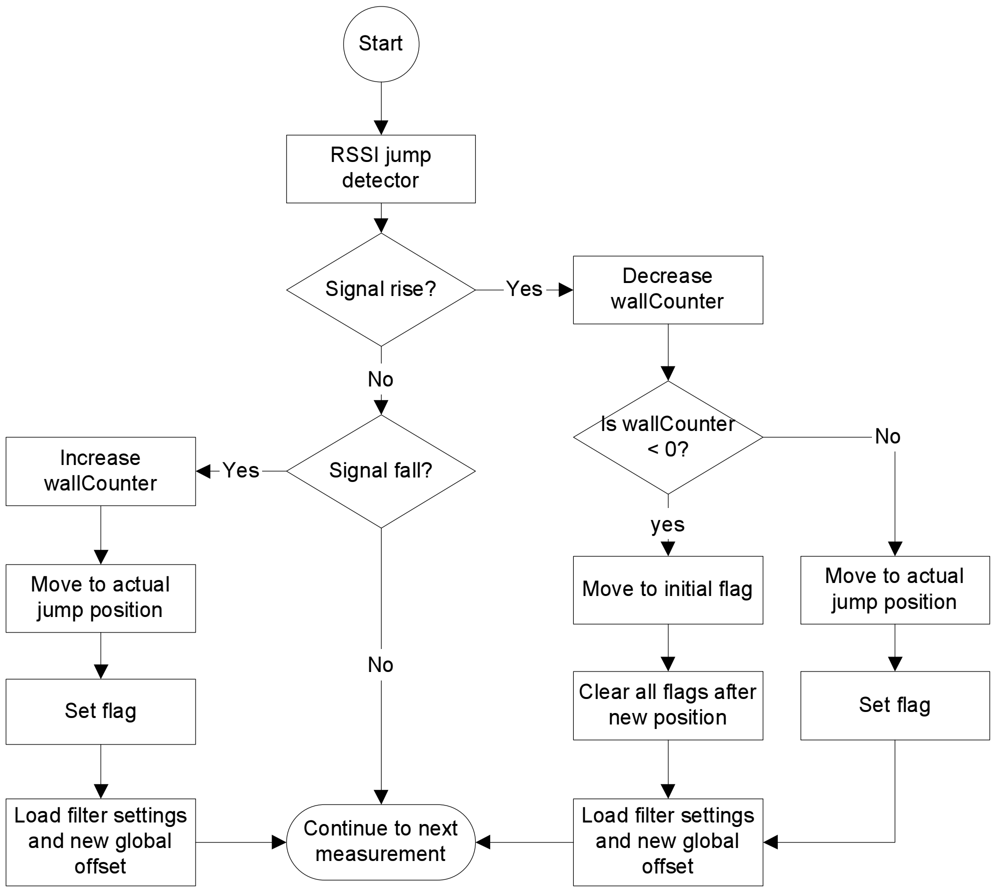

Start
RSSI jump detector
Signal rise?
Yes
Decrease wallCounter
No
Is wallCounter < 0?
No
Increase wallCounter
Yes
Signal fall?
yes
Move to actual jump position
Move to initial flag
Move to actual jump position
No
Set flag
Clear all flags after new position
Set flag
Load filter settings and new global offset
Continue to next measurement
Load filter settings and new global offset

Figure 12.2: Flowchart diagram of the SLAMWiN Core.

12.2 Design of SLAMWiN Core

The method used in this project to detect jumps is to apply two moving windows after each other on the obtained RSSI measurements. For each window the average RSSI value is calculated and the difference between the two values are found. An example of this is shown in Figure 12.3.

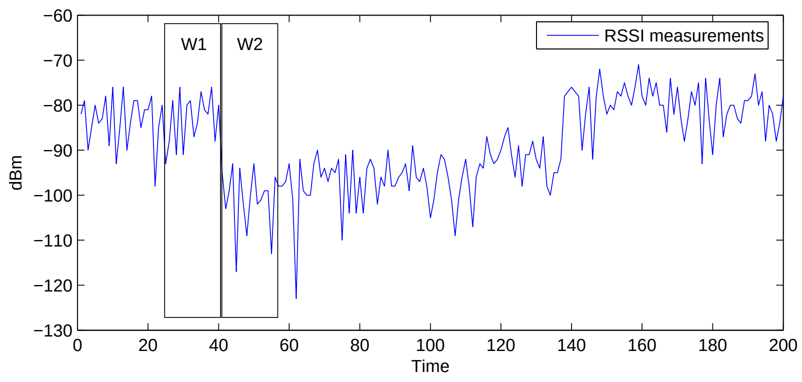


Figure 12.3: Two moving average windows applied on the RSSI measurements of a MU moving into a shadow behind a wall and out of it again.

By plotting these differences, peak values emerge which represent jumps in the signal. The reason for using average windows is to reduce the fluctuations caused by the noise on the measurements. However, caused by the moving window method, peaks will be delayed by the length of the last window. The size of the window should therefore depend on the degree of noise.

An illustration of this method applied on the previously shown signal 12.1, is illustrated in Figure 12.4, 12.5 and 12.6 where a window size on 2, 5, 10 is used on both windows respectively. The y-axis in the figures are the jump size in dB and the x-axis denotes the time in measurements. Two threshold lines are added at $\pm 11dB$ to distinguish between variation in the signal caused by noise and movement and variation caused by walls. Only peaks that are larger than the threshold are considered to be due to walls. If a peak consists of more than one measurement, the largest measurement is chosen as the actual jump.

The size of the threshold depends on the degree of noise and the expected abroption of the walls in the environment and will thus to a certain degree depend on the scenario. In Figure 12.6 jumps are detected at the time 31 and 128 which complies with the time for the actual jumps when considering the delay caused by the last window.

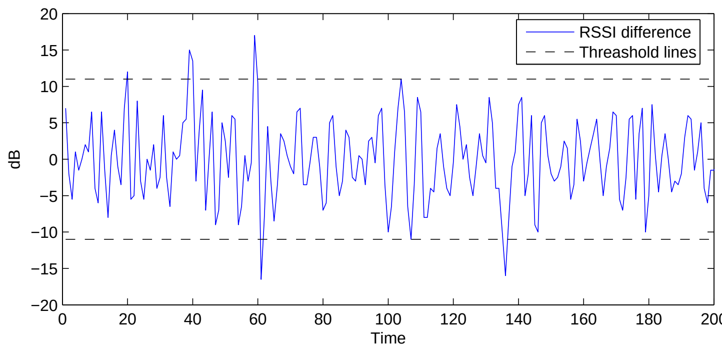


Figure 12.4: Detection of jumps using a window size of 2.

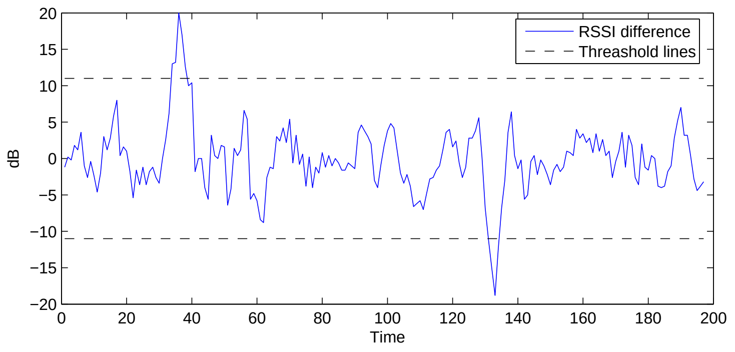


Figure 12.5: Detection of jumps using a window size of 5.

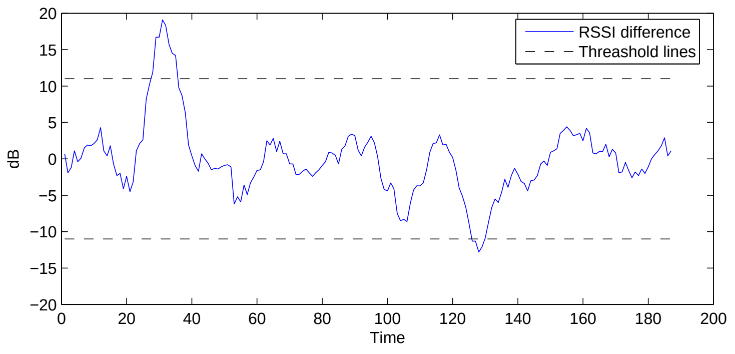


Figure 12.6: Detection of jumps using a window size of 10.

Figures 12.4, 12.5 and 12.6 indicate that a larger window size provides more significant jumps since noise is reduced.

12.3 Evaluation of SLAMWiN Core

This section will document the evaluation of the SLAMWiN Core. Two aspects will be evaluated, the jump detector and the general approach. The section will end with a short list of possible problems that can make out a thread to the chosen approach.

SLAMWiN Core approach

To evaluate to what extent the implemented SLAMWiN Core is able to detect walls, different scenarios are used. All scenarios in this evaluation are created by using the radio propagation emulator described in Appendix D on page 227. In all scenarios the wall absorption are assumed known because estimating the wall absorption will be treated in the next development step. The wall absorption is set to reduce the signal propagation with 15 dB which corresponds to a concrete wall.

Detection of walls along the path

In this scenario one wall is placed parallel to the MU moving direction. Five AP are spread out and the MU is moving from $(0, 0)$ to $(15, 0)$. The estimated path without using the SLAMWiN Core is shown in Figure 12.7. The Track-Metric error is 3.91 meter and the Path-Metric error is 3.74 meter. The tracking is good from the start and three meters ahead when all APs are in LoS. After that the estimated path is pushed away from the wall caused by the two upper right APs that are hidden behind the wall. These APs estimate a larger distance to the MU than there actually is due to lower RSSI measurements. Later on along the path also a third AP located in $(1, 7)$ loses LoS.

The reason for that the end of the path again approaches the real path is that the measurements taken during the final couple of meters of the path again are less influenced by the wall as only the AP in $(1, 7)$ has no LoS.

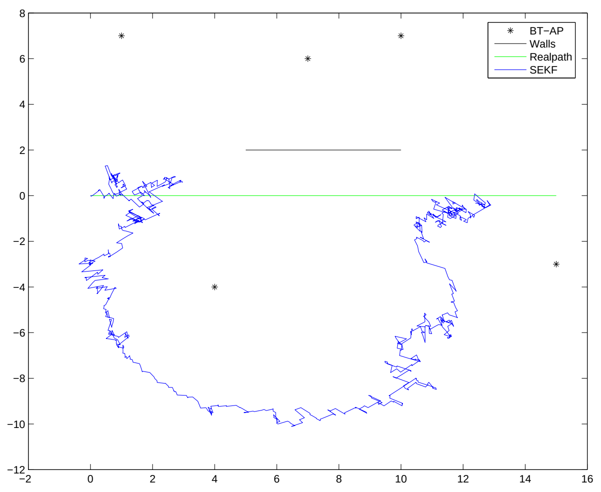


Figure 12.7: Tracking by using SEKF in a scenario with one wall.

In Figure 12.8 the same scenario is shown, but now the SLAMWiN Core is used. The Track-Metric error is 1.93 meter and the Path-Metric error is 0.52 meter. Now the estimated path follows the real path which means that the wall is detected and the offset for the APs located behind the wall is removed.

It is possible to deduce some general knowledge based on the observations made in this scenario. As walls are detected by changes in the RSSI measurements and these may be observed independently of each others, the results shown in this scenario may be extended also to apply for scenarios with several walls. Only limitation is that the initial location of the MU plus approximately the first meter of the path has LoS to all APs. Figure 12.9 shows a scenario with two parallel walls and it can be seen that the stated assumption holds for this scenario. Here the Track-Metric erroris is 2.04 meters and the Path-Metric error is 0.46 meters.

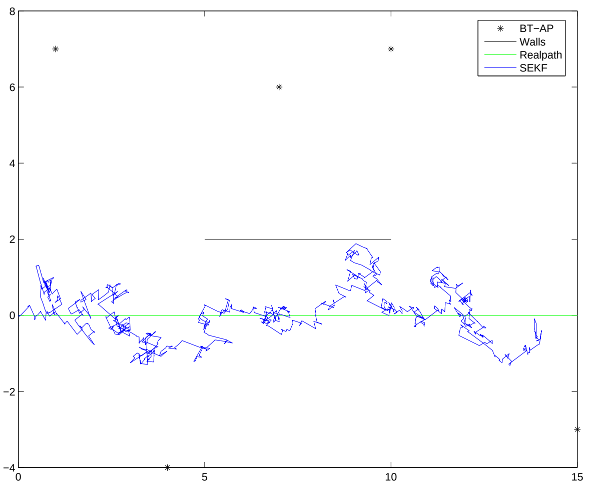


Figure 12.8: Tracking by using SLAMWiN Core in a scenario with one wall.

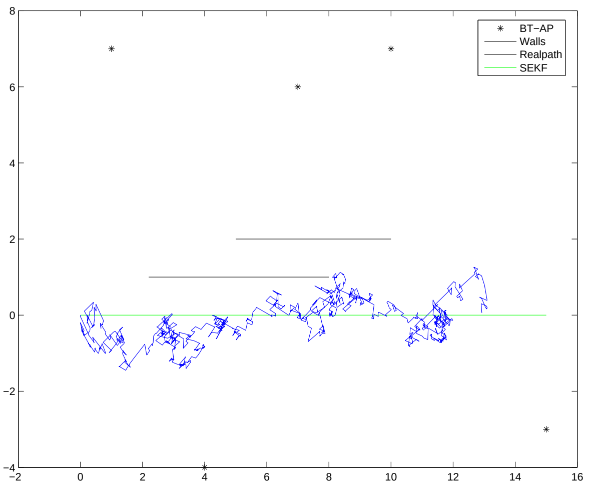


Figure 12.9: Tracking by using SEKF in a scenario with two walls.

Furthermore, some attention should be paid to a special case where the wall detection will not work. As described, jumps are detected based on differences in the average of the two detection windows that are larger than a threshold. Unfortunately, the situation where two jumps are detected shortly after each other can be caused by two very different situations.

The first situation is due to noise peaks close to actual jumps. In such a case the detected peak in the difference of the average between the two detection windows may look like the peak shown in Figure 12.10. The peak has a notch in the middle which causes it shortly to fall below the threshold. In order to avoid multiple detection of the same wall

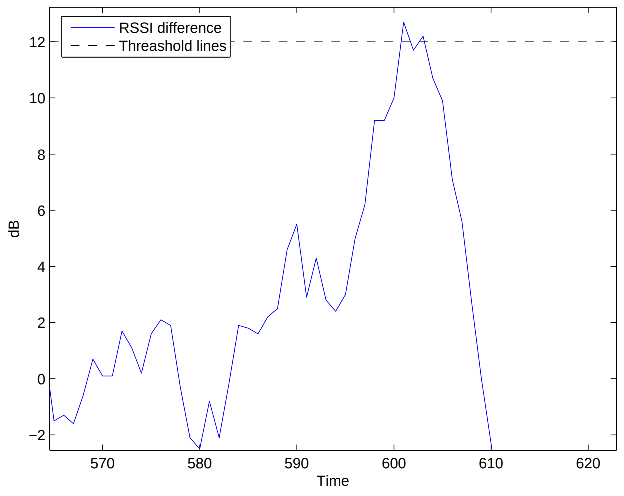


Figure 12.10: This figure illustrates the detection of a peak with a notch in the middle.

due to this notch the jump detector works with a quarantine period of 15 measurements from the same AP. Within this period no new jump may be detected. The second peak will thus be ignored.

However, while this quarantine solves one problem, it causes another. The second situation where two jumps are detected shortly after each other is the situation where two walls actually are causing two jumps shortly after each other. Because of the quarantine period, a second wall starting shortly after a jump detection will not be detected and future locations will be estimated wrong.

Since the situation with false detections of jumps due to peaks with notches occurs more often than situations where two walls are detected close to each other, the quarantine period is kept.

Detection of walls affecting the initial MU location

In this scenario two new walls are placed parallel to the MU movement direction and the AP positions are the same as in the scenario above. The MU is starting in a situation with non LoS to some of the APs. This means that the path has to be recalculated from initial point when the wall covering the initial point is discovered.

In Figure 12.11 the scenario is illustrated with the estimated path calculated by the SEKF without applying SLAMWiN Core. The Track-Metric error is 4.33 meter and the Path-Metric error is 4.33 meter. Again the path is pushed away from the wall, but now from the start as expected. Also the path again approaches the real path towards the end as for the same reasons as in the previous scenario.

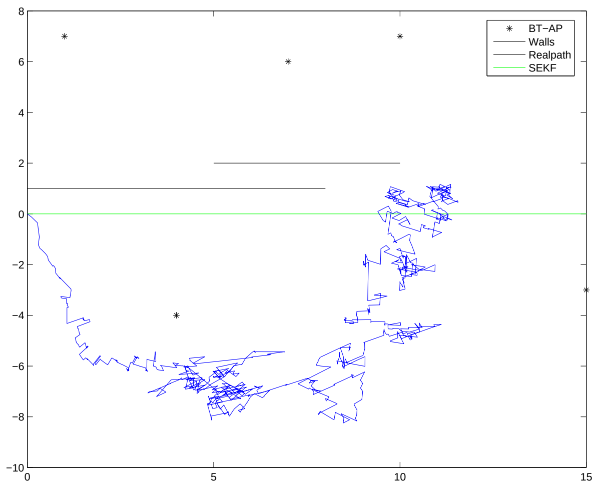


Figure 12.11: Tracking by using SEKF in a scenario with two walls and non LoS to all APs at the initial MU location.

The tracking by using the SLAMWiN Core is illustrated in Figure 12.12. The error metric is 2.14 meter when considering time and without considering time 0.50 meter. Again, SLAMWiN Core provides an estimated path that follows the real path.

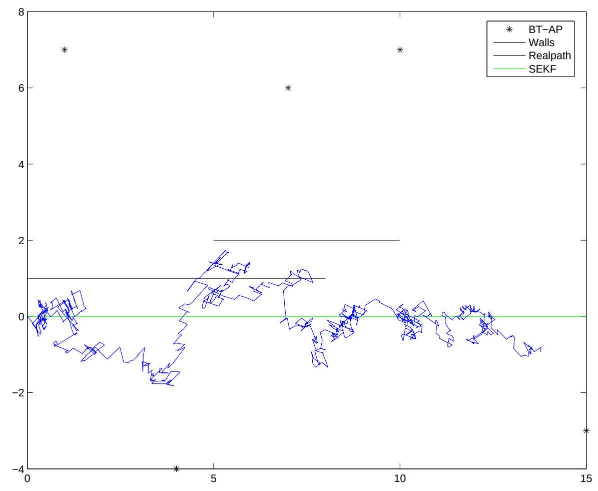


Figure 12.12: Tracking by using SLAMWiN Core in a scenario with two walls.

An additional remark should be made regarding scenarios where the initial location of the MU does not have LoS to all APs. As walls, shadowing the initial location, first can be compensated for when they are detected, this may lead to situations where the wall is detected very late, or not at all.

A wall between a MU and an AP will never be detected if the MU at no point during its movement in the scenario moves into a LoS situation. This will cause the complete estimated path to be biased by the missing offset of the undetected wall.

Another situation that may cause problems is a scenario where a wall affecting the path estimation starting from the initial location is first detected very late. This will cause recalculation of large parts of the path. In an off-line tracking situation this will make a difference in the needed computational power as the path recalculation in these cases requires considerable amounts of extra calculation. However, in an online tracking situation this will result in wrong location estimates for most of the path as the correct number of walls in the environment first is detected very late. Figure 12.13 illustrated this using the scenario shown in Figure 12.12 as example. The x-axis denotes the number of iterations and the y-axis denotes the time in measurements.

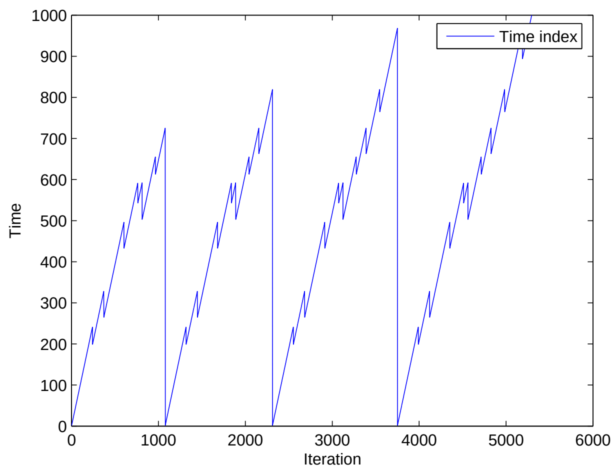


Figure 12.13: This figure illustrates how SLAMWiN Core jumps through the measurements.

It can be seen how jumps result in smaller recalculations and how the detection of the last wall for each of the three APs with non LoS to the initial location results in a larger recalculation.

Based on the evaluated scenarios it can be seen that the SLAMWiN Core is capable of detecting walls and thereby compensate for these even though the MU is starting with non LoS to some APs. By comparing the results with tracking where only SEKF is used it can be seen that tracking with SLAMWiN Core is considerable better.

Jump detector

First some general comments regarding the implemented jump detector should be made. The current implementation does not represent the best possible solution, but aims at providing a satisfactory detection capability with respect to the needed performance level in this project. The reason for this is that the task of fast detection of changes in a process corrupted by noise is not trivial and furthermore not the goal of this project. Therefore only limited effort was put into the development of this functionality and it may be investigated further if the work with SLAMWiN is continued after this project.

To see which consequence the choice of window and threshold sizes has on the jump detector, two analyses have been made based on measurements taken from the emulator. The purpose with the two analyses was to determine the probability for false positive and false negative jump detections when varying window and threshold size in the jump detector.

A false positive jump detection describes the situation where a jump is detected, but no jump has taken place. Because the RSSI measurements are very noisy, variation caused by noise can be interpreted as a jump. The number of false positive jump detection can vary depending on the window size and the used threshold in the jump detector e.g. if the threshold is set too low this will cause many wrongly detected jumps. If a false positive jump is detected, this will have a negative influence on the tracking, because the system will compensate for a wall which does not exist. This wrong compensation will in worse case lead to a wrong re-estimation of the complete path.

A false negative jump detection describes the opposite situation where a jump has taken place without being detected. Again the number of false detections depends on the window size and threshold size used in the jump detector e.g. if the threshold is set to high this may lead to no detection of jumps at all.

The results from the two analyses are described in detail in Appendix E on page 231 and they will in the following be evaluated together.

Evaluation of false negative and positive jump detections

Based on the two conducted analyses, pdfs for different window sizes for observed jumps caused by noise and observed jumps caused by actual jumps have been calculated. These will now be used to calculate the threshold that minimizes the probability of having either false positive or false negative detections.

Before the pdfs can be used to find the optimum threshold, where the probability for detecting one of the false detection types is minimized, it is necessary to consider the probability for a jump to happen at a given measurement. This is necessary since false positive jump detections only can take place if no actual jump is detected and false negative detections only can take place if a actual jump is detected. These circumstances are expressed in the following Equations 12.1 and 12.2.

$$P(\text{False positive jump}) = P(\text{Jump}|\text{Jump size}) \cdot (1 - P(\text{Actual jump})) \qquad (12.1)$$

$$P(\text{False negative jump}) = P(\text{No jump}|\text{Jump size}) \cdot P(\text{Actual jump}) \qquad (12.2)$$

Since the number of walls can vary from scenario to scenario, it is difficult to give a precise estimate of the probability of having an actual jump in a measurement ($P(\text{Actual jump})$).

Based on experiences form earlier experiments the probability is approximated to 0.5%. No detailed analysis will be made on this probability as it depends on scenarios and no value can be found that would be applicable to a wide range of scenarios. The optimal threshold for the jump detector should in theory be equal to the jump size that corresponds to the local minimum of the summation of two pdfs. At this point the probability of detecting neither a false positive nor a false negative jump will be smallest here.

Figure 12.14 illustrates an example of two pdfs that are generated with a window size of 5 measurements and a probability of having an actual jump of 5%. These values are chosen only for illustration purposes, as the real values generate such small pdfs for observing jumps due to actual jumps that they are not visible when shown together with the pdf for observing jumps due to noise. The dark blue pdf is the probability of observing a jump caused by noise and the light blue pdf is the probability to observing a jump caused by a actual jump. The light blue pdf is represented twice, as actual jumps both can be positive due to a rise in the observed RSSI signal, or negative due to a fall in the observed RSSI signal. The red line denotes the summation of the three pdfs.

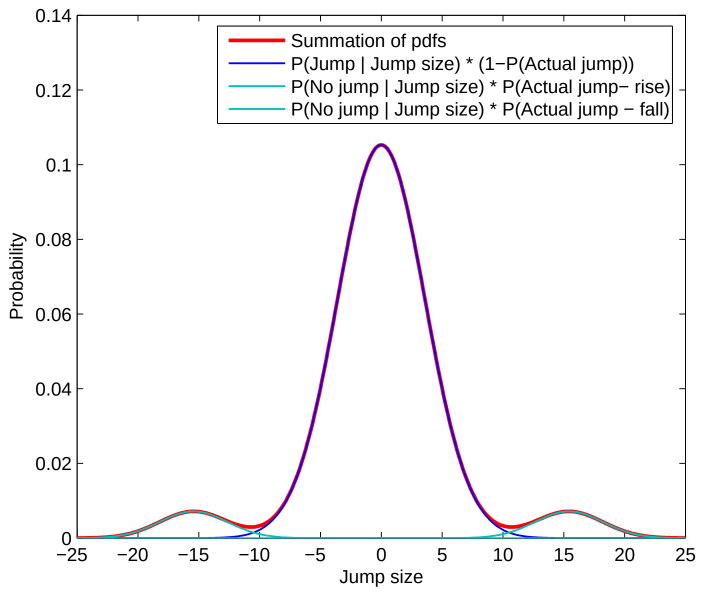


Figure 12.14: This figure illustrates an example of the pdfs for observed jumps due to either noise or actual jumps.

Figure 12.15 illustrates the same pdfs but for a window size of 20 and a probability of having an actual jump of 0.5%. The figure focusses on the part where the probability for false detections is minimized. It can here be seen that the jump detector should have a threshold of 10.5 dB in order to minimize the probability of false detections.

The evaluation of four different window sizes has resulted in the optimal thresholds shown in Table 12.2. The table furthermore shows an estimate of the probabilities for a false jump detection. The indicated probabilities have been calculated based on normal

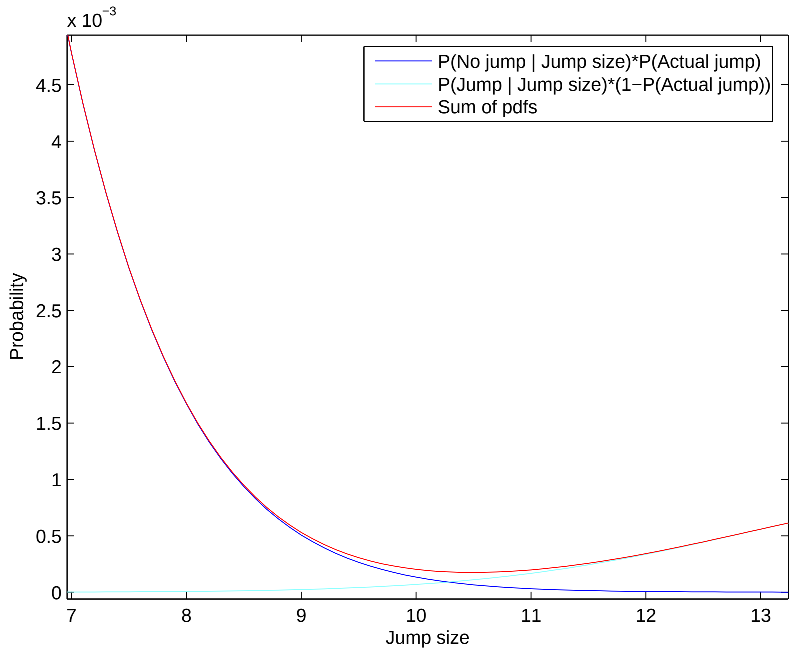


Figure 12.15: This figure illustrates a segment of the modified pdfs for false positive and false negative jump detections.

distributions fitted to histograms of measurements taken from the environment emulator as described in Appendix E on page 231. The real probabilities will be lower than the indicated values as the normal distribution describes a larger variation of measurements than the one found in both the emulator and real measurements.

A couple of remarks should be made regarding the shown evaluation results.

- The threshold for the smallest window size (window size 2) did not have a local minimum between the two pdfs. This is due to the pdf for observing a jump due to noise overlapped significantly with the pdf for observing jumps due to actual jumps. This describes a situation where it is difficult to distinguish between whether a jump

Window size:	Threshold:	P(false positive detection)	P(false negative detection)
2 measurements:	−−dB	−−	−−
5 measurements:	13.1dB	$1.2 \cdot 10^{-2}$	$2.02 \cdot 10^{-3}$
10 measurements:	10.5dB	$1 \cdot 10^{-2}$	$2.1 \cdot 10^{-4}$
15 measurements:	9.8dB	$1 \cdot 10^{-2}$	$6 \cdot 10^{-5}$

Table 12.2: This Table shows the optimal thresholds for four different window sizes and the resulting probabilities for detecting the two types of false detections.

is detected due to noise or due to an actual jump. The window size of 2 has thus been deselected for further investigation as it is too small to be useful for jump detection.

- The threshold values are determined based on a scenario where the MU movement is perpendicular to the AP as illustrated in Figure 12.16.

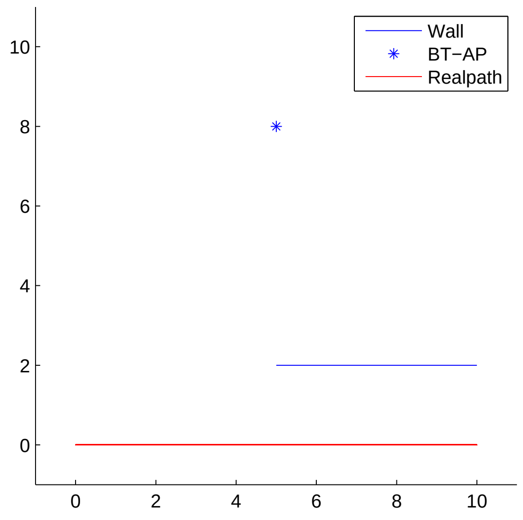


Figure 12.16: Scenario used to determine the probability for false positive jump detections.

The size of detected jumps depend on how the MU moves in relation to the AP. The used thresholds are based on an average jump size where the jump size is neither increased or decreased due to movement of the MU. The worst case, which would correspond to a MU moving away from an AP while experiencing a positive jump in the RSSI signal, is here not considered. Basing the threshold on such a scenario would cause it to be smaller but with larger probabilities for detection of false jumps.

- The probability of having an actual jump in a measurement (P(Actual jump)) that, based on experiences form earlier experiments, is estimated to be 0.5% is a important parameter. If this probability decreases this results in a larger probability for observing jumps due to noise and a smaller probability of observing jumps due to actual jumps. This will have a direct affect on the threshold. A larger chance of having jumps will result in a lower threshold and vice versa. This shall be considered when applying the jump detector to new scenarios.

The described approach should be applied on real life measurements in order to evaluate the window size and threshold level and to get the best combination of threshold and window for the final SLAMWiN system. However, the task of finding the distribution of real life measurements will not be carried out due to the limited gain in knowledge about the jump detector. The measurements from the emulator already provide the needed indication of how the threshold and the window size depend on each other and the real life measurements would thus only contribute by finding the exact values.

Jump detector applied on real life measurements

To verify that the jump detector also works on real life measurements, a simple test is performed. The setup is illustrated in Figure 12.17 where the used wall is made of concrete. The RSSI measurements are illustrated in Figure 12.18 and the result from the jump detector is illustrated in Figure 12.19 where a window size on 15 is used. Here it can be seen that the jump detector is able to detect the wall and estimate the absorption to be approximately 11 dB. Based on this very simple real life test it can be concluded that the jump detector is able to detect a real wall. It should be mentioned that the wall used in the test is a relatively thick concrete wall and is still only estimated to be approximately 11 dB. By using a threshold on 10 dB in the jump detector this could be a problem because variations in the detections easily could result in a false negative detection. If the jump detector should be used in real life it would be necessary to optimize the jump detector so a lower threshold can be used.

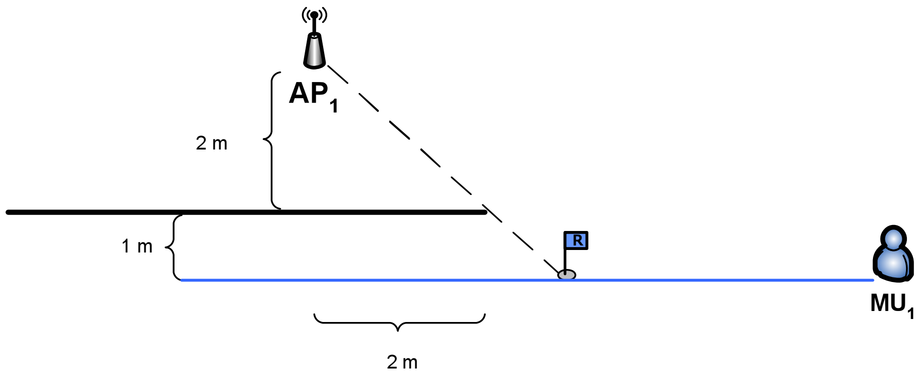


Figure 12.17: Real life scenario to verity the jump detector.

Limitations of the current jump detector

From the results shown above it can be seen that the window size has a major effect on the detection quality and that it should be kept as high as possible while having the increase in recalculation in mind. The following will describe some of the limitations and problems which the current jump detector suffers from and it will also introduce some methods that may solve the problems and extend the capabilities of the jump detector.

Heavy noise: One of the problems with the moving window averages is that the threshold has to be set so that jump is detected and the noise is ignored. In scenarios with heavy noise the difference between relevant jumps and noise peaks may be very small and it will thus difficult to set an appropriate threshold.

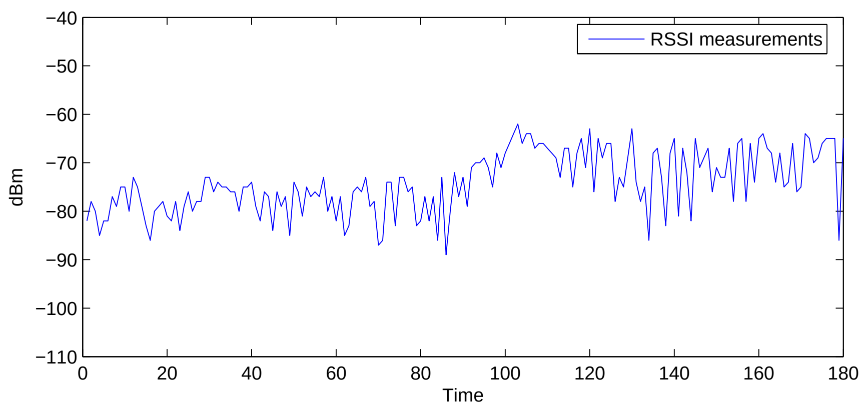


Figure 12.18: RSSI measurements from a real life scenario used to verity the jump detector.

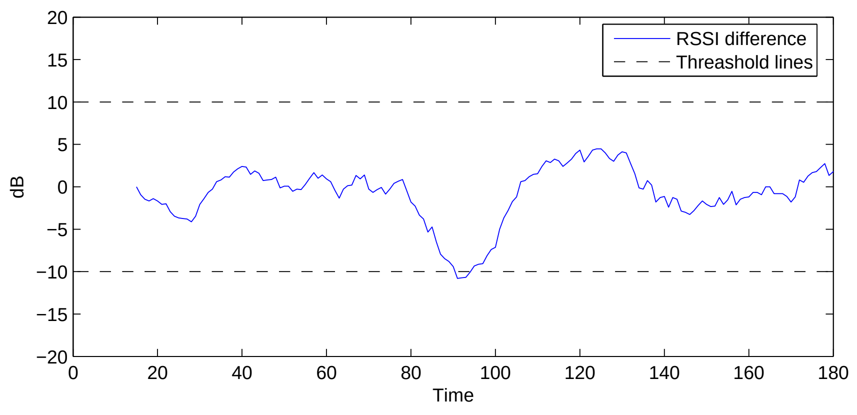


Figure 12.19: Jump detector applied on real life scenario.

Caused by the design of the jump detector, noise peaks in the measurements will normally be observed as a opposite jumps shortly after each other in the jump detector. This is illustrated in Figure 12.20 were a noise peak at measurement 135 is followed by a similar noise peak at measurement 139. Using this knowledge it could thus be possible to exclude wrongly detected jumps if they are followed by a jump with the opposite sign e.g. a fall followed by a rise because this would probably just be noise in the RSSI measurements.

Path recalculation: As described above and illustrated in Figure 12.2 the detection of a jump in the RSSI measurements will cause the system go back a number of measurements in order to set the flag on the actual position at which the jump took place. It was also described that the number of measurements that the system needs to go back is equal to the size of one of the windows used in the jump detector.

This exposes a trade off in the implemented jump detector. A larger window size

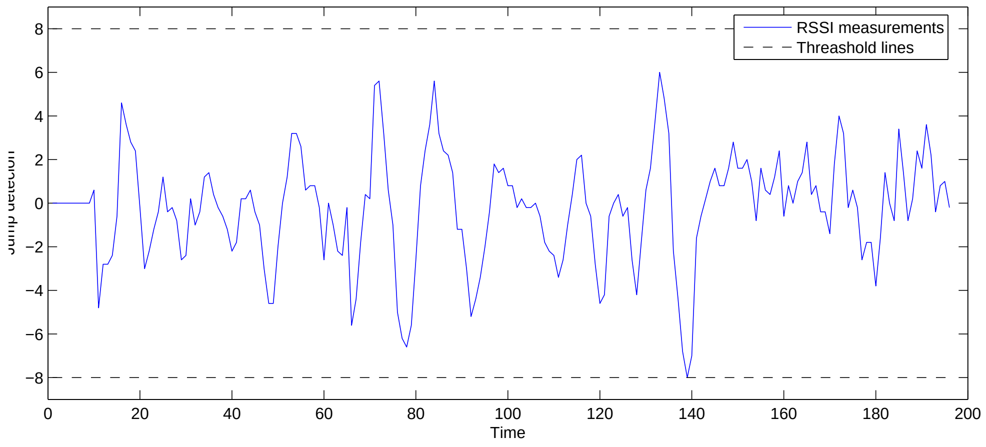


Figure 12.20: A noisy measurement sample.

that provides more noise resistant jump detection will entail more recalculation and thus increase the overall needed computational power of the SLAMWiN system. Furthermore, a larger window size results in that the path is estimated wrongly during a longer period until it is corrected through recalculation. This of cause is also a unwanted effect of larger window sizes. The window size used in the jump detector should thus be as small as possible in order to avoid unnecessary recalculation.

