UNIVERSITEIT•STELLENBOSCH•UNIVERSITY
jou kennisvennoot • your knowledge partner

Numerical integration for probabilisitc reasoning skripsie

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
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
$\boldsymbol{x}^T$	-	transpose of vector
X	-	matrix
X^T	-	transpose of matrix
X^{-1}	-	inverse of matrix
$\|X\|$	-	determinant 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

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 precise knowledge of their location in order to navigate their surroundings.

When localising a robot, the control input to the robot, and a map of the environment is usually available. Further the robot is 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 location always have noise to some degree.
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 non-linear systems these integrals cannot be calculated analytically and one must use numerical methods. Kalman filters have been extensively used for localisation. Techniques such as the extended Kalman filter use primitive numerical integration that can lead to inaccurate estimations, especially when measurements are also non-linear. 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 random variables and allows to reason about their likelihood based on knowledge that is available to the system. Reasoning about random variables 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 investigate different techniques of numerical integration and to implement each in a PGM to reason about a non-linear robotics 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, as it is very easy to work with. Although Gaussian distributions make severe assumptions such as 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 as all probability distributions are approximated as Gaussian distributions. Key concepts, features and representations of the Gaussian distribution are discussed in this chapter. The following chapter is based on the work of Peebles and Shi [1] and Koller and Friedman [2].

2.1 Covariance Form

The value if of a Random variable (RV) is of random nature and drawn from some function. The value of a Gaussian RV is drawn from Gaussian distribution.

The univariate Gaussian distribution is 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 which the distribution decays. The probability to draw of a value X out of a Gaussian distribution decreases exponentially as the value of X moves farther from the mean. The standard deviation of the Gaussian distribution is denoted as σ and is an indication of the spread of the graph.

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

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

The variance of x can be calculated as

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

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

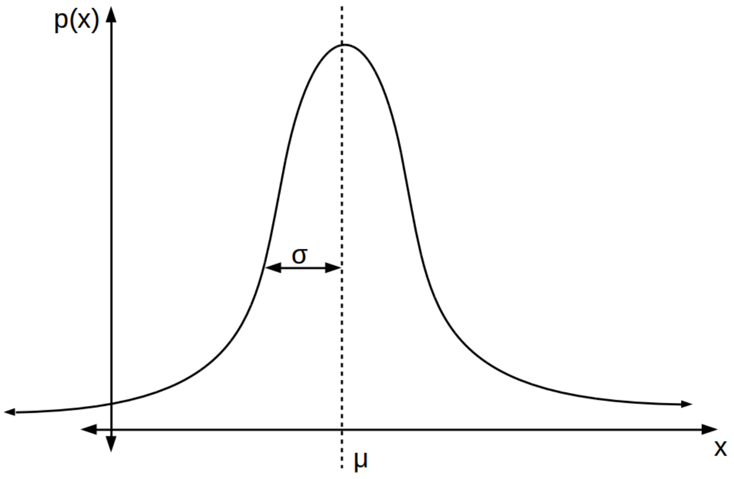

Figure 2.1: Univariate Gaussian PDF

The definition of the Gaussian distribution can be extended to describe the PDF of N random variables, 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. The multivariate Gaussian distribution of a set of N random variables, or random vector

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

is denoted as

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

and it has a PDF that is described by

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

with a normalisation coefficient

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

$|\Sigma|$ is the the determinant of Σ.

For this report it is required that Σ is positive definite and thus also non-singular, which guarantees a determinant that is non-zero. 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

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

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

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

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 each other suggest a steep incline and contours farther apart suggest a gradual incline in the PDF. Hence, small and concentrated contours represent confident Gaussian distributions and contours that are larger and further apart represent uncertain Gaussian distributions. Error ellipsis are an effective way to indicate the mean, uncertainty and correlation of 2D Gaussian distributions.

