UNIVERSITEIT•STELLENBOSCH•UNIVERSITY
jou kennisvennoot • your knowledge partner

Numerical Integration for Probabilistic Reasoning

by

Jacobus Martin Louw

Bachelor of Engineering (Electrical and Electronic)

Final Report

Study leader: Dr CE van Daalen

2018

Declaration

By submitting this report electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: .....

Contents

List of Figures

List of Acronyms

CPD	-	conditional probability distribution
EKF	-	extended Kalman filter
KL	-	Kullback-Leibler
KF	-	Kalman filter
PDF	-	probability density function
PGM	-	probabilistic graphical model
RV	-	random variable
UKF	-	unscented Kalman filter

List of Notations

x	-	scalar
$\boldsymbol{x}$	-	vector
$\mathcal{E}[\boldsymbol{x}]$	-	expected value of vector
$\boldsymbol{x}^T$	-	transpose of vector
X	-	matrix
X^T	-	transpose of matrix
X^{-1}	-	inverse of matrix
$\|X\|$	-	determinant of matrix
$\mathrm{tr}(X)$	-	trace of matrix
$p(x)$	-	probability density function
$\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}, \Sigma)$	-	covariance form
$\mathcal{C}(\boldsymbol{x}; K, \boldsymbol{h})$	-	canonical form
$[0]$	-	zero matrix
$\mathbf{0}$	-	zero vector

List of Symbols

C	-	cluster
$\boldsymbol{h}$	-	potential vector
K	-	information matrix
δ	-	message in a cluster graph
η	-	normalisation coefficient
ψ	-	initial belief cluster
Σ	-	covariance matrix
σ	-	standard deviation
μ	-	mean

Abstract

Text in default language ...

Chapter 1

Introduction

Robot localisation is the process of estimating where a robot is located relative to its environment. It is essential for autonomous robots to have a precise knowledge of their location in order to navigate their surroundings.

When estimating the location of a robot, a map of the environment and the control input to the robot are usually available. The robot is also generally fitted with sensors which, together with beacons, measure where the robot is located relative to the map. Unfortunately the information of the robot's movements and measurements of its location are uncertain due to noise. The position and orientation of the robot should rather be estimated in a probabilistic manner from the information gathered from the sensors. The localisation techniques must therefore be able to deal with noise and describe the robot's location with a measure of uncertainty.

Probabilistic reasoning of systems with continuous random variables (RV) use integration for various operations. In most nonlinear systems these integrals cannot be calculated analytically and one must use numerical methods. Kalman filters have been used extensively for localisation. Techniques such as the extended Kalman filter use primitive numerical integration that can lead to inaccurate estimations, especially when measurements are also nonlinear. However, there are several alternative numerical techniques available that are more accurate.

A probabilistic graphical model (PGM) is a technique that is used to model systems with uncertainty in an organised manner. It describes the relationship between RVs and allows one to reason about their likelihood based on knowledge that is available to the system. Reasoning about random variables (RVs) in a PGM consists of a number of steps and one can easily identify where integration is used.

The end goal of this project is to investigate different techniques of numerical integration and to implement each technique in a PGM to reason about a nonlinear localisation problem. The techniques of numerical integration will be compared in terms of accuracy and efficiency.

Chapter 2

Gaussian Random Variables

The Gaussian or normal distribution is commonly used in probability theory, and is amongst other things, very easy to use. Although Gaussian distributions make severe assumptions such as the exponential decay of the distribution away from its mean, they are often surprisingly good estimations for real world distributions. The Gaussian distribution is a very important concept in this report, because all the probability distributions are approximated as Gaussian distributions. Key concepts, features and representations of the Gaussian distribution are discussed in this chapter and is based on the work of Peebles and Shi [1] and Koller and Friedman [2].

2.1 Covariance Form

The value of a random variable (RV) is of a random nature and is drawn from a certain function. In the case of a Gaussian RV, the value is drawn from a Gaussian distribution.

The univariate Gaussian distribution is a probability density function (PDF) of a single Gaussian RV. The covariance form fully describes a univariate Gaussian distribution by a mean μ and variance σ^2. The univariate Gaussian distribution of a RV x, denoted

$$x \sim \mathcal{N}(\mu, \sigma), \tag{2.1}$$

has a PDF that is defined by

$$p(x) = \eta \exp\left[\frac{-(x-\mu)^2}{2\sigma^2}\right], \tag{2.2}$$

with a normalisation coefficient

$$\eta = \frac{1}{\sqrt{2\pi\sigma^2}}. \tag{2.3}$$

The mean parameter μ describes the location of the peak of the distribution and the variance parameter σ^2 describes the tempo at which the distribution decays. The probability to draw a value of X out of a Gaussian distribution decreases exponentially as X moves farther away from the mean of the Gaussian distribution. The standard deviation of the Gaussian distribution is denoted as σ and it describes the spread of the PDF.

If x is a continuous RV, then the mean or expectation of x is

$$\mu = \mathcal{E}[x] = \int x \cdot p(x) dx. \tag{2.4}$$

The variance of x can be calculated as

$$\begin{aligned}\sigma^2 &= \mathcal{E}\left[(x - \mathcal{E}[x])^2\right] \\ &= \mathcal{E}[x^2] - (\mathcal{E}[x])^2\end{aligned} \tag{2.5}$$

Figure 2.1 indicates the mean μ and standard deviation σ of an univariate Gaussian PDF.

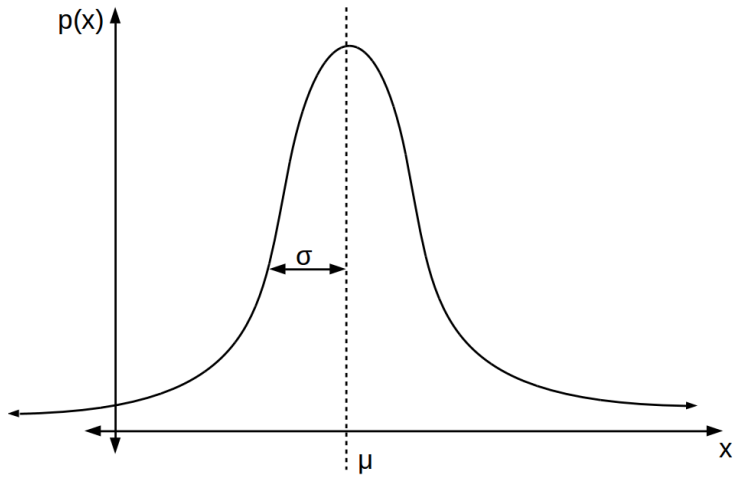

Figure 2.1: An univariate Gaussian PDF can be observed, where the mean is indicated with μ and the standard deviation is indicated with σ.

The definition of the Gaussian distribution can be extended to describe the PDF of N random variables (RVs). This is called the multivariate Gaussian distribution. The multivariate Gaussian distribution is described by an N-dimensional mean vector $\boldsymbol{\mu}$ and an $N \times N$ covariance matrix Σ. A random vector $\boldsymbol{x}$, with N random variables (RVs), is defined as

$$\boldsymbol{x} = [x_1 \ ... \ x_N]^T, \tag{2.6}$$

where $\boldsymbol{x}$ is drawn from a multivariate distribution which is described by a mean vector $\boldsymbol{\mu}$ and a covariance matrix Σ, defined as

$$\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}, \Sigma). \tag{2.7}$$

The PDF of $\boldsymbol{x}$ is then defined as

$$p(\boldsymbol{x}) = \eta \exp\left[-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu})^T \Sigma^{-1} (\boldsymbol{x} - \boldsymbol{\mu})\right], \tag{2.8}$$

where η is a normalisation coefficient and is defined as

$$\eta = \frac{1}{(2\pi)^{\frac{N}{2}} |\Sigma|^{\frac{1}{2}}}. \tag{2.9}$$

For this project it is required that Σ is positive definite and thus also nonsingular, which guarantees a determinant that is nonzero. Positive definite covariance matrices are also invertible, thus the alternative canonical or information form, which requires Σ^{-1}, can be used. The canonical form is discussed in the following section.

The mean vector of RV vector $\boldsymbol{x}$ is equal to the expectation of $\boldsymbol{x}$ and is defined as

$$\boldsymbol{\mu} = \mathcal{E}[\boldsymbol{x}]. \tag{2.10}$$

The covariance matrix Σ specifies the correlation between RVs and is equal to

$$\begin{aligned}\Sigma &= \mathcal{E}\left[(\boldsymbol{x} - \mathcal{E}[\boldsymbol{x}])^2\right] \\ &= \mathcal{E}[\boldsymbol{x}\boldsymbol{x}^T] - \mathcal{E}[\boldsymbol{x}]\mathcal{E}[\boldsymbol{x}]^T.\end{aligned} \tag{2.11}$$

2.1.1 Error Ellipse

The following section is based on an article by Abdi [3] and on a webpage from "Computer vision for dummies" [4].

A multivariate Gaussian distribution with RV vector

$$\boldsymbol{x} = [x_1 \; x_2]^T \tag{2.12}$$

can be visualised as a series of ellipsoidal contours around the mean vector $\boldsymbol{\mu}$. The contours are an indication of the correlation between x_1 and x_2 and also show the uncertainty of the PDF. Contours close to one another suggest a steep incline and contours that are farther apart suggest a gradual incline in the PDF. Hence, small and concentrated contours represent certain Gaussian distributions and contours that are larger and farther apart represent uncertain Gaussian distributions. The drawing of error ellipses is an effective way to indicate the mean, the uncertainty and the correlation of a 2D Gaussian distribution.

An ellipse has a major axis and a minor axis as shown in Figure 2.2. The major axis of the error ellipse is always aligned in the direction in which the Gaussian distribution varies most. This direction can be found by determining the eigenvectors of the distribution. Each eigenvector has a corresponding eigenvalue which indicates the spread of the distribution in the direction of the eigenvector.

We can use eigenvalue decomposition to factorise the covariance matrix Σ,

$$\Sigma = Q\Lambda Q^{-1}. \tag{2.13}$$

Each column of the matrix Q contains an eigenvector $\boldsymbol{v}$. The matrix Q is defined as

$$Q = [\boldsymbol{v_1} \; \boldsymbol{v_2}] = \begin{bmatrix} v_{11} & v_{21} \\ v_{12} & v_{22} \end{bmatrix}. \tag{2.14}$$

The matrix Λ is a diagonal matrix and each of its diagonal entries contains an eigenvalue λ. The matrix Λ is defined as

$$\Lambda = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}. \tag{2.15}$$

An error ellipse of a Gaussian PDF can be specified in terms of confidence levels, which is the probability that a value drawn from the distribution will fall inside the ellipse. Thus, differently sized ellipses can be plotted for different confidence levels. The lengths of the major and minor axes are sequentially specified as $2\sqrt{s\lambda_1}$ and $2\sqrt{s\lambda_2}$, where $\lambda_1 \geq \lambda_2$. The value of s is determined by the confidence level of the error ellipse. In the case of an error ellipse with a confidence level of 95%, s is set to 5.991. The Chi-Square distribution can be used to find s for other confidence levels, but it is beyond the scope of this document.

The orientation α of the error ellipse is shown in Figure 2.2. To obtain α we calculate the angle, relative to the x-axis, of the eigenvector with the largest spread. The angle α is defined as

$$\alpha = \arctan\left(\frac{v_{11}}{v_{12}}\right), \tag{2.16}$$

where $\lambda_1 \geq \lambda_2$.

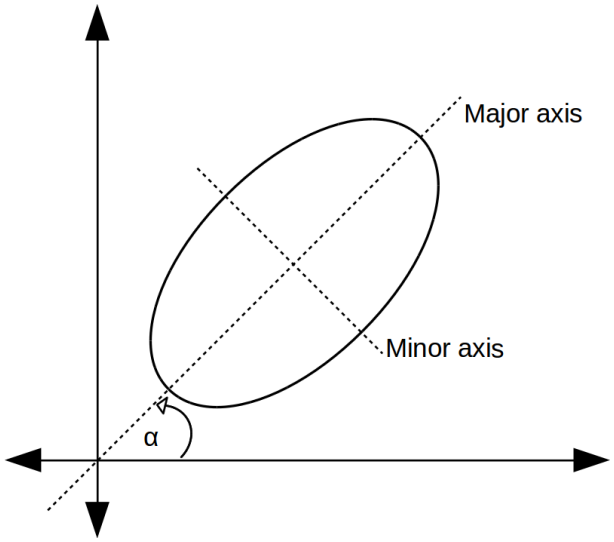

Figure 2.2: An error ellipse is shown. The major and minor axes are indicated with the dotted lines and the orientation of the ellipse is indicated with α.

2.2 Canonical Form

The canonical or information form is an alternative way to describe Gaussian distributions. Using the canonical form has benefits, for example that certain operations are easier to perform. It can also be used to represent Gaussian conditional probability densities (CPDs), which will be discussed later in this section. This section is based on the work of Koller and Friedman [2] and Schoeman [5].

The definition of the Gaussian distribution, shown in Equation 2.8, can be rewritten in the following way

$$\begin{aligned} p(\boldsymbol{x}) &= \eta \exp\left[-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right] \\ &= \exp\left(-\frac{1}{2}\boldsymbol{x}^T\Sigma^{-1}\boldsymbol{x} + \boldsymbol{\mu}^T\Sigma^{-1}\boldsymbol{x} - \frac{1}{2}\boldsymbol{\mu}^T\Sigma^{-1}\boldsymbol{\mu} + \ln\eta\right). \end{aligned} \tag{2.17}$$

We define the information matrix as

$$K = \Sigma^{-1}, \tag{2.18}$$

the potential vector as

$$\boldsymbol{h} = \Sigma^{-1}\boldsymbol{\mu}, \tag{2.19}$$

and the scalar as

$$g = -\frac{1}{2}\boldsymbol{\mu}^T\Sigma^{-1}\boldsymbol{\mu} + \ln\eta. \tag{2.20}$$

We can substitute the definitions from Equation 2.18 to Equation 2.20 into Equation 2.17, and the define the canonical form as

$$\mathcal{C}_g(\boldsymbol{x}; K, \boldsymbol{h}, g) = \exp\left(-\frac{1}{2}\boldsymbol{x}^T K\boldsymbol{x} + \boldsymbol{h}^T\boldsymbol{x} + g\right). \tag{2.21}$$

Inspecting Equation 2.18 to Equation 2.20 again, it can be seen that one can always determine g, if the values of $\boldsymbol{h}$ and K is available. This enables us to describe the canonical form in a simpler form, therefore

$$\mathcal{C}(\boldsymbol{x}; K, \boldsymbol{h}) \propto \mathcal{C}_g(\boldsymbol{x}; K, \boldsymbol{h}, g). \tag{2.22}$$

One can easily recover the covariance parameters from the canonical parameters, where

$$\Sigma = K^{-1} \tag{2.23}$$

and

$$\boldsymbol{\mu} = \Sigma \boldsymbol{h}. \tag{2.24}$$

If K is not invertible and the covariance can not be recovered, the canonical form is therefore more general than the covariance form.