Map granularity: Another problem with the current use of moving windows in the jump detector is that they introduce a lower bound of how short walls it can detect. This problem may cause two different effects depending on the length of the wall, the window size and the quarantine period after a jump detection. If the window size and the quarantine period is held constant and the length of a wall that has to be detected is iteratively shortened, one will observe the two effects in the following order:

- The first effect is introduced when the number of measurements that are obstructed by the wall gets lower than the number of measurements defined in the quarantine period. In that case the start of the wall will be detected and the following measurements will be modified. However, since the end of the wall is detected during the quarantine period, the jump detector will not react on it and the rest of the path will be estimated wrong. This is shown in Figure 12.21 where the three upper APs react on the beginning of the short wall in the middle of the scenario but not the ending.
- The second effect is introduced when the wall is shortened even more. In this case the window size becomes relevant as jumps are detected by calculating the difference between the two moving windows averages and comparing this to a threshold. If not enough RSSI measurements are affected by the wall, the

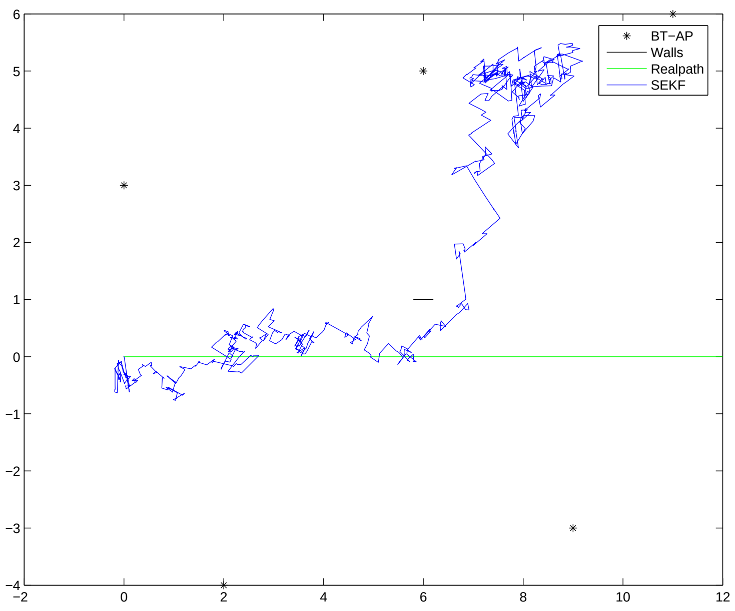


Figure 12.21: A path estimation is badly affected by a short wall due to the quarantine period in the jump detector.

calculated averages will not become large enough to result in a jump detection.

It is difficult to provide a clear solution to the problems with the jump detector described above. On one hand it would be preferable to use a small window size so that the amount of recalculation is kept low and shorter walls can be detected. But on the other hand a larger window size provides a more noise resistant jump detection.

The best window size for the jump detector depends thus on the scenario and should, as a rule of thump, be kept as small as possible where the lower bound of the size is defined by the noise in the specific scenario. The lower bound could be found by a set of test measurements in the scenario where the jump detector shall be applied.

Possible problems

There are still some problems with the approach used in SLAMWiN Core that occur either due to detection problems, or due to especially difficult scenarios that can have a very negative effect on the tracking. These issues are described in the following.

False positive jump detection: False positive jump detections will, as described above, have a negative influence on the tracking, because the system will compensate for a wall which does not exist. This wrong compensation will in worse case take place for the rest of the time after the detection.

One way to cope with this problem is to store previous MU paths in a virtual map in the system and somehow compare if multiple users has detected a jump in the same area and if not then neglect the detection. This will of cause introduce some problems in the initial phase because the virtual map is empty, but hopefully the map will converge when more path are added to the map. This map approach is investigated in the next design step of SLAMWiN.

False negative jump detection: False negative jump detections will, like the false positive jump detection, have a negative influence on the tracking as for the same reason as in false positive jump detection case.

This problem is more difficult to handle. Since nothing is detected, there exist no indication of that the system may be tracking a MU wrong. The SLAMWiN system should therefore aim at minimizing the possibility for false negative jump detections by allowing more false positive jump detections since these can be detected and handled.

MUs not leaving shadows: This is the situation where a MU at no point during its presence in a scenario has LoS to a specific AP. The problem with this situation is that no jump will occur for the wall between the MU and the AP. This means that the wall will never be detected.

This problem could also be solved by using the previously introduced virtual map. Instead of relying on a jump caused by the wall and in that way detect the wall, information about previous paths could be used to make an guess on the initial wall offset for the MU. This assumes that the previous MUs have detected the wall and the initial position is known.

MUs moving out of AP range: The described approach is based on the observability of jumps and the possibility to review the complete sequence of jumps from a MU's initial location to its current. As long as a MU stays within the range of all APs this is not a problem as there are no 'gabs' in the measurement history. However, if a MU leaves the range of an AP and thus cannot receive measurements from it, the MU will not be able to detect jumps due to new walls that block the LoS to the AP. This is a problem in the chosen approach, as MUs leaving the range of an AP with a known number of walls between the MU and the AP cannot be sure that this number has not changed when the MU returns into range of the AP. The MU may very well have moved in behind another wall while being out of range and the assumed number of walls would thus be wrong.

The SLAMWiN system could handle this problem by considering a MU returning into the range of a AP as a new MU for this specific AP. This way the localization of the MU will not rely on possibly false data, but will be initialized again as if the MU had not previously been observed by the AP. This means that the additional absorption by walls and the wallcount to the AP could be estimated based on shadows covering the estimated re-entry point of the MU into the AP range.

For simplicity, the scenarios introduced in this report will abstract from MUs leaving the range of APs. MUs moving out of AP range will therefore be a task under further work.

Status of SLAMWiN

By adding SLAMWiN Core to the Kalman filter the system is now able to detect walls in a given environment and compensate for these so that a satisfactory localization accuracy can be achieved. However, the system requires that the start position of the MU and the position of all APs in the scenario are assumed known. Furthermore, the SLAMWiN Core assumes that the path loss due to walls is known and that APs have a unlimited range so that MUs always have connectivity to each AP in a scenario.

The current status is thus a SLAMWiN system that is able to perform localization in an indoor environment under the condition of knowledge regarding both APs and wall absorptions in the scenario. Hence, the PnP capability of the system is at this point very limited.

Chapter 13

SLAMWiN Knowledge sharing

This chapter will describe the development of an enhancement to the SLAMWiN Core. The enhancement will use the history of all MUs in order to provide new functionality to the SLAMWiN system while also enhancing some of the existing functionality.

As in the previous chapter the approach and design of SLAMWiN Knowledge sharing will be explained first after which an evaluation of the component is followed.

13.1 SLAMWiN Knowledge sharing approach

The SLAMWiN Core approach described in the previous chapter uses only measurements from the MU it tries to estimate the location of. However, one may easily think of a scenario where more than one MU passes through e.g. the museum from the project scenario. If one could assume that the environment does not change significantly over time, it could be an advantage if measurements from other MUs were combined with the measurements from the currently tracked MU into a knowledge base. This knowledge base could then provide a more comprehensive description of the environment. This functionality is basically what the SLAMWiN Knowledge sharing component adds to the SLAMWiN system.

The motivation for including knowledge obtained from other MUs in the localization process of a MU is that it is expected to provide better real time tracking capabilities and that it will enhance the SLAMWiN systems capabilities to estimate properties of the environment such as wall locations and wall absorptions. It is at this point still assumed that the position of APs and the initial position of MUs are known.

Flags and shadows

The SLAMWiN Knowledge sharing approach is based on flags described in the previous chapter. These flags are set along every path any MU has travelled in a given scenario. As described, these flags mark locations were MUs experience changes in the RSSI due to walls. Figure 13.1 shows a path travelled by a MU. Along the path the MU experiences two significant changes in the RSSI, which were caused by a wall. These, together with the initial position, are marked with flags as described in the previous chapter regarding the SLAMWiN Core.

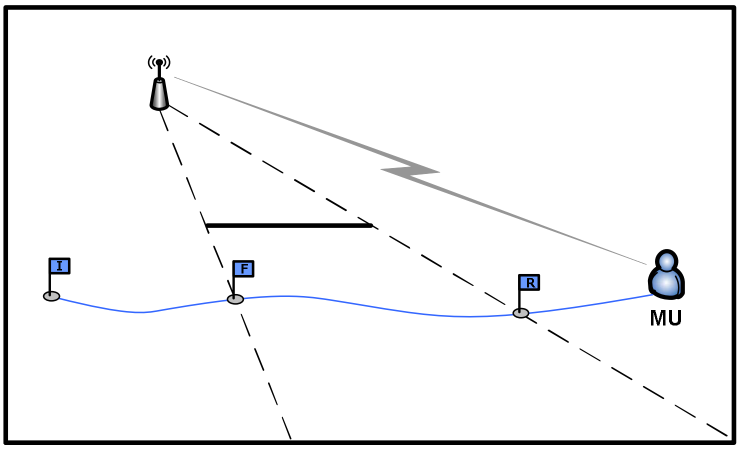


Figure 13.1: This figure illustrates the flags set along a path travelled by a MU through a scenario.

For each scenario a knowledge base will thus be build by iteratively adding all flags set by MUs passing through the scenario. As the knowledge base will hold information regarding all flags and the SLAMWiN Knowledge sharing component also can access information regarding APs in the scenario, a virtual map of the scenario can be build. This map is build by calculating 'shadows' caused by the estimated walls in the scenario.

The definition of shadows is that they describe an area in the scenario that has the same properties with respect to LoS-conditions to a specific AP. An example of this is shown in Figure 13.2 where the shadows caused by the path from the previous figure are illustrated. The shadows illustrated in the figure are defined by two angles, two distances and a reference to a flag which holds information regarding the relevant AP, the wallcount and the offset caused by the wall.

This information is enough to describe the area that is covered by the shadow and the LoS-condition to a specific AP. The wallcount states the estimated number of walls between any point in the shadow and a specific AP while the offset sums up the absorption from the walls. The angles are used to calculate two lines that each start in the AP and bounds the shadow on two sides.

The angles are set so that each of them points towards a location estimate from the given AP. The spread between the two angles of a shadow is determined by a predefined number of location estimates on the path that will be located within the shadow. If the first angle points towards location estimate number 350 from a MU, the second angle will point towards estimate $350 + dist$ from the same MU. Here $dist$ is a number of location estimates that shall be located within the shadow. The number can be varied according to the wanted resolution of the shadows. However, a new shadow is always starting if a flag is encountered.

The reason for dividing shadows into smaller parts than the ones defined by flags is that

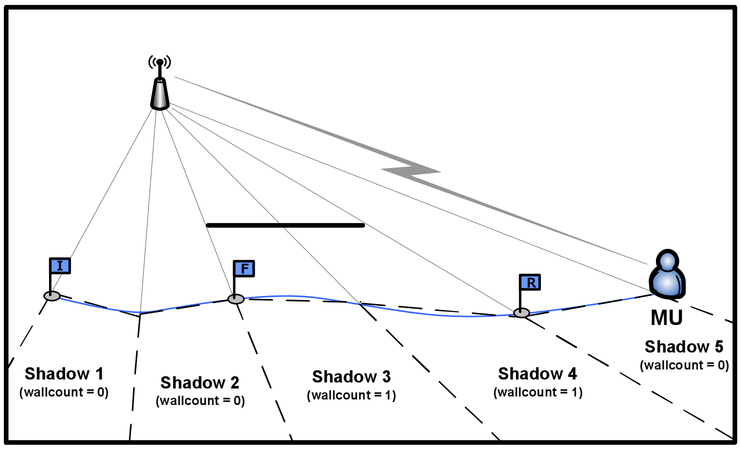


Figure 13.2: This figure illustrates the shadows that are set along the path of a MU through a scenario. The dashed line indicates an approximation of the path.

the lower boundary of the shadow is described by an approximation of the path. The path is approximated by fitting a straight line to the part of the path that is bounded by the two angles. Since there can be many measurements in-between two flags, the narrower shadows guarantee a reasonable approximation of the path as a maximum length in measurements of the approximated part of the path is defined.

Finally, the shadows are not considered to have a maximum range and in theory they thus have an infinite upper boundary. This property is based on the consideration that the direct signal from an AP to a MU located behind a wall, no matter how far away, will always be influenced by the wall. In practice the radio range of an AP will of cause be limited because of the propagation channel is not infinite.

Sharing flags and shadows for enhanced performance

This leads to that for every MU passing trough the scenario numerous shadows will be calculated for each AP that observes the MU. If we again focus on the virtual map of the scenario, one can now imagine that this will hold a huge number of shadows. Every point in the map will, over time, be covered by one or more shadows describing the estimated wallcount and offset to any AP in the scenario. The time frame of this depend on the movement pattern of MUs in the scenario. Furthermore, the number of flags in the scenario will increase.

These two aspects can be used for two different approaches of knowledge sharing between the MUs. The shadows from previous MUs allow for estimation of both wallcounts and offsets to APs in every location in the virtual map and the flags from previous MUs indicate locations in the virtual map where a moving MU can expect to move from LoS to non-LoS, or vice versa. Figure 13.3 shows two MUs moving through a scenario causing both flags and shadows to be set in the virtual map. For simplicity, shadows in the figure are only set according to flags.

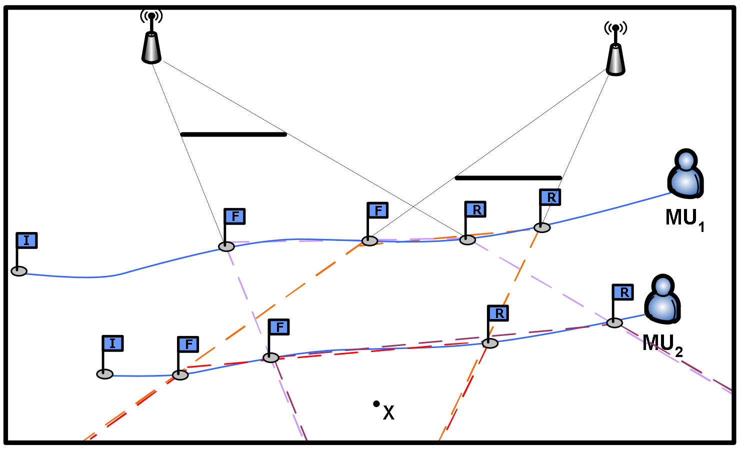


Figure 13.3: This figure illustrates how shadows with common AP overlap each other.

Using shared shadows

Figure 13.3 illustrates how shadows from different MUs but common AP may overlap each other. This way the estimation of the wallcount and offset to a specific AP given a specific location can be estimated based on knowledge from several MUs. The shadows from both MU_1 and MU_2 can be used to estimate the wallcount and offset to each of the APs in location X. If all shadows hold information that state the same wallcount and offset this is not a problem. However, since detection of flags will not be 100% accurate, there can occur situations due to false detection of jumps, where shadows will have different wallcounts and offsets although they cover the same area. In these situations, combining knowledge from all shadows covering the location can be used to estimate the correct wallcount and offset.

The estimation of the wallcount and offset for a given location is in this project mainly used as part of the initial location estimation, but can be used for different purposes. The initial location estimation is an important part of the SLAMWiN system with respect to its PnP capability. There is a large difference in the needed human effort between a SLAMWiN system that needs manual input of the exact initial location for each MU and a SLAMWiN system that does not need any specific information about the initial location. Different approaches have been considered which provide varying degrees of PnP capability. Three fundamentally different approaches are described in the following.

Basic triangulation of unknown location: The first, most basic, approach is simply to triangulate the initial location of a MU. This can be done by using the Equations 3.1 to 3.6 from the *Localization methods* chapter (Chapter 3) which defines a least square approximation of the initial location of the MU. This is done based on the average of the first measurements from three different APs that can observe the MU. The chosen number of measurements to average over depends on the wanted performance. Averaging over a larger number of measurements will provide a better protection against noisy measurements, while averaging over a small number of measurements will provide a faster result. The preferred choice will here depend on the scenario.

This simple approach can provide a fast estimate of the initial location without needing any human interaction with the SLAMWiN system. However, as this approach does not consider possible walls in the environment it will not be very reliable. The triangulation will provide false estimations of the initial position if any of the used APs does not have LoS to the MU. Even in very simple scenarios the estimation can be off by several hundreds of meters and the approach is thus not preferable.

Estimation of offsets via known location and shadows: The second approach is based on the assumption that the initial location is known. This could either be due to a default initial location defined in the SLAMWiN system, or due to manual configuration for each MU. Based on the known location and knowledge from shadows generated by previous MUs it is then possible to provide an estimation of the wallcount and offset to any AP that covers the initial location of the MU.

Keeping in mind that unknown walls that cover the initial location of a MU will cause the SLAMWiN system to recalculate the complete path when the wall is detected, the advantage of this approach is clear. It significantly reduces the need for recalculation for any MU of which the wallcount and offset can be estimated at the initial location.

Unfortunately, this is not possible for the first MUs in a scenario as their initial location may not be covered by shadows of which they can take advantage. Altogether this approach is not very PnP-minded as it requires knowledge of the initial location. As PnP capability is one of the wanted characteristics of the SLAMWiN system this solution is not preferable.

Estimation of location by running paths backwards: The third approach takes a radically different way. Based on observations of tracked MUs that, with a wrongly estimated initial position, over time find the right path, this approach initially defines that all MUs start in $(0, 0)$. $(0, 0)$ is chosen since the coordinate is also used for one of the APs in the scenario and it will thus be represented every scenario. The estimated path of these MUs will in general be wrong until the SEKF has compensated for the wrong initial location. This will, depending on the distance from $(0, 0)$ to the actual initial location, take several measurements and is therefore not by itself a good solution to estimating the initial location of MUs. It is therefore necessary to do more.

This is done by considering that the measurements of all MUs are stored in the system. Having the measurements available it is possible to run the measurements again in opposite direction. As the last estimated location of the MU mostly will be close to its actual location, this will result in tracking the MU path backwards. Through this the initial location of the MUs can be estimated.

This knowledge is not of relevance to the MU itself as the location estimates will come too late for it to be relevant for its tracking. However, if this procedure is performed for all MUs in a scenario the location estimates can be used to estimate if there exist locations at which MUs will be more likely to enter a scenario. This will typically be the case in indoor scenarios as MUs are expected to enter through doors, or to be activated at approximately the same locations.

When groups of initial locations of MUs can be identified this can be combined with knowledge regarding wallcounts and offsets at these locations. The location of MUs entering a scenario can then be estimated by calculating the probability for the MU being in one of the locations.

The last approach has been chosen based on the PnP capability it provides. It will be described further in the design section later in this chapter.

Using shared flags

As an alternative to shadows, flags can also be used to share knowledge obtained from previous MUs with a MU moving through a scenario. The shadow based method described above has advantages if the wallcount and offset for a limited number of locations in the virtual map has to be determined. However, if the knowledge about wallcounts should be used in the ongoing tracking of a MU, this would require to check each location that the MU is estimated to be in. This approach does not perform very well, as it would require an increasingly large number of calculations as the number of shadows in the virtual map increases.

Furthermore, shadows may cause problems in situations where their offset is used for the tracking of MUs. It is critical for a correct location estimation that the estimated offset approximately corresponds to the actual offset caused by walls. A too low offset will pull the estimated location to much towards the AP for which the offset applies and a too high offset will push it away. Since different MUs detect walls at different times, the offset provided by shadows from previous MUs will in most cases not correspond to the offset that is valid for a new MU. The tracking will therefore suffer considerably. This problem is related to another problem that was encountered in an approach to use groups of flags for enhanced tracking and MUs which is discussed on the next page. The problem will be illustrated with an example there.

A flag based approach is thus chosen that focuses on changes in the wallcount rather than on the current value for each new location estimate of the MU. The approach thus uses the assumption that the wallcount is constant for all locations between two lines marking the beginning and end of an area affected by the shadowing of a wall and that changes in the wallcount and offset most likely take place near these lines defined by flags. This is done by considering the flags set along a path as these together with APs define a line, at which changes in the wallcount and offset to a specific AP are likely to take place. This can be seen in Figure 13.3 where the flags for the two different paths are set on a line going through both flags and an AP.

The shown example is an ideal case where the flags fit the line perfectly. For real location estimates this will not be the case, as the estimated position of flags is not completely accurate and different MU may experience the effects of a wall at slightly different locations. The approach therefore groups flags based on there wallcount and angle to each AP in groups that hold all flags that are likely to have been caused by the same wall. Based on the flag groups a groupflag is set that holds information about the wall absorption that a wall causes. This offset is an average of the offsets estimated by all MUs that have detected the wall. Over time this shall lead to a better estimation of the absorption caused by the wall and this information can be passed on to new MUs detecting the wall. A MU detecting a jump close to a relevant groupflag will thus use the offset passed on by the groupflag while returning its own estimated absorption to the groupflag.

The angle from an AP to this groupflag is found by calculating the mass-center of all flags that are included in the group. It thus defines the most likely angle from the AP at which the effects of a wall start. Furthermore, the group of flags is used to calculate the distance to the groupflag. Since there is a risk that a flag group contains flags that should not have been part of the group, the distance to the groupflag is not set equal to the distance of the flag closest to the AP. Instead the distance to the groupflag is set equal to the distance of the flag that is closest after removing a the closest tenth of the flags in the groupflag.

The initial implementation of SLAMWiN pursued the idea of relying only on knowledge gained by previous MUs when the virtual map had converged. In this case a MU should have ignored jumps detected by its own jump detector. A MU entering a scenario should continue to detect jumps and set flags, but the idea was to rely only on the established groupflags to tell a MU when it was entering or leaving a shadow thus using only the knowledge base and disregarding any knowledge obtained by the jump detector. The converged state was defined as a state where most new flags could be assigned to already established groups. However, this implementation was flawed in two ways and was abandoned.

The first problem was that it turned out to be difficult to make a satisfactory definition of when the virtual map had reached the converged state that did not involve manual inspection of the map by a user of the SLAMWiN system. This interaction was not wanted since it would limit the PnP capability of the system. The second problem was that MUs do not experience the effects of a wall at the same location in the virtual map. This was a problem since these effects influenced the estimated location, which again was used as basis for the comparison with the groupflag. As illustrated in Figure 13.4 this meant that MUs experiencing the effects of a wall before reaching the groupflag would be pushed away from the wall, and thus also the groupflag. For this approach to work the MU shall start to compensate for the effects of walls at close to measurement at which they start. Since it was not possible to detect this point without running the jump detector, another approach was necessary that used both the jump detector and the knowledge gained by previous MUs.

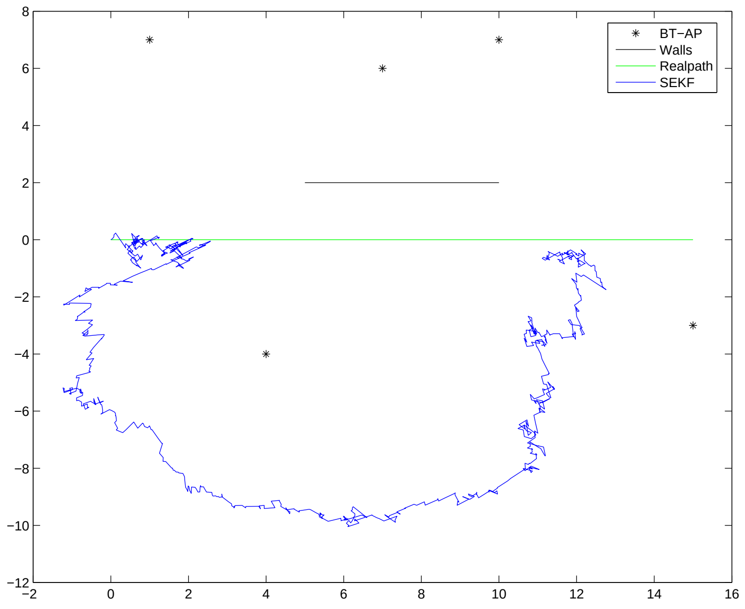


Figure 13.4: This figure illustrates the how the tracking of a MU performed in the initial SLAMWiN approach. The MU is clearly affected by the wall before it starts to compensate for its effects.

New jump detector approach

Instead of the approach described above another approach was introduced that continued to use the jump detector and used the knowledge base for validation of the detected jumps. This entailed a change in the jump detector described in the previous SLAMWiN Core chapter.

As something new, two window sizes were introduced, a large window and a small, which are used in parallel. The reason for using a combination of a large and a small window was that the approach should combine the advantages of both window sizes: the reduced need for recalculation when using a small window and the increased reliability of detected jump when using a large window. The large window size would thus be larger than the estimated window size described in the previous chapter and the small window size would be smaller.

The approach starts with a situation were the jump detection based on the large window is trusted and allowed to set flags while any jump detection based on the small window is rejected. This will over time create a number of flags in the virtual map and create groupflags. In this initial phase the use of the large window leads to a large amount of recalculation caused by the size of the large window. However, as groupflags start to appear in the virtual map, the jump detection based on the small flags becomes relevant.

Jumps detected with the small window size are trusted if they are detected close to groupflags and are rejected if they are further away. The reason for this is that the groupflags mark locations in the map at which jumps are expected to take place. The disadvantage of small window sizes, which is a larger probability of false positive and true negative detection of jumps, is in this case reduced considerably.

In order to handle two jumps detected by two different window sizes the new jump detector has become more complicated. The flowchart diagram shown in Figure 13.5 shows a simplification of the approach that is necessary to handle the two window sizes.

For each new measurement it is necessary first to evaluate if some basic conditions, such as a required minimum number of measurements since the last flag, are satisfied. If these conditions are meet, the jump detector can check wether or not a jump has been detected with the small window. If a jumps is detected with the small window size, it is necessary to evaluate if it confirms to a set of requirements, e.g. if it has been detected close to a group flag. If the detection lives up to all requirements a non-shared flag is set. 'Non-shared' means that this flag cannot contribute to the formation of groupflags and has to be confirmed by jump detection based on the large window.

First after having checked for eventual jumps detected with the small window, the jump detector will evaluate the detection done with the large window size. If a jump is detected, a set of rules that have been defined for jumps based on large windows will be evaluated. This can lead to three different outcomes:

1. Either a new shared flag is set because the detected jump has not been detected before, or was detected with the small window but too far away from a groupflag.

2. A non-shared flag is replaced by a shared flag because the jump has confirmed a jump detected with the small window close to a groupflag.

3. SLAMWiN jumps back to either the previous flag in order to remove it, or to the initial flag because a new wall has been detected.

Each new measurement can thus lead to no detection of jumps, jump detection with the small window, jump detection with the large window, or jump detection with both windows.

The expected results of the introduction of SLAMWiN Knowledge sharing are:

- Increased real time capability.
- Better estimates of the wall absorption of walls in the scenarios.
- More reliable detection of flags due to larger window sizes in the jump detector.

13.2 Design of SLAMWiN Knowledge sharing

In terms of functionality the SLAMWiN Knowledge sharing component includes three larger extensions to the existing system. (1) The ability to estimate the wallcount and offset to any AP for a given location in the virtual map based on shadows, (2) the ability to detect changes in the wallcount based on groupflags and (3) the ability to estimate the initial position of MUs. The following will describe how the implementation of these three functionalities has been designed.

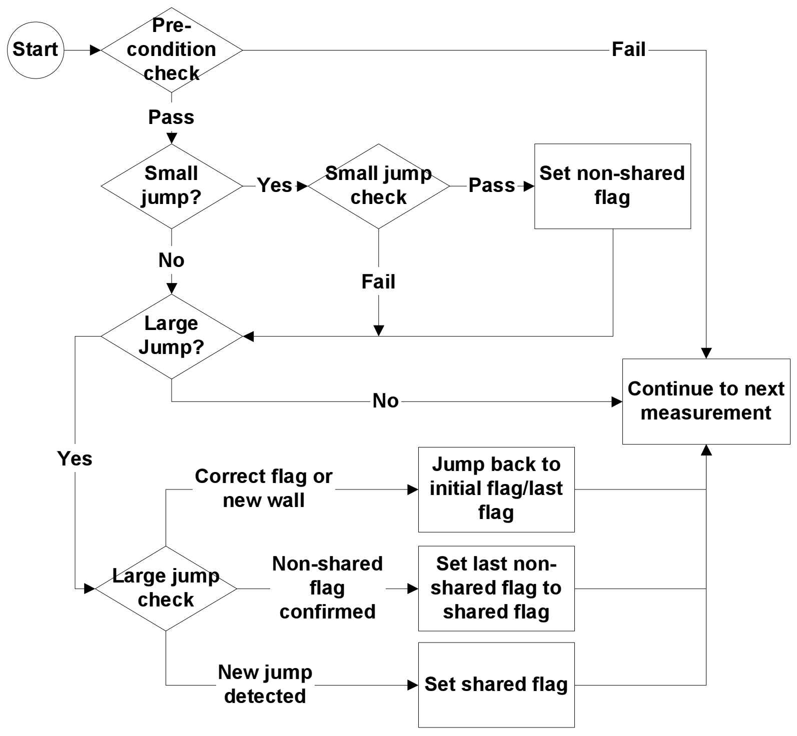


Figure 13.5: This figure shows the flowchart diagram used in the jump detector when using SLAMWiN Knowledge sharing.

Wallcount and offset estimation based on shadows

The first extension to the SLAMWiN Core is the ability to estimate the wallcount and offset for a given location. When the wallcount and offset for a given location to a specific AP is determined based on shadows there will typically be several shadows to consider. As previously discussed, it should be expected that these will not always agree on the wallcount and offset. An example of this can be seen in Figure 13.6. The example demonstrates that there can be different reasons for the shadows not to agree on the same wallcount and offset. Typically three situations exist:

MU has no knowledge of wall: This is the case for MU_1 that passes the wall on the side at which the AP is located and thus has no information regarding the wall. This results in a shadow with a wallcount of 0.

MU has correct knowledge of wall: This is the case for MU_2 that passes the wall on the opposite side of the AP and thus provides the 'correct' wallcount for the location at which the wallcount is estimated. This results in a shadow with a wallcount of 1.

MU has wrong knowledge of wall: This is the case for MU_3 that passes the wall on the opposite side of the AP but has estimated the wallcount wrongly. This could be due to e.g. a false jump detection. This results in a shadow with a wallcount of 2.

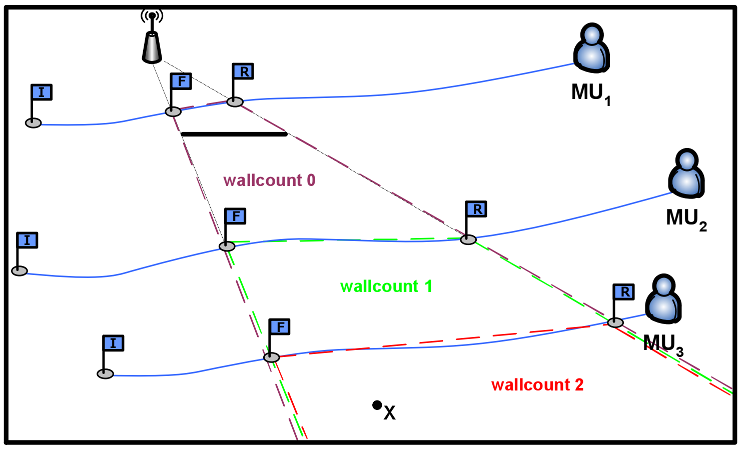


Figure 13.6: This figure illustrates the how several shadows with different wallcounts cover the same location.

There are two problems in this. Possibly 'wrong' shadows caused by MUs that have made false positive jump detections and shadows that are caused by MUs that do not have any knowledge of relevant walls. In order to handle this, the wallcount for a given location is chosen based on a 'highest significant group' approach.

This approach sorts all wallcounts and chooses the highest wallcount that is represented by a significant number of shadows. A significant number is in this case defined as more than 5% of all shadows covering the location at which the wallcount is estimated. This number has been chosen based on the assumption that, on average, less than every 20^{th} flag in the map will will be set on a false location.

Wallcount and offset estimation based on groupflags

The second extension to the SLAMWiN Core is the flag grouping ability. The Mean-shift algorithm is here again use to group flags according to there angle to APs. This means that the angle from all APs to all flags is considered and that flags with almost the same angle to a specific AP are defined as a group. This is done based on the assumption that flags originating from different MUs with the same angle to an AP are set because of the same wall and thus describes the same change in wallcount and offset. The grouping furthermore considers the wallcount of the flags so that two different walls ending on the same line may be separated. This is illustrated in Figure 13.7 where the coloured circles represent different flag groups. The Mean-shift algorithm is used to evaluate the angles and find groups of flags that have a similar angle. The algorithm has furthermore been modified so that it only groups flags that have the same change in the wallcount. This results in that the flag groups 3 and 4 in the illustrated example may be separated based on their wallcount.

The used algorithm is furthermore able to define a lowest number of flags required in order to detect a group of flags. This is used to counteract the fact that the jump detector

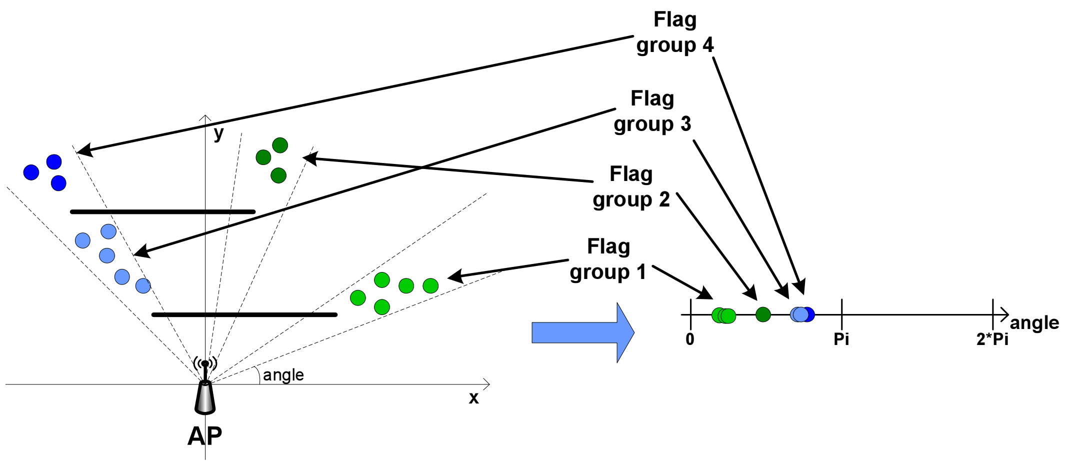


Figure 13.7: This figure illustrates the how flags are grouped in the virtual map.

introduced in the previous chapter may detect jumps caused only by noise and may thus produce flags that are wrong. A requirement to the minimum number of flags with a similar angle to an AP ensures that randomly distributed flags in the virtual map will not result in flag groups, which again makes the flag groups a more reliable indicator of walls.

The approach is based on the assumption that flags set due to false positive jump detections will be distributed more than flags set due to walls. In practice it is difficult to make a clear definition of the lowest number. As the number of flags in the virtual map increases over time, the required minimum number of flags cannot be specified as a fixed number. If it was, the over time increasing number of false positive jump detections would at some point result in enough flags so that false flag groups could appear. Furthermore, the lowest number will depend on factors such as the complexity of the scenario, on how distributed the traffic is and on how large the percentage of false positive detections are.

Therefore, the required minimum number of flags used in this project has been identified by observing how flag groups behaved in some of the evaluation scenarios. Based on these observations the required minimum number of flags is defined as 2% of all flags set in the virtual map. As the use of percentages is a problem in the initial phases where only few flags are set, another requirement is defined; the required minimum number of flags must be larger than 10.

The flag groups are used to describe an area in which jumps detected with small window sizes are trusted. This area is a rectangle that lies parallel with the line from the AP through the groupflag and that has a width of 5 meters, 2.5 meters to each side of the line. The relative large width is necessary because, as mentioned earlier, the tracking of the MU is tailing behind approximately 2 meters. Due to possible wrongly detected flags the rectangle does not start at the flag with the shortest distance to the AP. The closest 5% of the group are sorted out and the closest flag after this is chosen to define the distance to the AP at which the rectangle starts. The area continuous along the line outwards. An illustration of the area is given in Figure 13.8.

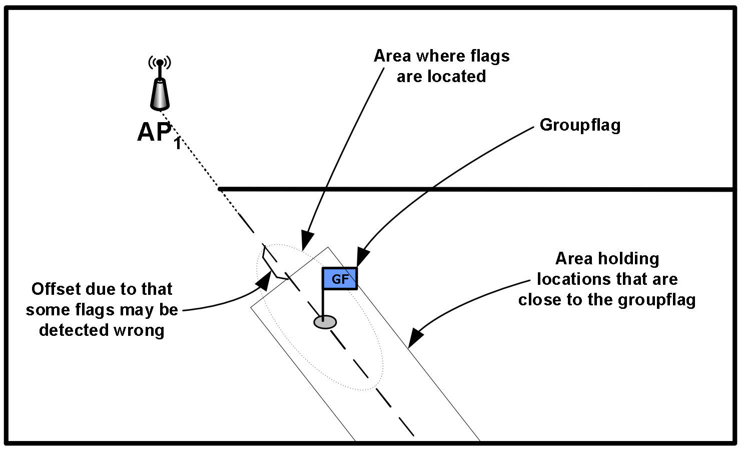

AP1
Area where flags are located
Groupflag
GF
Area holding locations that are close to the groupflag
Offset due to that some flags may be detected wrong

Figure 13.8: This figure shows the area in which jumps detected with the small window size are trusted.

Estimation of the initial location

The third extension to the SLAMWiN Core is the ability to estimate the initial MU location. The first step of the process of estimating likely entry locations for MUs is trivial. The initial location of MUs is simply set to $(0, 0)$. Also the task of estimating the MU path backwards is not difficult as all measurements are stored in the system and can be run backwards through the SEKF. This will provide a path similar to the paths normally estimated by SLAMWiN.

Beyond this it is necessary to find likely entry points based in the initial locations estimated by running the measurements backwards through the SEKF. This is done by using the Mean-shift algorithm[Che95] which can find locations that are likely to belong together in groups. Furthermore, this algorithm can calculate the mass-center of these groups which then defines the likely entry location to which wallcounts and offsets are estimated.

After having determined one, or more, likely entry points the wallcount and offset to each of this locations is found by using shadows in the virtual map as described above. For new MUs entering the scenario it will then be necessary to triangulate its location based on the first measurements from different APs that are modified by the previously found offset. This must be done for each of the combination of offsets that are found for the different likely entry points identified until that time. The estimated locations will then be compared to the likely entry point locations and the location that provides the smallest location error is chosen. If only one likely entry point has been identified at the time a new MU enters the scenario, this entry point will be used.

13.3 Evaluation of SLAMWiN Knowledge sharing

This section will document the evaluation of the SLAMWiN Knowledge sharing component. The evaluation will first focus on the three main aspects of the SLAMWiN Knowledge sharing approach: The ability to estimate the initial location, the ability to provide better real time localization and the ability to estimate wall absorption. It will then make a list of possible problems and how these may be treated.

In general it should be noted that many aspects of SLAMWiN Knowledge sharing are scenario depended. Much of the evaluation performed in this chapter is therefore also scenario depended. This limits the possibilities for making general performance evaluation based on statistical results that can be generalized. However, an evaluation based on selected scenarios will still give a good indication of the performance.

Estimation of the initial MU location

The SLAMWiN system has through the SLAMWiN Knowledge sharing component gained a higher PnP capability through its ability to estimate the initial location without human interaction. This is in the following evaluated by first investigating the SEKF's ability to compensate for false initial locations and then by looking at the Mean-shift algorithm in more detail.

Compensation for false initial locations estimates

The approach for estimating the initial location of MUs is based on that SLAMWiN places the first couple of MUs entering a scenario in $(0, 0)$. After that the SEKF must be able to estimated locations that get increasingly closer to the path as the tracking progresses along the path.