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 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 eigenvector matrix Q contains an eigenvector $\boldsymbol{v}$. Q is defined as

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

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

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

An error ellipsis 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 ellipsis can be plotted for different confidence levels. The lengths of the major an 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 $\lambda_1 \geq \lambda_2$.

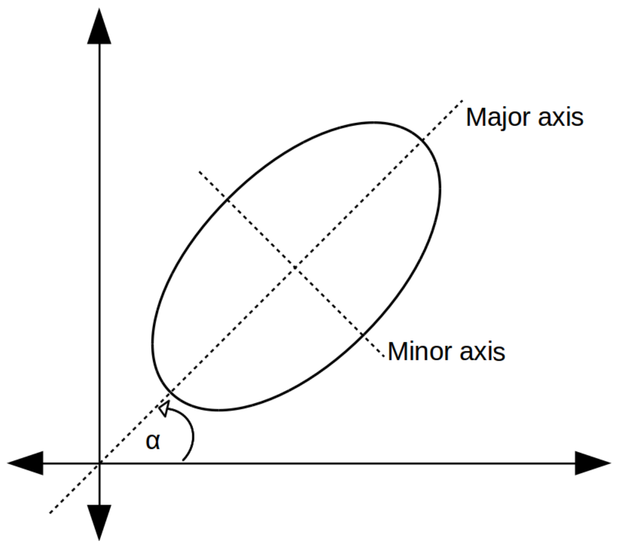

Figure 2.2: Error Ellipse

2.2 Canonical Form

The canonical or information form is an alternative way to describe Gaussian distributions. Using the canonical form has benefits, such as 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. The following section is based on the work of Koller and Friedman [2] and Schoeman [5].

Equation 2.8, which is the definition of a Gaussian PDF, can be rewritten

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

with the information matrix

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

the potential vector

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

and scalar

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

Thus the definition of the canonical form is

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

As one can always determine g from $\boldsymbol{h}$ and K, the canonical form can be described as

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

The covariance parameters can easily be recovered:

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

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

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 an covariance form.

2.2.1 Operations using the Canonical Form

This section will cover useful operations using the canonical form.

Extending and rearranging scopes of canonical forms may be necessary before doing operations, as it is important that 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 K and $\boldsymbol{h}$:

$$\mathcal{C}\left(\begin{bmatrix}\boldsymbol{x}\\ \boldsymbol{y}\end{bmatrix};\begin{bmatrix}K_{\boldsymbol{xx}} & K_{\boldsymbol{xy}}\\ K_{\boldsymbol{yx}} & K_{\boldsymbol{yy}}\end{bmatrix},\begin{bmatrix}\boldsymbol{h_x}\\ \boldsymbol{h_y}\end{bmatrix}\right) = \mathcal{C}\left(\begin{bmatrix}\boldsymbol{x}\\ \boldsymbol{y}\\ \boldsymbol{z}\end{bmatrix};\begin{bmatrix}K_{\boldsymbol{xx}} & K_{\boldsymbol{xy}} & [0]\\ K_{\boldsymbol{yx}} & K_{\boldsymbol{yy}} & [0]\\ [0] & [0] & [0]\end{bmatrix},\begin{bmatrix}\boldsymbol{h_x}\\ \boldsymbol{h_y}\\ \boldsymbol{0}\end{bmatrix}\right) \tag{2.25}$$

Rearranging the Scope of a Canonical Form:

The scope can be rearranged by rearranging the rows and columns of K and rearranging the entries of $\boldsymbol{h}$:

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

Multiplication of Canonical Forms:

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

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

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 $\mathcal{C}(\boldsymbol{x}, \boldsymbol{y}; K, \boldsymbol{h}, g)$ 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

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

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

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

where

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

$$\boldsymbol{h'} = \boldsymbol{h_x} - K_{\boldsymbol{xx}}K_{\boldsymbol{yy}}^{-1}\boldsymbol{h_y} \tag{2.32}$$

and

$$g' = g - \frac{1}{2}\left(\ln|2\pi K_{\boldsymbol{yy}}^{-1}| + \boldsymbol{h_y}^T K_{\boldsymbol{yy}}^{-1}\boldsymbol{h_y}\right). \tag{2.33}$$

