UTRECHT UNIVERSITY

Robot Localization and Kalman Filters

On finding your position in a noisy world

by

Rudy Negenborn

A thesis submitted to the
Institute of Information and Computing Sciences
in partial fulfillment of the requirements for the degree of
Master of Science, specialized in Intelligent Systems

Thesis number: INF/SCR-03-09

September 1, 2003

"Every day I watched closely for the sun or stars to appear, to correct my chronometer, on the accuracy of which our lives and the success of the journey would depend."

(F.A. Worsley, 1916 [53])

Abstract

The robot localization problem is a key problem in making truly autonomous robots. If a robot does not know where it is, it can be difficult to determine what to do next. In order to localize itself, a robot has access to relative and absolute measurements giving the robot feedback about its driving actions and the situation of the environment around the robot. Given this information, the robot has to determine its location as accurately as possible. What makes this difficult is the existence of uncertainty in both the driving and the sensing of the robot. The uncertain information needs to be combined in an optimal way.

The Kalman Filter is a technique from estimation theory that combines the information of different uncertain sources to obtain the values of variables of interest together with the uncertainty in these. The filter has been successfully applied in many applications, like missions to Mars, and automated missile guidance systems. Although the concept of the filter is relatively easy to comprehend, the advantages and shortcomings can only be understood well with knowledge of the pure basics and with experience.

In this work we provide a thorough discussion of the robot localization problem and Kalman Filter techniques. First, we look at current methods to obtain location information, pointing out advantages and disadvantages. We formalize how to combine this information in a probabilistic framework and discuss several currently used methods that implement it. Second, we look at the basic concepts involved in Kalman Filters and derive the equations of the basic filter and commonly used extensions. We create understanding of the workings, while discussing the differences between the extensions. Third, we discuss and experimentally show how Kalman Filters can be applied to the localization problem. We look at system and measurement models that are needed by the filter; that is, we model a driving system, a GPS-like sensor, and a landmark-based sensor. We perform simulations using these models in our own general Kalman Filter simulator showing different behaviors when applying the Kalman Filter to the localization problem. In order to use the landmark-based sensor when it can not uniquely identify landmarks, we extend the Kalman Filter to allow for multiple beliefs.

The material presented in this work forms a basis for further studies in localization literature, application of Kalman Filters in any domain, and in particular practical application of Kalman Filters and localization on physical robots.

Preface

This Thesis

This thesis is the result of the project that I performed in partial fulfillment for the degree of Master of Science, specialized in Intelligent Systems, to be acquired at the University of Utrecht, the Netherlands. I conducted the research involved in this project at the Datalogisk Institut, Københavns Universitet, Denmark, 2002-2003.

In this thesis I look at two large research fields. First, the field of robotic research, which is a relatively new field, dynamic and full of interesting challenges. The research in this field focuses at making robots more and more useful in practical settings. The potential applicability of robotic systems grows constantly, and with every new application, ideas for other applications arise. I investigate a key problem in making robots capable of doing their own thing, the problem of *Robot Localization*. The second field which I investigate in this thesis is the field of estimation theory. I explore how we can deal with the Robot Localization problem using a certain estimation technique, the technique of *Kalman Filters*.

Audience

The audience intended to read this thesis are graduate computer science students with interests in robotics and state estimation. A major part of the theory involved in this thesis builds on probabilities and matrix calculations and therefore a basic understanding of statistics and linear algebra is an advantage. In particular knowledge of the properties of Gaussian distributed variables makes understanding this thesis easier. Introductionary texts on these topics can be found in [32, 39, 22].

Online

At the website of this project, an HTML, pdf, and postscipt version of this thesis can be found. The first two of these include direct links to referenced articles and an interactive table of contents. Besides this, the website also contains the source codes of the used simulator, experimental settings, and the final presentation of the project. The website of the project can be found at

http://www.negenborn.net/kal_loc/.

Feel free to contact me through e-mail with any comments regarding this work at

rudy@negenborn.net.

Acknowledgments

The help and support of a number of people has contributed to the achievements of this project. First of all, I want to thank Prof. Dr. Phil. Peter Johansen for his hospitality and for offering me the opportunity to perform my research at the Department of Computer Science at the Copenhagen University. I want to thank him and Dr. Marco Wiering for supervising and supporting my work.

For proof-reading and commenting on preliminary drafs of this work I want to thank Michael Folkmann, Frederik Rønn, and Stine Søndergaard. For interesting discussions I also want to thank Jacob Blom Andersen and Anders Christian Kølle.

Finally I want to thank Dr. Michael Egmont-Petersen for the help in establishing the connection with Denmark.

Rudy Negenborn
Copenhagen, August 2003

Contents

Chapter 1

Introduction

Imagine yourself walking down a street. The street is filled with obstacles, like houses, trees, and other people. You want to go to the supermarket. With your eyes closed.

You will most probably find it rather difficult to find your way to the supermarket. Even if you know exactly which route to walk and even if there are no other people blocking the route.

Fortunately, most of us have eyes that help us in finding our way. At every movement we make, our eyes tell us where that movement brought us in the world. They constantly correct the imperfections of our movements.

You do not realize how inaccurate the movements that you make are, until you try to walk a straight line with eyes closed. Even though you think that the steps you make are steps on a straight line, it will probably quickly feel like you are moving away from the line. You will probably even start to loose your balance trying to stay on the line.

Walking down a virtual line with your eyes open in general does not cause many problems. From the information that you get from your eyes, and other senses, you constantly obtain an idea of where you are in the world. You use this idea to decide what to do next, for example to move towards the line if you are moving away from it.

Robots that move freely in the world have the same problems as humans when walking through the world. If they only move around not looking at where their actions take them, they will get lost because of imperfections in their moving mechanisms and the environment. Like humans, using eyes or other sensors, robots can detect these imperfections and get a better idea of where they are.

1.1 Autonomous Mobile Robots

In 1920 Karel Capek introduced the word *robot* in the English language in his play *R.U.R., Rossum's Universal Robots*[1][9]. In that play Capek presents a future in which all workers are automated, and how this leads to the ultimate revolt when the automated workers get souls.

Although much progress has been made in robot research since the days of Capek, robots are still far away from having souls. However, from *static*, *non-autonomous* robots in the beginning, robots now become more and more *mobile* and *autonomous*.

1.1.1 Mobility

The *mobility* of robots is the degree to which robots are able to freely move through the world. The first practical use of robots was in industrial settings with so-called *robot manipulators* [12]. These robots have the task to manufacture products like cars. They are programmed in such a way that they can repeat the same sequence of actions over and over again, faster, cheaper, and more accurate than humans. Typically they consist of a movable arm fixed to a point on the ground, which can assemble or manufacture different parts. Besides moving around this fixed point, the robots are not capable of moving freely and therefore they are not very mobile.

If we want robots that can do everyday tasks like cleaning the floor or delivering mail, they will have to be more mobile than the robot manipulators. That is, they have to be able to move around freely in a world while performing their tasks.

1.1.2 Autonomy

Autonomy of robots depends on to what extent a robot relies on prior knowledge or information from the environment to achieve its tasks. Robots can be divided into three classes of autonomy: *non-*, *semi-* and *fully-autonomous* robots [11].

- *Non-autonomous* robots are completely remotely steered by humans. The intelligence involved in these robots consists of interpreting the commands received from human controllers.

- *Semi-autonomous* robots can either navigate by themselves, or be steered by humans. Krotkov and colleagues [26] do research in how delayed commands from humans can be combined with autonomously made decisions for controlling robots that have to navigate on the Moon and on Mars. In dangerous situations the robot takes full control; in less dangerous situations humans can control the robot.

[1] Although his brother Josef invented it.

Another form of semi-autonomy is achieved by adjusting the area or by providing a map [16, 43] of the area in which the robot navigates. Engineers are required to adjust the environment such that the robot can safely move in that environment or to provide a map with which the robot can find its way by itself.

- *Fully-autonomous* robot vehicles are steered solely by the robots themselves. The robots do not require human interaction to fulfill their tasks. Fully autonomous robot vehicles are capable of intelligent motion and action, without requiring any external guidance to follow or control them [12].

Whether or not autonomy is desired depends on the situation. For robot manipulators in industrial settings it is not a problem being non-autonomous. In fact, a producer of products probably does not want his robots to do things on themselves, but rather sees them do exactly what he wants them to do. Besides this, non-autonomous robots that can perform the same sequence of actions over and over again are less expensive and more reliable than manpower.

On the other hand, when using robots to perform tasks of which we do not know each and every action to be taken, we do not want to specify each action. We rather want a robot to achieve some general goal that we specify, determining on itself how to get to this goal. A robot should therefore be semi- or fully-autonomous.

1.1.3 Application Areas

Whereas robots in the early years of robot usage were mainly used in industrial settings, the usage of robots is nowadays more and more seen in a wider perspective. A couple of examples should give an idea of the wide applicability of autonomous robots.

Autonomous robots can be used to explore environments that are *difficult for humans* to explore. Missions to planets and other places in space, or investigations of dangerous sites like radioactive environments are examples of these. The Lunar Rover Initiative [26] has the goal of making a trip to the Moon with private sponsoring. The sponsoring parties strive for public participation in space exploration. They want to let untrained people experience places in space by letting them control robots that are located at those places.

Another area in which autonomous robots can be used is underseas in the form of autonomous underwater vehicles. These vehicles can explore the sea surface while following reefs or pipes, and study marine creatures while chasing fish to get large amounts of new scientific data [36]. Also on land and in the air autonomous mobile robot applications can be found. Robots may for example perform *repairs and maintenance* remotely.

A fourth application area is in the *entertainment* sector. Robots can act as museum guides to attract people and show them around, both physically and through Internet [43].

Finally, a potentially interesting area where autonomous vehicles can be used is the *service* sector. Intelligent wheel chairs and vacuum cleaners can improve the life quality of many people. Medicin or food delivery robots can relief the workload of care workers.

In each of the mentioned application areas the robot requires autonomy and mobility. In order to achieve its tasks, it needs to move through some environment. This moving around is known as *robot navigation*.

1.2 Robot Navigation

1.2.1 Three Questions

Robot navigation is the task of an autonomous robot to move safely from one location to another. The general problem of navigation can be formulated in three questions [27],

- *Where am I?* The robot has to know where it is in order to make useful decisions. Finding out the whereabouts of the robot is called *robotic localization*.
- *Where am I going?* In order to fulfill some task the robot has to know where it is going. It has to identify a goal and this problem is therefore known as *goal recognition*.
- *How do I get there?* Once the robot knows where it is and where it has to go it has to decide on how to get there. Finding a way to get to the goal is known as *path planning*.

1.2.2 Difficulties

The navigation task is difficult due to a number of complex problems. Some issues that complicate navigation are limits on *computational power*, difficulties in *detecting and recognizing objects*, difficulties in *avoiding collisions* with objects, and the difficulties involved in *using information* provided by the environment [37].

1.2.2.1 Computational Power

The *computational power* that is needed to do for example real time image processing, computer vision, and learning is high. Although CPUs become faster and faster, they are still not fast enough to provide the power to perform those tasks on the fly. The chip technologies used today for processor

design will reach their limits. According to some, it will take years before new processing technologies become available [37].

1.2.2.2 Object and Landmark Recognition

Robots need to *recognize* structures like objects and landmarks in order to be able to perform their tasks. If these structures in an environment are not known in advance, image processing may need a lot more computational power than if the structures are known. Also, solutions that work in environments with known structures may not work if the structures are not known. Whether or not structures are known in advance or not, recognition of these can come with a significant amount of uncertainty.

1.2.2.3 Obstacle Avoidance

At all costs robots should *avoid* colliding with the obstacles in their environments. In dynamic environments, that is, where there are multiple moving obstacles, obstacle avoidance is a difficult problem. Although the dynamics of moving obstacles can sometimes be predicted, like when trains run on a track, sometimes the dynamics may be more uncertain, like playing children running around. A robot uses path planning techniques to plan a collision free path from one location to another location. If the obstacles in the environment are dynamic, the path planning problem is NP-hard [37].

1.2.2.4 Multi-Modal Sensor Fusion

The information that robots obtain using their sensors needs to be combined to determine what is going on in the environment. Different sensors on a robot can give different information about the environment. Different sensors can also give inconsistent or incorrect sensor readings. Sensor fusion is the problem of combining data from different sensors into one unified view of the environment.

Methods developed so far mainly solve the *uni-modal* sensor fusion problem, where the different sensors give information about a specific part of the state of the robot, like for example the location. There are no clear extensions yet to *multi-modal* sensor systems, where the different sensors give different kinds of information that in combination give information about the state of the robot, similar to the way humans reason with sensor information [37].

1.3 Errors and Uncertainty

A general factor that complicates some of the difficulties mentioned in the last section is the existence of noise and errors, in particular in the sensor

readings. To function, a robot has to get an idea of the world surrounding it. To be able to do this, it uses sensors like biological creatures. However, what the robot thinks it senses is not necessary what is actually the case. Uncertainties in object recognition, dynamic environments, and sensor readings make obtaining an accurate idea of the environment difficult. An example can point this out.

While navigating, a robot often uses *odometry* sensors to estimate its position [5]. These sensors count the number of revolutions that the wheels make while driving and turning. The readings can be used to help estimating the displacement over the floor to give an indication of the location of the robot. However, due to wheel slippage or other small noise sources [6], the odometer readings may give inaccurate results. Moreover, odometer readings are only based on the number of wheel revolutions and there is no way for the robot to see from these readings alone how accurate they are. The error in where the robot thinks it is will increase. Therefore, simply assuming that the odometer readings are correct can result in incorrect position estimates. Thus, the robot needs to have some notion of the error in the odometer readings. If it has some idea of the error, it can take this into account when finding out its location.

To keep the unbounded growth of the location error within certain bounds, the robot might use sensors that visually sense the environment. If a human wants to be sure about his location, he might look around for aspects of the environment that look familiar and that might confirm his location idea. The autonomous robot can do the same. It can look around in the environment using its sensors and try to use the information received from those to make its location more sure. It can use vision, compasses, active beacons, landmarks or perhaps GPS systems [5]. The information from the environment can be used to correct the error that was introduced by the odometry errors.

However, just like the odometry sensors, the sensors used to decrease the error in the position are also subject to errors. Reflections of laser beams, shadows in images, low level landmark detection algorithms, they all can cause errors in what the robot thinks is the case in the environment. Therefore, the robot also needs to have some notion of this uncertainty in order to be able to make a good estimate of its location.

This example shows the need for taking errors into account when dealing with robot navigation in the real world. Considering errors gives a more realistic view of the situation in the world and therefore has the potential of improved decision making and acting in that world. By looking at the localization problem from a *probabilistic* point of view, the notions of *uncertainty* and the *beliefs* of where a robot thinks it is can be modeled naturally.

1.4 Kalman Filters

The measurements a robot makes need to be combined to form an estimate of the location of the robot. The *Kalman Filter* (KF) is the best possible, optimal, estimator for a large class of systems with uncertainty and a very effective estimator for an even larger class [23, 21]. It is one of the most well-known and often-used tools for so called *stochastic state estimation* from noisy sensor measurements. Under certain assumptions, the KF is an *optimal*, *recursive data processing* or *filter* algorithm [21].

- The KF is *optimal*, because it can be shown that, under certain assumptions, the KF is optimal with respect to virtually any criterion that makes sense [21], for example the mean squared error. One of the reasons the filter performs optimally is because it uses all available information that it gets. It does not matter how accurate or precise the information is. It just uses all it gets to make an overall best estimate of a state, i.e., the values of the variables of interest. It does this by incorporating knowledge about the system dynamics, statistical descriptions of the system noises, measurement noise, uncertainty in the dynamics model, and any available information about initial conditions of the variables of interest.

- The KF is *recursive*, which brings the useful property that not all data needs to be kept in storage and re-processed every time when for example a new measurement arrives. Information gained in successive steps is all incorporated into the latest result.

- The KF is a *data processing algorithm* or *filter*, which is useful for the reason that only knowledge about system inputs and outputs is available for estimation purposes. Variables of interest can not be measured directly, but have to somehow be generated using the available data. A filter tries to obtain an optimal estimate of variables from data coming from a noisy environment.

For robotic purposes, the KF may be used to combine the information from sensors to estimate the state the robot is in, that is, the location the robot is at. The KF takes into account the different uncertainties and error sources that disturb the robot system and measurements.

1.5 Problem and Outline

In this thesis we want to make the reader familiar with the current research issues in robot localization and with the theory and practical use of Kalman Filters. The problem of robot localization is sometimes referred to as the most fundamental problem in making truly autonomous robots [11]. The

problem is difficult to solve, due to the existence of uncertainties in several areas. Kalman Filters are a classical approach to estimating states of noisy systems in noisy environments. In this thesis we investigate how we can use Kalman Filters in the robot localization problem.

Outline

The first chapters of this work thoroughly deal with the robot localization problem. In Chapter 2 we introduce the problem of robot localization and describe different instances of the problem. We also look at different sensing devices to give the reader concrete examples of where uncertainties play a role in a robotic context. We describe a framework that we can use to formalize the uncertainty and beliefs in the localization problem in Chapter 3. We approach the localization problem from a probabilistic point of view and look at different solutions to the localization problem that implement this framework.

Having covered the localization theory, we go into more detail on the Kalman Filter in Chapter 4. The chapter gives a thorough introduction of what the Kalman Filter is. We do this in a general way to give understanding of the Kalman Filter in its basic form to give readers interested in Kalman Filters a good basis to start from when wanting to apply the filter in any application domain. We describe the computational origins in great detail, while also paying attention to the general ideas and intuition behind the different equations. In Chapter 5 we describe some extensions to the standard Kalman Filter that make it wider applicable. From the theory of the Perturbation Kalman Filter we move on to the Extended Kalman Filter, finishing with a discussion of the Iterated Extended Kalman Filter.

The chapters following the localization and Kalman Filter theory have a more practical emphasis. In order to use the Kalman Filter as tool in the localization problem, the robot's system and measurements have to be modeled. In Chapter 6 we derive the models that we need to perform localization. From a practical point of view we show how to derive mathematical models that describe sensing and acting devices. We discuss ways to apply the Kalman Filter to the different localization problem instances in Chapter 7, where we also analyze the Kalman Filter using the derived models experimentally. We do this by means of a simulator in order to easily look at different scenarios. In Chapter 8 we continue our analyzing of Kalman Filters when we look at problems that can occur with some of the extensions, and when we extend the standard Kalman Filter framework to be wider applicable in localization problems.

In Chapter 9 we summarize the presented work with concluding remarks. Here, we also present ideas and possibilities for future research.

Chapter 2

Robot Localization

In this chapter we will take a look at robot localization. We will discuss the general problem and different instances of it in Section 2.1. We will look at the kind of information to which a robot has access for localization in Section 2.2. In Sections 2.3 and 2.4 we will discuss different techniques that provide a robot with that information. In Section 2.5 we look at how the information can be used for solving the robot localization problem.

2.1 Localization Problem

The problem of robot localization consists of answering the question *Where am I?* from a robot's point of view. This means the robot has to find out its *location* relative to the environment. When we talk about *location*, *pose*, or *position* we mean the x and y coordinates and heading direction of a robot in a global coordinate system.

The localization problem is an important problem. It is a key component in many successful autonomous robot systems [45]. If a robot does not know where it is relative to the environment, it is difficult to decide what to do. The robot will most likely need to have at least some idea of where it is to be able to operate and act successfully [5, 25]. By some authors the robot localization problem has been stated as the "most fundamental problem to providing robots truly autonomous capabilities" [11].

2.1.1 Problem Instances

The general localization problem has a number of increasingly difficult problem instances [45].

In the *position tracking* problem the robot knows its initial location. The goal of the localization is to keep track of the position while the robot is navigating through the environment. Techniques that solve this problem are called *tracking* or *local* techniques [19].

The *wake-up robot* or *global positioning* problem is more difficult than the position tracking problem, since the robot does not know its initial position. It has to localize itself from scratch. It hereby possibly needs to be able to deal with multiple ideas about its location. Methods that solve this problem are called *global* techniques [19].

An even harder problem to solve is the *kidnapped robot problem*. The robot does exactly know where it is localized, but all of a sudden it is transferred, or 'kidnapped', to another location without the robot being aware of this. The problem for the robot is to detect that it has been kidnapped and to find out what its new location is. Techniques that solve this problem can also be used to solve the wake-up robot problem. The wake-up robot problem is a special case of the kidnapped robot problem in which the robot is told that it has been kidnapped [19].

A factor that complicates each of these problems is the *dynamics* of the environment the robot is driving around in. Most localization research has been focused on performing localization in *static* environments. This means that the robot is the only moving object in the environment. Obviously this is not the case in the real world. *Dynamic* environments contain other moving objects and in these environments localization is significantly more difficult, since these other objects might confuse the robot about its location by corrupting the information used for localization.

2.2 Available Information

In determining its location, a robot has access to two kinds of information. First, it has *a-priori information* gathered by the robot itself or supplied by an external source in an *initialization* phase. Second, the robot gets information about the environment through every observation and action it makes during *navigation*.

2.2.1 A-priori Information

In general, the *a-priori information* supplied to the robot describes the environment where the robot is driving around. It specifies certain features that are time-invariant and thus can be used to determine a location. The a-priori information can come in different flavors. Examples of these are *maps* and *cause-effect relationships*.

2.2.1.1 Maps

The robot may have access to a map describing the environment. Such a map can be *geometric* or *topological* [37]. Geometric maps describe the environment in metric terms, much like normal road maps. Topological maps describe the environment in terms of characteristic features at specific

locations and ways to get from one location to another. A map can be learned by the robot in advance in an *exploration* phase, or it can be given by an external source in an *initialization* phase. A third option is that the robot learns the map of the environment while it is navigating through it, which is known as *SLAM*, *Simultaneous Localization and Mapping* [42].

2.2.1.2 Cause-effect relationships

Another way of supplying a-priori information to the robot is in terms of *cause-effect* relationships [37]. Given the input from the observations, these relationships tell the robot where it is. Possibly the robot can adjust these cause-effect relationships while navigating through the environment.

2.2.2 Navigational Information

The second type of information to which a robot has access is *navigational information*, i.e., information that the robot gathers from its sensors while navigating through the environment. A robot typically performs two alternating types of actions when navigating; it *drives* around or *acts* in the environment on one hand, and it *senses* the environment on the other. These two types of actions give rise to two different kinds of position information.

2.2.2.1 Driving

To be able to move around in an environment a robotic vehicle has a *guidance* or *driving* system [12]. A guidance system can consist of wheels, tracks or legs, in principle anything that makes the vehicle move around. These components are called *actuators*.

Obviously, the guidance system plays an important role in the physical position of a robot. The guidance system directly changes the location of the vehicle. Without a guidance system the robot is not driving around, which makes localizing itself a lot easier.

Assuming that a robot does have a guidance system, the way the guidance system changes the location contains valuable information in estimating the location. Knowing the effects of actions executed by the driving system gives a direct indication of the location of the vehicle after execution of these actions.

By monitoring what the driving system actually does using sensors, the displacement of the robot vehicle can be estimated. This results in *relative position measurements*, or also sometimes referred to as *proprioceptive measurements* [33]. Relative position measurements are measurements that are made by looking at the robot itself only. No external information is used and these measurements can therefore only supply information relative to the point where the measurements were started.

In Section 2.3 we take a closer look at different techniques that acquire relative position information and what the advantages and disadvantages of these techniques are.

2.2.2.2 Sensing

The robot *senses* the environment by means of its *sensors*. These sensors give momentary situation information, called *observations* or *measurements*. This information describes things about the environment of the robot at a certain moment.

Observations made from the environment provide information about the location of the robot that is independent of any previous location estimates. They provide *absolute position measurements*, also sometimes called *exteroceptive measurements* [33] to emphasize that the information of these measurements comes from looking at the environment instead of at the robot itself.

In Section 2.4, we discuss different kinds of techniques used in todays mobile robots to acquire absolute position measurements.

2.3 Relative Position Measurements

Acquiring relative measurements is also referred to as *dead reckoning*, which has been used for a long time, ever since people started traveling around [53, 1]. Originally, this is the process of estimating the position of an airplane or a ship, only based on the speed and direction of travel and the time that passed since the last known position [1]. Since the position estimates are based on earlier positions, the error in the estimates increases over time.

In robotic applications, relative position measurements are either acquired by *odometry* or *inertial navigation*.

2.3.1 Odometry

The word *odometry* comes from the Greek words for *road* and *measure* [30] and is the name of the most used technique for making a robot find out its position. Odometry works by integrating incremental information over time. By using *wheel encoders* to count the number of revolutions of each wheel, the robot measures the distance it traveled and its heading direction. Odometry is widely used, because it gives good short-term accuracy, is inexpensive, and allows for very high sampling rates [6, 4].

However, due to drift and slippage the integration of the wheel revolutions leads to errors in both traveled distance and orientation [4]. These errors accumulate over time. In particular errors in the orientation cause large positioning errors. Another disadvantage of odometry is its sensitivity

to terrain. If the surface the robot is driving on is not smooth, it can result in considerable position errors, since the odometry system cannot detect the irregularities in the terrain. Also differences in wheel diameter can cause position errors that are not detected by odometry measurements [35].

Although odometry causes increasing error in the location estimate, it is the most easy to access form of position information and therefore it is an important source of information for localization.

2.3.2 Inertial Navigation

Inertial navigation techniques use *gyroscopes* and *accelerometers* to measure the rate of rotation and acceleration of the robot [6]. *Gyroscopes*, or *rate gyros*, or simply *gyros*, detect small accelerations in orientation. Until recently, gyros were expensive with prices in the order of tens to hundreds of thousands dollar. With the introduction of fiber-optic and laser gyros, the prices have dropped considerably, allowing the use of gyroscopes on a larger scale in robotic applications [6].

Accelerometers measure small accelerations along the x or y axis of a robot vehicle. They suffer from extensive drift and are sensitive to bumpy ground, since if there is a bump in the floor, the sensor will detect a component of the gravitational acceleration. This problem can partially be solved by including a *tilt* sensor that can cancel the gravity component. However, the drift will still be high [4].

Like with odometry, position estimates from inertial navigation are acquired by integrating the obtained information from the sensors; once for obtaining the speed, twice for obtaining the traveled distance of the robot. The systems are independent of external information sources. However, since measurements are made by integration, the position estimates drift over time, and thus the errors increase without bound.

2.4 Absolute Position Measurements

Absolute position measurements supply information about the location of the robot independent of previous location estimates; the location is not derived from integrating a sequence of measurements, but directly from one measurement. This has the advantage that the error in the position does not grow unbounded, as is the case with relative position techniques. Absolute measurements can either supply the full location, or just a part of it, like for example the orientation.

The methods to obtain absolute measurements can be divided into methods based on the use of *landmarks* or *beacons* and methods based on the use of *maps*.

2.4.1 Landmark Based

One group of methods relies on the detection of *landmarks*. Landmarks are features in the environment that a robot can detect. Sensor readings from a robot are analyzed for the existence of landmarks in it. Once landmarks are detected, they are matched with a-priori known information of the environment to determine the position of the robot. Landmarks can be divided into *active* and *passive* landmarks.

2.4.1.1 Active Landmarks

Active landmarks, also called *beacons*, are landmarks that actively send out location information. Active landmarks can take on the form of satellites or other radio transmitting objects. A robot senses the signals sent out by the landmark to determine its position.

Two closely related methods are commonly used to determine the absolute position of the robot using active landmarks: *triangulation* and *trilateration* [37]. Triangulation techniques use distances and angles to three or more active landmarks; trilateration techniques only use distances. The angles and/or distances are then used to calculate the position and orientation of the robot.

The *GPS*, or *Global Positioning System* [14], uses trilateration techniques to determine latitude, longitude and elevation. It uses time of flight information from uniquely coded radio signals sent from satellites. Twenty-four satellites orbit the earth in 12 hours in six different orbital planes. Ground stations track the satellites and send them information about their position. On their turn, the satellites broadcast information back to the earth. The result gives a position accuracy between 100 and 150 meters [14].

To be able to use the mentioned methods, the robot needs to know the location of the landmarks in advance. Besides this, no a-priori information is required.

There are some problems with these techniques though. The transmitting of the active signals can be disturbed by atmospheric and geographic influences while going from sender to receiver [37]. These disturbances can be refractions and reflections, and will result in incorrect measurements. Another problem is that active landmarks in practice often cannot send out their signals in all directions, and thus cannot be seen from all places. Furthermore, active landmarks may be expensive to construct and maintain.

2.4.1.2 Passive Landmarks

If the landmarks do not actively transmit signals, the landmarks are called *passive landmarks*. The robot has to actively look for these landmarks to acquire position measurements.

Techniques using passive landmarks in determining the position of the robot rely on detection of those landmarks from sensor readings. The detection of landmarks depends on the type of sensor used. For example, in detecting landmarks in images from a vision system, image processing techniques are used. When three or more landmarks are detected by the robot, it can use the triangulation or trilateration techniques to compute its location.

Passive landmarks can be either *artificial* or *natural* and the choice of which kind of landmarks to use can play a significant role in the performance of the localization system.

Artificial Landmarks. *Artificial* landmarks are landmarks designed to be recognized by robots. They are placed at locations in the environment that are known in advance and that are well visible to the robot's sensors. Since these landmarks are specifically designed and placed in the environment for robot localization purpose only, a robot should not have problems determining its position unambiguously. Examples of passive artificial landmarks are bar codes, and colored geometric figures, like squares and circles.

Using artificial landmarks has a number of disadvantages. One of them is that the further away a robot is from a landmark, the less accurate the position estimation becomes [37]. This can for example be seen when detecting landmarks using vision systems. The further away a landmark is from a robot, the smaller the landmark will be in the images of the vision system and the more difficult it becomes to base an accurate position measurement on this landmark. If the distance is smaller, the position measurements will be more accurate.

Besides having less accurate measurements when landmarks are further away, measurements can be inaccurate, incorrect or even absent because of for example different lighting conditions or partial visibility [37]. Also, compared to active landmarks, detecting artificial landmarks takes a lot more processing power and the landmarks and environment have to be engineered to make the recognition work.

However, artificial landmarks can well be used when the environment is structured and does not change too much, like in office and factory environments. Outdoors however, the disadvantages might be too large to successfully use artificial landmarks. In that case, using natural landmarks seems more useful.

Natural Landmarks. *Natural* landmarks are landmarks that are not specifically engineered to be used as localization means for robots. Natural landmarks are already part of the environment of the robot. In indoor environments, examples of passive natural landmarks are doors, windows, and ceiling lights, whereas in outdoor environments roads, trees, and traffic signs are candidates.

Natural landmarks have the same disadvantages as artificial landmarks, except that the environment does not have to be engineered. Compared to artificial landmarks, the computational complexity of recognizing the natural landmarks is higher and the reliability of the recognition lower [37]. Especially when the landmarks are outdoors this can be a problem, since outdoor landmarks are often not as geometrically (lines, squares, etc.) shaped as indoor landmarks, which complicates the recognition.

Since the landmarks are not specifically engineered for localization purposes, the robot might need a number of observations before it can uniquely determine its position. For example, several doors in a row may look all the same and the robot needs a number of observations to find out in front of which door it is standing. This problem can be overcome by including more and more natural landmarks that uniquely determine a location, but this again will increase the computational complexity. A balance between the computational complexity and amount of natural landmarks that has to be used can be learned with neural networks. Learning landmarks increases the flexibility, optimality, and autonomy of the robot [40].

2.4.2 Map Based

Another group of localization techniques are *map based positioning* or *model matching* techniques. These approaches use geometric features of the environment to compute the location of the robot. Examples of geometric features are the lines that describe walls in hallways or offices. Sensor output, from for example sonars, is then matched with these features. Model matching can be used to update a global map in a dynamic environment, or to create a global map from different local maps [37].

The representation of maps can differ. It can either be *geometric* or *topological*. Geometric maps contain the environment in a global coordinate system. Topological maps contain the environment in the form of a network where nodes represent places in the environment and arcs between the nodes represent actions relating one location to another.

Using model matching techniques to determine the absolute position of a robot has the disadvantage that there needs to be enough sensor information to be matched with the map to come up with a position. Furthermore, techniques for matching sensor data with maps often require large amounts of processing power and sensing [37].

2.5 Multi-Sensor Fusion

Algorithms that solve the localization problem combine initial information and relative and absolute position measurements to form estimates of the location of the robot at a certain time. If the measurements are considered to be readings from different sensors, the problem becomes how to combine

the readings from different sensors to form a combined representation of the environment. This problem is studied by research in *multi-sensor fusion* [37, 28].

Fusion of information from multiple sensors is important, since combined information from multiple sensors can be more accurate. In particular when not all sensors are able to sense the same. Some features may be occluded for some sensors, while visible to others. Together the sensors can provide a more complete picture of a scene at a certain time. Multi-sensor fusion is also important since it can reduce the effects of errors in measurements.

Multi-sensor fusion methods can rely on a *probabilistic approach*, where notions of uncertainty and confidence are common terminology. In the following chapter we will describe a probabilistic framework for multi-sensor fusion in the robot localization problem. This framework is a general framework describing the probabilistic foundations of many existing, currently used, methods for solving the localization problem.

2.6 Summary

The robot localization problem is the problem of answering the question *Where am I?* from a robot's point of view. The general localization problem has increasingly difficult problem instances. In position tracking the robot knows its initial position, whereas in global positioning it does not. In the kidnapped robot problem, the robot knows where it is, but all of a sudden it is transported to a new location.

To perform localization, a robot has access to a-priori and navigational information. The a-priori information comes in the form of maps and cause-effect relationships that describe characteristics of the environment. Navigational information is the information the robot acquires while it is navigating in the environment. This information consists of relative and absolute measurements. The relative information provides high frequency, low cost, detailed information about the relative displacement of the robot, independent of features in the environment. However, due to slippage and drift of the robot, localization based only on relative information has an error that increases without bounds over time. The absolute information provides position measurements based on observations made from the environment. This position information is independent of previous position estimates. However, this comes at the price of higher computational costs, lower frequency and lower accuracy. Since the absolute measurements do not depend on previous position estimates, they do not suffer from unbounded error growth.

Using absolute position measurements alone can be done, but the disadvantages suggest that a combination of relative and absolute position measurements is better. The relative position measurements provide precise positioning information constantly, and at certain times absolute measurements

are made to correct the error in the relative measurements. Multi-sensor fusion provides techniques to combine information from different sensors in providing a best estimate of the robot's position.

Chapter 3

A Probabilistic Framework

In Chapter 2 we discussed different techniques to make relative and absolute position measurements. We discussed different sources of errors and found that there is not one best localization technique. We came to the conclusion that combining relative and absolute measurements is necessary.

In this chapter we focus on how the position measurements can be combined in a formal probabilistic framework. In Section 3.1 we describe the probabilistic view-point on robot localization, introduce the notion of belief, and formalize the acting and sensing of a robot in probabilistic models. We use these models in Section 3.2 to derive a general probabilistic formula for localization. We discuss problems that arise when implementing this formula in Section 3.3 and look at how several implementations deal with these problems in Section 3.4.

3.1 Probabilistic Localization

The general localization problem can be described as a *Bayesian estimation problem* [7]. We want to estimate the location of a robot given noisy measurements. If we look at the problem probabilistically, we can say that the robot has a *belief* about where it is. At any time, it does not consider one possible location, but the whole space of locations. Based on all available information, the robot can believe to be at a certain location to a certain degree. The localization problem consists of estimating the probability density over the space of all locations.

A Bayesian framework that estimates this density is the *Markov Localization* framework originally derived by Thrun and colleagues [19]. This framework captures the probabilistic foundations of many currently used localization methods. The Markov Localization framework combines information from multiple sensors in the form of relative and absolute measurements to form a combined belief in the location.

3.1.1 Beliefs

Belief. The robot has a *belief* about where it is. This is the probability density over all locations $x \in \Xi$, where Ξ is the set of all locations. We denote the belief by Bel. Localization can be seen as maintaining the belief,

$$Bel(x_k) = P(x_k|d_{0...k}). \tag{3.1}$$

That is, the probability that the robot is at location x_k at time k, given all information or data $d_{0...k}$ up to that time. This information also includes the a-priori information like for example a map of the environment. The location that given this probability distribution has the highest probability is the location at which the robot is most likely to be.

The goal of localization is to make this belief get as close as possible to the real distribution of the robot location. The real distribution of the robot location has a single peak at the true location and is zero everywhere else. If the robot achieves this goal, then it knows exactly where it is located.

Prior versus Posterior. During navigation, the robot has access to relative and absolute measurements. The robot incorporates these measurements into its belief to form a new belief about where it is. Commonly, a distinction is made between *prior* and *posterior* belief. The *prior* belief $Bel^-(x_k)$ is the belief the robot has after incorporating all information up to step k, including the latest relative measurement, but prior to incorporating the absolute measurement at step k. The *posterior* belief $Bel^+(x_k)$ is the belief the robot has after it has also included the latest absolute measurement in its belief.

States. In the following we will describe how we can compute the beliefs about the location of a robot. Although we will discuss this from a localization point of view, many of the ideas and assumptions are applicable to the more general problem of estimating a state of any kind. In fact, the framework that we will discuss is a specialization of the general Bayesian state estimation framework that can be used to estimate any state that can be observed by measurements.

3.1.2 Probabilistic Acting and Sensing

To be able to update the beliefs with the latest measurement information, we need to express measurement information in probabilistic terms. We need to define a probabilistic model for the acting, that is, the relative measurements, and a probabilistic model for the sensing, that is, the absolute measurements. We will use these two models in the next section to derive a probabilistic model for robot localization that computes the belief in the location after incorporating both types of information.