In order to evaluate this, measurements are made in a scenario where a MU moves along a straight line. The estimated initial location is then placed at different locations with an increasing distance to the actual initial location and it is observed how the SEKF compensates for the wrong estimate. The following cases are evaluated:

- Initial position on path.
- Initial position 2 m away from path.
- Initial position 4 m away from path.
- Initial position 8 m away from path.
- Initial position 12 m away from path.

The results from this evaluation are shown in Figure 13.9. The figure shows that the SEKF is able to handle initial locations estimates that are located up to 12 meters away from the actual initial location in this scenario. The SEKF is even able to compensate for the

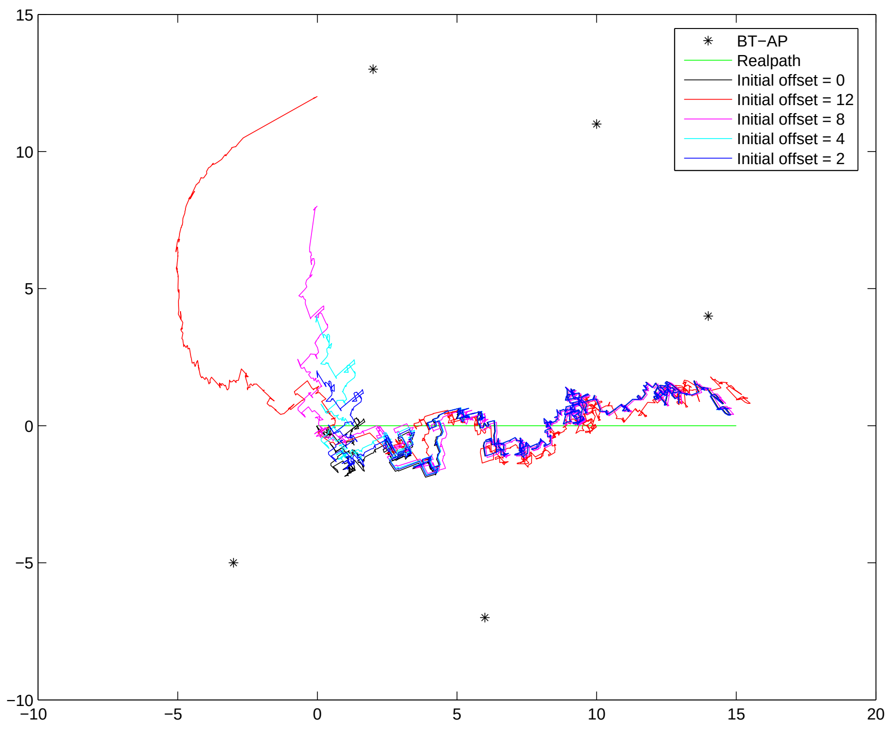


Figure 13.9: This figure illustrates the estimated path of a MU for the same set of measurements but for different initial locations.

wrong initial location so fast that the average location error does not suffer considerably. This is illustrated in Table 13.1 that shows the average location error for the estimated paths. The metrics introduced in Section 11.4 are here again used to illustrate the average fault.

Initial location offset:	Track-Metric:	Path-Metric:
Initial offset $= 0$	$2.166m$	$0.820m$
Initial offset $= 2$	$2.172m$	$0.820m$
Initial offset $= 4$	$2.229m$	$0.870m$
Initial offset $= 8$	$2.366m$	$1.024m$
Initial offset $= 12$	$3.015m$	$2.066m$

Table 13.1: This table illustrates the average location error measured with the two previously introduced metrics.

There is no reason to believe that these results will be any different for scenarios that have walls. Here the recalculation of the path will result in measurements that are similar to measurements obtained without walls.

It should be noted that situations were observed which the SEKF was not able to handle. These situations are in general scenarios where the initial location was estimated to be outside the group of APs available in the scenario. In these situations the path was estimated to continue outwards away from the actual path and the APs. This is not a problem as the initial location always will be located inside the AP group as $(0, 0)$ will always be inside the group.

Evaluation of the Mean-shift algorithm

Beyond the compensation for false initial location estimates, the proposed approach for estimation of the initial MU location depends on the ability to identify groups of initial locations that are considered likely entry point to the scenario. The implementation of this identification has been based on the Mean-shift algorithm that requires knowledge regarding the required minimum number of members in groups and knowledge regarding the expected spread of the members. Finding values for these two parameters that are valid for a large range of scenarios are difficult. A general evaluation of both parameters are discussed in the following.

Minimum number of required members of a group: When the total number of estimated initial locations in a scenario increases, this will, for a static minimum number of required members in a group, lead to that the Mean-shift algorithm will identify more and more groups. These groups will be due to real likely entry points that have a large concentration of estimated initial locations, but also due to wrongly placed initial location estimates that by chance are close to each other.

By considering that groups due to real likely entry points are expected to have a higher concentration of members than the wrong groups, this problem can be solved by making the minimum number of members depended on the total number of estimated initial locations in the map. This is illustrated in the Figures 13.10 and 13.11 that show two examples of a virtual map where initial locations are plotted.

The figures shows that the concentration of members in the groups at the likely entry point increases as more and more initial location estimates are added. In Figure 13.11 the required minimum number of members are increased. This results in that the group of estimated initial locations that is marked with the red circle is not marked as a likely entry point in the scenario. This group would have been identified if the minimum number of required members of the group had not been increased.

However, it is difficult to make a general description of the required minimum number of members in the groups. The ratio between the number of initial location estimates that are placed at a likely entry point and the number of initial location estimates that are not depends heavily on the scenario. Parameters such as number of actual entry points in a scenario, the size of the actual entry points and number of falsely estimated initial locations are scenario depended.

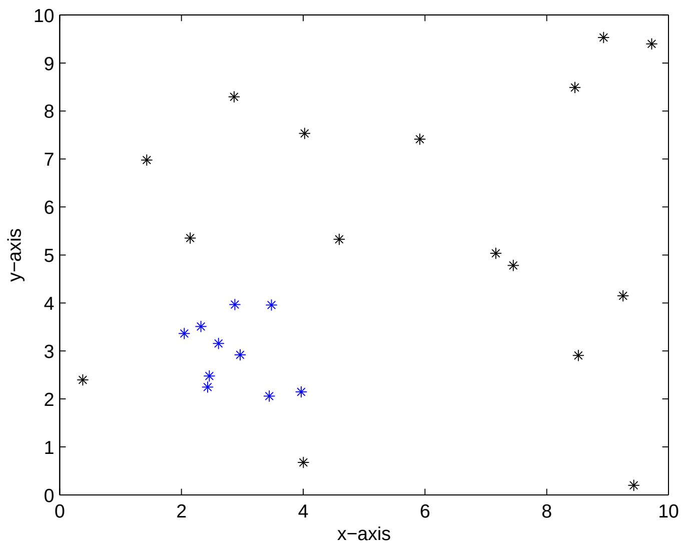


Figure 13.10: This figure illustrates a virtual map with 25 initial location estimates. The locations marked blue are estimated to be part of a group of likely entry point.

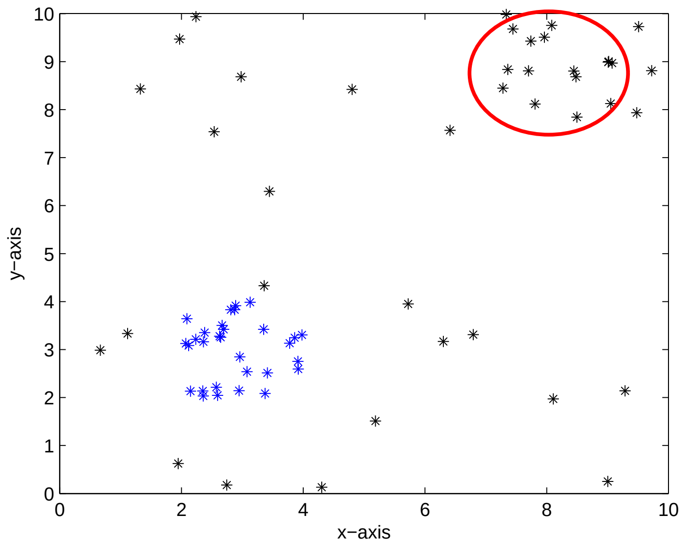


Figure 13.11: This figure illustrates a virtual map with 65 initial location estimates. The locations marked blue are estimated to be part of a group of likely entry point.

It is therefore necessary to use another algorithm than the Mean-shift algorithm if the PnP capability of the SLAMWiN system shall be supported better. This algorithm should be able to identify groups without the need for a scenario depended minimum number of required members in a groups. Such a algorithm was not investigated in this project as the Mean-shift algorithm performed good enough to provide a proof of concept. It is expected that such an algorithm will be available as the described problem arises in many contexts such as pattern recognition and image analysis.

Allowed location spread in groups: The second parameter of the Mean-shift algorithm suffers under the same problems as the required minimum number of member is a group: It depends on the scenario. The allowed spread describes how far away members of a group can be while still being identified as a group.

As the intention with the Mean-shift algorithm is to identify estimated initial locations that belong to the same likely entry point to a scenario, the spread of such locations will depend on the inaccuracy of the estimation and the size of the actual entry point. A considerably larger spread should thus be allowed in scenarios where the entry point is a 4 meter wide gate and the scenario suffers from heavy noise that worsens the location accuracy as compared to scenarios where the entry point is a small door and the noise is less strong.

The Mean-shift algorithm is thus shown not to be suitable for the task of finding likely entry points. This further underlines that the Mean-shift algorithm should be exchanged if SLAMWiN is taken beyond a proof of concept stage.

Real time localization ability

Another advantage that the SLAMWiN system has gained through the SLAMWiN Knowledge sharing component is an enhanced real time localization capability. The following will evaluate how the use of a modified jump detector and the ability to estimate wallcounts and offsets affects the SLAMWiN systems ability to perform localization in real time.

Faster jump detection

The introduction of SLAMWiN Knowledge sharing entailed that the jump detector was modified. The new jump detector used two window sizes in order to detect jumps: one that is larger than the previously used window size and one that is smaller. The previous jump detector used a threshold of 10.5 dB and a window size of 10 measurements. The new jump detector will be evaluated for the same threshold, as this was shown not to have a large effect on the jump detection. The window size will be 5 for the small window and 15 for the large. These values are chosen so that the small window will react as fast as possible on jumps with a very little noise filtering and the large was chosen so that the chance of making a false detection is minimized.

Two situations shall be used as basis for a comparison of the original jump detector and the new jump detector. The first is a situation illustrating the initial phase of a scenario where no groupflags have been set and the second situation is the same scenario, but with all groupflags placed in the virtual map.

Jump detection without groupflags: As the small window size is only used when a jump is detected close to a groupflag, only the large window size can be used for jump detection in the initial phases of MU localization as there will not be any groupflags. This means that the new jump detector will always use a window size of 15 for the detection of jump, compared to the window size of 10 that the old jump detector uses.

The window sizes describes the number of measurements taken from one specific AP that will be jumped back. Since measurements are taken from different APs in a random manner where the probability of taking a measurement from an AP is equally distributed on all APs, this means that a jump detection results, not in 'windows size' number of measurements that need to be recalculated, but on average in 'window size · number of APs' measurements. In the light of this it is clear that the new jump detector will result in on average $\frac{15}{10}$ times more measurements that need to be recalculated than the old jump detector in the initial phase of a scenario where no groupflags have been set.

Jump detection with group flags: When groupflags are introduced, the calculation of the average number of measurements that needs to be recalculated caused by jumps becomes more complex.

Jumps that not are detected close to groupflags will still require $\frac{15}{10}$ more measuremetns to be recalculated by the new jump detector, but as the purpose of groupflags is to mark locations where jumps are likely to take place, this entails that these jumps will be unlikely.

Jumps correctly detected close to groupflags will with the new jump detector, on average, only require half the number of recalculated measurements that was necessary with the old jump detector. This is due to the reduction of the applied window size from 10 to 5 measurements. However, as the use of the smaller window size entails a larger probability of making false positive detections, attention should also be paid to this situation. A false positive detection close to a group flag will first result in '5· number of APs' recalculated measurements due to the false jump detection with the small window size. After that the location of a MU will be estimated wrongly in '15· number of APs' number of measurements until the large window is able to evaluate the jump detected with the small window. If the jump found with the small window size cannot be confirmed by using the large window size, this will result further '15· number of APs' recalculated measurements.

Thus three situations can occur resulting in different numbers of recalculated meaurements:

1. Jumps detected away from groupflags: 15 measurements.
2. Correctly detected jumps close to groupflags: 5 measurements.
3. Wrongly detected jump close to groupflags: 5 + 15 measurements.

In a fully converged map most jumps will be correctly detected jumps close to groupflags for reasons described above. In such a situation the new jump detector will outperform the old. How significant the increased performance is is difficult to determine as factors such as noise and the complexity of the environment is scenario depended. In general more noise and a more complex scenario will lead to a smaller performance increase as this will lead to more wrongly detected jumps and jumps detected away from groupflags.

It should furthermore be noted that it will take longer time before all groupflags in a complex scenario are found and that the performance gain with the new jump detector therefore will take longer time to become significant. The actual time before all groupflags are detected also depends on the movement pattern of MUs in a scenario.

Overall it is expected that the new jump detector will reduce the needed recalculation in scenarios where the virtual map is either partly or fully converged, meaning that most relevant groupflags have been set.

Initial wallcount estimation

More even than the new jump detector, the ability to estimate wallcounts and offsets for the estimated initial location reduces the needed recalculation. If the wallcount and offset for the initial location of a MU is estimated correctly, this will lead to that walls that obstruct the LoS to APs from the initial location will not any longer lead to recalculations. An example of this is illustrated in Figure 13.12 where the initial location of a MU has no LoS to three different APs. During the tracking of this MU SLAMWiN is forced to go

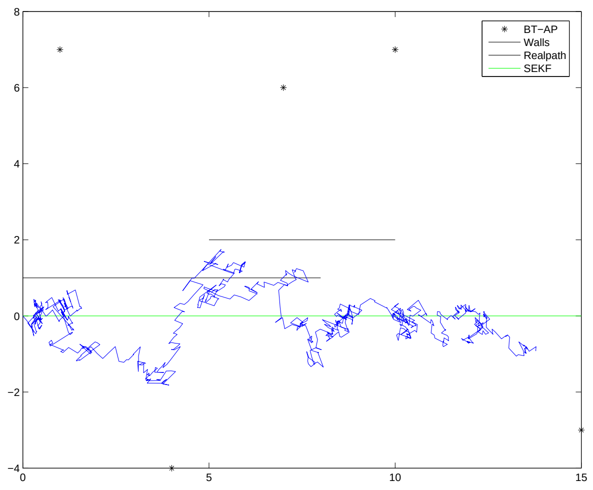


Figure 13.12: Tracking by using SLAMWiN Core in a scenario with two walls.

back to the initial location and recalculate three times. This is illustrated in Figure 13.13 that shows the order in which SLAMWiN processes the obtained measurements.

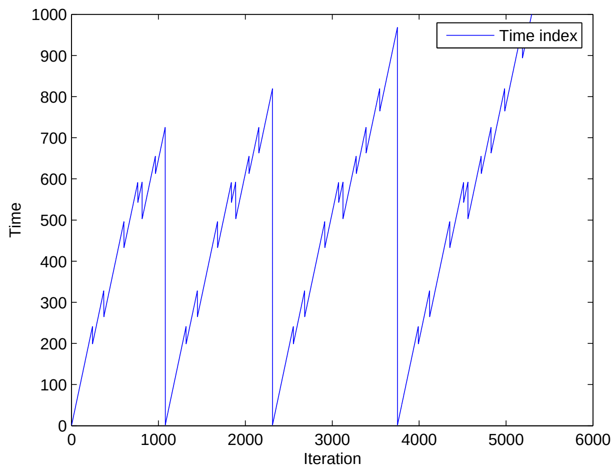


Figure 13.13: This figure illustrates how SLAMWiN Core jumps through the measurements as time progresses.

With SLAMWiN Knowledge sharing this could have been avoided if another MUs had detected the walls previous to the tracking shown in the figure above. In that case the SLAMWiN system could have estimated the correct wallcount and offset to the MU's initial location from the beginning.

The gain of the ability to estimate the correct wallcount and offset for the initial locations of MUs depends on the scenario and the path of the MU. If a MU itself would have detected a wall obstructing the LoS to its initial location early, the gain of the known wallcount and offset is limited. However, the later the MU detects a wall obstructing LoS to its initial location, the more recalculation is necessary and the more gain does the knowledge provide. This can also be seen in Figure 13.13. The MU is forced to recalculate from its initial location three times, each time for a larger part of the path resulting in a larger amount of locations that have to be recalculated.

Estimation of wall absorption

The third and last aspect of SLAMWiN Knowledge sharing that will be evaluated here is the ability to estimate the wall absorption of walls in the environment. The first implementation of SLAMWiN assumed knowledge of this absorption and that it was equal for all walls. This assumption is not needed when SLAMWiN Knowledge sharing is introduced as SLAMWiN is now able by itself to estimate the wall absorption independently for each jump. In the following it will be evaluated which window size in the jump detector should be used for estimating the wall absorption and how good the performed estimation fits the actual wall absorption.

Window size used for determining the wall absorption

As described in the SLAMWiN Knowledge sharing approach, the absorption caused by a wall shall be estimated by averaging over the size of jumps detected due to the wall. As walls can be detected by the jump detector using both the large and the small window size, this begs the question whether to average over the values found by the large, or the small window sizes.

Small window sizes are exposed to noise as they represent a jump between two values which are averaged over very few measurements. Jumps based on the small window can thus indicate a significantly larger, or smaller, change in the signal than the one actually caused by the wall. This is a problem if the absorption of a wall is estimated based only on few jumps. When more MUs detect a wall and more jumps become available, more estimates will become available that are affected by the same noise and averaging over all these estimations will provide a more accurate estimation of the actual wall absorption.

Large windows on the other side include measurements from such a large time span that the movement of MUs will have an effect on the size of the detected jump and thus also on the estimated wall absorption. The measurements are most significantly effected by MUs' movement if the movement is either away or towards the AP. The mean of the RSSI values will naturally increases or decrease as the MU moves either closer to, or further away from the AP. This influences the detected jump size, as it will be detected either too large or too small. An example of this is illustrated in Figure 13.14.

The effect of MU movement depends on the MU speed, as a higher MU speed results in a larger effect. In the light of this it should be noted that the MU movement used for generating the signal shown in Figure 13.14 was slow. The shown window size was therefore chosen relatively large in order to illustrate the effects of MU movement on the detected jump size.

Based on these considerations, SLAMWiN will be designed to estimate wall absorption based on averages over the jump sizes that are detected with the small window size.

Detailed estimation of wall absorption

As described above, the estimation of wall absorption will be based on jumps detected with a small window size. It was also described that the detection is influenced by noise that affects the estimated jumps sizes and that this is handled by estimating the wall absorption as an average over all estimations of the wall absorption performed by different MUs. The previous chapter regarding SLAMWiN Core has already evaluated the jump detector and some of the results from that evaluation can be reused here.

The evaluation of the jump detector showed that the average over an increasing number of jump size estimations approached the the actual wall absorption. E.g. if a wall would absorb 15 dB, the average size of 20+ detected jumps caused by this wall would be 15 dB ± 0.5. This is illustrated in Figure 13.15 that shows the wall absorption estimated over an increasing number of estimates. The figure shows that the average over time approaches the actual wall absorption of 15 dB.

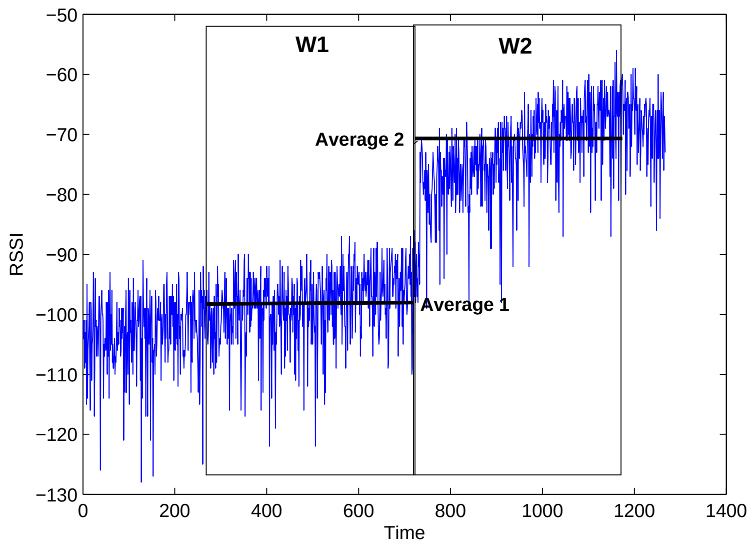


Figure 13.14: This figure illustrates how the movement of a MU effects the size of a detected jump with a large window size.

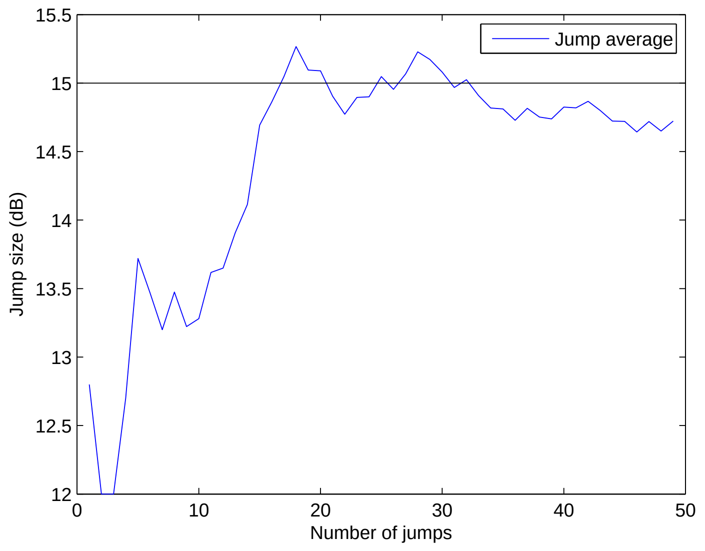


Figure 13.15: This figure illustrates the development of the wall absorption averaged over an increasing number of estimations.

The influence of a correct estimate of the wall absorption is illustrated in Figure 13.16 that shows how the location estimation of a MU is affected be different estimates of wall absorption of a wall. The figure shows different paths that are based on the same RSSI measurements, but estimated with different values for the wall absorption. The actual wall absorption in the scenario is 15 dB, while the paths are calculated for an estimated wall absorption of 5, 10, 15, 20 and 25 dB.

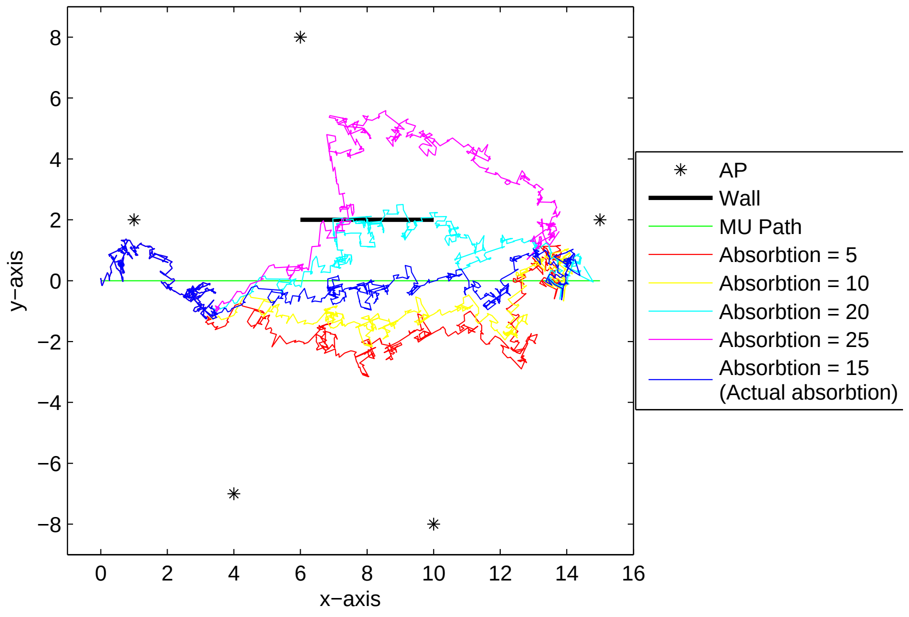


Figure 13.16: This figure illustrates the estimated path with different values for the wall absorption.

The consequences are also apparent in the following Table 13.2 where the metrics introduced in Section 11.4 are used to illustrate the average location error of each path.

Estimated wall absorption:	Track-Metric:	Path-Metric:
5	$2.518m$	$1.130m$
10	$2.315m$	$0.862m$
15	$2.073m$	$0.536m$
20	$2.330m$	$0.882m$
25	$3.089m$	$2.065m$

Table 13.2: This table illustrates the average location error caused by different wall absorption estimates.

From both the Figure 13.16 and Table 13.2 it can easily be seen that a more precise estimation of the wall absorption has a positive effect on the location estimation. A wrong offset of 5 dB worsens the localization of MUs considerable. SLAMWiN applies therefor an average over all estimates of the wall absorption in order to provide a better localization estimation.

Possible problems

The introduction of the SLAMWiN Knowledge sharing component has added new functionality to SLAMWiN but it has also introduced new problems. These are shortly discussed in the following.

Inaccuracy in placement of shadows and flags The placement of flags and shadows are determined by the position of the MUs. Because these positions are not absolutely correct, the placement of shadows and flags will suffer from this inaccuracy. When estimating the wallcount and offset based on shadows, this lag of accuracy can entail a wrong estimation. However, when the number of shadows and flags increases in the map this problem should be reduced.

Probability of detecting false flags: An indication of how many flags that on average are placed based on a false jump detection, or in the wrong location, would provide very useful information on the performance of the SLAMWiN system. It would give an overall measure of how reliable the system is and could be used in the more detailed evaluation of SLAMWiN. Unfortunately, this parameter cannot be derived other than for one specific scenario at a time. The parameter depends on one side on the jump detector and on how likely it is to detect either false positive or false negative jumps, and on the other side on at which point during the tracking of a MU that the false detection takes place.

Regarding the likeliness of the jump detector to detect either false positive or false negative jumps, this has been discussed before. It is possible to derive a probability for both types of false detections, but this will depend on both the scenario and on the configuration of the jump detector.

And even if the probability for the two types of faults could be determined, it is impossible to make a qualified guess about when these faults take place. The time when a fault takes place has a huge impact on how large consequences it has. A false jump detected early in along a path will result in that all following flags are placed at the wrong locations. A late false detection may only result in one wrongly placed flag.

Distribution of wrongly placed flags: Another characteristic of wrongly placed flags that has not been able to be evaluated thoroughly is the distribution of their location. The project group expects that some of the wrongly placed flags are not distributed randomly in the virtual map, but will have a tendency to be placed together. This expectation is based on the fact that false negative detections will occur at the same places and that these will result in paths that are equally biased by the missing detection of the same wall. This tendency has furthermore been indicated during tests with the emulator.

The problem with this is that a non-random distribution of wrongly placed flags will result in a higher probability of false flag groups. Since flag groups are trusted as being correct indicators of walls, this would results in fatal errors in the system.

This problem can be avoided if the number of false negative detections is kept low. The use of increasing required minimum number of members in flag groups can then be used to prevent false flag groups.

Wrongly placed flags due to false positive detections are expected to be more randomly distributed, as they are caused by random peaks in the noise that result in false jump detections.

Use of threshold in jump detector: By using a large threshold jumps that are smaller than the used threshold cannot be detected by the jump detector, in this case 9.8 dB. This is a problem as the previous evaluation of the consequences of wall absorption show that a undetected wall with a wall absorption of 5 dB has a considerable influence on the localization of MUs.

It is difficult to make a optimal, scenario-independent configuration of the current jump detector as the used threshold depends on the noise. The jump detector is thus not ideal for a complectly PnP capable SLAMWiN system. It does, however, prove that it is possible to detect walls that cause jumps larger than the noise in a scenario. Implementing a jump detector that is able to detect jumps due to walls that are smaller than the noise in the observed RSSI signal would solve the problem discussed here and would greatly enhance the capabilities of the SLAMWiN system.

Status of SLAMWiN

With the addition of the SLAMWiN Knowledge sharing component, the SLAMWiN system is now able to estimate both the initial position of MUs and the path loss caused by walls. This means that the localization has become considerably more PnP capable.

The SLAMWiN system is at this point still lacking the capability to estimate the position of APs in a scenario. This is the final obstacle for a fully PnP capable localization system.

Chapter 14

Estimation of AP positions

This chapter will describe the development of the AP positioning system, which covers the process of estimating the position of all APs in a scenario. This will be done by using AP to AP measurements to estimate the distance between APs in a given scenario and furthermore utilize wall information between APs from the map to make a better AP estimates. In the following the approach and design is described first, after which the implemented design will be evaluated.

The purpose of this chapter is to document the last development step of SLAMWiN that enhances the PnP capability of the system.

14.1 Estimation of AP positions approach

The goal of this project is to develop a PnP localization system for indoor environments that requires as little human interaction as possible. The SLAMWiN system should therefore not depend on obtaining information about placement of APs from the user. This means that the SLAMWiN system should be able to estimate these positions through measurements made in each specific scenario.

The approach for estimating the position of APs is to use AP to AP measurements in order to estimate the distance between APs in a scenario and based on these, guess the position of all APs. However, because walls can be located between APs, this will have an effect on the estimation of the distances between APs and thus also on the estimation of AP positions. One way to cope with this problem is to use wall information from the virtual map of the environment. This map should give information about possible walls between APs, and the AP positioning should then be able to compensate for these when calculating distances between APs. Because the position of APs may change when new information about walls becomes available, this will of cause also have an influence on the virtual map. It is therefore necessary to clear the virtual map and rebuild it when changing AP positions.

But before the virtual map can be generated in the first place, it is necessary to make an initial estimation of AP positions without using the virtual map. Furthermore it is preferable that the estimated AP positions are approximately correct as this will be a good basis for a converging map. Since walls have a major influence on the error when estimating the distances between APs, it is thus advantageous to consider possible walls when making the initial AP position estimates.

To get initial information about walls, the selected approach involves a MU. This does not initially go well together with the main goal of making the totally PnP SLAMWiN system. However, the procedure is kept very simple and fast and provides a huge help in getting good initial AP position estimates compared to the needed human effort.

A MU starts close to one AP and moves then close to the next AP. This procedure is continued until the MU has been close to all APs and stops when the MU moves close to the first AP again. Because the MU has moved close to each AP it is possible through the RSSI measurement to detect when the MU was close to a given AP during its movement through a scenario. By looking at possible detected jumps detected between being close to different APs, the difference in rises and falls can give an estimate of the wall count between APs. Possible walls are then found between any combinations of APs placed in the scenario. Because the wall thickness initially is unknown, the estimated wall absorption from the detected jumps are used. The wall absorption is then subtracted from each estimated AP to AP distance. When the virtual map for a scenario has converged it should be possible to get better wall absorption estimates which again should contribute to making a better estimation of AP positions. For each iteration the virtual map containing estimates of walls and the position of APs should converge against the true environment.

14.2 Design of estimation of AP positions

Before the initial position of APs can be found it is necessary to estimate the distance between the APs. These can be found by using RSSI measurements obtained between APs. This transformation can easily be done by using Equation 14.1 where R is RSSI measurement and $dist$ is the estimated distance.

$$dist = 10^{(\frac{R-\alpha}{10\cdot\beta})} \tag{14.1}$$

Normally the position of APs are stationary because they are fixed on e.g. walls. This could be a problem when estimating the distance due to positive and negative interference. Because the APs are stationary, this could result in that one AP experiences almost complete signal cancellation from another AP due to negative interference. The consequence of this are that even though a mean is based on several RSSI measurements between the APs, a wrong distance estimation would be achieved. One element that has a positive influence on this problem is people moving around in the scenario. This will change the reflection pattern for the wireless channel and cause the signal strength to vary. By averaging over several measurements, while people are moving around, a more reliable distance estimation can be made. To evaluate this claim a simple analysis is made. The analysis consists of the following three test cases:

1. Communication between two stationary devices with no human movement in the environment.

2. Communication between two devices with no human movement in the environment where one of the devices moves around within a area of 10 cm^2.

3. Communication between two stationary devices with human movement taking place in the environment.

All evaluation cases were carried out in a classroom similar to the room illustrated in Figure 11.6 on page 94. The two devices were placed in the middle of the room on two tables. The distance between the two devices was approximately 4 meters. RSSI measurements were only taken in one direction and the duration for each test was 10 min. The histograms for the three evaluation cases are illustrated in Figure 14.1, 14.2 and 14.3 respectively. For each histogram an exponential distribution is plotted with $\sigma = 4$. For more information about the exponential distribution see Appendix D on page 227. The direct ray is estimated to 65 dBm which is subtracted from all measurements.

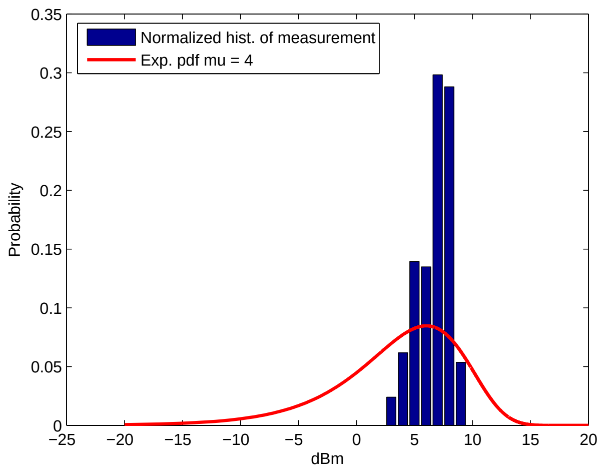


Figure 14.1: Histogram of RSSI measurements taken in an environment with no movement.

Based on these histograms is can be seen that test case 2 is fitting best to the exponential distribution which is expected since one of the devices was moving in this scenario. However, the fit is not so good as the fit made with the emulator in Appendix D. This is probably because the evaluation shown here was carried out in a bigger room with less obstacles.

When comparing test case 1 and 3 it can be seen that human movement in the room makes a difference in the received signal strength in form of larger variations, and the distribution is fitting better to the exponential distribution. The average of all measurements in test 1, 2 and 3 are -58, -61 and -59 dBm respectively. Based on this evaluation it

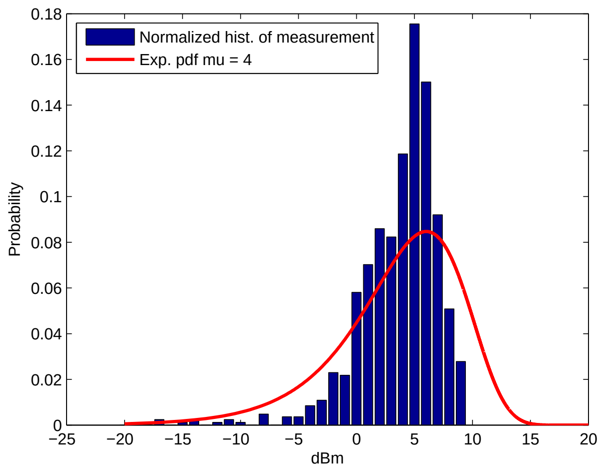


Figure 14.2: Histogram of RSSI measurements taken in a scenario when moving one of the devices within a area of 10 cm^2.

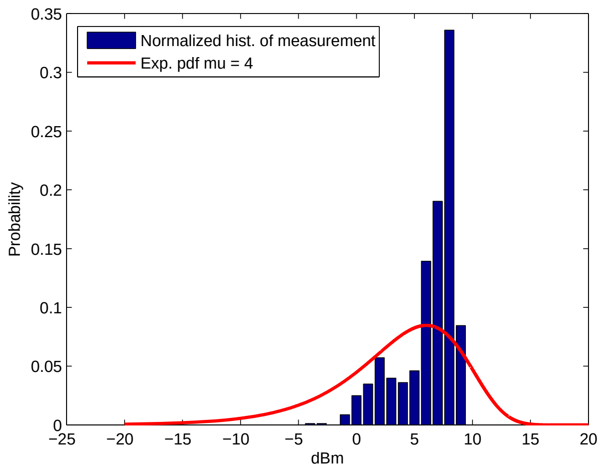


Figure 14.3: Histogram of RSSI measurements taken in a scenario where human movement takes place.

is assumed that the effect caused by multipath fading is reduced by meaning over several measurements taken in a scenario where people are moving around.

In order to emulate AP to AP measurements, measurements are generated by adding noise to the actual distance. The added noise results in that the distance estimates are uniformly distributed within ±10% of the actual distance. This represents a significant simplification of the real noise. Unfortunately, no useful approximation of the real noise could be achieved since this required real life measurements that could be used to determine both the variation due to noise and due to distance.

A series of approximately 800 real AP to AP measurements where obtained in order to estimate the variance due to noise, but in order to determine the variance due to distance it would have taken several more measurements. For each desired distance at least $50-100$ real AP to AP measurements would have been necessary and several different distances would have to be measured. This task was not executed due to the knowledge gain / needed effort ratio. It would simply have required too much work for too little extra realism of the measurements.

Estimating distances between APs

Provided that it is possible to estimated distances between APs with a reasonable accuracy, the next step is to convert these to relative positions of APs. To do this Euclidean distance geometry[AKW99] can be used. Given n AP locations $\mathbf{p_1}, \dots, \mathbf{p_n}$ in the space $\mathbb{R}^m$, the corresponding distances between APs can be expressed in an Euclidean distance matrix $Q(i,j) = [q_{ij}]$ where $[q_{ij}]$ is defined by:

$$q_{ij} = \|\mathbf{p_i} - \mathbf{p_j}\|^2, i, j = 1, \dots, n \tag{14.2}$$

One way to approximate a solution that conform to the distances, is to minimize the distance function Equation 14.3.

$$f(\mathbf{p_1}, \dots, \mathbf{p_n}) = \sum_{i=1}^{n} \sum_{j=1}^{n} \left(\|\mathbf{p_i} - \mathbf{p_j}\|^2 - S(i,j)\right)^2 \tag{14.3}$$

Where $\mathbf{p_i}$ is the i^{th} AP coordinate that has to be estimated and S is an Euclidean distance matrix for all known distances between the APs. By using this method it is not required that all distances are known, since missing distances simply will not be considered in the distance function. To minimize Equation 14.3 the function `fminunc` in Matlabs Optimization Toolbox is used. To optimize the minimization process the gradient and the hessian matrix are calculated based on the distance function, and passed to the `fminunc` function. As initial $\mathbf{p_1}, \dots, \mathbf{p_n}$ are placed randomly. Because a numerical method is used to estimate the positions of the APs, the solution could end up in a local minimum instead of a global. To increase the chance for finding a global minimum the minimization process is executed several times with different initial guesses. The combination of AP positions that generates the smallest error is then used.