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

Reduction with Evidence:

Let $\mathcal{C}(\boldsymbol{x},\boldsymbol{y};K,\boldsymbol{h},g)$ 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 $\mathcal{C}(\boldsymbol{x};K',\boldsymbol{h'},g')$, where

$$K' = K_{\boldsymbol{xx}}, \tag{2.34}$$

$$\boldsymbol{h'} = \boldsymbol{h_x} - K_{\boldsymbol{xy}}\boldsymbol{Y}, \tag{2.35}$$

and

$$g' = g + \boldsymbol{h_y}^T\boldsymbol{Y} - \frac{1}{2}\boldsymbol{Y}^T K_{\boldsymbol{yy}}\boldsymbol{Y}. \tag{2.36}$$

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

2.2.2 Linear Transformations using the Canoncical Form

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

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

$$p(\boldsymbol{y}|\boldsymbol{x}) = \frac{p(\boldsymbol{y},\boldsymbol{x})}{p(\boldsymbol{x})}. \tag{2.37}$$

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

$$\boldsymbol{y} = F\boldsymbol{x} + \boldsymbol{g} + \boldsymbol{n}, \tag{2.38}$$

where

$$\boldsymbol{n} \sim \mathcal{N}(\boldsymbol{n};\boldsymbol{0},\Sigma_n). \tag{2.39}$$

The probability distribution is

$$\begin{aligned} p(\boldsymbol{y}|\boldsymbol{x}) &= \mathcal{N}(\boldsymbol{y};F\boldsymbol{x}+\boldsymbol{g},\Sigma) \\ &= \eta \exp\left[-\frac{1}{2}(\boldsymbol{y}-(F\boldsymbol{x}+\boldsymbol{g}))^T\Sigma^{-1}(\boldsymbol{y}-(F\boldsymbol{x}+\boldsymbol{g}))\right]. \end{aligned} \tag{2.40}$$

A part of Equation 2.40 can be rewritten as

$$\begin{aligned}\boldsymbol{y} - (F\boldsymbol{x} + \boldsymbol{g}) &= \begin{bmatrix} I & -F & -I \end{bmatrix} \begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{x} \\ \boldsymbol{g} \end{bmatrix} \\ &= \begin{bmatrix} I \\ -F^T \\ -I \end{bmatrix}^T \begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{x} \\ \boldsymbol{g} \end{bmatrix}.\end{aligned} \tag{2.41}$$

Substituting Equation 2.41 into Equation 2.40 gives

$$p(\boldsymbol{y}|\boldsymbol{x}) = \eta \exp \left[-\frac{1}{2} \begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{x} \\ \boldsymbol{g} \end{bmatrix}^T \begin{bmatrix} I \\ -F^T \\ -I \end{bmatrix} \Sigma^{-1} \begin{bmatrix} I \\ -F^T \\ -I \end{bmatrix}^T \begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{x} \\ \boldsymbol{g} \end{bmatrix} \right]. \tag{2.42}$$

Defining a combined random vector

$$\boldsymbol{w} = \begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{x} \\ \boldsymbol{g} \end{bmatrix} \tag{2.43}$$

and information matrix

$$K' = \begin{bmatrix} I \\ -F^T \\ -I \end{bmatrix} \Sigma^{-1} \begin{bmatrix} I \\ -F^T \\ -I \end{bmatrix}^T. \tag{2.44}$$

We can substitute Equation 2.43 and Equation 2.44 into Equation 2.42 which gives

$$p(\boldsymbol{y}|\boldsymbol{x}) = \eta \exp \left[-\frac{1}{2} \boldsymbol{w}^T K' \boldsymbol{w} \right]. \tag{2.45}$$