3.1.2.1 Acting

A robot performs actions in the environment that change the position of the robot. If we let action a_k from a set of possible actions A be the action performed by the robot at time k, we can express the way the location of the robot changes probabilistically by a transition density as [40, 19]

$$P(x_k|x_{k-1}, a_{k-1}). \tag{3.2}$$

This probability density gives the probability that if at time step $k-1$ the robot was at location x_{k-1} and performed action a_{k-1}, then it ended up at location x_k at time step k. In other words, the transition density describes how the actions of the robot change its location. This density is therefore called the *action* or *motion model*. Often the action model is time-invariant. We can then omit the time subscript k.

Notice how actions contain relative information about the new location of a robot. Given the last location, the robot can estimate its current location based on the performed action. Without the last location, the robot only knows it made a certain move; it is not able to label an absolute location to the resulting position.

In practice we can roughly approximate this transition density from the kinematics and dynamics of the robot. Another option is to have the robot learn the model itself [40].

3.1.2.2 Sensing

We can also describe the sensing of the robot in probabilistic terms. Let S be the space of all possible measurements coming from a sensor, and let s_k denote an element in S observed at time k. We can describe the probability that a sensor observes s_k from a certain location x_k at time k by the density [40, 19]

$$P(s_k|x_k). \tag{3.3}$$

This is called the *sensor* or *perceptual model*. As with the motion model, the perceptual model is often time-invariant. In that case we can omit the time subscript k.

Unlike the transition density of the acting of the robot, this probability density is difficult to compute. The reason for this is the sometimes high dimensionality of the measurements. Consider for example how complex the probability density is if the measurements come from a camera. The probability density will have to give a probability for each possible camera picture at each possible location, which would require a large amount of computing power.

Feature Vectors. To reduce the dimensionality, it is common to project raw sensor data s to a lower-dimensional *feature vector* z using a *feature extractor*, $\sigma : S \rightarrow Z$ [40]. The probability density from (3.3) is then taken over the space of feature vectors Z instead of raw sensor data S. For example, in landmark-based localization approaches a feature vector can contain the presence or absence of landmarks, based on the information from an image.

The obtained density does not relate pure sensor measurements to different locations in the environment, but it relates the feature vectors to different locations in the environment. This density can either be given or it can be learned from measurement examples. Most approaches assume that it is given [5][1].

3.2 Localization Formula

Suppose a robot wants to localize itself. The robot starts with an initial belief. At every time step, the robot performs an action that ends at the next time step. This action changes the location of the robot according to the transition density from (3.2). Besides this, the robot can also get information from sensing the environment. It perhaps extracts features from this sensor information to form a feature vector $z_k = \sigma(s_k)$, which is distributed according to the probability distribution from (3.3).

The robot now has to update its belief with the new information in order to get the best location estimate. We will in Section 3.2.1 describe how we represent the initial belief probabilistically, and then in Section 3.2.2 how we incorporate navigation information in the belief.

3.2.1 Initial Belief

Before the robot starts acting in the environment it has an *initial belief* of where it is. We model this belief by the prior belief at time step 0, $Bel^-(x_0)$.

If the robot knows where it initially is, then $Bel^-(x_0)$ is a distribution with a peak at the location where the robot knows it is. The goal of the localization becomes to compensate for slippage, drift and possible other noise sources to keep track of the location. In Section 2.1.1 we named this problem the *position tracking* problem.

In the case that the robot does not know where it starts, the initial belief $Bel^-(x_0)$ is a uniform distribution. The problem of localization is to

[1] In order to use a feature vector instead of a raw sensor reading to form the sensor model, the feature extractor has to be a sufficient statistic for estimating the location. This means that the probability density using the feature vectors should be the same as when using raw data. Otherwise, the result can be sub-optimal, because important sensor information is ignored. In practice sub-optimality of the density is tolerated, because of the computational advantages of feature vectors [40].

make the robot localize itself, not having any idea of where it is. In Section 2.1.1 we described this problem as the *wake-up robot* or *global localization* problem.

Finally, in the case that the robot thinks it is at a certain location, but it actually is not there, the initial belief is initialized with a peak at the location where the robot thinks it is. Since it is not actually located there, the robot has to detect this and adjust its belief. In Section 2.1.1 we called this the *kidnapped robot* problem.

3.2.2 Updating the Beliefs

Starting with the initial belief the robot starts querying its sensors and performing actions in the environment. The resulting measurements and actions have to be incorporated into the belief of the robot to give it the most up-to-date location estimate.

The belief the robot has after it has incorporated the action a_{k-1} executed at step $k-1$, and before it gets a new measurement z_k, is the prior belief,

$$Bel^{-}(x_k) = P(x_k|z_1, a_1, z_2, a_2, \ldots, z_{k-1}, a_{k-1}). \tag{3.4}$$

Once it has received an absolute measurement z_k at step k, it incorporates this measurement to obtain the posterior belief,

$$Bel^{+}(x_k) = P(x_k|z_1, a_1, z_2, a_2, \ldots, z_{k-1}, a_{k-1}, z_k). \tag{3.5}$$

The question now is how to compute these probability densities in an efficient way.

3.2.2.1 Incorporating Acting

Assume the robot has performed an action and wants to include the relative position measurement monitoring the result of this action into its belief. In equation (3.4) we defined the belief in which the latest action information is incorporated, the prior belief $Bel^{-}(x_k)$. By means of the *theorem of total probability* and the use of the *Markov assumption*, we can rewrite the original definition into a computationally efficient formula.

Total Probability. The theorem of total probability states that the probability of an outcome is equal to the sum of the probabilities of each of its dependent, partial, outcomes [49]. Using this theorem, we rewrite the definition of the prior belief (3.4) to

$$\begin{aligned} Bel^{-}(x_k) = \int_{\Xi} & P(x_k|x_{k-1}, z_1, a_1, \ldots, z_{k-1}, a_{k-1}) \\ & \times P(x_{k-1}|z_1, a_1, \ldots, z_{k-1}, a_{k-1}) dx_{k-1}. \end{aligned} \tag{3.6}$$

This equation expresses that the prior belief of being in state x_k is the sum of the probabilities of coming from state x_{k-1} to state x_k given all the earlier actions and measurements, $P(x_k|x_{k-1}, z_1, a_1, \ldots, z_{k-1}, a_{k-1})$, multiplied by the probability of actually being in state x_{k-1} given all the earlier measurements and actions, $P(x_{k-1}|z_1, a_1, \ldots, z_{k-1}, a_{k-1})$.

The second term of the integral in (3.6) is the probability of being at location x_{k-1} given all information up to step $k-1$; in particular the action performed at step $k-1$. However, the physical location of the robot at step $k-1$ does not depend on the action that is performed at that step. Therefore, we do not have to take a_{k-1} into account when expressing this probability. Using this and the definition of the posterior belief from (3.5), we rewrite (3.6) into

$$\begin{aligned} Bel^-(x_k) &= \int_\Xi P(x_k|x_{k-1}, z_1, a_1, \ldots, z_{k-1}, a_{k-1}) \\ &\qquad \times P(x_{k-1}|z_1, a_1, \ldots, z_{k-2}, a_{k-2}, z_{k-1}) dx_{k-1} \\ &= \int_\Xi P(x_k|x_{k-1}, z_1, a_1, \ldots, z_{k-1}, a_{k-1}) \\ &\qquad \times Bel^+(x_{k-1}) dx_{k-1}. \end{aligned} \tag{3.7}$$

Markov Assumption. To simplify the expression of the first term of the integral in (3.7) we make a *Markov assumption* [7], which states that given knowledge of the current state, the past is independent of the future, and vice-versa. With knowledge of the previous location x_{k-1}, it is of no importance how the robot ended up at that location or what it sensed. With this, we have that

$$P(x_k|x_{k-1}, z_1, \ldots, z_{k-1}, a_{k-1}) = P(x_k|x_{k-1}, a_{k-1}). \tag{3.8}$$

Notice that the right hand side of this equation is the conditional probability of being in state x_k given knowledge of the previous state and the performed action. We defined this as the action model in (3.2). By substituting the result into (3.7) we obtain an equation that can be used to efficiently incorporate the robot's actions into its belief,

$$Bel^-(x_k) = \int_\Xi P(x_k|x_{k-1}, a_{k-1}) Bel^+(x_{k-1}) dx_{k-1}. \tag{3.9}$$

That is, the prior belief of the robot in being at location x_k, after having finished the action of the previous time step $k-1$, is the result of integrating, over all last locations x_{k-1}, the probability that the performed action a_{k-1} brought the robot from location x_{k-1} to x_k times the posterior belief that the robot had in being at x_{k-1} at the last time step.

3.2.2.2 Incorporating Sensing

Assume that the robot has the prior belief $Bel^-(x_k)$, the belief in the location after it has performed its last action. The robot makes a measurement of the environment and extracts a feature vector z_k from this measurement. We want to incorporate this measurement into the prior belief to form the posterior belief as we defined in equation (3.5). With *Bayes' rule* and the *Markov assumption* we can rewrite this posterior belief into a computationally efficient form.

Bayes' Rule. *Bayes' rule* [31, 49] explains how the robot has to change its belief when a new measurements arrives. Using Bayes' rule and the definition of the prior belief from (3.4), we can rewrite (3.5),

$$\begin{aligned} Bel^+(x_k) &= \frac{P(z_k|x_k, z_1, a_1, \ldots, z_{k-1}, a_{k-1})P(x_k|z_1, a_1, \ldots, z_{k-1}, a_{k-1})}{P(z_k|z_1, a_1, \ldots, z_{k-1}, a_{k-1})} \\ &= \frac{P(z_k|x_k, z_1, a_1, \ldots, z_{k-1}, a_{k-1})Bel^-(x_k)}{P(z_k|z_1, a_1, \ldots, z_{k-1}, a_{k-1})} \end{aligned} \tag{3.10}$$

That is, the posterior belief is the conditional probability of observing z_k, $P(z_k|x_k, z_1, \ldots, a_{k-1})$, times the prior belief of being in state x_k, $Bel^-(x_k)$, divided by the probability of observing measurement z_k conditioned on all information so far, $P(z_k|z_1, \ldots, a_{k-1})$.

Markov Assumption. To make the computations of equation (3.10) less complex, we again make the Markov assumption. In this case we use it to state that a sensor reading only depends on the current state. The sensor reading is not influenced by previous locations of the robot. It does not matter how the robot got at the current location. The probability of observing a measurement is independent of the actions and observations that were made before the robot arrived in its current state. We use this assumption to rewrite the first term in the nominator of (3.10),

$$P(z_k|x_k, z_1, a_1, \ldots, z_{k-1}, a_{k-1}) = P(z_k|x_k). \tag{3.11}$$

We see that when we make the Markov assumption, the conditional probability of observing measurement z_k given the current state and past actions and observations reduces to the sensor model from (3.3). If we substitute this into (3.10), we obtain

$$Bel^+(x_k) = \frac{P(z_k|x_k)Bel^-(x_k)}{P(z_k|z_1, a_1, \ldots, z_{k-1}, a_{k-1})}. \tag{3.12}$$

The denominator of this equation is a normalizing constant ensuring that the probability density integrates to 1. This constant is calculated by integrating

the numerator over all possible locations x_k [31, 40],

$$P(z_k|z_1, a_1, \ldots, z_{k-1}, a_{k-1}) = \int_{\Xi} P(z_k|x_k) Bel^-(x_k) dx_k. \tag{3.13}$$

Equation (3.12) shows how we can express the posterior belief in terms of the prior belief. It also shows how we update the posterior belief to incorporate a new absolute measurement. It is a computationally efficient equation due to the use of the sensor model and the prior belief.

Localization Formula. We can combine the derived results into a single localization equation for the posterior belief in the location of a robot taking into account sensing and action information. Substituting equation (3.9) into equation (3.12), the posterior belief becomes

$$\begin{aligned} Bel^+(x_k) &= \frac{P(z_k|x_k) Bel^-(x_k)}{P(z_k|z_1, \ldots, a_{k-1})} \\ &= \frac{P(z_k|x_k) \int_{\Xi} P(x_k|x_{k-1}, a_{k-1}) Bel^+(x_{k-1}) dx_{k-1}}{P(z_k|z_1, \ldots, a_{k-1})} \\ &= \eta_k P(z_k|x_k) \int_{\Xi} P(x_k|x_{k-1}, a_{k-1}) Bel^+(x_{k-1}) dx_{k-1}, \end{aligned} \tag{3.14}$$

where η_k is the probability density normalizer $P(z_k|z_1, \ldots, a_{k-1})^{-1}$, calculated as in equation (3.13).

3.2.3 Parameters

To be able to use equation (3.14), we need to specify three probability distributions.

1. The *action model*, $P(x_k|x_{k-1}, a_{k-1})$, at which we looked in Section 3.1.2.1. It represents the mapping from locations and actions to new locations. It represents the probability of ending up at location x_k, given previous location x_{k-1} and given the action a_{k-1} taken from that location.

2. The *perceptual model*, $P(z_k|x_k)$, at which we looked in Section 3.1.2.2. It represents the probability that a certain observation or feature vector z_k is observed from location x_k. It maps locations to observations or feature vectors.

3. The belief the robot has at step zero, $Bel^-(x_0)$, also called the *initial belief* of the robot. We looked at the initial belief in Section 3.2.1.

With this, we have derived and described a probabilistic formula that recursively incorporates absolute and relative measurements into the belief of

a robot. The formula represents the probabilistic foundation of many currently used probabilistic localization methods. In Section 3.3 we will discuss the main issues in designing methods that use the derived formula (3.14). In Section 3.4 we will discuss how different methods deal with these issues.

3.3 Complexity Issues

In the design of methods that implement the localization formula (3.14), the complexity of the *location space representation* and the complexity of the *action and sensor models* are important issues [44].

3.3.1 Representational Complexity

Representational complexity is the complexity of the representation of the location space. Even with a small number of variables describing a location, the size of the whole location space can be extremely large. The size of the location space depends on the number of variables describing a location and the number of possible values that these variables can take on. Often the total number of possible locations is much larger than memory and hardware constraints permit; the belief needs to represent a probability for every possible value of every location variable. In fact, if the location contains continuous variables, then the size of the whole location space is infinite.

One way to deal with this issue is by making *conditional independence assumptions* [7]. These are assumptions about independence or conditional independence of aspects of a location. The *Markov assumption* that we saw in the previous section is one example of such an assumption. By making the Markov assumption future and past observations become conditionally independent if the current location is known. The Markov assumption implies that the location of the robot is the only thing that changes in the environment. Although this is not true in many environments, the Markov assumption has practically proven to be useful [41].

Probabilistic localization methods deal with representational complexity in different ways. They try to reduce the complexity of the location space on one hand, while keeping the strength of the probabilistic framework on the other [19]. We will see some examples of how methods deal with the representational complexity issue when we discuss a number of localization implementations in Section 3.4.

3.3.2 Modeling Complexity

Modeling complexity is the complexity of the action and sensor models. It can be a difficult task to make models that adequately describe real actuator and sensor systems. Especially when measurements are high-dimensional,

for instance when camera images are used, the complexity of the models can become very high.

One way to deal with high modeling complexity is to let the robot *learn* the action and sensor models. It can learn the consequences of its actions and it can learn a mapping from high-dimensional sensor data into a lower dimensional space. *Neural networks*, for example, can be used to learn how to extract landmarks from raw images [44].

In Section 3.4 we will see some examples of how existing implementations deal with the modeling complexity issue.

3.4 Implementations

Depending on the way the belief is represented, we can divide the methods that implement the derived framework into two groups: methods that represent the belief *discretely* and methods that represent the belief *continuously* [3]. These two groups of methods compute the beliefs differently.

3.4.1 Discrete Belief

A way of dealing with continuous location spaces is by *discretization* or *factorization* of the space [19]. When the location space is discretized, the integrals in equation (3.14) become sums and the belief over this space can be computed and stored explicitly. This way of representing the belief is captured by *Hidden Markov Models* [3]. These are general models in terms of transition and measurement probabilities. A number of methods has been developed using different representations for the discretization. *Topological graphs*, *grids*, and *particle filters* are examples of these.

3.4.1.1 Topological Graphs

The location space can be discretized into a coarse, *topological* representation with a limited set of robot actions, like for example four turning directions [10]. Topological structures store information in the form of a network, in which nodes represent locations and arcs between locations represent information on how to get from one location to another.

The topological representation supports multiple ideas about the location of the robot, since the representation supports multi-modal distributions. This may be necessary if the robot does not know where it is. It might then have to maintain multiple hypotheses about its location. For example, an environment might have three doors. If the robot does not know where it is and it sees one door, it still does not know exactly where it is; it can be in front of any one of the three doors. The topological representation supports this situation since each node in the topological graph can have its

own probability of corresponding to the true location. Therefore, this representation can be used to solve the *global localization* problem. However, the resolution of the representation in nodes and arcs is coarse and thus the accuracy of the localization estimates can be rough.

With the topological representation, the complexity of the sensor and action models reduces. The nodes of the network store only specific characteristics of the locations. The observation models supply these characteristics. The probabilities of observing the observations can be computed from ideal observations as specified by the topological structure and an error model [10]. The acquired probability distributions are not over the raw data, but over the computed features. This has as disadvantage that if there are no features to extract in the raw sensor data, then the robot cannot improve its belief. The action model can be obtained from the connectivity of the topological structure, since this structure connects locations with each other with certain probabilities [10].

3.4.1.2 Grids

Instead of having a topological network, the location space can be represented by a fine-grained, regularly spaced *grid* with fixed spatial and angular resolution, for example between 10 and 40 centimeters as x and y resolution, and between 2 and 5 degrees as orientation resolution [8]. Since the representation is more fine-grained than topological approaches, grid maps in general offer more accurate position estimates. As with the topological representation, multi-modal distributions are supported, since probabilities are assigned with every cell in the grid. However, the accuracy of the representation comes at computational costs increasing with finer resolution. These costs can be reduced by using *selective updating* strategies [19]. These strategies have as consequence that locations that are likely to be the true location get more attention than locations that are less likely to be the true location.

Selective updating can be implemented by *thresholded updating* [19]. That is, only the belief in those states x for which $Bel(x) > \epsilon$, where ϵ if some threshold, is updated. Another way of selective updating is to update the belief of a *fixed fraction* of all locations, chosen according to the latest belief.

With selective updating arises a problem when a location that is very unlikely to be the true location is in fact the true location. If this is ignored completely, then the belief will not adequately represent the true location. An algorithm that uses selective updating should be aware of this and be able to recover when the true location is not in the belief anymore.

Since no specific features are stored in the grid, the measurement model does not have to extract any features from sensor data. Raw sensor data can be used instead to update the grid [40]. This has the advantage that

the localization also works in environments where no identifiable features exist. However, the complexity of the sensor models increases, because a probability has to be maintained for every raw sensor measurement.

3.4.1.3 Particle Filters

Yet another way of discretization of the location space is by representing the belief by a set of m weighted samples distributed according to the belief [20]. Each sample is a location and the weights are non-negative factors called *importance factors*, which sum up to one. These factors indicate how important each sample is. The importance factor is determined by the likelihood of that sample given the latest absolute observation.

This recent technique is used in *particle filter* methods. A particle filter implementation for the localization problem is *Monte Carlo Localization* [20]. Localization results achieved with this technique have shown to be more efficient and accurate than the earlier discussed methods. The technique does not rely on a grid representation and therefore does not suffer from the inherent computational costs. By adjusting the number of samples taken from the belief distribution online, the balance between accuracy and computational costs can be adjusted. Particle filters are the interest of current state-of-the-art localization research.

3.4.2 Continuous Belief

Instead of using a discrete representation to deal with continuous spaces, we can maintaining a probability density *function* over the locations. There is not a probability stored for every location; instead only the parameters of the probability function that describes the probability density are stored.

In order to keep the computations involved in calculating the probabilities analytically low, the assumption can be made that the probability densities of the belief and models are *Gaussian* distributed. A Gaussian density is fully characterized by two parameters, the *mean* and the *variance*. The *mean* gives the weighted average over the probabilities; the *variance* gives the uncertainty. The larger the variance is, the more uncertainty there is in the mean being the true average.

3.4.2.1 Kalman Filters

A *Kalman Filter* (KF) is a mathematical tool to estimate the state of a *noisy dynamic system* using noisy measurements related to the state [3]. KFs assume that the action and sensor models are subject to Gaussian noise, and assume that the belief can be represented by one Gaussian function. In practice, this might not always be the case, but it does allow the KF to efficiently make its calculations [19].

With the Gaussian noise assumption, the KF can be used to propagate the belief through time and incorporate information from measurements. If the dynamic system can be described using linear equations, then the KF is an optimal state estimator for virtually any kind of meaningful criterion of optimality [21].

There is a disadvantage of choosing the Gaussian representation for the belief. Choosing this representation is a restrictive way of representing the location space. A Gaussian is a *uni-modal* density, i.e., it has only one peak. Thus, it does not allow multiple ideas about locations. There is only one best location estimate, corresponding to the mean of the belief. Therefore, representing the state space with a Gaussian can only be done when the initial location of the robot is known. The representation can only be used to solve the *position tracking* problem. For the other localization problems ways to get around the Gaussian belief have to be found.

3.5 Summary

We can approach the robot localization problem from a probabilistic point of view. By looking at the problem probabilistically, the concepts of uncertainty and probabilities automatically come into the picture. Instead of being sure of one location, the robot considers the whole space of locations to be possible locations. It has a belief over the location space.

A robot starts with an initial belief. This belief can be a uniform distribution if the robot has no idea where it is, or it can be a distribution with one peak at the right location if the robot knows, or thinks it knows, where it is.

While navigating through the environment the robot gets position information in the form of action and sensor measurements corresponding to relative and absolute measurements respectively. If the probability distributions of these measurements are known, then they can be incorporated into the location belief of the robot. We refer to the belief resulting from incorporating a relative measurement as prior belief, and to the belief resulting from incorporating an absolute measurement as posterior belief.

By means of the Bayes' rule and the Markov assumption, we derive a formula that recursively computes the posterior belief given relative and absolute measurements,

$$Bel^{+}(x_k) = \frac{P(z_k|x_k) \int_{\Xi} P(x_k|x_{k-1}, a_{k-1}) Bel^{+}(x_{k-1}) dx_{k-1}}{P(z_k|z_1, a_1, \ldots, z_{k-1}, a_{k-1})}.$$

To implement the derived formula, we need to specify three probability densities: the action model, the perceptual model, and the initial belief. Implementing the localization formula is not straightforward. The complexity of the location space representation and the complexity of the action and sensor models play important roles.

Different localization methods deal with these two complexity issues in different ways. By representing the location space continuously as a parameterized probability density, like a Gaussian as in Kalman Filters, the computations involved are greatly reduced. However, unimodal distributions with one peak, like Gaussians, do not allow more than one hypothesis about the location. By discretization of the location space and representing it as a topological or grid map, distributions with multiple peaks are possible. Grid-based approaches are more accurate than topological based, but they come at higher computational costs. State-of-the-art particle filters are accurate and efficient by using a selected set of weighted samples as belief representation.

To deal with the complexity in the sensor and action models, the models can be assumed to be Gaussian distributed and linearly related to the location. However, in practice this might not always be the case. By representing the sensor models as probability distributions over extracted features, the complexity can be decreased significantly. However, if no features are present, then that can be a problem. By using raw sensor data the data is always useful, but the complexity of the sensor model is high. In those cases it might be useful to let the robot learn the sensor model itself.

Chapter 4

Kalman Filters

As discussed in Chapter 3, the representation of the belief has a significant effect on the computational efficiency in calculating the belief. One way to deal with the computational complexity of beliefs over continuous spaces is by representing the belief as a parameterized continuous function. The Kalman Filter (KF) is a means to calculate beliefs that are represented by Gaussians. In this chapter we thoroughly investigate how KFs work and how they use Gaussian functions to update beliefs.

We start our KF discussion in Section 4.1 with a quick overview of what a KF is, how it is used, and where it comes from. In Section 4.2 we show by example the basic concepts involved in the KF. In Section 4.3 we go into more detail when we more formally state what kind of problem the KF solves. We discuss the assumptions made by the KF in Section 4.4, and prepare the derivation of the KF equations using these assumptions in Section 4.5. We are then ready to formally derive the KF equations in Section 4.6, after which we in Section 4.7 present the resulting algorithm as the Linear Kalman Filter and look at the meaning of the equations. We proof that the KF is a so called *minimum variance estimator* in Section 4.8 and make the KF equations that we have derived more generally applicable in Section 4.9.

4.1 Kalman Filters

In short, a *Kalman Filter* [23, 29, 21, 15, 38, 50] is a recursive data processing algorithm that estimates the *state* of a *noisy linear dynamic system*.

When we talk about the *state* of a system we mean a vector x consisting of n variables describing some interesting properties of the system. An example of a state is the location of a robot, consisting of the x and y coordinates, and the orientation ϕ of a robot.

The fact that the variables of the state might be noisy and not directly observable makes the state estimation difficult. To estimate the state a KF

has access to *measurements* of the system. Those measurements are *linearly related* to the state and are corrupted by noise. If these noise sources are *Gaussian distributed*, then the KF estimator is statistically optimal with respect to any reasonable measure for optimality [21].

The KF processes all available measurements to estimate the state, both accurate and inaccurate measurements. It uses knowledge of the system and sensor dynamics, probabilistic descriptions of the system and measurement noises, and any available data about the initial values of the state.

The methodology and terms used in the KF have their origins in many different disciplines [21]. Least squares, least mean squares, probability theory, dynamic systems, stochastic systems, and mathematical foundations like linear algebra all have had their influence on the design of the KF. On itself, the KF is part of the foundations of disciplines like modern control theory and statistical decision theory.

Some authors have called the discovery of the KF one of the greatest discoveries in the history of statistical estimation theory, and possibly the greatest in the twentieth century [21]. The KF has been used in many application areas ever since Richard Kalman discovered the idea in 1960 [23]. The KF has made it possible for humans to do things that would not have been possible without it. In modern technology, KFs have become as indispensable as silicon chips [29].

4.1.1 Applications

The KF has been used in a wide range of applications. *Control* and *prediction* of dynamic systems are the main areas [21].

When a KF *controls* a dynamic system, it is used for state estimation. When controlling a system, it is important to know what goes on in the system. In complex systems it is not always possible to measure every variable that is needed for controlling the system. A KF provides the information that can not directly be measured by estimating the values of these variables from indirect and noisy measurements. A KF can for example be used to control continuous manufacturing processes, aircrafts, ships, spacecrafts, and robots [3, 12, 13, 21, 24, 29, 48, 41].

When KFs are used as *predictors*, they *predict* the future of dynamic systems that are difficult or impossible for people to control. Examples of these systems are the flow of rivers during flood, trajectories of celestial bodies, and prices of traded goods [29, 21].

4.2 Example

Before getting started with the formal derivation of the KF equations, it is beneficial to get a first idea of how the KF works. Suppose that a robot navigates around in an environment and wants to localize itself. As we have

seen, a robot is subject to sources of noise when it drives around. To estimate its location we assume that the robot has access to absolute measurements.

Models. In order to say someting about the location of the robot, we model the system of a navigating robot. That is, we model how the location of the robot changes over time. We assume for simplicity that the robot drives at constant speed s, and that we have a *system model* that describes the true location x_k of the robot over time,

$$x_k = x_{k-1} + s + w_k, \tag{4.1}$$

where the new location x_k depends on the previous location x_{k-1}, the constant speed per time step s, and a noise term w_k. We assume that the noise is *zero-mean random* noise, and moreover Gaussian distributed. This means that on average the noise is zero, but sometimes somewhat higher, and sometimes somewhat lower. Let us denote the deviation in the noise by σ_w.

If we want to use absolute measurements in estimating the location, we need to describe how these measurements are related to the location. We assume that we have a *measurement model* that describes how measurements z_k depend on the location x_k of the robot,

$$z_k = x_k + v_k, \tag{4.2}$$

where the sensor in this case gives measurement z_k of the location of the robot x_k, corrupted by measurement noise v_k. We assume that the noise is zero on average, Gaussian distributed, and that it has a deviation of σ_v.

Initialization. Assume that we have an *initial estimate* of the location of the robot $\hat{x}_0$ and that we have an uncertainty, that is, variance, of σ_0^2 that this is the true location.

Prediction. Assume that the robot drives for one time step. From the system model we know that the location will on average change with about s. We can update the estimate of the location with this knowledge. That is, with the system model we can *predict* what the location of the robot most likely is after one step. We calculate the new location $\hat{x}_1$ at step $k = 1$ as

$$\hat{x}_1 = \hat{x}_0 + s + 0. \tag{4.3}$$

We hereby took the noise component in the system equation as zero. Although from equation (4.1) we know that the state is corrupted by noise, we do not know the exact amount of noise at a certain time. Since we know that the noise on average is zero, we used $w_k = 0$ in calculating the new location estimate.

We do also know how the noise varies around zero. We can use this to update the uncertainty in the new estimate. We calculate the uncertainty σ_1^2 that we have in our new estimate as

$$\sigma_1^2 = \sigma_0^2 + \sigma_w^2. \tag{4.4}$$

Correction. If the robot keeps on driving without getting any absolute measurements, the uncertainty in the location given by equation (4.4) will increase more and more. If we do make an absolute measurement, we can update the belief in the location and reduce the uncertainty in it. That is, we can use the measurement to *correct* the prediction that we made.

Assume that we make an absolute measurement z_1. We want to combine this measurement into our estimate of the location. We include this measurement in the new location estimate using a weighted average between the uncertainty in the observed location from the measurement z_1 and the uncertainty in the estimate that we already had $\hat{x}_1$,

$$\begin{aligned}\hat{x}_1^+ &= \frac{\sigma_v^2}{\sigma_1^2 + \sigma_v^2}\hat{x}_1 + \frac{\sigma_1^2}{\sigma_1^2 + \sigma_v^2}z_1 \\ &= \hat{x}_1 + \frac{\sigma_1^2}{\sigma_1^2 + \sigma_v^2}(z_1 - \hat{x}_1).\end{aligned} \tag{4.5}$$

This way of incorporating the measurement has as consequence that if there is relatively much uncertainty σ_1^2 in the old location estimate, that we then include much of the measurement. On the other hand, if there is relatively much uncertainty σ_v^2 in the measurement, then we will not include much of it.

Absolute measurements do not depend on earlier location estimates; they provide independent location information. Therefore they decrease the uncertainty in the location estimate. Realize that probabilities represent populations of samples in a way like mass represents populations of molecules. With this, we notice that we can combine the uncertainty in the old location estimate with the uncertainty in the measurement in the same way as we can combine *moments of inertia*, measures of the distribution of the mass in objects [13]. This gives us the uncertainty $\sigma_1^{2,+}$ in the new location estimate as

$$\frac{1}{\sigma_1^{2,+}} = \frac{1}{\sigma_1^2} + \frac{1}{\sigma_v^2},$$

which we can rewrite into

$$\sigma_1^{2,+} = \sigma_1^2 - \frac{\sigma_1^2}{\sigma_1^2 + \sigma_v^2}\sigma_1^2. \tag{4.6}$$

Notice in this equation that incorporating new information always results in lower uncertainty in the resulting estimate. The uncertainty $\sigma_1^{2,+}$ is smaller

than or equal to both the uncertainty in the old location estimate σ_1^2 and the uncertainty in the measurement σ_v^2. Notice also that we in (4.5) and (4.6) use the same weighting factor. We introduce a factor K representing this weighting factor and rewrite (4.5) and (4.6) into

$$\hat{x}_1^+ = \hat{x}_1 + K(z_1 - \hat{x}_1), \tag{4.7}$$

$$\begin{aligned} \sigma_1^{2,+} &= \sigma_1^2 - K\sigma_1^2 \\ &= (1-K)\sigma_1^2, \end{aligned} \tag{4.8}$$

where

$$K = \frac{\sigma_1^2}{\sigma_1^2 + \sigma_v^2}. \tag{4.9}$$

Factor K is a weighting factor that determines how much of the information from the measurement should be taken into acount when updating the state estimate. If there is almost no uncertainty in the last location estimate, that is, if σ_1^2 is close to zero, then K will be close to zero. This has as a consequence that the received measurement is not taken into great account. If the uncertainty in the measurements is small, that is, if σ_v^2 is small, then K will approach one. This implies that the measurement *will* in fact be taken into account.

Essence. With these equations we have in essence shown the equations that the KF uses when the state and measurements consist of one variable. The KF estimates the state of a system that can be described by a linear equation like (4.1). For the purpose of reducing the uncertainty, the KF uses measurements that are modeled according to a linear equation like (4.2). Starting from an initial state, the KF incorporates relative information using equations (4.3) and (4.4). To incorporate absolute information, the KF uses equations (4.7) and (4.8) with means of the K factor from equation (4.9).

In the following sections we will formalize the concepts that we used here and derive the general KF equations that can also be used when the state we want to estimate consists of more than one variable.

4.3 Concepts

The KF is a *state estimator* that works on a *prediction-correction* basis. This means that it computes a *belief* in a certain state estimate by first making a prediction based on the dynamics of the system and later correcting this prediction using measurements of the system.

4.3.1 State Estimator

The main idea of the KF is that it estimates the *true state* of some system, like for example the location of a mobile robot. In particular, it estimates the state and gives a measure of how certain it is that the state estimate is the true state. What makes estimating the state difficult is that the state may change over time and is subject to noise. Besides this, the variables of the state may not be directly observable. The KF uses sensor readings that can make observations of the state; sensor readings that are also corrupted by noise. With all these complicating factors, the KF still has to optimally use the measurements to estimate the state.

4.3.2 Beliefs

More formally, the KF estimates the conditional probability of being in state x_k given available measurements $z_1, \ldots, z_k$. We call the probability of being in state x_k given observations $z_1, \ldots, z_k$ the *belief*[1],

$$Bel(x_k) = P(x_k|z_1, \ldots, z_k). \tag{4.10}$$

We can split this belief definition into the prior belief $Bel^-(x_k)$ and the posterior belief $Bel^+(x_k)$ using Bayes' rule, the theorem of total probability, and the Markov assumption. We then get[2],

$$\begin{aligned} Bel^-(x_k) &= P(x_k|z_1, \ldots, z_{k-1}) \qquad (4.11)\\ &= \int_{\Xi} P(x_k|x_{k-1}) Bel^+(x_{k-1}) dx_{k-1} \\ Bel^+(x_k) &= P(x_k|z_1, \ldots, z_k) \\ &= \frac{P(z_k|x_k) Bel^-(x_k)}{P(z_k|z_1, \ldots, z_{k-1})}. \qquad (4.12) \end{aligned}$$

The prior belief is the conditional probability of being at state x_k given all the measurements z up to step k. The posterior belief is the conditional probability of being at state x_k given all the measurements z up to and including step k. In order to compute the beliefs, we need to find expressions for the system model $P(x_k|x_{k-1})$ and the measurement model $P(z_k|x_k)$.

4.3.3 Prediction-Correction

Prediction. The KF computes the belief by first computing the prior belief and then computing the posterior belief. The computation of the prior belief $Bel^-(x_k)$ in equation (4.11) can be seen as a *prediction* of the state of the system after one time step. Without looking at new measurement

[1] This is in a similar way as we did in Chapter 3.
[2] See Chapter 3 for details.

information, the prior belief estimates what the state of the system after one step most likely is. It uses a model of the system $P(x_k|x_{k-1})$ and the belief in what the state was at the last time step $Bel^+(x_{k-1})$.

Correction. Due to noise in the system, there may be a prediction error. That is, the prediction may be different from the true state. The computation of the posterior belief $Bel^+(x_k)$ in equation (4.12) can be seen as a *correction* of the state estimate resulting from the prediction. After the KF has calculated the prior belief, new measurement information gives direct information about the true state of the system. This measurement information can be used to correct the predicted state. For this the KF uses a model of the measurements $P(z_k|x_k)$. This model describes how likely it is that given a state x_k a sensor reading results in measurement z_k. Given this model and a real measurement, the KF corrects the prior belief in the state $Bel^-(x_k)$.

This described the main idea of the KF. It first predicts the true state of the system after a time step has passed and it then corrects its prediction using measurements that are made of the true state. The KF is therefore also called a *predictor-corrector* state estimator [29].

4.4 Assumptions

In order to predict and correct the belief, the KF needs a model of the system and measurements. The KF assumes a *Linear Dynamic System* description of the system of which it is estimating the state. The dynamic system may be corrupted by *noise* sources, which the KF assumes can adequately be modeled by *independent, white, zero-mean, Gaussian* distributions.

4.4.1 Linear Dynamic System

The KF assumes that the system state and measurements can be described by a *linear dynamic system* [29, 21]. This is a set of linear equations that models the evolution of the state of the system over time and that describes how measurements are related to the state. The KF assumes a linear model, since it simplifies the computations and since often a linear approach is adequate for the problem to be modeled. When the problem is not linear, then we can use linearizing techniques to transform a non-linear problem into a linear[3]. A linear dynamic system consists of a *system* and a *measurement* model.

[3] We will discuss this in Chapter 5.

System Model. The *system model* describes how the true state of the system evolves over time. The KF needs this model in order to make predictions about the state. The KF assumes that the state of the system evolves according to the linear equation

$$x_k = Ax_{k-1} + w_{k-1}. \tag{4.13}$$

The true state $x_k \in \mathbb{R}^n$ of the system at time k depends on the state of the system one step earlier x_{k-1} and some noise. Matrix A is an $n \times n$ matrix that, without taking into account possible noise in the system, relates the state of the previous time step $k-1$ to the state at the current step k. The vector $w_{k-1} \in \mathbb{R}^n$ models the noise in the system; it models the effects of unmodeled influences on the system state.