It can be seen from the above that it is very easy to convert between the canonical and covariance forms.

2.2.1 Operations using the Canonical Form

This section will cover useful operations using the canonical forms.

Extending and rearranging the scopes of canonical forms may be necessary before doing operations, because it is important that the scopes of canonical forms are identical when doing operations.

Extending the Scope of a Canonical Form:

The scope of a canonical form can be extended by adding zero entries to the matrix K and vector $\boldsymbol{h}$. The scope of a canonical form of a M sized random vector $\boldsymbol{x}$, can easily be extended to the size of $M + N$, by adding zeros in the following way:

$$\mathcal{C}\left(\boldsymbol{x}; K, \boldsymbol{h}\right) = \mathcal{C}\left(\begin{bmatrix}\boldsymbol{x}\\ \boldsymbol{y}\end{bmatrix}; \begin{bmatrix} K & [0]_{M\times N} \\ [0]_{N\times M} & [0]_{N\times N}\end{bmatrix}, \begin{bmatrix}\boldsymbol{h}\\ \boldsymbol{0}\end{bmatrix}\right). \tag{2.25}$$

Rearranging the Scope of a Canonical Form:

The scope of a canonical form can be rearranged by changing the order of the rows and columns of the matrix K and repositioning the entries of the vector $\boldsymbol{h}$:

$$\mathcal{C}\left(\begin{bmatrix}\boldsymbol{x}\\ \boldsymbol{y}\\ \boldsymbol{z}\end{bmatrix}; \begin{bmatrix} K_{\boldsymbol{xx}} & K_{\boldsymbol{xy}} & K_{\boldsymbol{xz}} \\ K_{\boldsymbol{yx}} & K_{\boldsymbol{yy}} & K_{\boldsymbol{yz}} \\ K_{\boldsymbol{zx}} & K_{\boldsymbol{zy}} & K_{\boldsymbol{zz}}\end{bmatrix}, \begin{bmatrix}\boldsymbol{h}_{\boldsymbol{x}}\\ \boldsymbol{h}_{\boldsymbol{y}}\\ \boldsymbol{h}_{\boldsymbol{z}}\end{bmatrix}\right) = \mathcal{C}\left(\begin{bmatrix}\boldsymbol{y}\\ \boldsymbol{x}\\ \boldsymbol{z}\end{bmatrix}; \begin{bmatrix} K_{\boldsymbol{yy}} & K_{\boldsymbol{yx}} & K_{\boldsymbol{yz}} \\ K_{\boldsymbol{xy}} & K_{\boldsymbol{xx}} & K_{\boldsymbol{xz}} \\ K_{\boldsymbol{zy}} & K_{\boldsymbol{zx}} & K_{\boldsymbol{zz}}\end{bmatrix}, \begin{bmatrix}\boldsymbol{h}_{\boldsymbol{y}}\\ \boldsymbol{h}_{\boldsymbol{x}}\\ \boldsymbol{h}_{\boldsymbol{z}}\end{bmatrix}\right). \tag{2.26}$$

Multiplication of Canonical Forms:

If the scopes of two canonical forms are identical, one can multiply them by simply adding the K and $\boldsymbol{h}$ parameters of the two canonical forms:

$$\mathcal{C}(\boldsymbol{x}; K_{\boldsymbol{x}}, \boldsymbol{h}_{\boldsymbol{x}}) \times \mathcal{C}(\boldsymbol{y}; K_{\boldsymbol{y}}, \boldsymbol{h}_{\boldsymbol{y}}) = \mathcal{C}(\boldsymbol{z}; K_{\boldsymbol{x}} + K_{\boldsymbol{y}}, \boldsymbol{h}_{\boldsymbol{x}} + \boldsymbol{h}_{\boldsymbol{y}}). \tag{2.27}$$

Marginalisation of a Canonical Form:

A marginal distribution can be found by integrating over a subset of variables, for example the marginal distribution over $\boldsymbol{x}$ can be found by integrating over $\boldsymbol{y}$. Let $\mathcal{C}(\boldsymbol{x}, \boldsymbol{y}; K, \boldsymbol{h})$ be a canonical form with subsets $\boldsymbol{x}$ and $\boldsymbol{y}$ where

$$K = \begin{bmatrix} K_{\boldsymbol{xx}} & K_{\boldsymbol{xy}} \\ K_{\boldsymbol{yx}} & K_{\boldsymbol{yy}} \end{bmatrix} \tag{2.28}$$

and

$$h = \begin{bmatrix} h_{\boldsymbol{x}} \\ h_{\boldsymbol{y}} \end{bmatrix}. \tag{2.29}$$

To obtain the marginal distribution over $\boldsymbol{x}$, we have to find the integral over $\boldsymbol{y}$. Therefore the marginal distribution over $\boldsymbol{x}$ is

$$\int \mathcal{C}(\boldsymbol{x}, \boldsymbol{y}; K, \boldsymbol{h}) d\boldsymbol{y} = \mathcal{C}(\boldsymbol{x}; K', \boldsymbol{h}'), \tag{2.30}$$

where

$$K' = K_{\boldsymbol{xx}} - K_{\boldsymbol{xy}} K_{\boldsymbol{yy}}^{-1} K_{\boldsymbol{yx}} \tag{2.31}$$

and

$$\boldsymbol{h'} = \boldsymbol{h}_{\boldsymbol{x}} - K_{\boldsymbol{xx}} K_{\boldsymbol{yy}}^{-1} \boldsymbol{h}_{\boldsymbol{y}}. \tag{2.32}$$

It is important that $K_{\boldsymbol{yy}}$ is positive definite for the result to be finite.

Reduction with Evidence:

Let $\mathcal{C}(\boldsymbol{x}, \boldsymbol{y}; K, \boldsymbol{h})$ be a canonical form with subsets $\boldsymbol{x}$ and $\boldsymbol{y}$. If the value of subset $\boldsymbol{y}$ is known to be $\boldsymbol{Y}$, then the canonical form can be reduced to $\mathcal{C}(\boldsymbol{x}; K', \boldsymbol{h}')$, where

$$K' = K_{\boldsymbol{xx}}, \tag{2.33}$$

$$\boldsymbol{h'} = \boldsymbol{h}_{\boldsymbol{x}} - K_{\boldsymbol{xy}} \boldsymbol{Y}, \tag{2.34}$$

The operations discussed above are very simple to perform and will be used in the following chapters.

2.2.2 Conditional Probability Distributions in the Canonical Form

One of the advantages of the canonical form is that it is possible to represent conditional probability distributions (CPDs).

The CPD $p(\boldsymbol{y}|\boldsymbol{x})$ is a distribution over the vector $\boldsymbol{y}$, given we know the value of the vector $\boldsymbol{x}$, and is defined as

$$p(\boldsymbol{y}|\boldsymbol{x}) = \frac{p(\boldsymbol{y}, \boldsymbol{x})}{p(\boldsymbol{x})}. \tag{2.35}$$

If the arguments of the CPD $p(\boldsymbol{y}|\boldsymbol{x})$ can be described by a linear function, therefore

$$\boldsymbol{y} = F\boldsymbol{x} + \boldsymbol{g} + \boldsymbol{n}, \tag{2.36}$$

where the noise $\boldsymbol{n}$ is described as

$$\boldsymbol{n} \sim \mathcal{N}(\boldsymbol{n}; \boldsymbol{0}, \Sigma_n), \tag{2.37}$$

then the PDF of $p(\boldsymbol{y}|\boldsymbol{x})$ is defined as

$$\begin{aligned} p(\boldsymbol{y}|\boldsymbol{x}) &= \mathcal{N}(\boldsymbol{y}; F\boldsymbol{x}+\boldsymbol{g}, \Sigma) \\ &= \eta \exp\left[-\frac{1}{2}(\boldsymbol{y}-(F\boldsymbol{x}+\boldsymbol{g}))^T\Sigma^{-1}(\boldsymbol{y}-(F\boldsymbol{x}+\boldsymbol{g}))\right]. \end{aligned} \tag{2.38}$$

A part of Equation 2.38 can be rewritten as

$$\begin{aligned} \boldsymbol{y}-(F\boldsymbol{x}+\boldsymbol{g}) &= \begin{bmatrix} I & -F & -I \end{bmatrix} \begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{x} \\ \boldsymbol{g} \end{bmatrix} \\ &= \begin{bmatrix} I \\ -F^T \\ -I \end{bmatrix}^T \begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{x} \\ \boldsymbol{g} \end{bmatrix}. \end{aligned} \tag{2.39}$$

Substituting Equation 2.39 into Equation 2.38 gives

$$p(\boldsymbol{y}|\boldsymbol{x}) = \eta \exp\left[-\frac{1}{2}\begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{x} \\ \boldsymbol{g} \end{bmatrix}^T \begin{bmatrix} I \\ -F^T \\ -I \end{bmatrix} \Sigma^{-1} \begin{bmatrix} I \\ -F^T \\ -I \end{bmatrix}^T \begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{x} \\ \boldsymbol{g} \end{bmatrix}\right]. \tag{2.40}$$

We now define a new combined random vector

$$\boldsymbol{w} = \begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{x} \\ \boldsymbol{g} \end{bmatrix} \tag{2.41}$$

and an information matrix

$$K' = \begin{bmatrix} I \\ -F^T \\ -I \end{bmatrix} \Sigma^{-1} \begin{bmatrix} I \\ -F^T \\ -I \end{bmatrix}^T. \tag{2.42}$$

We can substitute Equation 2.41 and Equation 2.42 into Equation 2.40 which gives

$$p(\boldsymbol{y}|\boldsymbol{x}) = \eta \exp\left[-\frac{1}{2}\boldsymbol{w}^T K' \boldsymbol{w}\right]. \tag{2.43}$$

We can now use Equation 2.21 to write the conditional distribution of Equation 2.43 in the canonical form as follows

$$p(\boldsymbol{y}|\boldsymbol{x}) \propto \mathcal{C}(\boldsymbol{w}; K', \boldsymbol{0}) = \mathcal{C}\left(\begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{x} \\ \boldsymbol{g} \end{bmatrix}; \begin{bmatrix} I \\ -F^T \\ -I \end{bmatrix} \Sigma^{-1} \begin{bmatrix} I \\ -F^T \\ -I \end{bmatrix}^T, \boldsymbol{0}\right) \tag{2.44}$$

The parameter $\boldsymbol{g}$ is known and can be used as evidence to reduce the canonical form in Equation 2.44, therefore

$$\begin{aligned} \mathcal{C}(\boldsymbol{w}; K', \boldsymbol{0}) &= \mathcal{C}\left(\begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{x} \end{bmatrix}; \begin{bmatrix} I \\ -F^T \end{bmatrix} \Sigma^{-1} \begin{bmatrix} I \\ -F^T \end{bmatrix}^T, \begin{bmatrix} I \\ -F^T \end{bmatrix} \Sigma^{-1}\boldsymbol{g}\right) \\ &= \mathcal{C}\left(\begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{x} \end{bmatrix}; \begin{bmatrix} \Sigma^{-1} & -\Sigma^{-1}F \\ -F^T\Sigma^{-1} & F^T\Sigma^{-1}F \end{bmatrix}, \begin{bmatrix} -\Sigma^{-1}\boldsymbol{g} \\ -F^T\Sigma^{-1}\boldsymbol{g} \end{bmatrix}\right). \end{aligned} \tag{2.45}$$

The canonical form can therefore also be used to represent a Gaussian conditional probability distribution (CPD) with arguments which have a linear relationship. The canonical form is used in later chapters to do operations on various Gaussian distributions.

Chapter 3

Motion and Measurement Models

Before locating a robot, it is important to model the way the robot moves. A simple linear motion model will be described, which will later be used to illustrate basic principles when estimating the location of a linear moving robot. In practice the movement of a robot is seldom linear, therefore a more complex nonlinear motion model will also have to be explored. The motion model describes $p(\boldsymbol{x}_t|\boldsymbol{u}_t, \boldsymbol{x}_{t-1})$, which is the state transition probability density function (PDF). This is important and is used in the *prediction step* of the Bayes filter, discussed in Chapter 4. Robot motion will be limited to planar movement to make it easier to convey concepts or ideas to the reader. Note that the theory discussed is not limited to planar movement and can easily be extended to three dimensions. This chapter is based on the work of Thrun, Burgard and Fox [6].

3.1 Linear Motion Model

In this section a simple linear motion model is investigated which describes the linear movement of a robot. The robot's position describes the kinematic state of a robot and is defined as

$$\boldsymbol{x}_t = \begin{bmatrix} x_t \\ y_t \end{bmatrix}. \tag{3.1}$$

The movement of the robot is independent of its orientation and can be controlled with two translational velocities in a control vector $\boldsymbol{u}_t$, defined as

$$\boldsymbol{u_t} = \begin{bmatrix} v_{x,t} \\ v_{y,t} \end{bmatrix}, \tag{3.2}$$

where $v_{x,t}$ is the velocity in the x-direction and $v_{y,t}$ is the velocity in the y-direction at time t. The position $\boldsymbol{x_t}$ of the robot at time t can therefore be described by a linear state transition function which is defined as

$$\begin{bmatrix} x_t \\ y_t \end{bmatrix} = \begin{bmatrix} x_{t-1} \\ y_{t-1} \end{bmatrix} + \begin{bmatrix} v_{x,t}\Delta t \\ v_{y,t}\Delta t \end{bmatrix} + \begin{bmatrix} \varepsilon_x \\ \varepsilon_y \end{bmatrix} \tag{3.3}$$

Equation 3.3 can be described as a linear function in terms of $\boldsymbol{x}_t$, $\boldsymbol{x}_{t-1}$ and $\boldsymbol{u}_t$, therefore

$$\boldsymbol{x_t} = A_t\boldsymbol{x_{t-1}} + B_t\boldsymbol{u_t} + \boldsymbol{\varepsilon}_t, \tag{3.4}$$

where

$$A_t = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \tag{3.5}$$

$$B_t = \begin{bmatrix} \Delta t & 0 \\ 0 & \Delta t \end{bmatrix}, \tag{3.6}$$

and the noise is described with a Gaussian distribution

$$\boldsymbol{\varepsilon}_t \sim \mathcal{N}(\boldsymbol{\varepsilon}_t; 0, R). \tag{3.7}$$

The covariance matrix R is defined as

$$R = \begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{bmatrix}. \tag{3.8}$$

The variances σ_x^2 and σ_y^2 are determined empirically and Δt is the duration of a time interval. This concludes the linear motion model. The next section explores a more complex nonlinear motion model.