We can now use Equation 2.21 to write the conditional distribution of Equation 2.45 in the canonical form as follows

$$\begin{aligned} p(\boldsymbol{y}|\boldsymbol{x}) &\propto \mathcal{C}(\boldsymbol{w} : K', \mathbf{0}) \\ &= \mathcal{C}\left(\begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{x} \\ \boldsymbol{g} \end{bmatrix}; \begin{bmatrix} I \\ -F^T \\ -I \end{bmatrix} \Sigma_n^{-1} \begin{bmatrix} I & -F & -I \end{bmatrix}, \mathbf{0}, - \right) \\ &= \mathcal{C}\left(\begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{x} \end{bmatrix}; \begin{bmatrix} I \\ -F^T \end{bmatrix} \Sigma_n^{-1} \begin{bmatrix} I & -F \end{bmatrix}, \begin{bmatrix} I \\ -F^T \end{bmatrix} \Sigma_n^{-1} \boldsymbol{g}, - \right) \\ &= \mathcal{C}\left(\begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{x} \end{bmatrix}; \begin{bmatrix} \Sigma_n^{-1} & -\Sigma_n^{-1} F \\ -F^T \Sigma_n^{-1} & F^T \Sigma_n^{-1} F \end{bmatrix}, \begin{bmatrix} -\Sigma_n^{-1} \boldsymbol{g} \\ -F^T \Sigma_n^{-1} \boldsymbol{g} \end{bmatrix}, - \right). \end{aligned} \tag{2.46}$$

This representation of conditional probability densities in the canonical form will be very useful in future chapters.

Chapter 3

Motion Models

Before localising a robot, it is important to model the way the robot moves. A simple linear motion model will be covered which will later be used to illustrate basic principles when localising linear moving robots. In reality the movement of a robot is rarely linear, therefore a more complex non-linear motion model will be explored. This chapter is based on the work of Thrun, Burgard and Fox [6].

3.1 Basic Concepts

The motion model describes $p(\boldsymbol{x}_t|\boldsymbol{u}_t, \boldsymbol{x}_{t-1})$, which is the transition probability and it is vital in the the *prediction step* of the Bayes filter, discussed in Chapter 5. The material will cover kinematics in a stochastic form. Robot motion will be limited to planar movement as it is easier to convey concepts or ideas to the reader. Note that the theory discussed is not limited to planar movement and can be easily extended to three dimensions.

The kinematic state of a robot moving in a plane can be described by three variables which is referred to as the pose of the robot. The pose is defined as

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

As shown in Figure 3.1, x_t is the x-coordinate, y_t is the y-coordinate and θ_t is the orientation of the robot at time t.

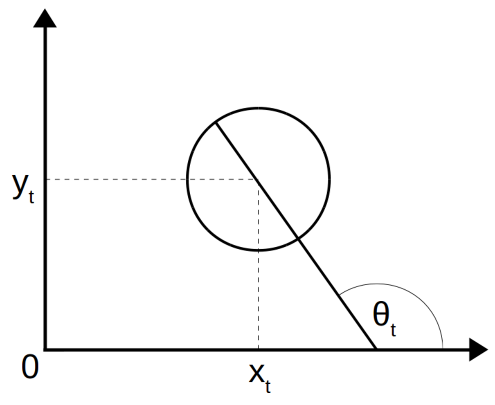

Figure 3.1: Robot pose

3.2 Linear Motion Model

This section describes the linear motion model. The parameter θ_t is omitted from the pose and therefore the kinematic state of the robot is described by only a location

$$\boldsymbol{y_t} = \begin{bmatrix} x_t \\ y_t \end{bmatrix}. \tag{3.2}$$

The motion 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

$$\boldsymbol{u_t} = \begin{bmatrix} v_{x,t} \\ v_{y,t} \end{bmatrix}, \tag{3.3}$$