Because only relative positions are found with this method, three APs are used as reference in the coordinate system. First all APs are moved so one dedicated AP is placed in $(0,0)$. Then all APs are rotated around $(0,0)$ by using the rotation matrix 14.4 where θ is the angle of the line going through a second dedicated AP and $(0,0)$. The APs are rotated so that the second AP are placed on the x-axis.

$$\mathbf{R}_\theta = \begin{bmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{bmatrix} \tag{14.4}$$

The new positions are calculated by using Equation 14.5.

$$\hat{\mathbf{p}} = \mathbf{R}_\theta \mathbf{p} \tag{14.5}$$

Lastly, all APs are reflected in the x-axis if a third dedicated AP is placed in the *3rd* or *4th* quadrant. The three dedicated APs can then also be used as references between the virtual map and the real environment.

Estimating initial AP positions

As mentioned above it is necessary to estimate the number of walls between APs in order to get a proper position estimations. By letting a MU move between all APs in a given scenario it is possible, based on jumps detected along the path, to make an estimation of the wall absorption between all APs. The wall count will then be the difference between experienced rises and falls. To determine where the MU is moving from and where to, the MU has to move close to each AP. In that way it is possible to detect when the MU is reaching a given AP, simply by using an average window on the RSSI measurements for each AP. The maximum value is then the point where the MU is closest to the AP.

Because the MU is starting and stopping at the same AP, is it possible to estimate the number of walls between two given AP in four different ways. This is illustrated in Figure 14.4 which shows when a jump has been detected for AP 1 and AP 2. The red lines marks the time when the MU is closest to the APs. Here it can be seen that in the period when the MU is moving from AP 1 to AP 2, AP 1 detects a rise and AP 2 detects a fall. Both detections indicates that one wall is placed between the two APs. The same procedure can be used in the period when the MU is moving from AP 2 and back to AP 1 again. Here AP 1 detects a fall and AP 2 detects a rise which also indicates one wall between the two APs. To find the number of walls between all other AP the same principles can be used.

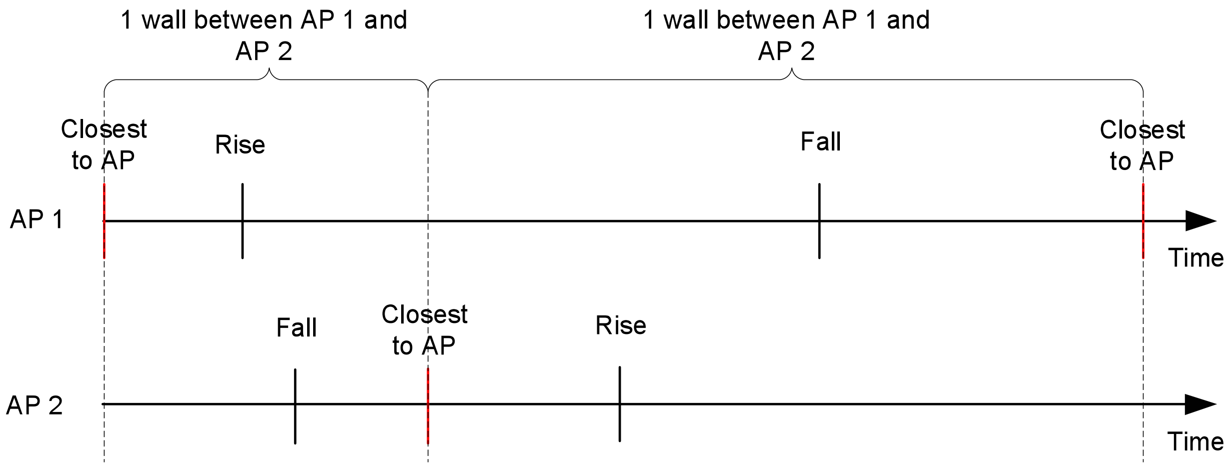


Figure 14.4: The figure shows when jump has been detected for AP 1 and AP 2. The red lines marks the time when the MU has been closest to the APs.

Having four different ways to estimate the wall count is practical, because wrong jump detections will happen and the estimated wall count between two APs can vary depending on which way it is estimated. Because the possibility for making a wrong detection within a period of time increases with the length of the period, only the shortest two paths between two APs are chosen. Thereby the number of relevant wall estimates between two APs is reduced to two. Since it is possible easily to calculate a distance error for each estimate, the distance error is calculated for both wall estimates and the one with the lowest distance error is chosen.

Because the thickness of walls is unknown it is necessary to estimate the wall absorption before the influence of the wall can be removed and a proper distance estimation can be made. It is obvious to use the detected jumps from the initial phase to estimate the walls absorptions with. However, because the wall count is determined by the difference in rises and falls, from a given AP in the time between the MU is moving from one AP to another, multiple jumps could occur. It is impossible to say which of the jumps that was a result from the wall between the two APs so an average of all the jumps will be used to estimate the wall absorption.

Even though different wall combinations are estimated and tested, there is still a possibility for a wrong detections and thereby wrong estimations of the AP positions. The last method for improving the estimations is to evaluate the distance function described above. As the distance error is calculated for each distance, it is possible to detect outliers. Based on this knowledge it is possible to run the AP position estimation again, ignoring distance estimates that have proved to provide a large error. If this is done recursively a couple of times, false distance estimates can be sorted out, resulting in a higher chance for getting a good estimation of AP positions. This approach has the limitation that it only works when the majority of distance estimates are correct.

14.3 Evaluation of estimation of AP positions

To evaluate the estimation of AP positions, different scenarios are used to test different aspects of the approach. In all tests the wall absorption is assumed known and is set to 15 dB. Estimation of AP positions with unknown wall absorption will be tested in the final evaluation. Furthermore, it is assumed that all APs are able to make measurements from all other APs.

The first test is a very simple scenario where 5 APs are placed as illustrated in Figure 14.5. The position of AP is then estimated as described earlier in this chapter and the result from this estimation is shown in Figure 14.6. The distance error is calculated to be 190.47 which is due to the noise added to the distances.

Next scenario uses the same placement of APs, but introduces two walls which are placed perpendicular to each other. This scenario is illustrated in Figure 14.7. By estimating AP positions without considering possible wall in the scenario, the best fit is illustrated in Figure 14.8. The distance error is 64982 and here it can be seen that the walls have a major influence on the estimated distances and thereby the estimated AP positions. It is thus clear that walls need to be considered if proper AP position estimates shall be made. In Figure 14.9 shows AP positions that are estimated by using information about walls in the scenario obtained from a MU moving through the scenario as described earlier in this

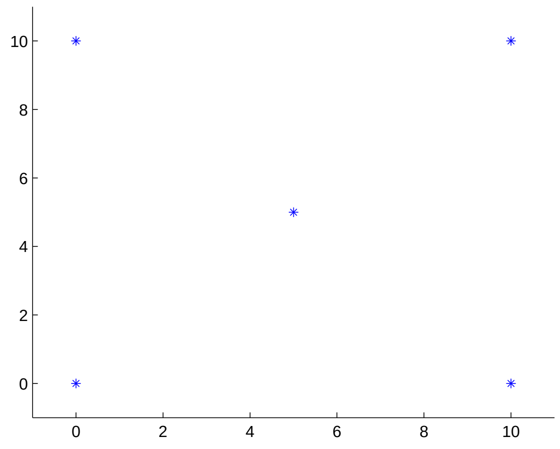


Figure 14.5: Placement of APs in a scenario without walls.

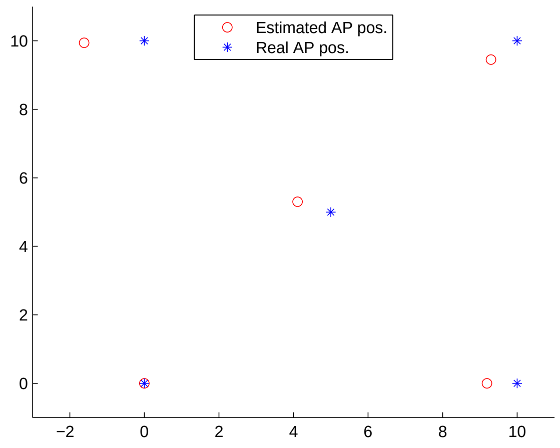


Figure 14.6: Placement of APs in a scenario without walls combined with estimated AP positions.

chapter. The distance error is 223 and the two guesses of wallcounts are listed in Table 14.1. Here it becomes clear that guess 2 is matches the actual AP positions and guess 1 is very close to. Guess 2 is chosen to be the combinations that minimizes the distance function.

Status of SLAMWiN

By having added the possibility to estimate the position of APs to the SLAMWiN system, this has now become considerably more PnP capable. The integration of this system into

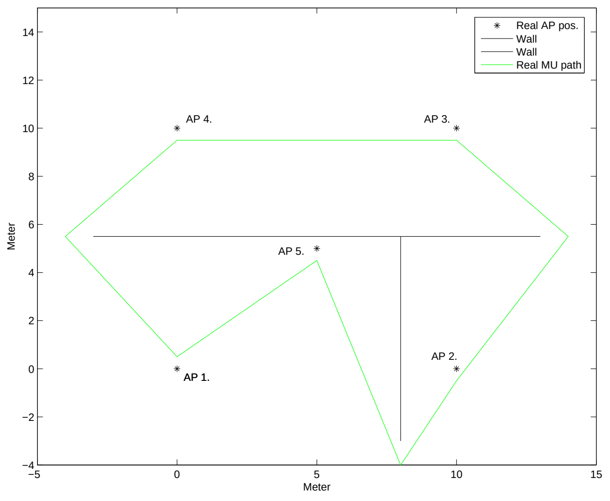


Figure 14.7: Placement of APs in a scenario with two walls.

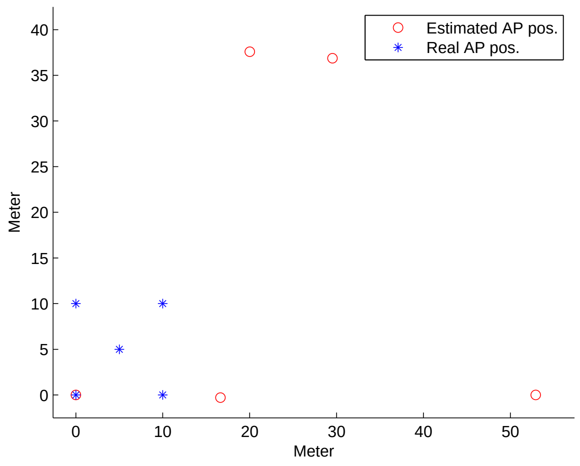


Figure 14.8: Placement of APs in a scenario with walls combined with estimated AP positions, calculated without considering possible walls in the scenario.

AP nr.	(1, 2)	(1, 3)	(1, 4)	(1, 5)	(2, 3)	(2, 4)	(2, 5)	(3, 4)	(3, 5)	(4, 5)
Guess 1	1	1	1	0	2	3	1	0	1	1
Guess 2	1	1	1	0	1	2	1	0	1	1

Table 14.1: The estimated wallcount between all combinations of APs. The AP numbers are illustrated in Figure 14.7.

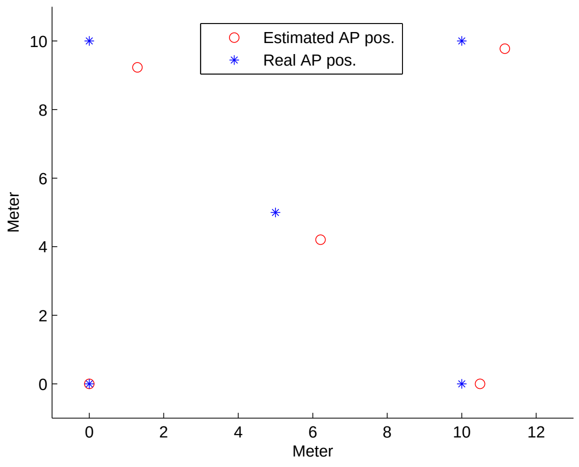


Figure 14.9: Placement of APs in a scenario with walls combined with estimated AP positions, calculated by considering possible walls in the scenario.

an existing wireless network will only require a server running the SLAMWiN software and a walk in the covered environment.

Chapter 15

Build real map

This chapter will describe the development of the Build real map component, which covers the process of merging available knowledge of AP positions and shadow information from multiple MUs, in order to make guesses of the location of walls in the environment. The map should then further be used as input to the advanced mobility models component so a higher accuracy can be gained when estimating MU locations. In the following the approach and design is described first, after which the implemented design will be evaluated.

15.1 Build real map approach and design

As mentioned above the goal with the build real map component is to generate a real map that provides information about the possible location of walls in a given environment. The approach is to divide the map into cells and then, based on flags and inverse shadows from earlier paths, to estimate whether or not a wall may be located in a given cell. The map should then be used as input to the advanced mobility models component, where walls will be fitted to the estimated likely locations of walls. By having estimated the placement of walls in the environment, the advanced mobility models should then support the filter to provide better MU location estimates.

The procedure for calculating whether or not the presents of a wall in a given cell is likely, is based on knowledge from all inverse shadows which cover the cell. While shadows indicate that the area they cover are obstructed by 'wallcount' number of walls seen from a specific AP, inverse shadows indicate that 'wallcount' number of walls are likely to be located within the inverse shadow. The inverse shadows of a MU describe the areas that lies within the triangle that has a corner in an AP and two on the MU path. The corners on the MU path are defined by the same points on the path that are used for the shadows used in SLAMWiN Knowledge sharing. This is illustrated in Figure 15.1 The approach behind generating the map is to estimate the likely location of walls by delimiting the locations where a wall cannot be.

From each AP inverse shadows will radiate out with different wall counts. By looking at all inverse shadows from each AP with a wallcount equal 0, an area can be drawn where no

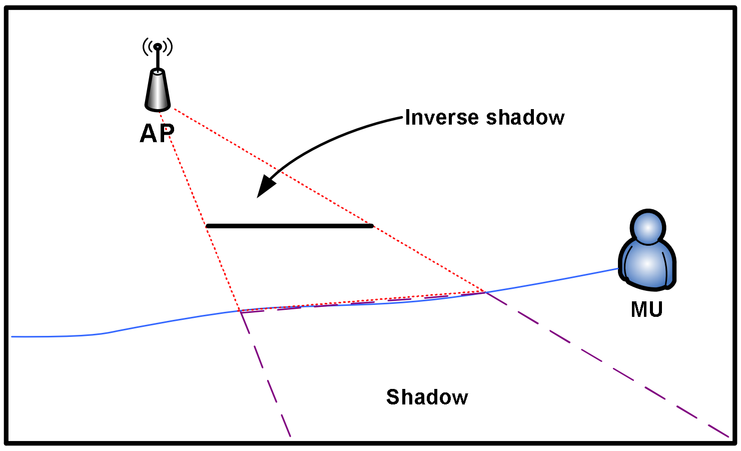


Figure 15.1: This figure illustrates the relation of shadows and inverse shadows.

walls have been detected. Because erroneous shadows will occur due to incorrect position estimations of MUs, more than a certain number of inverse shadows in a given cell has to be detected before it can be assumed free of walls. The area from each AP is then drawn with an unique color. Because these areas can overlap each other, each cell is divided into four mini-cells for illustration purposes. If multiple areas overlaps in a cell, the cells mini-cells will contain colors from each overlapping cell.

By looking at the generated map it should be possible to get a feeling of where a wall is placed. One likely placement of wall can then be at the cells that are not covered by any cell. However, this can also be due to that no MUs have been moving at that given cell. A better indication of a wall is thus, if two areas get close to each other without overlapping, or with only little overlapping. This indicates that a wall is placed between the to areas and that MUs have moved very close to the wall.

15.2 Evaluation of build real map

To evaluate the approach and implementation a test is carried out based on a scenario where two walls are placed. The scenario is illustrated in Figure 15.2. 30 paths are randomly selected with some limitations as illustrated in Figure 15.2. Based on all flags and inverse shadows created by these paths a map is created which is illustrated in Figure 15.3 where each color represent the LoS area from a AP and dark blue cells represents uncertain areas, or likely walls.

In the figure we can see that both walls are partly covered by cells indicating a likely location of walls and furthermore are placed close to the border between different colors.

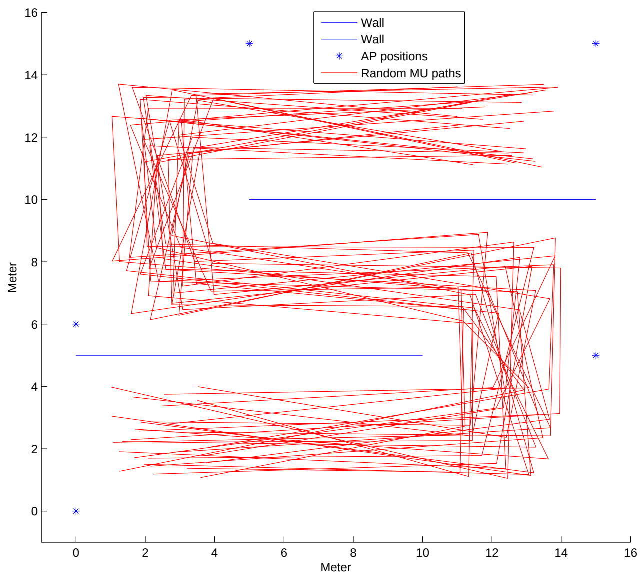


Figure 15.2: The scenario used for evaluating the Map generator.

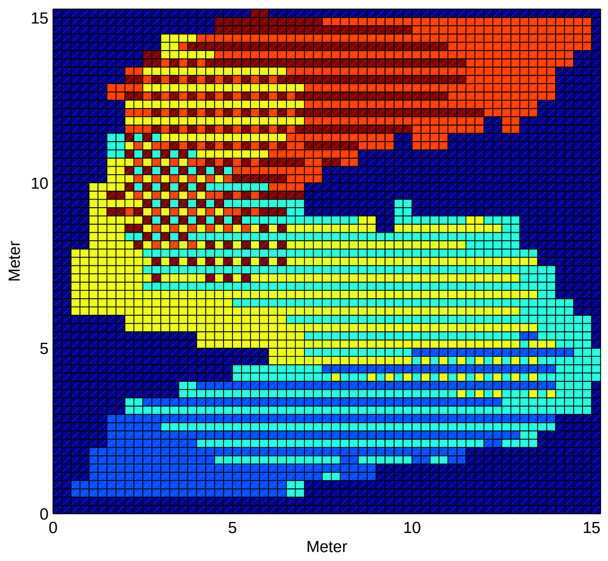


Figure 15.3: The map generated based on the scenario illustrated in Figure 15.2. Each color represent the area from each AP and dark blue cells represents uncertain area or likely walls.

Possible problems

As described above the approach utilizes LoS areas in scenarios to generate a map of the environment. As a result of this, areas in the environment which do not have LoS to any

AP will be marked as uncertain areas or likely walls, even though several MUs may have passed through them.

The marking of the areas will still be correct, since the area both represents wall(s) and uncertainty, however, the problem is that it will not be possible to reduce the uncertainty in area further so the location of possible walls in the areas can be identified more explicit. This problem can be solved by having LoS to at least one AP at any place in the scenario.

Status of SLAMWiN

By adding the map generation to the SLAMWiN approach it is now possible to get an estimate and visual representation of where walls are most likely placed in a given scenario. As mentioned above, this can be used in the advanced mobility models component to support the location estimate provided by the Kalman filter to provide better MU location estimates, but it can also be used for manual verification of the SLAMWiN system through visual inspection of the estimated map.

Chapter 16

Advanced mobility models

This chapter will discuss possible methods to enhance SLAMWiN. It will introduce a selection of methods that could combine advanced mobility models with the SLAMWiN approach in order to enhance the localization accuracy provided by the system.

The purpose of this chapter is to present some thoughts on possible enhancements to SLAMWiN in order to demonstrate future potential of the system.

16.1 SLAMWiN and advanced mobility models

The final design of the SLAMWiN architecture illustrated in Figure 10.10 on page 73 in the System design chapter states that the system should have a component which can combine the estimated MU position and map information. The purpose of this component should be to enhance the localization of MUs by using knowledge of the environment and movement of previous MUs in the scenario. Unfortunately, the time span of this project has not allowed for a thorough analysis or implementation of the component. However, since the component was included in the system architecture for a reason, a short discussion of the ideas relating to it will be presented in this chapter.

The reason for including the advanced mobility models component is that SLAMWiN over time accumulates a considerable amount of information regarding the environment and previous MUs in a scenario. Information regarding the location of walls or typical movement directions can with advantage be combined with mobility models in order to make more precise descriptions of MU movement behavior in a scenario. Generalized examples of this are presented in the mobility models appendix (Appendix B on page 218).

The following section will introduce and discuss two ideas on how to enhance SLAMWiN by introducing the advanced mobility models component.

16.2 Discussion of possible enhancements with advanced mobility models

As mentioned, SLAMWiN accumulates information regarding the environment and previous MUs in a scenario. One way to use this information is to use information regarding the environment to limit the mobility of MUs and to use information regarding the movement of previous MUs to predict the movement of future MUs. These ideas are discussed in the following.

MU movement with geographic restrictions

The first idea for the advanced mobility models component was to use knowledge of estimated locations of walls in a scenario to limit the movement of MUs. MUs should in general be restricted from moving across walls and in stead be forced to move along them. An example of this is shown in Figure 16.1 where the estimated path of a MU illustrated by the blue line temporarily crosses a wall. The red line illustrates the corrected path estimated by the advanced mobility models component. Although this enhancement of the

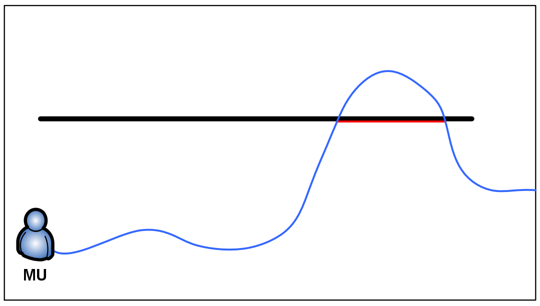


Figure 16.1: This figure illustrates how a path could be restricted by the presence of a wall.

estimated MU path may seem simple, special attention has to be paid to certain aspects of it.

Wrongly estimated walls: The first aspect of correcting paths due to walls is that the location of walls is only estimated and cannot be verified 100%. Even in cases where the presence of a wall can be estimated with a high certainty, the precise location of it will still be hard to estimate. The danger thus exists of correcting paths based on false information about walls, either because these do not exist or because they are estimated to be in the wrong location. Examples of this could be a door that is not considered in the virtual map. A MU moving through this door would never be tracked onto the other side.

The use of geographic restrictions estimated by SLAMWiN should thus only be considered if the virtual map is largely converged. In this case wall locations can be assumed to be correct and can be used as geographic restrictions.

Initial MU location: Another argument for waiting with the use of geographic restrictions until the virtual map has converged is that the initial location of MUs is estimated to be in $(0, 0)$ until likely entry points have been identified. If walls would prevent the estimated MU path to cross estimated walls, this could restrict MUs with a wrongly estimated initial position from closing in on the actual MU location as they move along the path. An example of this is shown in Figure 16.2

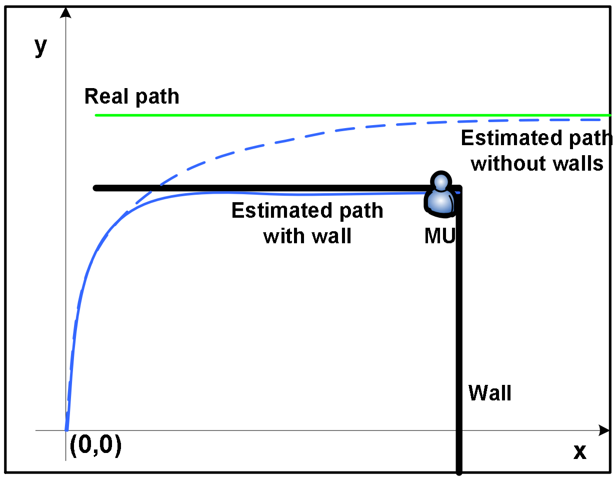


Figure 16.2: This figure illustrates a MU with a wrongly estimated initial location that cannot close in on its actual location.

Location depended MU movement

Another idea for the advanced mobility models component was to use knowledge of how previous MUs had moved in certain locations. The basic thought was that if all previous MUs had moved straight forward in a specific region of the virtual map, it would be likely that future MUs would do the same. The advanced mobility models component could use this in two ways.

Path correction based on location dependent movement direction tendencies: The advanced mobility models component could divide the virtual map into cells and for each cell use paths from previous MUs through the cell to estimate overall movement direction tendencies. These tendencies should describe the likely direction of movement that a MU, given that it is located in the cell, would move in. This description could be extended into a multi-level approach where the likely movement direction would not only depend on the location, but also on the previous location of the MU. A Markovian model could be used to describe the behaviour of a MU in each cell e.g. it could define the probability to leave the cell towards selected directions.

Filter feedback based on location depended movement direction tendencies: A more advanced way to use knowledge of how previous MUs have moved in certain regions of the virtual map would be to create a feedback loop to the filter that would modify the mobility model used in the filter. An example of this could be to change the SEKF so that it does not choose filter based on the smallest error, but based on which filter describes the most likely movement direction.

Such a feedback would have to be investigated thoroughly, as it would change the fundamental characteristics of the filter. The filter would become biased in the way that it would have a tendency towards estimating the paths of new MUs similar to the paths of previous MUs if these have the same entry point in the scenario. This can both be an advantage if the movement of MUs is in fact similar, but it can be a equally large disadvantage for MUs with a different movement pattern. It should furthermore be considered that the feedback can be self-perpetuating.

Status of SLAMWiN

Since the advanced mobility models component is not implemented, the SLAMWiN system has not gained any new functionality or performance enhancement. However, if the component would be developed and implemented this could potentially increase the localization accuracy of SLAMWiN.

Part III

Project evaluation

Chapter 17

Final SLAMWiN evaluation

This chapter will document the final evaluation of the SLAMWiN system. After a short introduction, the chapter starts out with a description of the scenarios used for the SLAMWiN evaluation. Then the evaluation of the main aspects of SLAMWiN, the PnP capability and the location accuracy, is documented. The chapter closes with an evaluation of other relevant aspects of SLAMWiN.

The purpose of this chapter is to perform an integration test of the SLAMWiN system through a comprehensive evaluation of all its features.

17.1 Introduction to final SLAMWiN evaluation

Each of the chapters in the analysis part of this report has demonstrated a evaluation of the SLAMWiN component treated in it. Through this, each component was tested in a situation where it was isolated from the rest of the SLAMWiN system. The final SLAMWiN evaluation will therefore perform a detailed evaluation of the complete SLAMWiN system where all component will be used together. Through this an integration test of the SLAMWiN components will be performed.

The overall structure of this chapter will be:

- Description of the used evaluation scenario
- Evaluation of SLAMWiN based on knowledge levels
- Evaluation of other relevant aspects of SLAMWiN

The evaluation will be based on a scenario that corresponds to a imaginary real life scenario that the SLAMWiN system has to be applied in. The goal is to test the interaction of the SLAMWiN components and to find the performance limitations of the system.

The evaluation will focus on the two most important aspects of the SLAMWiN system: PnP capability and provided location accuracy. The approach used for evaluating these aspects has been introduced in the problem domain, see Chapter 8 on page 45, and is based on a stepwise approach that explores the capabilities of SLAMWiN through increasing complexity of the evaluation. For this purpose, knowledge levels where defined for knowledge regarding walls and APs in the evaluation scenario. To shortly summarize the levels they are here shown again:

- Knowledge levels regarding walls in the evaluation scenario starting with the lowest level first:

  No knowledge: At this knowledge level any combination of walls within the limits of the project delimitation is possible.

  All walls are either $\|$ or $\perp$: At this level all walls will either be parallel or orthogonal. This knowledge is reflected in many real life scenarios. It provides better conditions for estimating the position of walls by setting rules for their relationship to each other.

  All walls are equally thick: At this level all walls are equally thick. The conditions makes it easier to generalize knowledge regarding walls.

  Known thickness of walls: At this level the thickness of walls in the scenario are known. This knowledge significantly simplifies the scenarios providing better conditions for estimating the number and position of walls.

- Knowledge levels regarding APs in the evaluation scenario starting with the lowest level first:

  No knowledge: At this level no knowledge regarding the position of APs are available.

  Two AP on the same wall: At this level two known APs will have to be placed on the same wall. This will in combination with the scenario restriction regarding $\|$ or $\perp$ walls provide favorable conditions for estimating the position of walls.

  Known number of walls between APs: At this level the number of walls between any pair of APs are known. This sets some restrictions to the PnP capability of the system, but is expected to provide good conditions for estimating the position of APs.

  Known AP to AP distance: At this level the exact distance between all AP is known. This does not go well together with the project goal of developing a PnP system. However, it is included to ensure a basic start-scenario that will provide results to which results from the more difficult scenario can be compared. It should be noted that knowing the AP to AP distances corresponds to knowing the location of all APs since this knowledge can be directly derived.

One remark should be made regarding the knowledge levels. The knowledge levels formulated in the problem domain assumed a full implementation of the proposed SLAMWiN system. However, the advanced mobility models component has not been implemented.

Since two of the knowledge levels where formulated with this specific component in mind, these will not be evaluated. The knowledge levels in question are the second knowledge regarding walls (*All walls are either* $\parallel$ *or* $\perp$) and the second knowledge level regarding APs (*Two AP on the same wall*). The knowledge levels should have been used in combination with advanced mobility models in order to make qualified guesses of how users would move in relation to walls.

On top of the knowledge levels regarding walls and APs, another aspect that provides better PnP capability will be evaluated. This is the ability of SLAMWiN to estimate the initial position of the MU. In order to limit the number of required evaluation cases, the different knowledge levels will only be evaluated with a known initial MU position. The effect of a unknown initial MU location on the localization accuracy will then be evaluated for the best knowledge level combination in a additional evaluation case.

Beyond the main aspects of the SLAMWiN system, PnP capability and provided location accuracy, the evaluation will also comment on other relevant aspects of localization which are listed below. These aspects will also have a considerable impact on how applicable the SLAMWiN system is, if it is used for localization of MUs.

- Real time tracking capability
- Choice of wireless technology

Before the evaluation is performed, the following section will described the evaluation scenario that will be used.

17.2 Evaluation scenario

As described above, the final evaluation of SLAMWiN will be based on a imaginary real life scenario. This means that the chosen scenario will reflect characteristics of real life environments and MUs and that the evaluation will take its starting point in actions that are necessary to set up a SLAMWiN system in a real life scenario. These actions are:

1. Obtaining initial MU measurements used to calculate the AP positions.
2. Estimation of number of walls between APs and make estimations of their wall absorptions.
3. Estimation of initial AP positions.
4. Performing localization of MUs in the initial phase where no previous MUs have gained any knowledge regarding the environment.
5. Re-estimation of AP positions.
6. Performing localization of MUs where MUs can use knowledge gained by previous MUs.

7. Drawing virtual map of the environment and estimating the position of walls based on the information the virtual map holds.

The blueprint of the chosen scenario is illustrated in Figure 17.1. This blueprint was chosen since it represents a good compromise between complexity and simplicity. Although the scenario's number of rooms is relatively low, problems such as jumps close by another can be evaluated. The following will introduce the central scenario parameters. The parameters can overall be divided into two groups: Environmental characteristics and MU movement patterns.

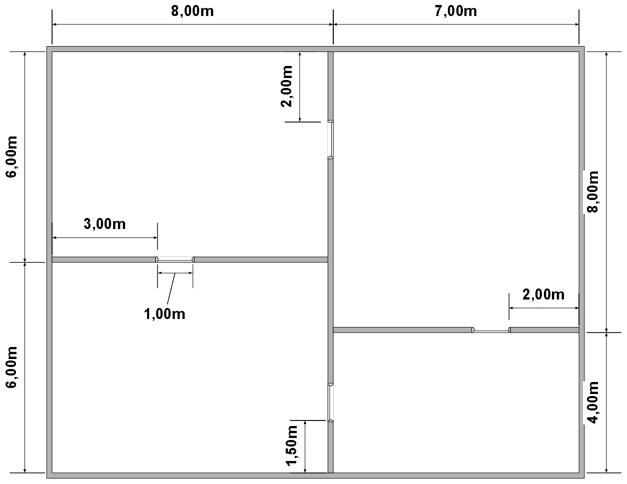


Figure 17.1: The figure illustrates the scenario used for the final SLAMWiN evaluation.

Environment characteristics

The first group of scenario parameters are the environmental characteristics. These describes three different aspects of the scenario environment: The wireless channel, walls and APs.

Wireless channel

The evaluation of each of the SLAMWiN components has been based on either real life measurements, or measurements generated by the emulator described in Appendix D on page 227 using an approximation of the real wireless channel. They thus have been evaluated based on measurements with a considerable level of noise.

However, the scenario used for the final evaluation is considerably more complex than the previously investigated scenarios. Preliminary test showed that the jump detector could not handle the final evaluation scenario to full satisfaction. On average every 50^{th} jump was detected falsely, either due to false negative or false positive jump detections. Since the path used for evaluation contained approximately 25 jumps, this resulted in every 2^{nd} path being estimated wrong. Even though this does provide some insight into the capabilities of the jump detector, it is hard to evaluate the SLAMWiN approach based on such results. In order to be able to evaluate the SLAMWiN approach the variance of the noise was reduced by 20%.

The wireless channel used in the final evaluation is defined as:

- The direct ray path loss with additive noise described with an exponential function with $\mu = 4$ in the power domain. For further information see Appendix D on page 227. As described above the noise is reduced by 20%.

Walls

The scenario chosen for the final SLAMWiN evaluation has three inner walls as shown in Figure 17.1. Due to the simplification in the emulator about all walls has to have the same absorptions in a scenario, all walls are set to have an absorption on 15 dB which corresponds to a concrete wall.

Between each of the rooms in the scenario there is a 1 meter broad door. These will in a real life scenario have a considerably lower absorption than walls. This means that a MU moving across a room may temporarily get a better signal from an AP in the neighboring room if the MU moves past a door. Such a movement would cause two jumps in the signal shortly after each other. The evaluation of the jump detector has already shown that this may cause troubles with the correct detection of jumps (see the Map granularity subsection of the SLAMWiN Core evaluation 12.3 on page 121). This problem is further amplified by the previously selected assumption that MUs never leave the range of APs.

In order to avoid problems with jumps close by another, doors in the evaluation scenario will have the same absorption as walls.

- Doors have an absorption of 15 dB.

APs

The last environmental characteristic describes the number and placement of APs. This is an important parameter as the distance to APs and the number of AP that are able to cover a MU have a large effect on the location accuracy. Furthermore, has the number of APs also a effect on how good the initial AP position estimate can be. The APs in the evaluation scenario are placed so that they provide a good coverage of the environment.

The chosen AP setup is defined in the Figure 17.2:

MU movement patterns

The second group of scenario parameters concern the MU movement patterns. These describe how MUs moves during the initial setup and in the ongoing localization in a scenario.

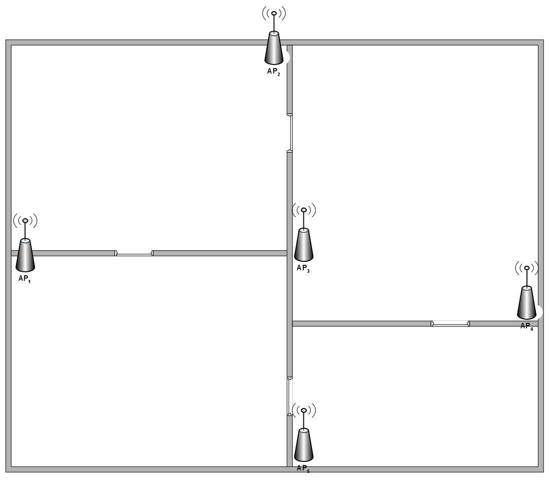


Figure 17.2: This figure illustrates the evaluation scenario with five APs.

Initial MU movement

The initial MU movement used for the AP position estimation is defined to start at one AP and go from there to all APs one by one ending up by the AP where the MU started. The natural movement will be a shortest path that leads past every AP. The initial movement that will be used for the AP setup is shown in Figure 17.3 and the MU will move with a speed of $0.03m/s$.

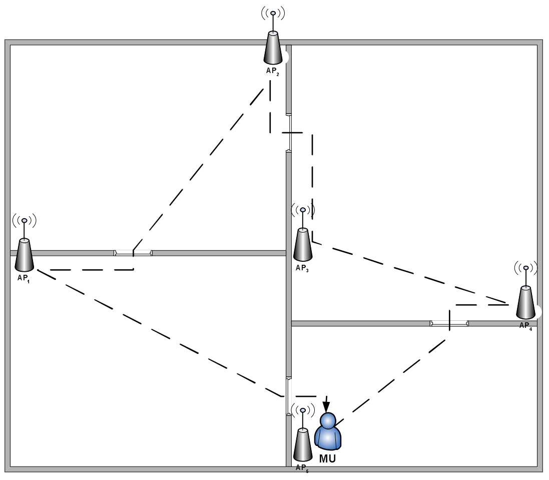


Figure 17.3: The figure illustrates the required initial MU movement for the evaluation scenario with five APs.

General MU movement

In contrast to the movement needed for the initial estimation of AP positions, the movement of MUs in general is expected to vary considerably more. However, as the scenario represents a museum, some restrictions are put onto the MU movement.

- MUs will enter the scenario at approximately the same location e.g. a front desk. An 'entry area' of $1 \cdot 2$ meter is defined.
- MUs will move in the same direction through the rooms. This reflects a 'recommended tour' by the museum.
- MUs will at no point during their stay in the scenario be stationary. This may not be very realistic, but is a necessary restriction due to the Kalman filters bad tracking of stationary MUs.

Considering these restrictions, a distributed MU movement pattern is defined that allows MUs to move over the whole scenario. A real life example of this could be single visitors that moves freely around in the museum watching selected items. The distributed MU movement is based on a Random Waypoint Model (RWM) that uses $T_{pause} = 0$ and $V_{min} = V_{max}$ and is modified so that it can be controlled by destination areas. The destination areas are subsets of the complete scenario area in which the MU chooses its next destination towards which it will move until it reaches it. The MU chooses a predefined number of destinations from each area and moves through them in a predefined order. In the areas, the destinations are randomly selected. Both the areas and the order of the areas are illustrated in Figure 17.4. Movement will start in *Area* 0 and end in *Area* 13, which is the same area. The speed of the MU will be $0.03 m/s$.

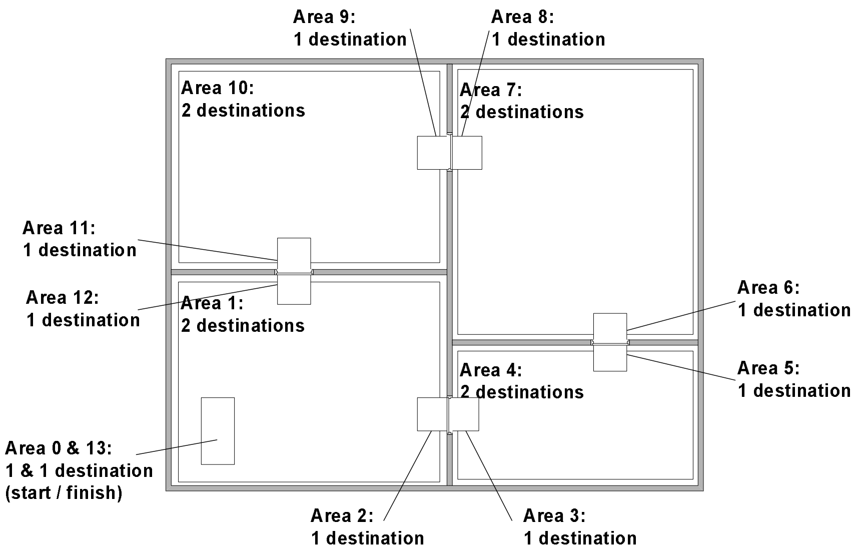


Figure 17.4: This figure illustrates the destination areas for the distributed MU movement.