3.2 Nonlinear Motion Model

The kinematic state of a robot moving in a plane is redefined for the nonlinear motion model. For the nonlinear motion model it is described by three variables which is referred to as the pose of the robot and is defined as

$$\boldsymbol{x_t} = \begin{bmatrix} x_t \\ y_t \\ \theta_t \end{bmatrix}. \tag{3.9}$$

The pose of the robot is illustrated in Figure 3.1, where x_t is the x-coordinate, y_t is the y-coordinate and θ_t is the orientation of the robot at time t.

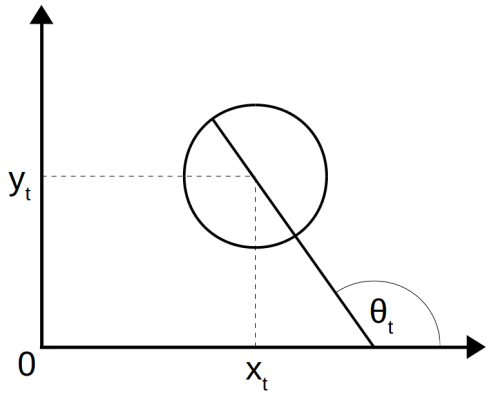

Figure 3.1: The pose of a robot is illustrated where the position of the robot at time t is indicated with x_t and y_t. The orientation of the robot is indicated with θ_t.

The general state transition function of a nonlinear motion model can in be described as

$$\boldsymbol{x}_t = \boldsymbol{g}(\boldsymbol{u}_t, \boldsymbol{x}_{t-1}) + \boldsymbol{\varepsilon}_t \tag{3.10}$$

Two nonlinear motion models were considered for this project, namely the velocity and the odometry model. The odometry motion model makes use of sensor measurements to describe a robot's movement over time. Odometry models can only be used in retrospect after a robot has moved and cannot be used for motion planning. The velocity model is

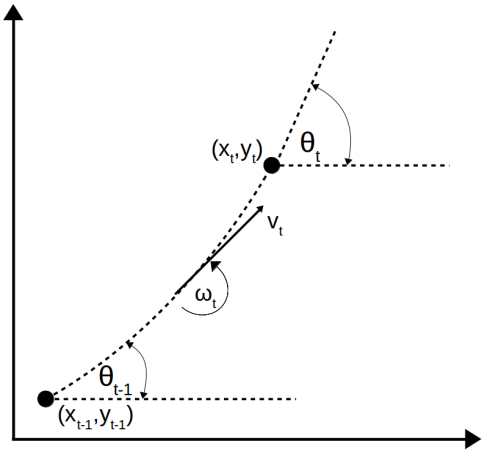

Figure 3.2: The pose of the robot is indicated at time $t-1$ and at time t. The direction of the rotational ω_t and translational v_t velocities are shown.

suitable for motion planning and assumes that a robot can be controlled with a rotational and translational velocity. The velocity motion model therefore predicts how the robot will move and is not as accurate as the odometry motion model. Both the odometry and velocity motion models are common and are widely used. As the velocity motion model meets the demands for the goal this project, it was chosen as the motion model.

The velocity motion model assumes that the movement of a robot can be described by a translational velocity v_t and a rotational velocity ω_t, shown in Figure 3.2. The control vector u_t is thus described as

$$\boldsymbol{u_t} = \begin{bmatrix} v_t \\ \omega_t \end{bmatrix}. \tag{3.11}$$

Positive rotational velocities are defined to be counterclockwise and positive translational velocities are specified to be in the "forward" direction. The state transition function of the velocity motion model is described as

$$\begin{bmatrix} x_t \\ y_t \\ \theta_t \end{bmatrix} = \begin{bmatrix} x_{t-1} - \frac{v_t}{\omega_t}\sin\theta_{t-1} + \frac{v_t}{\omega_t}\sin(\theta_{t-1}+\omega_t\Delta t) \\ y_{t-1} + \frac{v_t}{\omega_t}\cos\theta_{t-1} - \frac{v_t}{\omega_t}\cos(\theta_{t-1}+\omega_t\Delta t) \\ \theta_{t-1}+\omega_t\Delta t \end{bmatrix} + \boldsymbol{\varepsilon}_t \tag{3.12}$$

where

$$\boldsymbol{\varepsilon}_t \sim \mathcal{N}(\boldsymbol{\varepsilon}_t; 0, R). \tag{3.13}$$

The covariance matrix R is defined as

$$R = \begin{bmatrix} \sigma_x^2 & 0 & 0 \\ 0 & \sigma_y^2 & 0 \\ 0 & 0 & \sigma_\theta^2 \end{bmatrix}, \tag{3.14}$$

where the variances ${\sigma_x}^2$, ${\sigma_y}^2$ and ${\sigma_\theta}^2$ are determined empirically.

The motion models described in this chapter will be used in subsequent chapters to describe the movement of a robot.

Chapter 4

Localisation using Kalman Filters

This chapter will focus on estimating the location of a robot, using traditional estimators namely, the Kalman filter, the extended Kalman filter (EKF) and the unscented Kalman filter (UKF). The general theory of the EKF and UKF are discussed in terms of both nonlinear movement and measurements, but when the estimators are applied in Python code, only nonlinear movement is considered while measurements are assumed to be linear. One should pay close attention to the Bayes filter as discussed in Section 4.2, because all subsequent estimators discussed, are variations of the Bayes filter. The operations of the Bayes filter are also similar to a PGM which is modelled in Chapter 6. This chapter is based on the work of Thrun [6].

4.1 Background

A concept that is often used is in this report is that of *belief*. As mentioned in the introduction, a robot cannot know its exact pose. A robot can infer a belief or can have an internal knowledge of its state from information obtained from the sensors. There is thus a difference between internal belief and true state.

In this report the belief, also known as the posterior, is the conditional probability distribution (CPD) of a state variable $\boldsymbol{x}_t$ given that all the previous measurements $\boldsymbol{z}_{1:t}$ and controls $\boldsymbol{u}_{1:t}$ are known. The belief of a state variable is denoted as

$$\boldsymbol{x}_t \sim bel(\boldsymbol{x}_t), \tag{4.1}$$

which is a reduced notation for the posterior

$$bel(\boldsymbol{x}_t) = p(\boldsymbol{x}_t|\boldsymbol{z}_{1:t}, \boldsymbol{u}_{1:t}). \tag{4.2}$$

The belief $bel(\boldsymbol{x}_t)$ is usually calculated from a *predicted belief* $\overline{bel}(\boldsymbol{x}_t)$. This calculation step is known as the *measurement update*.

The *predicted belief* is defined as

$$\overline{bel}(\boldsymbol{x}_t) = p(\boldsymbol{x}_t|\boldsymbol{z}_{1:t-1}, \boldsymbol{u}_{1:t}), \tag{4.3}$$

and is the CPD of the state variable $\boldsymbol{x}_t$ without taking the measurement $\boldsymbol{z}_t$ into consideration.

4.2 Bayes Filter

The *Bayes Filter* algorithm is the generic cornerstone for determining beliefs and is covered to understand the principle on which subsequent estimators work. Algorithm 1 shows the general process of the Bayes filter which is recursive as the belief $bel(\boldsymbol{x}_t)$ at the time t is determined from the previous belief $bel(\boldsymbol{x}_{t-1})$ at the time $t-1$. The algorithm receives and uses the previous belief $bel(\boldsymbol{x}_{t-1})$, the latest control input $\boldsymbol{u}_t$ and the measurements $\boldsymbol{z}_t$ to calculate the belief $bel(\boldsymbol{x}_t)$. Algorithm 1 shows one iteration of the process which consists of two steps, namely the *prediction* step in line 3, and *measurement update* step in line 4.

The prediction step calculates a predicted belief of the state $\boldsymbol{x}_t$ based on the belief of the previous state $\boldsymbol{x}_{t-1}$ and the current control input $\boldsymbol{u}_t$.

In the *measurement update* step, the final belief of state $\boldsymbol{x}_t$ is calculated by multiplying the predicted belief $\overline{bel}(\boldsymbol{x}_t)$ by the probability that $\boldsymbol{z}_t$ is observed. The result is usually not a valid PDF and has to be normalized by multiplying it with a normalisation constant, η. Lastly in line 6, the final belief $bel(\boldsymbol{x}_t)$ at time t is returned.

Algorithm 1 Bayes Filter

1: **procedure** BAYES_FILTER($bel(\boldsymbol{x}_{t-1}), \boldsymbol{u}_t, \boldsymbol{z}_t$)
2: **for all** $\boldsymbol{x}_t$ **do**
3: $\overline{bel}(\boldsymbol{x}_t) = \int p(\boldsymbol{x}_t|\boldsymbol{u}_t, \boldsymbol{x}_{t-1}) bel(\boldsymbol{x}_{t-1})\, d\boldsymbol{x}_{t-1}$
4: $bel(\boldsymbol{x}_t) = \eta p(\boldsymbol{z}_t|\boldsymbol{x}_t)\overline{bel}(\boldsymbol{x}_t)$
5: **end for**
6: **return** $bel(\boldsymbol{x}_t)$
7: **end procedure**

4.3 Kalman Filter

The Kalman filter is an variation of the Bayes filter and is used for prediction in linear Gaussian systems. The Kalman filter is used for the belief estimation of systems with continuous states and is not applicable in discrete state spaces.

4.3.1 Description of the Kalman Filter

The Kalman filter makes three assumptions in addition to the Markov assumptions. The outcome of this is that each belief $bel(\boldsymbol{x}_t)$, which is calculated with the Kalman filter, is a Gaussian distribution.

The three assumptions are as follows:

1. Linear system dynamics are assumed. The motion model describing the movement must be linear, with Gaussian noise $\boldsymbol{\varepsilon}$ added, and it must be possible to express it in the form of

$$\boldsymbol{x}_t = A_t\boldsymbol{x}_{t-1} + B_t\boldsymbol{u}_t + \boldsymbol{\varepsilon}_t, \tag{4.4}$$

where the noise $\boldsymbol{\varepsilon}_t$ is described by

$$\boldsymbol{\varepsilon}_t \sim \mathcal{N}(\boldsymbol{\varepsilon}_t; \mathbf{0}, R). \tag{4.5}$$

The arguments of the state transition CPD, $p(\boldsymbol{x}_t|\boldsymbol{u}_t, \boldsymbol{x}_{t-1})$, can therefore be described by a linear function.

2. Linear measurements with Gaussian noise $\boldsymbol{\zeta}_t$ are assumed. The measurement PDF $p(\boldsymbol{z}_t|\boldsymbol{x}_t)$ is thus also linear in its arguments and can be described by

$$\boldsymbol{z}_t = C_t\boldsymbol{x}_t + \boldsymbol{\zeta}_t, \tag{4.6}$$

where the noise $\boldsymbol{\zeta}_t$ is described by

$$\boldsymbol{\zeta}_t \sim \mathcal{N}(\boldsymbol{\zeta}_t; \mathbf{0}, Q). \tag{4.7}$$

3. Lastly it is assumed that the initial belief $bel(\boldsymbol{x}_0)$ of the system is a Gaussian distribution.

The process for the Kalman filter is shown in Algorithm 2. It is a variation of the Bayes filter and therefore the same principle is followed as in Algorithm 1. The algorithm receives the parameters, $\boldsymbol{\mu}_{t-1}$ and Σ_{t-1}, of the previous belief $bel(\boldsymbol{x}_t)$, along with the control vector $\boldsymbol{u}_t$ and measurement vector $\boldsymbol{z}_t$. In line 2 the previous mean $\boldsymbol{\mu}_t$ is transformed through the linear state transition function to determine the predicted mean $\overline{\boldsymbol{\mu}}_t$ of the predicted belief. In line 3 the covariance Σ_{t-1} is multiplied twice with A_t, then R_t is added to determine the predicted covariance $\overline{\Sigma}_t$ of the predicted belief. Lines 2 and 3 therefore contain the *prediction* step of the Bayes filter where the predicted belief $\overline{bel}(x_t)$ is calculated.

Lines 4 to 6 correspond with the *measurement update* step of the Bayes filter in which the mean $\boldsymbol{u}_t$ and the covariance Σ_t of the the desired belief $bel(\boldsymbol{x}_t)$ is computed. The Kalman gain K_t is calculated in line 4 and is used to determine how much influence the measurement $\boldsymbol{z}_t$ has when it is incorporated. The difference between the actual measurement $\boldsymbol{z}_t$ and the expected measurement $C_t\overline{\boldsymbol{\mu}}_t$ is called the innovation. In line 5 the predicted mean $\overline{\boldsymbol{\mu}}_t$ is adjusted by adding it to the product of the Kalman gain and the innovation. In line 6 the covariance Σ_t of the posterior belief $bel(\boldsymbol{x}_t)$ is calculated by adjusting the predicted covariance $\overline{\Sigma}_t$.

Algorithm 2 Kalman Filter

1: **procedure** KALMAN_FILTER($\boldsymbol{\mu}_{t-1}, \Sigma_{t-1}, \boldsymbol{u}_t, \boldsymbol{z}_t$)
2: $\quad\overline{\boldsymbol{\mu}}_t = A_t\boldsymbol{\mu}_{t-1} + B_t\boldsymbol{u}_t$
3: $\quad\overline{\Sigma}_t = A_t\Sigma_{t-1}A_t^T + R_t$
4: $\quad K_t = \overline{\Sigma}_t C_t^T(C_t\overline{\Sigma}_t C_t^T + Q_t)^{-1}$
5: $\quad\boldsymbol{\mu}_t = \overline{\boldsymbol{\mu}}_t + K_t(\boldsymbol{z}_t - C_t\overline{\boldsymbol{\mu}}_t)$
6: $\quad\Sigma_t = (I - K_tC_t)\overline{\Sigma}_t$
7: $\quad$**return** $\boldsymbol{\mu}_t, \Sigma_t$
8: **end procedure**

4.3.2 Application of the Kalman Filter

The Kalman filter is applied to determine the belief of the location of a linear moving robot. The movement of the robot is therefore described by the motion model in Equation 3.4. The moving robot is simulated and the Kalman filter (Algorithm 2) is implemented in Python to find the belief $bel(\boldsymbol{x}_t)$ at each time step t.

The initial belief $bel(\boldsymbol{x}_0)$ of the robot's location is known and five control commands in the form of a list is given to the robot. At each time step the controls and measurements

are given to the Kalman filter, which produces the belief of the robot's state at time t. The result of the simulation is shown in Figure 4.1, where the purple error ellipses indicate the 95% confidence level region of each belief $bel(\boldsymbol{x}_t)$. The purple dots indicate the mean of each belief $bel(\boldsymbol{x}_t)$ at time t. The exact position of the robot at each time step is indicated with a black cross.