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 $\boldsymbol{y_t}$ of the robot at time t can be described as linear function in the form of

$$\boldsymbol{y_t} = A\boldsymbol{y_{t-1}} + B\boldsymbol{u_t} + \boldsymbol{\varepsilon}, \tag{3.4}$$

where

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

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

$$\boldsymbol{\varepsilon} = \begin{bmatrix} \varepsilon_x \\ \varepsilon_y \end{bmatrix}, \tag{3.7}$$

and the noise is described as Gaussian distributions

$$\varepsilon_x \sim \mathcal{N}(\varepsilon_x; 0, \sigma_x^2), \tag{3.8}$$

$$\varepsilon_y \sim \mathcal{N}(\varepsilon_y; 0, \sigma_y^2). \tag{3.9}$$

The variances σ_x^2 and σ_y^2 is determined empirically and Δt is the length of a time step.

3.3 Non-linear Motion Model

Two non-linear motion models were considered for this project, namely the velocity and odometry model. The odometry motion model uses sensor measurements to describe a robot's movement over time. Odometry models can only be used for retrospect after a robot moved and can not be used for motion planning. The velocity model is 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 widely used. As the velocity motion model was sufficient for the goal this project, it was the motion model of choice.

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

$$\boldsymbol{u_t} = \begin{bmatrix} v_t \\ \omega_t \end{bmatrix}. \tag{3.10}$$

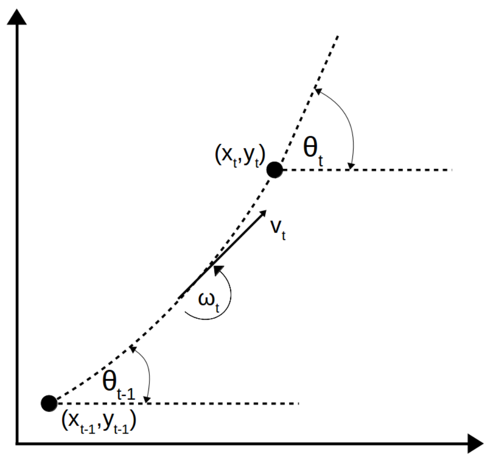

Figure 3.2: Velocity motion model

Positive rotational velocities are defined to be counter-clockwise and positive translational velocities are specified to be in the "forward" direction. The pose at time t can be determined

$$\begin{bmatrix} x_t \\ y_t \\ \theta_t \end{bmatrix} = \begin{bmatrix} x_{t-1} - \frac{v_t}{\omega_t} \sin\theta_{t-1} + \frac{v_t}{\omega_t} \sin(\theta_{t-1} + \omega_t \Delta t) \\ y_{t-1} + \frac{v_t}{\omega_t} \cos\theta_{t-1} - \frac{v_t}{\omega_t} \cos(\theta_{t-1} + \omega_t \Delta t) \\ \theta_{t-1} + \omega_t \Delta t \end{bmatrix} + \begin{bmatrix} \varepsilon_x \\ \varepsilon_y \\ \varepsilon_\theta \end{bmatrix}, \tag{3.11}$$

where

$$\varepsilon_x \sim \mathcal{N}(\varepsilon_x; 0, {\sigma_x}^2), \tag{3.12}$$

$$\varepsilon_y \sim \mathcal{N}(\varepsilon_y; 0, {\sigma_y}^2) \tag{3.13}$$

and

$$\varepsilon_\theta \sim \mathcal{N}(\varepsilon_\theta; 0, {\sigma_\theta}^2). \tag{3.14}$$

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

The model is non-linear due to the fact due to the fact that the movement of the robot cannot be described by a linear translation in the form of

$$\boldsymbol{y} = A\boldsymbol{x} + B. \tag{3.15}$$

Chapter 4

Localisation using Kalman Filters

This chapter will focus on localisation using traditional techniques such as the Kalman filter, extended Kalman filter and unscented Kalman filter. This is important to investigate and understand as there is a strong correspondence to the PGM technique that will be discussed in a following section.

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 truly know its pose. A robot can infer a belief or have internal knowledge of its state from information gathered from sensors. There is thus a difference between internal belief and true state.