Measurement Model. The *measurement model* describes how measurements are related to states. The KF needs a model of the measurements in order to correct the state prediction when a measurement is available. If it has a model that given the true state of the system describes what the measurement will be, then it can compare the real measurement with the measurement that the model gives to correct the state prediction. The KF assumes that the measurements can be modeled by an equation that linearly relates the state of the system to a measurement,

$$z_k = Hx_k + v_k. \tag{4.14}$$

The true measurement $z_k \in \mathbb{R}^m$ at time k depends linearly on the state of the system x_k. The $m \times n$ matrix H relates the current state x_k to the measurement z_k. Given a state, H models what the real measurement should be when there is no noise in the sensors. However, there is noise in the measurements, which is modeled by the vector $v_k \in \mathbb{R}^m$.

Markov process. Notice in equation (4.13) that the state x_k at time k does not depend on all other states and measurements given x_{k-1}. Notice also in equation (4.14) that, given x_k, the measurement z_k does not depend on all other states and measurements. These properties make the system a *Markov process*.

4.4.2 Noise Characteristics

As mentioned and modeled, the system and sensors are subject to noise. The KF assumes that the system noise w_k in (4.13) and measurement noise v_k in (4.14) are random variables that are *independent*, *white*, *zero-mean Gaussian* probability distributions. Besides this, the KF assumes that the initial state of the system x_0 at time $k = 0$ is independent and Gaussian distributed.

Independence. The *independence* assumption makes the computations involved in the state estimation easier. In general it is fair to assume that the noise in system and measurements are independent. That is, the amounts of noise have no influence on each other. For example, noise in color intensities in camera images does not influence noise in the guidance system of a robot.

White Noise. The *white noise* assumption also greatly simplifies the mathematics involved in the filter. White noise is noise that has power at all frequencies in the spectrum and that is completely uncorrelated with itself at any time except the present [29]. At the current time, the noise is either there, or it is not, which is also true for the next time steps. This assumption implies that errors are not correlated through time. This means that knowing the amount of noise at the current time does not help in predicting what the amount of noise will be at any other time.

Zero-mean. The *zero-mean* assumption implies that the errors in system and measurements are random. Noise can be classified into *systematic* noise and *non-systematic* or *random* noise [39]. Systematic noise is noise that constantly corrupts the system state or measurements in a certain way. It is *biased* noise, often caused by inaccurate parameters. For example, if the diameter of wheels of a guidance system is not accurately determined, there will be a systematic error in the position estimation. Random noise is noise that is not systematic in that way. Random noise is sometimes positive, sometimes negative, but, in the case of *zero-mean*, on average zero.

Gaussian. The *Gaussian* assumption deals with the amplitude of the noise. It states that the amount of noise involved can be modeled as a bell-shaped curve. The Gaussian noise assumption is justified by assuming that system and measurement noise are often caused by multiple small noise sources. No matter how the individual noise sources are distributed, the sum of these independent sources will be Gaussian distributed [29]. The Gaussian noise assumption makes the computations of the filter more tractable.

Another reason why Gaussian noise is a convenient assumption is that we often only know the first and second order statistics of the noise sources [38], that is, the *mean* and *variance*. Many measuring devices provide only a nominal value of the measurement. By running experiments and looking at sensor specifications we can estimate the average error [38]. Gaussians are fully characterized by their mean and variance and therefore capture all available noise information.

With the zero-mean and Gaussian distribution assumptions for the noises, we can write down how they are distributed. If we let $Q_k = E[(w_k)(w_k)^T]$ be the process noise covariance at time step k and $R_k = E[(v_k)(v_k)^T]$ be the measurement noise covariance at time step k, we can express the system

noise w_k and the measurement noise v_k as

$$w_k \sim N(0, Q_k) \tag{4.15}$$
$$v_k \sim N(0, R_k), \tag{4.16}$$

where $N(\mu, \Sigma)$ denotes the Gaussian function with mean μ and covariance Σ. The main diagonal of the covariance matrices Q_k and R_k contains the variance in the state and measurement vector variables respectively. The off-diagonal elements are zero, since we assume that the noises are independent.

4.5 Gaussian Implications

The assumption that the random variables involved are Gaussian distributed brings some useful features. Gaussian distributions are fully characterized by their mean and their covariance. With the Gaussian assumption it therefore suffices to compute the mean and the covariance of the prior and posterior belief. In this section we look at how we can express the system and measurement models in terms of Gaussians and what consequences the Gaussian assumption has for the computation of the beliefs themselves.

4.5.1 System and Measurement Models

Gaussian System Model. We want to know how the system model from equation (4.13) is distributed given the Gaussian assumption, since this is the distribution $P(x_k|x_{k-1})$ needed to compute the prior belief from equation (4.11). As mentioned the evolution of the state of the system is modeled by a linear equation as

$$x_k = Ax_{k-1} + w_{k-1},$$

in which system noise w_{k-1} is zero-mean Gaussian distributed with covariance Q_k, as in (4.15). We can look at the term Ax_{k-1} in this equation as Gaussian distributed with mean Ax_{k-1} and covariance 0, so, $Ax_{k-1} \sim N(Ax_{k-1}, 0)$. There is no uncertainty in this term since the system model distribution $P(x_k|x_{k-1})$ we are looking for is conditioned on x_{k-1}. Given that the sum of two Gaussian random variables results in another Gaussian random variable, we derive the probabilistic system model as

$$P(x_k|x_{k-1}) = N(Ax_{k-1}, Q_k). \tag{4.17}$$

Gaussian Measurement Model. In the same way we want to find the probabilistic characteristics of the measurement model from equation (4.14), since this is the distribution $P(z_k|x_k)$ needed to compute the posterior belief from equation (4.12). The measurements are modeled according to the

measurement model

$$z_k = Hx_k + v_k,$$

where measurement noise v_k is zero-mean Gaussian distributed with covariance R_k, as in (4.16). In this, we again look at the term Hx_k as a Gaussian distribution with mean Hx_k and no uncertainty, so, $Hx_k \sim N(Hx_k, 0)$. There is no uncertainty since the measurement model is conditioned on x_k. Given these two Gaussian distributions, we see that the conditional probability of observing z_k given x_k is Gaussian distributed as

$$P(z_k|x_k) = N(Hx_k, R_k). \tag{4.18}$$

4.5.2 Gaussian Beliefs

Recall that the *prior* belief is the belief in a state x_k given all the information up to step k and that the *posterior* belief is the belief using all the information up to and including step k. With the Gaussian assumption we can express the way those beliefs are distributed more precisely.

Gaussian Prior Belief. Since Gaussians are fully characterized by their mean and covariance, the prior belief is distributed as $Bel^-(x_k) = N(\hat{x}_k^-, P_k^-)$, where we let $\hat{x}_k^-$ be the mean and P_k^- be the covariance. We will also refer to the mean $\hat{x}_k^-$ as the *prior state estimate*, and to the covariance P_k^- as the *prior error covariance*. The prior state estimate and error covariance are defined as the first and second statistical moments,

$$\hat{x}_k^- = E[x_k|z_1, \ldots, z_{k-1}] \tag{4.19}$$

$$P_k^- = E[(e_k^-)(e_k^-)^T|z_1, \ldots, z_{k-1}], \tag{4.20}$$

where e_k^- is the *prior estimation error* in the state estimate,

$$e_k^- = x_k - \hat{x}_k^-. \tag{4.21}$$

Gaussian Posterior Belief. Similarly, the posterior belief is distributed as $Bel^+(x_k) = N(\hat{x}_k^+, P_k^+)$, where we let $\hat{x}_k^+$ be the mean and P_k^+ be the covariance. We will mention the mean $\hat{x}_k^+$ also as the *posterior state estimate*, and the covariance P_k^- as the *posterior error covariance*. The posterior state estimate and error covariance are defined as

$$\hat{x}_k^+ = E[x_k|z_1, \ldots, z_k] \tag{4.22}$$

$$P_k^+ = E[(e_k^+)(e_k^+)^T|z_1, \ldots, z_k], \tag{4.23}$$

where e_k^+ is the *posterior estimation error* in the state estimate,

$$e_k^+ = x_k - \hat{x}_k^+. \tag{4.24}$$

State Estimates and Error Covariances. The state estimates $\hat{x}_k^-$ and $\hat{x}_k^+$ are vectors that given the available data are most likely to represent the true state x_k of the system. The error covariance matrices P_k^- and P_k^+ contain the covariances between the different elements of the state estimate. In particular, the leading diagonal of these matrices contains the variances of the separate elements of the state estimate. The variances give an indication of how precise the KF thinks the state estimate elements are estimated. The smaller the variances are, the more precise the state estimate is.

The state estimates in (4.19) and (4.22) and error covariances in (4.20) and (4.23) depend on the random measurements $z_1, \ldots, z_k$, and thus they are random variables themselves. We can proof however that the measurements do not influence the prior and posterior error covariances [51]. We proof that the measurements do not influence the posterior error covariance by showing that the correlation between the posterior error e_k^+ and the measurements $z_1, \ldots, z_k$ disappears. For notational convenience we let z^k be the sequence of measurements $z_1, \ldots, z_k$. We use the definition of the posterior error from (4.24) and the definition of the posterior state estimate from (4.22) to compute the correlation as

$$\begin{aligned} E[e_k^+ z^k] &= E[x_k z^k] - E[\hat{x}_k^+ z^k] \\ &= \int p(x_k, z^k) x_k z^k dx_k dz^k - \int \left(\int p(x_k|z^k) x_k dx_k \right) p(z^k) z^k dz^k \\ &= \int p(x_k, z^k) x_k z^k dx_k dz^k - \int p(x_k|z^k) p(z^k) x_k z^k dx_k dz^k \qquad (4.25) \\ &= 0. \end{aligned}$$

Expression (4.25) equals zero, since $p(x_k, z^k) = p(x_k|z^k)p(z^k)$. So, the correlation between the estimation error and the measurements disappears, indicating that they are independent of one another when they are Gaussian distributed [51].

4.6 KF Equations

With the discussion of the assumptions and the implications that the assumptions have on the state estimation, we are now prepared to derive the equations that make up the KF. We start by deriving the equations for the prior belief and then use the results in deriving the equations for the posterior belief.

4.6.1 Prior Belief

Calculating the prior belief comes down to calculating the prior state estimate $\hat{x}_k^-$ from equation (4.19) and the prior error covariance P_k^- from equation (4.20).

Prior State Estimate. The prior state estimate is defined as the expected state x_k given all measurements up to and including the last step $k-1$. By using knowledge of the evaluation over time of the state captured in the system equation (4.13) we can express the prior state estimate $\hat{x}_k^-$ as

$$\begin{aligned}\hat{x}_k^- &= E[x_k|z_1,\dots,z_{k-1}]\\ &= E[Ax_{k-1}+w_{k-1}|z_1,\dots,z_{k-1}]\\ &= AE[x_{k-1}|z_1,\dots,z_{k-1}]+E[w_{k-1}|z_1,\dots,z_{k-1}]\\ &= AE[x_{k-1}|z_1,\dots,z_{k-1}]+E[w_{k-1}]\\ &= A\hat{x}_{k-1}^+. \end{aligned} \tag{4.26}$$

Here we have used that the system noise is assumed to be independent of the measurements, which implies that $E[z_k, w_l] = 0$ for $k \leq l$. Besides this, we have also used that the system noise has zero mean from (4.15).

Prior Error Covariance. To calculate the prior covariance from equation (4.20), we first have to calculate the prior error e_k^- in the state estimate from (4.21). We evaluate this error estimate using the system equation (4.13) and the equation for the prior state estimate $\hat{x}_k^-$, equation (4.26),

$$\begin{aligned}e_k^- &= x_k - \hat{x}_k^-\\ &= Ax_{k-1}+w_{k-1}-A\hat{x}_{k-1}^+\\ &= A(x_{k-1}-\hat{x}_{k-1}^+)+w_{k-1}\\ &= Ae_{k-1}^+ + w_{k-1}. \end{aligned} \tag{4.27}$$

Note that the posterior error estimate e_{k-1}^+ is independent of the system noise w_{k-1}. To see this, notice that the posterior error estimate e_{k-1}^+ is a function of x_{k-1} and $\hat{x}_{k-1}^+$, equation (4.24); the latter of these is a function of $z_1,\dots,z_{k-1}$, equation (4.22). We assumed that the system noise w_{k-1} is independent of the true state and the true measurements. Thus, the system noise is independent of the posterior error estimate.

We can now calculate the prior error covariance P_k^- from (4.20). We substitute the prior error estimate (4.27) into (4.20),

$$\begin{aligned}P_k^- &= E[(e_k^-)(e_k^-)^T]\\ &= E[(Ae_{k-1}^+ + w_{k-1})(Ae_{k-1}^+ + w_{k-1})^T]\\ &= AE[(e_{k-1}^+)(e_{k-1}^+)^T]A^T + E[(w_{k-1})(w_{k-1})^T]\\ &= AP_{k-1}^+A^T + Q_{k-1}. \end{aligned} \tag{4.28}$$

Prediction. We have derived the equations that the KF uses to update the belief when the system moves one step forward in time. Using equation

(4.26) the KF predicts what the state of the system most likely is. Using equation (4.28) the KF expresses how much uncertainty there is in the state estimate being the true state. If we combine this, we can express the prior belief in terms of a Gaussian distribution as

$$Bel^-(x_k) = N(\hat{x}_k^-, P_k^-) \tag{4.29}$$

$$= N(A\hat{x}_{k-1}^+, AP_{k-1}^+A^T + Q_{k-1}). \tag{4.30}$$

4.6.2 Posterior Belief

In order to find expressions for the posterior state estimate and error covariance we will do some rewriting first. We substitute the Gaussian representation of the measurement model (4.18) and the prior belief (4.29) into the definition for the posterior belief (4.12),

$$Bel^+(x_k) = \frac{N_{z_k}(Hx_k, R_k)N_{x_k}(\hat{x}_k^-, P_k^-)}{P(z_k|z_1, \ldots, z_{k-1})}, \tag{4.31}$$

where the subscripts z_k and x_k in N_{z_k} and N_{x_k} denote the spaces over which the Gaussians are distributed. That is, the prior belief $N_{x_k}(\hat{x}_k^-, P_k^-)$ is a probability distribution over the space of states x_k, and the measurement model $N_{z_k}(Hx_k, R_k)$ is a probability distribution over the space of measurements z_k.

We want to combine these probability distributions into a distribution over the state space. We achieve this in two steps. First we *convert* the measurement model from measurement space to state space, and then we *combine* the resulting Gaussian with the prior belief to form the posterior belief in state space.

Converting Measurement Model to State Space. Since we want the posterior belief to be a distribution over the state space, we convert the measurement model to the state space. This can be done by means of the identity [51]

$$\begin{aligned} N_z(Ax, \Sigma) &= k_1(z)N_x(d, D), \\ d &= (A^T\Sigma^{-1}A)^{-1}A^T\Sigma^{-1}z \\ D &= (A^T\Sigma^{-1}A)^{-1}. \end{aligned}$$

This identity states that a Gaussian distribution N_z over some z space that has as mean a linear combination A of some vector x and as variance Σ can be converted into a factor k and a Gaussian distribution with mean d and covariance D over the space of which vector x is an element.

If we apply this identity to the measurement model $N_{z_k}(Hx_k, R_k)$ we get as result the Gaussian $N_{x_k}(\mu', \Sigma')$ over the state space,

$$\begin{aligned} N_{z_k}(Hx_k, R_k) &= k_1(z_k) N_{x_k}(\mu', \Sigma'), \\ \mu' &= (H^T R_k^{-1} H)^{-1} H^T R_k^{-1} z_k \\ \Sigma' &= (H^T R_k^{-1} H)^{-1}. \end{aligned} \tag{4.32}$$

Substituting this result into (4.31), we get

$$Bel^+(x_k) = \frac{k_1(z_k) N_{x_k}(\mu', \Sigma') N_{x_k}(\hat{x}_k^-, P_k^-)}{P(z_k | z_1, \ldots, z_{k-1})}. \tag{4.33}$$

The posterior belief is now reduced to the product of two Gaussians in state space times the factor $k_1(z_k)/P(z_k|z_1, \ldots, z_{k-1})$.

Combining Measurement Model and Prior Belief. To combine the two Gaussians over state space from equation (4.31) into one Gaussian over state space we use another identity. This identity combines the product of two Gaussians in the same space to the product of one Gaussian in that same space and another in the space of which the mean of the first is part [51],

$$\begin{aligned} N_x(a, A) N_x(b, B) &= N_x(c, C) N_a(d, D), \\ c &= (A^{-1} + B^{-1})^{-1} (A^{-1} a + B^{-1} b) \\ C &= (A^{-1} + B^{-1})^{-1} \\ d &= b \\ D &= A + B \end{aligned} \tag{4.34}$$

We use this identity to rewrite the posterior belief as formulated in (4.33) once more. Applying (4.34), letting $a = \hat{x}_k^-, A = P_k^-, b = \mu'$, and $B = \Sigma'$, we obtain

$$\begin{aligned} Bel^+(x_k) &= \frac{k_1(z_k)}{P(z_k | z_1, \ldots, z_{k-1})} N_{x_k}(\mu'', \Sigma'') N_{\hat{x}_k^-}(\mu''', \Sigma''') \\ \mu'' &= ((P_k^-)^{-1} + \Sigma'^{-1})^{-1} ((P_k^-)^{-1} \hat{x}_k^- + \Sigma'^{-1} \mu') \\ \Sigma'' &= ((P_k^-)^{-1} + \Sigma'^{-1})^{-1} \\ \mu''' &= \mu' \\ \Sigma''' &= P_k^- + \Sigma'. \end{aligned} \tag{4.35}$$

In this, the factor

$$\frac{k_1(z_k) N_{\hat{x}_k^-}(\mu''', \Sigma''')}{P(z_k | z_1, \ldots, z_{k-1})},$$

equals 1 according to [51], since the posterior belief must be normalized with respect to x_k. We thus have that

$$Bel^+(x_k) = N_{x_k}(\mu'', \Sigma''). \tag{4.36}$$

With the result of (4.36) we have in fact determined the posterior state estimate as μ'', and the posterior error covariance as Σ''. With some rewriting we can make the computation of these quantities more efficient.

Posterior State Estimate. From (4.36) we see that we can express the posterior state estimate as

$$\begin{aligned}
\hat{x}_k^+ &= \mu'' \\
&= \left(\left(P_k^-\right)^{-1} + \left(\left(H^T R_k^{-1} H\right)^{-1} \right)^{-1} \right)^{-1} \\
&\quad \times \left(\left(P_k^-\right)^{-1} \hat{x}_k^- + \left(\left(H^T R_k^{-1} H\right)^{-1} \right)^{-1} \left(\left(H^T R_k^{-1} H\right)^{-1} H^T R_k^{-1} z_k \right) \right) \\
&= \left(\left(P_k^-\right)^{-1} + H^T R_k^{-1} H \right)^{-1} \times \left(\left(P_k^-\right)^{-1} \hat{x}_k^- + H^T R_k^{-1} z_k \right).
\end{aligned}$$

If we use the *Matrix Inversion Lemma* [29, 51] we can simplify this further. The Matrix Inversion Lemma states that if we have a $d \times d$ matrix $P > 0$, a $k \times k$ matrix $R > 0$ and a $k \times d$ matrix H, where $P > 0$ implies $a^T P a > 0$ for all a, then

$$\left(P^{-1} + H^T R^{-1} H\right)^{-1} = P - PH^T \left(HPH^T + R\right)^{-1} HP \tag{4.37}$$

$$\left(P^{-1} + H^T R^{-1} H\right)^{-1} H^T R^{-1} = PH^T \left(HPH^T + R\right)^{-1}. \tag{4.38}$$

Using these identities in our rewriting, we obtain

$$\begin{aligned}
\hat{x}_k^+ &= \left(P_k^- - P_k^- H^T \left(HP_k^- H^T + R_k\right)^{-1} HP_k^- \right) \left(P_k^-\right)^{-1} \hat{x}_k^- \\
&\quad + P_k^- H^T \left(HP_k^- H^T + R_k\right)^{-1} z_k \\
&= \left(P_k^- - K_k HP_k^-\right) \left(P_k^-\right)^{-1} \hat{x}_k^- + K_k z_k \\
&= \hat{x}_k^- - K_k H \hat{x}_k^- + K_k z_k \\
&= \hat{x}_k^- + K_k (z_k - H\hat{x}_k^-),
\end{aligned} \tag{4.39}$$

where

$$K_k = P_k^- H^T \left(HP_k^- H^T + R_k\right)^{-1}. \tag{4.40}$$

Posterior Error Covariance. We see directly from (4.36) that the posterior error covariance is

$$\begin{aligned} P_k^+ &= \Sigma'' \\ &= \left(\left(P_k^-\right)^{-1} + \left(\left(H^T R_k^{-1} H\right)^{-1} \right)^{-1} \right)^{-1} \\ &= \left(\left(P_k^-\right)^{-1} + H^T R_k^{-1} H \right)^{-1} \\ &= P_k^- - P_k^- H^T (H P_k^- H^T + R_k)^{-1} H P_k^- \\ &= P_k^- - K_k H P_k^- \\ &= (I - K_k H) P_k^-, \end{aligned} \tag{4.41}$$

where we have used the definition for K_k in (4.40).

Correction. With this final result we now also have derived the equations that the KF uses to correct the prior belief into the posterior belief. The KF uses equation (4.39) to calculate the posterior state estimate $\hat{x}_k^+$ when a measurement z_k of the true state arrives. Using equation (4.41) the KF updates the uncertainty P_k^+ that it has in the posterior state estimate being the true state. In both equations, the KF uses equation (4.40) to calculate the factor K_k. With the derived posterior state estimate and error covariance we characterize the posterior belief as

$$\begin{aligned} Bel^+(x_k) &= N(\hat{x}_k^+, P_k^+) \\ &= N(\hat{x}_k^- + K_k(z_k - H\hat{x}_k^-), (I - K_k H) P_k^-), \end{aligned} \tag{4.42}$$

where K_k comes from (4.40).

4.7 Linear Kalman Filter

With the equations of Section 4.6, we have derived the equations of the standard KF algorithm. Since this KF assumes that the system and measurements are governed by linear equations, this KF is commonly known as the *Linear Kalman Filter* (LKF). We first shortly describe the LKF algorithm in Section 4.7.1 and give the equations in the algorithm more meaning in Section 4.7.2.

4.7.1 Algorithm

The LKF consists of an *initialization* step after which it alternately performs *prediction* and *correction* steps.

Initialization. The LKF is initialized with the posterior belief $Bel^+(x_0) = N(\hat{x}_0^-, P_0^-)$. That is, to initialize the LKF we have to specify the initial posterior state estimate $\hat{x}_0^+$ and the initial uncertainty in this state estimate P_0^+.

Prediction Equations. At every time step the system can be in a different state. Therefore, the LKF calculates a new prior belief at every step. The *prediction*, or also *time update* or *propagation*, equations predict the new state of the system by projecting forward the most recent belief. They compute the prior belief $Bel^-(x_k) = N(\hat{x}_k^-, P_k^-)$ as we derived in Section 4.6.1, where

$$\hat{x}_k^- = A\hat{x}_{k-1}^+ \tag{4.43}$$
$$P_k^- = AP_{k-1}^+A^T + Q_{k-1}. \tag{4.44}$$

Correction Equations. The *correction*, or also *measurement update*, equations deal with the measurements. They are only used when there is a measurement. The measurements give direct information about the current state of the system. The correction equations correct the most recent belief by incorporating the information gained from measurements. They compute the posterior belief $Bel^+(x_k) = N(\hat{x}_k^+, P_k^+)$ as we derived in Section 4.6.2, where

$$\hat{x}_k^+ = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-) \tag{4.45}$$
$$P_k^+ = (I - K_kH)P_k^-, \tag{4.46}$$

in which

$$K_k = P_k^-H^T\big(HP_k^-H^T + R_k\big)^{-1}. \tag{4.47}$$

The new posterior belief is used in the next time step to compute the new prior belief. This recursive nature of KFs allows for practical implementations, because not all of the data is required to estimate the states.

4.7.2 Interpretation

4.7.2.1 Prediction Equations

The KF computes the prior state estimate $\hat{x}_k^-$ based on the last posterior state estimate $\hat{x}_{k-1}^+$ and the model that it has of the system. The best guess that the KF can make about the state of the system after it has progressed one step forward in time is the last best guess propagated through the model that the KF has of the system.

The KF also knows that the system evolution is subject to noise and therefore it has an increased uncertainty P_k^- in the prior state estimate.

The first term of the prior error covariance, $AP_{k-1}^{+}A^T$, propagates the uncertainty in the last posterior state estimate forward to the current prior state estimate. The second term, Q_{k-1}, is the system noise that corrupts the state of the system at every time step.

4.7.2.2 Correction Equations

The KF computes the posterior state estimate by taking the prior state estimate and combining this with the *Kalman Gain* K_k times the difference between a measurement z_k and a *measurement prediction* $H\hat{x}_k^-$, called the *innovation*.

Measurement Prediction. The term $H\hat{x}_k^-$ in equation (4.45) is a *measurement prediction*. Given the prior state estimate $\hat{x}_k^-$ and the measurement matrix H from the measurement model (4.14) the KF predicts what measurement it will receive. We denote the measurement prediction by

$$\hat{z}_k = H\hat{x}_k^- + \hat{v}_k, \tag{4.48}$$

where the measurement noise $\hat{v}_k$ is zero. This measurement prediction is a random variable that follows a Gaussian distribution. We see this by noticing that it depends linearly on the prior state estimate $\hat{x}_k^-$ and the measurement noise, which both are Gaussian random variables. We easily derive that the measurement prediction $\hat{z}_k$ follows the distribution

$$\hat{z}_k \sim N_z(H\hat{x}_k^-, HP_k^-H^T + R_k). \tag{4.49}$$

Innovation. We call the difference between a measurement z_k and a predicted measurement $\hat{x}_k^-$ the measurement *innovation* or *residual* $\tilde{z}_k$. The innovation tells how much a predicted measurement differs from a real measurement. We define the innovation as

$$\tilde{z}_k = z_k - \hat{z}_k. \tag{4.50}$$

If the innovation equals zero, then the predicted measurement exactly reflects the real measurement. This implies that the state estimate with which the measurement prediction was made was very close to the true state from which the measurement was made. However, if there is a difference between the predicted and observed measurement, then the prior state estimate needs to be updated with some quantity.

The innovation depends on the variables z_k and $\hat{z}_k$. Since the true measurements z_k is given it does not add any uncertainty to the innovation. The uncertainty in the innovation only depends on the uncertainty in the

measurement prediction $\hat{z}_k$, and is thus Gaussian distributed,

$$\tilde{z}_k \sim N_z(\mu_{\tilde{z},k}, \Sigma_{\tilde{z},k}) \tag{4.51}$$

$$\mu_{\tilde{z},k} = z_k - \hat{z}_k$$

$$\Sigma_{\tilde{z},k} = HP_k^- H^T + R_k. \tag{4.52}$$

Notice that if there is in fact uncertainty in the true measurement, due to for example uncertainty in the feature extraction, then the uncertainty in the innovation should be increased with this. The distribution of the innovation gives an idea of the spread of the innovations. It gives an idea of the errors in the measurement estimations. We can use this distribution to detect innovations that are very unlikely to occur. This gives a means of detecting occassional extreme noise levels in measurements. When we use the innovation distribution to reject measurements, we say that we have a *validation gate* [21].

As mentioned, when the innovation is large this indicates that there is a large difference between the prior state estimate and the true state. A naive way to combine the prior state estimate and the measurement innovation is by simply summing the two. However, if the measurements or measurement predictions are very uncertain relative to the prior state estimate this is not a good solution; in that case it is better to ignore the innovation. On the other hand, if the measurement innovation has relatively low uncertainty compared to the prior state estimate, then it *is* good to take much of the innovation into account.

Kalman Gain. We call the factor K_k in (4.47) the *Kalman Gain* (KG). This factor determines to what extend the innovation should be taken into account in the posterior state estimate. It determines this by looking at the relative uncertainty between the prior state estimate and the measurement innovation,

$$K_k = P_k^- H^T \left(HP_k^- H^T + R_k\right)^{-1}.$$

To compare the prior state estimate uncertainty in the state space with the innovation uncertainty in the measurement space, the KF converts the uncertainty in the measurement space to the state space by means of the matrix H^T.

Notice what happens when the noise in the measurements is almost zero. When the measurement error covariance R_k approaches zero, the KG weights the innovation more heavily. That is, the measurement innovation $\tilde{z}_k$ is trusted to contain more information than the prior state estimate,

$$\lim_{R_k \to 0} K_k = H^{-1}. \tag{4.53}$$

The KF relies less on the system model, and more on the measurements.

Contrarily, when the prior error covariance P_k^- approaches zero, the KG weights the residual less heavily. That is, as the prior error covariance P_k^- approaches zero, the measurement residual $\tilde{z}_k$ is taken less into account,

$$\lim_{P_k^- \to 0} K_k = 0. \tag{4.54}$$

The KF relies more on the system model, and less on the measurements.

Posterior Uncertainty. The KF also uses the KG to update the uncertainty that the KF has in the posterior state estimate being the true state. If the KG is close to H^{-1} the measurement innovation is taken into account almost completely. This means that the KF believes that the innovation contains relatively much information compared to the prior state estimate. This on its turn results in maximum decrease of uncertainty in the state estimate.

In particular, if the observations observe the values of every state variable directly, and there is a large amount of uncertainty in the prior state estimate relative to the measurements, then the KG will be close to H^{-1} and the posterior error estimate will be close to zero. This has as consequence that the KG will take following measurements not much into account, since the uncertainty in the state is so small and thus the KG will be low. The time it takes before the KF will significantly take innovations into account again depends on the amount of system noise added to the uncertainty of the prior state estimate at every time step.

The KG forms the heart of KFs. In Section 4.8 we will proof that the KG that we derived has as result that the KF is a Minimum Variance Estimator. That is, it minimizes the variance in the state estimates.

4.8 Minimum Variance Estimator

We can proof that the KG that is used by the KF to determine which part of the measurement innovation it should incorporate in its posterior state estimate minimizes the posterior error covariance. That is, when a new measurement arrives, the KF uses the KG to incorporate this measurement into the state estimate in such a way that the uncertainty in the state estimate is minimal after incorporation.

To proof that the derived KG minimizes the posterior error covariance, we show that the K_k that minimizes the sum of the variances in the posterior error covariance P_k^+ is in fact the K_k that we derived. The variances in the posterior error covariance are located on the leading diagonal. The *trace* of a matrix is the sum of the leading diagonal of that matrix. If we express the posterior error covariance P_k^+ in terms of K_k, then we can take the

derivative of the trace of this expression with respect to K_k. To find the K_k that minimizes the trace, we set the derivative to zero and solve for K_k [50].

1. We express the posterior error covariance P_k^+ in terms of K_k, using (4.23), (4.24), and (4.39),

$$\begin{aligned} P_k^+ &= E[(e_k^+)(e_k^+)^T] \\ &= E[(x_k - \hat{x}_k^- - K(z_k - H\hat{x}_k^-))(x_k - \hat{x}_k^- - K(z_k - H\hat{x}_k^-))^T]. \end{aligned}$$

2. We take the indicated expectations, where we also use the measurement model from (4.14), the definition for the prior error covariance from (4.20), and the zero mean assumption of the system noise,

$$\begin{aligned} P_k^+ &= E[(x_k - \hat{x}_k^- - K_k(Hx_k + v_k - H\hat{x}_k^-)) \\ &\qquad (x_k - \hat{x}_k^- - K_k(Hx_k + v_k - H\hat{x}_k^-))^T] \\ &= E[((I - K_kH)(x_k - \hat{x}_k^-) + (K_kv_k)) \\ &\qquad ((I - K_kH)(x_k - \hat{x}_k^-) + (K_kv_k))^T] \\ &= (I - K_kH)E[(x_k - \hat{x}_k^-)(x_k - \hat{x}_k^-)^T](I - K_kH)^T \\ &\quad + K_kE[v_kv_k^T]K_k^T + 2(I - K_kH)E[(x_k - x_k^-)v_k^T]K_k^T \\ &= (I - K_kH)P_k^-(I - K_kH)^T + K_kR_kK_k^T. \end{aligned}$$

3. We compute the derivative of the trace of this result, where we use that $\partial \mathrm{tr}(ABA^T)/\partial A = 2AB$,

$$\begin{aligned} \frac{\partial \mathrm{tr}(P_k^+)}{\partial K_k} &= \frac{\partial}{\partial K_k}[\mathrm{tr}((I - K_kH)P_k^-(I - K_kH)^T + K_kR_kK_k^T)] \\ &= \frac{\partial}{\partial K_k}[\mathrm{tr}((I - K_kH)P_k^-(I - K_kH)^T) + \mathrm{tr}(K_kR_kK_k^T)] \\ &= -2(I - K_kH)P_k^-H^T + 2K_kR_k. \\ &= -2P_k^-H^T + 2K_k(HP_k^-H^T + R_k), \end{aligned}$$

4. We set the derivate of the trace to zero and solve for K_k,

$$K_k = P_k^-H^T(HP_k^-H^T + R_k)^{-1}.$$

The K_k that we have derived in this way is exactly the K_k that we derived for the KF. We have hereby proved that this K minimizes the posterior error covariance.

4.9 Additional System Dependencies

So far we have only looked at systems of which the state depends only on the previous state and on system noise. Sometimes however the evolution of the state depends on extra, possibly random, variables. *Drift* and *external input* [29, 21, 51] are two common types of additional influences on the state of the system.

Drift. When the state of the system at every time step changes with a certain constant the system state moves away from the ideal state trajectory as modeled by the transition matrix A. We call this constant influence *drift*. Drift causes devices that make relative measurements to have an increasing measure error. The influence of drift does not have to be the same at every time step. Noise can influence the amount of drift.

External Input. *External input* is input that influences the evolution of the state, but that is not part of the state. An example of external input is *control input*. In control applications there often is control input that changes the state of the system. In robot navigation applications a robot sends control commands to its guidance system to drive over a certain distance at a certain speed. These commands make that the robot changes its location. We should therefore include them in the model of the evolution of the system state.

External input does not have to be deterministic; it can be *noisy*. In the robot navigation case, when a robot sends control commands to its guidance system, there is no guarantee that the wheels will exactly execute these commands. In fact, due to for example slippage and irregularities of the surface, the robot can easily execute the commands slightly different from expected. By monitoring the actual execution of the actions by means of relative position measurements, we can find out more precisely what the wheels actually have done. However, noise in these measurements can still corrupt the information.

Either way, if we can somehow estimate the uncertainty in the control commands, we can include this uncertainty in our system model to make it more adequate.

4.9.1 System Model Revisited

We can adjust our system model such that we can also include extra input like drift and external input influences. We revise the system model to describe the evolution of a system state x_k based on previous state x_{k-1}, extra input $u_{k-1} \in \mathbb{R}^l$, and system noise w_{k-1},

$$x_k = Ax_{k-1} + Bu_{k-1} + w_{k-1}. \tag{4.55}$$

In this system equation the $l \times n$ matrix B determines what the contribution of the extra input u_{k-1} is to the state x_k at time k. We will assume that the extra input is Gaussian distributed, and independent of the other random variables,

$$u_k \sim N(\hat{u}_k, U_k). \tag{4.56}$$

4.9.2 Beliefs Revisited

We also revise the belief from (4.10). It is no longer the conditional probability of system state x_k given only the measurements z. The conditional probability now also takes into account the extra inputs u.

The prior belief is the conditional probability of the system state x_k at time k, given all measurements and external inputs up to and including time $k-1$. This has consequences for the computation of the prior belief, since the extra input changes the system state.

The posterior state is now also conditioned on all the extra inputs, but this has no consequences for the computation of the posterior belief, since we assume that the extra input is independent of the measurements.

4.9.3 KF Prediction Revisited

We want to update the KF prediction equations that we derived in Section 4.6.1 to compute the new prior belief of a system governed with the new system equation. We expect that since the extra input simply adds a quantity to the new state, the new prior state estimate and error covariance equations will simply get an additional term that takes this extra input into account.

Revised Prior State Estimate. We derive the equation for the prior state estimate $\hat{x}_k^-$ as we did in Section 4.6.1 by taking the expectation of the system state x_k conditioned on all the measurements and extra inputs up to and including time $k-1$,

$$\begin{aligned}
\hat{x}_k^- &= E[x_k | z_1, \ldots, z_{k-1}, u_1, \ldots, u_{k-1}] \\
&= E[Ax_{k-1} + Bu_{k-1} + w_{k-1} | z_1, \ldots, z_{k-1}, u_1, \ldots, u_{k-1}] \\
&= AE[x_{k-1} | z_1, \ldots, z_{k-1}, u_1, \ldots, u_{k-1}] \\
&\quad + BE[u_{k-1} | z_1, \ldots, z_{k-1}, u_1, \ldots, u_{k-1}] \\
&\quad + E[w_{k-1} | z_1, \ldots, z_{k-1}, u_1, \ldots, u_{k-1}] \\
&= AE[x_{k-1} | z_1, \ldots, z_{k-1}, u_1, \ldots, u_{k-1}] \\
&\quad + BE[u_{k-1} | u_1, \ldots, u_{k-1}] + E[w_{k-1}] \\
&= A\hat{x}_{k-1}^+ + B\hat{u}_{k-1},
\end{aligned} \tag{4.57}$$

where $\hat{u}_{k-1}$ is the best estimate of the extra input from (4.56), and where we used that the extra input is independent of the measurements and system noise.

We notice that indeed the addition of extra input u_{k-1} simply adds an extra term to the expression for the prior state estimate, where B is the matrix that determines how the extra input contributes to the state estimate.