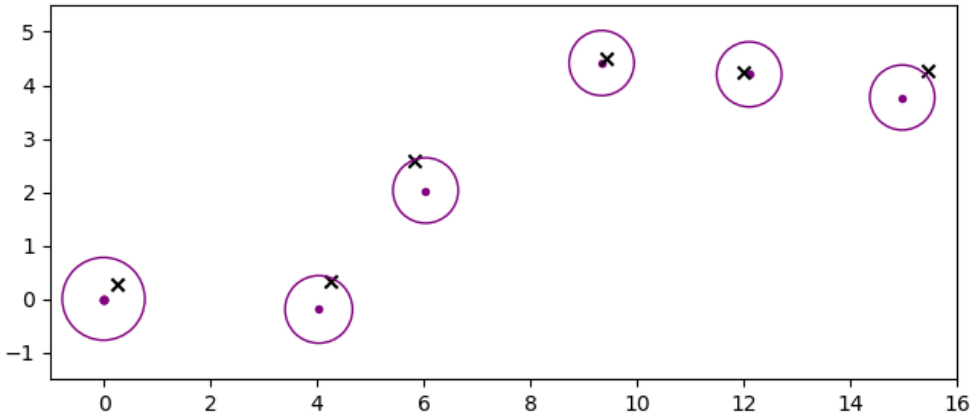

Figure 4.1: The figure above shows the result where the Kalman filter is used to estimate the position of a linear moving robot at five time steps. The exact position of the robot at each time step is indicated with a black cross. The estimated belief at each time step is illustrated with a 95% confidence purple error ellipse and the mean is indicated with a purple dot. The left most error ellipse represents the initial belief.

4.4 Extended Kalman Filter

The Kalman filter as discussed above is very efficient due to the fact that state transitions and measurements are assumed to be linear, hence the mean $\boldsymbol{\mu}_t$ and covariance Σ_t of a resulting belief $bel(\boldsymbol{x}_t)$ can be computed analytically. As mentioned in Chapter 3, state transitions and measurements are in practice seldom linear and therefore the normal Kalman filter is not suitable for most robotics problems.

4.4.1 Description of EKF

The extended Kalman filter (EKF) lifts the constraint of linear state transition and measurement functions. These functions are now described in a more general form. The state transition of a nonlinear system can be described by

$$\boldsymbol{x}_t = \boldsymbol{g}(\boldsymbol{u}_t, \boldsymbol{x}_{t-1}) + \boldsymbol{\varepsilon}_t, \tag{4.8}$$

and the nonlinear measurements can be described by

$$\boldsymbol{z}_t = \boldsymbol{h}(\boldsymbol{x}_t) + \boldsymbol{\zeta}_t, \tag{4.9}$$

where $\boldsymbol{h}(.)$ and $\boldsymbol{g}(.)$ are nonlinear vector functions. If one performs a nonlinear transformation on a Gaussian random variable (RV), it will result in a non-Gaussian RV. Therefore, if the state transition or measurement functions of a system are nonlinear, the belief $bel(\boldsymbol{x}_t)$ will not be a Gaussian distribution and cannot be computed analytically.

The extended Kalman filter (EKF) is a variation of the Kalman filter and enables one to approximate the belief $bel(\boldsymbol{x}_t)$ of a nonlinear system as a Gaussian distribution. The EKF accomplishes this by using Taylor expansion to linearise the functions $\boldsymbol{h}(.)$ and $\boldsymbol{g}(.)$.

In general, with Taylor expansion a nonlinear vector function

$$\boldsymbol{f}(\boldsymbol{x}) = \begin{bmatrix} f_1(\boldsymbol{x}) & ... & f_M(\boldsymbol{x}) \end{bmatrix}^T, \tag{4.10}$$

where the vector $\boldsymbol{x}$ is defined as

$$\boldsymbol{x} = \begin{bmatrix} x_1 & ... & x_N \end{bmatrix}^T, \tag{4.11}$$

can be approximated as a linear vector function that is tangent to a given point $\boldsymbol{k}$

$$\boldsymbol{f}(\boldsymbol{x}) \approx \boldsymbol{f}(\boldsymbol{k}) + \boldsymbol{f}'(\boldsymbol{k})(\boldsymbol{x} - \boldsymbol{k}), \tag{4.12}$$

where $\boldsymbol{f}'(\boldsymbol{x})$ is the Jacobian matrix and is calculated as

$$\boldsymbol{f}'(\boldsymbol{x}) = \frac{\partial \boldsymbol{f}(\boldsymbol{x})}{\partial \boldsymbol{x}} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_N} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_M}{\partial x_1} & \cdots & \frac{\partial f_M}{\partial x_N} \end{bmatrix}. \tag{4.13}$$

Taylor expansion can now be applied to linearise the nonlinear vector functions $\boldsymbol{g}$ and $\boldsymbol{h}$. The function $\boldsymbol{g}$ can be approximated as a linear vector function around $\boldsymbol{\mu}_{t-1}$ by means of Equation 4.12

$$\begin{aligned} \boldsymbol{g}(\boldsymbol{u_t}, \boldsymbol{x}_{t-1}) &\approx \boldsymbol{g}(\boldsymbol{u}_t, \boldsymbol{\mu}_{t-1}) + \boldsymbol{g}'(\boldsymbol{u}_t, \boldsymbol{\mu}_{t-1})(\boldsymbol{x}_{t-1} - \boldsymbol{\mu}_{t-1}) \\ &= \boldsymbol{g}(\boldsymbol{u}_t, \boldsymbol{\mu}_{t-1}) + G_t(\boldsymbol{x}_{t-1} - \boldsymbol{\mu}_{t-1}), \end{aligned} \tag{4.14}$$

where G_t is the Jacobian matrix and can be determined with Equation 4.13, therefore

$$G_t = \boldsymbol{g}'(\boldsymbol{u}_t, \boldsymbol{\mu}_{t-1}) = \frac{\partial \boldsymbol{g}(\boldsymbol{u}_t, \boldsymbol{x}_{t-1})}{\partial \boldsymbol{x}_{t-1}}. \tag{4.15}$$

The same procedure can be followed to approximate the measurement function $\boldsymbol{h}$ as a linear vector function around the predicted mean $\overline{\boldsymbol{\mu}}$

$$\begin{aligned} \boldsymbol{h}(\boldsymbol{x}_t) &\approx \boldsymbol{h}(\overline{\boldsymbol{\mu}}_t) + \boldsymbol{h}'(\overline{\boldsymbol{u}}_t)(\boldsymbol{x}_t - \overline{\boldsymbol{\mu}}_t) \\ &= \boldsymbol{h}(\overline{\boldsymbol{\mu}}_t) + H_t(\boldsymbol{x}_t - \overline{\boldsymbol{\mu}}_t), \end{aligned} \tag{4.16}$$

where

$$H_t = \boldsymbol{h}'(\boldsymbol{u}_t) = \frac{\partial \boldsymbol{h}(\boldsymbol{x}_t)}{\partial \boldsymbol{x}_t}. \tag{4.17}$$

In Algorithm 3 the procedure of the EKF is shown and one will notice that it is very similar to Algorithm 2 of the Kalman filter. It can be seen that G_t and H_t in Algorithm 3 are respectively equivalent to A_t and C_t in Algorithm 2. It must also be remembered that the nonlinear functions are approximated as linear functions, therefore the belief computed by the EKF is only an approximated belief and not the ground truth.

This section covers the basic theory of the EKF, and the next section will discuss how the EKF can be applied to estimate the belief of a nonlinear system.

Algorithm 3 Extended Kalman Filter

1: **procedure** EKF($\boldsymbol{\mu}_{t-1}, \Sigma_{t-1}, \boldsymbol{u}_t, \boldsymbol{z}_t$)
2: $\quad \overline{\boldsymbol{\mu}}_t = \boldsymbol{g}(\boldsymbol{u}_t, \boldsymbol{\mu}_{t-1})$
3: $\quad \overline{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + R_t$
4: $\quad K_t = \overline{\Sigma}_t H_t^T (H_t \overline{\Sigma}_t H_t^T + Q_t)^{-1}$
5: $\quad \boldsymbol{\mu}_t = \overline{\boldsymbol{\mu}}_t + K_t(\boldsymbol{z}_t - h(\overline{\boldsymbol{\mu}}_t))$
6: $\quad \Sigma_t = (I - K_t H_t)\overline{\Sigma}_t$
7: $\quad$ **return** $\boldsymbol{\mu}_t, \Sigma_t$
8: **end procedure**

4.4.2 Application of EKF

In this subsection the EKF is applied to estimate the location of a robot which movement is described by the velocity motion model.

The robot's pose $\boldsymbol{x}_t$ at time t is dependent on the previous pose $\boldsymbol{x}_{t-1}$ at time $t-1$ and the control $\boldsymbol{u}_t$. The transition of the robot's pose can be described in general as

$$\boldsymbol{x}_t = \begin{bmatrix} x_t \\ y_t \\ \theta_t \end{bmatrix} = \boldsymbol{g}(\boldsymbol{u}_t, \boldsymbol{x}_{t-1}) + \boldsymbol{\varepsilon}_t, \tag{4.18}$$

where $\boldsymbol{g}$ is described by the velocity motion model as

$$\boldsymbol{g}(\boldsymbol{u}_t, \boldsymbol{x}_{t-1}) = \begin{bmatrix} x_{t-1} - \frac{v_t}{\omega_t}\sin\theta_{t-1} + \frac{v_t}{\omega_t}\sin(\theta_{t-1} + \omega_t \Delta t) \\ y_{t-1} + \frac{v_t}{\omega_t}\cos\theta_{t-1} - \frac{v_t}{\omega_t}\cos(\theta_{t-1} + \omega_t \Delta t) \\ \theta_{t-1} + \omega_t \Delta t \end{bmatrix}. \tag{4.19}$$

The noise $\boldsymbol{\varepsilon}$ is defined as

$$\boldsymbol{\varepsilon_t} \sim \mathcal{N}(\boldsymbol{\varepsilon_t}; \mathbf{0}; R_t), \tag{4.20}$$

where the covariance matrix R is defined as

$$R = \begin{bmatrix} \sigma_x^2 & 0 & 0 \\ 0 & \sigma_y^2 & 0 \\ 0 & 0 & \sigma_\theta^2 \end{bmatrix}. \tag{4.21}$$

R is a diagonal matrix, hence the noise added to each RV is independent.

The next step is to linearise the nonlinear vector function $\boldsymbol{g}$ by means of Taylor expansion. Equation 4.13 and Equation 4.15 can be used to calculate the Jacobian matrix G_t of $\boldsymbol{g}$:

$$G_t = \boldsymbol{g}'(\boldsymbol{u}_t, \boldsymbol{x}_{t-1}) = \begin{bmatrix} 1 & 0 & -\frac{v_t}{\omega_t}\cos\theta + \frac{v_t}{\omega_t}\cos(\theta + \omega_t \Delta t) \\ 0 & 1 & -\frac{v_t}{\omega_t}\sin\theta + \frac{v_t}{\omega_t}\sin(\theta + \omega_t \Delta t) \\ 0 & 0 & 1 \end{bmatrix}. \tag{4.22}$$

Using Equation 4.14 and the Jacobian matrix G_t calculated in Equation 4.22, the vector function $\boldsymbol{g}$ can be linearised around $\boldsymbol{u_{t-1}}$, thus

$$\boldsymbol{g}(\boldsymbol{u_t}, \boldsymbol{x}_{t-1}) \approx \boldsymbol{g}(\boldsymbol{u}_t, \boldsymbol{\mu}_{t-1}) + G_t(\boldsymbol{x}_{t-1} - \boldsymbol{\mu}_{t-1}). \tag{4.23}$$

Using the result in Equation 4.23, the state transition in Equation 4.18 can be approximated as

$$\boldsymbol{x}_t \approx \boldsymbol{g}(\boldsymbol{u}_t, \boldsymbol{\mu}_{t-1}) + G_t(\boldsymbol{x}_{t-1} - \boldsymbol{\mu}_{t-1}) + \boldsymbol{\varepsilon}. \tag{4.24}$$

Linear measurements are assumed for this application and therefore we can use Equation 4.7 to describe $\boldsymbol{z}_t$, where

$$C_t = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{4.25}$$

Five time steps of the movement of a robot were simulated in Python. The initial belief of the robot is known and five control inputs are given to the robot in by means of a list. Algorithm 3, tailored for the velocity motion model, is implemented and the result of the application can be seen in Figure 4.2. The green ellipses represent the 95% confidence region and the green dots represent the mean of each belief $bel(\boldsymbol{x}_t)$ at time t. The exact position of the robot of each time step t is represented by the black cross.

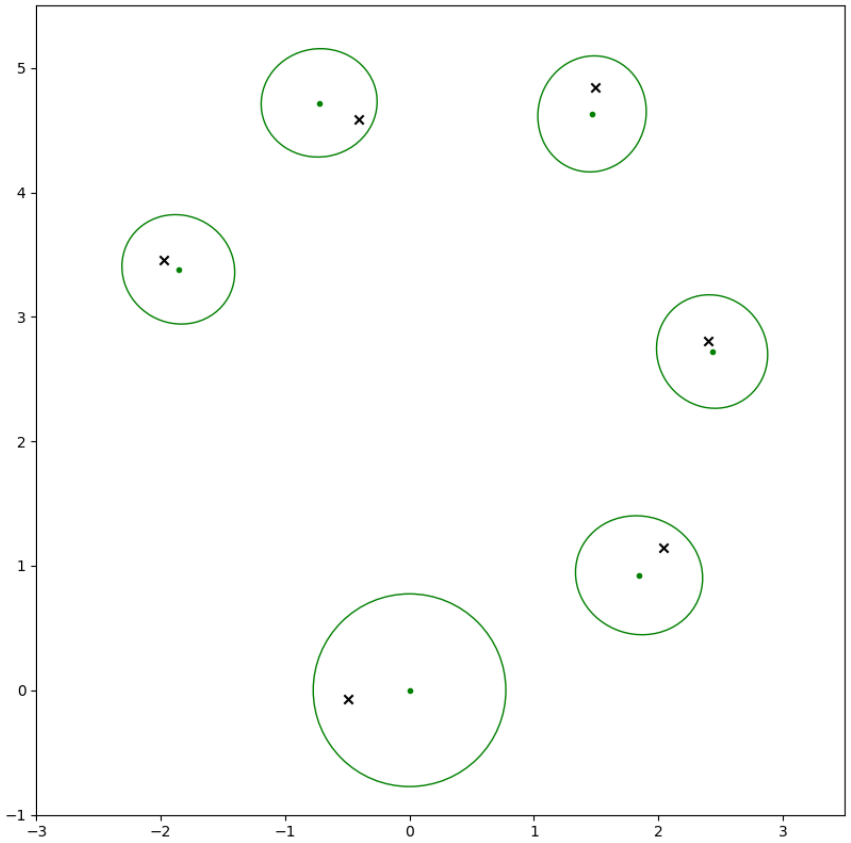

Figure 4.2: The figure shows the result where the extended Kalman filter is applied to to estimate the pose of a robot performing a nonlinear movement. The exact position of the robot at each time step is indicated with a black cross. The beliefs which are computed with the EKF are shown in green and the initial belief is indicated in black. The uncertainty of a belief is illustrated with an 95% confidence error ellipse and a dot is used to indicate the mean.