In this report the belief is a probability density function of a state variable $\boldsymbol{x}_t$ given all the previous measurements $\boldsymbol{z}_{1:t}$ and controls $\boldsymbol{u}_{1:t}$ and is also known as the posterior. 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 posterior $bel(\boldsymbol{x}_t)$ is usually calculated from a *predicted belief* $\overline{bel}(\boldsymbol{x}_t)$, this calculation is known as the *update step.*

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 PDF of the state variable $\boldsymbol{x}_t$ without taking the measurement $\boldsymbol{z}_t$ in to consideration and it is known as the *prediction step.*

4.2 Bayes Filter

The *Bayes Filter* algorithm is the cornerstone for determining beliefs and filters such as the Kalman filter is a variation of it. The pseudo-algorithm for the Bayes filter is shown in Algorithm 1. One can observe that the algorithm is recursive as it takes the previous belief $\boldsymbol{x}_{t-1}$ at time $t-1$ as an argument to estimate and return the belief $\boldsymbol{x}_t$ at time t.

The algorithm takes three parameters, namely the previous belief $bel(\boldsymbol{x}_{t-1})$, the latest control input $\boldsymbol{u}_t$ and the measurements $\boldsymbol{z}_t$. The pseudo-algorithm shows one iteration of the algorithm that consists out of two steps, namely the *prediction step* in line 3, and *update step* in line 4.

The *prediction step* calculates a predicted belief of the state $\boldsymbol{x}_t$ based on the previous state $\boldsymbol{x}_{t-1}$ and the control input $\boldsymbol{u}_t$. In this step the marginal distribution over $\boldsymbol{x}_{t-1}$ is calculated of the product of two distributions: the prior belief $bel(\boldsymbol{x}_{t-1})$, and the transition probability $p(\boldsymbol{x}_t|\boldsymbol{u}_t, \boldsymbol{x}_{t-1})$.

In the *update step* the final belief of state $\boldsymbol{x}_t$ is calculated by multiplying the predicted belief $\overline{bel}(\boldsymbol{x}_t)$ with the probability of $\boldsymbol{z}_t$ given $\boldsymbol{x}_t$. The result usually does not integrate to one, hence it is usually not a valid PDF and has to be normalized by multiplying it with η. The final belief $bel(\boldsymbol{x}_t)$ 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 implementation of the Bayes filter and is used for prediction in linear Gaussian systems. The Kalman filter is used for belief estimation of systems with continuous states and is not applicable in discrete state spaces.

4.3.1 Description of Kalman Filter

The Kalman filter makes three assumptions in addition to the Markov assumptions which have the outcome that posterior beliefs calculated with the Kalman filter are Gaussian distributions.

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

$$\boldsymbol{y}_t = A\boldsymbol{y}_{t-1} + B\boldsymbol{u}_t + \boldsymbol{\varepsilon}, \tag{4.4}$$

with

$$\boldsymbol{\varepsilon} \sim \mathcal{N}(\boldsymbol{\varepsilon}; \boldsymbol{0}, R_t) \tag{4.5}$$

therefore the transition PDF $p(x_t|u_t, x_{t-1})$ is linear its arguments.

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

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

with

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

3. The initial belief $bel(\boldsymbol{x}_0)$ of the system must be a Gaussian distribution.

The algorithm for the Kalman filter is shown in Algorithm 2 and it is very similar to the Bayes filter in Algorithm 1, as it is a variation of it. Line 1 and 2 of the Kalman filter is the *prediction step* of the Bayes filter. The predicted mean $\overline{\boldsymbol{\mu}}_t$ and predicted covariance $\overline{\Sigma}_t$ is computed without including the measurements z_t and describes the predicted belief $\overline{bel}(x_t)$. The previous mean $\boldsymbol{\mu}_t$ is transformed through the deterministic linear state transition function to determine the predicted mean $\overline{\boldsymbol{\mu}}$. The covariance Σ_{t-1} is multiplied twice with A_t and R_t is added to determine the predicted covariance $\overline{\Sigma_t}$.