Revised Prior Error Covariance. We also compute the uncertainty in the prior state estimate as we did earlier in Section 4.6.1. We first compute the error in the prior state estimate,

$$\begin{aligned} e_k^- &= x_k - \hat{x}_k^- \\ &= Ax_{k-1} + Bu_{k-1} + w_{k-1} - A\hat{x}_{k-1}^+ - B\hat{u}_{k-1} \\ &= A(x_{k-1} - \hat{x}_{k-1}^+) + B(u_{k-1} - \hat{u}_{k-1}) + w_{k-1} \\ &= Ae_{k-1}^+ + Be_{u,k-1} + w_{k-1}, \end{aligned} \tag{4.58}$$

where $e_{u,k-1} = u_{k-1} - \hat{u}_{k-1}$ is the error in the extra input estimate. We now substitute the prior error into the estimate e_k^- in the definition of the prior error covariance, and obtain

$$\begin{aligned} P_k^- &= E[(e_k^-)(e_k^-)^T] \\ &= E[(Ae_{k-1}^+ + Be_{u,k-1} + w_{k-1})(Ae_{k-1}^+ + Be_{u,k-1} + w_{k-1})^T] \\ &= AE[(e_{k-1}^+)(e_{k-1}^+)^T]A^T + BE[(e_{u,k-1})(e_{u,k-1})^T]B^T \\ &\quad + E[(w_{k-1})(w_{k-1})^T] \\ &= AP_{k-1}^+A^T + BU_{k-1}B^T + Q_{k-1}, \end{aligned} \tag{4.59}$$

where U_{k-1} is the covariance of the extra input from (4.56). The last equation was obtained by exploiting the fact that $E[e_{k-1}^+ e_{u,k-1}] = E[e_{k-1}^+ w_{k-1}] = E[e_{u,k-1}w_{k-1}] = E[w_{k-1}e_{k-1}^+] = 0$.

Again, we notice that addition of independent extra input to the system simply adds an extra term, in this case the term $BU_{k-1}B^T$. Notice also that if there is no uncertainty in the extra input, that is, $U_{k-1} = 0$, then the term $Be_{u,k-1}$ in (4.58), and the term $BU_{k-1}B^T$ in (4.59) will be zero, resulting in the same error covariance as derived in the case of no extra input.

With the prediction update equations (4.57) and (4.59) we have extended our basic KF to systems that have extra input as (4.56) and can be modeled according to system equation (4.55). We notice that making the system state depend on extra Gaussian distributed, independent variables, results in simple extensions of the prior state and error covariance equations.

4.10 Summary

The Kalman Filter (KF) is a recursive data processing algorithm that has succesfully been used in a wide range of applications from prices of traded goods prediction, to spacecraft control. The KF estimates the state of a noisy system using noisy measurements. More precisely, it calculates the conditional probability distribution over the space of states given measurements, the belief. It does this in a prediction-correction fashion, where it first predicts the state of the system based on system dynamics and then corrects its prediction based on measurements.

The KF makes a number of assumptions on the system, measurements and different noises that are involved in the estimation problem. The KF assumes that the system and measurements are adequately modeled by a linear dynamic system, and that noises are independent, white, Gaussian and with zero-mean. Moreover, the KF assumes that the initial state of the system is also independent and Gaussian distributed.

The KF calculates the belief by performing two alternating steps. In the prediction step the KF calculates the prior belief. This belief consists of the prior state estimate and the prior error covariance, which describes the uncertainty in the state estimate. Given the model and the last state estimate the prediction step predicts the most likely current state of the system after a time step.

The KF applies the correction step when it has access to a measurement. It uses the measurement to form the posterior belief. The KF computes the measurement innovation as the difference between the true measurement and a measurement prediction. It uses the Kalman Gain to determine to what extent the measurement innovation should be incorporated into the state estimate. We can proof that the KF incorporates measurements in such a way that the posterior error in the state estimate is minimized, which makes the KF a minimum variance estimator.

We can extend the basic KF equations to systems of which the state is influenced by drift and external inputs. We can use the revised equations in control applications, like for example when we want to model navigating robots.

Chapter 5

Kalman Filter Extensions

In Chapter 4 we described the basic Kalman Filter that addresses the problem of estimating the state of a noisy system that can be described by a linear system and a linear measurement model. Since the invention of the KF, many extensions relaxing some of the assumptions that the basic KF poses on the system and measurements have been developed. In this chapter we will discuss how the linear model assumption can be relaxed. We will discuss how to generalize the KF to estimate the state of a problem that is adequately modeled by nonlinear instead of linear system and measurement equations.

In Section 5.1 we define how we model nonlinear dynamic systems. In Section 5.2 we derive and discuss the *Perturbation KF* that estimates the state of nonlinear problems using linearization. The linearization technique of the Perturbation KF has some disadvantages. In Section 5.3 we discuss the *Extended KF*, that takes away one of the disadvantages, and in Section 5.4 we discuss the *Iterated Extended KF*, that improves the performance of the Extended KF. In Section 5.5 we generalize the Extended KF equations to system and measurement models with any number of parameters. We end the chapter with some references to related work in Section 5.6.

5.1 Nonlinear Dynamic Systems

Many dynamic system and sensor models are not completely linear, but not far from it either. This means that the functions that describe the system state and measurements are nonlinear, but approximately linear for small differences in the values of the state variables. Instead of assuming a linear dynamic system, we now consider a *nonlinear dynamic system*, consisting of a *nonlinear* system and a *nonlinear* measurement model.

Nonlinear System Model. The system of which we want to estimate the state is no longer governed by the linear equation from (4.13), but by a

nonlinear equation. We have that

$$x_k = f(x_{k-1}) + w_{k-1}, \tag{5.1}$$

where $f(\cdot)$ is a *nonlinear system function* relating the state of the previous time step to the current state, and where w_{k-1} represents the noise corrupting the system. The noise is assumed to be independent, white, zero-mean, and Gaussian distributed.

Nonlinear Measurement Model. We also no longer assume that the measurements are governed by a linear equation as in (4.14). Instead, we have that

$$z_k = h(x_k) + v_k, \tag{5.2}$$

where $h(\cdot)$ is a *nonlinear measurement function* relating the state of the system to a measurement, and where v_k is the noise corrupting the measurement. This noise is also assumed to be independent, white, zero-mean, and Gaussian distributed.

5.2 Perturbation Kalman Filter

Linear models have many advantages. They are easy to compute and to understand. They give predictable outputs, in time and over iterations. Besides this, linear theory is complete and developed, and linear differential equations are easy to solve [52].

In this section we will derive the *Linearized* or *Perturbation Kalman Filter* (PKF) [29, 21] that estimates the state of nonlinear dynamic systems by *linearizing* its nonlinearities. Linearization techniques simulate linear behavior locally at a point or along a small interval [52]. The results of this simulation are then extrapolated to the general domain. The extrapolation depends on the direction of the linearity, that is, the direction of the derivatives at a point on a surface. Linearization around a point x means approximating the function at a very small distance from x.

In Section 5.2.1 we will first look at the concepts involved in linearization, after which we linearize the nonlinear system and measurement functions in Section 5.2.2. In Section 5.2.3 we summarize the equations of the PKF, and in Section 5.2.4 we discuss the strengths and weaknesses of the PKF.

5.2.1 Linearization Concepts

In short, to linearize the nonlinear system and measurement functions, we define a *nominal trajectory* and linearize the *perturbations* that occur around that nominal trajectory using a *first order Taylor series* approximation. That is, we assume that we can decompose the nonlinear functions into two components: a known *nominal* component and an unknown *perturbation* component. Let us go into a bit more detail on these linearization concepts.

Trajectories. A *trajectory* [21] is a particular solution of a noisy system. For every time step the trajectory contains an instantiation of the random variables involved. We describe the trajectory by a sequence of vectors. A *nominal* or *reference* trajectory is a trajectory in which the random variables take on their expected values.

Perturbations. The true values of the random variables will not follow the nominal trajectory exactly. There will be small differences between the nominal and true trajectory due to the presence of random noise like system, measurement and initial state noise. We call these small differences *perturbations* [21].

Taylor Series. If the function that we want to linearize is continuously differentiable infinitely many times, then we can represent the influence of perturbations on the trajectory by a *Taylor series expansion* about the nominal trajectory [21]. In general, the Taylor series expansion of a function f around a point x is defined by

$$f(x+\Delta x) = f(x) + f^{(1)}(x)\Delta x + \frac{f^{(2)}(x)}{2!}\Delta x + \ldots + \frac{f^{(n)}(x)}{n!}\Delta x + \ldots, \tag{5.3}$$

where $f^{(i)}$ is the ith derivative of f with respect to x, evaluated at the linearization point x, and where Δx is the perturbation. To linearize a nonlinear function we simply drop the Taylor series expansion terms with derivative order higher than one.

There are some points that need attention when linearizing functions in this way. If the first derivative of the nonlinear function is infinitely large or undefined at the point where we want to linearize, then we can not linearize at that point. Moreover, the perturbations have to be relatively small compared to the higher-order terms in order to result in meaningful linearizations. We can determine the size of the perturbations from the variances of the involved variables. If the variances are small, then we can obtain good approximations by ignoring the higher order terms [21].

5.2.2 Nominal Trajectory Linearization

With the concepts involved in linearization we can perform a linearization of the nonlinear system and measurement model from equations (5.1) and (5.2) around a nominal trajectory.

Linearized System Model. We assume that we can decompose the nonlinear equation for the system state x_k from (5.1) into a linear combination

of a *nominal* component x_{k-1}^{nom} and a *perturbation* component Δx_{k-1},

$$x_k = x_{k-1}^{nom} + \Delta x_{k-1}. \tag{5.4}$$

The nominal state x_{k-1}^{nom} is part of the *nominal trajectory* of the system states. We generate this trajectory by recursively computing the nominal states for all k, where we take x_0^{nom} as initial state,

$$x_k^{nom} = f(x_{k-1}^{nom}). \tag{5.5}$$

As mentioned, the true trajectory of the system state differs from the nominal trajectory. The *state perturbation* component in (5.4) models the difference between the true and nominal state,

$$\begin{aligned} \Delta x_k &= x_k - x_k^{nom} \\ &= f(x_{k-1}) + w_{k-1} - f(x_{k-1}^{nom}), \end{aligned} \tag{5.6}$$

where we substituted (5.1) and (5.5) into the second expression. Notice that this expression for the perturbation is nonlinear. In order to linearize it, we expand the system function $f(x)$ with a Taylor series expansion around the nominal state x_{k-1}^{nom},

$$\begin{aligned} f(x_{k-1}) &= f(x_{k-1}^{nom} + \Delta x_{k-1}) \\ &= f(x_{k-1}^{nom}) + \left.\frac{\partial f(x)}{\partial x}\right|_{x=x_{k-1}^{nom}} \Delta x_{k-1} + \text{h.o.t.} \end{aligned} \tag{5.7}$$

where we used equations (5.3) and (5.4), and where *h.o.t.* is short for *higher-order terms*, which are the terms in the expansion with derivative order higher than one. We substitute this expansion into the definition of the state perturbations (5.6) to obtain a linear expression for the perturbation Δx_k at time k in terms of the perturbation in the state one step earlier Δx_{k-1},

$$\begin{aligned} \Delta x_k &= f(x_{k-1}) + w_{k-1} - f(x_{k-1}^{nom}) \\ &= f(x_{k-1}^{nom}) + \left.\frac{\partial f(x)}{\partial x}\right|_{x=x_{k-1}^{nom}} \Delta x_{k-1} + \text{h.o.t.} + w_{k-1} - f(x_{k-1}^{nom}) \\ &= \left.\frac{\partial f(x)}{\partial x}\right|_{x=x_{k-1}^{nom}} \Delta x_{k-1} + \text{h.o.t.} + w_{k-1} \\ &\approx A_k \Delta x_{k-1} + w_{k-1}, \end{aligned} \tag{5.8}$$

where A_k is the $n \times n$ matrix of partial derivatives of $f(x)$ with respect to x, evaluated at the point of linearization x_{k-1}^{nom},

$$A_k = \left.\frac{\partial f(x)}{\partial x}\right|_{x=x_{k-1}^{nom}}, \tag{5.9}$$

and where w_{k-1} is the system noise. We call a matrix with partial derivatives a *Jacobian matrix*. The Jacobian matrix A_k indicates how the different output variables of $f(x)$ change when the variables of state x change. Since the system function is nonlinear this change can be different from step to step and therefore A_k has to be recomputed for every nominal state.

Notice that we linearized the perturbation from equation (5.6) by dropping the higher order terms of the Taylor series expansion in equation (5.8). Recall that we can only do this meaningfully when the perturbations of the nominal state trajectory are small enough for the higher order terms to be negligible.

Linearized Measurement Model. In the same way as we linearized the system model we can linearize the measurement model from (5.2). We assume that we can express the measurement model as a linear combination of a *nominal measurement* z_k^{nom} and a *measurement perturbation* Δz_k,

$$z_k = z_k^{nom} + \Delta z_k, \tag{5.10}$$

where we define the *nominal measurement trajectory* as sequence of nominal measurements z_k^{nom} for every k as

$$z_k^{nom} = h(x_k^{nom}), \tag{5.11}$$

That is, the nominal measurement trajectory is the sequence of measurements that are ideally made when the system follows the nominal state trajectory. With this definition and the measurement model from equation (5.2) we find the perturbation in the measurement trajectory as

$$\begin{aligned} \Delta z_k &= z_k - z_k^{nom} \\ &= h(x_k) + v_k - h(x_k^{nom}). \end{aligned} \tag{5.12}$$

To linearize this perturbation equation, we first expand the nonlinear measurement function $h(x)$ with a Taylor series expansion around the nominal state x_k^{nom},

$$\begin{aligned} h(x_k) &= h(x_k^{nom} + \Delta x_k) \\ &= h(x_k^{nom}) + \left.\frac{\partial h(x)}{\partial x}\right|_{x=x_k^{nom}} \Delta x_k + \text{h.o.t.} \end{aligned} \tag{5.13}$$

We substitute this into equation (5.12) and drop the higher order terms to express the measurement perturbation as a linear combination of the system

perturbation and measurement noise,

$$\begin{aligned}\Delta z_k &= z_k - z_k^{nom} \\ &= h(x_k) + v_k - h(x_k^{nom}) \\ &= h(x_k^{nom}) + \left.\frac{\partial h(x)}{\partial x}\right|_{x=x_k^{nom}} \Delta x_k + \text{h.o.t.} + v_k - h(x_k^{nom}) \\ &= \left.\frac{\partial h(x)}{\partial x}\right|_{x=x_k^{nom}} \Delta x_k + \text{h.o.t.} + v_k \\ &\approx H_k \Delta x_k + v_k, \end{aligned} \tag{5.14}$$

where

$$H_k = \left.\frac{\partial h(x)}{\partial x}\right|_{x=x_k^{nom}}. \tag{5.15}$$

The matrix H_k is the Jacobian matrix that determines what the contribution of the state perturbation is to the current measurement. Matrix H_k contains the partial derivatives of the measurement function $h(\cdot)$ with respect to x evaluated at the nominal state x_k^{nom}. Notice that we do not need to know the true measurement z_k to compute the measurement perturbation.

For two reasons there will be a difference between a true measurement z_k and the nominal measurement z_k^{nom} at a time k. The true measurements are corrupted by measurement noise v_k and due to differences between a true state and the nominal state at a time step, the measurement prediction $h(x_k)$ does not have to equal $h(x_k^{nom})$. Besides this, the perturbation in the measurement Δz_k differs from the $z_k - z_k^{nom}$, because of the approximation when we drop the higher order terms.

5.2.3 Algorithm

The PKF estimates the state of a nonlinear system by combining a nominal state with an estimate of the state perturbation $\widehat{\Delta x}_k$. It assumes that we can model the system and measurement equations by

$$\begin{aligned} x_k &= x_{k-1}^{nom} + \Delta x_{k-1} \\ z_k &= z_k^{nom} + \Delta z_k, \end{aligned}$$

where the nominal state and measurement trajectories are defined by equations (5.5) and (5.11), and where the system perturbation Δx_{k-1} and measurement perturbation Δz_k are approximated by

$$\begin{aligned} \Delta x_k &\approx A_k \Delta x_{k-1} + w_{k-1} \\ \Delta z_k &\approx H_k \Delta x_k + v_k. \end{aligned}$$

Initialization. The PKF is initialized with the perturbation estimate $\widehat{\Delta x}_0^+$ and uncertainty P_0^+ at time step 0. This initial perturbation estimate will in general be zero.

Prediction. At a new time step, the PKF propagates the perturbation estimate and uncertainty from the last time step to the current using the prediction equations

$$\widehat{\Delta x}_k^- = A_k \widehat{\Delta x}_{k-1}^+ \tag{5.16}$$

$$P_k^- = A_k P_{k-1}^+ A_k^T + Q_{k-1}, \tag{5.17}$$

where A_k is the Jacobian matrix with partial derivatives of the system function $f(x)$ with respect to the state x, evaluated at the nominal state x_{k-1}^{nom},

$$A_k = \left.\frac{\partial f(x)}{\partial x}\right|_{x=x_{k-1}^{nom}}. \tag{5.18}$$

Correction. When a new measurement z_k arrives, the PKF incorporates this measurement into the perturbation estimate using the correction equations

$$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + R_k)^{-1} \tag{5.19}$$

$$\widehat{\Delta x}_k^+ = \widehat{\Delta x}_k^- + K_k(z_k - h(x_k^{nom}) - H_k \widehat{\Delta x}_k^-) \tag{5.20}$$

$$P_k^+ = (I - K_k H_k) P_k^-, \tag{5.21}$$

where H_k is the Jacobian matrix with partial derivatives of the measurement function $h(x)$ with respect to the state x, evaluated at the nominal x_k^{nom},

$$H_k = \left.\frac{\partial h(x)}{\partial x}\right|_{x=x_k^{nom}}. \tag{5.22}$$

Notice that instead of full true and predicted measurements, the PKF incorporates true and predicted measurement perturbations into the state perturbation estimates.

Full State Estimate. Given the optimal prior or posterior state perturbation estimate, we find the optimal full prior or posterior state estimate by combining the prior or posterior perturbation estimate with the nominal state as respectively

$$\hat{x}_k^- = x_k^{nom} + \widehat{\Delta x}_k^-, \tag{5.23}$$

and

$$\hat{x}_k^+ = x_k^{nom} + \widehat{\Delta x}_k^+. \tag{5.24}$$

5.2.4 Remarks

Linearization around a nominal trajectory as in the PKF can work well if the actual trajectory of x_k is close to the nominal trajectory x_k^{nom}. The higher-order terms in the Taylor series expansion can then be ignored and then this method linearizes the nonlinear problem adequately.

Using a nominal trajectory implies that the trajectory of the states is more or less known in advance. This has as computational advantage that the Jacobian matrices A_k and H_k can be computed off-line, before the state estimation begins. They do not depend on the state estimates. This can be an advantage for real-time applications.

Also, since the Jacobians are not coupled to the state estimates, the Kalman gain K_k and the error covariances P_k^- and P_k^+ can be computed off-line. This allows for performance analyses without any real or simulated states and measurements.

However, if the state perturbations are not small, then the PKF can make large estimation errors. In fact, in practice the deviation of the actual trajectory from the nominal trajectory often increases over time [21]. The reason for this is that the nominal trajectory is made without any knowledge of the state estimates. If the true state of the system at a certain point in time starts following a significantly different trajectory than the nominal trajectory, then the following state estimates will have large errors, since the PKF continues using the nominal trajectory. The perturbations can become too large for the PKF to work well.

5.3 Extended Kalman Filter

The main problem of the PKF lies in the fact that it uses the same nominal trajectory throughout the estimation process. The nominal trajectory does not change when state estimates indicate that there might be a large difference between the nominal and true states. The nominal trajectory is static. To prevent the error in the state estimates to become increasingly large, we can make the trajectory about which the perturbations are linearized dynamic. This is exactly the idea of the *Extended Kalman Filter* (EKF) [29, 21], originally called *Kalman-Schmidt Filter* [29]. The nominal trajectory is updated with the most recent state estimates.

5.3.1 Estimated Trajectory Linearization

Instead of evaluating the Taylor series expansions of the system and measurement functions at the nominal trajectories that are made independent of the measurements and state estimates, we can evaluate them at trajectories that are updated with the latest state estimates. If the system state is sufficiently observable, that is, if the measurements provide information

about the state at a high enough frequency, then the deviations between the estimated trajectory and the actual trajectory will stay small [21].

A new estimate of the state of the system is a better reference for the true state than the independently generated nominal state. We can incorporate the better state estimate in the perturbation estimation process. The assumption that the difference between the nominal trajectory and the true trajectory stays small enough to make linear perturbation estimates is hereby stronger justified.

Nominal Trajectory Update. Assume that we have incorporated the last measurement z_{k-1} at time $k-1$ into the state estimate, obtaining the posterior state estimate $\hat{x}_{k-1}^{+}$. Instead of linearizing about the pre-computed nominal state x_{k-1}^{nom} at the next time step, we want to linearize about the new posterior state estimate. This redefinition of the nominal trajectory has consequences for the computation of the prediction and correction of the estimates.

When we have a new state estimate $\hat{x}_{k-1}^{+}$, we recompute the nominal trajectory starting from time step $k-1$, with as initial condition the new state estimate $\hat{x}_{k-1}^{+}$, instead of x_{k-1}^{nom},

$$x_{k-1}^{nom} = \hat{x}_{k-1}^{+} \tag{5.25}$$

$$x_{k}^{nom} = f(x_{k-1}^{nom}). \tag{5.26}$$

Having updated the nominal trajectory with the latest state estimate, the best estimate of the state perturbation $\widehat{\Delta x}_{k-1}^{+}$ at step $k-1$ is zero, since the best estimate of the true state x_{k-1} is our latest state estimate $\hat{x}_{k-1}^{+}$,

$$\widehat{\Delta x}_{k-1}^{+} = \hat{x}_{k-1} - x_{k-1}^{nom} = 0. \tag{5.27}$$

Note that the uncertainty in this estimate of the perturbation does not have to be zero.

Prediction. At time step k we want to predict the new perturbation given the last perturbation estimate and the updated nominal trajectory. According to the prediction equation of the PKF in (5.16), we propagate the perturbation by multiplying the last perturbation estimate with the Jacobian A_k,

$$\begin{aligned} \widehat{\Delta x}_{k}^{-} &= A_k \widehat{\Delta x}_{k-1}^{+} \\ &= 0. \end{aligned} \tag{5.28}$$

The propagated perturbation equals zero, since according to equation (5.27), right after updating the nominal trajectory, the best estimate of the state perturbation is zero. Notice that the Jacobian matrix A_k is evaluated at

the most recent state estimate; it is evaluated at the updated nominal state x_{k-1}^{nom}, which according to equation (5.25) is the posterior state estimate $\hat{x}_{k-1}^{+}$.

According to equation (5.23), the best estimate of the full state is the sum of the predicted perturbation $\widehat{\Delta x}_k^-$ and the updated nominal state x_k^{nom}. Using equations (5.26) and (5.28), we have that

$$\begin{aligned} \hat{x}_k^- &= x_k^{nom} + \widehat{\Delta x}_k^- \\ &= f(\hat{x}_{k-1}^+). \end{aligned} \tag{5.29}$$

We see that we can use the original nonlinear system function $f(\cdot)$ instead of the linearized perturbation to estimate the prior state estimate $\hat{x}_k^-$.

Correction. The nominal measurement trajectory depends on the nominal state trajectory. Since we updated the latter with the latest state estimate, we can use it to update the nominal measurement trajectory. The measurement trajectory is computed in the same way as before, only with the updated state trajectory as input. For all k starting from the current time,

$$z_k^{nom} = h(x_k^{nom}).$$

With this and with (5.28), we rewrite the correction equation of the PKF from equation (5.20) into

$$\begin{aligned} \widehat{\Delta x}_k^+ &= \widehat{\Delta x}_k^- + K_k(z_k - z_k^{nom} - H_k \widehat{\Delta x}_k^-) \\ &= K_k(z_k - h(x_k^{nom})) \\ &= K_k(z_k - h(\hat{x}_k^-)). \end{aligned} \tag{5.30}$$

Notice that the Jacobian matrix H_k and measurement function $h(\cdot)$ are evaluated at the nominal state x_k^{nom}. According to (5.26) and (5.25) this equals $f(\hat{x}_{k-1}^+)$, which according to (5.29) equals the prior state estimate $\hat{x}_k^-$, which is the latest state estimate.

According to equation (5.24), we obtain the best full posterior state estimate by summing the updated nominal state with the computed posterior state perturbation. Using (5.26), (5.25), and (5.30), we obtain

$$\begin{aligned} \hat{x}_k^+ &= x_k^{nom} + \widehat{\Delta x}_k^+ \\ &= f(\hat{x}_{k-1}^+) + K_k(z_k - h(\hat{x}_k^-)) \\ &= \hat{x}_k^- + K_k(z_k - h(\hat{x}_k^-)). \end{aligned} \tag{5.31}$$

We see that we do not need to explicitly compute the state perturbation $\widehat{\Delta x}_k^+$ to compute the posterior state estimate.

With this result we have found a way to compute both prior and posterior state estimates, without explicitly computing the state and measurement perturbations, while incorporating the latest state estimates in the trajectories.

5.3.2 Algorithm

The EKF uses the equations that we have derived to estimate the state of nonlinear systems with nonlinear measurements. The structure of the EKF closely resembles the structure of the LKF from Section 4.7.

Initialization. The EKF is initialized with the posterior state estimate $\hat{x}_0^+$ and uncertainty P_0^+ at time step 0.

Prediction. At every time step, the EKF propagates the state and uncertainty of the system at the previous time step to the current using the prediction equations

$$\hat{x}_k^- = f(\hat{x}_{k-1}^+) \tag{5.32}$$
$$P_k^- = A_k P_{k-1} A_k^T + Q_{k-1}, \tag{5.33}$$

where the Jacobian matrix A_k contains the partial derivatives of system function $f(\cdot)$ with respect to state x, evaluated at the posterior state estimate $\hat{x}_{k-1}^+$ of the last time step,

$$A_k = \left.\frac{\partial f(x)}{\partial x}\right|_{x=\hat{x}_{k-1}^+}. \tag{5.34}$$

Correction. The EKF corrects the prior state estimate with a full measurement z_k by means of the correction equations

$$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + R_k)^{-1} \tag{5.35}$$
$$\hat{x}_k^+ = \hat{x}_k^- + K_k(z_k - h(\hat{x}_k^-)) \tag{5.36}$$
$$P_k^+ = (I - K_k H_k) P_k^-, \tag{5.37}$$

where the Jacobian matrix H_k contains the partial derivatives of the measurement function $h(\cdot)$ with respect to the state x, evaluated at the prior state estimate $\hat{x}_k^-$,

$$H_k = \left.\frac{\partial h(x)}{\partial x}\right|_{x=\hat{x}_k^-}. \tag{5.38}$$

5.3.3 Remarks

The Jacobian matrices A_k and H_k are evaluated at the most recent state estimates. Since the state estimates are not known in advance, the Jacobians can not be computed off-line as is the case with the PKF. In the EKF case, the Jacobians have to be computed on-line, since they depend on the state estimates. This results in higher computational costs.

Although the computational costs are higher, the state estimates will be better than the estimates from the PKF, since the nominal trajectory is updated on-line with the latest state estimates. This allows the EKF to make better nominal trajectories, making deviations from the nominal trajectory smaller. This allows the EKF to more accurately predict and correct its state estimates.

The propagation and update equations for updating the error covariance depend on the state estimates. Thus, the covariance and gain matrices can not be analyzed without knowledge about the state estimates, and thus true measurements. In the LKF and PKF we could in fact analyze error and Kalman gain behavior without having state estimates.

The EKF has been shown to be successful in many practical nonlinear applications [29, 21]. It can in particular be used in applications where the model of the system is well described by a linear model, but in which there are some uncertain parameters in the system and measurement models. By treating these parameters as additional state variables the problem becomes nonlinear; the parameters then have to be estimated on-line.

5.4 Iterated Extended Kalman Filter

The EKF linearizes the nonlinear system and measurement function, redefining the nominal trajectories using the latest state estimates once. When there are significant nonlinearities, it can be beneficial to iterate the nominal trajectory redefinition a number of times using the new nominal trajectory. The idea of the *Iterated Extended Kalman Filter* (IEKF) [29] is to use all information in a measurement by *repeatedly adjusting* the nominal state trajectory.

We can see that this can be beneficial by looking at how the EKF incorporates a measurement. If we have a measurement that we want to incorporate in a state estimate, we should use the best state estimate available. At a time k when there is a new measurement available, the best estimate so far will be the prior state estimate $\hat{x}_k^-$ at that time. However, at the point that the EKF has computed the posterior estimate, the prior state estimate is no longer the best estimate at time k. Instead, the new posterior estimate is the best estimate of the state at time k. Since we should use the best state estimate to incorporate the measurement, we should use this posterior estimate instead of the prior estimate.

5.4.1 Repeated Estimated Trajectory Update

The IEKF repeatedly redefines the nominal trajectory to find the best state estimate to use for including a measurement. It does this by computing the Kalman Gain K_k and an *intermediate* posterior state estimate $\hat{x}_k^i$, where i is the iteration number. Starting from $\hat{x}_k^0 = \hat{x}_k^-$, the IEKF repeatedly performs the calculations

$$\begin{aligned} K_k &= P_k^- H_k^T (H_k P_k^- H_k^T + R_k)^{-1} \\ \hat{x}_k^{i+1} &= \hat{x}_k^- + K_k(z_k - h(\hat{x}_k^i) - H_k(\hat{x}_k^- - \hat{x}_k^i)). \end{aligned} \tag{5.39}$$

In this, the Jacobian matrix H_k is evaluated at the most recent intermediate state estimate $\hat{x}^i$. After a fixed number of iterations or when the intermediate state estimate $\hat{x}_k^i$ does not differ with more than a certain threshold from $\hat{x}_k^{i-1}$, the IEKF sets the posterior state estimate to $\hat{x}_k^i$ and it computes the posterior uncertainty P_k^+ in that state estimate using the latest evaluation of K_k and H_k.

5.4.2 Remarks

Notice that although the IEKF computes intermediate state estimates, it does not compute intermediate state estimate uncertainties. It only computes the uncertainty in the state estimate once it has found the most accurate intermediate state estimate.

Although the computations involved in the IEKF are larger than in the EKF, the posterior state estimates will be better because of the re-evaluation of the measurement function h and the Jacobian H_k. This will again improve future state estimates, because the nominal trajectories will be better [29].

5.5 Additional Dependencies Revisited

As we discussed in Section 4.9 the system model can depend on more than just the last state and system noise. Additional dependencies like drift and external input commonly appear in practical applications. In the previous chapter we derived equations for the LKF that we can apply in these situations. In this section we discuss how we can generalize the EKF equations. We will do this for the EKF, since it is generally more applicable than the PKF, and since the extension from generalized EKF to generalized IEKF is straightforward.

5.5.1 Dynamic Nonlinear Systems Revisited

Generalized Nonlinear System Model. Assume that we have a system model that depends on system noise w_{k-1}, and a system function f that depends on $n-1$ parameters $u_{i,k}$. Instead of treating the system noise w_{k-1}

as a separate term in the system model, we can mathematically include it into the description of the system function f. We then obtain the generalized system model

$$x_k = f(u_{1,k-1}, \ldots, u_{n-1,k-1}) + w_{k-1} \tag{5.40}$$
$$= f^*(u_{1,k-1}, \ldots, u_{n-1,k-1}, u_{n,k-1}) \tag{5.41}$$

where we assume that the n parameters u_i are Gaussian distributed with mean $\hat{u}_{i,k}$ and covariance $U_{i,k}$,

$$u_i \sim N(\hat{u}_{i,k}, U_{i,k}), \tag{5.42}$$

for $1 \leq i \leq n$, and where we assume that the parameter vectors are independent of one another.

We can include the noise in the system function, since for every variable in the result of f there is a corresponding variable in the system noise vector w_{k-1}. This noise variable can thus just as well be added to the corresponding result variable inside the function f. That is exactly what f^* does. In the following we denote f when we refer to f^*.

Generalized Nonlinear Measurement Model. In the same way, assume that we have a measurement model that depends on measurement noise v_k, and a measurement function h that depends on $m-1$ parameters $y_{j,k}$. As with the system noise and the system function, we can include the measurement noise in the measurement function. We then obtain the generalized measurement model

$$z_k = h(y_{1,k}, \ldots, y_{m-1,k}) + v_{k-1} \tag{5.43}$$
$$= h(y_{1,k}, \ldots, y_{m-1,k}, y_{m,k}), \tag{5.44}$$

where we again assume that the m parameters y_j are Gaussian distributed with mean $\hat{y}_{j,k}$ and covariance $Y_{j,k}$,

$$y_j \sim N(\hat{y}_{j,k}, Y_{j,k}), \tag{5.45}$$

for $1 \leq j \leq m$, and where we assume that the vectors are independent of one another.

5.5.2 Generalized EKF Prediction

Realize that the best prediction $\hat{x}_k^-$ of the state is given by the system function $f(\cdot, \ldots, \cdot)$ with as parameters the best estimates of the parameters $u_{i,k}$. The best estimates of these parameters are the means $\hat{u}_{i,k}$ of the Gaussians describing their distributions. The uncertainty P_k^- in the best prediction depends on the uncertainty in the parameters. The Jacobian

matrices with the derivatives of the system function with respect to the parameters determine the contribution of each of the parameters to the uncertainty of the state estimate. This leads us to form the generalized EKF prediction equations as

$$\hat{x}_k^- = f(\hat{u}_{1,k-1}, \ldots, \hat{u}_{n,k-1}) \tag{5.46}$$

$$P_k^- = A_{u_1,k} U_{1,k-1} A_{u_1,k}^T + \ldots + A_{u_n,k} U_{n,k-1} A_{u_n,k}^T \tag{5.47}$$

$$= \sum_{i=1}^{n} A_{u_i,k} U_{i,k-1} A_{u_i,k}^T, \tag{5.48}$$

where $A_{u_i,k}$ is the Jacobian matrix with partial derivatives of system function $f(\cdot, \ldots, \cdot)$ with respect to u_i, evaluated at $u_1 = \hat{u}_{1,k-1}, \ldots, u_n = \hat{u}_{n,k-1}$, for $1 \leq i \leq n$,

$$A_{u_i,k} = \left. \frac{\partial f(u_1, \cdots, u_n)}{\partial u_i} \right|_{u_1 = \hat{u}_{1,k-1}, \ldots, u_n = \hat{u}_{n,k-1}}. \tag{5.49}$$

Commonly the system function f will depend on the last posterior state estimate $\hat{x}_{k-1}^+$, though this does not have to be the case.

5.5.3 Generalized EKF Correction

To correct a prediction, the EKF predicts what the true measurement will be. Realize that estimating the measurement $\hat{z}_k$ is in fact a different estimation process without correction and not depending on previous measurements. Therefore, we predict the best measurement estimate similar to the way we predicted the best state estimate.

The best estimate of the measurement $\hat{z}_k$ is given by the measurement function $h(\cdot, \ldots, \cdot)$ with as parameters $y_{i,k}$ the best estimates of the parameters, which are the means of their distributions $\hat{y}_{i,k}$. The uncertainty in this measurement estimate depends on the uncertainties in the parameters. The Jacobian matrices with the derivatives of the measurement function with respect to the parameters determine how much uncertainty of each parameter contributes to the uncertainty in the measurement estimate. This leads us to equations for the estimated, or predicted, measurement $\hat{z}_k$ with uncertainty Z_k,

$$\hat{z}_k = h(\hat{y}_{1,k-1}, \ldots, \hat{y}_{m,k-1}) \tag{5.50}$$

$$Z_k = H_{y_1,k} Y_{1,k} H_{y_1,k}^T + \cdots + H_{y_m,k} Y_{m,k} H_{y_m,k}^T \tag{5.51}$$

$$= \sum_{i=1}^{m} H_{y_i,k} Y_{i,k} H_{y_i,k}^T, \tag{5.52}$$

where $H_{y_i,k}$ is the Jacobian matrix with partial derivatives of the measurement function $h(\cdot, \ldots, \cdot)$ with respect to y_i, evaluated at $y_1 = \hat{y}_{1,k-1}, \ldots, y_m =$

$\hat{y}_{m,k-1}$, for $1 \leq i \leq m$,

$$H_{y_i,k} = \left. \frac{\partial h(y_1, \cdots, y_m)}{\partial y_i} \right|_{y_1=\hat{y}_{1,k-1},\ldots,y_m=\hat{y}_{m,k-1}} . \tag{5.53}$$

The correction of the prediction of the states uses the Kalman gain to determine to which extend the measurement residual $z_k - \hat{z}_k$ has to be included in the new state estimate. The Kalman gain weights the uncertainty in the state estimate against the uncertainty in the measurement residual. It hereby uses the Jacobian with partial derivatives of the measurement function with respect to x. In order to use the measurement model to correct the state estimate, the measurement model must be related to the system model. Thus, the state x_k must always be a parameter of the measurement function. Assume that the parameter $y_{i,k}$ of the measurement function that corresponds to the state x_k is $i = j$. We then form the generalized equations for the Kalman gain and the posterior state estimate and uncertainty as

$$K_k = P_k^- H_{y_j,k}^T (Z_k)^{-1} \tag{5.54}$$

$$\hat{x}_k^+ = \hat{x}_k^- + K_k(z_k - \hat{z}_k) \tag{5.55}$$

$$P_k^+ = (I - K_k H_{y_j,k}) P_k^- . \tag{5.56}$$

5.6 Related Work

Besides the KF extensions that we have discussed so far in this chapter, alternative extensions have been proposed throughout the years. Here we mention the key ideas of three of these shortly.