17.3 Evaluation of PnP capability and localization accuracy

This section will perform the final evaluation of the SLAMWiN approach based on the described knowledge levels and thereafter for the unknown initial location. Each knowledge level combination is documented in its own subsection and the following will evaluate the different combinations of knowledge levels according to the policy that was introduced in the problem domain. The policy is shortly summed up:

1. Start with high level of knowledge regarding walls and position of APs. Evaluate the achieved accuracy.
2. Iteratively decrease level of wall knowledge to the next lower level and evaluate the achieved accuracy. Stop when minimum required accuracy cannot be reached or lowest level has been evaluated.
3. Set level of wall knowledge to the highest level and decrease the level of AP knowledge to next lower level. Evaluate achieved localization accuracy.
4. Repeat step 2 with the new level of knowledge regarding APs.
5. Repeat step 3 and 4 until either lowest level for both wall and AP knowledge is reached, or a combination with the highest level of AP knowledge fails to provide the desired minimum localization accuracy.

It should be noted that the initial evaluation case revealed that the jump detection based on the small window with a window size of 5 did not perform as expected. It was not able to handle the noise and provided too many wrong jump detections. The small window was therefore set to the same size as the large window in order to utilize knowledge from possible groupflags in form of better estimated wall absorptions. By changing the small window size this has a negative influence on the real time capabilities, but this is not considered a large drawback. Since a better jump detector already has been proven necessary, this should solve this problem.

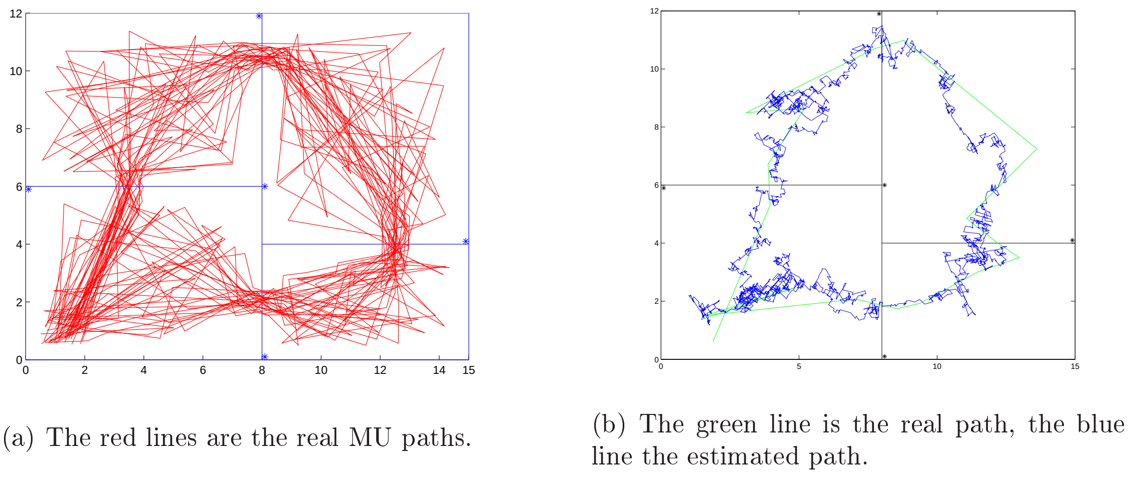
Evaluation of the different knowledge level combinations

This subsection will list the evaluation of each knowledge level combination. The combinations are listed as evaluation cases for easier identification.

Case 1: Localization with known AP to AP distance and known wall absorption

The first knowledge level combination evaluates SLAMWiN in a situation where a high level of knowledge is available regarding both APs and walls in the scenario. This knowledge level combination evaluates SLAMWiNs ability to provide localization of MUs under optimal conditions where it is not necessary to estimate neither the position of APs, nor the location and absorption of walls. 30 MUs are simulated moving through the scenario by following the described movement pattern.

Figure 17.5 illustrates the actual MU paths used for the evaluation in (*a*) and one example of an estimated path in (*b*). The properties of the localization error achieved by


(a) The red lines are the real MU paths.

(b) The green line is the real path, the blue line the estimated path.

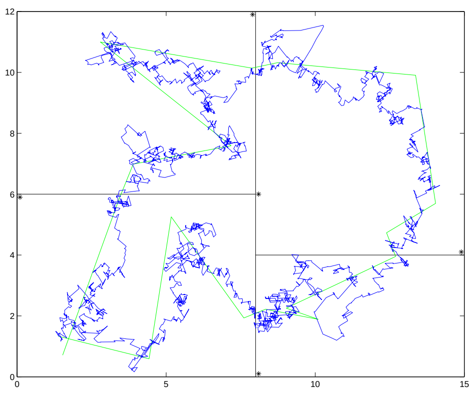
Figure 17.5: This figure illustrates the actual MU paths (*a*) and an example from the evaluation with the highest knowledge level combination (*b*).

SLAMWiN have been investigated with respect to mean error, worst and best case and the standard deviation. Furthermore, the paths have been divided into three groups depending on how satisfactory the location error has been. The error threshold values used in the groups have been found by comparing the calculated error with the visual impression of each path. Through this it has been found that paths with an Track-Metric error of up to 2.25 meter still follows the real path and that this error mainly is due to the estimated location is lacking behind the real position. This problem has already been discussed in the filter chapter. Paths with an error between 2.25 and 2.75 meter were only observed temporarily to estimate wrong locations, while paths with an error larger than 2.75 meter in all cases where estimated wrongly along almost the complete path. The groups are shortly summarized in the following:

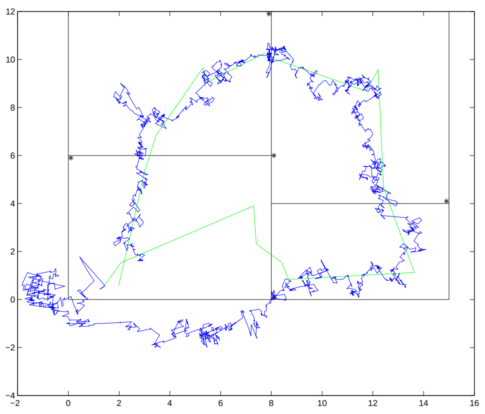
Good estimations: These are paths with a Track-Metric error < 2.25. The group describes paths that have a satisfactory estimation along the complete path. An example of this is shown in Figure 17.6 (*a*).

Reasonable estimations: These are paths with a Track-Metric error of > 2.25 and < 2.75. The group describes paths that may be estimated wrongly along some parts of the path, but which are mostly satisfactory estimated. An example of this is shown in Figure 17.6 (*b*).

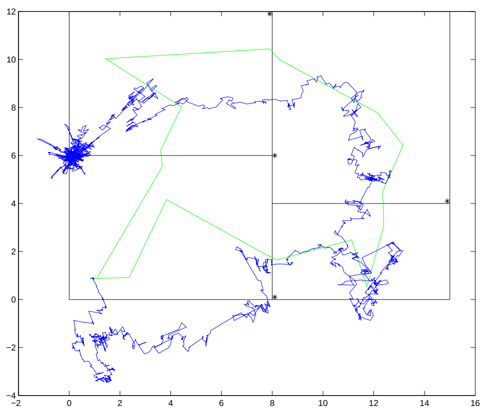
Bad estimations: These are paths with a Track-Metric error > 2.75. The group describes paths that are estimated wrongly at large parts of the path. An example of this is shown in Figure 17.6 (c).



(a) A good path estimation.



(b) A reasonable path estimation.



(c) A bad path estimation.

Figure 17.6: This figure illustrates three example paths. One from each group of estimation satisfaction.

The general properties of the localization error achieved with the highest knowledge level combination are described in Table 17.1 on the following page. It should be noted that the values for the Path-Metric error have not been included since they cannot be trusted. The algorithm that calculates the Path-Metric is not always able to handle paths that cross themselves, which is the case for several of the randomly generated paths used in this evaluation. Thus only the Track-Metric error is stated.

Property:	Average	Best case	Worst case	Std. deviation
Track-Metric error:	2.09 m	1.82 m	3.66 m	0.39 m

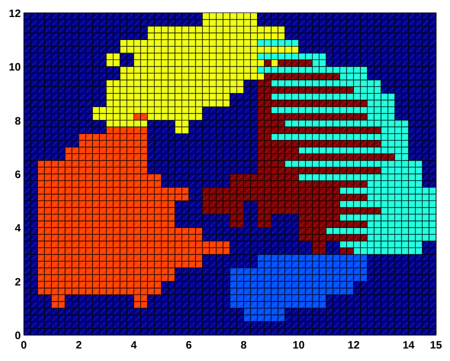
Table 17.1: This table shows the properties of the localization error achieved with the highest knowledge level combination.

For the first knowledge level combination the distribution of the paths over the groups is as shown in Table 17.2. It should be noted that 21 of the paths in the 'Good estimation' group even had a Track-Metric error below 2 meters and that the average Track-Metric error of the 'Good estimation' group is 1.98.

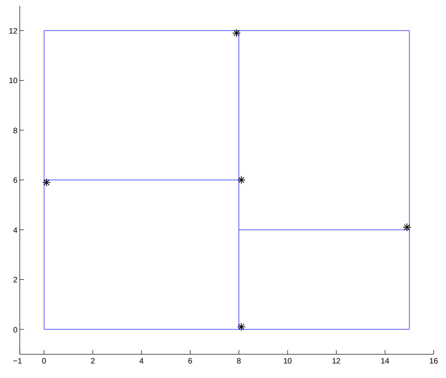
Good estimation	Reasonable estimation	Bad estimation
27 paths	1 paths	2 paths

Table 17.2: This table shows the distribution of the estimated paths over the three groups describing the degree of satisfaction with the achieved path estimation.

Based on the estimated paths, the build real map component was able to estimate the map of the environment shown in Figure 17.7 (*a*). As a comparison, Figure 17.7 (*b*) shows the evaluation scenario.



(a) Estimated map: Each color defines the LoS area of an AP.



(b) Real environment of the evaluation scenario.

Figure 17.7: This figure illustrates the estimated map of the environment.

Case 2: Localization with known AP to AP distance and equal wall absorption of all walls

This knowledge level combination evaluates SLAMWiNs ability to provide localization of MUs when the wall absorption is unknown and has to be estimated. However, the assumption is made that all walls have equal absorption. This has the advantage that the wall absorption can be estimated fast as it can be calculated as an average over all jumps detected by any MU in the scenario. The position of APs are still assumed known and also in this evaluation will 30 MUs be simulated moving through the scenario with the previously described movement pattern. In order to be able to compare the evaluation of the different knowledge level combinations the exact same paths and noise is used.

The general properties of the localization error achieved with no knowledge regarding the wall absorption, but with the assumption that all walls are equally thick, are described in Table 17.3. Again, the values for the Path-Metric error have not been included since they cannot be trusted.

Property:	Average	Best case	Worst case	Std. deviation
Track-Metric error:	2.12 m	1.80 m	3.80 m	0.41 m

Table 17.3: This table shows the properties of the localization error achieved without knowledge regarding wall absorption, but with the assumption that all walls are equally thick.

For the second knowledge level combination the distribution of the paths over the groups is as shown in Table 17.4. This time 17 of the paths in the 'Good estimation' group had a Track-Metric error below 2 meters and the average Track-Metric error of the 'Good estimation' group is 1.97.

Good estimation	Reasonable estimation	Bad estimation
26 paths	1 paths	3 paths

Table 17.4: This table shows the distribution of the estimated paths over the three groups describing the degree of satisfaction with the achieved path estimation.

As described, the wall absorptions are unknown in this knowledge level combination and is estimated as an average of all detected jump. Figure 17.8 shows how the estimated wall absorption develops over time and how it approaches the actual wall absorption of 15 dB.

Based on the estimated paths, the build real map component was able to estimate the map of the environment shown in Figure 17.9 (*a*). Again Figure 17.9 (*b*) shows the evaluation scenario for comparison.

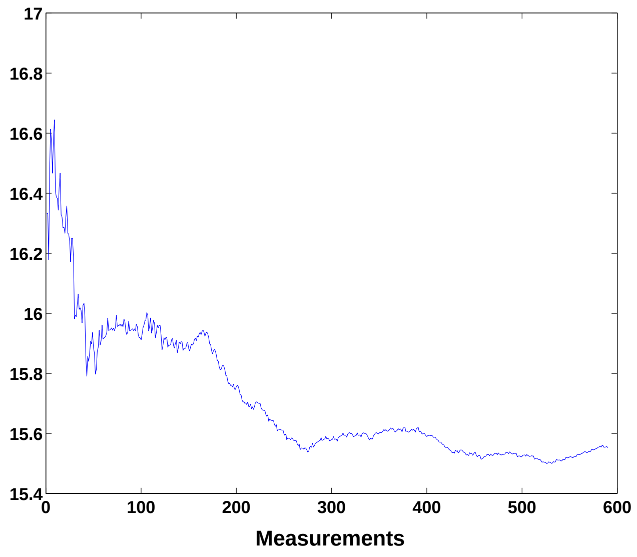
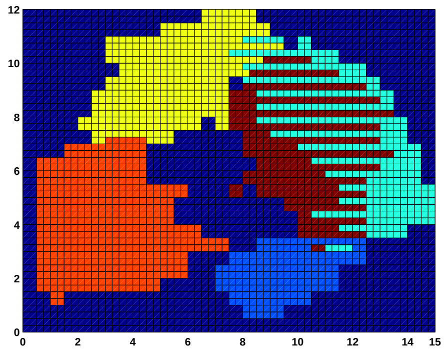
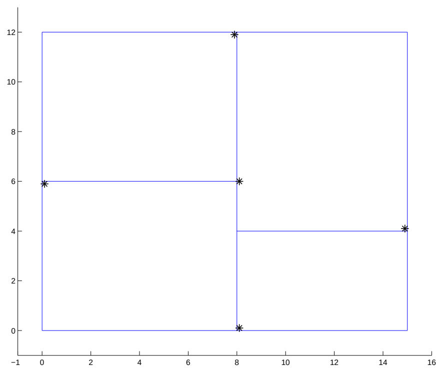


Figure 17.8: This figure illustrates the development over time of the estimated wall absorption.



(a) Estimated map: Each color defines the LoS area of an AP.



(b) Real environment of the evaluation scenario.

Figure 17.9: This figure illustrates the estimated map of the environment.

Case 3: Localization with known AP to AP distance and no knowledge regarding walls

This knowledge level combination again evaluates SLAMWiNs ability to provide localization of MUs when the wall absorption is unknown and has to be estimated. In contrast to the previous case, this combination does not assume that all walls are equally thick and the estimation of wall absorptions can thus not be estimated as a average over all jumps.

Instead groupflags are used to calculate an average over all jumps that have taken place close to each other. This requires that the groupflags have to be set first. Therefore a group of 30 MUs where simulated moving through the scenario in order to set the group flags. Then a second group of 30 MUs where simulated moving through the scenario which used the knowledge provided by the group flags. In order to evaluate how many MUs are necessary in order to get a good estimate of the wall absorption at the groupflags, a third group of 30 MUs where simulated moving through the scenario and it was investigated if these would be located with a smaller location error.

The general properties of the localization error for each of the three runs with 30 MUs are described in Table 17.5. The table shows that the localization error decreases when the group flags are used from the second MU group on. However, there seams not to be any significant gain from the second to the third group. This indicates that groupflags based on 30 flags already provide a good basis for estimating the wall absorption of a wall.

Property:	Average	Best case	Worst case	Std. deviation
Track-Metric error first group:	2.82 m	2.04 m	4.08 m	0.58 m
Track-Metric error second group:	2.59 m	1.93 m	3.84 m	0.64 m
Track-Metric error third group:	2.61 m	2.01 m	3.86 m	0.59 m

Table 17.5: This table shows the properties of the localization error for each of the three MU groups.

The groupflags used here are illustrated in Figure 17.10. The blue lines indicate the direction and distance from each AP to the groupflags associated to them. All flags that have been considered in the flaggroups thus lie in a 5 m wide area in extention of this line (Se Figure 13.8 in Section 13.2 on page 135 for explanation of groupflags). The red circles indicate areas in which several lines cross each other. These are thus likely location where MUs cross the walls e.g. doors. It should be noted that not all AP have detected four groupflags. E.g. has the leftmost AP only detected three groupflags. Figure 17.11 furthermore illustrates two examples of the flags that form the basis the groupflags. Figure 17.11 (*a*) shows the flags set by the first group of MUs that move through the scenario. Figure 17.11 (*b*) shows the flags set by the second MU group. It can here be seen that the flags are estimated closer to together in the second example. This is because the paths in this example are estimated based on knowledge from the 30 first MUs.

The distribution of the paths for each of the three MU groups are shown in Table 17.6. Also here there is no significant gain from the second to the third group.

Based on the estimated paths from each of the three groups, the build real map component was able to estimate the map of the environment. Figure 17.12 (*a*) shows the estimate based on the first group while Figure 17.12 (*b*) shows the estimate based on the third group. Figure 17.13 shows the evaluation scenario for comparison.

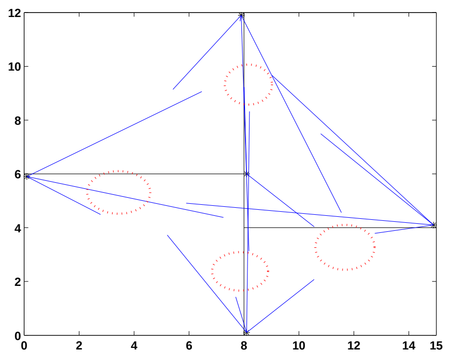
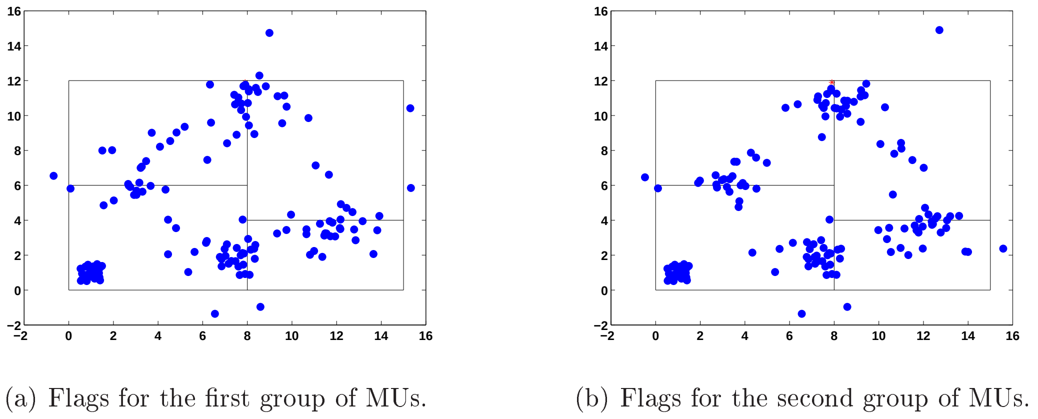


Figure 17.10: This figure shows the estimated groupflags. The blue lines indicate the direction and distance from each AP to the groupflags associated to them while the red circles indicate likely location of doors.



(a) Flags for the first group of MUs. (b) Flags for the second group of MUs.

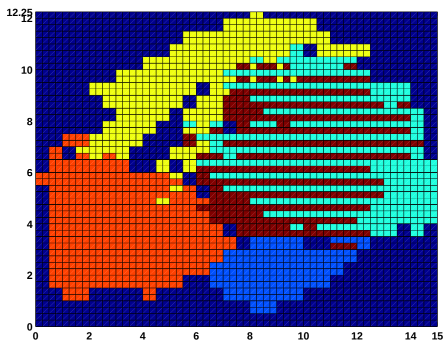
Figure 17.11: This figure illustrates two examples of flags that are used to calculate groupflags.

	Good estimation	Reasonable estimation	Bad estimation
First group:	5 paths	10 paths	15 paths
Second group:	15 paths	4 paths	11 paths
Third group:	13 paths	7 paths	10 paths

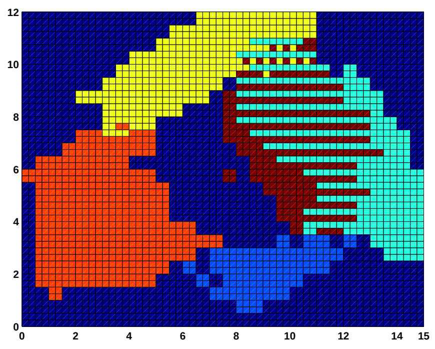
Table 17.6: This table shows the distribution of estimated paths over the three groups describing the degree of satisfaction with the achieved path estimation for each of the three MU groups.

Case 4: Localization with known number of walls between APs and known wall absorption

This knowledge level combination is the first that evaluates SLAMWiN without having knowledge about the exact position of APs in the scenario. With full knowledge regarding



(a) Estimated map after 30 MUs.



(b) Estimated map after 90 MUs.

Figure 17.12: This figure illustrates the estimated map of the environment after 30 and 90 MUs.

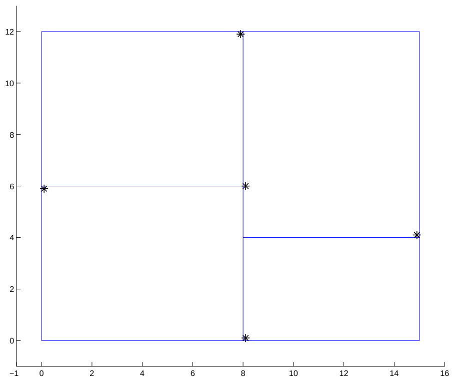


Figure 17.13: This figure illustrates the scenario environment.

walls, this knowledge level combination will evaluate SLAMWiNs ability to estimate the position of APs based on knowledge about the number of walls between any pair of APs and the absorption of the walls. As in the other evaluations the results will be based on measurements from 30 MUs moving through the scenario.

Figure 17.14 shows the location of the actual APs and the location of the estimated APs. The estimated APs are placed according to the method described in the problem domain that places the estimated AP so the combined distance error of any pair of real AP, estimated AP is minimized. Since both the number of walls and the wall absorptions are known the distance between the APs can be computed without the need for a MU making an initial trip past each AP.

The average distance error between the estimated and the real APs is 0.16 meter and no AP are estimated further away from its real position than 1 meter.

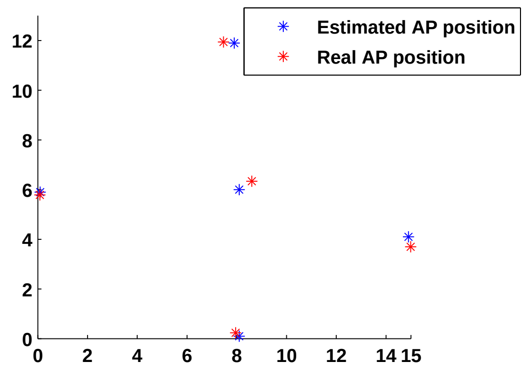


Figure 17.14: This figure illustrates the position of the real and estimated APs.

The general properties of the localization error for the 30 MUs are described in Table 17.7. The Table shows that the error in the estimation of the AP positions has almost no effect on the achieved localization accuracy. The average Track-Metric error achieved in this evaluation is only 0.04 meters worse than in the first knowledge level combination.

Property:	Average	Best case	Worst case	Std. deviation
Track-Metric error:	2.13 m	1.78 m	3.66 m	0.38 m

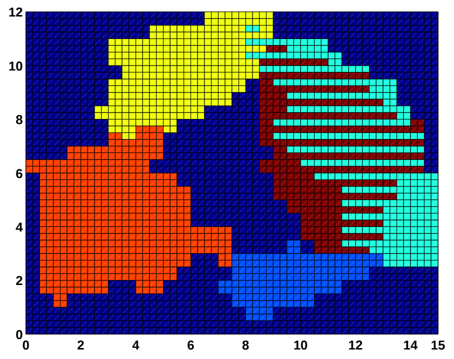
Table 17.7: This table shows the properties of the localization error achieved without having knowledge about the exact position of APs, but with the knowledge of the estimated distances between AP and absorption of walls.

The distribution of the paths over the location error groups are as shown in Table 17.8. This time only 12 of the paths in the 'Good estimation' group had a Track-Metric error below 2 meters and the average Track-Metric error of the 'Good estimation' group is increased slightly to 1.99 meter. This shows that even though fewer paths have a smaller Track-Metric error than 2 meter, the accuracy provided by the 'god' paths does not suffer considerably.

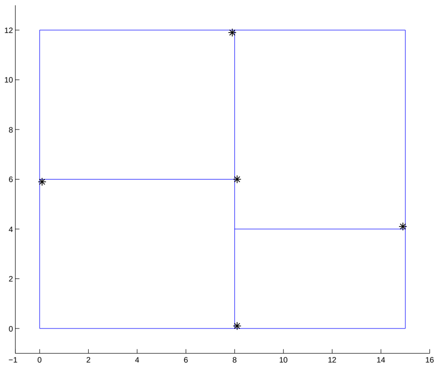
Good estimation	Reasonable estimation	Bad estimation
25 paths	3 paths	2 paths

Table 17.8: This table shows the distribution of estimated paths over the three groups describing the degree of satisfaction with the achieved path estimation.

Based on the estimated paths, the build real map component again estimated the map of the environment shown in Figure 17.15 (a). Like the location accuracy, the map estimate of the environment does not suffer from the unknown AP position. Figure 17.15 (*b*) shows the evaluation scenario for comparison.



(a) Estimated map: Each color defines the LoS area of an AP.



(b) Real environment of the evaluation scenario.

Figure 17.15: This figure illustrates the estimated map of the environment.

Case 5: Localization with known number of walls between APs and equal wall absorption of all walls

In addition to not having knowledge about the exact position of APs, this knowledge level combination evaluates SLAMWiN without having knowledge about the exact wall absorptions either. Only knowledge regarding walls is that all walls are expected to be equally thick. Thus the wall absorption will again be estimated as an average over all jumps detected in the scenario. As in the other evaluations the results will be based on measurements from 30 MUs moving through the scenario.

Figure 17.16 shows the location of the actual APs and the location of the estimated APs. The AP positions are in this knowledge level combination estimated based on the wallcount found by the initial MU moving past all APs and on the wall absorption that is calculated as the average jump size of all jumps detected by the initial MU on its trip. The initial MU has in this case found the correct number of walls between the APs and has estimated the wall absorption to be 14.63 dB - 0.37 dB smaller than the actual wall absorption.

The average distance error between the real and estimated APs is 0.28 meter. This is almost twice the error compared to the knowledge level where the wall absorption was known.

The general properties of the localization error for the 30 MUs are described in Table 17.9. The Table shows that the error in the estimation of the AP positions, which in this knowledge evaluation combination is worse than in the previous, only has a small effect on the achieved localization accuracy.

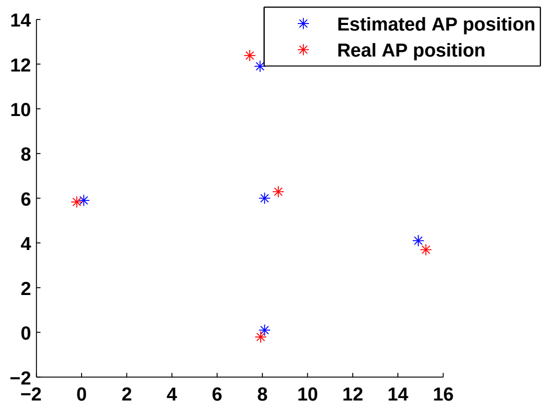


Figure 17.16: This figure illustrates the position of the real and estimated APs.

Property:	Average	Best case	Worst case	Std. deviation
Track-Metric error:	2.15 m	1.79 m	3.84 m	0.40 m

Table 17.9: This table shows the properties of the localization error achieved without knowledge of exact position of APs and with the absorption of walls estimated over all jumps in the scenario.

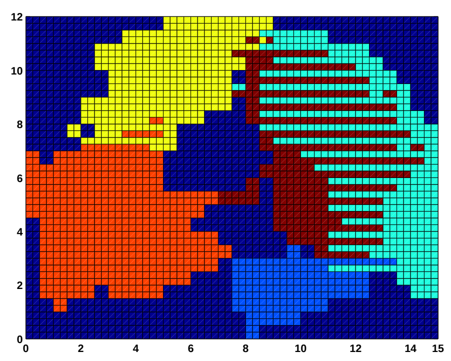
The distribution of the paths over the location error groups is as shown in Table 17.10. This time 11 of the paths in the 'Good estimation' group had a Track-Metric error below 2 meters and the average Track-Metric error of the 'Good estimation' group is increased further to 2.02 meter. Although the Track-Metric error so far has not gone much worse with less knowledge, a tendency is starting to emerge. It seams that the error is getting slightly bigger as the knowledge is reduced.

Good estimation	Reasonable estimation	Bad estimation
26 paths	2 paths	2 paths

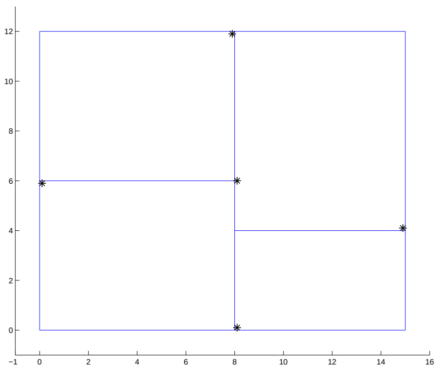
Table 17.10: This table shows the distribution of estimated paths over the three groups describing the degree of satisfaction with the achieved path estimation.

As described, the wall absorption is unknown in this knowledge level combination and is estimated as an average of all detected jump. As the same MU paths are used for all knowledge level combinations, the development of the estimated wall absorption will be equal to the one shown in Figure 17.8.

Based on the estimated paths, the build real map component again estimated the map of the environment shown in Figure 17.17 (a). Also this estimate has not suffered from the unknown AP location. Figure 17.17 (b) shows the evaluation scenario for comparison.



(a) Estimated map: Each color defines the LoS area of an AP.



(b) Real environment of the evaluation scenario.

Figure 17.17: This figure illustrates the estimated map of the environment.

Case 6: Localization with known number of walls between APs and no knowledge regarding walls

The sixth knowledge level combination tries to estimate the location of MUs based only on knowledge regarding the number of walls that obstruct the LoS between any pair of APs. The wall absorption will be estimated for each wall separately which requires MUs moving through the scenario in order for SLAMWiN to be able to generate groupflags. For this purpose 30 MUs are simulated moving through the scenario. After that the group flags have been set, another 30 MUs will move through the scenario. Based on knowledge from these MUs the AP positions will be re-estimated and a third group of 30 MUs is simulated in order to evaluate the effects of the re-estimation of the APs. The localization error for all three groups of MUs will be evaluated.

The positions of APs are in this knowledge level combination first estimated based on a known number of walls between the APs as in the previous knowledge level combination, but this time the absorption of each wall is estimated separately. Table 17.11 shows the estimated wall absorption between the AP pairs. It can be seen that the average absorption estimation error is 0.85 dB.

Based on these estimations Figure 17.18 shows the location of the actual APs and the location of the estimated APs based only on knowledge from the initial MU. Here the average distance error between the real and the estimated APs is 0.88 meter. This is a considerably larger error than in the two previous knowledge level combinations.

Although the AP positions also are re-estimated after 31 MUs, this first re-estimation will not be evaluated as it only is an intermediate step. In order to evaluate if SLAMWiN

AP pair	Estimated absorption	Actual absorption
1	15.40 dB	15 dB
2	32.47 dB	30 dB
3	14.28 dB	15 dB
4	15.27 dB	15 dB
5	16.20 dB	15 dB
6	14.20 dB	15 dB
7	0 dB	0 dB
8	14.40 dB	15 dB
9	16.20 dB	15 dB
10	15.37 dB	15 dB

Table 17.11: This table shows the estimated wall absorption between the AP pairs.

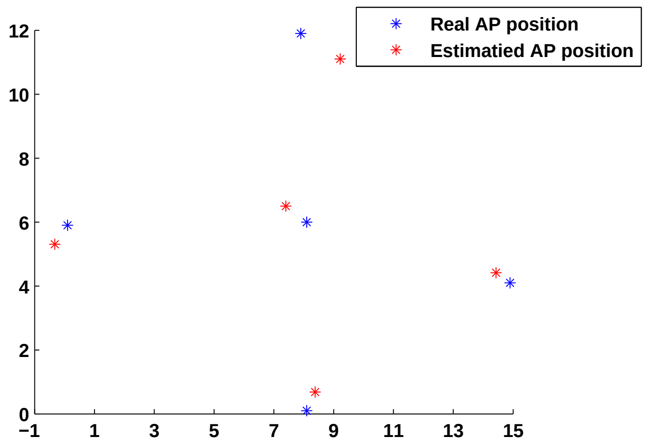


Figure 17.18: This figure illustrates the position of the real and estimated APs based on the initial MU.

is able to make a better estimate of the AP positions, the AP positions are re-estimated again after the second group of 30 MUs, that is after 61 MUs have moved through the scenario. Since the wallcounts based on the initial MU trip through the scenario are estimated correctly, these will be used again. However, the estimated absorption of walls will this time be based on the information provided by shadows covering each AP. This way the information about the estimated absorption of each wall by any MU that has passed it can be included in the calculation of an average estimated wall absorption. The larger number of detection of the walls should thus provide a better estimation of the wall absorptions.

Table 17.12 shows the wall absorption between the AP pairs after the second re-estimation. It can be seen that the average absorption estimation error is decreased by

0.36 dB to 0.49 dB.

AP pair	Estimated absorption	Actual absorption
1	15.60 dB	15 dB
2	31.08 dB	30 dB
3	15.37 dB	15 dB
4	15.05 dB	15 dB
5	15.64 dB	15 dB
6	15.80 dB	15 dB
7	0 dB	0 dB
8	15.25 dB	15 dB
9	14.36 dB	15 dB
10	14.29 dB	15 dB

Table 17.12: This table shows the re-estimated wall absorption between the AP pairs after 61 MUs.

Based on these re-estimated absorptions Figure 17.19 shows the location of the actual APs and the location of the re-estimated APs after 61 MUs. Here the average distance error between the real and the estimated APs is reduced to 0.65 meter.

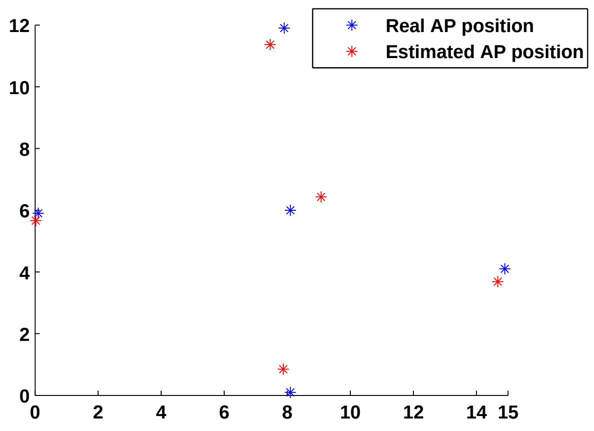


Figure 17.19: This figure illustrates the position of the real and re-estimated APs.

The evaluation of the AP position estimation in this knowledge level combination shows that the position of APs can be re-estimated with a better estimation result. This will over time contribute to a converging map of the environment.

The general properties of the localization error for the three groups of MUs are described in Table 17.13. As in the evaluation of the third knowledge level combination, the Track-Metric error decreases from the first to the second group, but does not change significantly from the second to the third. This indicates that the recalculation of the AP positions has not improved the achieved localization accuracy.

Property:	Average	Best case	Worst case	Std. deviation
Track-Metric error first group:	2.94 m	2.15 m	4.12 m	0.54 m
Track-Metric error second group:	2.68 m	2.07 m	3.81 m	0.55 m
Track-Metric error third group:	2.67 m	1.96 m	3.89 m	0.57 m

Table 17.13: This table shows the properties of the localization error achieved only with knowledge of the number of walls between the APs.

The distribution of the paths over the location error groups is as shown in Table 17.14.

	Good estimation	Reasonable estimation	Bad estimation
First group:	3 paths	9 paths	18 paths
Second group:	9 paths	9 paths	12 paths
Third group:	9 paths	8 paths	13 paths

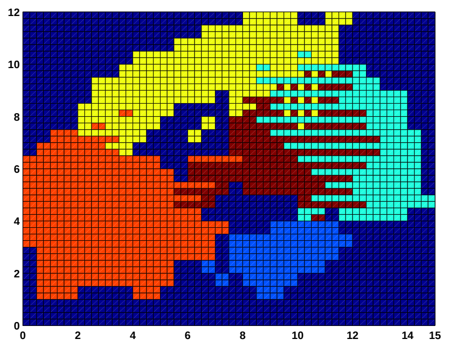
Table 17.14: This table shows the distribution of estimated paths over the two MU groups describing the degree of satisfaction with the achieved path estimation.

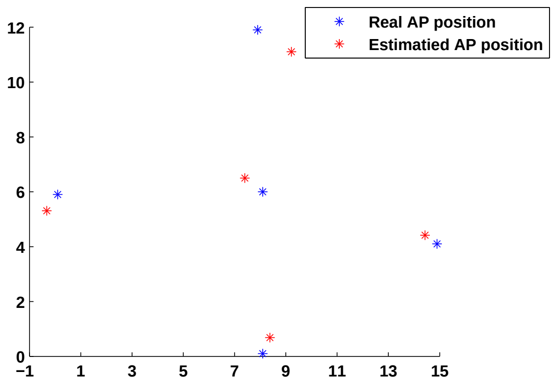
Based on the estimated paths, the build real map component again estimated the map of the environment. This was done twice for this knowledge level combination in order to show how the re-estimation of the AP positions affects the estimated map. Figure 17.20 (*a*) shows the map based on the first estimation of AP positions. The map has clearly suffered from the wrong estimation of AP positions. Figure 17.21 (*a*) shows the map based on the second re-estimation of AP positions which was shown in Figure 17.21 (*b*). The map here clearly has gained from the two re-estimations of the AP positions.

Case 7 to 9: Localization with no knowledge regarding APs

According to the evaluation approach described in the problem domain the next step would be to evaluate the three knowledge levels regarding walls for the last knowledge level regarding the APs: No knowledge regarding APs. However, the results from the previous evaluation cases combined with the used SLAMWiN approach have made this superfluous.

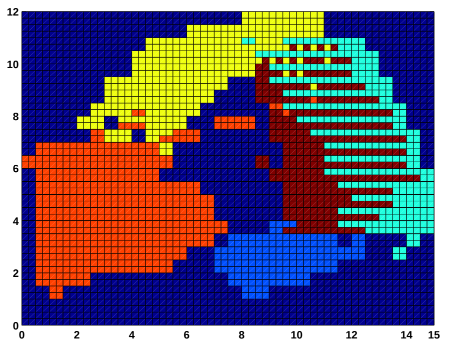
The only difference between the previous three evaluation cases and the three that should have followed here is that SLAMWiN also would have to estimate the number of walls between the APs. The evaluation cases with no knowledge regarding APs will thus provide the exact same results as the previous evaluation cases if the correct number of walls is detected.