4.5 Unscented Kalman Filter

The linearisation used in the EKF can sometimes be very inaccurate, as the nonlinear functions are linearised only around the mean $\boldsymbol{u}_t$. The unscented Kalman filter (UKF) is an alternative approach which in many cases deliver better results than that of the EKF, especially when the measurement and state transition functions are severely nonlinear.

As mentioned before, a Gaussian distribution that is transformed through a nonlinear function results in a distribution that is not Gaussian. The principle on which UKF is based, is to draw carefully chosen points called *sigma points* around the mean of an existing Gaussian distribution and transform them through a nonlinear function. The transformed sigma points can then be used to approximate the resultant distribution as a Gaussian distribution.

Usually one sigma point is located at the mean and for each dimension two sigma points are located symmetrically around the mean along the main axes. For an n-dimensional Gaussian distribution, there are $2n + 1$ sigma points selected. Sigma points are denoted as

$$\mathcal{X}^{[i]},\ i\ =\ 0,\ ...,\ 2n, \tag{4.26}$$

where i indicates the index of the sigma point.

Sigma points are determined as follows:

$$\mathcal{X}^{[0]} = \boldsymbol{\mu}, \tag{4.27}$$

$$\mathcal{X}^{[i]} = \boldsymbol{\mu} + \left(\sqrt{(n+\lambda)\Sigma}\right) \text{ for } \ i = 1, ..., n, \tag{4.28}$$

$$\mathcal{X}^{[i]} = \boldsymbol{\mu} - \left(\sqrt{(n+\lambda)\Sigma}\right) \text{ for } \ i = n+1, ..., 2n, \tag{4.29}$$

where the scaling parameter λ is defined as

$$\lambda = \alpha^2(n + \kappa) - n. \tag{4.30}$$

The parameters α and κ determines the spread of the sigma points around the mean.

Each sigma point $\mathcal{X}^{[i]}$ has corresponding weights $w_m^{[i]}$ and $w_c^{[i]}$, which are respectively used to recover the mean $\boldsymbol{\mu}_t$ and covariance Σ_t. Weights corresponding to the sigma point $\mathcal{X}^{[0]}$ are determined as

$$w_m^{[0]} = \frac{\lambda}{n+\lambda}, \tag{4.31}$$

$$w_c^{[0]} = \frac{\lambda}{n+\lambda} + (1 - \alpha^2 + \beta). \tag{4.32}$$

The optimal choice for the parameter β is normally 2. The rest of the weights ($i = 1\ ,...,\ 2n$) are determined as

$$w_m^{[i]} = w_c^{[i]} = \frac{1}{2(n+\lambda)} \ \text{ for } i = 1,\ ...,\ 2n. \tag{4.33}$$

After sigma points and weights have been calculated, the chosen sigma points can be transformed through a nonlinear vector function $\boldsymbol{f}$ to determine the resulting sigma points $\mathcal{Y}^{[i]}$ given by

$$\mathcal{Y}^{[i]} = \boldsymbol{f}\left(\mathcal{X}^{[i]}\right). \tag{4.34}$$

The distribution as a result of the nonlinear transformation can be approximated as a Gaussian distribution by estimating the mean $\boldsymbol{u_y}$ and covariance $\Sigma_{\boldsymbol{yy}}$ with

$$\boldsymbol{\mu_y} = \sum_{i=0}^{2n} w_m^{[i]} \mathcal{Y}^{[i]}, \tag{4.35}$$

$$\Sigma_{\boldsymbol{yy}} = \sum_{i=0}^{2n} w_c^{[i]} \left(\mathcal{Y}^{[i]} - \boldsymbol{\mu_y}\right) \left(\mathcal{Y}^{[i]} - \boldsymbol{\mu_y}\right)^T \tag{4.36}$$

The procedure of the UKF is shown in Algorithm 4. Lines 1 to 5 relates to the *prediction* step of the Bayes filter. Sigma points are deterministically selected in line 2 and passed through the nonlinear function $\boldsymbol{g}$ in line 3. The parameters of the predicted belief $\overline{bel}(\boldsymbol{x}_t)$ are computed in lines 4 and 5. Lines 6 to 13 correspond to the *measurement update* step of the Bayes filter. In line 6, the predicted mean $\overline{\boldsymbol{\mu}}_t$ and covariance matrix $\overline{\Sigma}_t$ are used to extract new sigma points $\overline{\mathcal{X}}_t$. The predicted measurement for each sigma point is determined in line 7. The resulting sigma points $\overline{\mathcal{Z}}$ are used in lines 8 and 9 to calculate the mean $\hat{\boldsymbol{z}}_t$ and covariance S_t of the predicted measurement. In line 10 the cross covariance $\overline{\Sigma}_t^{\boldsymbol{x},\boldsymbol{z}}$ is calculated and in line 11 the Kalman gain K_t is determined. In lines 12 and 13 the updated parameters, $\boldsymbol{\mu}_t$ and Σ_t, of the belief $bel(\boldsymbol{x}_t)$ are calculated.

Algorithm 4 Unscented Kalman Filter

1: **procedure** UKF($\boldsymbol{\mu}_{t-1}, \Sigma_{t-1}, \boldsymbol{u}_t, \boldsymbol{z}_t$)
2: $\quad \mathcal{X}_{t-1} = \left(\boldsymbol{\mu}_{t-1} \quad \boldsymbol{\mu}_{t-1} + \sqrt{(n+\lambda)\Sigma_{t-1}} \quad \boldsymbol{\mu}_{t-1} - \sqrt{(n+\lambda)\Sigma_{t-1}}\right)$
3: $\quad \overline{\mathcal{X}}^*_{t-1} = \boldsymbol{g}(\boldsymbol{u}_t, \mathcal{X}_{t-1})$
4: $\quad \overline{\boldsymbol{\mu}}_t = \sum_i^{2n} w_m^{[i]} \overline{\mathcal{X}}_t^{*[i]}$
5: $\quad \overline{\Sigma}_t = \sum_i^{2n} w_c^{[i]} \left(\overline{\mathcal{X}}_t^{*[i]} - \overline{\boldsymbol{\mu}}_t\right) \left(\overline{\mathcal{X}}_t^{*[i]} - \overline{\boldsymbol{\mu}}_t\right)^T + R_t$
6: $\quad \overline{\mathcal{X}}_t = \left(\overline{\boldsymbol{\mu}}_t \quad \overline{\boldsymbol{\mu}}_t + \sqrt{(n+\lambda)\overline{\Sigma}_t} \quad \overline{\boldsymbol{\mu}}_t - \sqrt{(n+\lambda)\overline{\Sigma}_t}\right)$
7: $\quad \overline{\mathcal{Z}}_t = \boldsymbol{h}\left(\overline{\mathcal{X}}_t\right)$
8: $\quad \hat{\boldsymbol{z}}_t = \sum_{i=0}^{2n} w_m^{[i]} \overline{\mathcal{Z}}_t^{[i]}$
9: $\quad S_t = \sum_{i=0}^{2n} w_c^{[i]} \left(\overline{\mathcal{Z}}_t^{[i]} - \hat{\boldsymbol{z}}_t\right) \left(\overline{\mathcal{Z}}_t^{[i]} - \hat{\boldsymbol{z}}_t\right)^T + Q_t$
10: $\quad \overline{\Sigma}_t^{x,z} = \sum_{i=0}^{2n} w_c^{[i]} \left(\overline{\mathcal{X}}_t^{[i]} - \overline{\boldsymbol{\mu}}_t\right) \left(\overline{\mathcal{Z}}_t^{[i]} - \hat{\boldsymbol{z}}_t\right)^T$
11: $\quad K_t = \overline{\Sigma}_t^{x,z} S_t^{-1}$
12: $\quad \boldsymbol{\mu}_t = \overline{\boldsymbol{\mu}}_t + K_t(\boldsymbol{z}_t - \hat{\boldsymbol{z}}_t)$
13: $\quad \Sigma_t = \overline{\Sigma}_t - K_t S_t K_t^T$
14: $\quad$ **return** $\boldsymbol{\mu}_t, \Sigma_t$
15: **end procedure**

The general theory of the UKF is covered in this section and is used in the next section to apply the UKF to solve a localisation problem.

4.5.1 Application of UKF

The unscented Kalman filter (UKF) is applied to solve the same localisation problem as the one given to the EKF in Subsection 4.3.2. An advantage of using the UKF is that it is not necessary to determine a Jacobian matrix of the transition or measurement functions beforehand. The UKF is therefore generic and suitable for a wide range of problems, where as in the case of the EKF, it is not always possible to calculate the Jacobian matrix of a function.

For this application of the UKF the parameter λ is defined as

$$\lambda = n - 1, \tag{4.37}$$

where n is the length of the state vector $\boldsymbol{x}_t$. The parameter κ is set to 0 and the parameter α can then be obtained from Equation 4.30. Finally, β is set to 2.

Algorithm 4 is implemented in Python to estimate the belief of the state $\boldsymbol{x}_t$ of the robot at time t. At each time step the controls, noisy measurements and the previous belief $bel(\boldsymbol{x}_{t-1})$ are given to the UKF, which produces the belief of the robot's state at time t. The result can be viewed in Figure 4.3, where the blue error ellipses indicate the 95% confidence level region of each belief $bel(\boldsymbol{x}_t)$. The blue dots indicate the mean of each belief $bel(\boldsymbol{x}_t)$ at time t. The precise position of the robot at each time step is indicated with a black cross.

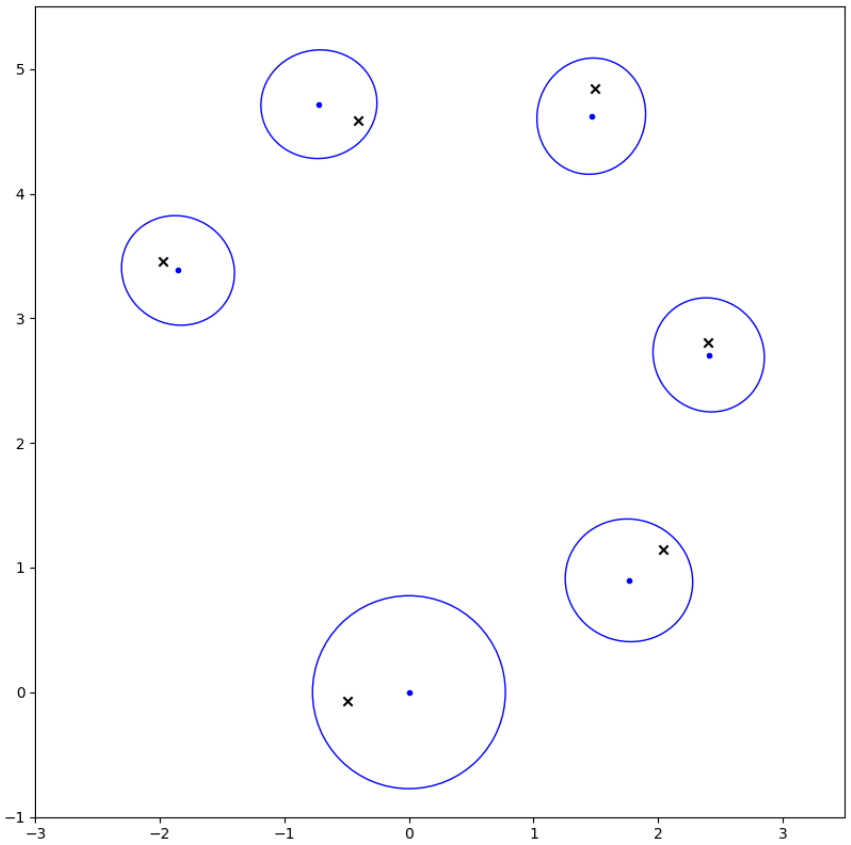

Figure 4.3: The robot is performing a nonlinear movement which is described by the velocity motion model. The belief of the robot's pose at each time step which is estimated by the UKF is shown in blue. The uncertainty of a belief is indicated with a 95% confidence error ellipse and the mean is indicated with a dot. The initial belief is illustrated in black.

This concludes the investigation to use the Kalman filter to solve the localisation problem. The following chapter covers the theory of the PGM which can be used as an alternative method to reason about the kinematic state of a robot.

Chapter 5

Probabilistic Graphical Models

This chapter gives a brief introduction to the probabilistic graphical models which will be used to solve the localisation problem, using different techniques of numerical integration. PGMs form part of a vast study field, therefore only the theory relevant to this report will be discussed. This chapter is based on the work of Koller and Friedman [2], Barber [7] and Schoeman [5].

5.1 Overview

Reasoning about systems that have a measure of uncertainty can become very complex. Sometimes these systems are completely unmanageable. The PGM is a graphical technique which makes problems tractable by modelling probabilistic systems in a logical and compact manner. PGMs are thus used to describe the relationship between variables and allow one to reason about it. The querying of a system or reasoning about a variable is called inference. It is a very popular technique, as it is intuitive, transparent and easy to manipulate. Hence, PGMs have countless applications from diagnosing medical problems to estimating the location of a robot.

PGMs used for modelling can be divided into two categories. The first includes Markov networks, which are used for non-causal systems. The second includes Bayesian networks, where relationships between random variables are causal and specified as CPDs. As the localisation problem in most cases is causal, the focus of this chapter is on Bayesian networks. Due to the relevance to localisation, the theory is discussed in terms of continuous random variables, but it can also be applied to discrete random variables.

5.2 Basic Concepts of Graphical Models

There are a few basic concepts to understand before defining a Bayesian network. Graphical models consist of nodes and edges. Edges are the links between nodes and can be directed or undirected. A Markov network is described by an undirected graph, where all edges are undirected. A Bayesian network is described by a directed graph, where all the edges are directed. An example of an undirected and directed graph can be seen in Figure 5.1 and Figure 5.2 respectively.