The *Unscented Kalman Filter* [47] uses an *unscented transformation* to perform state estimation of nonlinear systems without linearizing the system and measurement models. This transformation uses a set of carefully deterministically chosen weighted samples that parameterize the mean and covariance of the belief. The system function is applied to each sample, which results in a group of transformed points. The mean and covariance of this group of points are the propagated mean and covariance. Since there is no linearization involved in the propagation of the mean and covariance, the Jacobians of the system and measurement model do not have to be calculated. This makes the Unscented Kalman Filter practically attractive.

Particle filters [20] are an alternative technique for state estimation. Particle Filters represent the complete posterior distribution of the states. Therefore they can deal with any nonlinearities and noise distributions. Particle filters have been combined with the Unscented Kalman Filter in the *Unscented Particle Filter* [46].

The *Ensemble Kalman Filter* [17] allows for states with huge amounts of variables. Due to the computations involved in propagating the error covariance in the KF, the dimension of the states is restricted to no more

than a few thousand state variables. In for example the prediction of the state of an ocean, the number of variables can increase to millions. The Ensemble Kalman Filter is a Monte Carlo-based alternative to the standard KF, in which an ensemble of model trajectories is integrated, and in which the statistics of the ensemble are used to estimate the errors of the models.

5.7 Summary

We can extend the LKF that estimates the state of linear systems that are governed by linear system and measurement models to nonlinear systems. Nonlinear systems are governed by a nonlinear system model and a nonlinear measurement model.

In order to use the ideas and techniques from the LKF chapter, we use linearization techniques to linearize the nonlinear system and measurement models. By defining nominal state and measurement trajectories, and by defining state and measurement perturbations, we linearize the nonlinear models by dropping the higher order terms of a Taylor series expansion.

Using the Perturbation KF, we estimate the perturbations around the nominal trajectories. Advantages of this approach are the possibility to off-line compute needed Jacobian matrices, and perform off-line analysis of Kalman gain and state uncertainty behavior. However, if the perturbations do not stay small, the PKF will make large estimation errors due to the fact that it does not update the nominal trajectory with information from the true system.

We deal with this problem by updating the nominal trajectory with the latest information about the true state of the system. The Extended KF incorporates a new state estimate into the nominal trajectory once a measurement has been incorporated. This allows the EKF to make future state and measurement predictions more accurately. However, the improved accuracy of the nominal trajectory comes at the price of loss of off-line analysis possibilities, since in the EKF state estimates are used in the computation of the nominal trajectories.

We can reduce the errors that can occur due to large nonlinearities in the system further by iteratively updating the nominal trajectory with the best estimate of the state of the system. The idea implemented in the Iterated Extended KF is that a measurement should be incorporated into the best estimate of the system at all times. The IEKF implements this idea by iteratively computing an intermediate state estimate, until it notices no further improvement.

In the light of possible additional system and measurement dependencies we can generalize the EKF equations. Noticing that we can include noise descriptions into the nonlinear functions, and that the prediction of measurements is in fact an estimation process on its own, we can create a

general framework for state estimation in the case of independent random parameter dependencies.

Chapter 6

System and Measurement Models

As we have seen in Chapters 4 and 5, the KF uses a system model and a measurement model. Depending on the application area, these models differ. In the robot localization problem from Chapter 2 we have a sensing robot driving around in an environment trying to find out its location. In this chapter we will develop models that model different aspects of a navigating robot, in particular driving and sensing aspects. In following chapters we will use these models in KF instances.

In Section 6.1 we derive a system model that models the location of a robot using relative displacement measurements. In Section 6.2 we describe a GPS-like sensor that senses the full location of the robot directly. In Section 6.3 we derive a measurement model that models measurements made from the location of landmarks from the robot's point of view. We end the chapter with some general modeling remarks in Section 6.4.

6.1 Driving System

Mobile robots have a driving system allowing them to move around in an environment. Commonly the driving system consists of wheels, but a legged, tracked, or snake shaped vehicle is a possibility. Legged vehicles have the potential of traveling over rough terrain where wheeled or tracked vehicles can not go. Snake-shaped vehicles may be able to move over loose sand and rocks which might be difficult for other systems. However, most robot vehicles use wheels to move themselves. Vehicles with wheels are less complex, while suitable for navigation in the daily environments of many people.

Wheel Encoders. Wheels and other driving mechanics are controlled by motors. These motors can make wheels move either forward or backward. By means of *wheel encoder sensors* we can monitor the number of wheel rotations of a specific wheel [11]. We can use the wheel encoder readings from these sensors together with knowledge about the diameter of the wheel to estimate its displacement. Wheel encoder sensors provide relative position measurements.

In general we are not interested in the relative displacement of each wheel, but in the displacement of the robot as a result of the displacements of all wheels in the drive system together. Depending on the configuration of the guidance system, the conversion between output from the wheel encoders from the different wheels and the relative displacement of the robot is different.

6.1.1 The Perfect System Model

In order to model the way in which a drive system changes the location of a robot, we first take a look at the ideal situation in which there is no noise in the environment. Once we have a noise-free system model, we extend the model to take into account noises that influence the system.

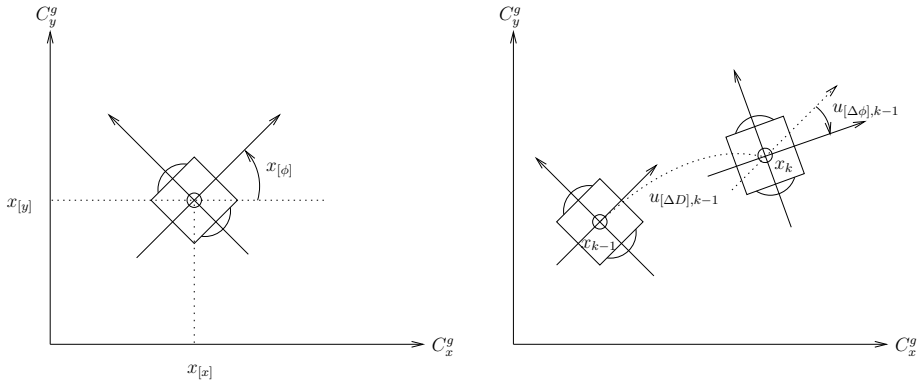
In our case, the system model that we are looking for describes how the *location* of the robot changes due to the dynamics of the guidance system. Since we can monitor the relative dynamics of the guidance system with wheel encoders, we are looking for a system model that given the *last location* and a *relative displacement* determines the new location. In order to derive this model, let us formalize a location and a relative displacement more precisely.

Location. Figure 6.1(a) shows schematically how we define the location of a robot. The location of a robot is defined in a global coordinate system C^g and consists of the x and y coordinates of the center of the robot, the *center point*, and the orientation ϕ of the robot in that general coordinate system at a certain time k,[1]

$$x_k = \begin{bmatrix} x_{[x],k} \\ x_{[y],k} \\ x_{[\phi],k} \end{bmatrix}. \tag{6.1}$$

Relative Displacement. The relative displacement is the displacement of the center point of the robot over a certain time. We assume that we can convert the separate displacements from the wheels of the robot into

[1] For readability purposes, we refer to an element e of a vector v_k by subscripting the element name, instead of subscripting the element index in the vector. E.g, we use $v_{[e],k}$ instead of $v_{1,k}$.



(a) (b)

Figure 6.1: Concepts of (a) the location x and (b) the relative measurement u_{k-1} at a certain time.

a relative displacement measurement u_k of the center point of the robot using some function $j(\cdot)$ that takes as argument raw data d_k from the wheel encoders. We then define the relative displacement as

$$u_k = j(d_k) = \begin{bmatrix} u_{[\Delta D],k} \\ u_{[\Delta\phi],k} \end{bmatrix}, \tag{6.2}$$

where $u_{[\Delta D],k}$ is the distance over which the center point of the robot travels, and $u_{[\Delta\phi],k}$ is the change in orientation during the travelling. Figure 6.1(b) shows these concepts schematically.

Depending on the *number of degrees of freedom* of a driving system, the complexity of the conversion model $j(\cdot)$ increases [4]. In the following, we assume that the guidance system dependent function $j(\cdot)$ is given, and that we thus directly have access to the relative displacement u_k of the robot.

6.1.1.1 Location Update

Given the relative information u_{k-1}, and given the last location of the robot x_{k-1}, we want to express the current location of the robot. That is, we look for the function f,

$$x_k = f(x_{k-1}, u_{k-1}) = \begin{bmatrix} f_x(x_{k-1}, u_{k-1}) \\ f_y(x_{k-1}, u_{k-1}) \\ f_\phi(x_{k-1}, u_{k-1}) \end{bmatrix}, \tag{6.3}$$

consisting of f_x and f_y to update the x and y coordinates of the location, and f_ϕ to update the orientation of the location.

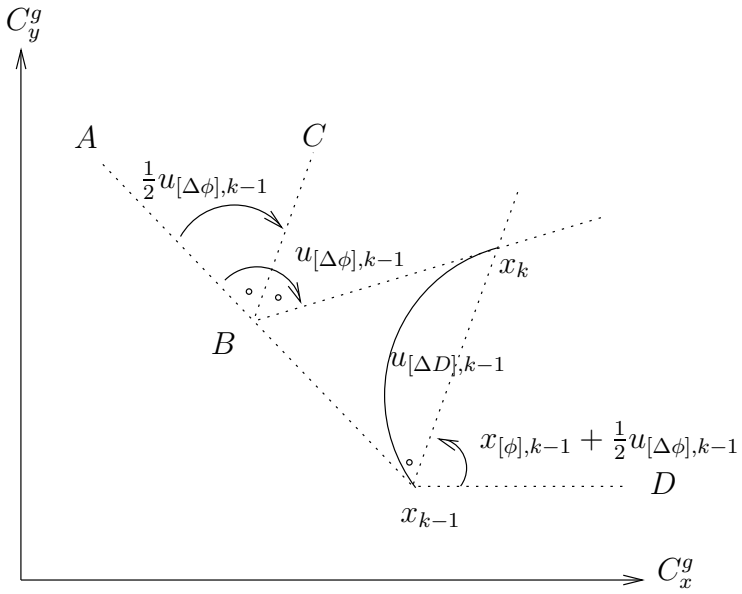


Figure 6.2: Schematic overview of the computation of the new location. The solid line indicates the path the robot drives from x_{k-1} to x_k.

Coordinates. We derive expressions for f_x and f_y by means of Figure 6.2. The figure shows the path that the robot drives from x_{k-1} to x_k. The robot starts at x_{k-1} facing direction A; thus, $x_{[\phi],k-1} = \angle A x_{k-1} D$. The robot then drives a path of length $u_{[\Delta D],k-1}$ to x_k, whereby the orientation changes $u_{[\Delta\phi],k-1} = \angle A B x_k$. It can be shown that $\angle A x_{k-1} x_k = (1/2) u_{[\phi],k-1}$, [48], which gives us

$$\angle x_k x_{k-1} D = \angle A x_{k-1} D - \angle A x_{k-1} x_k = x_{[\phi],k-1} + \frac{1}{2} u_{[\Delta\phi],k-1}.$$

Besides this angle, we need to know the length of $x_{k-1}x_k$ in order to determine the coordinates $x_{[x],k}$ and $x_{[y],k}$. In general this length is not known, though a common way to deal with this is to use $u_{[\Delta D],k-1}$ as an approximation [48]. We then calculate the coordinates of x_k as

$$x_{[x],k} = f_x(x_{k-1}, u_{k-1}) = x_{[x],k-1} + u_{[\Delta D],k-1} \cdot \cos(x_{[\phi],k-1} + \frac{u_{[\Delta\phi],k-1}}{2}) \tag{6.4}$$

$$x_{[y],k} = f_y(x_{k-1}, u_{k-1}) = x_{[y],k-1} + u_{[\Delta D],k-1} \cdot \sin(x_{[\phi],k-1} + \frac{u_{[\Delta\phi],k-1}}{2}). \tag{6.5}$$

The interested reader can find equations for the case where the robot is assumed to drive in a circular path in [48].

Orientation. The expression for f_ϕ is trivially found. If the relative displacement information indicates that the orientation of the robot changes

by $u_{[\Delta\phi]}$, then the updated orientation of the robot is the sum of this and the last orientation of the robot. Thus,

$$x_{[\phi],k} = f_\phi(x_{k-1}, u_{k-1}) = x_{[\phi],k-1} + u_{[\Delta\phi],k-1}. \tag{6.6}$$

6.1.2 The Noisy System Model

In the derivation of the system model so far we assumed that the driving of the robot was not disturbed by any noise sources. We assumed that the wheel encoders worked perfectly; we assumed that we could perfectly map wheel encoder countings to traveled wheel distance; we assumed that there were no influences from the environment that disturbed the driving, and that thus our approximated system function perfectly describes the movement of the robot over time. In the real world most of these assumptions do not hold, and thus we will introduce noise in the system model.

6.1.2.1 Noise Modeling

Relative Displacement Noise. Assume that we can model the noise in the relative displacement by a random noise vector q_k being zero-mean independent Gaussian distributed,

$$q_k \sim N(\hat{q}_k, U_k), \tag{6.7}$$

where

$$\begin{aligned}
\hat{q}_k &= \begin{bmatrix} 0 \\ 0 \end{bmatrix} \\
U_k &= E[(q_k - \hat{q}_k)(q_k - \hat{q}_k)^T] \\
&= \begin{bmatrix} \sigma^2_{q_{[\Delta D],k}} & \sigma_{q_{[\Delta\phi],k}}\sigma_{q_{[\Delta D],k}} \\ \sigma_{q_{[\Delta D],k}}\sigma_{q_{[\Delta\phi],k}} & \sigma^2_{q_{[\Delta\phi],k}} \end{bmatrix}.
\end{aligned}$$

In the covariance matrix U_k, the off-diagonal elements are zero, since we assume that the noise sources are independent of each other. We adjust the noise-free relative displacement from equation (6.2) by including the term q_k to obtain the noisy relative displacement,

$$\begin{aligned}
u_k &= j(d_k) + q_k \\
&= \begin{bmatrix} j_{\Delta D}(d_k) \\ j_{\Delta\phi}(d_k) \end{bmatrix} + \begin{bmatrix} q_{[\Delta D],k} \\ q_{[\Delta\phi],k} \end{bmatrix}.
\end{aligned} \tag{6.8}$$

Since we include the random noise term q_k in the model, the relative information vector u_k also becomes a random vector. If we assume that $j(\cdot)$ is deterministic, the uncertainty in u_k equals the uncertainty in the noise term q_k.

System Noise. We can model noise sources that are not directly related to the relative displacement with a system noise term. Assume that we can model these noise sources with a term w_k, that is independent Gaussian distributed,

$$w_k \sim N(\hat{w}_k, Q_k), \tag{6.9}$$

where

$$\begin{aligned}
\hat{w}_k &= \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \\
Q_k &= E[(w_k - \hat{w}_k)(w_k - \hat{w}_k)^T] \\
&= \begin{bmatrix} \sigma^2_{w_{[x],k}} & \sigma_{w_{[y],k}}\sigma_{w_{[x],k}} & \sigma_{w_{[\phi],k}}\sigma_{w_{[x],k}} \\ \sigma_{w_{[x],k}}\sigma_{w_{[y],k}} & \sigma^2_{w_{[y],k}} & \sigma_{w_{[\phi],k}}\sigma_{w_{[y],k}} \\ \sigma_{w_{[x],k}}\sigma_{w_{[\phi],k}} & \sigma_{w_{[y],k}}\sigma_{w_{[\phi],k}} & \sigma^2_{w_{[\phi],k}} \end{bmatrix}.
\end{aligned}$$

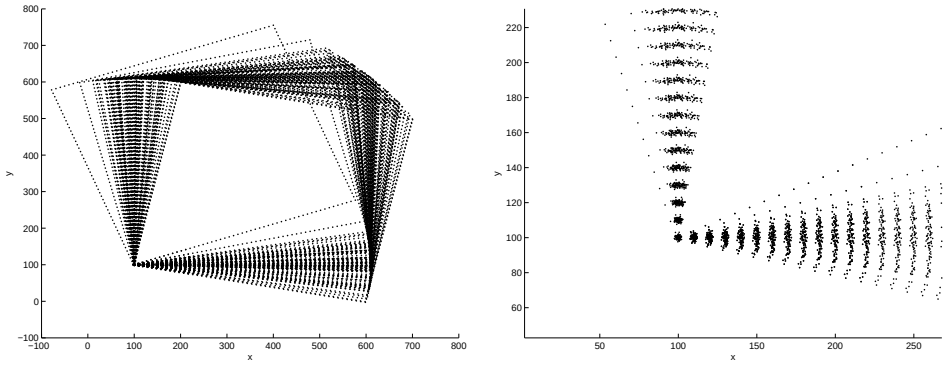
We assume that the noise sources in the location elements are independent of each other, and thus the off-diagonal elements of the covariance matrix Q_k are zero. The diagonal elements of the covariance matrix contain the variances of the noise vector w_k. Including this noise term in the noise-free system model from equation (6.10) we obtain

$$\begin{aligned}
x_k &= f(x_{k-1}, u_{k-1}) + w_{k-1} \\
&= \begin{bmatrix} f_x(x_{k-1}, u_{k-1}) \\ f_y(x_{k-1}, u_{k-1}) \\ f_\phi(x_{k-1}, u_{k-1}) \end{bmatrix} + \begin{bmatrix} w_{[x],k-1} \\ w_{[y],k-1} \\ w_{[\phi],k-1} \end{bmatrix}.
\end{aligned} \tag{6.10}$$

Instead of a deterministic location x_k, the location is now a random vector. The variance of the location grows with every time step, since the location x_k at step k depends on the location x_{k-1} one step earlier and since at every time step the system noise increases the variance. Besides this, the relative displacement vector u_{k-1} is also a random vector with influence on the uncertainty in x_k.

6.1.2.2 Remarks

Although the modeled noise sources are Gaussian, the distribution of the locations does not have to be Gaussian. This can be seen from Figure 6.3. This figure shows the locations computed in 100 runs simulating a robot driving in a square. At each run, the model was initialised with a slightly different starting position, distributed according to the system noise. Thus the initial uncertainty in the position was Gaussian distributed. Figure 6.3(a) shows the full runs, Figure 6.3(b) zooms in on the first and last steps. From this last figure we can see that the locations at each step are



(a) Full trajectories.

(b) Zoomed trajectories.

Figure 6.3: 100 true x and y trajectories of robot driving squares given initial location with some uncertainty. Runs start at (100,100) and are counter-clockwise.

not Gaussian distributed. This non-Gaussianness is caused by the nonlinear relationship between the x and y coordinates on one hand and the orientation and orientation change on the other.

Notice in Figure 6.3(a) that given uncertainty in the initial location only, after four turns of 90 degrees, the model computes the same location as it started with. If there would be no uncertainty at all in the orientation of the location, then the uncertainty in the x and y coordinates would stay the same throughout the run. However, if there is uncertainty in the orientation, then the uncertainty in the x and y coordinates changes with every step, due to the fact that x and y depend on the orientation. In particular, the uncertainty in the y increases until the first turn; then the uncertainty in the x increases until the next turn; following that, the uncertainty in the y decreases again; finally, the uncertainty in the x decreases again, to end up with the same uncertainty in the location as the model started with. This makes sense, since if there is no uncertainty in the turning, then after four turns of 90 degrees, the orientation will be the same as initially, no matter what orientation the model was initialized with.

6.2 Full State Sensor

In this section we model a full state sensor. A full state sensor measures the full state of the system, in this case the location of the robot. In practical applications we may be able to construct a *virtual sensor*, that combines the information from different other sensors with partial state information into a measurement that measures the full state. An example of this could be a combined GPS-compass system.

6.2.1 The Perfect Measurement Model

Assuming no noise, the perfect measurement z_k of all the variables of a state x_k is simply a vector containing for each state variable a variable that takes on the value of the corresponding state variable. Thus, in our localization case, the measurement z_k is

$$z_k = \begin{bmatrix} z_{[x],k} \\ z_{[y],k} \\ z_{[\phi],k} \end{bmatrix}. \tag{6.11}$$

We model the measurement model that given the location of a robot x_k returns the measurement of the full state sensor, as

$$z_k = h(x_k), \tag{6.12}$$

where function $h(\cdot)$ is the measurement function relating a state to a measurement, and which in this case simply is

$$h(x_k) = \begin{bmatrix} h_x(x_k) \\ h_y(x_k) \\ h_\phi(x_k) \end{bmatrix} = \begin{bmatrix} x_{[x],k} \\ x_{[y],k} \\ x_{[\phi],k} \end{bmatrix}. \tag{6.13}$$

6.2.2 The Noisy Measurement Model

Assuming that there are Gaussian distributed noise sources corrupting the measurements, we model these with the independent Gaussian noise vector v_k,

$$v_k \sim N(\hat{v}_k, R_k), \tag{6.14}$$

where

$$\begin{aligned} \hat{v}_k &= \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \\ R_k &= E[(v_k - \hat{v}_k)(v_k - \hat{v}_k)^T] \\ &= \begin{bmatrix} \sigma^2_{v_{[x],k}} & \sigma_{v_{[y]},k}\sigma_{v_{[x]},k} & \sigma_{v_{[\phi]},k}\sigma_{v_{[x]},k} \\ \sigma_{v_{[x],k}}\sigma_{v_{[y],k}} & \sigma^2_{v_{[y]},k} & \sigma_{v_{[\phi]},k}\sigma_{v_{[y]},k} \\ \sigma_{v_{[x],k}}\sigma_{v_{[\phi],k}} & \sigma_{v_{[y]},k}\sigma_{v_{[\phi],k}} & \sigma^2_{v_{[\phi]},k} \end{bmatrix}. \end{aligned}$$

With this measurement noise vector we can update the noise-free measurement model from equation (6.12) to incorporate noise,

$$z_k = h(x_k) + v_k. \tag{6.15}$$

Notice that z_k is a random vector, since the measurement noise v_k is a Gaussian vector. Besides the influence of the measurement noise, uncertainty in the location x_k can also have its influence on the uncertainty in z_k. If the uncertainty in x_k is Gaussian distributed, then z_k is also Gaussian distributed.

6.3 Landmark Detection Sensor

In this section we model a sensor that can detect a landmark. The concept of a landmark should be read in a wide sense. In this section we only assume that a landmark has an x and y coordinate and an orientation in the same coordinate system as the robot's location. Contrarily to the full state sensor from the previous section, the landmark sensor makes measurements from the robot's point of view using for example a vision system.

6.3.1 The Perfect Measurement Model

Assuming no noise corrupting the measurements, we first define a landmark location and measurement, after which we derive the measurement model.

Landmark Location. We assume that the robot has a map with the locations of landmarks in the global coordinate system C^g of which the location of the robot x_k is also an element. The location l_k of a landmark is defined as

$$l_k = \begin{bmatrix} l_{[x],k} \\ l_{[y],k} \\ l_{[\phi],k} \end{bmatrix}, \tag{6.16}$$

where $l_{[x],k}$ and $l_{[y],k}$ are the coordinates and $l_{[\phi],k}$ is the orientation of the landmark in the global coordinate system C^g.

Landmark Measurement. We define the measurement z_k as the location of a landmark from the viewpoint of the robot, that is,

$$z_k = \begin{bmatrix} z_{[x],k} \\ z_{[y],k} \\ z_{[\phi],k} \end{bmatrix}, \tag{6.17}$$

where $z_{[x],k}$ and $z_{[y],k}$ are the coordinates and $z_{[\phi],k}$ is the orientation in the coordinate system C^r having as origin the location of the robot.

Figure 6.4 schematically shows the relation between the robot at location x, a landmark l, and a measurement z. The measurement model that we are looking for gives the location of the landmark from the robot's point of view, given the location of the robot x_k, and given the location of the landmark l_k, both in global coordinates. That is,

$$z_k = h(x_k, l_k) = \begin{bmatrix} h_x(x_k, l_k) \\ h_y(x_k, l_k) \\ h_\phi(x_k, l_k) \end{bmatrix}, \tag{6.18}$$

where $h(\cdot, \cdot)$ is the measurement function that relates the robot's location and the landmark's location to a measurement.

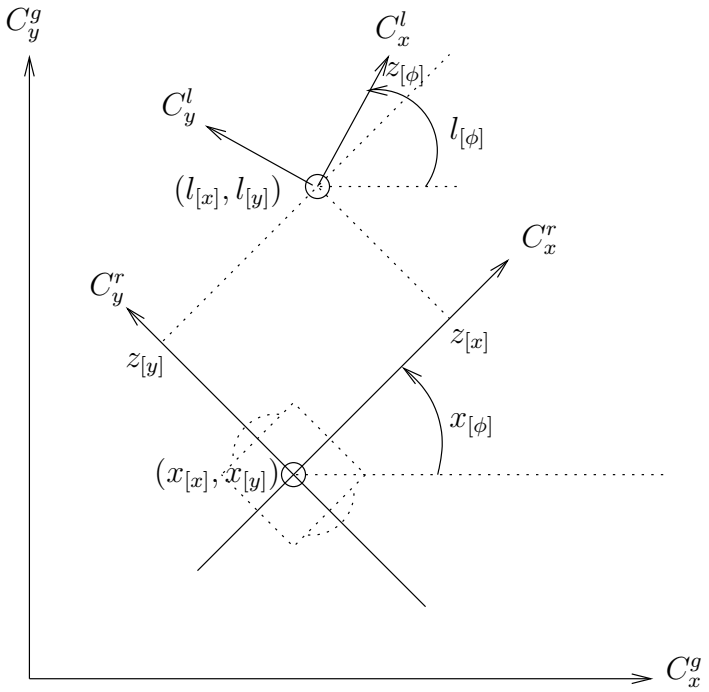


Figure 6.4: Schema of sensor and robot.

Coordinate Transformation. We have to transform the global landmark location to the coordinate system of the robot. Transformation between coordinate systems is done with the geometrical transformations *translation*, *rotation* and possibly *scaling* [18]. Assuming that the unit length of the robot's coordinate system is the same as that of the global coordinate system, by doing a translation over $-x_k$, followed by a rotation over $-x_{[\phi]}$ we go from the global coordinate system to the robot's coordinate system.

In particular, the global location of landmark l is converted to measured location z as seen from the robot's perspective by

$$z = R(-x_{[\phi]})l', \tag{6.19}$$

where l' represents the translation of the landmark,

$$l' = \begin{bmatrix} l_{[x]} \\ l_{[y]} \\ l_{[\phi]} \end{bmatrix} - \begin{bmatrix} x_{[x]} \\ x_{[y]} \\ x_{[\phi]} \end{bmatrix} = \begin{bmatrix} l_{[x]} - x_{[x]} \\ l_{[y]} - x_{[y]} \\ l_{[\phi]} - x_{[\phi]} \end{bmatrix},$$

and $R(-x_{[\phi]})$ represents the rotation matrix over $-x_{[\phi]}$ degrees,

$$R(-x_{[\phi]}) = \begin{bmatrix} \cos(-x_{[\phi]}) & -\sin(-x_{[\phi]}) & 0 \\ \sin(-x_{[\phi]}) & \cos(-x_{[\phi]}) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos(x_{[\phi]}) & \sin(x_{[\phi]}) & 0 \\ -\sin(x_{[\phi]}) & \cos(x_{[\phi]}) & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

With this, we derive the measurement function $h(\cdot,\cdot)$ by rewriting the matrix form of (6.19) into

$$\begin{aligned} z_k = h(x_k, l_k) &= \begin{bmatrix} h_x(x_k, l_k) \\ h_y(x_k, l_k) \\ h_\phi(x_k, l_k) \end{bmatrix} \\ &= \begin{bmatrix} (l_{[x],k} - x_{[x],k})\cos(x_{[\phi],k}) + (l_{[y],k} - x_{[y],k})\sin(x_{[\phi],k}) \\ (x_{[x],k} - l_{[x],k})\sin(x_{[\phi],k}) + (l_{[y],k} - x_{[y],k})\cos(x_{[\phi],k}) \\ l_{[\phi],k} - x_{[\phi],k} \end{bmatrix}. \end{aligned} \tag{6.20}$$

6.3.2 The Noisy Measurement Model

In the ideal case the derived model gives the exact location and orientation of a landmark. However, in practice, there may be noise in landmark detection systems. Besides specific sensor device dependent noise sources, there can be uncertainty in the location of the robot and in the location of the landmarks on the map. This uncertainty can be due to inaccurate modeling of the map, or slight variations of the true positions of the landmarks.

To deal with the uncertainty in the mapped locations of the landmarks, we assume that the location of a landmark l_k is Gaussian distributed according to

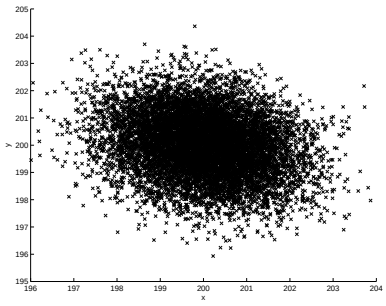
$$l_k \sim N(\hat{l}_k, L_k), \tag{6.21}$$

where $\hat{l}_k$ is the best estimate of the location of the landmark with covariance L_k,

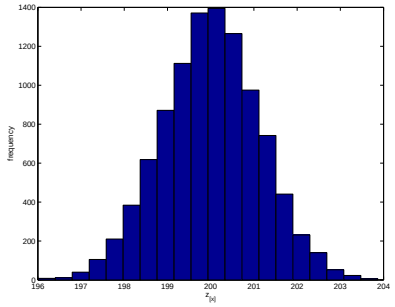
$$L_k = \begin{bmatrix} \sigma^2_{l_{[x],k}} & \sigma_{l_{[x],k}}\sigma_{l_{[y],k}} & \sigma_{l_{[x],k}}\sigma_{l_{[\phi],k}} \\ \sigma_{l_{[y],k}}\sigma_{l_{[x],k}} & \sigma^2_{l_{[y],k}} & \sigma_{l_{[y],k}}\sigma_{l_{[\phi],k}} \\ \sigma_{l_{[\phi],k}}\sigma_{l_{[x],k}} & \sigma_{l_{[\phi],k}}\sigma_{l_{[y],k}} & \sigma^2_{l_{[\phi],k}} \end{bmatrix}.$$

Besides the uncertainty in the knowledge about the landmark locations, the sensors used to detect the landmark can also have noise. We assume that the noise caused by these sources is also Gaussian distributed and we model this noise with the measurement noise term w_k,

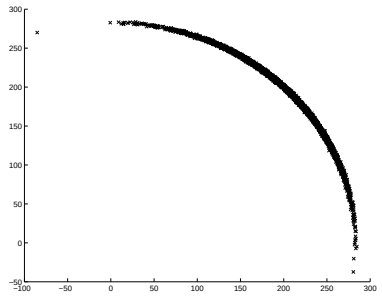
$$w_k \sim N(\hat{w}_k, R_k), \tag{6.22}$$



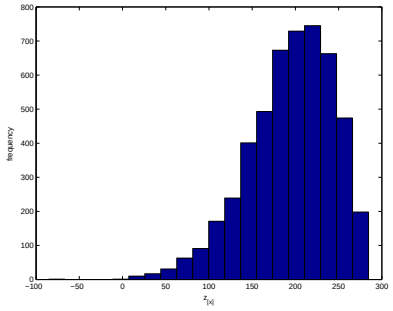
(a) $z_{[x]}$ and $z_{[y]}$, $\sigma^2_{x_{[\phi]}} = 0.05^2$.



(b) $z_{[x]}$ only.



(c) $z_{[x]}$ and $z_{[y]}$, $\sigma^2_{x_{[\phi]}} = 0.50^2$.



(d) $z_{[x]}$ only.

Figure 6.5: Influence of orientation uncertainty on coordinates.

where

$$\hat{w}_k = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$R_k = \begin{bmatrix} \sigma^2_{z_{[x],k}} & \sigma_{z_{[x],k}}\sigma_{z_{[y],k}} & \sigma_{z_{[x],k}}\sigma_{z_{[\phi],k}} \\ \sigma_{z_{[y],k}}\sigma_{z_{[x],k}} & \sigma^2_{z_{[y],k}} & \sigma_{z_{[y],k}}\sigma_{z_{[\phi],k}} \\ \sigma_{z_{[\phi],k}}\sigma_{z_{[x],k}} & \sigma_{z_{[\phi],k}}\sigma_{z_{[y],k}} & \sigma^2_{z_{[\phi],k}} \end{bmatrix}.$$

Adding this noise to the measurement model as derived in equation (6.20), we obtain

$$z_k = h(x_k, l_k) + w_k. \tag{6.23}$$

6.3.2.1 Remarks

Since the measured x and y coordinates have nonlinear relationships with the orientation of the location, the distribution of the measurements need

not be Gaussian. Depending on the amount of noise in the orientation of the location of the robot, the distribution of the x and y coordinate of the measurements is less Gaussian. We see this in Figure 6.5 in which the distribution of the x coordinate of the measurement to a landmark is plotted, given different amounts of uncertainty in the orientation of the robot's location.

6.4 Modeling Remarks

As we have seen in this chapter, modeling takes insight into the system. The basis for models often comes from physical, dynamical, and geometrical insight into the systems. Often alternative model designs for a certain system exist. In the end, models are always approximations of the true systems and some models may be more accurate than others. Which approximation works best differs per application. It is often not necessary to make models that perfectly model the behaviour of a system. An almost perfect model may result in slow evaluation, or may be too complex to analyze effectively. In particular determining noise values for different components may be a difficult task if the systems are complex.

The models that we derived in this chapter assume simple noise models. We assumed that we can model the noise with noise models that do not change over time and we assumed that moreover all the noise sources are independent of each other. In practice, it may be more accurate to make more complex noise models that give the noise distribution at different times. For example, in vision applications, objects further away may be more difficult to detect than objects nearby. The measurement noise assigned with measurements to objects far away could be adjusted to keep this in mind. Also, noises may change over time. Due to ageing, sensors may give different measurements, possibly resulting in a larger variance. For the same reason, driving systems may drive less accurate if tires get old.

Besides the models that we derived in this chapter and giving information about every state variable directly or indirectly, we could also have modeled sensors that gave only partial information. For example, a sensor that detected only the orientation or only the distance to a landmark [13].

6.5 Summary

We can model the driving and sensing of a navigating robot into system and measurement models. Modeling takes insight into the system that we want to model. Starting with the assumption that there is no noise influencing the actual system, we use dynamical and geometrical insights to derive the models. Having created noise-free models, we refine the models by including terms that model the noise aspects that can have their influence in practice.

In robot localization context, we can model the driving system of a robot into a system model, that models how actions, monitored by relative measurements from wheel encoders, result in new locations. Due to nonlinear relationships between the orientation of the robot and the x and y coordinate of its location, the distributions of these need not be Gaussian.

We can model sensors that sense the environment into measurement models. We can combine measurements from different physical sensors, like GPS and compasses, as a virtual full state sensor that supplies information about the full location of the robot. We model sensors that measure the location of landmarks from the robot's point of view with the landmark sensor model. Given the location of the robot and of a landmark in global coordinates, this model transforms the global location of the landmark into the location of the landmark from the robot's point of view. Again, due to nonlinear relationships with the orientation, the distribution of the measurements need not be Gaussian.

Chapter 7

Kalman Localization

In this chapter we combine the theory from Chapters 2, 3, 4, 5, and 6 to show how we can apply Kalman Filters (KF) to the Robot Localization problem. We implemented a simulator for doing experiments, that allows us to step-by-step combine a part of the localization problem with the technique of KFs. Along the way we discuss several practical and theoretical topics giving a better understanding of how KFs work, what their possibilities are, and how we can analyze their performance. We look at how the KF behaves under different circumstances, considering practical situations.

In Section 7.1 we start with some general remarks about the use of simulators and about applying KFs to the localization problem. In Sections 7.2 and 7.3 we look at the basic workings of KFs while we discuss how to use the driving system and full state sensor from Chapter 6 to perform position tracking. In Section 7.4 we look at validation gates and discuss how we can use these in the kidnapped robot problem. In Section 7.5 we discuss how to deal with uniform beliefs in KFs, when we look at how to apply KFs in the global positioning problem. We provide some references to related work in Section 7.6.

7.1 Analyzing Kalman Filters

This chapter has a more practical character than the previous chapters. We show the behavior of Kalman Filters, pointing out different aspects of the theory that we have discussed in the previous chapters. We start with some remarks about the way we perform our Kalman Filter analyses.

Simulator. In order to illustrate the workings of KFs, we have implemented a simulator in which we can model different systems and sensors. The simulator consists of a *truth model* and a *KF model*. The truth model models the true system and sensors. The KF model describes the KF part of the simulations. See Appendix A for more details on the simulator.

A simulator makes it is easy to experiment with different parameter settings to explore the behavior of the KF. It is for example easy to adjust the amount of noise in the simulated environment to see how the KF responds to this.

However, the truth model of the simulator should generate data representing true data as accurately as possible. Therefore, the models describing the true system and sensors have to be as accurate as possible. Making these models is very time consuming and takes deep insight into the systems to be modeled. For that reason, in our simulator, the models used to represent the true systems are the same as the models used in the KF model[1]. This has as consequence that the KF models perfectly describe the simulated truth, including the noise modeling. Although this provides a good setting for testing and exploring KF behavior, results achieved with this setting do not guarantee the same results in the true world.

Noise Instantiations. In order to perform the simulations, we have to instantiate the truth and KF model with quantities of the noise levels caused by the different noise sources. In the true world, these noise levels will differ from environment to environment, from system to system, and from sensor to sensor. Therefore, in our simulations, we are not in particular interested in the exact amounts of noise in the different components. We are interested in the general behavior and will therefore only mention exact amounts of noise when necessary. In our discussions we will mainly speak about the amounts of noise in relative context, and in terms of high and low. Again, what these terms exactly mean depends on a specific true world application.