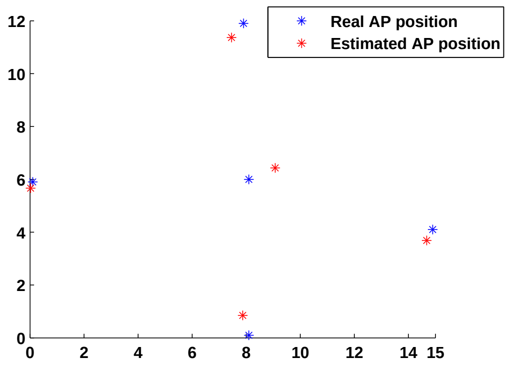

(a) Estimated map: Each color defines the LoS area of an AP.



(b) First estimation of AP positions.

Figure 17.20: This figure illustrates the estimated map of the environment.


(a) Estimated map: Each color defines the LoS area of an AP.



(b) Second estimation of AP positions.

Figure 17.21: This figure illustrates the estimated map of the environment.

Even though the number of walls between APs have been assumed known in the previous evaluation cases, it has been estimated anyway. This has been done because it is necessary to detect a wall before its absorption can be estimated. Based on this it has been observed that if the walls are detected wrongly, this will lead to wrong estimation of the wall absorption between APs and result in extremely bad AP position estimates.

To summarize, the last three evaluation cases with no knowledge regarding APs are not performed as they will provide the same results as the previous three if performed successful and because wrong detection of the number of walls can be detected very fast by observing the estimated AP positions. In such cases a new trip past all APs can be performed which, if successful, will provide the same results as shown for test case 4 to 6.

Case 10: Evaluation of unknown initial MU position

As stated in the beginning of this chapter the effect of unknown initial MU positions will be evaluated for the evaluation scenario. This is done based on the paths estimated in the last evaluation case, case 6. The evaluation will consider three different positions that can be used as default initial locations until the likely entry point or points of MUs are estimated. The three points are $(0, 0)$, the AP closest to the actual initial position of the MUs and the mass center of the APs.

Table 17.15 shows the properties of the localization accuracy for paths estimated with the actual initial MU position and for the three situations based on the 30 last paths from evaluation case 6. These are the paths calculated based on the re-estimated AP positions. Figure 17.22 illustrates the estimated initial positions of the 30 MUs that are estimated

Property:	Average	Best case	Worst case	Std. deviation
Track-Metric error actual:	2.67 m	1.96 m	3.89 m	0.57 m
Track-Metric error $(0, 0)$:	2.69 m	1.98 m	3.97 m	0.58 m
Track-Metric error closest AP:	2.67 m	2.03 m	3.67 m	0.51 m
Track-Metric error AP mass center:	2.73 m	2.07 m	3.63 m	0.54 m

Table 17.15: This table shows the properties of the localization error for the different default initial positions.

by tracking the MU paths backwards. The actual initial location of all MUs has been randomly selected within the green box. The figure shows that most estimated initial locations are very close to the green box. Considering that the used filter will cause the estimated location to lack behind the actual location the results are very good and would probably have been even closer to the green box if the filter had not had the described disadvantage.

Discussion of results from evaluation of PnP capability and localization accuracy

This subsection will discuss the results from the previous evaluation of SLAMWiN. It will give a joint evaluation of all evaluation cases and answer the questions:

- What knowledge has been most beneficial for a good localization accuracy of the SLAMWiN system?
- Can SLAMWiN work in a fully PnP setting?
- Does the position of APs and the map converge?
- What knowledge can be extracted from the estimated map of the environment?

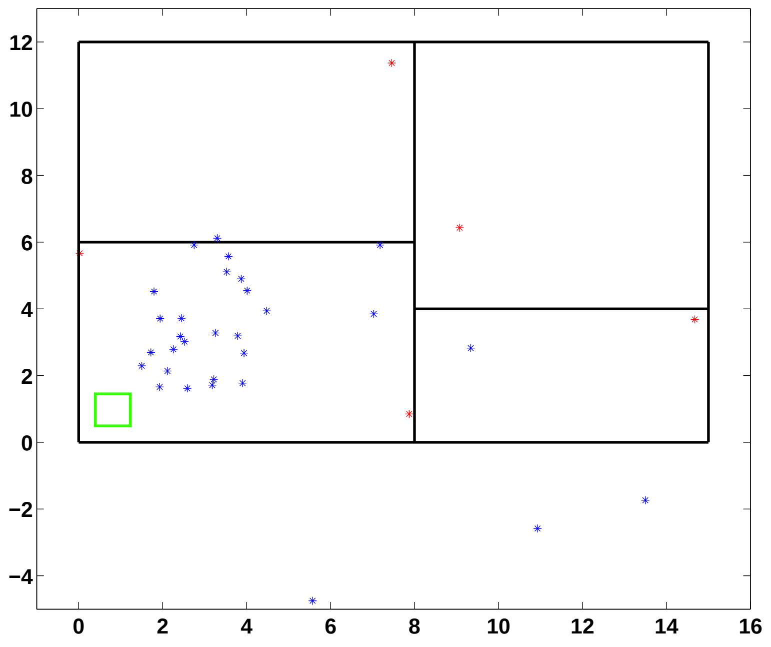


Figure 17.22: This figure shows estimated initial MU position (blue) of the back-tracked paths based on the estimated position of AP (red). Also the actual initial location is shown (green box).

Beneficial knowledge for MU localization

In order to evaluate which knowledge has been most beneficial the achieved Track-Metric error can be evaluated. An overview of the errors for the individual paths is shown in Table 17.16.

A pattern can be seen in the table which comes from all evaluation cases being based on the same RSSI measurements. Each row is based on the same set of RSSI measurements. It is thus almost always the same path that represents the worst case. Table 17.16 illustrates that the location accuracy drops when the wall absorption is unknown and furthermore cannot be estimated as an average over all jumps. Especially the first group of MUs in such situations that cannot rely on groupflags are located with a large error.

In general this shows that any knowledge that allows to base the estimation of wall absorption on more jump detections is beneficial. The more relevant jumps a wall absorption can be based on the better. This is not surprising since the jumps are affected by noise. This effect becomes smaller as the estimation is based on more jumps.

Table 17.16 also shows that a error in the estimated AP positions has only little effect on the localization error. The claim is based on that the Track-Metric error is not considerably larger if the location of APs is not estimated correctly. The average Track-Metric error is only approximately 0.18 meter larger in cases where the AP positions are estimated

	Case 1	Case 2	Case 3*a*	Case 3*c*	Case 4	Case 5	Case 6*a*	Case 6*b*	Case 6*c*
Path 1:	2.09 m	2.12 m	2.60 m	2.26 m	2.16 m	2.15 m	2.67 m	2.36 m	2.45 m
Path 2:	1.95 m	1.97 m	2.67 m	2.16 m	2.03 m	2.05 m	2.64 m	2.27 m	2.28 m
Path 3:	2.77 m	2.79 m	3.36 m	3.31 m	2.73 m	2.74 m	3.54 m	3.50 m	2.76 m
Path 4:	2.01 m	2.00 m	2.16 m	2.05 m	1.99 m	1.99 m	2.26 m	2.25 m	2.12 m
Path 5:	1.94 m	1.99 m	2.79 m	2.58 m	1.99 m	2.05 m	2.96 m	2.42 m	2.89 m
Path 6:	1.91 m	1.99 m	3.16 m	2.23 m	2.02 m	2.00 m	2.35 m	2.46 m	3.05 m
Path 7:	2.01 m	1.96 m	2.66 m	2.05 m	1.94 m	2.00 m	2.78 m	2.39 m	2.12 m
Path 8:	1.87 m	1.90 m	3.04 m	3.37 m	1.80 m	1.96 m	3.15 m	2.89 m	2.71 m
Path 9:	3.66 m	3.80 m	3.84 m	3.86 m	3.66 m	3.84 m	3.86 m	3.81 m	3.89 m
Path 10:	1.82 m	1.85 m	3.43 m	3.18 m	1.99 m	1.91 m	3.31 m	3.05 m	3.30 m
Path 11:	2.00 m	2.13 m	3.42 m	3.23 m	1.78 m	2.22 m	3.56 m	3.56 m	2.33 m
Path 12:	1.89 m	2.04 m	2.91 m	2.69 m	1.96 m	2.09 m	3.07 m	2.37 m	2.10 m
Path 13:	1.92 m	1.94 m	2.41 m	3.06 m	1.92 m	2.06 m	2.55 m	3.19 m	2.31 m
Path 14:	1.97 m	1.98 m	3.42 m	2.29 m	2.34 m	1.97 m	3.33 m	2.22 m	2.96 m
Path 15:	1.97 m	1.87 m	2.32 m	2.13 m	2.00 m	1.95 m	2.33 m	2.25 m	3.09 m
Path 16:	1.89 m	1.89 m	2.04 m	2.12 m	2.01 m	1.96 m	2.15 m	2.07 m	2.11 m
Path 17:	1.98 m	1.96 m	3.26 m	2.01 m	2.93 m	2.02 m	2.83 m	2.19 m	2.09 m
Path 18:	2.93 m	2.91 m	3.24 m	3.04 m	2.13 m	2.92 m	3.43 m	3.55 m	2.98 m
Path 19:	2.14 m	2.13 m	4.08 m	3.65 m	2.00 m	2.12 m	4.12 m	3.61 m	3.73 m
Path 20:	1.95 m	1.96 m	2.05 m	2.19 m	2.07 m	2.04 m	2.21 m	2.17 m	2.33 m
Path 21:	1.83 m	1.86 m	2.29 m	2.65 m	2.08 m	1.88 m	2.76 m	2.11 m	3.38 m
Path 22:	1.98 m	2.01 m	2.31 m	2.10 m	2.43 m	2.10 m	2.51 m	2.22 m	2.41 m
Path 23:	2.50 m	2.57 m	3.35 m	3.75 m	2.04 m	2.49 m	3.45 m	3.01 m	3.51 m
Path 24:	1.99 m	2.10 m	2.21 m	2.07 m	2.04 m	2.14 m	2.21 m	2.11 m	2.25 m
Path 25:	1.95 m	1.93 m	3.56 m	3.30 m	2.01 m	2.03 m	3.46 m	3.33 m	3.51 m
Path 26:	1.97 m	2.00 m	2.60 m	2.21 m	1.91 m	2.07 m	2.47 m	2.34 m	2.03 m
Path 27:	1.84 m	1.86 m	2.25 m	2.32 m	2.07 m	1.94 m	2.33 m	2.37 m	3.13 m
Path 28:	2.06 m	2.06 m	2.85 m	2.32 m	1.82 m	2.13 m	3.07 m	3.15 m	2.22 m
Path 29:	1.85 m	1.80 m	2.09 m	2.05 m	2.04 m	1.79 m	3.48 m	2.75 m	1.96 m
Path 30:	1.99 m	1.93 m	2.34 m	2.13 m	1.92 m	1.98 m	2.46 m	2.43 m	2.06 m
Avg.:	2.09 m	2.11 m	2.82 m	2.16 m	2.13 m	2.15 m	2.94 m	2.68 m	2.67 m
Std.dev:	0.39 m	0.41 m	0.58 m	0.59 m	0.38 m	0.40 m	0.54 m	0.55 m	0.57 m
AP error:	0 m	0 m	0 m	0 m	0.16 m	0.28 m	0.88 m	0.88 m	0.65 m

Table 17.16: This table shows the errors of the individual paths. The worst case paths are marked with red, best case paths with green and all paths in the 'good estimation' group with light green.

compared to the cases where they are known.

Based on the table and the discussion above the following sub-conclusion is made:

- The decisive factor for wether SLAMWiN provides a good localization accuracy or not is the number of jump detections on which the estimation of the wall absorption is based. The error in the estimated AP position does not have a large influence in the evaluated cases where the average error was below 1 meter.

Full PnP functionality in SLAMWiN

The problem domain set a requirement to the SLAMWiN system that it should be able to provide a localization accuracy of 2 meter. This localization accuracy was defined as:

- 'The distance between the estimated MU location and the real MU location is measured over time and the average deviation is defined as the system inaccuracy.'

This correspond to the Track-Metric used in the knowledge level combination evaluations. Regrettably, the SLAMWiN lives in non of the knowledge level combinations up to the defined goal. However, it has been shown that this is not due to the SLAMWiN approach but due to the noise on the RSSI measurements.

With the emulated environment, simplified noise and full knowledge of the evaluation scenario the achieved average localization accuracy was 2.09 meter, which comes very close to the defined goal. Of the 30 paths simulated under these conditions 21 did in fact meet the goal of a localization accuracy of under 2 meter. This shows that even under the best precondition the goal was hard to achieve. It is therefore more important to look at how much the localization accuracy grows when the SLAMWiN system becomes more PnP.

In the initial phase of the lowest knowledge level combination the achieved average localization accuracy was approximately 3 meter. This decreased to 2.67 meters over time. A converged SLAMWiN system thus only 'looses' 0.58 meter in the average localization accuracy.

Based on the table and the discussion above the following sub-conclusion is made:

- The initial goal of a localization accuracy below 2 meter was not achieved. However, this goal may have been too ambitious and the emphasis is thus laid on that the SLAMWiN system only 'looses' 0.58 meter in the localization error when all knowledge regarding walls and APs is removed.

AP position and map converges

The approach to estimate AP positions has shown to provide good position estimates with an error of under 1 meter on average provided that the correct number of walls between most AP pairs is detected. The approach can handle a limited amount of falls wallcounts by automatically excluding these from the AP position estimation based on the estimated distance error.

The dependability on correct wallcounts is not considered a large drawback of the SLAMWiN system as false wallcounts will lead to AP position estimates that have a significant error. This error can be detected early on in the process of setting up SLAMWiN and can be solved by re-doing the initial trip past all APs. The system will thus not suffer from long term affects of such a fault, but will only be more demanding in the initial setup.

Also the re-estimation of AP positions has been shown to work and have a positive effect on the SLAMWiN system. The positive effect is not so apparent in the localization accuracy, as wrongly estimated AP positions only have limited effects on this, but the environment map gains from the better AP position estimates. This was shown in evaluation case 6 where the environment map provided a better estimate of the environment after that the AP positions had been re-estimated.

It should be noted that the position estimates with an error of under 1 meter on average are achieved based on a simplified noise model. In real life the AP position estimation can thus be considerably larger than experienced here.

Based on the table and the discussion above the following sub-conclusion is made:

- The estimation and re-estimation of AP positions does not provide a considerably better localization accuracy, but supports better estimation of the environment map.

Knowledge based on map

The build real map component was introduced as an additional feature that should use the knowledge collected by SLAMWiN to support advanced mobility models which where not implemented. However, it has been shown that this feature also can be used to provide good visual representations of estimated walls in an environment. The environment maps estimated throughout the evaluation are a good example of the usability of this feature.

The environment maps has in this evaluation been used as a way to illustrate the capabilities of SLAMWiN to estimate the location of walls. It may however also be used to validate the capabilities of SLAMWiN in new scenarios. Here a user would easily be able to see whether SLAMWiN is estimating the environment correct or not.

One additional remark should be made on the environment maps. All environment maps shown here suffer from that MUs only move one way through the rooms. Since the filter causes the estimated position of the MU mostly to be tailing behind the actual position, this results in biased maps. This is reflected in that walls in the environment maps in general are estimated to be further towards the direction of which the MUs crossing it are coming. Since all MUs are moving counter-clockwise this means that the actual location of the walls is likely to be the estimated position moved slightly counter-clockwise. This bias occurs only when a majority of MU crossing a wall are coming from the same direction.

Based on the table and the discussion above the following sub-conclusion is made:

- The environment map is also without the implementation of the advanced mobility models component a useful feature of the SLAMWiN system that can be used to validate the environment estimate.

17.4 SLAMWiN in real life scenarios

The evaluation performed previously in this chapter is based on RSSI measurements generated by an environment emulator and on simplified noise. Doors are furthermore assumed to have the same absorption as walls and all walls are assumed to be solid concrete, which makes the jumps due to walls more significant and thus easier to detect. The questions is thus how the SLAMWiN system would perform in a real life scenario with real life noise and larger complexity in the environment.

In parallel to the final evaluation documented above, SLAMWiN has been applied on a real life scenario with a MU moving out of a room and into another. The scenario was chosen so that optimal conditions where provided: the walls where thick, the MU moved slow and the APs where placed advantageous with respect to jump detections. Regrettably, SLAMWiN was not able to handle this scenario due to too large variation in the noise. This was not only due to background noise, but also due to soft partitions in the environment which have been delimited in the environment emulator.

The following aspects should be considered if SLAMWiN is to be used in real life scenarios:

Angle to walls: The SLAMWiN system and the emulator does not consider that the absorptions of a signal depends on the angle it penetrates a wall with. If the angle is close to 0 degrees the penetration length will be larger and so will the absorption.

Number of measurements: As already mentioned several times in this project, the number of measurements per second is crucial for the achieved localization accuracy and for the capability to detect jumps. The measurements used in the evaluation in this report had a higher sample rate than Bluetooth can provide. A change of wireless technology would thus be required in order to get a better sample rate in real life scenarios.

Soft partitions: Another large simplification is the delimitation of soft partitions in the environment. Neither furniture nor people moving in the environment have been considered. It is obvious that especially the person holding the device tracked by SLAMWiN will have a large effect on the wireless channel. This has also been observed during test measurements where jumps could be masked by standing between the mobile wireless device and the AP that was supposed to detect the jump.

17.5 Evaluation of other relevant aspects of localization

Beyond evaluation of the main aspects of the SLAMWiN system performed in the section 17.3 it is necessary to look at some other relevant aspects of localization with SLAMWiN. This section will discuss a selection of central aspects that should be considered if SLAMWiN should be taken further than a proof of concept.

Real time localization

The implemented version of SLAMWiN performs localization of MUs off-line. This means that obtained RSSI measurements are treated after that all MU have left the scenario. For SLAMWiN to become usable in a real life scenario the treatment of measurements should be performed as a parallel process to the measurement collection.

The parameters that determines how good the real time localization capability of SLAMWiN is, are how fast a correct location estimate can be made available to MUs after that they have entered a scenario and how fast SLAMWiN can react on walls so that MUs are not located wrongly for too long after they has moved through a door, or around a corner.

It is difficult to determine exact numbers that give an indication of how good the performance of SLAMWiN is with respect to real time localization since the parameters to some degree will be scenario depended. The following will discuss both parameters and provide a numeric indication of the performance where this is possible.

Real time localization of MUs entering a scenario

The first parameter depends highly on the state of the virtual map. In a virtual map with no knowledge from previous MUs about the scenario, MU locations estimated in real time will be wrong until the MU has achieved LoS to with all APs that are making RSSI measurements of it at one point in time. In this situation recalculation of the path is necessary for every AP that did not have LoS to the MU when it entered the scenario and first when the last AP has gained LoS, its is possible to provide a correct location estimate of the MU. The time until correct localization of a MU happens depends on the complexity of the scenario and on the movement pattern of the MU. A scenario with many walls and APs will take longer than a scenario with one room and the same number of APs. Also, if the MU does not move into LoS of all APs during its stay in a scenario, the localization will never become correct.

In a converged virtual map where it is possible to estimate the wallcount and offset for the initial location of a MU, the real time localization can be provided from the first measurement off and on.

Overall the need for knowledge from previous MUs in order to be able to perform correct real time localization of MUs from the first measurement on off is by the project group expected to be a fundamental prerequisite for localization systems based on RSSI measurements.

Real time localization of MUs moving through doors

The second parameter depends on the jump detectors ability to detect jumps as fast as possible. The implemented jump detector does this after 5 or 15 measurements from the same AP depending on the used window size. Observations made during the development and the evaluation have furthermore shown that each AP only takes RSSI measurements for every 'number of APs' measurement. A jump is thus detected after either 5 or 15 times the number of APs in a scenario. In the evaluation scenarios this means that most jumps in the initial phase are detected after approximately 75 measurements while jumps detected after that groupflags have been set are detected after approximately 25 measurements due to the smaller window size.

As described in the evaluation scenario, the MU in the evaluation moves with a speed of $0.03m/s$ and it has previously been described that Bluetooth provides about $3-4$ measurements per second. This results in that a MU will have moved up to 0.75 or 0.25 meters, depending on the used window size, before SLAMWiN detects that the location estimation is wrong and is able to provide correct estimations of the MU location based on the new knowledge.

Unfortunately, such 'fast' correction of the localization can only be provided in the evaluation scenario. The actual distance moved by a MU in a real life scenario would be considerably larger as the actual, average speed of human MU walking through the scenario is approximately $1.4m/s$ - 47 times faster than in the evaluation scenario. With the current jump detector a MU would thus be able to move up to 11 meter in a fully converged map before SLAMWiN detects a jump.

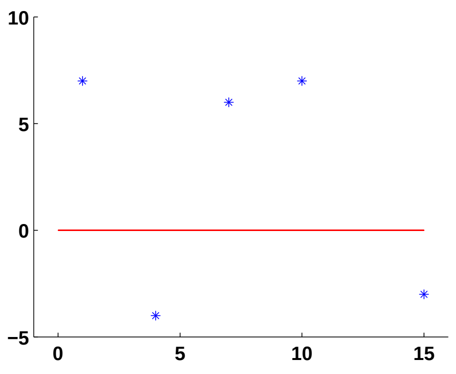
Overall is the performance of jump detector / wireless technology combination used in this project not good enough for real time localization of MUs. This is due to the unfortunate combination of a jump detector that needs too many measurements and a wireless technology that provides too few measurements per second. This further supports the indications provided by other evaluations that Bluetooth has not been the best choice of wireless technology for this project and that the jump detector should be replaced by a more advanced one.

Bluetooth as wireless technology for SLAMWiN

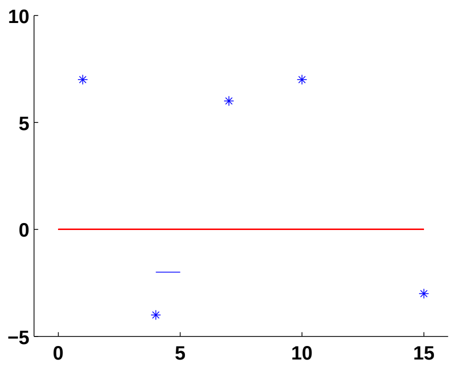
As mentioned above the Bluetooth technology does not provide enough measurement to make it possible to track a MU moving with normal human walking speed without violating the desired accuracy as described in the problem domain. Even if no walls were placed and thereby no jumps should be detected it would not be possible to track the MU with the desired accuracy.

In order to evaluate how fast a MU actually may move, this subsection will evaluate the localization accuracy for different MU speeds. The results can not directly be related to real life scenarios since the MUs are still simulated with the simplified noise model, but it will provide an indication of how fast a MU could move if Bluetooth should be used.

The relationship will be evaluated in two scenario. On with no walls shown in Figure 17.23 (a) and one with a short wall shown in Figure 17.23 (b). Figure 17.24 (a) shows a graph that illustrates the relationship between the MU speed and the localization accuracy in the first scenario without walls. Figure 17.24 (b) shows the results for the second. It can be seen that the error increases proportional to the speed and that the wall introduces a higher offset on the error. One may furthermore notice that the error increases earlier in the second scenario. This is due to that not only the localization accuracy, but also the jump detection suffers from higher MU speeds. For the middle range speeds only one jump

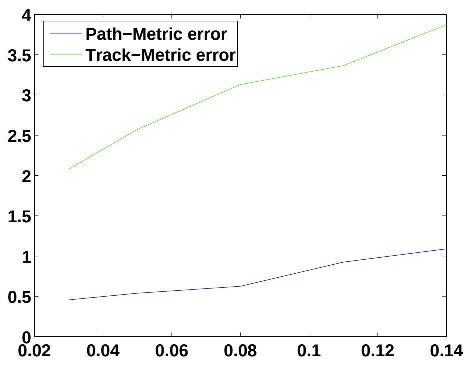


(a) First speed evaluation scenario.

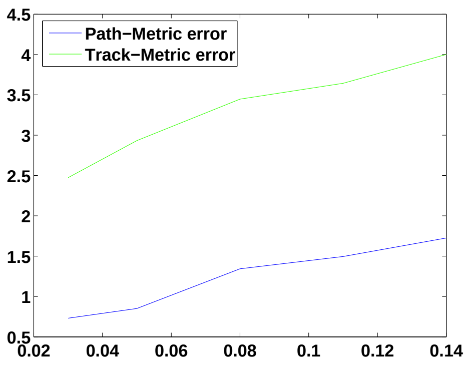


(b) Second speed evaluation scenario.

Figure 17.23: This figure shows the two scenarios used for the speed evaluation.



(a) Results from first speed evaluation scenario.

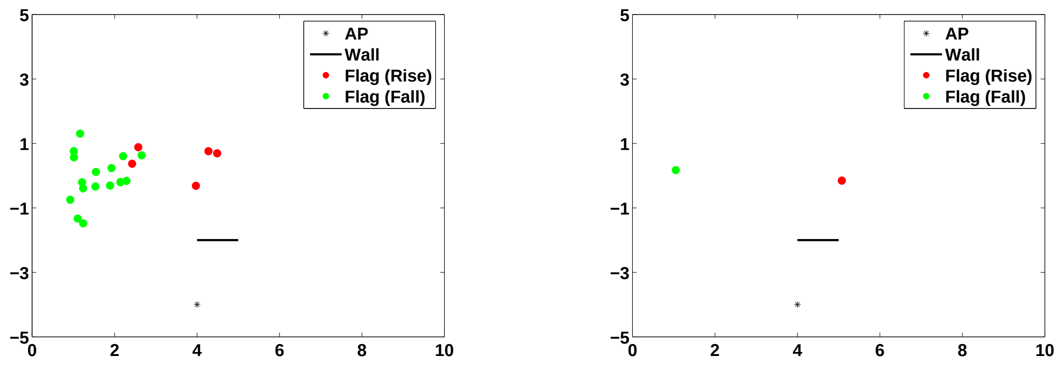


(b) Results from second speed evaluation scenario.

Figure 17.24: This figure shows the results form the two speed evaluation scenarios.

is detected while no jumps can be detected if the speed increases to much. The reason for this is that too few measurements are affected by the wall in order for them to have a large enough impact on the average of the windows used in the jump detector. This can be seen in Figure 17.25 that shows the position of estimated flags for the 30 paths. Figure 17.25 (a) shows the flags for a simulated speed of $0.08m/s$. The Figure shows that only the flags indicating a fall are set. In Figure 17.25 (b) only two flags are set since most of the MUs move too fast to notice the wall $(0.11m/s)$.

Based on the results it can be seen that Bluetooth does not provide enough measurements to track a MU moving with a normal moving speed on $1.4m/s$ that satisfy the wanted localization accuracy. Especially since the results can be expected to be worse for



(a) Flags set by MUs moving with $0.08 m/s$.

(b) Flags set by MUs moving with $0.11 m/s$.

Figure 17.25: This figure shows estimated flags from two different speeds.

real life measurements.

Chapter 18

Conclusion

This chapter will conclude on the analysis and development of the Simultaneous Localization And Mapping for Wireless Networks (SLAMWiN) approach and system described in this report. The conclusion will also comment on the filter developed in the first part of this project separately and on the SLAMWiN approach as a hole. After that the achieved results regarding the SLAMWiN PnP capability and localization accuracy will be summarized.

SLAMWiN

The goal of this project has been to investigate how indoor localization of Mobile Units (MUs) using existing wireless short-range communication networks can be performed without the need for a demanding setup phase by using the SLAM approach from self locating robots. This investigation has lead to the development of a suitable Switching Extended Kalman Filter (SEKF) and a completely new approach for mapping walls in an environment through observations of changes in the wireless channel due to walls.

The developed approach has been tested in an emulated environment and the results have shown that the approach is able to provide localization of MUs with very limited human interaction needed. Selected elements of the implementation, which need further work, have been identified and possible solutions have been pointed out. Although the investigation is based on RSSI measurements from an experimental Bluetooth testbed and an environment emulator, the results of the work are in no way restricted to this wireless technology.

The following subsections will shortly summarize and conclude on the main aspects of the work performed in this project.

SLAMWiN filter

The first step towards a SLAMWiN system was the design and implementation of a filter for making position estimates based on RSSI measurements. A SEKF using joint estimation was developed that benefitted from using knowledge regarding typical human movement behaviour in order to reduce the required number of APs needed to perform localization and to enhance the localization accuracy. An evaluation of the SEKF showed to have superior performance over two different Extended Kalman Filters (EKF) developed in previous research. The approach was evaluated based on the achieved accuracy when tracking a MU through different evaluation scenarios.

The chosen SEKF provided a better average localization accuracy in most evaluation scenarios and a more stable localization with less fluctuating position estimates.

Altogether the chosen filter was deemed suitable for the developed SLAMWiN approach.

SLAMWiN approach

The developed SLAMWiN approach is based on the assumption that the effect of walls on the wireless channel can be detected by observing significant changes in the wireless channel. These changes are used to estimate both the position and the absorption of walls in the environment. Furthermore, the method was extended so that the SLAMWiN system can take advantage of multiple MUs by collecting information about the environment through every MU passing through a scenario. This knowledge is used for better localization of future MUs, for estimation of AP positions and for estimating the likely entry point of MUs into a scenario.

The evaluation of the chosen approach showed that it is possible to enhance localization of MUs by observing changes in the wireless channel and that the location of walls, the position of APs and likely entry points of MUs into a scenario can be estimated. However, the approach heavily depends on the ability to detect significant changes in the wireless channel due to walls. A better detection of these changes has been identified as a subject for further work as the current implementation does not perform well enough to be useful under real life conditions.

SLAMWiN PnP capability

The PnP capability of the SLAMWiN system covers estimation of AP positions, estimation of the initial MU positions and estimation of walls in the environment based on only one simple initial action, where a MU has to move past all APs in a scenario. Although the approach still needs human interaction, it is considerably less than needed in the other investigated approaches to indoor localization.

Unfortunately, the current implementation relies on sub-optimal implementations of the jump detector and a classification algorithm which need scenario-specific configuration. These will have to be enhanced according to the proposed solutions in order to take the SLAMWiN system beyond the proof-of-concept state.

SLAMWiN localization accuracy

The problem domain stated a goal for the localization accuracy of under 2 meter, which could not be achieved. However, it was shown that localization based on Bluetooth and

RSSI measurements even under the best conditions cannot live up to this goal. It has instead been shown that the SLAMWiN system only looses approximately 0.58 meter when tracking a MU in real time with all prior knowledge about the scenario is removed and no assumptions are made.

It was thus shown that the developed approach was able to introduce PnP capability to a localization system while still providing a good localization accuracy when considering the accuracy achieved with a comprehensive configuration phase.

Chapter 19

Outlook

This chapter will present some of the aspects and ideas that have not been covered by this project, but which would be a natural extension to the SLAMWiN approach.

The purpose of this chapter is to formulate possible further work and demonstrate the importance of the project in the future.

19.1 Further work

In this project a method has been developed, which enables localization in indoor scenarios with a close to PnP configuration phase. However, there are still parts of the project which should be enhanced or extended and areas which have been delimited in the problem domain, but which in broader view are interesting for further work. This section will comment on some of these subjects of further work.

In the following, six tasks will be discussed, each representing either an enhancement or extension of the SLAMWiN approach.

Use smoother when recalculating

As described in the Kalman smoother section (Section 11.3 on page 89) the localization accuracy can be increased further by applying a Kalman smoother. The forward-backward algorithm of the Kalman smoother can be used to compute better location estimates if future measurements are available. This describes situations where either the complete set of measurement data is available, or as the case in SLAMWiN during recalculation, when some sub-set of the future measurements is available. When recalculating the path, previous data has to be recalculated and this gives the opportunity to use the Kalman smoother on all previous measurements. The Kalman smoother can thus be used to estimate better recalculated paths since more knowledge is available than used by the current SLAMWiN implementation.

Unfortunately, this approach only works in situations where the path is recalculated. In 'online' localization situations the system would have to compromise with the ability to provide real time localization. This delimits the Kalman smoother from being used in 'online' situation.

Other wireless technology

Bluetooth was chosen to be the short range technology used during the project. Through the project it has been discovered that $3-4$ measurements per second are not sufficient if the SLAMWiN system shall be able to track a MU moving with the normal human velocity of approximately $1.4m/s$.

Since the SLAMWiN approach is independent of the underlying wireless technology, it would be obvious to investigate other short-range wireless technologies. E.g. could IEEE 802.11 WLAN be an obvious candidate, since this technology also is very widely used and thus meets some of the same criteria that lead to the choice of Bluetooth.

Path evaluation

One of the problems with the SLAMWiN approach are false negative and false positive jump detections. If one wrongly detection is made during localization of a MU, this can in worst case influence the complete estimated path of the MU. By introducing a method that is able to detect when the localization of a MU gets worse, it could be possible to either locate the wrongly detected jump and correct it, or simply to restart localization of the MU. The consequence of this would either be increased recalculations, or many short paths. However, compared to erroneous localization of MUs, both these consequences seam preferable.

One method for detecting erroneous localization of MUs could be monitoring the trace of the state covariance. Sudden rises in the trace could indicate that the localization error is getting larger.

Leaving AP range

All tests performed in this project assume that MUs never leaves the range of any APs. For Bluetooth this assumption is under no circumstances realistic and even for wireless technologies with a larger range, it considerably limits the scalability of SLAMWiN. Therefore, this situation will have to be considered, if the approach should be used in real life scenarios.

As described in the report, one way to handle this problem could be to treat MUs returning into the range of an AP as a new MUs for this specific AP. This way the tracking of the MUs would not rely on possibly false data, but the offset and wallcount to the AP would be initialized again as if the MUs had not previously been observed by it.

However, this solution may not be feasible for SLAMWiN systems with wireless technologies with short ranges, since SLAMWiN often would have to estimate the offset and wallcount to APs. The problem is that the approach heavily relies on present shadows in the map and thus on a converged map.

Enhance jump detector

The performance of the SLAMWiN approach heavily depends of the jump detector. To gain accuracy and avoid erroneous localization of MUs due to false positive or false negative jump detections, enhancements should be made on the jump detector. The problem of finding significant changes in a noisy signal is a common problem in many areas. Thus techniques should exist that perform better than the jump detector used in the project.

One idea could be to utilize techniques used in image analysis to find edges in a picture. Another idea could be to look into methods used for monitoring biomedical signals[DF98][LV00]. Edge detection, or change-point detection, of biometrical signals can probably also be applied on noisy RSSI measurements and will hopefully lead to a better detection of jumps.

Enhancing the SLAMWiN approach by using finger printing

The SLAMWiN approach heavily depends on the jump detector which means that if jumps are detected wrongly, this will have negative consequences for the localization accuracy. Furthermore, because recalculations are necessary, some time periods will occur where localization is not correct. By combining the SLAMWiN approach with the finger printing technique, described in the Chapter 3 on page 16, one could imagine a finger printing system that could automatically be trained by using RSSI measurements from APs and the location of the MU provided by SLAMWiN.

A RSSI map could then be build and the finger printing technique could assist the localization process. In that way wrongly detected jumps would not have as large an impact on the localization accuracy and periods where recalculation is necessary would not have an influence on the accuracy, since the finger printing could take over. By adding the finger printing technique to the SLAMWiN system this would also make the initial guess much easier, since the RSSI map easily could be used to estimate the initial position.

19.2 SLAMWiN in future

Nowadays, everyone carries a mobile devices at almost any given time and wireless networks are getting more and more common in public places. By adding localization to these wireless places this would bring very strong services such as location based services, or navigation services, to consumers in e.g museums, shops and airports. However, introducing localization into present indoor wireless networks needs an excessive configuration phase, which requires a lot of time and knowhow. Furthermore these systems are not able to adapt if the environment changes over time and thus require continuing maintenance. The excessive configuration and the maintenance result in larger costs of operating localization systems which naturally affect the diffusion of such systems.

The introduction of a fully implemented, functional SLAMWiN system would bring this cost down by providing a almost PnP capable localization system that would be able to reduce the needed effort to install a localization system and the necessary maintenance.

Part IV

Appendix

Appendix A

Bluetooth

This appendix gives an outline of the Bluetooth standard and is included as nice-to-have backgroundknowledge for understanding how RSSI measurements can be made with Bluetooth.

A.1 The Bluetooth standard

The following introduction of Bluetooth is based on [sBS06] and [nBS06] and will focus on the current version of Bluetooth, version 2.0 with Enhanced Data Rate (EDR), which was adopted in November 2004. This is done even though the current version is not implemented in many devices.

Introduction to Bluetooth

Bluetooth wireless technology is a short-range communications technology intended to replace cables connecting portable and/or fixed devices. The key features of Bluetooth technology are robustness, low power consumption and low cost.

Bluetooth enables electronic devices to connect and communicate wireless through ad hoc networks known as piconets. Piconets are characterized by that all devices in the piconet share the same physical radio channel and are synchronized to a common clock and frequency hopping pattern. A Bluetooth device can simultaneously communicate with up to seven active devices within a single piconet, which are established dynamically and automatically as Bluetooth enables devices to enter and leave radio proximity. 248 further Bluetooth devices can be loosely associated to the piconet by being 'parked' in a passive mode. However, in each piconet only one of the Bluetooth devices will be master, while the remaining up to seven devices will be active slaves. The master may at any point chose to activate or deactivate a member of the piconet. Communication in the piconet is controlled by the master which handles point-to-point connections to each of the slaves as shown in Figure A.1. Each Bluetooth device can either be a master or a slave and can change its role at any time.

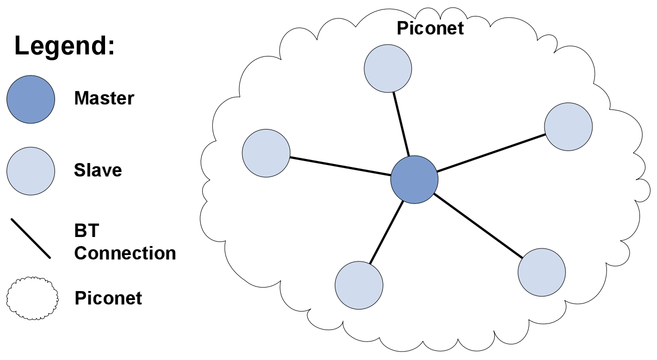


Figure A.1: Example of a piconet with one master and five slaves.

A Bluetooth device can belong to several piconets simultaneously to ensure scalability. This device, which is called a bridge node, will either be a master or a slave in each of the piconets it is part of, with the single limitation that it cannot be master in more than one piconet at a time. Preferably the device should be slave in both networks in order to obtain a higher average throughput in the network. If the device would be master and bridge node, the throughput would be limited, since the same node is responsible for forwarding traffic locally in the piconet as well as forwarding packets to other piconets. A network consisting of more than one piconet is refereed to as a scatternet. An example of a scenario with different interconnections between Bluetooth piconets are shown in Figure A.2.