In the case of directed graphs, nodes can be classified as ancestors, parents, children and descendants. If there exists a directed path from $x_1 \rightarrow x_k$, then x_1 is an ancestor of x_k, and x_k is a descendant of x_1. In Figure 5.2, a is an ancestor of c and c is a descendant

of a. A parent is an ancestor with only one edge between the ancestor and descendant. A child is a descendant with only one edge between the descendant and the ancestor. In Figure 5.1, a is a parent of b and b is a child of a. [7]

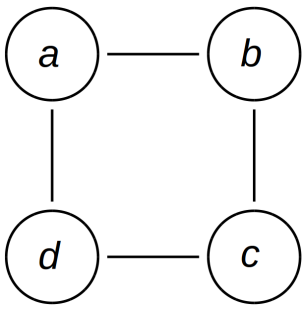

Figure 5.1: Undirected graph

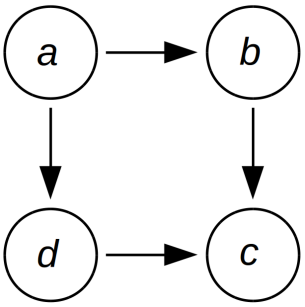

Figure 5.2: Directed graph

5.3 Bayesian Networks

Bayesian networks consists of random variables in the form of nodes and are connected by directed edges. Nodes that are not directly connected to each other, are considered conditionally independent. Relationships between nodes are indicated as conditional probability distributions (CPDs) and each node can be associated with a CPD

$$x_i \sim p(x_i|Par(x_i)), \tag{5.1}$$

where $Par(x_i)$ indicates the parents of node x_i.

CPDs can be written as a factors. A factor is a function that takes a number of random variables as arguments and is defined as

$$\phi_i(x_i, Par(x_i)) = p(x_i|Par(x_i)), \tag{5.2}$$

where the scope of a factor is its arguments

$$\text{Scope}\{\phi_i\} = \{x_i, Par(x_i)\}. \tag{5.3}$$

The concept of a factor is important that will be used in the rest of the report.

Figure 5.3 shows an example of a Bayesian network with seven nodes labelled from a to g. Each node is associated with a CPD. Directed edges between nodes are indicated with arrows.

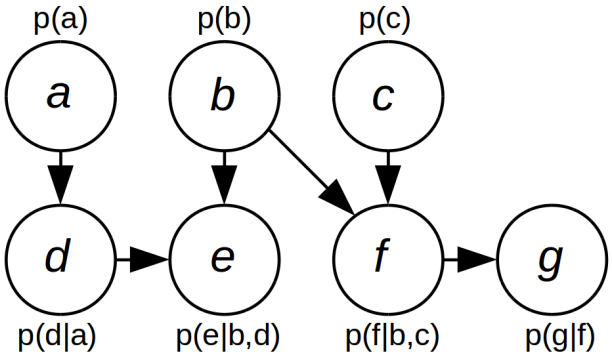

Figure 5.3: Bayesian network

The chain rule specifies that the joint probability density distribution of all the variables in a Bayesian network can be determined by calculating the product of all the CPDs associated with the nodes [2]. It is defined as

$$p(x_1, ..., x_n) = \prod_i p(x_i|Par(x_i)). \tag{5.4}$$

The chain rule can be applied to the Bayesian network in Figure 5.3

$$p(a, b, c, d, e, f, g) = p(a)p(b)p(c)p(d|a)p(e|b, d)p(f|b, c)p(g|f) \tag{5.5}$$

A marginal PDF of the subset of variables a, b and c can be determined by integrating over all the variables not in the subset, therefore

$$p(a, b, c) = \int\int\int\int p(a, b, c, d, e, f, g)dd\, de\, df\, dg. \tag{5.6}$$

The chain rule together with marginalisation can be used to infer a Bayesian network, but it can be very tedious. The next section covers an alternative method to reason about a Bayesian network in a structured manner.

5.4 Cluster Graphs

The next step is to reason about variables of a Bayesian network in a more efficient manner. Various methods can be used to inference a Bayesian network; one method is to construct a cluster graph. The cluster graph consists of clusters connected by undirected edges.

Clusters can be formed by grouping factors together such that each cluster C_i is a subset of the variables of the Bayesian network

$$C_i \subseteq \{x_1, ..., x_n\}. \tag{5.7}$$

The undirected edges between clusters are called sepsets and are responsible for passing messages between clusters. A sepset between two clusters contains information about variables that are common to both clusters. An edge between C_i and cluster C_j is associated with a sepset which contains variables that are common to both C_i and C_j and is defined as

$$S_{i,j} \subseteq C_i \cap C_j. \tag{5.8}$$

There are multiple ways to construct clusters, but it should adhere to two requirements [2]:

1. Family preservation requires that every factor in the Bayesian network is accommodated by some cluster.

2. The running intersection property states that there exists an unique path connecting a pair of clusters containing the same variable x, and every cluster and sepset along the path also contain x. This path allows clusters to share their beliefs of x. In other words, for any variable x, the set of sepsets and clusters containing x form a tree [2]. This prevents feedback loops and thus counters the phenomena where clusters reinforce their own beliefs of variables.

Figure 5.3 is converted into Figure 5.4 where CPDs are replaced with factors and possible cluster boundaries are indicated with dashed rectangles. Note that there are other ways to construct the cluster graph.

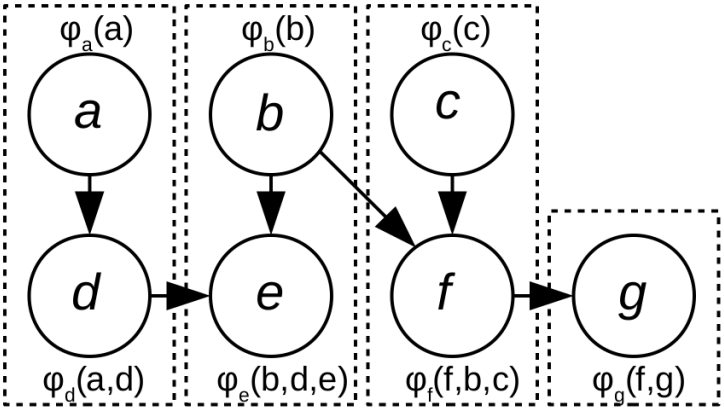

Figure 5.4: Bayesian network with cluster boundaries

The cluster factor ψ_i can be calculated by finding the product of all the factors inside the cluster [2] and is therefore defined as

$$\psi_i(C_i) = \prod_k \phi_k. \tag{5.9}$$

Figure 5.4 is transformed in Figure 5.5. Clusters are indicated by ellipsis. Initial beliefs are calculated and shown inside the ellipsis. Sepsets are indicated by rectangles.

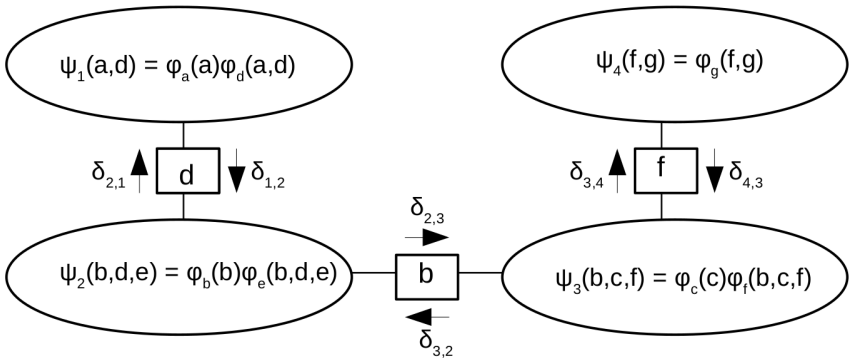

Figure 5.5: Cluster graph showing sepsets and clusters with initial beliefs

This section shows how to construct a cluster graph with clusters and sepsets. The cluster graph allows clusters to share beliefs of variables by means of message passing, which will be explored in the next section.

5.5 Message Passing

Clusters can share beliefs of common variables by using message passing. Every sepset $S_{i,j}$ has two messages, $\delta_{i,j}$ and $\delta_{j,i}$, associated with it. This allows clusters, connected by a sepset, to communicate in both directions. An outgoing message $\delta_{i,j}$ of a cluster C_i can be calculated by multiplying the cluster factor ψ_i with the incoming message $\delta_{k,i}$ from each other cluster C_k and then integrating the result over the variables that are not in the sepset $(C_i - S_{i,j})$. This procedure is defined as [2]

$$\delta_{i,j}(S_{i,j}) = \int_{C_i - S_{i,j}} \psi_i \times \prod_{k \neq j} \delta_{k,i}(S_{k,i}). \tag{5.10}$$

Equation 5.10 can be demonstrated by calculating arbitrary messages $\delta_{2,3}$ and $\delta_{3,4}$ in Figure 5.5, therefore

$$\delta_{2,3}(b) = \int_d \int_e \delta_{1,2}(d)\psi_2(b,d,e)dd\ de \tag{5.11}$$

and

$$\delta_{3,4}(f) = \int_b \int_c \delta_{2,3}(b)\psi_3(b,c,f)db\ dc. \tag{5.12}$$

Sometimes the value of a random variable (RV) in a PGM is known, in other words there is evidence of a RV available. Evidence of random variables is available to the entire PGM, therefore after evidence of a RV is inserted, the RV itself doesn't appear in the PGM anymore. Evidence is indicated by a capital letter, thus X is evidence of the RV x. Evidence is used to reduce a PDF before calculating messages. For example if the value of the variable c in the Bayesian network 5.3 is known, then it is indicated with C. The message in Equation 5.12 can be recalculated by first reducing the cluster factor ψ_3 with the evidence which is available, therefore the message $\delta_{3,4}$ can be calculated as

$$\delta_{3,4}(f) = \int_b \delta_{2,3}(b)\psi_3(b,c=C,f)db. \tag{5.13}$$

After all the incoming messages of a cluster have been determined, the belief of a cluster can be calculated by multiplying the cluster factor ψ_i with all the incoming messages, and normalising the end result. Koller and Friedman [2] specifies this as

$$\beta_i(C_i) \propto \psi_i \times \prod_k \delta_{k,i}(S_{k,i}). \tag{5.14}$$

This chapter covers the properties of the Bayesian network and how a cluster graph can be used to reason about the variables of a Bayesian network. The Bayesian network will be used in the next chapter to model the localisation problem.

Chapter 6

Localisation using Probabilistic Graphical Models

In this chapter a PGM will be used as an alternative method to reason about a localisation problem. The problem is modelled with a Bayesian network in Section 6.1 and in Section 6.2 a cluster graph is constructed to reason about the location of the robot. This technique then is compared to that of the Bayes filter in Algortim 1. This chapter is based on the work of Koller and Friedman [2].

6.1 Modelling the Localisation problem

A Bayesian network, shown in Figure 6.1, is used to model the localisation problem. There are nodes representing the control $\boldsymbol{u}_t$, the state of the robot $\boldsymbol{x}_t$ and the measurement $\boldsymbol{z}_t$ at each time step ($t = 1, \ ..,T$). The relationship between the nodes are indicated as conditional probability densities (CPDs) which can also be written as factors, where

$$\phi_{\boldsymbol{u}_t}(\boldsymbol{u}_t) = p(\boldsymbol{u}_t), \tag{6.1}$$

$$\phi_{\boldsymbol{x}_t}(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{x}_{t-1}) = p(\boldsymbol{x}_t|\boldsymbol{u}_t, \boldsymbol{x}_{t-1}) \tag{6.2}$$

and

$$\phi_{\boldsymbol{z}_t}(\boldsymbol{z}_t, \boldsymbol{x}_t) = p(\boldsymbol{z}_t|\boldsymbol{x}_t). \tag{6.3}$$

The current state of the robot's pose $\boldsymbol{x}_t$ is dependent on the previous state $\boldsymbol{x}_{t-1}$ and control $\boldsymbol{u}_t$. The measurement $\boldsymbol{z}_t$ is dependent on the state $\boldsymbol{x}_t$. The belief of the initial state $bel(\boldsymbol{x}_0)$ is known, and the values of the control vectors $\boldsymbol{u}_t$ and measurement vectors $\boldsymbol{z}_t$ are given as evidence by $\boldsymbol{U}_t$ and $\boldsymbol{Z}_t$, respectively.

6.2 Reasoning with a Cluster Graph

The Bayesian network in Figure 6.1 is converted to a cluster graph, as shown in Figure 6.2. The cluster graph allows one to reason about the RVs of the Bayesian network. There are a numerous ways to group the factors to form clusters. In this cluster graph factors $\phi_{\boldsymbol{u}_t}, \phi_{\boldsymbol{x}_t}$ and $\phi_{\boldsymbol{z}_t}$ at each time step t were grouped together to form a cluster C_t. The cluster factor of each cluster C_t can be determined as

$$\psi_t(\boldsymbol{x}_t, \boldsymbol{x}_{t-1}, \boldsymbol{u}_t, \boldsymbol{z}_t) = \phi_{\boldsymbol{x}_t}(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{x}_{t-1})\phi_{\boldsymbol{z}_t}(\boldsymbol{z}_t, \boldsymbol{x}_t)\phi_{\boldsymbol{u}_t}(\boldsymbol{u}_t). \tag{6.4}$$

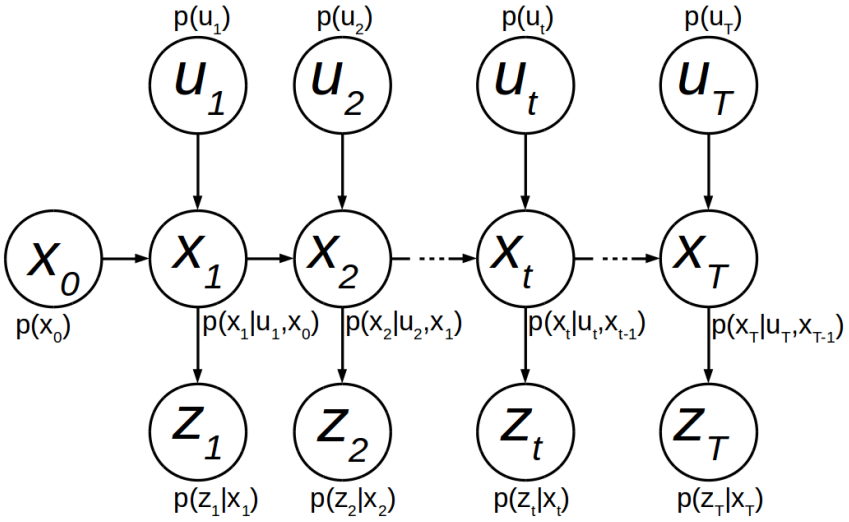

Figure 6.1: The Bayesian network used to model the localisation problem. The nodes of the Bayesian network are indicated with circles and edges are indicated with arrows. The CPD associated with each node is shown.

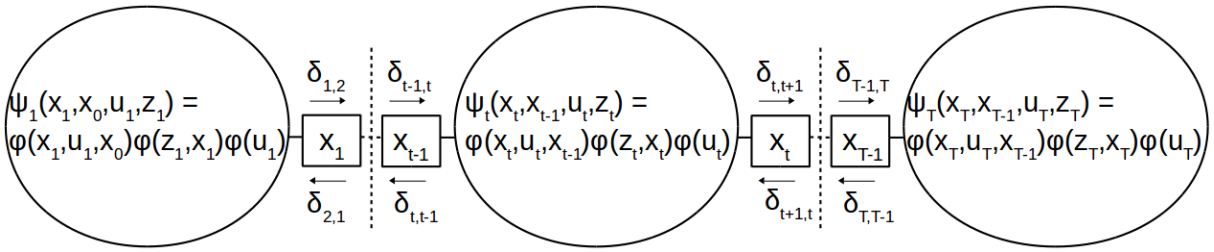

Figure 6.2: The Bayesian network in Figure 6.1 is transformed into a cluster graph where clusters are indicated with ellipsis. The sepsets are illustrated as rectangles and associated messages between clusters are indicated with arrows.