Uncertain Analyzing. Since we are dealing with uncertainties we have to be careful not to jump to conclusions too quick; one successful simulation does not guarantee that others will be successful as well, due to the uncertainties involved. When drawing conclusions, we have to keep this in mind. We can deal with the uncertainty by performing multiple runs of an experiment and averaging the results. This gives an indication of the average performance. However, in practice, we are not interested if a robot can use the information from a KF on average, but if the information at a certain time is useful. Instead of looking at the average of the runs, we will perform multiple runs and pick out those runs that show interesting aspects of KFs.

[1] The truth model can easily be extended with alternative models though. See Appendix A for more details on the simulator.

7.2 Predictive Position Tracking

In this first of two sections we will investigate how we can apply the KF techniques in the localization problem of position tracking. Given the initial location of the robot, we want the KF to keep track of the position. In this first section we assume that we have no measurements that correct the predictions of the KF. Thus, we try to do predictive position tracking.

7.2.1 KF Modeling

In the following we assume that we have a robot equipped with some sort of driving system that can be modeled using the driving system from Section 6.1. In order to use this model in a KF, we have to instantiate the KF prediction equations.

Prediction Equations. Since the system model that we obtained in Section 6.1 is nonlinear in the orientation of the robot, we use the Extended KF from Section 5.3 to perform the state estimation. Recall that the prediction equations of the EKF are

$$\begin{aligned}\hat{x}_k^- &= f(\hat{x}_{k-1}^+, \hat{u}_{k-1}) + \hat{w}_{k-1}\\ P_k^- &= A_{x,k} P_{k-1} A_{x,k}^T + A_{u,k} U_{k-1} A_{u,k}^T + Q_{k-1},\end{aligned}$$

where $\hat{x}_{k-1}^+$ is the posterior state estimate of the previous time step with covariance matrix P_{k-1}; $\hat{u}_{k-1}^+$ is the control input that comes from a relative measurement with covariance U_{k-1}; $f(\cdot,\cdot)$ is the nonlinear system function of the guidance system; and $\hat{w}_{k-1}$ is the best estimate of the system noise with covariance Q_{k-1}. The matrices $A_{x,k}$ and $A_{u,k}$ are the Jacobian matrices with the partial derivatives of the system function $f(\cdot,\cdot)$ with respect to the state x and the control input u respectively, evaluated at the last state estimate $\hat{x}_{k-1}^+$ and control input $\hat{u}_{k-1}$. We instantiate the EKF prediction equations by calculating these Jacobians as

$$\begin{aligned}A_{x,k} &= \left.\frac{\partial f(x)}{\partial x}\right|_{x=\hat{x}_{k-1}^+, u=\hat{u}_{k-1}}\\ &= \left[\begin{array}{ccc} \frac{\partial f_x}{\partial x_{[x]}} & \frac{\partial f_x}{\partial x_{[y]}} & \frac{\partial f_x}{\partial x_{[\phi]}} \\ \frac{\partial f_y}{\partial x_{[x]}} & \frac{\partial f_y}{\partial x_{[y]}} & \frac{\partial f_y}{\partial x_{[\phi]}} \\ \frac{\partial f_\phi}{\partial x_{[x]}} & \frac{\partial f_\phi}{\partial x_{[y]}} & \frac{\partial f_\phi}{\partial x_{[\phi]}} \end{array}\right]_{x=\hat{x}_{k-1}^+, u=\hat{u}_{k-1}}\\ &= \left[\begin{array}{ccc} 1 & 0 & -u_{[\Delta D]} \cdot \sin(x_{[\phi]} + \frac{u_{[\Delta\phi]}}{2}) \\ 0 & 1 & u_{[\Delta D]} \cdot \cos(x_{[\phi]} + \frac{u_{[\Delta\phi]}}{2}) \\ 0 & 0 & 1 \end{array}\right]_{x=\hat{x}_{k-1}^+, u=\hat{u}_{k-1}}\end{aligned} \tag{7.1}$$

and

$$
\begin{aligned}
A_{u,k} &= \left.\frac{\partial f(x)}{\partial u}\right|_{x=\hat{x}^+_{k-1},u=\hat{u}_{k-1}} \\
&= \left[\begin{array}{cc} \frac{\partial f_x}{\partial u_{[\Delta D]}} & \frac{\partial f_x}{\partial u_{[\Delta \phi]}} \\ \frac{\partial f_y}{\partial u_{[\Delta D]}} & \frac{\partial f_y}{\partial u_{[\Delta \phi]}} \\ \frac{\partial f_\phi}{\partial u_{[\Delta D]}} & \frac{\partial f_\phi}{\partial u_{[\Delta \phi]}} \end{array}\right]_{x=\hat{x}^+_{k-1},u=\hat{u}_{k-1}} \\
&= \left[\begin{array}{cc} \cos(x_{[\phi]} + \frac{u_{[\Delta\phi]}}{2}) & u_{[\Delta D]} \cdot \sin(x_{[\phi]} + \frac{u_{[\Delta\phi]}}{2}) \\ \sin(x_{[\phi]} + \frac{u_{[\Delta\phi]}}{2}) & u_{[\Delta D]} \cdot -\cos(x_{[\phi]} + \frac{u_{[\Delta\phi]}}{2}) \\ 0 & 1 \end{array}\right]_{x=\hat{x}^+_{k-1},u=\hat{u}_{k-1}} .
\end{aligned}
\tag{7.2}
$$

These two matrices compute the relative change in the variables of the state, when respectively the last state estimate and control input change. The columns of $A_{x,k}$ and $A_{u,k}$ determine per element of x_{k-1} and u_{k-1} respectively what their contribution to the elements of x_k is. Notice that the x and y variables of the state are the nonlinear components in the system function due to nonlinear relationships with the orientation control input variable $u_{[\Delta\phi]}$ and the orientation state variable $x_{[\phi]}$. If these variables are constant, then the system function is linear in the state variables. Moreover, if the distance control input variable $u_{[\Delta D]}$ also does not change, then the Jacobians do not have to be recomputed.

Since in this section we are only interested in the predicting behavior of the KF, we do not use the correction step of the EKF. Therefore, after every prediction step, we simply copy the prior state and uncertainty estimates to the posterior state and uncertainty estimates for use in the following prediction step.

7.2.2 Prediction

With the instantiated EKF prediction equations we can generate predictions of the location of the robot. In order to illustrate the behavior of the prediction step, assume that we let the robot drive a square shaped path.

Initialization. Assuming that we can measure the true initial location x_0 of the robot with some precision, we initialize the EKF with the true position of the robot together with the uncertainty. We furthermore assume that we have access to relative measurements $\hat{u}_k$ monitoring the control commands executed by the driving system at every time step. Although there is uncertainty in this, we do not model this explicitly. Thus, we let the uncertainty U_k in the relative measurements be zero. Instead, we incorporate

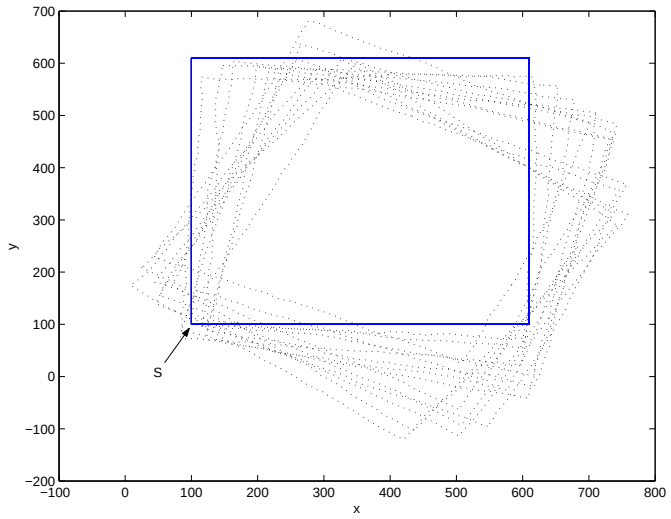


Figure 7.1: True (dotted) versus estimated (solid) x and y state element values of robot driving in squares without sensing. Robot starts at S and drives counter-clockwise.

the uncertainty in the relative measurements in the system noise. We assume that the system noise mean $\hat{w}_k$ is zero at all times, but with some covariance Q_k.

A Run. We perform a number of runs in which we let the robot drive squares and let the EKF estimate the location of the robot. Figure 7.1 shows a resulting trajectory of the true and estimated states of a representative run. As we saw in the Chapter 6, the robot does not drive in a perfect square due to the system noise. Instead, the difference between the true and noise-free trajectory, the trajectory of a perfect square path, increases over time.

The EKF has no way of knowing the exact difference between the noise-free and true trajectory, since it does not receive any measurements of the true state. It only knows that on average the system noise is zero, which makes the predictions of the EKF follow the noise-free path, as we see in Figure 7.1. Thus, in the case of no measurements, the estimated state trajectory is not adjusted towards the true trajectory; it follows the trajectory as modeled by the system model.

To make the estimation error, that is, the difference between the true and estimated states, more clear, we have plotted the estimation error per state element in Figure 7.2. Notice that the variation in the estimation error seems to increase over time. The EKF knows that the uncertainty in the system will decrease the chance that the predicted state estimates equal the true states more and more. The EKF estimates the variance in the state estimates. These variances indicate the error in the state estimate elements.

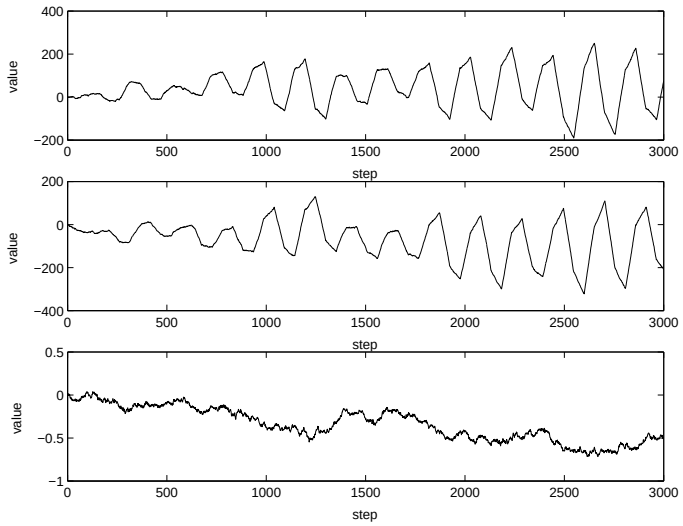


Figure 7.2: Estimation error for x, y and orientation estimate respectively.

Ideally, the variance in the state estimates that the KF computes should at all times include the true state as possible state. We can visually see if this is the case by plotting the deviations of the state estimates, that is, the square roots of the variances, into the estimation error plots. The deviation that the EKF calculates indicates how certain it is that the true state lies within a certain distance from the estimated state. The EKF is about 66% certain that the true state element lies within one deviation from the estimated element; this is called the 1σ confidence interval. The EKF is about 95% sure that the true state element lies within two deviations from the estimated element; this is called the 2σ confidence interval. If we plot the deviations into the error plot, ideally the error stays under the deviation lines.

In Figure 7.3 we have plotted the estimation error together with the 1σ and 2σ confidence intervals. We see that in this particular run the error that the EKF makes falls largely within the 1σ uncertainty region and completely within the 2σ region. This implies that at all times the true state lies within 2 deviations from the estimated states. We say that the state estimates are *consistent*.

We also see that there is an increasing trend in the deviation in the different state estimate variables. If we would let the robot drive around without taking pre-cautions, the uncertainty would grow without bounds, decreasing the information content of the state estimates. They become less *informative*.

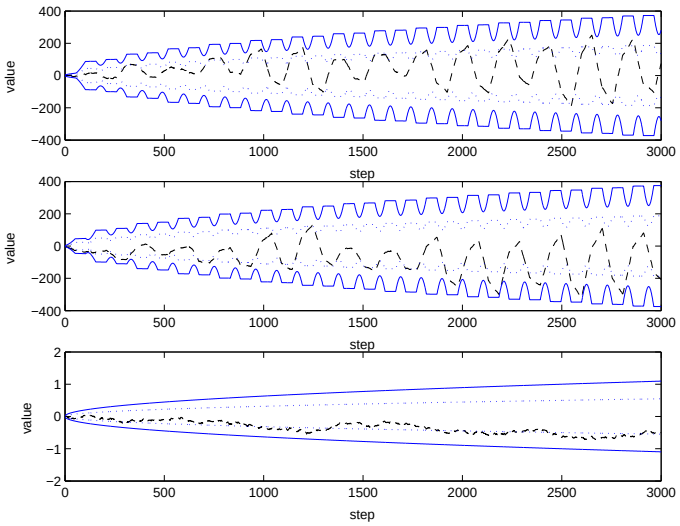


Figure 7.3: Estimation error (dashed) with 1σ (dotted) and 2σ (solid) confidence intervals for x, y and orientation estimates respectively.

7.2.3 Initial State Estimate Issues

In the given example, the EKF was initialized with the true location of the robot and some uncertainty. In practical situations this location may be available only within some certainty, incorrect, or not available at all.

Uncertain Initial State Estimate. If we are only certain about the initial state estimate to some degree, we initialize the uncertainty in the initial state estimate with the uncertainty we have in it. In this way, we tell the filter that we are not too confident that the initial state given is the true state.

However, if the initial uncertainty is high and we do not have correcting measurements, then the state estimates will not be very informative, since as we saw, the uncertainty only increases. If the initial uncertainty is low, then it will take a longer time before the state estimates become useless, depending on the level of system noise. At what point state estimates become useless differs per application. As we shall see in the next section, the uncertainty in the state estimate has consequences for the way measurements are incorporated.

Incorrect Initial State Estimate. We saw that the EKF follows the noise-free trajectory if it does not get any measurements. This has the consequence that if the initial state does not equal the true state, then the noise-free trajectory will have an incorrect starting state. All following state estimates will be based on this incorrect starting state and thus the whole trajectory will be different from the true trajectory. This in itself is

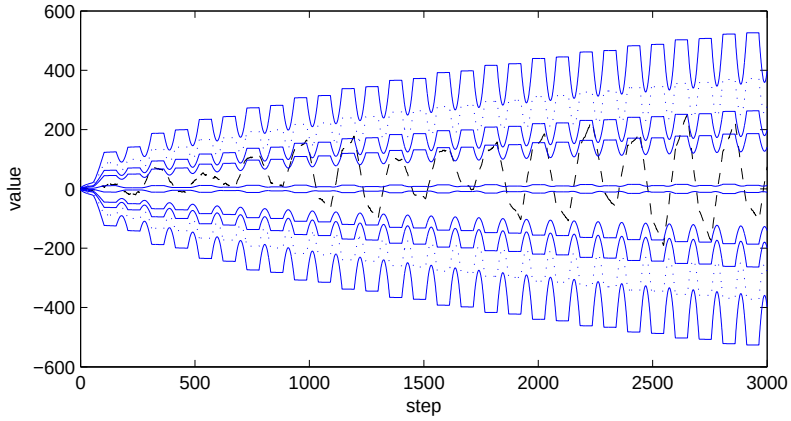


Figure 7.4: Estimation error (dashed) and 95% confidence intervals for x estimate, estimated at KF system noise set as c times the true system noise, where $c \in \{2, 1, 0.5, 0.25, 0.001\}$, from outside to inside. True system noise ($c = 1$) is dotted.

not a problem, since we already know that due to system noise the true trajectory will be different from the estimated. However, if the true state is not captured by the uncertainty in the initial state estimate, then the state estimates will be inconsistent. Moreover, they might stay inconsistent, since there are no measurements to correct the estimates.

Unavailable Initial State Estimate. When we do not have access to the initial state, then we cannot track it. In that case we will have to do global localization. We return to this in Section 7.5, where we discuss different options for determining an initial state.

7.2.4 KF System Noise Issues

In practical situations the modeled system noise may not always reflect the true system noise. It can be difficult to obtain accurate values for the variance in the state variables, and besides this, amounts of noise may change over time. Therefore it is interesting to look at what the influence of higher or lower system noise is on the performance of the KF. Ideally the system noise as modeled in the system model reflects the amount of system noise in the true system. The KF uses the modeled system noise to express the growth in uncertainty in the state estimates due to unmodeled influences.

Figure 7.4 illustrates the influence of higher and lower system noise settings in the KF. We initialized the KF with system noise values varying between 0.001 and 2 times the true system noise. Note that letting the KF estimate the states of the system with different noise initialization has no consequences for the values of the state estimates. Although the uncertainty in the state estimates uses the state estimates to propagate prior

uncertainty, the state estimates themselves are not influenced by the amount of state uncertainty, as long as there are no correcting measurement. Thus, the estimation error that the KF makes is the same as earlier.

Lower KF System Noise. If the modeled system noise is lower than the actual system noise, then the KF will incorrectly assume that the uncertainty in the state of the system increases with the modeled system noise. In our localization case, modeling lower system noise than the true system noise has as consequence that the location estimates have less uncertainty than they should have. This can result in decision making problems, since these can be based on the location of the robot. Having lower system uncertainty can result in the true state not being captured by the confidence intervals. The information content of the location estimates is higher, but it may not be consistent.

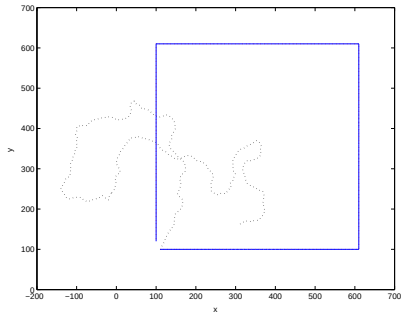
Higher KF System Noise. If the modeled system noise is higher than the actual system noise, then the KF will assume more noise in the system than that there is and increase the uncertainty in its state estimates unnecessary much. In localization context, modeling higher system noise than the true system noise makes the uncertainty in the state estimates increase quicker than that they strictly should. Thus, location estimates are more unsure and therefore decision making controllers that base their decisions on the location estimates will be more conservative. The information content of the location estimates is lower.

It is important to realize that if the noise is modeled too low, that it then will take a longer time before the KF becomes unsure about its state estimates. The height of the system noise determines the deterioration of the state estimates. This has consequences for the incorporation of measurements as we shall see in the next section.

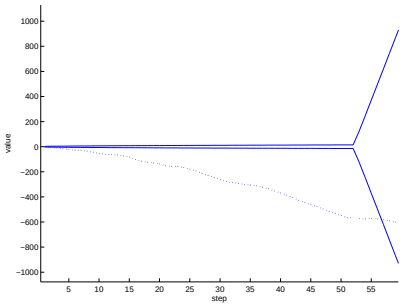
7.2.5 Linearization Errors

We used the EKF in this section since the system function that describes the state transitions of the location is nonlinear due to the orientation. The advantage of the EKF over the LKF is that the EKF can deal with nonlinear functions, since it linearizes the system functions around the latest state estimates. In Section 5.2 we discussed that we could only do this when the perturbations stay small. In the examples in this section so far, the uncertainty in the state elements was small enough to make this assumption hold. For illustrative purposes, we show what happens when this assumption does not hold.

Assume that we have a significant amount of system noise in the orientation of the robot. The orientation of the robot is the element that causes the nonlinearities in the system function, and thus the perturbations in this



(a) True (dotted) and estimated (solid) state trajectories.



(b) Estimation error in x element (dotted) and estimated 95% confidence intervals (solid).

Figure 7.5: Linearization error due to high orientation noise.

element have to stay small. If they do not stay small, the EKF can easily make inconsistent state estimates.

Figure 7.5 shows the result of a run of the 'square' driving robot in an environment with high noise in the orientation. In Figure 7.5(a) we see that due to the large amount of noise in the orientation the trajectory of the true states does not resemble that of a square at all. On the contrary, the estimated state trajectory still follows a square since the state estimates do not depend on the uncertainty when there are no measurements. Figure 7.5(b) shows the estimation error and estimated 95% confidence intervals for the x coordinate of the state estimate. Due to the high perturbations in the orientation, the EKF is not able to enclose the estimation error with the 95% confidence interval during the first 52 steps. Due to the linearization errors, the x element estimates are then inconsistent. After that, the confidence intervals all of a sudden grow significantly, with as result that the estimation error *is* encaptured by them. The confidence intervals grow, since at the 52^{nd} step the robot turns the first corner of the square its driving with as consequence that the uncertainty in x direction increases.

Figure 7.6 shows another example of the influence of nonlinearities. In this hypothetical case we modeled no system noise, and only initial uncertainty in the estimate of the orientation. The EKF is initialized with the true location of the robot, after which we performed 50 runs, each time starting from the same initial location. Due to the initial uncertainty in the orientation, the robot drives different paths. The figure shows the locations the robot was at during the different runs and the regions in which the EKF believed the true state was located as lines. These lines are in fact 95% error ellipses with very small minor axis in x direction. The figure shows that the EKF is not able to encapture the distribution of the locations of

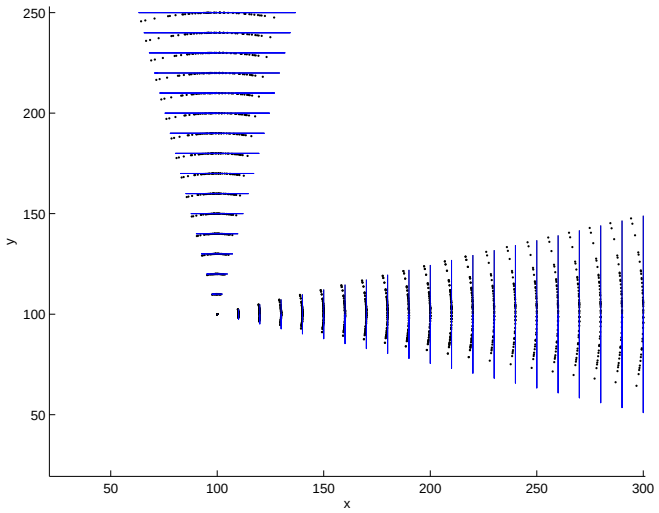


Figure 7.6: Linearization error due to initial uncertainty in orientation. 50 true state trajectories (dotted) and estimated 95% error ellipses (solid).

the robot adequately. The error ellipses do not encapture the shape of the non-Gaussian distribution of the location. Thus, the state estimates are inconsistent.

In both examples the state estimates are inconsistent. A way to deal with this is by adding extra noise terms to the system model that represent the uncertainty due to linearization errors. However, the uncertainty caused by linearization errors can be hard to determine and is system model specific. In Chapter 8 we will look closer at how linearization errors occur and look at how the Iterated EKF deals with them.

7.3 Corrective Position Tracking

In this second section on position tracking we use measurements from a full state sensor to correct the predictions made with the model from the previous section in order to reduce the unbounded uncertainty growth. Using a full state sensor allows us to illustrate the nature of KFs, since we can easily compare measurements with locations.

7.3.1 KF Modeling

In the following we assume the prediction equations from the previous section. Besides that we assume that the robot has a full state sensor that we can model as in Section 6.2. In order to use this sensor to correct the predictions, we instantiate KF correction equations.

Correction Equations. Recall from Section 6.2 that the governing measurement function of the full state sensor is linear in all state elements and that we thus can use the LKF correction equations. However, we continue using the EKF and show that the resulting EKF equations are the same as when we would use the LKF. Therefore, recall that the correction equations of the EKF are

$$\begin{aligned} \hat{x}_k^+ &= \hat{x}_k^- + K_k(z_k - h(\hat{x}_k^-)) \\ P_k^+ &= (I - K_k H_k) P_k^-, \end{aligned}$$

where

$$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + R_k)^{-1},$$

where the Jacobian matrix H_k with partial derivatives of the measurement function $h(\cdot)$ with respect to the state x is evaluated at the prior state estimate $\hat{x}_k^-$, which in this case is

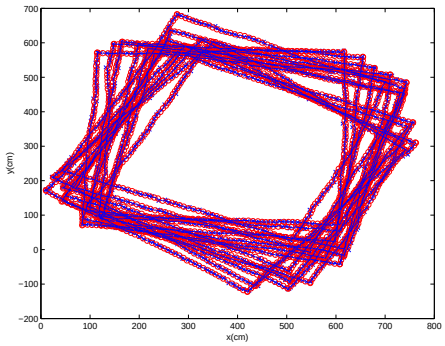
$$\begin{aligned} H_k &= \left.\frac{\partial h(x)}{\partial x}\right|_{x=\hat{x}_k^-} \\ &= \begin{bmatrix} \frac{\partial h_x}{\partial x_{[x]}} & \frac{\partial h_x}{\partial x_{[y]}} & \frac{\partial h_x}{\partial x_{[\phi]}} \\ \frac{\partial h_y}{\partial x_{[x]}} & \frac{\partial h_y}{\partial x_{[y]}} & \frac{\partial h_y}{\partial x_{[\phi]}} \\ \frac{\partial h_\phi}{\partial x_{[x]}} & \frac{\partial h_\phi}{\partial x_{[y]}} & \frac{\partial h_\phi}{\partial x_{[\phi]}} \end{bmatrix}_{x=\hat{x}_k^-} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \end{aligned} \tag{7.3}$$

Notice that since the Jacobian only contains constants, the measurement function is linear in all the state variables. We do not have to recompute the matrix at every time step, and it is easily seen that this would also be the matrix that we would use as measurement transfer matrix if we would use the LKF.

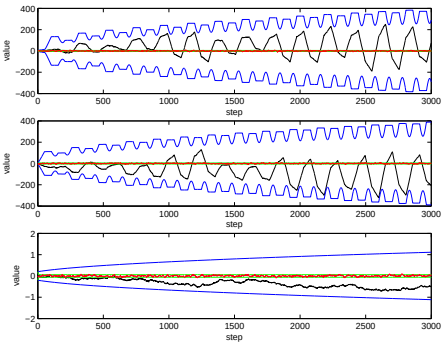
7.3.2 Correcting Locations

With the EKF correction equations we can correct the state estimates. Assume that our robot drives squares and that at every step it receives a measurement from its full state sensor. We initialize the measurement noise with the true measurement noise and determine measurements for the runs of the previous section.

Figure 7.7 shows the estimated state trajectory that the EKF generates during one of the runs. In Figure 7.7(a) we plotted the true and estimated state trajectories. In Figure 7.7(b) we plotted the estimation error and estimated 95% confidence intervals of the run from the previous section, together with the estimation error and confidence intervals of the new run. We clearly see that the EKF estimates the trajectory of the robot significantly more accurate when it has access to measurements. In fact, the



(a) True and estimated trajectory of square driving robot with measurements. Due to measurements at every step, the true and estimated trajectories overlap.



(b) Posterior estimation error (solid, black) and 95% confidence interval (solid, blue) for run without measurements and run with measurements (solid, red and solid, green respectively). Posterior estimation error and confidence interval with measurements almost reduces to line compared to without measurements.

Figure 7.7: Position tracking with corrections.

posterior uncertainty in the case with measurements reduces to a line when compared to the posterior uncertainty in the case without measurements. In the following we will look closer at this result and at how the EKF achieved it.

Step by Step. In Figure 7.8 we have plotted the first 19 steps of the true state and measurement trajectory, the prior and posterior estimated state trajectory and the estimated measurement trajectory. We first of all notice that the estimated trajectory is not the noise-free trajectory that the robot drives in the case that the EKF does not have access to any measurements, as in the previous section. Instead, we see that when there is a measurement, the EKF corrects the state estimate. Sometimes this brings the state estimate closer to the true state, for example at step 5, sometimes however it brings it further away, for example at step 3.

To see what the EKF exactly does, we look at an arbitrary step, the transition from step 4 to 5. The robot is at true location 4; the EKF estimates the location of the robot to be somewhat higher. The robot makes its step by moving forward to arrive at true location 5. The EKF uses the relative displacement information from the wheel encoders to predict where the robot will be at time 5, given that it was at estimated location 4 when it started. The prediction of the path of the robot is shown in blue in the

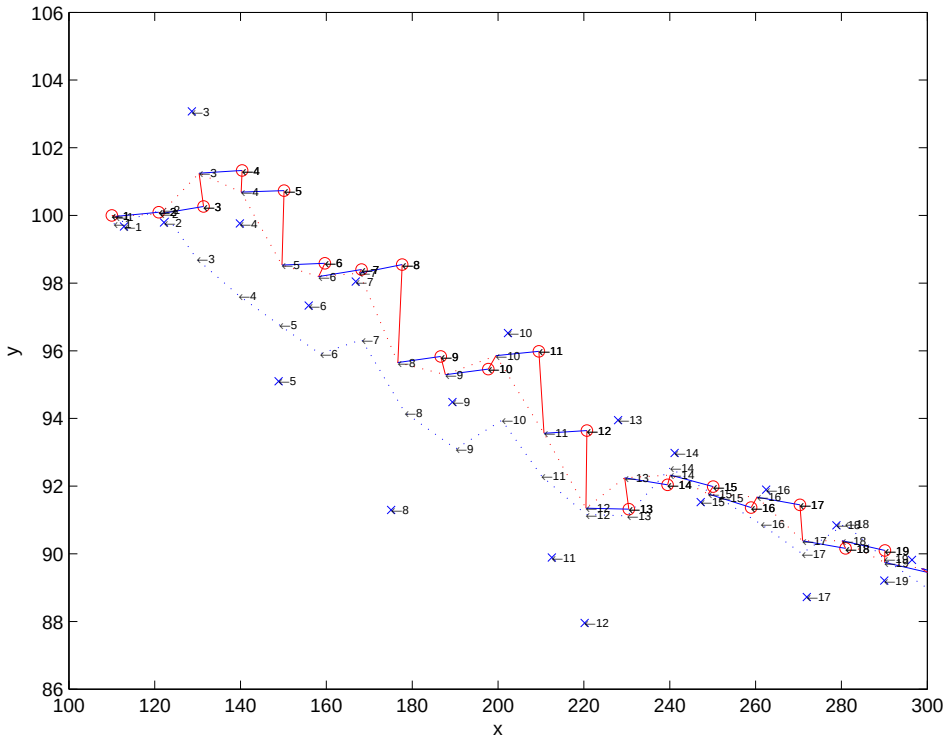


Figure 7.8: First 19 steps from square driving robot. True trajectory (dotted, blue); true measurements x (blue); predicted trajectory (solid, blue); predicted measurements o (blue); state corrections (solid, red); posterior estimate trajectory (dotted, red).

figure and results in predicted location 5.

At that time the robot receives a measurement from its location sensor, true measurement 5. Due to measurement noise, the sensor reading does not exactly point to the location of the robot at that time. In order for the EKF to incorporate the true measurement, it makes a measurement prediction. Since the EKF knows that the location sensor senses the location of the robot, the best prediction of the sensor reading is the best location estimate so far. The predicted location is denoted with the red 'o' at step 5. As we see in the figure, the distance between the predicted and true measurement is quite large in the y coordinate. The EKF adjusts its state prediction toward the true measurement by reducing the y coordinate of its state estimate with some amount. We denote this correction by the red line in the figure, resulting in corrected location estimate 5 at the end of the red line. After the correction, the robot drives to location 6, incorporates the movement in its last location estimate, receives a measurement, and incorporates the

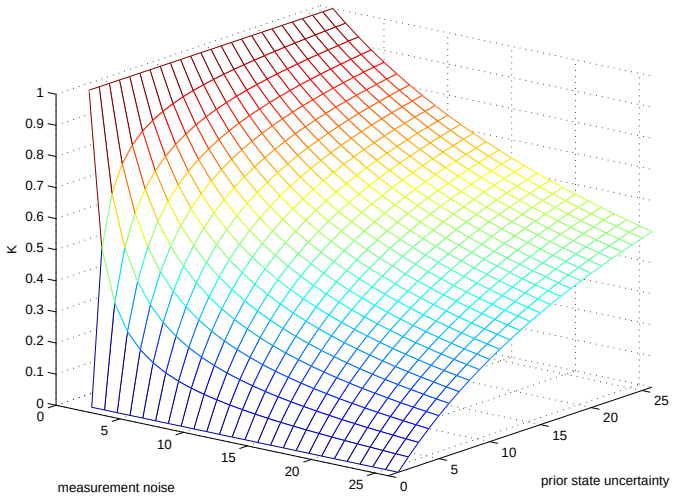


Figure 7.9: Relation between prior uncertainty, measurement noise and Kalman Gain for simple measurement model.

measurement, etc.

Notice that around step 12 the difference between the estimated trajectory and the true trajectory is significantly smaller than around step 5. One may think that this is due to the fact that the robot has seen more measurements and that thus by averaging the measurements the estimate of the location is closer to the true location. However, as will become clear later, this is not the reason. The reason is that the measurements around step 12 are closer to the true trajectory than earlier. To clarify this, we first look more precisely at the mechanism that determines how much of measurement residual is incorporated, the Kalman Gain.

7.3.3 Kalman Gain

The inclusion of measurements into state estimates is determined by the Kalman Gain, influenced by the uncertainty in the prior state estimate and the uncertainty in the measurement estimate. Intuitively, measurements with high precision should result in precise state estimates. The measurements measure the full state, and thus if they have high precision, the KF can count on them as good indicators for the true state. The Kalman Gain should reflect this.

In Figure 7.9 we have plotted the relation between the prior uncertainty, the measurement noise, and the Kalman Gain for some variable of which the measurements measure this variable directly and are corrupted by measurement noise. The plot shows that the higher the prior uncertainty the more the KG moves towards one. The higher the prior uncertainty, the less con-

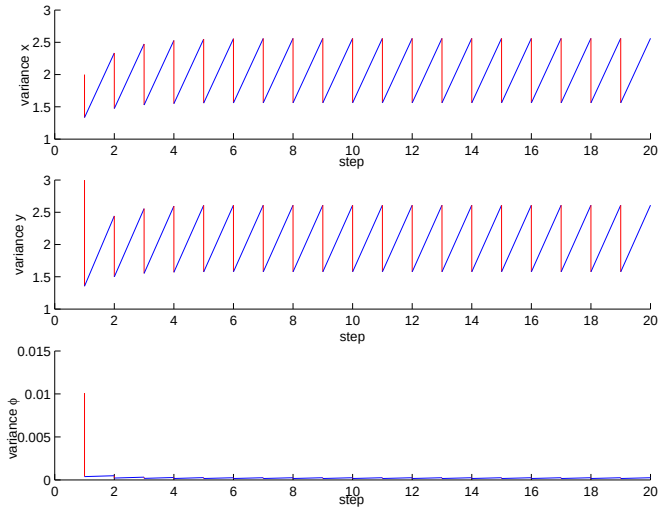


Figure 7.10: Prior and posterior variance in x, y and orientation estimates respectively. Predictions (blue); corrections(red).

fidence the EKF has in the state estimate and thus the more it wants to take information from the real world. On the other hand, the measurement noise tries to lower the Kalman Gain. Higher uncertainty in the measurements indicates that the information from the real world is not that trustworthy and that thus less of the information should be taken into account. The resulting Kalman Gain balances these two uncertainties. Notice that the Kalman Gain does not depend on the values of neither the predicted measurement, nor the true measurement.

7.3.4 Uncertainty Convergence

In Figure 7.10 we have plotted the predicted and corrected variance in the location estimates as computed by the EKF during the 19 steps of the example. We notice that the variances in all elements quickly reach a *stable* state. The prior and posterior variances *converge*. The increase in uncertainty due to prediction and the decrease in uncertainty due to correction keep each other in balance. The steps before the variances show their stable behavior are under the influence of the uncertainty in the initial location estimate. If the measurement noise is higher it will take a longer time before the variances reach a stable phase.

The convergence occurs, since the uncertainty in the system noise and the measurement noise are constant. The uncertainty in the predicted measurement consists of the uncertainty in the location estimate and the measurement noise. Based on this, the Kalman Gain determines the balance between the two uncertainty components. Since in our case the increase in noise is constant at every time, there is a moment at which the decrease

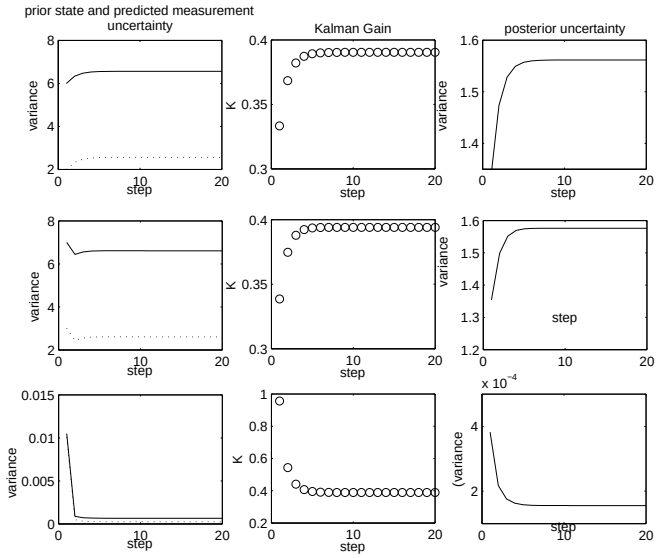


Figure 7.11: For each estimated location variable x (top), y (middle), ϕ (bottom):(left column) measurement residual (solid) and prior uncertainty (dotted); (middle column) Kalman Gain; (right column) posterior uncertainty.

in uncertainty equals the increase in uncertainty. From that time on, the Kalman Gain has equal values at every time step. We can see this in Figure 7.11.

The first column in the figure shows the uncertainty in the prior estimate and measurement residual respectively for the x, y and orientation estimates. Notice that the measurement residual is always larger than the prior uncertainty, since the uncertainty in the residual besides the location estimate also depends on the measurement noise.