Line 4 to 6 corresponds with the *update step* of the Bayes filter and the desired belief $bel(x_t)$ is computed and described by mean $\overline{\boldsymbol{\mu}}_t$ and covariance Σ_t. The Kalman gain K_t is calculated in line 4 and is used to determine how much influence the measurements $\boldsymbol{z}_t$ has when it is incorporated. The difference between the actual measurements $\boldsymbol{z}_t$ and the expected measurements $C_t\overline{\boldsymbol{\mu}}_t$ is called the innovation. In line 5 the predicted mean $\overline{\boldsymbol{\mu}}_t$ is adjusted by summing it with the product of the Kalman gain and innovation. In line 6 the covariance Σ_t is calculated.

Algorithm 2 Kalman Filter

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

4.3.2 Kalman Filter Implementation

For the implementation ten time steps of a robot's movement was simulated with The linear motion model described with Equation 4.4 was used with

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \tag{4.8}$$

$$B = \begin{bmatrix} \Delta t & 0 \\ 0 & \Delta t \end{bmatrix}, \Delta t = 1, \tag{4.9}$$

$$\boldsymbol{\varepsilon} = \begin{bmatrix} \varepsilon_x \\ \varepsilon_y \end{bmatrix}, \tag{4.10}$$

4.4 Extended Kalman Filter

The Kalman filter discussed above is very efficient due to the fact that state transitions and measurements were assumed to be linear, hence them 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 reality rarely linear and therefore the normal Kalman filter is unsuitable for most robotics problems. In this project, only non-linear state transitions were considered and measurements were assumed to be linear.

4.4.1 Description of EKF

The state transition of a non linear system can be described by

$$\boldsymbol{x}_t = \boldsymbol{g}(\boldsymbol{u}_t, \boldsymbol{x}_{t-1}) + \boldsymbol{\varepsilon}_t, \tag{4.11}$$

and non-linear measurements can be described by

$$\boldsymbol{z}_t = \boldsymbol{h}(\boldsymbol{x}_t) + \boldsymbol{\zeta}_t, \tag{4.12}$$

where $\boldsymbol{h}$ and $\boldsymbol{g}$ are non-linear vector functions. A Gaussian RV transformed through a non-linear function results in a non-Gaussian RV, therefore the belief $bel(\boldsymbol{x}_t)$ is no longer Gaussian distributed and cannot be computed analytically.

The extended Kalman filter (EKF) is a variation on the Kalman filter and enables one to approximate a Gaussian distributed belief $bel(\boldsymbol{x}_t)$ of a system with non-linear state transitions or measurements or both. The EKF linearises the functions $\boldsymbol{h}$ and $\boldsymbol{g}$ by means of Taylor expansion and as a result the belief $bel(\boldsymbol{x}_t)$ is approximated as a Gaussian distribution.

In general, with Taylor expansion a non-linear vector function

$$\boldsymbol{f}(\boldsymbol{x}) = \begin{bmatrix} f_1(\boldsymbol{x}) & \dots & f_M(\boldsymbol{x}) \end{bmatrix}^T, \tag{4.13}$$

with

$$\boldsymbol{x} = \begin{bmatrix} x_1 & \dots & x_N \end{bmatrix}^T, \tag{4.14}$$

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

$$\boldsymbol{f}(\boldsymbol{x}) \approx \boldsymbol{f}(\boldsymbol{k}) + \boldsymbol{f}'(\boldsymbol{k})(\boldsymbol{x} - \boldsymbol{k}), \tag{4.15}$$

where the jacobian matrix

$$\boldsymbol{f}'(\boldsymbol{x}) = \frac{\partial \boldsymbol{f}(\boldsymbol{x})}{\partial \boldsymbol{x}} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_N} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_M}{\partial x_1} & \cdots & \frac{\partial f_M}{\partial x_N} \end{bmatrix}. \tag{4.16}$$