Sepsets between clusters are indicated with rectangles and allow clusters to reason about mutual variables by means of message passing. In Figure 6.2 messages associated with sepsets are shown in both directions, but for this example only messages in the "forward" direction are calculated. The messages pointing from right to left can also be determined and can be then used to sent information "backwards" in time which can lead to better estimations of beliefs at previous time steps. The process where messages are sent in both directions is known as smoothing.

The initial belief $bel(\boldsymbol{x}_0)$ of the robot's state is known and can be viewed as the only incoming message of cluster C_1, therefore

$$\delta_{0,1}(\boldsymbol{x}_t) = bel(\boldsymbol{x}_0). \tag{6.5}$$

This allows one to calculate the messages of the cluster graph in Figure 6.2 by starting at the very left cluster, C_1, and then calculate each cluster's outgoing message, in a consecutive order. By using Equation 5.10, the message $\delta_{1,2}$ can therefore be calculated as

$$\begin{aligned} \delta_{1,2}(\boldsymbol{x}_1) &= \int_{\boldsymbol{x}_0} \psi_t(\boldsymbol{x}_1, \boldsymbol{x}_0, \boldsymbol{u}_1, \boldsymbol{z}_1)\delta_{0,1}(\boldsymbol{x}_0)\, d\boldsymbol{x}_0 \\ &= \int_{\boldsymbol{x}_0} \psi_t(\boldsymbol{x}_1, \boldsymbol{x}_0, \boldsymbol{u}_1, \boldsymbol{z}_1) bel(\boldsymbol{x}_0)\, d\boldsymbol{x}_0. \end{aligned} \tag{6.6}$$

The message $\delta_{2,3}$ can be calculated after the message $\delta_{1,2}$ has been calculated, and the message $\delta_{3,4}$ can again be calculated after $\delta_{2,3}$ has been calculated and so on. Keeping in mind that an outgoing message of a cluster includes the information of all the preceding clusters, and making the observation that the procedure of Equation 6.6 is very similar to that of the Bayes filter, the conclusion can be made that the result of the operations of this PGM is identical to that of the Bayes filter. The PGM has the advantage that it is generic and easy to change by simply adding or removing nodes. Smoothing can also be used in a PGM which can lead to better estimations. The conclusion is made the this approach of using a PGM leads to the same result as that of the Bayes filter. In the next chapter it is explored how the operations of the Bayes filter can be computed.

Chapter 7

Linearisation of Nonlinear Transformations

Using the PGM of the previous chapter to reason about the state of a robot, has the same result as that of the Bayes filter. This generic way of reasoning is advantageous as different methods can easily be implemented to do the operations of the Bayes filter. In this project is decided to do these operations in the canonical form. In order to do the operations of the Bayes filter, all the probability distributions must therefore be represented with the canonical form. However, a conditional probability distribution (CPD) that is described by a nonlinear transformation is usually a non-Gaussian distribution and cannot be represented with the canonical form. This section investigates three methods to approximate nonlinear transformations as linear transformations which allow one to approximate these CPDs as Gaussian distributions. This chapter is based on the work of Thrun [6] and notes from Van Daalen.

7.1 Taylor Expansion

Taylor expansion is again used to linearise a nonlinear functions which are associated with the state transition and measurement CPDs. The linearisation process is discussed in Section 4.4 and the results of Equations 4.14 to 4.17 are used again. The linearised functions can be represented by the canonical form by implementing it in the the result of Equation 2.44. A nonlinear state transition function can be defined as

$$\boldsymbol{x}_t = \boldsymbol{g}(\boldsymbol{u}_t, \boldsymbol{x}_{t-1}) + \boldsymbol{\varepsilon}_t, \tag{7.1}$$

where

$$\boldsymbol{\varepsilon}_t \sim \mathcal{N}(\boldsymbol{\varepsilon}, \mathbf{0}, R). \tag{7.2}$$

The nonlinear function, $\boldsymbol{g}(.)$ in Equation 7.1 can be linearised by means of Taylor expansion. The state transition function in Equation 7.1 can now be approximated as

$$\boldsymbol{x}_t \approx \boldsymbol{g}(\boldsymbol{u}_t, \boldsymbol{\mu}_{t-1}) + G_t(\boldsymbol{x}_{t-1} - \boldsymbol{\mu}_{t-1}) + \boldsymbol{\varepsilon}_t. \tag{7.3}$$

The result of Equation 7.3 can be written in the form $\boldsymbol{y} = F\boldsymbol{x} + \boldsymbol{g} + \boldsymbol{n}$:

$$\boldsymbol{x}_t \approx G_t\boldsymbol{x}_{t-1} + (\boldsymbol{g}(\boldsymbol{u}_t, \boldsymbol{\mu}_{t-1}) - G_t\boldsymbol{u}_{t-1}) + \boldsymbol{\varepsilon}_t. \tag{7.4}$$

By implementing the result of Equation 7.4 in Equation 2.45, the state transition distribution $p(\boldsymbol{x}_t|\boldsymbol{u}_t, \boldsymbol{x}_{t-1})$ can be approximated as a Gaussian distribution in canonical form:

$$\mathcal{C}\left(\begin{bmatrix}\boldsymbol{x}_t \\ \boldsymbol{x}_{t-1}\end{bmatrix}; \begin{bmatrix}R^{-1} & -R^{-1}G_t \\ -G_t^T\Sigma^{-1} & G_t^TR^{-1}G_t\end{bmatrix}, \begin{bmatrix}-R^{-1}(\boldsymbol{g}(\boldsymbol{u}_t, \boldsymbol{\mu}_{t-1}) - G_t\boldsymbol{u}_{t-1}) \\ -G_t^TR^{-1}(\boldsymbol{g}(\boldsymbol{u}_t, \boldsymbol{\mu}_{t-1}) - G_t\boldsymbol{u}_{t-1})\end{bmatrix}\right). \tag{7.5}$$

The the same procedure can be followed to approximate a measurement CPD with a nonlinear relationship in its arguments as a Gaussian distribution. The nonlinear function that describes the measurement CPD is defined as

$$\boldsymbol{z}_t = \boldsymbol{h}(\boldsymbol{x}_t) + \boldsymbol{\zeta}_t, \tag{7.6}$$

where

$$\boldsymbol{\zeta} \sim \mathcal{N}(\boldsymbol{\zeta}; \boldsymbol{0}, Q). \tag{7.7}$$

The nonlinear function $\boldsymbol{h}(.)$ in Equation 7.6 can be linearised using Taylor expansion. Equation 7.6 can therefore be approximated as

$$\boldsymbol{z}_t = \boldsymbol{h}(\overline{\boldsymbol{\mu}}_t) + H_t(\boldsymbol{x}_t - \overline{\boldsymbol{\mu}}_t) + \boldsymbol{\zeta}_t. \tag{7.8}$$

Equation 7.8 is written in the form of $\boldsymbol{y} = F\boldsymbol{x} + \boldsymbol{g} + \boldsymbol{n}$, therefore

$$\boldsymbol{z}_t = H_t\boldsymbol{x}_t + (\boldsymbol{h}(\overline{\boldsymbol{\mu}}_t) - H_t\overline{\boldsymbol{\mu}}_t) + \boldsymbol{\zeta}_t. \tag{7.9}$$

The measurement CPD can be be approximated as a Gaussian distribution by implementing the the result of Equation 7.4 in Equation 2.45, therefore:

$$\mathcal{C}\left(\begin{bmatrix}\boldsymbol{z}_t \\ \boldsymbol{x}_t\end{bmatrix}; \begin{bmatrix}Q_t^{-1} & -Q_t^{-1}H_t \\ -H_t^T\Sigma^{-1} & H_t^TQ_t^{-1}H_t\end{bmatrix}, \begin{bmatrix}-Q_t^{-1}(\boldsymbol{h}(\overline{\boldsymbol{\mu}}_t) - H_t\overline{\boldsymbol{\mu}}_t) \\ -H_t^TQ_t^{-1}(\boldsymbol{h}(\overline{\boldsymbol{\mu}}_t) - H_t\overline{\boldsymbol{\mu}}_t)\end{bmatrix}\right). \tag{7.10}$$

All the probability distributions of the Bayes filter can now be represented with the canonical form. This method will give exactly the same result as the EKF. This is due to the fact that both methods use Taylor expansion to linearise the nonlinear functions, $\boldsymbol{g}(\boldsymbol{u}_t, \boldsymbol{x}_{t-1})$ and $\boldsymbol{h}(\boldsymbol{x}_t)$.

This Taylor expansion along with the canonical form and Bayes filter is used to approximate the belief of the state of a robot. The exact same result is obtained as when the extended Kalman filter (EKF) is used.

7.2 Unscented Transform

As mentioned before, passing a Gaussian random variable (RV) through a nonlinear transformation will result in a non-Gaussian RV. The unscented transform can be used to approximate this result as a Gaussian RV by using a form of numerical integration. This method is already described and in Section 4.5 and used in the algorithm of the unscented Kalman filter. This section investigates how the unscented transform can be used as a method to linearise a nonlinear transformation. An arbitrary nonlinear function is defined as

$$\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x}). \tag{7.11}$$

The unscented transform carefully extracts Sigma points $\mathcal{X}$ from the Gaussian distribution which is associated with the RV $\boldsymbol{x}$. These sigma points are then passed through the nonlinear transformation $f(\boldsymbol{x})$ to produce new Sigma points $\mathcal{Y}$ which is associated with

the RV $\boldsymbol{y}$. The transformed Sigma points $\mathcal{Y}$ along with weights, w_c and w_m, can be used to estimate the resultant distribution over the RV $\boldsymbol{y}$ as a Gaussian distribution. The process of choosing Sigma points and calculating weights were discussed in Section 4.5.

The parameters of the approximated Gaussian distribution over the RV $\boldsymbol{y}$ are now determined. The mean, $\boldsymbol{\mu_y}$ is calculated as

$$\boldsymbol{\mu_y} = \sum_{i=0}^{2n} w_m^{[i]} \mathcal{Y}^{[i]}, \tag{7.12}$$

and covariance $\Sigma_{\boldsymbol{yy}}$ can be determined as

$$\Sigma_{\boldsymbol{yy}} = \sum_{i=0}^{2n} w_c^{[i]} \left(\mathcal{Y}^{[i]} - \boldsymbol{\mu_y}\right) \left(\mathcal{Y}^{[i]} - \boldsymbol{\mu_y}\right)^T. \tag{7.13}$$

The cross covariance $\Sigma_{\boldsymbol{xy}}$ can be calculated as

$$\Sigma_{\boldsymbol{xy}} = \sum_{i=0}^{2n} w_c^{[i]} \left(\mathcal{X}^{[i]} - \boldsymbol{\mu_x}\right) \left(\mathcal{Y}^{[i]} - \boldsymbol{\mu_y}\right)^T. \tag{7.14}$$

The next step is to describe the linear relationship between the RV $\boldsymbol{y}$ and the RV $\boldsymbol{x}$. It is therefore now investigated how the results of Equation 7.12 to 7.14 can be used to approximate Equation 7.11 as a linear function

$$\boldsymbol{y} \approx F\boldsymbol{x} + \boldsymbol{g} + \boldsymbol{n}, \tag{7.15}$$

where

$$\boldsymbol{n} \sim \mathcal{N}(\boldsymbol{n}; \boldsymbol{0}, \Sigma_{\boldsymbol{n}}). \tag{7.16}$$

The mean , covariance and cross-covariance are now determined in terms of Equation 7.15. The mean $\boldsymbol{\mu_y}$ of RV $\boldsymbol{y}$ can be determined by calculating the expected value of Equation 7.15, therefore

$$\begin{aligned} \boldsymbol{\mu}_y &= \mathcal{E}[F\boldsymbol{x} + \boldsymbol{g} + \boldsymbol{n}] \\ &= F\boldsymbol{\mu_x} + \boldsymbol{g}. \end{aligned} \tag{7.17}$$

The covariance matrix $\Sigma_{\boldsymbol{yy}}$ associated with RV $\boldsymbol{y}$ can be determined by using the definition of covariance which is defined in Equation 2.11, therefore

$$\begin{aligned} \Sigma_{\boldsymbol{yy}} &= \mathcal{E}\left[(\boldsymbol{y} - \boldsymbol{\mu_y})(\boldsymbol{y} - \boldsymbol{\mu_y})^T\right] \\ &= \mathcal{E}\left[(F\boldsymbol{x} + \boldsymbol{g} + \boldsymbol{n} - F\boldsymbol{\mu_x} - \boldsymbol{g})((F\boldsymbol{x} + \boldsymbol{g} + \boldsymbol{n} - F\boldsymbol{\mu_x} - \boldsymbol{g})^T\right] \\ &= F\mathcal{E}\left[(\boldsymbol{x} - \boldsymbol{u_x})(\boldsymbol{x} - \boldsymbol{\mu_x})^T\right] F^T + F\mathcal{E}\left[(\boldsymbol{x} - \boldsymbol{\mu_x})n^T\right] + \mathcal{E}\left[n(\boldsymbol{x} - \boldsymbol{\mu_x})^T\right] F^T + \mathcal{E}\left[\boldsymbol{n}\boldsymbol{n}^T\right] \\ &= F\Sigma_{\boldsymbol{xx}}F^T + \Sigma_{\boldsymbol{n}} \end{aligned} \tag{7.18}$$

and the cross-covariance $\Sigma_{\boldsymbol{xy}}$ can be calculated as

$$\begin{aligned} \Sigma_{\boldsymbol{xy}} &= \mathcal{E}\left[(\boldsymbol{x} - \boldsymbol{\mu_x})(\boldsymbol{y} - \boldsymbol{\mu_y})^T\right] \\ &= \Sigma_{\boldsymbol{xx}}F^T. \end{aligned} \tag{7.19}$$

The parameters of the linear function described in Equation 7.15 can now be calculated. Equation 7.19 can be used to determine the matrix F, therefore,

$$F = \Sigma_{\boldsymbol{xy}}^T \Sigma_{\boldsymbol{xx}}^{-1}, \tag{7.20}$$

the vector $\boldsymbol{g}$ can be determined by using Equation 7.17, therefore

$$\begin{aligned} \boldsymbol{g} &= \boldsymbol{\mu_y} - \Sigma_{\boldsymbol{xy}}^T \Sigma_{\boldsymbol{xx}}^{-1} \boldsymbol{\mu_x} \\ &= \boldsymbol{\mu_y} - F\boldsymbol{\mu_x} \end{aligned} \tag{7.21}$$

and the covariance matrix Σ_n can be determined as

$$\Sigma_n = \Sigma_{\boldsymbol{yy}} - \Sigma_{\boldsymbol{xy}}^T \Sigma_{\boldsymbol{xx}}^{-1} \Sigma_{\boldsymbol{xy}}. \tag{7.22}$$

The above shows how the unscented transform can be used to approximate a nonlinear transformation as a linear transformation. The non-linear functions $\boldsymbol{g}(\boldsymbol{u}_t, \boldsymbol{x}_{t-1})$ and $\boldsymbol{h}(\boldsymbol{x}_t)$ can be linearised in the same way which will allow one to represent the state transition and measurement distributions in the canonical form.