The second column in the figure shows the computed Kalman Gains. Notice that the Kalman Gains computing the adjustments of the location variables with the used measurement model are always values between zero and one. In general this is not the case. In this case it is because the measurement Jacobian H is the identity matrix. Furthermore, notice that when the measurement residual and prior uncertainty converge, that then the Kalman Gain also converges.

The third column shows the resulting posterior variances. The posterior uncertainty is smaller than the prior uncertainty, unless the uncertainty in the measurement residual, due to measurement noise, is infinitely high, resulting in a Kalman Gain of zero. In that case the posterior uncertainty equals the prior uncertainty.

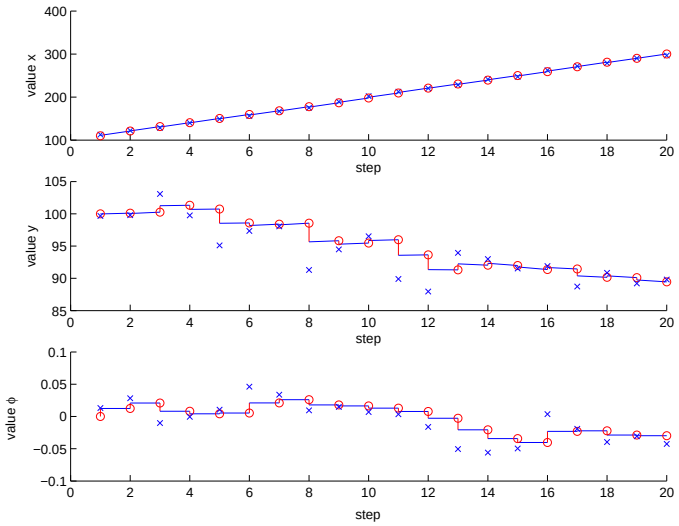


Figure 7.12: Prior and posterior values for location estimates, together with true (blue x) and predicted (red o) measurements; state predictions (blue), state corrections (red).

7.3.5 Measurement Residuals

A high Kalman Gain results in high reduction of uncertainty, but not necessarily a large absolute change in estimated values. In Figure 7.12 we have plotted the prior estimates of the state variables, together with the corrections. We can clearly see that, although the Kalman Gain has a stable value, the correction changes from time to time. Obviously, the reason for this is that the Kalman Gain only determines the proportional amount of measurement residual that is added to the location estimate to correct it. That is, a high Kalman Gain implies a high use of the measurement residual. We can see this in the figure if we look at the distances between the true and predicted measurements. From approximately step 5 the Kalman Gain stays constant, which expresses itself in a constant proportion of the measurement residuals being added to the location estimates, a little less than 0.4. Thus, although the Kalman Gain stabilizes, the state estimates do not become more accurate.

7.3.6 Consistency

A last thing to look at is whether or not the corrected uncertainty estimates capture the true states. Reducing the uncertainty in state estimates is only useful as long as the uncertainty estimates still capture the true state after correction.

We can visually see if the corrected uncertainty captures the true state by again looking at the estimation error that the EKF makes and the estimated

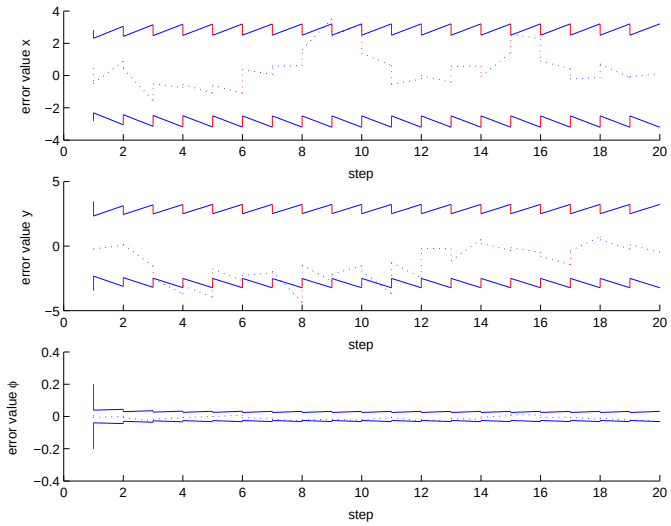


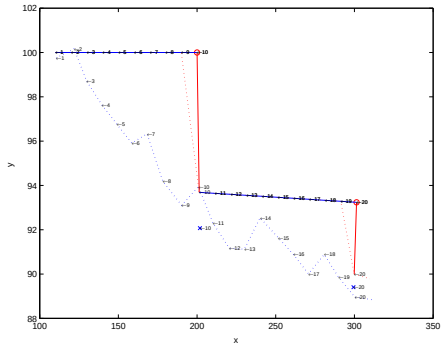
Figure 7.13: Prior (dotted, blue) and posterior (dotted, red) estimation errors, with prior (solid, blue) and posterior (solid, red) 2σ confidence intervals for x, y, and orientation respectively.

confidence intervals. Figure 7.13 shows the prior and posterior estimation errors and the prior and posterior 95% confidence intervals. We see that the estimation errors are not significantly larger than as indicated by the confidence intervals, although at some steps the error in the x and y coordinates gets close to the intervals or even passes it.

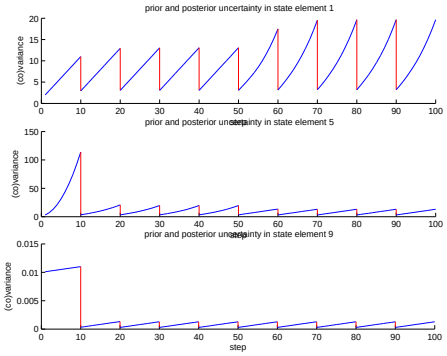
7.3.7 Infrequent Measurements

So far, the EKF had access to measurements at every time step. However, in practice, measurements may not always be available at every step of the run. As we mentioned in Chapter 2, relative position sensors may have a high frequency of measuring, whereas absolute position measurements may be less frequently available, due to for example computational time or costs.

In Figures 7.14(a) and 7.14(b) we have plotted the first steps of the square driving robot when it does not get measurements at every time step, but at every 10^{th} time step. In Figure 7.14(a) we see that as long as the robot does not receive measurements, the state estimates follow the trajectory as indicated by the system model. As discussed in the previous section, the uncertainty thereby increases, see Figure 7.14(b). As soon as a measurement arrives, the EKF corrects the state estimate and the uncertainty decreases. After 10 steps without measurements, the uncertainty in the state estimates has increased. With constant measurement noise, the Kalman Gain will then be relatively high, resulting in a relatively high decrease in uncertainty. See also Figure 7.9 with Kalman Gains. The decrease is higher than when the measurement would be incorporated one step at a time. Thus, sparsity of



(a) True (dotted,blue), prior (solid, blue), and posterior (solid, red) estimated x and y trajectory, together with true (blue x), and predicted (red o) measurements.



(b) Estimated prior (blue) and posterior (red) uncertainty for x, y and orientation estimates respectively.

Figure 7.14: Measurements at every 10^{th} step.

information leads to large use when it is available.

This 'greediness' can be a problem, if the measurement is a measurement with a relatively large error at that specific time. The state estimate will then have a large estimation error. Moreover, the uncertainty in the corrected state estimate will be low. Suppose that after a period without measurements there follows a period with a number of measurements, the Kalman Gain will then be relatively high for the first measurement, but relatively low for the following measurements. These will be taken less into account. We will see examples of this later.

7.4 Kidnapped Robot

Now that we have seen how we can use the KF in position tracking, we turn our attention to the problem of the kidnapped robot. The robot is fully aware of its location, but all off a sudden is transferred to another location. This will result in relatively high measurement residuals, since the measurements differ significantly from the measurement predictions. Assuming that we are estimating the location of the robot with a KF, we can discuss what happens in that case.

If the robot has been kidnapped, it is at a different location than where it thinks it is. Assuming that the KF had low uncertainty in its state estimate, the state estimate will be incorrect and inconsistent after the kidnap. When measurements arrive, the KF will simply incorporate the high measurement residuals into the state estimates if it does not take any action to determine

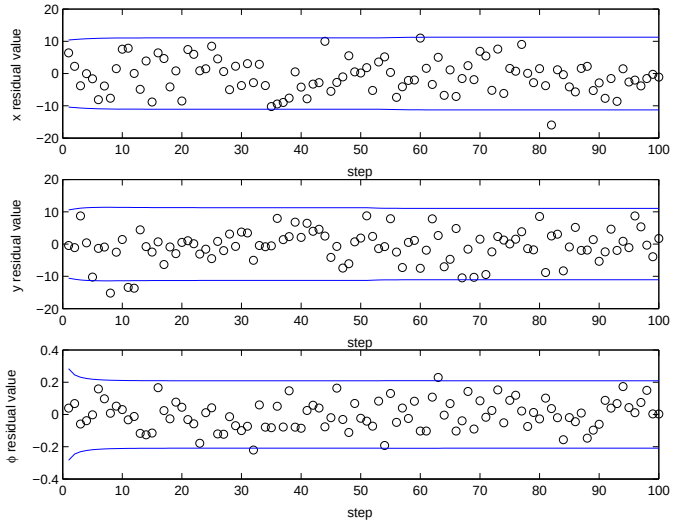


Figure 7.15: Measurement residuals (o) for x, y and orientation respectively, of run with measurements at every step, together with 95% measurement residual confidence intervals (solid).

kidnapping. This results in state estimates moving toward the new situation. However, until the state estimates have reached that situation, the state estimates stay inconsistent. It may take a while before the state estimates have adjusted to the new location, depending on the amount of system noise and prior uncertainty in the state estimates. If the prior uncertainty is relatively low, the Kalman Gain incorporates only a relatively small portion of the measurement residual. The uncertainty in the state estimate will stay small, since it does incorporate part of the measurement, and thus the Kalman Gain will keep adjusting the state with small steps.

If the robot would be able to detect a kidnap, it can take appropriate action it to re-localize itself quicker. Since due to a kidnap the location changes, measurements taken from the new location will be unexpected. This results in high measurement residuals. We can use a *validation gate* to detect these high measurements.

7.4.1 Validation Gates

Validation gates monitor the measurement residuals to detect unexpected measurements [29]. We can use the variance in the measurement residuals to determine whether or not a measurement residual is significantly unexpected. Figure 7.15 shows the measurement residuals of the square driving robot with measurements at every step and the 95% confidence intervals. In this figure the robot is not kidnapped, and all the measurement residuals are captured in the confidence interval.

When the robot has been kidnapped, the real measurement may be com-

pletely unexpected, resulting in an extreme measurement residual. Using the uncertainty in the measurement residual, the EKF computes the probability of the obtained measurement residual and determines if it is too unlikely to correspond to a valid measurement. In that case the EKF detected that something is wrong. In particular, if multiple consecutive measurement residuals are very unlikely, the likelihood that something is wrong increases.

Notice that since uncertainty in the measurement residual depends on the prior uncertainty, detecting strange measurements in this way may work well both when measurements appear frequently as when they appear infrequently, since the confidence intervals of the measurement residuals are automatically adjusted to the uncertainty in the state estimate. With frequent measurements, the error in the state estimates will stay relatively low, and the measurement residuals as well. When measurements appear infrequently, the uncertainty in the state estimates increases. As a consequence, the uncertainty in the measurement residual also increases, and the boundaries of rejection are increased automatically.

7.4.2 Remarks

When the validation gate notices that something is wrong, the robot does not always have to be kidnapped. Other sources of unexpected measurements can be caused by failing sensors and increasing measurement noise. In order to determine if the robot is truly kidnapped, it has to make sure that its sensor is working well.

Note that if due to detecting of unexpected measurements the robot decides to reject the measurement, the robot should be careful not to reject too many measurements. If measurements are rejected, state estimates are not corrected, and following measurement residuals may also be too high to be accepted, in particular in the case of a kidnapped robot. This leads to increasing uncertainty in the state estimates, resulting in that at a certain moment the measurement residual might get accepted again. This will result in a large state correction and decrease in uncertainty. However, if the robot is moved over a large distance, it may take a long time before this point is reached. The robot is then lost and can not adequately track its position.

Note that rejected measurements that are not due to failing of sensors and not due to re-localization of the robot may be due to objects corrupting the measurement. By collecting rejected measurements, the robot may obtain information about unexpected events in the environment. This may give options for using the KF in dynamic environments.

7.5 Global Localization

With the discussion of application of KFs in the position tracking and kidnapped robot problem in the previous sections, we are left with the global

localization problem. In the global localization problem, the robot does not know its initial position. Its belief in the location is *uniform*, that is, it assigns equal probability to every location in the location space. Since KFs represent the belief by Gaussians, and since Gaussians are unimodal probability distributions, we can not directly express a uniform belief. In this section we look at how to deal with this. We can approximate uniformness by injecting a very large amount of uncertainty in the initial state estimate, or we can try to obtain the result of uniformness, without explicitly representing it.

7.5.1 Approximating Uniformness

A first idea is to approximate a uniform distribution by injecting an enormous amount of initial uncertainty in the initial state estimate when we have no idea of the state. The Gaussian density over the state space will then assign probabilities close to zero to the possible states. In this way we can approximate a uniform distribution.

However, an enormous amount of initial uncertainty in the initial state estimate results in a very large decrease of uncertainty when a measurement comes available. This can lead to precision problems in the calculation of the uncertainty. Thus, the initial uncertainty should be lower than extremely large, but still large enough to adequately represent a uniform belief. Determining the amount of uncertainty needed is system specific and depends moreover on the unit size used and is therefore not practical in general.

7.5.2 Re-initialization

Instead of trying to approximate a uniform distribution, we can look at what the consequences of a uniform distribution are and then try to achieve those results in a way without using large amounts of uncertainty explicitly. A uniform belief in the state estimate means high uncertainty in that state estimate. Measurements will compared to this be very certain, and thus the Kalman Gain will be such that practically all of the measurement residual is taken into account.

Thus, to obtain the same result as initialization of a large amount of uncertainty in the initial state estimate, we can try to force the Kalman Gain to take all of the measurement residual into account without looking at the uncertainty in the state estimate. The idea is to initialize the KF once the first measurement comes available, using all the information from the measurement, including the uncertainty in it.

Maximal Measurement Information. To obtain a Kalman Gain that takes into account all of the measurement residual, consider the expression

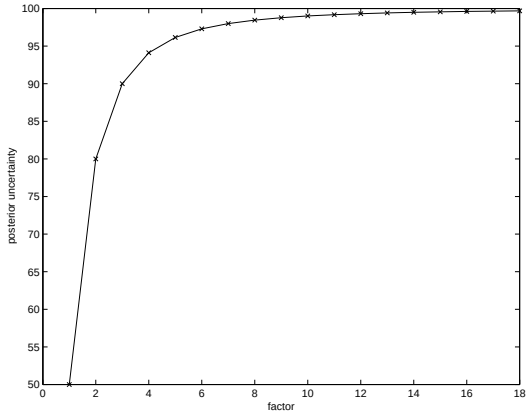


Figure 7.16: Factor that deviation of prior state estimate is the deviation of measurement noise versus resulting posterior uncertainty, that is, $\frac{\sigma_x}{\sigma_v}$, where measurement variance σ_v^2 is 10^2, for measurement model that measures x directly.

that computes the Kalman Gain,

$$K_k = P_k^- H_{x,k}^T (H_{x,k} P_k^- H_{x,k}^T + C_{rest,k})^{-1},$$

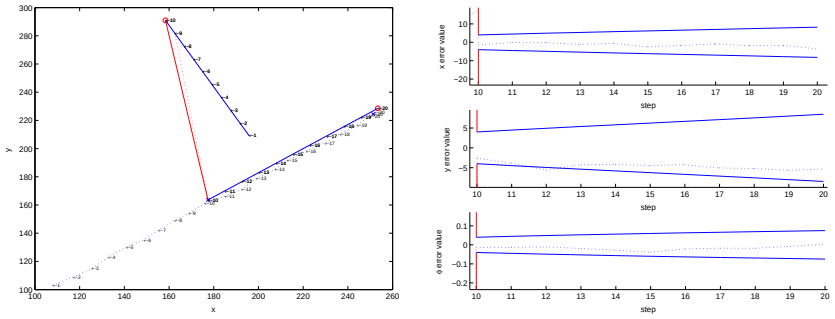
where $C_{rest,k}$ comes from uncertainty in the measurement prediction, caused by other sources than the uncertainty in the prior state estimate. If the uncertainty P_k^- in the state estimate becomes very large, the term $H_{x,k} P_k^- H_{x,k}^T$ expressing the uncertainty in the measurement prediction caused by the uncertainty in the state estimate will become the dominant noise source. In the limit, the Kalman Gain then goes to the product of the term $P_k^- H_{x,k}^T$ and $H_{x,k}$ times that term. We cancel these terms out, and obtain

$$K_k = H_{x,k}^{-1}. \tag{7.4}$$

Thus, if the uncertainty in the prior state estimate is extremely large, the Kalman Gain reduces to the inverse of the measurement transfer matrix.

Global Initialization. A robot that does not know its location, simply drives around until it receives its first measurement. Once this measurement arrives it initializes its state estimate using the KF correction equation. However, instead of using the regular Kalman Gain equation to compute the Kalman Gain, it sets the Kalman Gain to $H_{x,k}^{-1}$ to incorporate all of the measurement residual. This gives us for the state estimate

$$\begin{aligned}
\hat{x}_k^+ &= \hat{x}_k^- + K_k(z_k - H_{x,k}\hat{x}_k^-) \\
&= \hat{x}_k^- + H_{x,k}^{-1}(z_k - H_{x,k}\hat{x}_k^-) \\
&= \hat{x}_k^- + H_{x,k}^{-1} z_k - \hat{x}_k^- \\
&= H_{x,k}^{-1} z_k.
\end{aligned}$$



(a) True and estimated x, y, and measurement trajectories.

(b) Estimation error and 95% confidence intervals for x, y, and orientation estimates respectively.

Figure 7.17: Global localization when when measurement comes available.

Note that this is not necessarily the true state x_k, since measurement noise may have corrupted the measurement. For the uncertainty in the state estimate we obtain

$$\begin{aligned} P_k^+ &= (I - K_k H_{x,k}) P_k^- \\ &= (I - H_{x,k}^{-1} H_{x,k}) P_k^- \\ &= 0. \end{aligned}$$

The posterior uncertainty becomes zero, due to the fact that we ignored the measurement noise. We know that there is uncertainty in the measurements, and should thus initialize the new initial state estimate with this uncertainty. Figure 7.16 confirms that as the prior uncertainty becomes many times larger than the measurement noise, that then the posterior uncertainty approaches the uncertainty in the measurement noise. Therefore, we convert the uncertainty in the measurement noise to state space,

$$P_k^+ = H_{x,k}^{-1} C_{rest,k} (H_{x,k}^{-1})^T, \tag{7.5}$$

This expression assigns the uncertainty in the measurement prediction due to measurement noise, and possibly other sources except the prior state uncertainty, to posterior uncertainty by converting it from measurement space to state space with the inverse of the measurement transfer matrix.

It is important to note that in order to used the derived equations, we need to be able to calculate the left inverse of the measurement matrix. If the dimension of the state is larger than the dimension of the measurement, then this inverse may not exist.

Illustration. As an illustration, Figure 7.17(a) shows the true and estimated trajectory of a robot that has no initial idea about its position. The KF is initialized at an arbitrary position and orientation. In the figure we see that the robot is driving in a completely different direction during the first 10 steps. At the 10^{th} step, the robot makes a measurement. It incorporates it using the described technique. As we can see, the state estimate completely moves to the location indicated by the measurement.

Figure 7.17(b) shows the estimation errors and the 95% confidence intervals for the three state variables beginning at step 10, when the measurement arrived. Notice that after incorporation of the measurement, the estimated uncertainty does not reduce to zero.

7.6 Related Work

In this chapter we have looked at how we can use the Kalman Filter theory in the localization problem. We are not the first ones to do this. Over the years, many researchers have successfully applied the Kalman Filter to the localization problem resulting in a significant amount of scientific articles. For examples of these we refer to [2, 24, 38, 48].

7.7 Summary

With the theory of the previous chapters we can approach the localization problem using KFs. We can use KFs to do position tracking, global localization, and solve the kidnapped robot problem.

Using a driving system alone, the KF can perform predictive position tracking. The estimated trajectory is the noise-free trajectory given by the system function, and the uncertainty in the state estimates grows without bounds. Choosing the levels of system noise should be done with care, in order to obtain useful state estimates. If the system model is significantly nonlinear, the state estimates can have estimation errors outside the confidence intervals. This can easily happen with large noise in the orientation of the location.

Using a full state sensor we can correct the predictions that the predictive position tracking makes to obtain state estimates that are significantly more accurate, together with bounded uncertainty in the state estimates. Depending on the level of prior and measurement noise uncertainty, the Kalman Gain determines what amount of measurement residual to include in the state estimate. It does this independent of the actual measurement and measurement prediction. With fixed system and measurement noise, and frequent measurements, the uncertainty in the state estimates converges. With infrequent measurements, the uncertainty increases until a measurement arrives, resulting in a large uncertainty decrease.

If the robot has low uncertainty about its location and all of a sudden is kidnapped to a new location, the measurement residuals will be high. We can use a validation gate to monitor when the measurement residuals are unexpectedly high. Once the robot has detected this, it may try to globally re-localize itself.

In order to globally localize the robot if it does not know where it is, we can initialize the KF with the information from the first measurement that arrives. We can derive equations that use the left inverse of the measurement matrix to determine the state estimate and uncertainty given the measurement.

Chapter 8

Landmark Kalman Localization

In Chapter 7 we discussed how to apply KF techniques to the localization problem. In that chapter we assumed a full state sensor that directly measures the location of the robot in global coordinates. In this chapter we continue exploring the application of KFs to the localization problem when we discuss how to use KFs when we have sensors that measure the location of landmarks in the environment of the robot.

In Section 8.1 we look at two different cases of localization using landmarks. In Section 8.2 we discuss the case when we can uniquely identify the landmarks to which a measurement is made. In Section 8.3 we discuss the case when we can only determine the type of landmark to which a measurement is made. Along the way we extend the understanding of KFs by looking more precisely at how linearization errors occur, and by extending the standard KF framework in order to deal with multiple state beliefs.

8.1 Landmarks and Localization

When we want to use landmarks for localization, we generally have a map with the location of landmarks in global coordinates, whereas the measurements are locations to landmarks in coordinates from the robot's point of view. In order to use the information on the map and the measurement, there needs to be a relation between the two. The *correspondence* between the map and the measurement has to be established.

In some cases, raw sensor data provide direct information about the landmark on the map to which the measurement is made, for example encoded in bar codes. However, in general, raw sensor data do not contain the relationship between the measured data and mapped landmark. The robot has to establish this connection itself. For this purpose, it needs to have some *correspondence model*, that given raw sensor data determines the

landmark on the map to which the measurement corresponds. This sensor data will in general contain more than just the x, y and orientation of the landmark from the robot's point of view, like for example the color.

To simplify things, we can assume that the correspondence model can uniquely determine to which landmark a measurement corresponds. There is a one-to-one mapping between measurement and landmark, and this mapping is correct. There is one reference point to which the measurement is made. In that case, we can use the measurement model from Chapter 6 in order to correct state estimates. We will do this in Section 8.2.

The assumption of unique landmark identification may in practice not always hold. In many cases establishing the correspondence is not a trivial task. A correspondence model may be able to distinct between different groups of landmarks, but not between landmarks of the same group. Also, it might sometimes be able to indicate with some probability that a landmark is part of a certain group of landmarks, and with some other probability that it is part of another group. Thus, the correspondence model will not return one landmark to which the measurement corresponds, but a series of landmarks together with correspondence probabilities. We can create a framework around the KF that manages this situation by keeping the history of encountered landmarks into account when determining the state of the robot. In Section 8.3 we discuss how to do this.

8.2 Unique Landmark Localization

Assume that we have a sensor that measures the relative location to landmarks, and that we have a map with the locations of the landmarks with some uncertainty in the global coordinate space. We also assume that the correspondence model uniquely determines the landmark to which a measurement was made.

8.2.1 KF Modeling

In Section 6.3.1 we derived the measurement model that given the robot's and landmark's location in some general coordinate system calculates the coordinates of the landmark from the robot's point of view. In order to use this model in correcting state estimates, we instantiate EKF correction equations. Recall that in this case a measurement is the location of a landmark from the robot's point of view, and that the measurement model predicts this measurement given the location of the robot and a landmark in global coordinates.

Correction Equations. Since besides the location of the robot x_k and the measurement noise v_k, the location of a landmark l_k is also input to the

measurement model, we use the EKF equations

$$\begin{aligned}\hat{x}_k^+ &= \hat{x}_k^- + K_k(z_k - h(\hat{x}_k^-, \hat{l}_k)) \\ P_k^+ &= (I - K_k H_{x,k}) P_k^-,\end{aligned}$$

where

$$K_k = P_k^- H_{x,k}^T (H_{x,k} P_k^- H_{x,k}^T + H_{l,k} L_k H_{l,k}^T + R_k)^{-1},$$

where $\hat{l}_k$ is the best estimate of the location of the landmark with covariance L_k, and where $H_{x,k}$ and $H_{l,k}$ are the Jacobians with the partial derivatives of the measurement function $h(\cdot,\cdot)$ with respect to the robot's location x and the landmark's location l respectively,

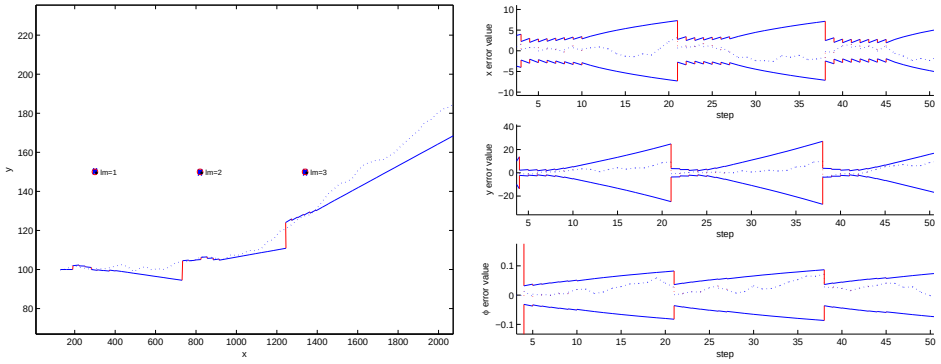
$$\begin{aligned}H_{x,k} &= \begin{bmatrix} \frac{\partial h_x}{\partial x_{[x]}} & \frac{\partial h_x}{\partial x_{[y]}} & \frac{\partial h_x}{\partial x_{[\phi]}} \\ \frac{\partial h_y}{\partial x_{[x]}} & \frac{\partial h_y}{\partial x_{[y]}} & \frac{\partial h_y}{\partial x_{[\phi]}} \\ \frac{\partial h_\phi}{\partial x_{[x]}} & \frac{\partial h_\phi}{\partial x_{[y]}} & \frac{\partial h_\phi}{\partial x_{[\phi]}} \end{bmatrix}_{x=\hat{x}_k^-, l=\hat{l}_k} \\ &= \begin{bmatrix} -\cos(x_{[\phi]}) & -\sin(x_{[\phi]}) & \begin{array}{l}-(l_{[x]} - x_{[x]})\sin(x_{[\phi]}) \\ +(l_{[y]} - x_{[y]})\cos(x_{[\phi]})\end{array} \\ \sin(x_{[\phi]}) & -\cos(x_{[\phi]}) & \begin{array}{l}(x_{[x]} - l_{[x]})\cos(x_{[\phi]}) \\ -(l_{[y]} - x_{[y]})\sin(x_{[\phi]})\end{array} \\ 0 & 0 & -1 \end{bmatrix}_{x=\hat{x}_k^-, l=\hat{l}_k},\end{aligned} \tag{8.1}$$

and

$$\begin{aligned}H_{l,k} &= \begin{bmatrix} \frac{\partial h_x}{\partial l_{[x]}} & \frac{\partial h_x}{\partial l_{[y]}} & \frac{\partial h_x}{\partial l_{[\phi]}} \\ \frac{\partial h_y}{\partial l_{[x]}} & \frac{\partial h_y}{\partial l_{[y]}} & \frac{\partial h_y}{\partial l_{[\phi]}} \\ \frac{\partial h_\phi}{\partial l_{[x]}} & \frac{\partial h_\phi}{\partial l_{[y]}} & \frac{\partial h_\phi}{\partial l_{[\phi]}} \end{bmatrix}_{x=\hat{x}_k^-, l=\hat{l}_k} \\ &= \begin{bmatrix} \cos(x_{[\phi]}) & \sin(x_{[\phi]}) & 0 \\ -\sin(x_{[\phi]}) & \cos(x_{[\phi]}) & 0 \\ 0 & 0 & 1 \end{bmatrix}_{x=\hat{x}_k^-, l=\hat{l}_k}.\end{aligned} \tag{8.2}$$

8.2.2 Correcting with Landmark Measurements

With the instantiated EKF equations we can estimate the location of the robot when it is placed in an environment with identifiable landmarks. We supply the robot with a list of landmarks containing the location of the landmarks in global coordinates together with an indication of the uncertainty in these locations. The EKF is initialized with the true initial position and some uncertainty.



(a) True (dotted, blue), predicted (solid, blue) and corrected (solid, red) trajectory of robot driving with unique landmark identification. The robot senses three landmarks.

(b) Predicted (dotted, blue) and corrected (dotted, red) estimation error and predicted (solid, blue) and corrected (solid, red) estimated 95% confidence intervals for x, y, and orientation.

Figure 8.1: Detecting identifiably landmarks.

Figure 8.1 shows the resulting trajectories after the robot has observed three landmarks. In Figure 8.1(a) we have plotted the true and estimated trajectory of the robot. While the robot is sensing the landmarks, the state estimates seem to be close to the true trajectory. Figure 8.1(b) confirms this, in which we plotted the estimation error and 95% confidence intervals. We clearly recognize the three phases in which the robot senses a landmark. We see that the uncertainty increases when there is no observable landmark and quickly decreases when there is a landmark.

8.2.3 EKF Linearization Errors

The landmark sensor that we use is governed by a nonlinear measurement model. The nonlinearity can cause linearization errors in the estimation. Increasing the x and y coordinates of the initial state estimate results in similar estimation results as in Figure 8.1, since the nonlinearities in the measurement model are caused by the orientation of the robot. When we adjust the orientation of the initial state estimate such that it is not close to the true orientation, the EKF suffers from significant estimation problems.

To illustrate this, consider Figure 8.2. The figure shows the true and estimated trajectory of the EKF when the initial orientation estimate is large, whereas the true initial orientation is zero. When the first measurement arrives, the first correction brings the state estimate far from the true state at that time. It takes a number of corrections before the state estimate gets in the neighborhood of the true state.

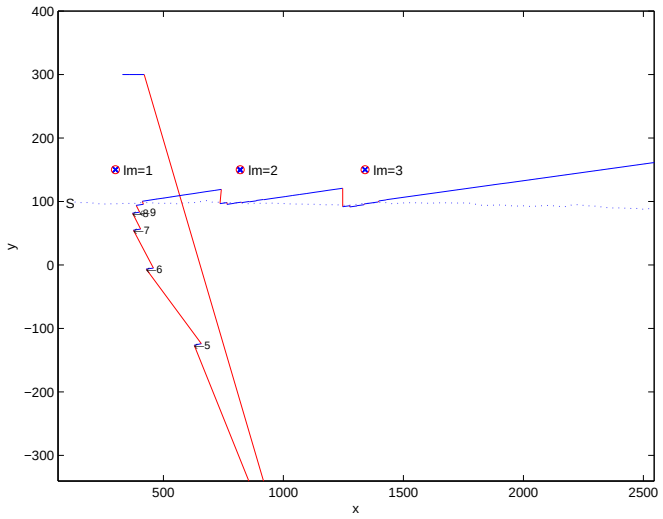


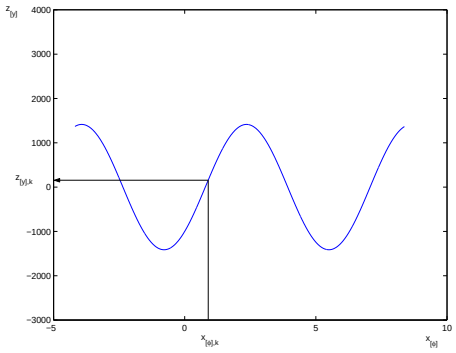
Figure 8.2: Robot starts at S. Initial orientation estimate is larger than true orientation, resulting in corrected state estimate far from true location. True (dotted, blue), predicted (solid, blue) and corrected (solid, red) trajectories, with true (blue x) and predicted (red o) measurements. Numbers indicate step numbers at which robot makes measurements.

To understand how these linearization errors occur, we look at an arbitrary EKF correction step. Consider a correction step of the EKF in which we include all of the measurement residual. Thus, we let $K = H_{x,k}^{-1}$. We can then rewrite the correction that the EKF performs as

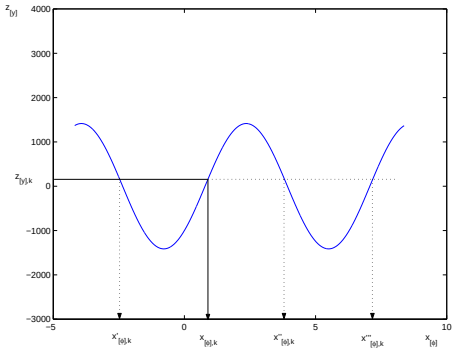
$$\begin{aligned}
\hat{x}_k^+ &= \hat{x}_k^- + K_k(z_k - h(\hat{x}_k^-)) && (8.3)\\
&= \hat{x}_k^- + H_{x,k}^{-1}(z_k - \hat{z}_k) \\
&= \hat{x}_k^- + H_{x,k}^{-1} z_k - H_{x,k}^{-1}\hat{z}_k && (8.4)\\
&\approx \hat{x}_k^- + x_{z,k} - \hat{x}_k^-. && (8.5)\\
&= \hat{x}_k^- + \Delta x_k,
\end{aligned}$$

where $x_{z,k}$ is the state from which the true measurement was made, and where Δx_k is the *state residual*, that is, the difference between the state from which the measurement was made and the prior state estimate. Thus, instead of considering the correction step of the EKF as including measurement residual into the state estimate, we can consider it as including state residual into the state estimate.

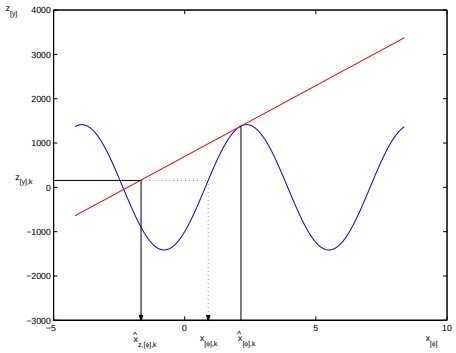
Seen from this perspective, the EKF correction determines what the state was that caused the true measurement, then computes the difference between this state and the estimated state, and then incorporates this state residual. The state residual may be incorporated partly depending on the uncertainty in the prior estimate and measurements.



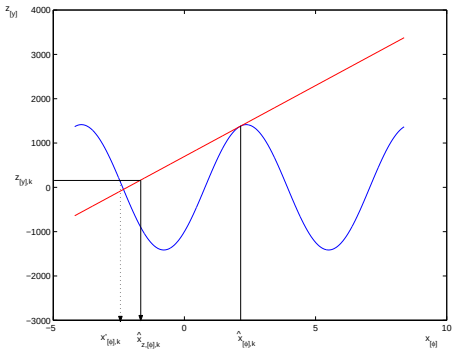
(a) Nonlinear measurement relation between robot's orientation and measured y coordinate. At true orientation $x_{[\phi],k}$, the robot measures $z_{[y],k}$.



(b) Inverse nonlinear measurement function. The true measurement of $z_{[y],k}$ can have been made from different states: $x_{[\phi],k}$, $x'_{[\phi],k}$, etc.



(c) Linearized measurement function at latest state estimate $\hat{x}_{[\phi],k}$ (red) indicating one state from which true measurement $z_{[y],k}$ was made: $\hat{x}_{z,[\phi],k}$. Notice that this is not the true state from which the measurement was made, $x_{[\phi],k}$.



(d) With a different true state, $x'_{[\phi],k}$, giving the same true measurement $z_{[y],k}$, the linearization error is smaller, altough the estimation error is higher.

Figure 8.3: Linearization error in EKF.

To visualize how linearization errors occur from this perspective, consider Figure 8.3(a), which shows the measurement function component that relates the orientation $x_{[\phi]}$ of the robot to the measured y coordinate $z_{[y]}$ of a landmark. At a time step k, the robot is in true state x_k with true orientation $x_{[\phi],k}$, resulting in a true y measurement $z_{[y],k}$. Ideally the EKF would find this true state x_k from which the measurement was made and use that state to adjust its state estimate.

Keeping uncertainty in measurement and landmark out of consideration, Figure 8.3(b) shows what the inverse of the measurement function would give for the state from which the measurement was made. In fact, we see that given the true measurement, there are multiple possibilities for these states.

The EKF does not have access to the inverse of the measurement function. Instead, it assumes that it can linearize the measurement function around the current state estimate. Figure 8.3(c) shows how this results in an estimate $\hat{x}_{z,[\phi],k}$ of the orientation of the state from which the measurement was made. Clearly there is a large difference between the true $x_{[\phi],k}$ and the estimated $\hat{x}_{z,[\phi],k}$ from which the measurement was made; there is a large linearization error.