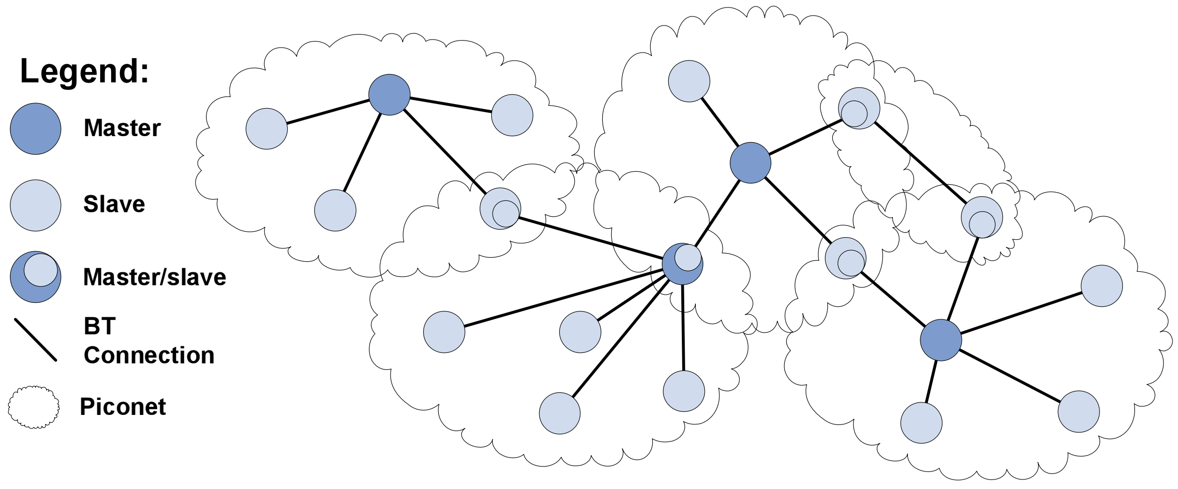


Figure A.2: This figure shows a scenario with Bluetooth devices connected through several piconets into a scatternet.

In general the Bluetooth technology provides a peak data rate of $1Mbps$ in version 1.2. This has been increased to $3Mbps$ in version 2.0 through the introduction of Enhanced Data Rate (EDR), which defines modulation changes and additional packet types. The operating range of Bluetooth devices depends on the device class of which three have been defined:

Class 3**:** Ranges of up to $1m$ by a transmitting maximum power of $0dBm$ ($1mW$).

Class 2**:** Ranges of up to $10m$ by a transmitting maximum power of $4dBm$ ($2.5mW$). This is commonly used in mobile devices where the transmitter power usually is set to $0dBm$ ($1mW$).

Class 1**:** Ranges of up to $100m$ by a transmitting maximum power of $20dBm$ ($100mW$). This is primarily used in industrial use-cases.

Bluetooth architecture

On the physical layer Bluetooth operates in the unlicensed Instrumentation, Scientific, and Medical (ISM) band at 2.4 to $2.485GHz$ using a frequency hop transceiver providing many Frequency Hop Spread Spectrum (FHSS) carriers at a nominal rate of 1600 hops per second and Time Division Multiple Access (TDMA) where each device is assigned a time slot with a length of 625μ. In order to transmit voice or other synchronous data transfers, Bluetooth provides a seemingly full-duplex signal. As TDMA cannot provide this, Bluetooth makes use of Time Division Duplex (TDD). TDD uses the same channel and antenna and continuously alternates between sending and receiving. By alternating fast enough, the signal becomes seemingly full-duplex even if it is half-duplex.

Bluetooth technology's Adaptive Frequency Hopping (AFH) capability, which was adopted in the version 2.0 specification, reduces interference between Bluetooth and other wireless technologies in the $2.4GHz$ spectrum. AFH works within the spectrum to take advantage of the available frequency. This is done by detecting other devices in the spectrum and avoiding the frequencies they are using. This adaptive hopping allows for more efficient transmissions within the spectrum, providing users with greater performance even if using other technologies along with Bluetooth technology. The signal hops among 79 frequencies at $1MHz$ intervals to give a high degree of interference immunity.

Above the Bluetooth physical layer is a layering of links and channels and associated control protocols. The hierarchy of channels and links from the lowest layer and up is:

1. Physical channel
2. Physical link
3. Logical transport
4. Logical
5. Host to Controller Interface (HCI)
6. Logical Link and Control Adaption Protocol (L2CAP) channel
7. Bluetooth profiles

Within the previously described physical channel, a physical link is formed between any two devices that transmit packets in either direction between them. The physical channel of a piconet has restrictions on, which devices may form a physical link. Only physical links between a slave and the master are allowed. Physical links are never formed directly between the slaves in a piconet.

The physical links are used as a transport for one or more logical links that support unicast synchronous (Synchronous Connection Oriented (SCO)) and asynchronous (Asynchronous Connection Oriented (ACL)) traffic and broadcast traffic within the piconet. Traffic on logical links is multiplexed onto the physical link by occupying time slots assigned by a scheduling function in the resource manager, which is described later in the Bluetooth core system subsection. In addition to user data, a control protocol, the Link Management Protocol (LMP), is also carried over the logical link.

Active Bluetooth devices in a piconet will always have a default asynchronous connection-oriented logical transport that is used to transport the LMP protocol signalling. This will be created whenever a device joins a piconet. Additional logical transports may be created to transport synchronous data streams when this is required.

The previously described layers belong to the Bluetooth controller sub-system and are defined so that the Bluetooth controller may be considered a standard part of every Bluetooth implementation. This results in that Bluetooth implementations can be spilt into two different parts: A Bluetooth controller that operates on the lowest three layers and Bluetooth host system that includes the L2CAP channel, the Bluetooth profiles and the Bluetooth Applications. The standard interface between these two parts is called the Host to Controller Interface (HCI). Implementation of this standard service interface is optional.

Above the HCI the Logical Link and Control Adaption Protocol (L2CAP) layer provides a channel-based abstraction to applications and services. It carries out segmentation and reassembly of application data and multiplexing and de-multiplexing of multiple channels over a shared logical link. L2CAP has a protocol control channel that is carried over the default logical transport. Application data submitted to the L2CAP protocol may be carried on any logical link that supports the L2CAP protocol.

In order to use the Bluetooth wireless technology, a device must be able to interpret certain Bluetooth profiles. The Bluetooth profiles define the possible applications that a given device can handle. The profiles describe the general behaviours through which a specific application on a Bluetooth enabled devices can communicate with other devices. A wide range of profiles are already available for all types of different use cases. At a minimum, each profile specification contains information on the following topics:

- Dependencies on other profiles.
- Suggested user interface formats.
- Specific parts of the Bluetooth protocol stack used by the profile.

Examples of widely used Bluetooth profiles are profiles supporting voice transmission, Transmission Control Protocol/Internet Protocol (TCP/IP), or Real-Time Protocol (RTP).

Figure A.3 gives an overview of the layers that have been described above. It includes the Bluetooth controller, the L2CAP controller and Bluetooth profiles and applications. Furthermore it shows how the layers fit into the OSI reference model[Zim80]. In the

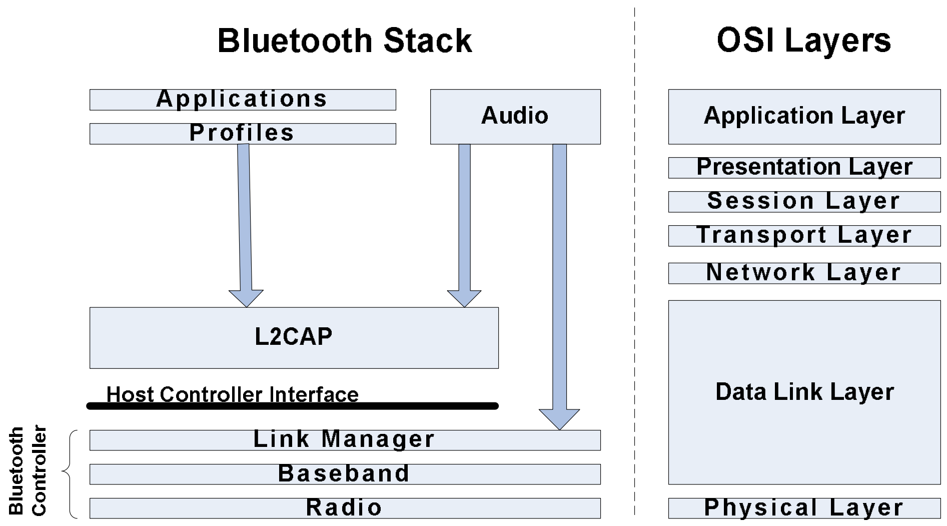


Figure A.3: This figure shows the protocol stack of Bluetooth.

following the details of the lower layers of Bluetooth, from the L2CAP on down, are described.

Bluetooth core system

The core system of Bluetooth consists of an RF transceiver, a baseband and a protocol stack. The system offers services that enable the connection of devices and the exchange of a variety of data classes between these devices. An overview of the Bluetooth core system is shown in Figure A.4, which is taken from the Bluetooth specification[rBS04] and gives an idea of how Bluetooth is implemented in devices. The Bluetooth core system specifies the details of the four lowest layers and the associated protocols in the Bluetooth protocol stack defined in the Bluetooth specification. In the following the main building blocks of the Bluetooth core system will shortly be introduced. The introduction of the blocks is based on the description of them in the Bluetooth specification[rBS04].

Channel Manager: The channel manager is responsible for creating, managing and destroying Logical Link and Control Adaption Protocol (L2CAP) channels. L2CAP channels are used for the transport of service protocols and application data streams between Bluetooth devices. The channel manager uses the L2CAP protocol to interact with the channel manager on remote Bluetooth devices to create these channels and connect their endpoints to the appropriate entities. The channel manager interacts with its local link manager to create new logical links and to configure these links to provide the required quality of service for the type of data being transported.

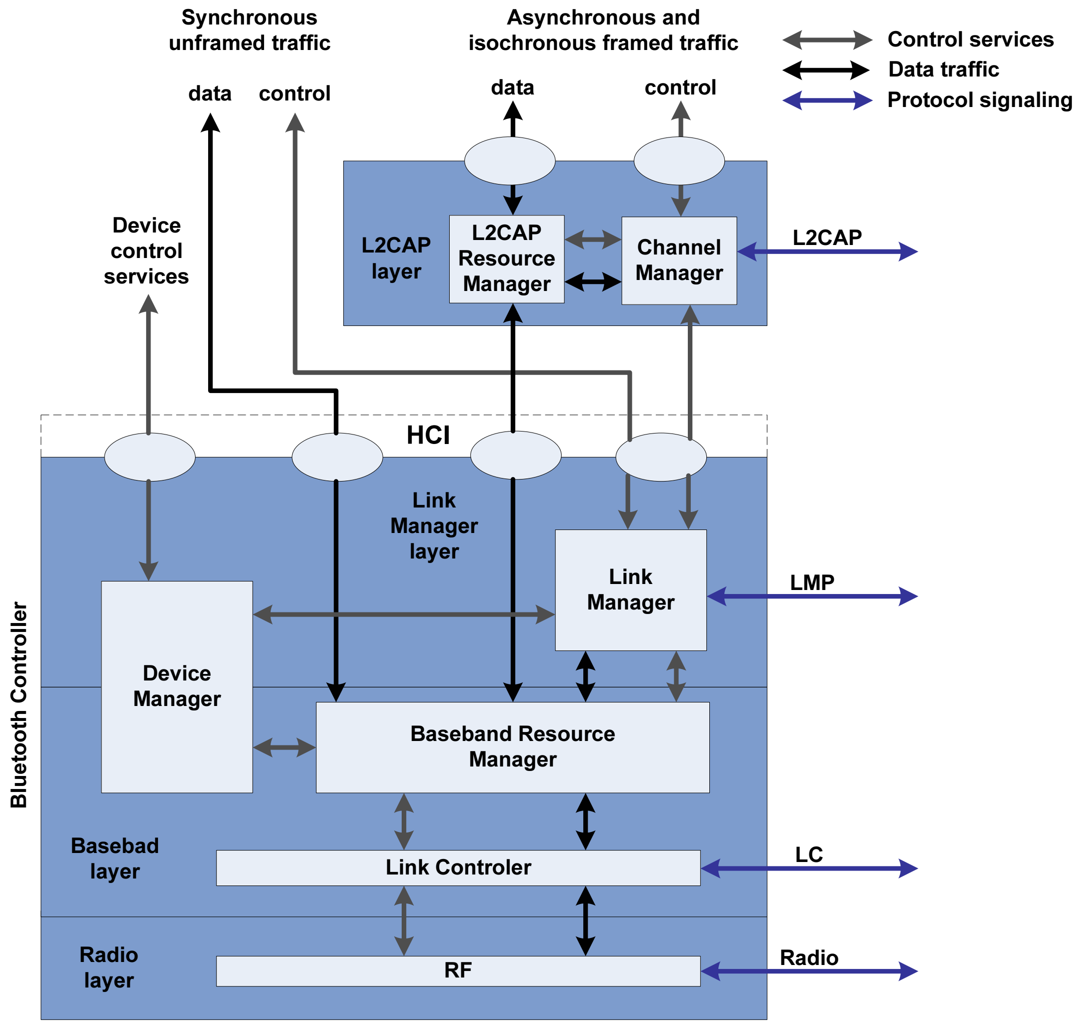


Figure A.4: This figure shows the Bluetooth core system.

L2CAP Resource Manager: The L2CAP resource manager is responsible for managing the ordering of submission of data packet fragments on the baseband and some relative scheduling between channels. In that way it ensures that L2CAP channels with Quality of Service (QoS) commitments are not denied access to the physical channel due to limited resources in the lower layers. The L2CAP resource manager is required, because the architectural model of Bluetooth does not assume that the Bluetooth controller has limitless buffering, or that the Host to Controller Interface (HCI) is a pipe of infinite bandwidth.

Link Manager: The link manager is responsible for the creation, modification and release of logical links as well as the update of parameters related to physical links between devices. This is achieves by communication between the two link managers on the connected Bluetooth devices using the Link Management Protocol (LMP).

The LMP allows the creation of new logical links between devices, as well as the general control of link and transport attributes e.g. the enabling of encryption, the adapting of transmit power on the physical link, or the adjustment of QoS settings for a logical link.

Device Manager: The device manager controls the general behaviour of the Bluetooth device. It is responsible for all operation of the Bluetooth system that is not directly related to data transport. This could be inquiring for the presence of other nearby Bluetooth devices, connecting to other Bluetooth devices, or making the local Bluetooth device discoverable by other devices. The device manager requests access to the transport medium from the baseband resource controller in order to carry out its functions.

The device manager also controls local device behaviour implied by a number of the HCI commands, such as managing the device local name, any stored link keys, and other functionality.

Baseband Resource Manager: The baseband resource manager is responsible for all access to the physical medium. It has two main functions: It is a scheduler that grants time on the physical channels to all of the entities that have negotiated an access contract and it negotiates the access contracts with these entities.

An access contract is effectively a commitment to deliver a certain QoS that is required in order to provide a user application with an expected performance. The access contract and scheduling function must take account of any behaviour that requires use of the Bluetooth radio. This could be the normal exchange of data between connected devices over logical links, and logical transports.

Link Controller: The link controller is responsible for the encoding and decoding of Bluetooth packets from the data payload and parameters related to the physical channel, logical transport and logical link. The link controller carries out the link control protocol signalling, which is used to communicate flow control and acknowledgement and retransmission request signals.

RF: The Radio Frequency (RF) block is responsible for transmitting and receiving packets of information on the physical channel. The timing and frequency carrier of the RF block is controlled by the baseband resource manager. The RF block transforms a stream of data to and from the physical channel and the baseband resource manager into required formats.

After having described how Bluetooth in general works, the following section will discuss how Bluetooth can be utilized for localization purposes.

Appendix B

Mobility model examples

This appendix gives an introduction to several mobility models and is included as a nice-to-have backgroundknowledge for understanding the mobility models used in the project.

B.1 Examples of mobility models

This appendix will present and shortly discuss different categories of mobility models which can be applied to the project scenario. Figure B.1 gives an overview of which categories of mobility models this section will present. Each of the categories are described in the

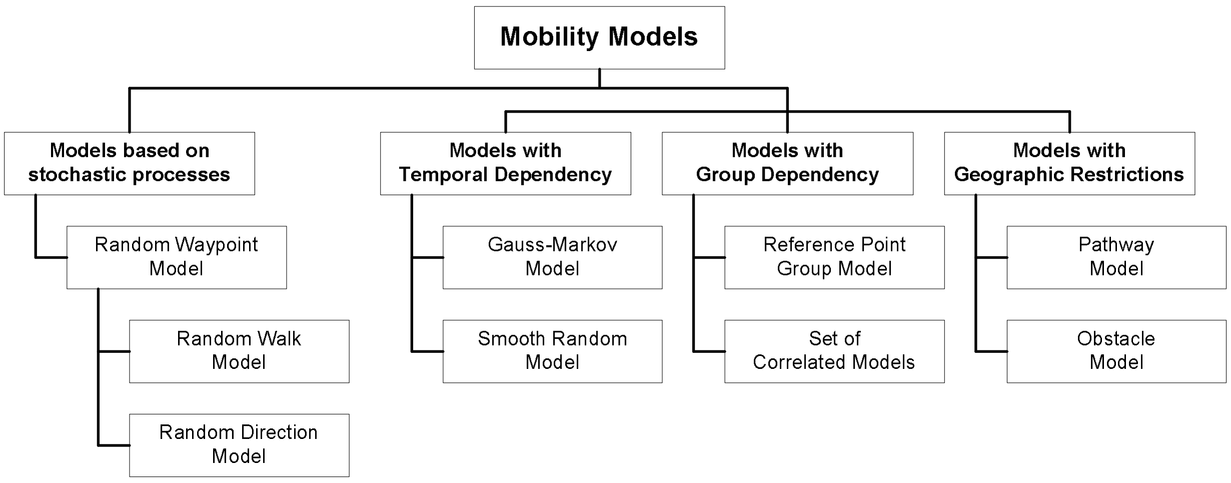


Figure B.1: The categories of investigated mobility models.

following.

Mobility models based on stochastic processes

In mobility models based on stochastic processes, initially MUs are assumed to move randomly and freely without any restrictions within the area covered by the network.

This is achieved by choosing either a destination and speed, or a direction of movement and speed based on a stochastic process. Furthermore the movement of MUs is assumed independent from other MUs. An example of such a mobility model which often is used is the Random Waypoint Model (RWM)[JBJ98]. However, there are also mobility models around, which do not meet the definition above in order to reflect some special case of MU behaviour. A selection of these models will be described in this chapter as well.

In the basic mobility model based on a stochastic process, each MU initially selects its start position randomly within the field. After that it selects a destination (x, y) within the same bound area and starts to move towards it with a randomly chosen speed within the range $[V_{min}, V_{max}]$. Normally V_{min} will be 0 and V_{max} will be the maximum allowed speed set by the model designer. Destination and speed are always chosen independent. Upon reaching the chosen destination the MU will wait for a duration of T_{pause} until it chooses a new direction and new speed. This procedure is repeated as long as needed and can be used in case with continuous time as well as discrete time. An example of the RWM in action is shown in Figure B.2 where the described procedure is completed seven times.

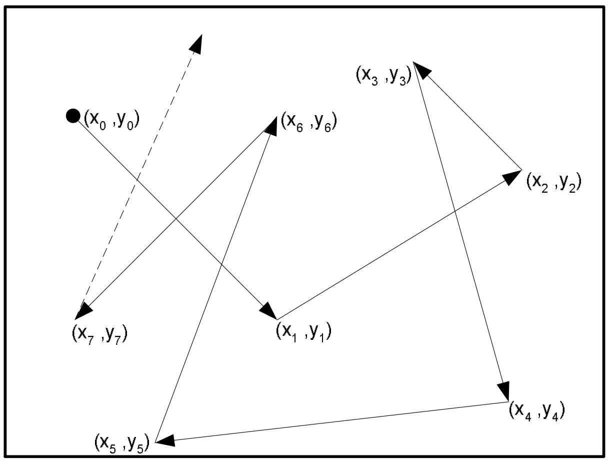


Figure B.2: Example of a MU moving according to the Random Waypoint mobility model.

The RWM allows for a creation of various mobility scenarios based on different values for the parameters V_{min}, V_{max} and T_{pause}. Examples of this could be to set the T_{pause} parameter to 0, which would reflect a MU with a high degree of mobility, or to set V_{min} equal to V_{max}, which would remove some of the dynamics from the mobility model.

Variations of the RWM are the Random Walk Model and the Random Direction Model.

- The Random Walk Model[JZH98] is based on a randomly chosen direction $\theta(t)$ from $(0, 2\pi]$ and speed $v(t)$ from $(0, V_{max})$ for every instance t in a discrete domain. This results in movement according to the velocity vector shown in Equation B.1.

$$\mathbf{V}(t) = \begin{bmatrix} v_x(t) \\ v_y(t) \end{bmatrix} = \begin{bmatrix} cos(\theta(t)) \cdot v(t) \\ sin(\theta(t)) \cdot v(t) \end{bmatrix} \tag{B.1}$$

- The Random Direction Model[NTLL05] is like the previous model based on a randomly chosen direction $\theta(t)$ from $(0, 2\pi]$ and speed $v(t)$ from $(0, V_{max})$. However, new values for the parameters are only chosen whenever the MU reaches the boundaries of the bound area and new value will be chosen so that the MU will move back into the field.

The T_{pause} parameter can be added to both models.

The RWM and its variants have been designed to model the movement of MUs in a simplified way. Therefore they may not always be sufficient since they do not capture certain mobility characteristics from real scenarios. The following subsection will discuss the three major characteristics which are not considered in the RWM: temporal dependency, group dependency and geographic dependency.

Models with temporal dependency

The mobility models described above all randomly chooses a value, e.g. speed, independently from previously chosen values. This does not reflect the behaviour of many real life situations where e.g. the speed of vehicles will change incrementally and also change of direction in most cases will be smooth.

Examples of a mobility models that respect temporal dependency are the Gauss-Markov Model[LH03] and the Smooth Random Model[Bet01a].

The Gauss-Markov Model assumes that the velocity of a MU is correlated over time and models it as a Gauss-Markov stochastic process. The Gauss-Markov Model for a two-dimensional field is shown in Equation B.2 and again Δt denotes the time slot size.

$$\mathbf{V}_t = \begin{cases} v_t^x = \alpha v_{t-\Delta t}^x + (1-\alpha)v^x + \sigma^x\sqrt{1-\alpha^2}w_{t-\Delta t}^x \\ v_t^y = \alpha v_{t-\Delta t}^y + (1-\alpha)v^y + \sigma^y\sqrt{1-\alpha^2}w_{t-\Delta t}^y \end{cases} \tag{B.2}$$

Here $\mathbf{V}_t = [v_t^x, v_t^y]^T$ represents the velocity vector at time t and $[w_{t-\Delta t}^x, w_{t-\Delta t}^y]^T$ represents the uncorrelated random Gaussian process with zero mean and variance σ^2 at $t - \Delta t$. α, $[v^x, v^y]^T$ and $[\sigma^x, \sigma^y]^T$ are vectors that represent the memory level, asymptotic mean and asymptotic standard deviation, respectively.

As it can be seen from equation B.2, the velocity $\mathbf{V}$ at time t is dependent on the velocity at time $t - \Delta t$. The degree of temporal dependency is determined by the value of the memory level α and by adjusting this value different scenarios can be simulated. Two special cases of the Gauss-Markov model exist, one were α is 0 and one were α is 1. In the first case the Gauss-Markov model will behave like the Random Walk Model as can be seen in Equation B.3 and in the second, where the memory is strong, the velocity will be constant as shown in Equation B.4. The second case is also referred to as the Fluid Flow Model.

$$\mathbf{V}_t = \begin{cases} v_t^x = v^x + \sigma^x w_{t-\Delta t}^x \\ v_t^y = v^y + \sigma^y w_{t-\Delta t}^y \end{cases} \tag{B.3}$$

$$\mathbf{V}_t = \begin{cases} v_t^x = v_{t-\Delta t}^x \\ v_t^y = v_{t-\Delta t}^y \end{cases} \tag{B.4}$$

If a MU should travel beyond the boundaries, its direction of movement is flipped 180°.

The Smooth Random Model represents another approach to model smooth MU movement. This model works with preferred speeds $\{V^1_{pref}, V^2_{pref}, ..., V^n_{pref}\}$ that a MU most likely will use and with smooth acceleration and deceleration. The velocity is calculated in the discrete domain for each time slot as in Equation B.5 where $a(t)$ is uniformly distributed among $[a_{min}, 0]$ or $[0, a_{max}]$.

$$v(t) = v(t - \Delta t) + a(t)\Delta t \tag{B.5}$$

In this model $a(t)$ determines the degree of temporal dependency in the way that a low $a(t)$ value will result in slow velocity changes and high $a(t)$ value will result in fast velocity changes. This represents then a low and high degree of temporal dependency, respectively.

Unlike the speed, this model assumes the movement direction to be uniformly distributed in the interval $[0, 2\pi]$. However, a new movement direction is not chosen for every time slot, but for a larger interval. This interval could be chosen according to a exponential distribution as in [Bet01b]. Furthermore, since a new movement direction potentially can vary from the current with an angel of up to 180°, the change in direction is spread over several time slots to smoothen the direction change. Hence, a change of direction by $\Delta\theta(t)$ in an scenario with an allowed maximum direction change per time slot of $\Delta\varphi(t)$ will take $\frac{\Delta\theta(t)}{\Delta\varphi(t)}$ time slots. An example of this is shown in Figure B.3 where a vehicle uses three time slots to perform a change in direction of $\Delta\theta(t)$ degrees.

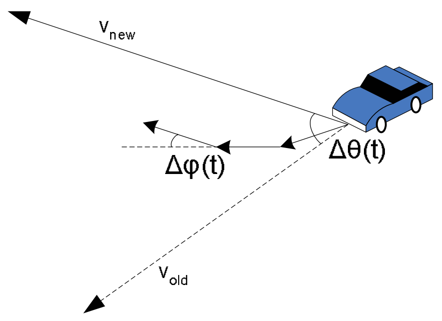


Figure B.3: A vehicle stepwise changing its direction of movement with the maximum allowed change per time slot.

The mobility models discussed in the next subsection treat dependencies between different MU in the same field.

Models with group dependency

Independently of temporal behaviour, all mobility models discussed so far have described the movement of a single MU. This assumes that a MU is independent from all other MUs. However, scenarios exist where this is not the case. In scenarios like battlefield communication and museum touring the movement pattern of a MU may very well be correlated with the movement pattern of other MUs near by. This could e.g. be a guided group tour through a museum where all visitors have the tendency to move at the same time and in the same direction as the tour guide.

The first mobility model with group dependency introduced here is the Reference Point Group Model[HGPC99]. Here MUs move as a group around a group leader or common group center that determines the mobility model for the group. For simplicity, we will here use an example with a group leader. Each group will comprise of a group leader and several group members and the group leader will determine the mobility behaviour of the entire group.

Figure B.4 shows an example of a group with one leader and two members. The movement of the group leader is represented by the motion vector $\mathbf{V}_{group\_t}$ for $t = (1, 2)$. $\mathbf{V}_{group\_t}$ thereby denotes the general motion not only of the leader, but of the entire group. All members of the group will then move according to the general motion of the group and some additional random vector $\mathbf{RM}_{i\_t}$ where i denotes that the MU is the i^{th} member of the group and t denotes the time slot. This is shown in Equation B.6.

$$\mathbf{V}_{i\_t} = \mathbf{V}_{group\_t} + \mathbf{RM}_{i\_t} \tag{B.6}$$

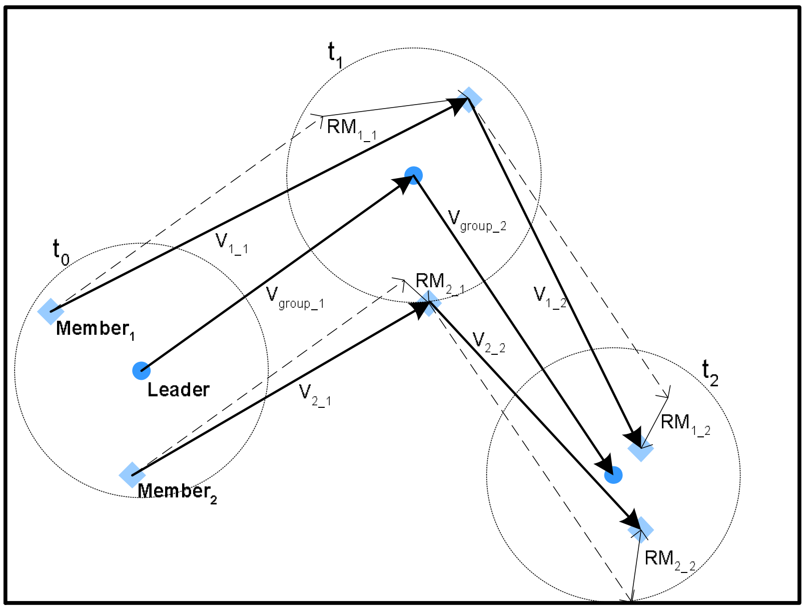


Figure B.4: Example of three MUs moving according to the Group Reference mobility model.

The vector $\mathbf{RM}_{i\_t}$ is chosen from a independent identically distributed random process whose length is distributed in the interval $[0, r_{max}]$ and whose direction is uniformly distributed in the interval $[0, 2\pi)$.

A larger set of correlated models for MUs exist, but these will not be discussed here. This set includes the Column Mobility Model, Pursue Mobility Model and the Nomadic

Mobility Model. Instead the next subsection will discuss mobility models which are able to reflect geographic restrictions.

Models with geographic restrictions

The mobility models introduced in the two previous subsections mainly considered special characteristics of a MU. However, also the surroundings of a MU affect the mobility model. In most scenarios the mobility of a MU is limited by obstacles such as walls, streets or other MUs. A mobility model for such scenarios should reflect the restriction of MU movement.

This subsection will discuss two mobility models which take such limitations into account. These models are the Pathway Model and the Obstacle Model.

In the Pathway Model MU movement is restricted to certain paths. These paths may either be randomly generated or carefully defined based on information about a scenario. Typically this model is used to simulate movement along streets or freeways. The Freeway and Manhattan Models are very similar to this model only with a more specific pattern in the deployment of paths. An example of all three models is shown in Figure B.5. Both the

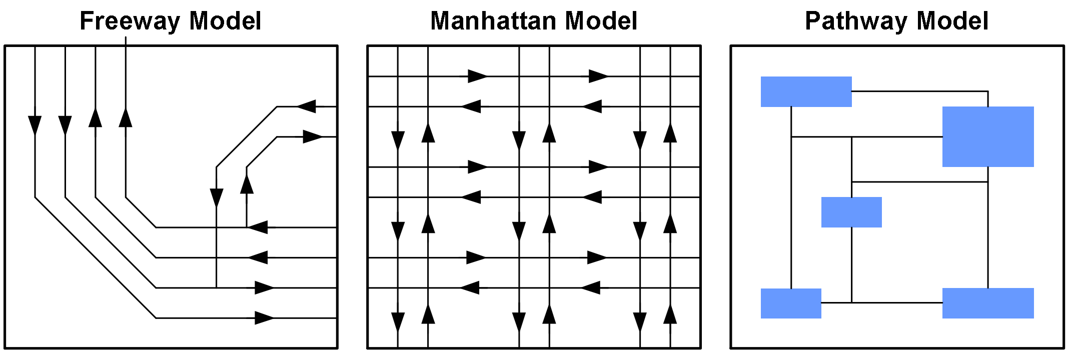


Figure B.5: Different pathway graphs used in the Freeway, Manhattan and Pathway Model.

initial position and the destination of the MUs are determined as random chosen points on the path. This ensures some degree of randomness in the models.

Another approach taken by [JLH$^+$99] is the Obstacle Model that introduces randomly placed obstacles. The obstacles affect the movement of a MU in the way that they force a MU to change its trajectory if it encounters an obstacle.

Appendix C

Filters

This appendix will introduce and describe the details of the filters used in this project. The filters are described in order of increasing complexity starting with the simple discrete time Kalman filter.

The purpose of this appendix is to provide the reader with details of the filters used in this project.

C.1 Discrete time Kalman filter

The Kalman filter introduced by R.E. Kalman in 1960 describes a recursive solution to the discrete-data linear filtering problem. It uses a set of mathematical equations that efficiently apply the least-squares method to support estimation of past, present and future states of systems with models of unknown nature. The following description of the Kalman filter is based on the original Kalman paper [Kal60] and a more up to date introduction to Kalman filters[WB01].

The Kalman filter solves the problem of estimating any given state $\mathbf{x}$ of a discrete time controlled process e.g. the movement of an object or person over time. The *process model* is denoted

$$\mathbf{x}(k) = \mathbf{A}\mathbf{x}(k-1) + \mathbf{B}\mathbf{u}(k) + \mathbf{v}(k) \tag{C.1}$$

where $\mathbf{A}$ is the *state transition matrix* that relates the previous state $\mathbf{x}(k-1)$ to the current state $\mathbf{x}(k)$, $\mathbf{B}$ relates the control input $\mathbf{u}$ to the state $\mathbf{x}$ and $\mathbf{v}$ represents a temporally uncorrelated process noise with zero mean and covariance. Note that matrix $\mathbf{A}$ in some systems might change with each time step and therefore may be denoted $\mathbf{A}(k)$. Here it is assumed constant.

Measurements are introduced in the form of a *observation model*

$$\mathbf{z}(k) = \mathbf{H}\mathbf{x}(k) + \mathbf{w}(k) \tag{C.2}$$

where $\mathbf{H}$ is the *measurement matrix* that describes the relation between the measurement $\mathbf{z}(k)$ and the state $\mathbf{x}(k)$. Also $\mathbf{H}$ might change over time and can be denoted $\mathbf{H}(k)$. Here it is assumed constant.

The Kalman filter needs the process and measurement noise covariance shown in Equation C.3 and C.4.

$$\mathbf{Q}(k) = E\{\mathbf{v}(k) \cdot \mathbf{v}(k)^T\} = \begin{bmatrix} Q_{xx} & 0 \\ 0 & Q_{yy} \end{bmatrix} \tag{C.3}$$

$$\mathbf{R}(k) = E\{\mathbf{w}(k) \cdot \mathbf{w}(k)^T\} = \begin{bmatrix} R_{xx} & 0 \\ 0 & R_{yy} \end{bmatrix} \tag{C.4}$$

Furthermore it is necessary to initialize the Kalman filter with a set of parameters. These parameters are the initial location $\mathbf{x}(0)$ based on the first measurement $\mathbf{z}(0)$ and $\mathbf{P}(0)$ which is the initial error covariance matrix. $\mathbf{P}(0)$ is an indication of how good the initial location estimation is on a scale from $\varepsilon = 0$ to $\varepsilon = 1$. Here 0 indicates a precisely known location whereas 1 indicates an unknown location. The parameters are shown in Equation C.5 and C.6.

$$\mathbf{x}(0) = \mathbf{H} \cdot \mathbf{z}(0) \tag{C.5}$$

$$\mathbf{P}(0) = \begin{bmatrix} \varepsilon & 0 \\ 0 & \varepsilon \end{bmatrix} \tag{C.6}$$

Based on the process model in C.1 and the observation model in C.2 the Kalman filter provides the estimation of a state $\mathbf{x}(k)$ through a recursive feedback loop computation. An overview of this is shown in Figure C.1 where the equations are divided into two groups: *time update* and *measurement update*.

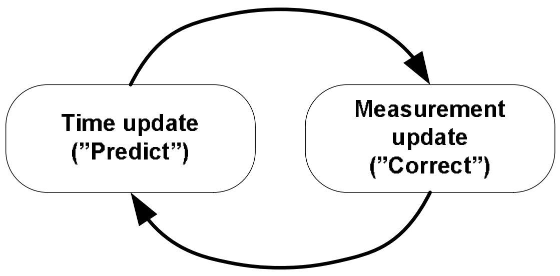


Figure C.1: This figure shows an overview of the feedback loop in Kalman filters.

In the first group the next state and error covariance are predicted. This is done through the equations C.7 and C.8.

$$\hat{\mathbf{x}}(k|k-1) = \mathbf{A}\hat{\mathbf{x}}(k-1|k-1) + \mathbf{B}u(k) \tag{C.7}$$

$$\mathbf{P}(k|k-1) = \mathbf{A}\mathbf{P}(k-1|k-1)\mathbf{A}^T(k) + \mathbf{Q}(k) \tag{C.8}$$

where $\mathbf{Q}(k)$ is the covariance of the process noise.

Following the prediction a measurement is made like described in the observation model C.2.

Then, in the correction, the Kalman filter gain is computed and both the state prediction and the error prediction are updated. This is shown in C.9, C.10 and C.11.

$$\mathbf{k}(k) = \mathbf{P}(k|k-1)\mathbf{H}^T(k)(\mathbf{H}(k)\mathbf{P}(k|k-1)\mathbf{H}^T(k) + \mathbf{R}(k))^{-1} \tag{C.9}$$

where $\mathbf{R}(k)$ is the covariance of the measurement noise.

$$\hat{\mathbf{x}}(k|k) = \hat{\mathbf{x}}(k|k-1) + k(k)(\mathbf{z}(k) - \mathbf{H}\hat{\mathbf{x}}(k|k-1)) \tag{C.10}$$

$$\mathbf{P}(k|k) = (\mathbf{I} - k(k)\mathbf{H})\mathbf{P}(k|k-1) \tag{C.11}$$

Appendix D

Environment emulator

This chapter will describe the details of the environment emulator used to generate RSSI measurements based on a virtual environment. The purpose of this chapter is to describe how the emulator works, detailing approximations and assumptions used in the emulator.

To verify that the SLAMWiN implementation works as intended it is necessary to test the implementation in different scenarios with varying wall configurations. Because the exact placement and number of walls is crucial in these scenarios, it can be difficult to find real life scenarios that matches exactly these requirements. To overcome these problems an emulator is used instead to emulate the radio propagations in different environments. Based on some inputs the emulator is able to generate RSSI measurements from APs placed in the environment. The needed inputs are:

Placement of walls: Where all the walls are placed. Each wall is defined by two sets coordinates which describe the end-points of the wall.

Placement of APs: Where all APs are placed. Each AP is defined by a set of coordinates (x, y).

Path of MU: A path that describes the movement of the MU. The path is defined by an array of coordinates that represent waypoints along the path.

To reduce the complexity of the emulator some simplifications are made to each emulated scenario. These are listed in the following:

- The scenario will only be emulated in the 2-dimensional plane.
- The scenarios do only have straight walls.
- The MU moves with a constant speed.
- The MU will move at all times.

- The frequency of measurements are constant. However, because it is not possible for a MU to receive multiple Bluetooth inquiries and thereby multiple measurements at the same time, the measurement from one AP will randomly be chosen for each time step.
- Each scenario will only have one type of wall.

The output from the emulator is a log file with same structure as described in the Filter evaluation testbed appendix (Appendix G on page 240) containing measurements from the APs. These RSSI measurements are calculated based on different propagation effects which is described in the following.

Path loss

To approximate the free-space path loss for the direct ray between APs and the MU, Equation D.1 is used which is identical with the observation model for the filter described in Section 11.2 on page 82. This model does not consider any obstacles such as walls in between and the model assumes that omni-directional antennas are used.

$$\text{Path Loss (in dBm)} = \alpha + \beta \cdot [\log(D \text{ in meters})] \tag{D.1}$$

The values of α and β depend on the environment, e.g. if it is a indoor or outdoor environment, and the hardware that is used to make the measurements. In this emulator the α and β values are set to -67 and 2.1 respectively which corresponds to the estimated values based on real life measurements in Chapter 11 on page 81 regarding the filter. To handle none-LoS situations, the number of walls between the AP and the MU is multiplied with a wall penetration loss constant. The penetration loss of walls is only approximated, as the entrance angle of a signal through the wall will not be considered even though this has an influence on the penetration loss. The penetration loss constant pl_c is as default set to -15 dBm which corresponds to a concrete wall (see Chapter 12 on page 105).