The unscented transform along with the canonical form and Bayes filter is used to approximate the belief of the state of a robot. The exact same result is obtained as when the unscented Kalman filter (UKF) is used.

7.3 Monte Carlo Integration

Monte Carlo integration is a method of numerical integration that can be used to approximate a distribution as a Gaussian distribution. This can therefore also be used to approximate a nonlinear relationship, which describes a conditional probability distribution, as a linear relationship. This technique draws N random points $\mathcal{X}^*$ from a Gaussian distribution, associated with a RV $\boldsymbol{x}$, and passes them through a nonlinear transformation to produce resultant points $\mathcal{Y}^*$ which are associated with a RV $\boldsymbol{y}$. The resultant points $\mathcal{Y}^*$ can be used to calculate the parameters of the Gaussian distribution over the RV $\boldsymbol{y}$. The mean $\boldsymbol{\mu_y}$ can be calculated with

$$\boldsymbol{\mu_y} = \frac{1}{N} \sum_{i=1}^{N} \mathcal{Y}^{*[i]}, \tag{7.23}$$

the covariance $\Sigma_{\boldsymbol{yy}}$ can be calculated with

$$\Sigma_{\boldsymbol{yy}} = \frac{1}{N} \sum_{i=1}^{N} (\mathcal{Y}^{*[i]} - \boldsymbol{\mu_y})(\mathcal{Y}^{*[i]} - \boldsymbol{\mu_y})^T \tag{7.24}$$

and lastly the cross-covariance $\Sigma_{\boldsymbol{xy}}$ can be calculated as

$$\Sigma_{\boldsymbol{xy}} = \frac{1}{N} \sum_{i=1}^{N} (\mathcal{X}^{*[i]} - \boldsymbol{\mu_x})(\mathcal{Y}^{*[i]} - \boldsymbol{\mu_y})^T. \tag{7.25}$$

The approximated linear transform ($\boldsymbol{y} \approx F\boldsymbol{x} + \boldsymbol{g} + \boldsymbol{n}$) from RV $\boldsymbol{x}$ to RV $\boldsymbol{y}$ can be obtained in the same manner as in the previous section, therefore one can use Equation 7.2 to Equation 7.22 again.

Chapter 8

Results

All three techniques covered in the previous chapter were successfully implemented into the Bayes filter in order to estimate the state of a robot. This chapter investigates how these techniques perform in terms of accuracy and efficiency. In order to compare the performance of the techniques, a common localisation problem has to be solved by using each technique. The velocity motion model is used to describe the movement of a robot which is simulated in Python. A measurement of the robot's pose at each time step is generated with the linear measurement model. Each technique, together with the Bayes filter, is then implemented in Python to solve the same localisation problem, which allows the performance of the techniques to be compared. A part of this chapter is based on the work of Duchi [8].

8.1 Accuracy

The beliefs, determined by implementing the different approximation techniques into the Bayes filter, are now compared in terms of accuracy. First the *ground truth*, which is the desired belief intended to be calculated, has to be obtained. This desired belief is defined as the optimal Gaussian distribution which describes the uncertainty of the state of a robot. It is not possible to determine a finite answer for the ground truth of this nonlinear system, but a very accurate approximation can be made by using a very high sample count in the Monte Carlo technique.

The next step is to compare the different estimated beliefs with the ground truth. Kullback-Leibler (KL) divergence is the method chosen to measure the difference between the Gaussian distributions. KL divergence gives a measure of how well one distribution approximates another, where an answer of zero indicates that the distributions are identical. KL divergence is generally defined as

$$D_{KL}(p_1(\boldsymbol{x})||p_2(\boldsymbol{x})) = \mathcal{E}\left(\ln\frac{p_1(\boldsymbol{x})}{p_2(\boldsymbol{x})}\right), \tag{8.1}$$

where the result is a measure of how well $p_2(\boldsymbol{x})$ approximates $p_1(\boldsymbol{x})$. The KL divergence can be applied to specifically compare multivariate Gaussian distributions, which is the nature of all the distributions that are compared. If the distributions $p_1(\boldsymbol{x})$ and $p_2(\boldsymbol{x})$ in Equation 8.1 are multivariate Gaussian distributions defined as

$$p_1(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_1, \Sigma_1), \tag{8.2}$$

and

$$p_2(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_2, \Sigma_2), \tag{8.3}$$

then, according to Duchi [8], Equation 8.1 can be modified to

$$D_{KL}(p_1(\boldsymbol{x})||p_2(\boldsymbol{x})) = \frac{1}{2}\left(\ln\frac{|\Sigma_2|}{|\Sigma_1|} - n + \operatorname{tr}\left(\Sigma_2^{-1}\Sigma_1\right) + (\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)^T\Sigma_2^{-1}(\boldsymbol{\mu_2} - \boldsymbol{\mu_1})\right), \tag{8.4}$$

where n is the number of dimensions of the compared multivariate Gaussian distributions.

To compare the accuracy of the different techniques, it is important that the prior belief $bel(\boldsymbol{x}_{t-1})$ which the different estimators receive, are the same. Ten thousand simulations are executed and the result of each technique is compared to the ground truth by using Equation 8.4. Figure 8.1 shows the result of running a single simulation. The prior belief is shown in black and the posterior beliefs, computed by each technique, are illustrated by the different colour error ellipsis. The result of each technique can be compared to the ground truth, which is shown by the red error ellipse. In this particular simulation the unscented transform performed the best and the Monte Carlo method (10 samples) performed the worst.

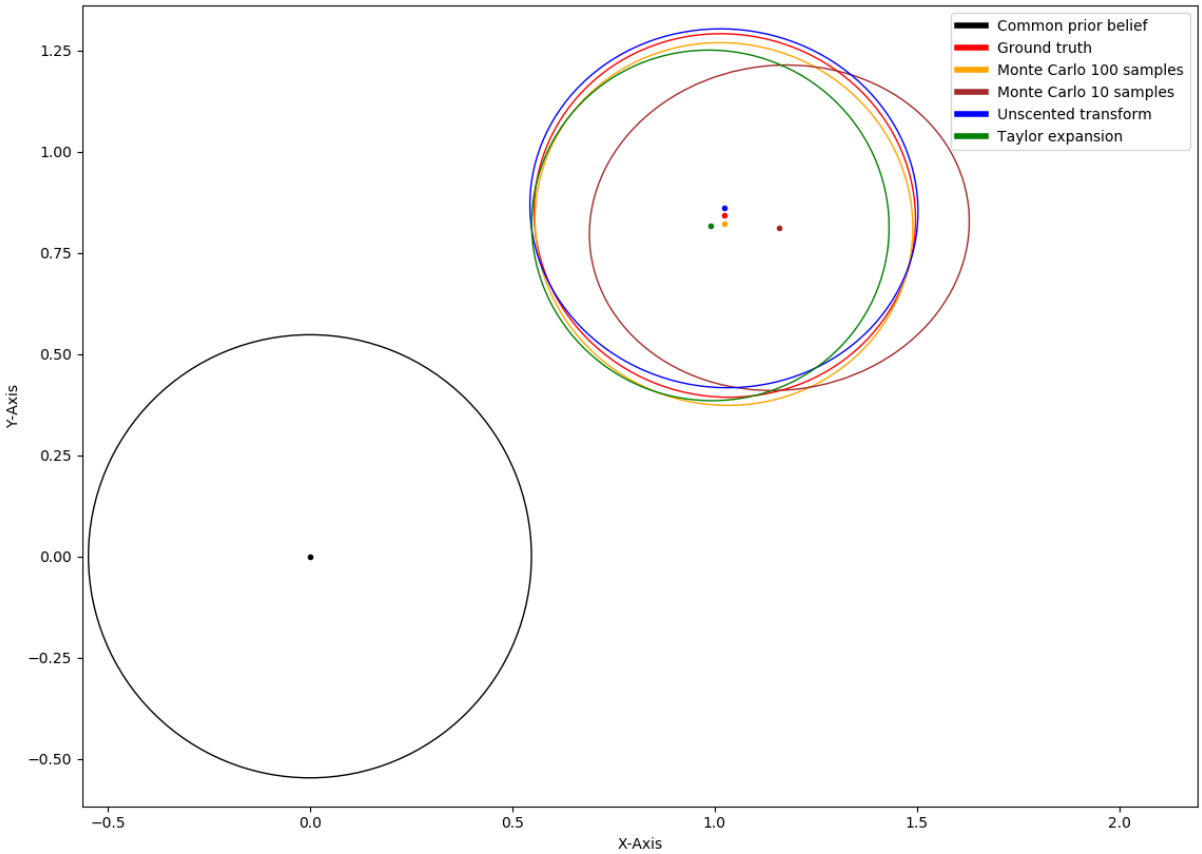

Figure 8.1: A visual comparison of the accuracy of the different approximation techniques. The black ellipse represents the prior belief at this specific time step. The posterior beliefs, calculated by each technique, are indicated by the colour error ellipsis.

The average results can be observed in Table 8.1 and results that are closer to zero indicate better approximations.

Technique Used	KL divergence results
1. Monte Carlo (100 samples)	0.0057
2. Monte Carlo (10 samples)	0.0744
3. Taylor Expansion	0.0758
4. Unscented Transform	0.0769

Table 8.1: Ten thousand simulations are executed. The approximation of each technique is compared to the ground truth by means of Kullback-Leibler (KL) divergence. The average results are shown, where a number closer to zero suggests a closer approximation to the ground truth. The table is in an ascending order.

The results in Table 8.1 suggests that the approximation methods which are investigated, performed very similar in terms of accuracy, with the exception of the Monte Carlo technique (100 samples) being substantially more accurate. It is also observed that if the Monte Carlo technique is performed with a few samples only, then the accuracy will depend on the specific samples that are extracted. This can lead to more accurate or less accurate results. It is clear that by using more samples with the Monte Carlo method, one can expect a result which is closer to the desired belief. The reduction in computational efficiency when using the Monte Carlo method with more samples will be investigated in the next section. As mentioned earlier, it is expected that the unscented transform will perform better than the Taylor expansion in terms of accuracy, but it is not proved in this simulation. The results in table 8.1 suggest that these techniques perform very similar in terms of approximating the desired belief. This can possibly be assigned to the fact that a linear measurement model is used for this problem. It is expected that if a nonlinear measurement model is used, one will be able to distinguish better between the techniques in terms of accuracy.

Monte Carlo integration draws points randomly where the unscented transform chooses points deterministically. If the concealed distribution is almost a Gaussian distribution, then the unscented transform will be very efficient and accurate. On the other hand if the concealed distribution is highly non-Gaussian then the unscented transform will perform poorly.

8.2 Computational Efficiency

The computational efficiency of each technique is determined empirically by measuring the time taken for an execution. Ten thousand simulations are executed and the execution time of each technique is measured. The average execution time of each technique can be seen in Table 8.2

Technique Used	Execution time (seconds)
1. Monte Carlo (10 000 samples)	1.27180
2. Monte Carlo (100 samples)	0.01404
3. Monte Carlo (10 samples)	0.00161
4. Unscented transform	0.00161
5. Taylor expansion	0.00160

Table 8.2: The average execution time of each linearisation technique is indicated and is sorted in a descending order.

It can be seen from Table 8.2 that the Monte Carlo (10 samples) unscented transform and Taylor expansion techniques have very similar execution times. The Monte Carlo technique's execution time increases as the sample count increases. This trend is shown in **Figure..**.

Chapter 9

Conclusion

This project investigates traditional methods to estimate the location of a nonlinear moving robot. The Kalman filter is first applied to solve a simple linear localisation problem in order to get a better understanding of the basic principles of the Bayes filter. The project then investigates how one can estimate the state of robot which is moving in a nonlinear manner. Traditional recursive filters namely the extended Kalman filter (EKF) and the unscented Kalman filter (UKF), are applied to solve a nonlinear localisation problem.

The PGM as an alternative method to reason about the localisation problem, has numerous advantages and is also discussed. The conclusion is made that the results obtained by reasoning with the PGM described in Chapter 6 is identical to that of the Bayes filter. The generic Bayes filter is therefore used to implement alternative linearisation techniques to approximate the belief of the state of a robot.

Taylor expansion along with two numerical integration techniques are explored to linearise a nonlinear transformation in a conditional probability distribution (CPD). These techniques are implemented in Python and applied to solve a common nonlinear localisation problem. The techniques are then compared in terms of accuracy and efficiency. It is found that by using Monte Carlo integration with very high sample count, the desired Gaussian belief can be computed, but it becomes quickly computationally inefficient. The results of using the Monte Carlo integration (10 samples), unscented transform and Taylor expansion are very similar in terms of accuracy and computational efficiency. However, because of the random nature of Monte Carlo technique it can sometimes deliver very accurate or inaccurate results and is therefore not reliable.

The unscented transform is considered superior to using Taylor expansion, but it is not evident in the results of this project. It is expected that if a nonlinear measurement model is used, the results of these techniques may be distinguished better.

List of References

[1] Peebles, P.Z.: *Probability, random variables, and random signal principles*, vol. 3. McGraw-Hill New York, NY, USA:, 2001.

[2] Koller, D., Friedman, N. and Bach, F.: *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[3] Abdi, H.: The eigen-decomposition: Eigenvalues and eigenvectors. *Encyclopedia of measurement and statistics*, pp. 304–308, 2007.

[4] Visiondummy, how to draw a covariance error ellipse. `http://www.visiondummy.com/2014/04/draw-error-ellipse-representing-covariance-matrix/`. Accessed: 2018-10-01.

[5] Schoeman, J.: *Simultaneous Localisation and Mapping of a Robotic Vehicle using a Probabilistic Graphical Model*. Stellenbosch University, 2016.

[6] Choset, H.M., Hutchinson, S., Lynch, K.M., Kantor, G., Burgard, W., Kavraki, L.E. and Thrun, S.: *Principles of robot motion: theory, algorithms, and implementation*. MIT press, 2005.

[7] Barber, D.: *Bayesian reasoning and machine learning*. Cambridge University Press, 2012.

[8] Duchi, J.: Derivations for linear algebra and optimization. *Berkeley, California*, vol. 3, 2007.