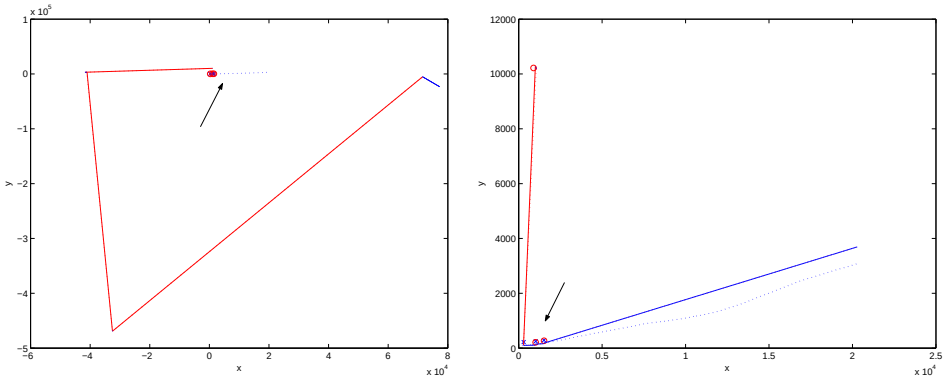
Notice that if the prior estimation error is low, that then the linearization error is more likely to be low than when the prior estimation error is high. The nonlinear function is then linearized more adequately. However, the linearization error may be low again or even reduce to zero, when the prior estimation error is high, as we can see in Figure 8.3(d).

Notice also that the figures only show the values of $z_{[y]}$ at different values of $x_{[\phi]}$. The state residual in these figures indicates what amount the orientation of the state estimate has to change in order to get in the neighborhood of the state from which the measurement was made. The state residuals in the figures will however not be included completely in the state estimate, due to the dependence of other measurement variables on the orientation of the robot, and due to weighting of the uncertainty in the prior estimate and measurement.

8.2.4 IEKF Linearization Errors

A way to reduce linearization errors may be by using the Iterated EKF instead of the EKF. To illustrate that this can be beneficial, Figure 8.4 shows another state estimation run using the landmark sensor model. The robot had to globally localize itself. In Figure 8.4(a) the EKF estimated the states. We see that the state estimates are far from the true locations, which are pointed out by the arrow. Figure 8.4(b) shows the same run, only now the states are estimated by the IEKF. Clearly, the state estimates of the IEKF are significantly closer to the true states than those of the EKF.

As we know from Chapter 5, the IEKF reduces the linearization error by re-linearizing the measurement function once it has corrected its prior



(a) Estimated trajectory using EKF. Estimated trajectory far from true trajectory.

(b) Estimated trajectory using IEKF. Estimated trajectory quickly moves toward true trajectory when a measurement arrives.

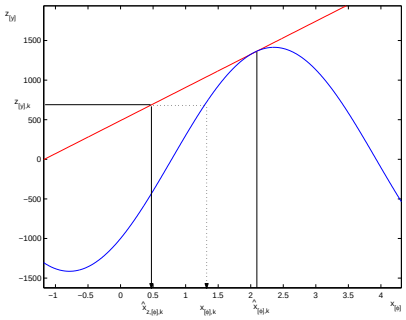
Figure 8.4: State estimation with EKF and IEKF. True (dotted, blue), predicted (solid, blue) and corrected (solid, red) state trajectories, with measurements to the three landmarks. Arrow indicates region with landmarks.

state estimate. The IEKF tries to find the best estimate of the state that resulted in the measurement. Whereas the EKF assumes that it can find the state that produced the measurement with one linearization around the prior state estimate, the IEKF repeatedly adjusts this state estimate to see if it can find a better state.

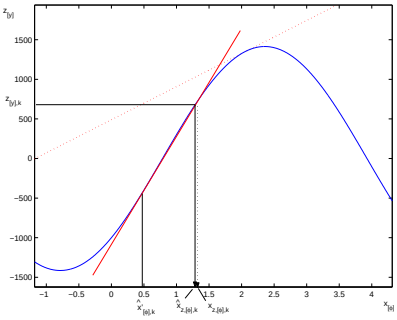
To illustrate this, consider Figure 8.5, which shows how re-linearization of the IEKF results in a lower linearization error. The sub-figures show how the orientation of the state of the robot is related to the y coordinate of the measurements at a certain time. Figure 8.5(a) shows how the first linearization results in a relatively large linearization error. Figure 8.5(b) shows that by re-linearizing the measurement function at the measurement state found in the first linearization, the linearization error reduces significantly. The IEKF uses the in this way estimated state from which the measurement was made to determine the state residual.

In the illustration, the measurement function is close to linear over the trajectory from the state found after one linearization and the true state. In the EKF example of Figure 8.3 the measurement function is not close to linear over the trajectory from the first found state and the true state. Figure 8.5(c) shows how the IEKF would re-linearize the found measurement state from the EKF. We see that the linearization error in fact becomes larger due to the re-linearization.

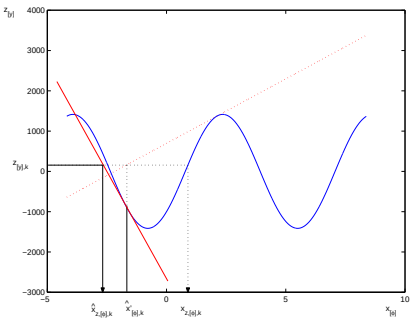
The reason that the IEKF does achieve good results lies in the fact that the orientation of the state estimate is not only adjusted based on the



(a) First linearization step. Nonlinear measurement function (blue) is linearized at estimated state $\hat{x}_{[\phi],k}$, resulting in state $\hat{x}_{z,[\phi],k}$ as state from where measurement was made. Linearization error is large.

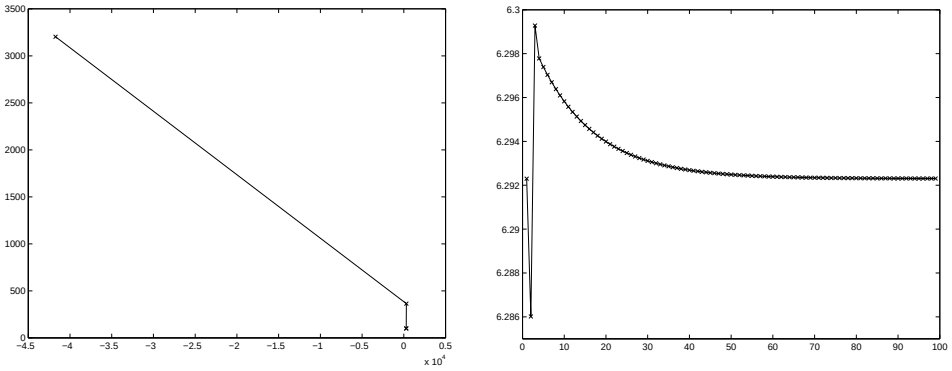


(b) Second linearization step. Nonlinear measurement function (blue) is re-lineared (solid, red) at estimated state $\hat{x}_{z,[\phi],k}$ from which measurement was made, resulting in a new state from where measurement was made. The linearization error is significantly smaller.



(c) Since measurement function is not close to linear between first found state $\hat{x}_{z,[\phi],k}$ and true state $x_{[\phi],k}$, the re-linearization (solid, red) does not reduce the linearization error.

Figure 8.5: Linearization error reduction in IEKF.



(a) Adjustment of x and y state estimates to find best state from which a measurement was made. Final resulting x and y are at the bottom-right.

(b) Adjustment of orientation estimate to find best state from which a measurement was made.

Figure 8.6: IEKF state estimate adjustment.

measured y coordinate, but also based on the measured x coordinate and the measured orientation. It is constrained by all three measured variables. This has as result that the IEKF does find a state close to the true state from which a measurement was made. Figure 8.6 shows one step of the IEKF in which the IEKF clearly adjusts the intermediate state estimates to find the state from which the measurement was made.

8.3 Type Based Landmark Localization

In this section we assume that we again have a sensor that measures the location of landmarks from the robot's point of view. However, the robot is now unable to uniquely identify the landmark to which the measurement was made. Instead, we assume it is able to identify the type of landmark and that its map contains the global locations of the different landmarks labeled with the type and some unique identification.

If the landmark detection sensor cannot uniquely determine to which landmark it makes an observation we have to create some extra administration around the standard KF equations that deals with this. The idea is that by assuming the different landmarks to be the true landmark of the measurement, the robot can maintain different location beliefs. By keeping track of the probabilities of observing measurements while being at a certain location, we are able to separate location beliefs that are more likely to represent the true location. We keep track of the probabilities of sequences of measurements occurring. Similar work, though less detailed and

general, has been done by Roumeliotis, who named this approach *Multiple Hypothesis Tracking* [34].

8.3.1 KF Modeling

Incorporating Acting. Assume that at step k we have n state beliefs. Let us denote the set of state estimate-uncertainty pairs as S_k. Assume that each element S_k^j has a probability of representing the true state. If the robot performed an action a_{k-1}, then we include the relative displacement information resulting from this action in each of the state estimates using the KF prediction equations. This results in n prior state beliefs.

We can combine the different prior state beliefs into one combined belief $\mathbf{Bel}^-(x_k)$ as the sum of the different state beliefs, weighted by the probabilities that they represent the true state. Using the theorem of total probability, we obtain

$$\mathbf{Bel}^-(x_k) = \sum_{i=1}^{n} Bel^-(x_k|S_k^i) \cdot P(S_k^i|z_1, a_1, \ldots, z_{k-1}, a_{k-1}) \tag{8.6}$$

The first term on the right hand side of equation (8.6) is one of the n prior state beliefs as computed by the KF prediction step. The second term in the same expression is the probability of this state belief representing the true state. We rewrite this term using Bayes' rule as

$$\begin{aligned} &P(S_k^i|z_1, a_1, \ldots, z_{k-1}, a_{k-1}) \\ &\quad = \frac{P(a_{k-1}|z_1, a_1, \ldots, z_{k-1}, S_k^i)}{P(a_{k-1}|z_1, a_1, \ldots, z_{k-1})} \cdot P(S_k^i|z_1, a_1, \ldots, z_{k-1}). \end{aligned}$$

Assuming that the probability of the action at step $k-1$ is independent of the probability of S_k^i representing the true state at step k, the first term on the right hand side of this expression becomes 1. Thus, the probability of S_k^i after action a_{k-1} is the same as the probability of S_k^i before the action. With this, the combined prior belief becomes

$$\mathbf{Bel}^-(x_k) = \sum_{i=1}^{n} Bel^-(x_k|S_k^i) \cdot P(S_k^i|z_1, a_1, \ldots, z_{k-1}). \tag{8.7}$$

Incorporating Sensing. When a measurement arrives, the correspondence model provides m corresponding landmarks with a correspondence probability. Let us denote L_k as the set of corresponding landmarks, and let L_k^j be an element of this set, consisting of the location and uncertainty of the j^{th} corresponding landmark.

Since every corresponding landmark can be the landmark to which the measurement was made, we consider every combination of prior state belief

and landmark. Thus, for every landmark L_k^j, and every prior state belief S_k^i, we compute

$$Bel^+(x_k|S_k^i, z_k \widehat{=} L_k^j), \tag{8.8}$$

using the KF correction step from the last section. This results in nm posterior state estimates.

We can combine the beliefs in these different state beliefs into one combined posterior belief $\mathbf{Bel}^+(x_k)$, which combines the state beliefs weighted by their probabilities of representing the true state,

$$\begin{aligned}&\mathbf{Bel}^+(x_k)\\ &\quad = \sum_{j=1}^{m}\sum_{i=1}^{n} Bel^+(x_k|S_k^i, z_k \widehat{=} L_k^j) \cdot P(S_k^i, z_k \widehat{=} L_k^j|z_1, a_1, \ldots, z_{k-1}, a_{k-1}).\end{aligned} \tag{8.9}$$

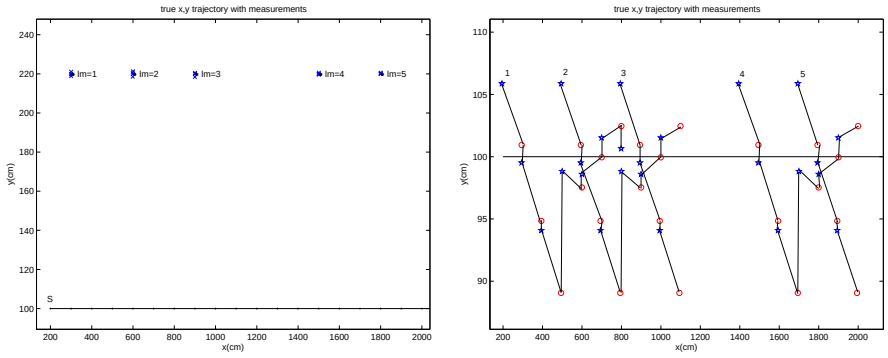
The first term in this expression is one of the posterior state beliefs as computed by the KF correction step, assuming a particular prior state belief S_k^i and a particular corresponding landmark L_k^j. The second term in the expression is the probability of observing this landmark, while having this state belief. We can rewrite this term into

$$\begin{aligned}&P(S_k^i, z_k \widehat{=} L_k^j|z_1, a_1, \ldots, z_{k-1}, a_{k-1})\\ &\quad = P(S_k^i|z_1, a_1, \ldots, z_{k-1}, a_{k-1}, z_k \widehat{=} L_k^j) \cdot P(z_k \widehat{=} L_k^j).\end{aligned} \tag{8.10}$$

In this expression, the second term on the right hand side is the prior probability that a measurement corresponds to a landmark; this probability is given by the correspondence model. The first term on the right hand side represents the probability of S_k^i representing the state belief, taking into account all the measurements and relative displacement, and assuming that the latest measurement corresponds to landmark L_k^j. Using Bayes' rule and the Markov assumption we rewrite this term into

$$\begin{aligned}&P(S_k^i|z_1, a_1, \ldots, z_{k-1}, a_{k-1}, z_k \widehat{=} L_k^j)\\ &\quad = \frac{P(z_k \widehat{=} L_k^j|z_1, a_1, \ldots, z_{k-1}, a_{k-1}, S_k^i)}{P(z_k \widehat{=} L_k^j|z_1, a_1, \ldots, z_{k-1}, a_{k-1})} \cdot P(S_k^i|z_1, a_1, \ldots, z_{k-1}, a_{k-1})\\ &\quad = \mu \cdot P(z_k \widehat{=} L_k^j|S_k^i) \cdot P(S_k^i|z_1, a_1, \ldots, z_{k-1}, a_{k-1}),\end{aligned} \tag{8.11}$$

where the last term is the prior probability of S_k^i; the second term is the probability that the measurement corresponds to landmark L_k^j given that S_k^i represents the state belief, which we compute using the measurement model; μ is a normalizing factor, ensuring that the probabilities of a measurement



(a) Environment setting with 5 landmarks and true state trajectory (solid, black) and measurements (blue x). Robot starts at S.

(b) Estimated trajectories kept for possible given different assumptions about first encountered landmark. Corrected state estimates (blue stars); predicted state estimates (red circles).

Figure 8.7: Environment and estimated trajectories.

corresponding to a landmark given a particular state belief sum to 1. This factor is computed as

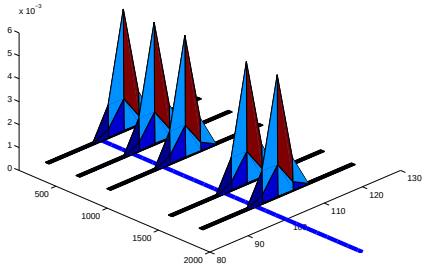
$$\mu = \left(\sum_{j=1}^{n} P(z_k \widehat{=} L_k^j | S_k^i) \cdot P(S_k^i | z_1, a_1, \ldots, z_{k-1}, a_{k-1}) \right)^{-1}. \tag{8.12}$$

8.3.2 Correcting Locations

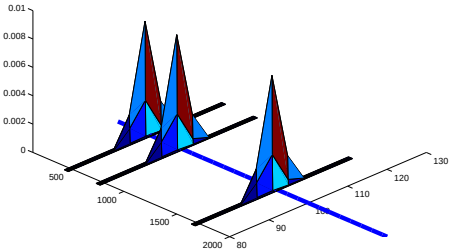
With the described framework we can now make the robot drive around in an environment where it can only identify the type of landmarks. For simplicity, assume that the robot drives in an environment with one type of landmark, and that the robot is not influenced by system noise. It has a map with the locations of these landmarks, together with some uncertainty.

In Figure 8.7(a) we have plotted the environment of the robot together with the true trajectory. The robot starts at S, but it is initialized with uniform belief. While passing the landmarks, the robot receives measurements which it uses to estimate its position. Figure 8.7(b) shows the resulting estimated state trajectories that the robot keeps for possible while navigating. The longer the state trajectory, the longer the robot keeps it into account as possible true state trajectory. Let us go into more detail on how the robot achieved this result.

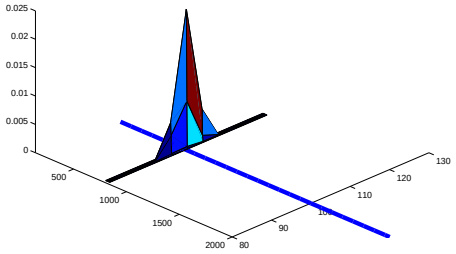
The first measurement that the robot receives is to landmark 1. However, the robot does not know this; all it knows is that it is near one of the



(a) After first measurement. The measurement can correspond to any of the 5 landmarks, and thus there are 5 state beliefs.



(b) After second measurement. The first measurement could not have corresponded to landmark 3 or 5, since then the robot would not sense a landmark after the displaced distance. The 3 possibilities that are left assume landmark 1, 2, or 4 corresponding to the first measurement, and 2, 3, and 5 respectively corresponding to the second.



(c) After third measurement. The second measurement could not have corresponded to landmark 3 or 5, since then the robot would not sense a landmark after the displacement distance. There is one possibility state belief left, the belief that assumed landmark 1 as first, landmark 2 as second, and landmark 3 as third corresponding landmark.

Figure 8.8: Location estimation with type identifiable landmarks.

5 landmarks. Therefore, using the global localization technique from the previous chapter, it creates 5 posterior state estimates, each conditioned on a different landmark. Figure 8.8(a) shows the combined posterior belief at that time.

After some driving, the robot receives a measurement to the next measurement. Again, the robot does not know this, and thus it incorporates each possible landmark into each of the 5 state estimates, resulting in 25 state estimates. Thresholding on the probability of each of the state beliefs being the true state estimate, we are left with 3 state estimates. These state estimates are depicted in Figure 8.8(b) and correspond to the scenarios in which the first landmark encountered was 1, followed by 2, or where the first was 2, followed by 3, or where the first was 4, followed by 5. The scenarios in which the robot first encounters 3 and then 4, 1 and then 4, etc. are too unlikely to proceed.

After some more driving the robot encounters the third landmark, landmark 3. The robot incorporates each of the 5 possible landmarks into the 3 state beliefs, resulting in 15 posterior state beliefs. Thresholding the probabilities of the state beliefs leaves one possible state belief, as shown in Figure 8.8(c). This estimate corresponds to the case in which the robot first encountered landmark 1, then 2, and then 3. The robot has uniquely determined its position.

8.3.3 Issues

When implementing the framework some issues have to be taken into account.

Threshold Selection. The number of state estimates grows with every measurement where there is more than one corresponding landmark. To keep the computational costs low, state beliefs that are almost the same can be combined, and state beliefs that have high uncertainty can be removed.

Co-existing State Beliefs. In our example the robot encountered enough landmarks to uniquely determine which of the state estimates kept as possible corresponded to the true state estimate. The environment provided enough information. However, if there is not enough information, multiple state beliefs can keep existing. If in our example the robot would not have encountered any other landmarks besides the first, then 5 state estimates would have co-existed.

No State Beliefs. On the other hand, depending on the method used for thresholding, it may happen that all state beliefs become so unlikely that none of them is seen as possible true state belief. In that case, the robot may choose to forget its state beliefs and begin again.

8.4 Summary

We can use KFs to perform localization when we have access to measurements that measure the location of landmarks from the robot's point of view. We hereby make a distinction between the situation in which the robot can uniquely identify the landmark to which a measurement is made, and the situation in which it can only determine a group of landmarks to which the measurement is made.

Using the landmark sensor model from Chapter 6 we can instantiate EKF correction equations. With these equations we can correct the location estimate when landmarks are in sight. However, if we use the EKF, then this only works well if the linearization errors are negligible; if the uncertainty in the orientation is relatively high, then this may not be the case, resulting in serious estimation errors.

By considering the correction step of KFs as including some amount of state residual into the prior state estimate, we can explain in more detail what linearization errors are and in particular how the Iterated EKF successfully reduces them by iteratively trying to find the state from which the true measurement was made.

If the robot can not uniquely determine to which landmark a measurement corresponds, then we can create a framework around the KF that allows multiple state beliefs. By keeping track of the probability of observing sequences of measurements, while being at certain states, we can determine which state beliefs are more likely to represent the true state. In order to keep this framework computationally efficient, the number of considered state beliefs should be kept low, while preventing the robot from loosing all its belief.

Chapter 9

Conclusion

In this work we have thoroughly discussed the problem of Robot Localization, the problem of state estimation of noisy dynamic systems using Kalman Filters, and finally how to apply Kalman Filter techniques to solve the Robot Localization problem. In order of appearance we discussed the following topics.

Robot Localization.

- From a practical point of view we discussed the need for a robot to localize itself while navigating through an environment. We identified three different localization instantiations, *position tracking*, *global localization*, and the *kidnapped robot problem*. In order to determine what information a robot has access to regarding its position, we discussed different sources of information and pointed out advantages and disadvantages. We concluded that due to the imperfections in actuators and sensors due to noise sources, a navigating robot should localize itself using information from different sensors.

- In order to formalize this, we considered the localization problem in probabilistic context as a Bayesian estimation problem. We defined the belief of the robot as the probability density over the space of possible locations, conditioned on types of location information it has access to. With understanding and application of Bayes' rule and the Markov assumption, we obtained a localization formula that formalizes the incorporation of relative and absolute position measurements into this belief. We found that in order to implement the localization formula we need a concrete representation of the location space, along with models that describe the influence of actions on the location, and that describe the relation between measurements and locations. Discussing several methods that implement the formula, we found different ways of implementing these.

Kalman Filters.

- We thoroughly discussed the basics of the Kalman Filter. We looked at the assumptions that the Kalman Filter poses on the system of which it estimates the state: a *linear dynamic system* with *linearly related measurements*, corrupted by *Gaussian distributed, white, zero-mean noise*. We looked at the implications of these assumptions, which led us to the equations that form the Linear Kalman Filter. We created better understanding of the Kalman Filter by describing the meaning of the different equations. We proved that under the posed assumptions the LKF optimally estimates the state of a system in minimum error variance sense, and we generalized the derived equations to allow for systems with multiple input parameters.

- Since the Linear KF only works for systems that can be described with linear system and measurement models, we discussed how we can use linearization techniques to obtain KF extensions that estimate the state of nonlinear problems. If we more or less now the state trajectory of the system we can use the Perturbation KF to estimate the state, if the perturbation around the trajectory will stay small. The disadvantage of this approach is that the estimation error can increase when the true state follows a significantly different trajectory than as modeled. The PKF does not take state estimates into account when linearizing. We argued that the Extended Kalman Filter improves this by incorporating the latest state estimates into the linearization process. In line with this, the Iterated Extended Kalman Filter repeatedly linearizes a corrected state estimate to find the best state estimate used for inclusion of a measurement. We showed how we can extend the formulation of the EKF to systems and measurements with any number of parameters, as long as the uncertainties in them are independent of each other. We furthermore pointed out the key ideas of alternative extensions, which do not require Jacobian matrices to be calculated and allow for other than Gaussian distributions. These extensions may further improve the estimation performance.

Kalman Localization.

- In order to show how we can use Kalman Filters in the Robot Localization context we derived models for acting and sensing and discussed the true behavior of these systems in the light of noise. Among others, we concluded that the Gaussian noise assumption is not always valid, and that uncertainty in the orientation of the robot causes the uncertainty in the location to increase.

- We used the models in showing how to use the EKF for each of the three posed localization problem instances. First, we discussed pre-

dictive position tracking, and predictive position tracking with corrections. While doing this, we pointed out several KF aspects, illustrated with charts. It is important to have accurate levels of system and measurement noise, such that measurements are incorporated to the right extent. Having access to fewer measurements increases the uncertainty in state estimates, which may lead to inconsistent state estimates. Second, we looked at the kidnapped robot problem, and found that we can use validation gates to detect a kidnap. Third, we discussed how we can use the EKF with full state sensor to perform global localization. We looked at possibilities to represent ignorance about the location, and showed the use of forced full measurement usage.

- We discussed the use of landmarks in localization with Kalman Filters. We used the model of a landmark detecting sensor that perfectly identifies landmarks to illustrate the errors that can occur due to nonlinearities in the models. We showed how the EKF fails when the nonlinearities are too large due to linearization errors, and how the Iterated EKF deals with this. We extended the KF framework in order to be able to deal with measurements that can correspond to different reference points, and showed this framework in action.

With the presented work we have given an in-depth theoretical discussion of the use of Kalman Filters in the Robot Localization problem. We have pointed out advantages and disadvantages of the different techniques, while keeping in mind a practical perspective. We have discussed the use of Kalman Filter based techniques for all posed Robot Localization problem instances, illustrated with experiments.

9.1 Future Work

This work can be used as theoretical basis for further studies in a number of different directions. First, in the localization chapters we discussed some localization methods implementing the localization formula. They give a starting point for further readings into the different localization methods. Second, also the KF chapters form a basis for further studies. With the basic derivations of the KF and some of the extensions, it is interesting to look more detailed into other extensions relaxing more assumptions. Third, the experiments we performed in the chapters on KFs and localization were performed in a simulator and merely meant as illustrations to concepts of the KF. In order to draw conclusions on the practical utility of the KF, the theory discussed in these chapters can be applied in practice.

Throughout the progress of this work, interesting ideas for future work came up. One of these is to look for ways to make the KF applicable in

dynamic environments, since the real world is not static. In this work we discussed the use of KFs and localization of a robot using static landmarks of which it knows the location with some uncertainty. It is interesting to look at what would happen if the landmarks are not static, but move through the environment according to some motion model. Using validation gates the robot could perhaps learn this model. Multiple robots may together learn a rough motion model of the moving objects and share their information regarding their own and the landmark's location.

Bibliography

[1] C. Adams, *Is 'dead reckoning' short for 'deduced reckoning'?* (2002). Available at http://www.straightdope.com/mailbag/mdeadreckoning.html.

[2] K. Arras and N. Tomatis, Improving robustness and precision in mobile robot localization by using laser range finding and monocular vision, *Proceedings of the Third European Workshop on Advanced Mobile Robots*, Zurich, Switzerland (1999).

[3] H. Baltzakis and P. Trahanias, Hybrid mobile robot localization using switching state-space models. *Proceedings of the 2002 IEEE International Conference on Robotics and Automation*. Washington D.C., USA (2002), 366–373. Available at http://www.ics.forth.gr/~trahania/.

[4] J. Borenstein, Control and kinematic design of multi-degree-of-freedom robots with compliant linkage. *IEEE Transactions on Robotics and Automation* (1995). Available at http://citeseer.nj.nec.com/borenstein95control.html.

[5] J. Borenstein, H. Everett, L. Feng, and D. Wehe, Mobile robot positioning: Sensors and techniques. *Journal of Robotic Systems* **14**,4 (1997), 231–249. Available at http://citeseer.nj.nec.com/borenstein97mobile.html.

[6] J. Borenstein and L. Feng, Measurement and correction of systematic odometry errors in mobile robots. *IEEE Transactions on Robotics and Automation* **12** (1996), 869–880. Available at http://citeseer.nj.nec.com/borenstein96measurement.html.

[7] H. Bruyninkx, *Bayesian probability* (2002). Available at http://people.mech.kuleuven.ac.be/~bruyninc.

[8] W. Burgard, D. Fox, D. Hennig, and T. Schmidt, Estimating the absolute position of a mobile robot using position probability grids. *Proceedings of the Fourtheenth National Conference on Artificial Intelligence* (1996), 896–901. Available at http://citeseer.nj.nec.com/burgard96estimating.html.

[9] K. Capek, *Rossum's Universal Robots* (1920). Available at http://capek.misto.cz/english/.

[10] A. R. Cassandra, L. P. Kaelbling, and J. A. Kurien, Acting under uncertainty: Discrete bayesian models for mobile robot navigation. *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems* (1996). Available at http://citeseer.nj.nec.com/cassandra96acting.html.

[11] I. J. Cox, Blanche: Position estimation for an autonomous robot vehicle, *Autonomous Mobile Robots: Control, Planning, and Architecture (Vol. 2)*, IEEE Computer Society Press, Los Alamitos, CA (1991), 285–292.

[12] I. J. Cox and G. Wilfong (Editors), *Autonomous Robot Vehicles*, Springer-Verlag, New York (1990).

[13] J. L. Crowley, Mathematical foundations of navigation and perception for an autonomous mobile robot. *Reasoning with Uncertainty in Robotics* (1995), 9–51. Available at http://citeseer.nj.nec.com/crowley95mathematical.html.

[14] P. H. Dana, *The Global Positioning System* (2000). Available at http://www.colorado.edu/geography/gcraft/notes/gps/gps_f.html.

[15] H. Durrant-Whyte, *Kalman Filtering* (1990).

[16] A. Elfes, Occupancy grids: a stochastic spatial representation for active robot perception, *Proceedings of the Sixth Conference of Uncertainty in AI* (1990), 60–70.

[17] G. Evensen, The ensemble kalman filter: Theoretical formulation and practical implementation (2003). Available at http://www.chebucto.ns.ca/~aimet/enkf/.

[18] J. Foley, A. van Dam, S. Feiner, J. Hughes, and R. Phillips, *Introduction to Computer Graphics*, Addison-Wesley Publishing Company, Inc., USA (1997).

[19] D. Fox, W. Burgard, and S. Thrun, Markov localization for mobile robots in dynamic environments. *Journal of Artificial Intelligence*

Research **11** (1999), 391–427. Available at
http://citeseer.nj.nec.com/fox99markov.html.

[20] D. Fox, S. Thrun, W. Burgard, and F. Dellaert, Particle filters for mobile robot localization. *Sequential Monte Carlo Methods in Practice*. Springer. New York (2001). Available at
http://citeseer.nj.nec.com/fox01particle.html.

[21] M. Grewal and A. Andrews, *Kalman filtering: theory and practice*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey (1993).

[22] J. Hefferon, *Linear Algebra*, Colchester, VT (2003).

[23] R. Kalman, A new approach to linear filtering and prediction problems, *Transactions ASME Journal of Basic Engineering* **82** (1960), 35–44.

[24] E. Kiriy and M. Buehler, *Three-state Extended Kalman Filter for Mobile Robot Localization* (2002).

[25] D. Kortenkamp, R. Bonasso, and R. Murphy (Editors), *AI-based Mobile Robots: Case studies of successful robot systems*, MIT Press, Cambridge, MA (1998).

[26] E. Krotkov, R. Simmons, F. Cozman, and S. Koenig, Safeguarded teleoperation for lunar rovers: from human factors to field trials. *Proceedings of IEEE Planetary Rover Technology and Systems Workshop*. Minn., MN (1996). Available at
http://citeseer.nj.nec.com/krotkov96safeguarded.html.

[27] J. Leonard and H. Durrant-Whyte, Mobile robot localization by tracking geometric beacons, *IEEE Transactions on Robotics and Automation* **7**,3 (1991), 376–82.

[28] J. Manyika and H. Durrant-Whyte, *Data Fusion and Sensor Management, a decentralized information-theoretic approach*, Ellis Horwood Limited, Chichester, West Sussex (1994).

[29] P. S. Maybeck, *Stochastic models, estimation and control*, Academic Press, Inc., New York, USA (1979).

[30] Merriam-Webster, *Merriam-Webster's Collegiate Dictionary*. 10-th Edition. Merriam-Webster, Inc. Springfield, MA (1998). Available at
http://www.m-w.com.

[31] K. Murphy, *A brief introduction to Bayes' Rule* (2003). Available at
http://www.ai.mit.edu/~murphyk/Bayes/bayesrule.html.

[32] J. Rice, *Mathematical Statistics and Data Analysis*, Duxbury Press, Belmont, CA (1995).

[33] S. Roumeliotis, *Reliable Mobile Robot Localization* (1999). Available at http://www-users.cs.umn.edu/~stergios/.

[34] S. Roumeliotis and G. Bekey, Bayesian estimation and kalman filtering: A unified framework for mobile robot localization. *Proceedings of IEEE International Conference on Robotics and Automation*. San Francisco, California (2000), 2985–2992. Available at http://www-users.cs.umn.edu/~stergios/.

[35] S. Shoval and J. Borenstein, *Measurement Of Angular Position Of A Mobile Robot Using Ultrasonic Sensors* (1999). Available at http://citeseer.nj.nec.com/shoval99measurement.html.

[36] C. Silpa-Anan, S. Abdallah, and D. Wettergreen, Development of autonomous underwater vehicle towards visual servo control. *Proceedings of the Australian Conference on Robotics and Automation*. Melbourne, Australia (2000), 105–110. Available at http://citeseer.nj.nec.com/silpa-anan00development.html.

[37] A. Singhal, *Issues in Autonomous Mobile Robot Navigation* (1997). Available at http://citeseer.nj.nec.com/singhal97issue.html.

[38] R. Smith and P. Cheeseman, On the estimation and representation of spatial uncertainty, *International Journal of Robotics Research* **5**,4 (1987), 56–68.

[39] J. Taylor, *An introduction to error analysis*, University Science Books, Sausalito, CA (1997).

[40] S. Thrun, Bayesian landmark learning for mobile robot localization. *Machine Learning* **33**,1 (1998), 41–76. Available at http://citeseer.nj.nec.com/thrun98bayesian.html.

[41] S. Thrun, Probabilistic algorithms in robotics. *AI Magazine* **21**,4 (2000), 93–109. Available at http://citeseer.nj.nec.com/thrun00probabilistic.html.

[42] S. Thrun, *Robotic mapping: A survey*, Technical Report CMU-CS-02-111, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA (2002).

[43] S. Thrun, M. Beetz, M. Bennewitz, W. Burgard, A. Cremers, F. Dellaert, D. Fox, D. ahnel, C. Rosenberg, N. Roy,

J. Schulte, and D. Schulz, Probabilistic algorithms and the interactive museum tour-guide robot minerva. *International Journal of Robotics Research* **19**,11 (2000). Available at http://citeseer.nj.nec.com/article/thrun00probabilistic.html.

[44] S. Thrun, D. Fox, and W. Burgard, Probabilistic methods for state estimation in robotics. *Proceedings of the Workshop SOAVE'97*. VDI-Verlag (1997), 195–202. Available at http://citeseer.nj.nec.com/article/thrun97probabilistic.html.

[45] S. Thrun, D. Fox, W. Burgard, and F. Dellaert, Robust monte carlo localization for mobile robots. *Artificial Intelligence* **128**,1-2 (2001), 99–141. Available at http://citeseer.nj.nec.com/thrun01robust.html.

[46] R. van der Merwe, de N. Freitas, A. Doucet, and E. Wan, The unscented particle filter. *Advances in Neural Information Processing Systems 13* (2001). Available at http://citeseer.nj.nec.com/article/vandermerwe00unscented.html.

[47] E. Wan and R. van der Merwe, The unscented kalman filter for nonlinear estimation. *Proceedings of the IEEE Symposium 2000 (AS-SPCC)*. Alberta, Canada (2000). Available at http://citeseer.nj.nec.com/wan00unscented.html.

[48] C. M. Wang, Location estimation and uncertainty analysis for mobile robots, *Proceedings of the IEEE International Conference on Robotics and Automation* (1988), 1230–1235.

[49] E. Weisstein, *MathWorld* (2003). Available at http://mathworld.wolfram.com/.

[50] G. Welch and G. Bishop, *An Introduction to the Kalman Filter*, Chapel Hill (2001). SIGGRAPH 2001.

[51] M. Welling, *The Kalman Filter* (2000). Available at http://www.cs.toronto.edu/~welling/.

[52] K. E. Willcox, *Modeling Engineering Systems* (2002). Available at http://www.mit.edu/afs/athena/org/a/aa-math/www/modules/.

[53] F. Worsley, *Shackleton's boat journey*, Jove Publications, Inc., New York, N.Y. (1978).

Appendix A

Simulator

In order to experiment with different KFs, we implemented a simulator that provides us with the necessary functionality to analyze KFs using different system and measurement settings. The simulator consists of a set of Matlab functions that implement a *truth model*, a *Kalman Filter model*, and visualization tools. The truth model simulates the real system and sensor systems; the KF model consists of the KF instantiation used for estimating the state of the modeled real system and sensors; the visualization tools provide functionality to display various kinds of plots and animations in a consistent way.

With the current implementation, the simulator contains truth and Kalman Filter model implementations for the drive and sensor systems described in this work. Moreover, the simulator supports analyses using the Linear KF, Extended KF, Iterated EKF, and KF with multiple state beliefs. The simulator allows any number of state influencers, using any state sensors, that again can have any number of parameters. Although in the current implementation the system and noise noise sources are all Gaussian distributed, the code could easily be adjusted to experiment with different kinds of noise.

The simulator can generate true state and measurement trajectories independent of KF analysis. Once the data has been generated, the user can experiment with the different KF implemententations on the same true data to investigate the effects of different models or noise settings. Besides this, the user can also regenerate true measurement trajectories, without regenerating state trajectories. This comes in useful when analyzing measurement noise influences.

The code including comments on use and adjustment can be found at http://www.negenborn.net/kal_loc/.