Multipath fading

Another effect that has a big influence on the propagation and thus needs to be considered, is the effect caused by multipath fading. Because of reflections from hard and soft partitions and scattering the received signal will be a sum of contributions coming from many different directions. Since these signals can differ in attenuation, delay and phase shift, constructive and destructive interference will happen which will cause the received signal to vary when a MU moves over small distances. Variation in the amplitude caused by multipath fading will vary randomly according to a Rayleigh distribution[Gol05]. Because the received signal is described in Watt these fluctuations will vary randomly according to an exponential distribution.

The probability density function of an exponential distribution is shown in D.2.

$$f(x) = \frac{1}{\mu} \cdot e^{\frac{x}{\mu}} \tag{D.2}$$

To estimate the μ value, real life measurements have been used. One AP and one mobile phone were placed approximately 3 meters apart in an indoor environment. The mobile phone was slowly moved around within 10 cm in radius to receive fast-fade fluctuations. A histogram of the measurements is shown in Figure D.1 which is based on 807 measurements.

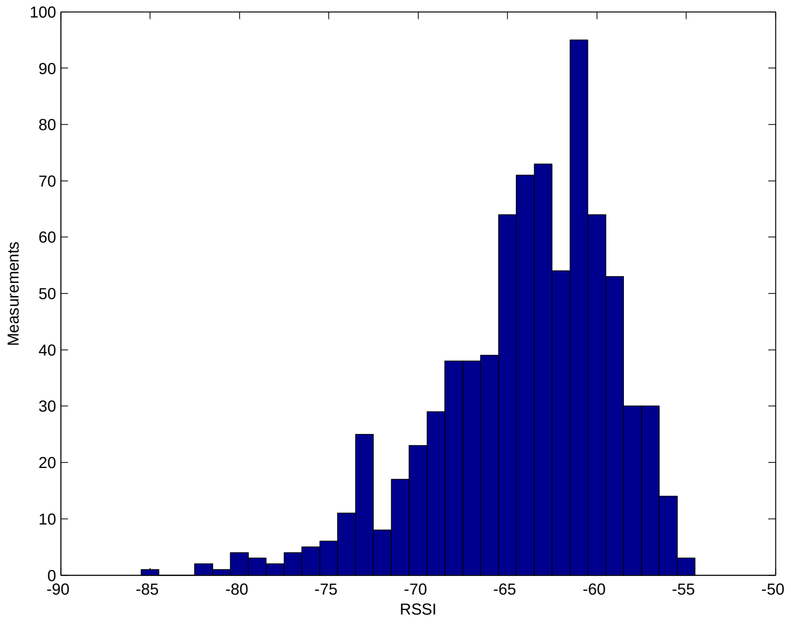


Figure D.1: A histogram of the taken measurements.

Since the exponential distribution is in linear scale, this distribution has to be converted into dBm units so it matches the RSSI values. After that, curve fitting can be done and the μ value can be found.

The fit is shown in Figure D.2 where μ is approximated to 4. The power value caused by path loss is estimated to -68 dBm based on the fitting. By using Equation D.1 the pass loss, caused by the distance of 3 meters, can be approximated to -77 dBm. The reason why these two values do not match, is likely a result of inaccurate α and β values that were experimentally determined in another environment as the multipath fading.

To include multipath fading in the emulator the *exprnd* function from Matlab is used. This function randomly draws numbers from a exponential distribution with a μ value of 4. Again, the drawn values have to be transformed into dBm which simply can be done by using $x = 10 \cdot log10(P)$ where x is in dBm and P is in milliwatts. This noise contribution is then added directly to the calculated path loss.

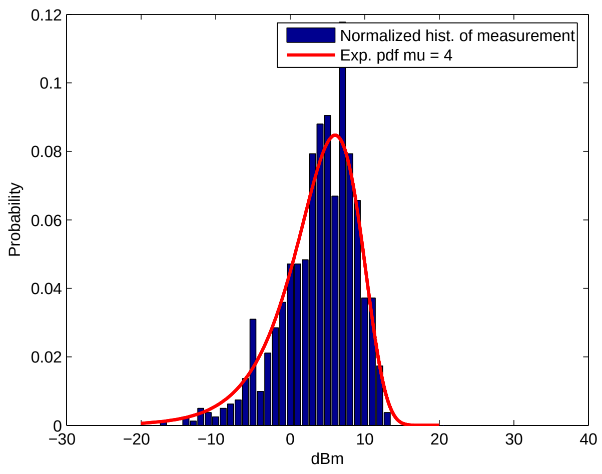


Figure D.2: Normalized measurements without the direct ray. A exponential pdf is approximated to the measurements.

Appendix E

SLAMWiN Core evaluation intermediate results

This appendix will document the intermediate results used to evaluate the effects of false positive and false negative jump detections. The results regarding false positive detections are described first, followed by the results from the false negative detections.

The purpose of this appedix is to document some of the results on which the evaluation of the jump detector is based.

Probability of false positive jump detections

To determine the probability for false positive jump detections the following small analysis is made based on measurements taken form the emulator. The measurements are taken from a scenario where a mobile user is moving in straight lines between point drawn randomly from a 10 times 10 meters square. The scenario with the MU path is illustrated in Figure E.1.

Approximately 20000 measurements are taken from five APs all placed in $(0, 0)$. Multiple AP are used in this analysis because the frequency of measurements that are taken from a given AP, depends on the total number of APs that are taking measurements. A MU will therefore have moved larger distances between measurements when the measurement frequency goes down which results in larger differences in the observed RSSI values.

The number of APs in a scenario will therefore have an influence on how good the jump detector performs. The average number of APs a MU is covered by is assumed to be 5. By applying the jump detector with different window sizes on the observed RSSI measurements and making histograms of the RSSI measurements, an evaluation of how combinations of the window size and the threshold in the jump detector perform is made. The histograms with different window sizes are illustrated in Figure E.2, E.3, E.4 and E.5 where a normal distribution is fitted to each histogram.

The normal distributions fitted to the histograms have a μ value equal to zero. The σ value for the fitted normal distributions with window size 2, 5, 10 and 15 is 5.57, 3.60, 2.67 and 2.35 respectively.

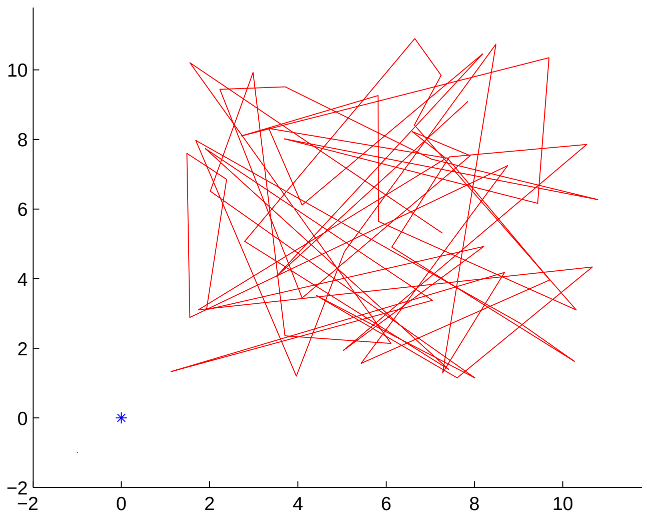


Figure E.1: Scenario used to determine the probability for false positive jump detections.

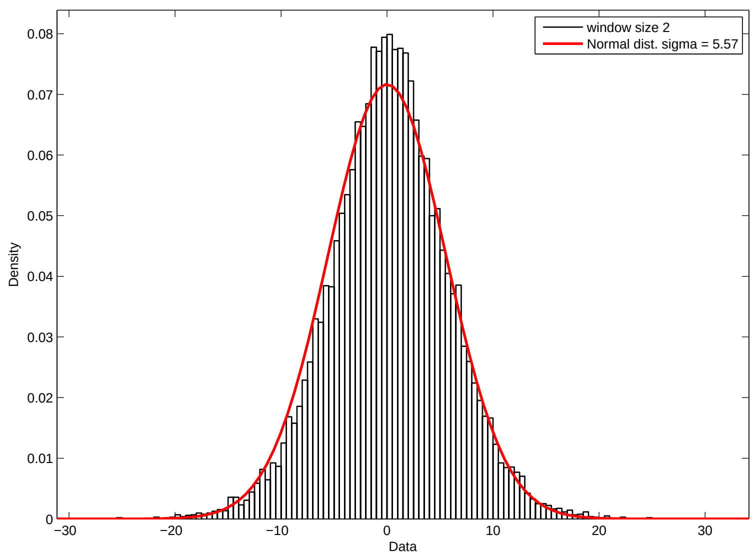


Figure E.2: Histogram of measurements taken from the jump detector with window sizes of 2. The fit is used to determine the probability of false positive jump detections. A normal distribution is fitted to the histogram.

Probability of false negative jump detections

To determine the probability for false negative jump detections the following small analysis is made based on measurements taken form the emulator. The measurements are taken

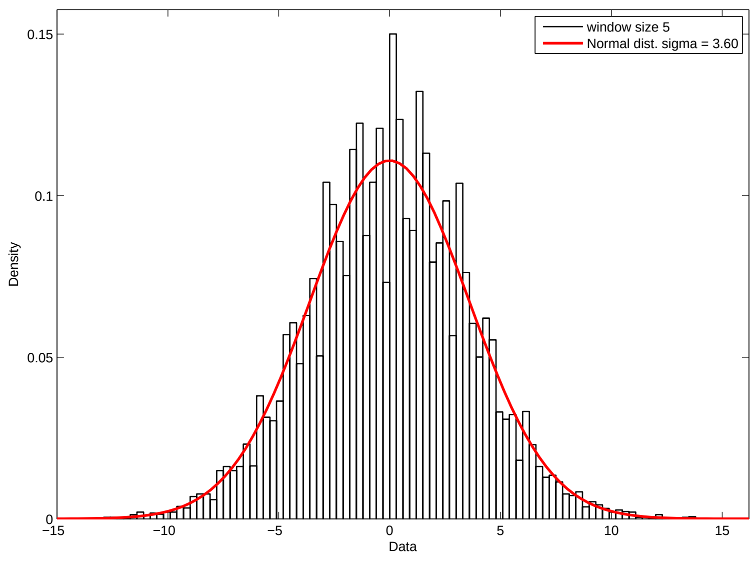


Figure E.3: Histogram of measurements taken from the jump detector with window sizes of 5. The fit is used to determine the probability of false positive jump detections. A normal distribution is fitted to the histogram.

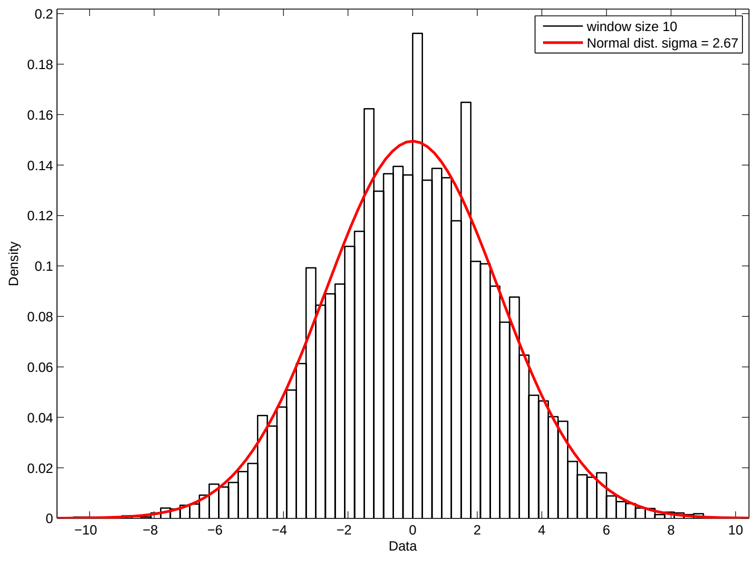


Figure E.4: Histogram of measurements taken from the jump detector with window sizes of 10. The fit is used to determine the probability of false positive jump detections. A normal distribution is fitted to the histogram.

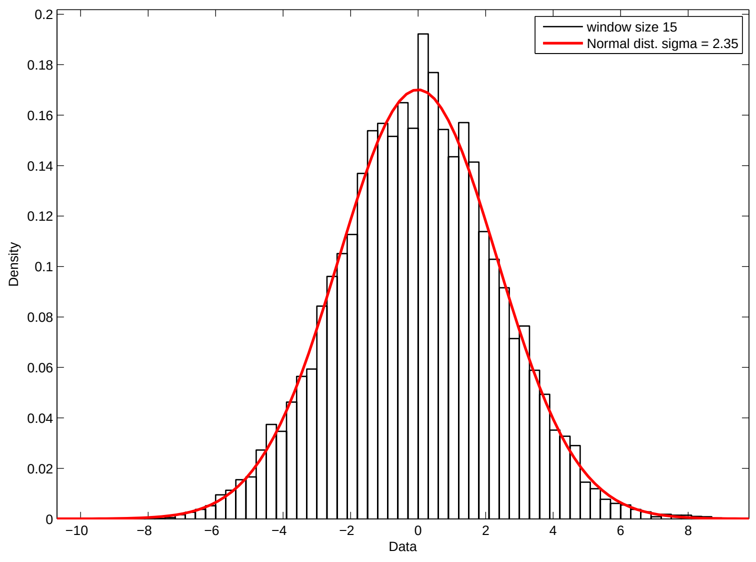


Figure E.5: Histogram of measurements taken from the jump detector with window sizes of 15. The fit is used to determine the probability of false positive jump detections. A normal distribution is fitted to the histogram.

from a scenario where a mobile user is moving from $(0, 0)$ to $(10, 0)$ and back. This is repeated 20 times. One wall is placed so the MU has LoS to the APs the first half way and non-Los the last half way. The scenario and the MU path are illustrated in Figure E.6. Again 5 APs are used as same reason as described in the previous subsection and are placed in $(5, 8)$.

By applying the jump detector with different window sizes on the observed RSSI measurements and making histograms of the sizes of all jumps caused by the wall, an evaluation of how combinations of the window size and the threshold in the jump detector perform is made. The histograms with different window sizes are illustrated in Figure E.7, E.8, E.9 and E.10 where a normal distribution is fitted to each histogram.

The σ value for the fitted normal distributions with window size 2, 5, 10 and 15 is 3.79, 2.79, 2.29 and 2.06 respectively and for μ the values are 16.92, 15.43, 15.16 and 15.00 respectively.

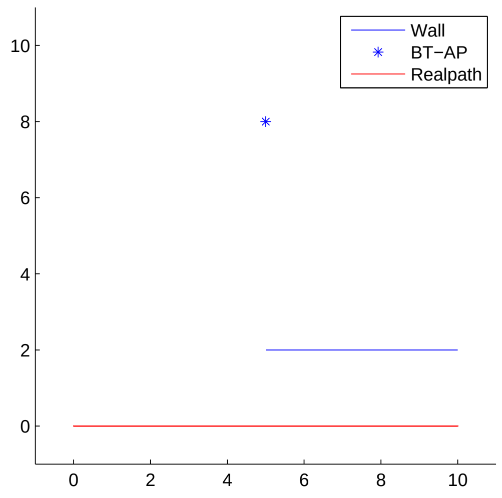


Figure E.6: Scenario used to determine the probability for false negative jump detections.

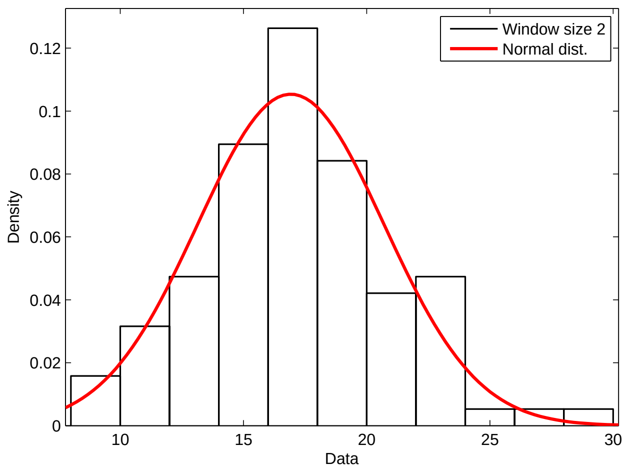


Figure E.7: Histogram of measurements taken from the jump detector with window sizes of 2. The fit is used to determine the probability of false negative jump detections. A normal distribution is fitted to the histogram.

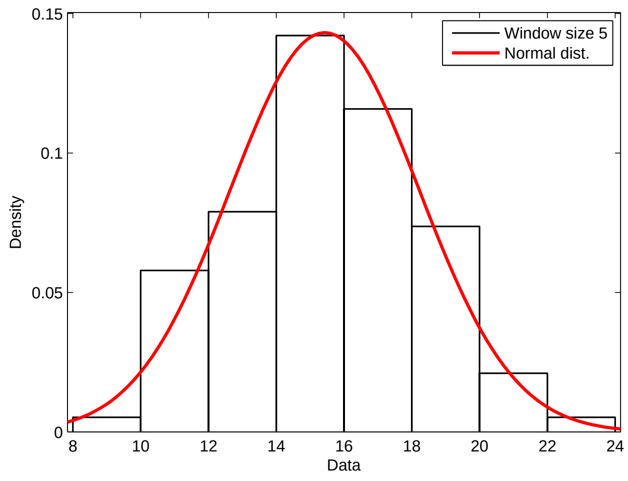


Figure E.8: Histogram of measurements taken from the jump detector with window sizes of 5. The fit is used to determine the probability of false negative jump detections. A normal distribution is fitted to the histogram.

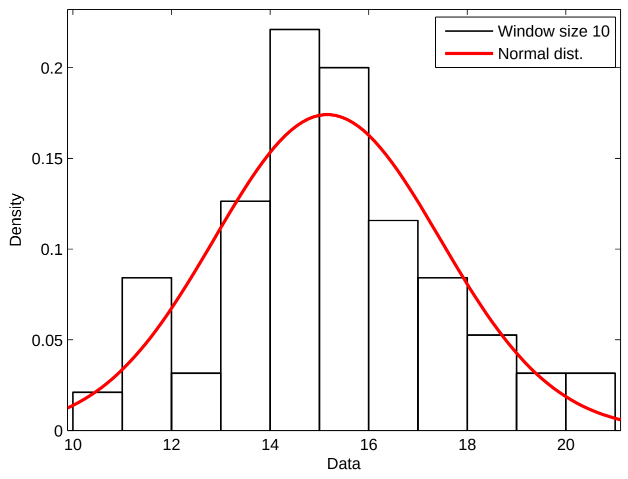


Figure E.9: Histogram of measurements taken from the jump detector with window sizes of 10. The fit is used to determine the probability of false negative jump detections. A normal distribution is fitted to the histogram.

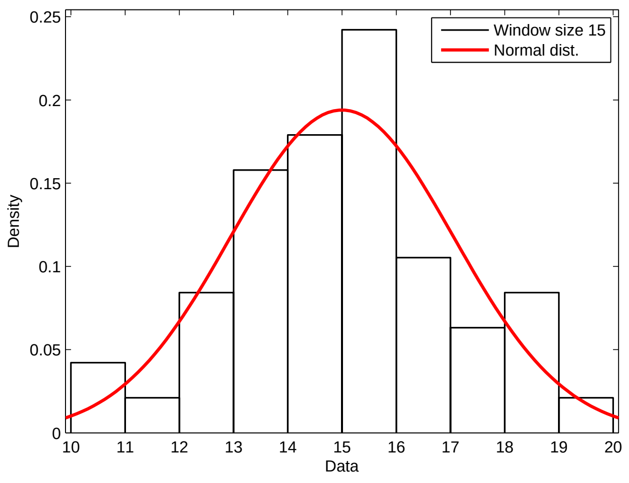


Figure E.10: Histogram of measurements taken from the jump detector with window sizes of 15. The fit is used to determine the probability of false negative jump detections. A normal distribution is fitted to the histogram.

Appendix F

Received Signal Strength Indication

RSSI is a measurement of the received radio signal strength. RSSI measurements can among other things be used for power level regulation on a transmitter node. By measuring the RSSI on the receiver side this information can be used to increase or decrease the output power level on the transmitter node. In that way the output power level can be adjusted so information can be transmitted without using superfluous signal power and thereby reducing energy consumption.

In Bluetooth the desired power level which satisfies the above described characteristic is known as the Golden Receiver Power Range. The Golden Receiver Power Range is defined by two thresholds. This is illustrated in Figure F.1. The lower threshold level corresponds to a received power between -56 dBm and 6 dB above the actual sensitivity of the receiver. The upper threshold level is 20 dB above the lower threshold level to an accuracy of ± 6 dB [sBS06].

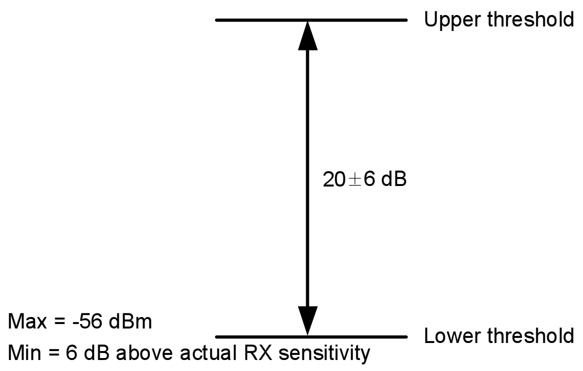


Figure F.1: Illustrated the Golden Receiver Power Range defined in the Bluetooth Specification.

Positive RSSI values indicates how many dB the RSSI is above the upper limit. Negative RSSI values indicates how many dB the RSSI is below the lower limit and if the RSSI value is zero this indicates that the RSSI is inside the Golden Receive Power Range. This can be expressed as follows:

$$\begin{aligned} S = RSSI + T_u, \qquad RSSI > 0 \\ S = RSSI - T_l, \qquad RSSI < 0 \\ T_u = T_l + 20db \end{aligned} \tag{F.1}$$

Where S is the received signal strength and T_u and T_l are the upper and lower thresholds respectively. How accurate these dB values will be measured depends on the Bluetooth hardware. According to the Bluetooth specification the only requirements to the Bluetooth hardware are that the device is able to tell whether the RSSI is inside, above or below the Golden Receiver Power Range. In Bluetooth the RSSI value can vary from -128 to 127 where the units are in dB.

Appendix G

Filter evaluation testbed

This appendix describes the testbed used for the filter evaluation scenarios. The description covers the used equipment, the test progress and some problems encountered during the tests.

G.1 Testbed setup

The filter evaluation testbed was established in a room of 11.70 × 8.70 meters with five APs placed at the following coordinates:

- AP 1: (1.7,1.7)
- AP 2: (1.7,7.7)
- AP 3: (9.7,7.7)
- AP 4: (9.7,1.7)
- AP 5: (5.7,1.7)

The APs were placed 1 meter above the ground on top of tables with the front panel facing downwards and the top panel facing towards the middle of the room. Further explanation on this is in the "Encountered problems" section later in this appendix. An overview of the room with APs locations is illustrated in Figure G.1.

The APs where connected to a laptop through Ethernet and the laptop was running a BlipNet Server, a BlipManager, DHCP server and a BlipNet Location Engine. A mobile phone was used as MU and was moved manually. The complete list of equipment used in the filter evaluation is as follows:

- 5 BlipNode L1 Bluetooth Class 1 Access Points.
- 6 Ethernet cables of varying length.
- 1 3com OfficeConnect 8 port Dual Speed Hub.

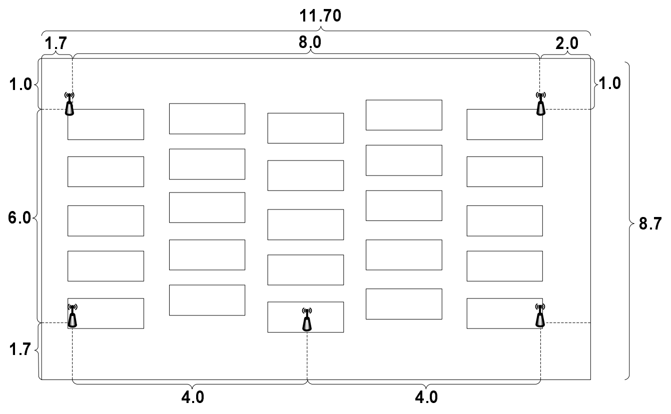


Figure G.1: This figure shows the room in which the filter evaluation scenario was performed.

- 1 BlipNet Server ver. 3.0 RC 1.
- 1 BlipManager ver. 4.0.
- 1 DHCP Turbo version 3.0.
- 1 BlipNet Location Engine version 4.0.
- 1 Laptop
- 1 Nokia 6270 mobile phone

G.2 Filter evaluation course

Each filter evaluation scenario was divided into 3 steps which are described here.

1. **Initialization:** First the filter evaluation scenarios were started with an initialization of the testbed. This included a reboot of all APs and a control period where the number of responses from the MU to inquiries from each APs where counted. The period was approximately 60 seconds and was needed to ensure a reasonable distribution of successful inquiries on all APs, which had proven vital for a good measurement results. Further explanation on this is in the ”Encountered problems” section later in this appendix.

2. **Data collection:** First the MU was placed at the start point defined in the evaluation scenario description. The MU was placed in the same height as the APs, approximately 1 meter above the ground, and was moved manually by one of the group members. There was paid special attention to that the mobile had LoS to all APs at all times during the data collection and that it was held with the backside of the mobile phone downwards.

Data collected during the evaluation scenarios was logged in a file containing information about the AP locations, successful measured inquiries and information about a selection of waypoints along the MU path. These waypoints were described with information regarding there location and the precise time they were passed by the MU. The log file had the following structure:

```
AP, AccessPointID, xpos, ypos, friendlyname
INQ, deviceID, Time, TimeStamp, AccessPointID, LastValue, CorrectedValue
REAL, deviceID, Time, TimeStamp, xpos, ypos
```

Most of the log file is self-explanatory. Those that are not are `LastValue`, which is the RSSI value measured by the AP, and `CorrectedValue`, which is the average RSSI value measured over the last 10 measurements by this specific AP.

The log recording was initialized simultaneously with the start of the MU movement. Since the MU was moved by a group member, its speed varied during the data collection. This was due to practical problems with the person moving the MU keeping out of the LoS to all APs and due to general difficulties keeping the exact same speed during the complete filter evaluation run. This should be considered when evaluating the collected data.

3. **Data treatment:** The last step was to treat the collected data. The data was first processed into the data structure used by the filter implementation, which is shown in the following:

```
track1 (struct)
    start (double)
    aps (array)
        ap (struct)
        addr (string)
        name (string)
        x_pos (double)
        y_pos (double)
        number (int)
    devices (array)
        device (struct)
        addr (string)
        number (int)
    measurements (array)
        measurement (struct)
        time (double)
        ap (int) - correspond to the "aps.number"
        device (int) - correspond to the "devices.number"
        rssi (double)
    real_parths (array)
        real_path (struct)
        time (double)
        device (string) - correspond to the "devices.number"
        x_pos (double)
        y_pos (double)
```

Then both filters were applied to the data and the results shown in the Filter chapter where found.

G.3 Encountered problems

During the filter evaluation scenario several problems where encountered. A description of each problem and how it was solved is described in the following.

AP antennas: Initially it had been expected that the antennas of the used BlipNodes could be assumed omni directional. However, measurements taken with the APs mounted on walls having the front panel facing to the center of the room provided unexpected measurements. This resulted in a short series of ad-hoc tests which lead to the conclusion that the AP antenna had a larger than average gain through the front panel and a smaller than average gain through the back panel. A examination of the AP hardware supported this conclusion as the AP antenna was mounted close to the front panel with the circuit board in the back.

Measurements of the antenna gain in several directions lead to that the APs where placed as described in the testbed setup section: on tables with the front panel facing downwards. This provided a close enough approximation of omni directional antennas and provided significantly netter results.

BlipNode registration on BlipManager: During the first step of the filter evaluation scenarios the APs where rebooted. This was done through the BlipManager. However, at different occasions the BlipNodes temporarily would not register on the BlipManager after having been rebooted or disconnected from the network.

Different measures were taken to get the AP to register including restart of the DHCP server, restart of the BlipNet Server, restart of the BlipNet Location Engine, repeated dis- and reconnection of the AP to the network and replacement of the resisting AP with another AP. No solution to this problem was found and the evaluation scenarios where interrupted until all AP would register. This happened after different time intervals and actions was needed to bring them back.

Inquery distribution on APs: Before the evaluation scenarios where performed some tests had been performed in order to get a feeling for the testbed. During these tests some unexpected characteristics of the data logs where discovered. In some of the log files certain APs had performed significantly more successful inquiries than others. In some cases three of the APs had made in the order of 300 successful inquiries where as the last two had made only 40. Since tracking of the MU based on such log files provided large location errors, this was investigated and the following observations where made.

1. The uneven distribution of successful inquiries did not occur on every restart of the APs.

2. In cases where the distribution was uneven typically two of the APs would make less successful inquiries than the others.
3. A restart of the all the ”slow” APs could in most cases provide a more equal distribution.
4. If the MU was placed at a point with approximately equal distance to all APs when they where restarted, there was a smaller chance of getting a uneven distribution.

Bullet 3 and 4 where used in the evaluation scenarios to avoid uneven distributions of successful inquiries.

Abbreviations

ACL	Asynchronous Connection Oriented
AFH	Adaptive Frequency Hopping
AoA	Angle of Arrival
AP	Access Point
BS	Base Station
EDR	Enhanced Data Rate
EKF	Extended Kalman Filter
EKF-JE	Extended Kalman Filter with Joint Estimation
FHS	Frequency Hop Synchronization
FHSS	Frequency Hop Spread Spectrum
GPS	Global Positioning System
HCI	Host to Controller Interface
HMM	Hidden Markov Model
ID	identity
ISM	Instrumentation, Scientific, and Medical
L2CAP	Logical Link and Control Adaption Protocol
LMP	Link Management Protocol
LOS	Line Of Sight
Mbps	Mega bit per second
MU	Mobile User

PDA	Personal Digital Assistant
PnP	Plug and Play
QoS	Quality of Service
ReBEL	Recursive Bayesian Estimation Library
RF	Radio Frequency
RSSI	Received Signal Strength Indicator
RTP	Real-Time Protocol
RWM	Random Waypoint Model
SCO	Synchronous Connection Oriented
SEKF	Switching Extended Kalman Filter
SLAM	Simultaneous Localization And Mapping
SLAMWiN	Simultaneous Localization And Mapping for Wireless Networks
TCP/IP	Transmission Control Protocol/Internet Protocol
TDD	Time Division Duplex
TDoA	Time Difference Of Arrival
TDMA	Time Division Multiple Access
ToA	Time Of Arrival
WLAN	IEEE 802.11 Wireless Local Area Network

Bibliography

[air] airespace, *Simplify wlan planning and deployment*, http://www.airespace.com/products/appnote_wlan_planning_design.php.

[AKW99] A.Y. Alfakih, A. Khandani, and H. Wolkowicz, *Solving Euclidean Distance Matrix Completion Problems Via Semidefinite Programming*, Computational Optimization and Applications **12** (1999), no. 1, 13–30.

[ARY95] JB Andersen, TS Rappaport, and S. Yoshida, *Propagation measurements and models for wireless communications channels*, Communications Magazine, IEEE **33** (1995), no. 1, 42–49.

[Bet01a] C. Bettstetter, *Mobility modeling in wireless networks: categorization, smooth movement, and border effects*, ACM SIGMOBILE Mobile Computing and Communications Review **5** (2001), no. 3, 55–66.

[Bet01b] ———, *Smooth is better than sharp: a random mobility model for simulation of wireless networks*, Proceedings of the 4th ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems (2001), 19–27.

[Bro86] R. Brooks, *A robust layered control system for a mobile robot*, Robotics and Automation, IEEE Journal of [legacy, pre-1988] **2** (1986), no. 1, 14–23.

[BSLK01] Y. Bar-Shalom, X.R. Li, and T. Kirubarajan, *Estimation with applications to tracking and navigation*, Wiley New York, 2001.

[BSPK06] Rusty O. Baldwin Brian S. Peterson and Jeffrey P. Kharoufeh, *Bluetooth inquiry time characterization and selection*, IEEE Transactions on Mobile Computing **5** (2006), no. 9, 1173–1187.

[BT02] H. Baltzakis and P. Trahanias, *Hybrid mobile robot localization using switching state-space models*, Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on **1** (2002).

[CFN] S.M. Chow, E. Ferrer, and J.R. Nesselroade, *An Unscented Kalman Filter Approach to the Estimation of Nonlinear Dynamical Systems Models.*

[Che95] Y. Cheng, *Mean shift, mode seeking, and clustering*, IEEE Transactions on Pattern Analysis and Machine Intelligence **17** (1995), no. 8, 790–799.

[CK97] F. Cozman and E. Krotkov, *Automatic mountain detection and pose estimation for teleoperation of lunar rovers*, Proc. IEEE Int. Conf. Robotics and Automation (1997), 2452–2457.

[DF98] Martin Daumer and Markus Falk, *On-line change-point detection for state space models using multi-process kalman filters*, Linear Algebra and its Applications **284** (1998), no. 1-3, 125–135.

[ea01] M.W.M.G. Dissanayaka et. al., *A solution to the simultaneous localisation and map building (slam) problem*, IEEE Transactions on Robotics and Automation **17** (2001), no. 3, 229–241.

[EEM05] Xiaoyan Li Eiman Elnahrawy and Richard P. Martin, *The limits of localization using signal strength: A comparative study*, Tech. report, Department of Computer Science, Rutgers University, January 2005.

[Fig04] João Figueiras, *Development and evaluation of mechanisms to obtain location information in bluetooth networks*, Tech. report, Aalborg University, June 2004.

[Gol05] A. Goldsmith, *Wireless Communications*, Cambridge University Press, 2005.

[HGPC99] X. Hong, M. Gerla, G. Pei, and C.C. Chiang, *A group mobility model for ad hoc wireless networks*, Proceedings of the 2nd ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems (1999), 53–60.

[JBJ98] David B. Johnson Yih-Chun Hu Josh Broch, David A. Maltz and Jorjeta Jetcheva, *A performance comparison of multi-hop wireless ad hoc network routing protocols*, Mobile Computing and Networking, 1998, pp. 85–97.

[JHB02] Barry Brumitt Jeffrey Hightower and Gaetano Borriello, *The location stack: A layered model for location in ubiquitous computing*, Tech. report, Aalborg University, June 2002.

[JLH$^+$99] P. Johansson, T. Larsson, N. Hedman, B. Mielczarek, and M. Degermark, *Scenario-based performance analysis of routing protocols for mobile ad-hoc networks*, Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking (1999), 195–206.

[Jør06] Thorbjørn Jørgensen, *Increasing wlan performance utilizing centralized management*, Tech. report, Aalborg University, June 2006.

[JZH98] B. Jabbari, Y. Zhou, and F. Hillier, *Random walk modeling of mobility in wireless networks*, Vehicular Technology Conference **1** (1998), no. 48, 639–643.

[Kal60] R.E. Kalman, *A New Approach to Linear Filtering and Prediction Problems*, Transactions of the ASME - Jornam of Basic Engineering **82** (1960), 35–45.

[KYT91] B.J. Kuipers and Byun Y-T., *A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations*, Journal on Robotics and Autonomous Systems (1991), no. 8, 47–63.

[LH03] B. Liang and Z.J. Haas, *Predictive distance-based mobility management for multidimensional PCS networks*, IEEE/ACM Transactions on Networking (TON) **11** (2003), no. 5, 718–732.

[Lib02] Mark Liberatore, *Background for mobility models*, http://signl.cs.umass.edu/˜liberato/wireless_background.html, 2002, General background for mobility models used in wireless networks.

[LV00] P. Lio and M. Vannucci, *Wavelet change-point prediction of transmembrane proteins*, Bioinformatics **16** (2000), no. 4, 376–382.

[nBS06] 2nd Bluetooth SIG, *The official bluetooth site*, http://www.bluetooth.com, 2006, The official homepage for information about Bluetooth standards and technical information.

[NTLL05] P. Nain, D. Towsley, B. Liu, and Z. Liu, *Properties of Random Direction Models*, Proceedings of IEEE Infocom (2005).

[Par92] J.D. Parsons, *The mobile radio propagation channel*, 1992.

[rBS04] 3rd Bluetooth SIG, *Specification of the bluetooth system*, Tech. report, Bluetooth SIG, November 2004.

[RSC90] M. Self R. Smith and P. Cheeseman, *Estimating uncertain spatial relationships in robotics*, Autonomous Robot Vehicles (1990), 167–193.

[RTS65] H.E. Rauch, F. Tung, and CT Striebel, *Maximum likelihood estimates of linear dynamic systems*, AIAA Journal **3** (1965), no. 8, 1445–1450.

[Sal01] D. Salmond, *Target tracking: introduction and Kalman tracking filters*, Target Tracking: Algorithms and Applications (Ref. No. 2001/174), IEEE (2001).

[SBM98] B. Southall, BF Buxton, and JA Marchant, *Controllability and Observability: Tools for Kalman Filter Design*, Proceedings. of British Machine Vision Conference BMVC **1** (1998), 164–173.

[sBS06] 1st Bluetooth SIG, *The official bluetooth membership site*, http://www.bluetooth.org, 2006, The official homepage for the members of the Bluetooth SIG.

[Sta01] EV Stansfield, *Introduction to Kalman Filters*, IEE Signal Processing, Kalman Filter Tutorial, March (2001).

[TC02] V. Davies T. Camp, J. Boleng, *A survey of mobility models for ad hoc network research*, Wireless Communication & Mobile Computing (WCMC): Special Issue on Mobile Ad Hoc Networking Research, Trends and Applications **2** (2002), no. 5, 483–502.

[TL00] S. Thiebaux and P. Lamb, *Combining kalman filtering and markov localization in network-like environments*, Pacific Rim International Conference on Artificial Intelligence (2000), 756–766.

[vdMW] Rudolph van der Merwe and Eric A. Wan, http://choosh.ece.ogi.edu/rebel/, ReBEL: Recursive Bayesian Estimation Library Toolkit for Matlab.

[WB01] Grag Welch and Gary Bishop, *Introduction to the kalman filter*, Tech. report, Department of Computer Science, University of North Carolina, February 2001.

[WvdMN00] Eric A. Wan, Rudolph van der Merwe, and Alex T. Nelson, *Dual estimation and the unscented transformation*, Advances in Neural Information Processing Systems 12 (S.A. Solla, T.K. Leen, and K.-R. Muller, eds.), MIT Press, November 2000, pp. 666–672.

[Ye05] Jason Yipin Ye, *Atlantis: Location based services with bluetooth*, Tech. report, Brown University, June 2005.

[Zha94] Z. Zhang, *Iterative point matching for registration of free-form curves and surfaces*, International Journal of Computer Vision **13** (1994), no. 2, 119–152.

[Zim80] Hubert Zimmermann, *Osi reference model - the iso model of architechture for open systems interconnection*, IEEE Transactions on Communications **28** (1980), no. 4, 425–432.