Taylor expansion can be applied to linearise the non-linear vector functions $\boldsymbol{g}$ and $\boldsymbol{h}$. Using Equation 4.13, $\boldsymbol{g}$ can be approximated as a linear vector function around $\boldsymbol{\mu}_{t-1}$

$$\begin{aligned} \boldsymbol{g}(\boldsymbol{u_t}, \boldsymbol{x}_{t-1}) &\approx \boldsymbol{g}(\boldsymbol{u}_t, \boldsymbol{\mu}_{t-1}) + \boldsymbol{g}'(\boldsymbol{u}_t, \boldsymbol{\mu}_{t-1})(\boldsymbol{x}_{t-1} - \boldsymbol{\mu}_{t-1}) \\ &= \boldsymbol{g}(\boldsymbol{u}_t, \boldsymbol{\mu}_{t-1}) + G_t(\boldsymbol{x}_{t-1} - \boldsymbol{\mu}_{t-1}), \end{aligned} \tag{4.17}$$

where G_t is the jacobian matrix and can be determined by using Equation 4.16, therefore

$$G_t = \boldsymbol{g}'(\boldsymbol{u}_t, \boldsymbol{\mu}_{t-1}) = \frac{\partial \boldsymbol{g}(\boldsymbol{u}_t, \boldsymbol{x}_{t-1})}{\partial \boldsymbol{x}_{t-1}}. \tag{4.18}$$

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

$$\begin{aligned} \boldsymbol{h}(\boldsymbol{x}_t) &\approx \boldsymbol{h}(\overline{\boldsymbol{\mu}}_t) + \boldsymbol{h}'(\overline{\boldsymbol{u}}_t)(\boldsymbol{x}_t - \overline{\boldsymbol{\mu}}_t) \\ &= \boldsymbol{h}(\overline{\boldsymbol{\mu}}_t) + H_t(\boldsymbol{x}_t - \overline{\boldsymbol{\mu}}_t), \end{aligned} \tag{4.19}$$

where

$$H_t = \boldsymbol{h}'(\boldsymbol{u}_t) = \frac{\partial \boldsymbol{h}(\boldsymbol{x}_t)}{\partial \boldsymbol{x}_t}. \tag{4.20}$$

In Algorithm 3 the procedure of the EKF is shown and one will notice that is very similar to Algorithm 2 of the Kalman filter. The only difference is that the non-linear state transition and measurement functions are linearised and therefore the state belief of the EKF is not exact.

Algorithm 3 Extended Kalman Filter

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

This section covered the basic theory of the EKF, the next section will discuss how the EKF was implemented in Python.

4.4.2 Application of EKF

In this subsection the EKF is applied to localise a robot whose movement is described by the velocity motion model. Measurements of the robot's location were assumed to be linear. A moving robot with measurements were simulated and beliefs of the robot's pose were estimated using the EKF.

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

$$\boldsymbol{x}_t = \begin{bmatrix} x_t \\ y_t \\ \theta_t \end{bmatrix} = \boldsymbol{g}(\boldsymbol{u}_t, \boldsymbol{x}_{t-1}) + \boldsymbol{\varepsilon}, \tag{4.21}$$

where the velocity motion model describes $\boldsymbol{g}$ as

$$\boldsymbol{g}(\boldsymbol{u}_t, \boldsymbol{x}_{t-1}) = \begin{bmatrix} x_{t-1} - \frac{v_t}{\omega_t}\sin\theta_{t-1} + \frac{v_t}{\omega_t}\sin(\theta_{t-1} + \omega_t \Delta t) \\ y_{t-1} + \frac{v_t}{\omega_t}\cos\theta_{t-1} - \frac{v_t}{\omega_t}\cos(\theta_{t-1} + \omega_t \Delta t) \\ \theta_{t-1} + \omega_t \Delta t \end{bmatrix}. \tag{4.22}$$

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

$$\boldsymbol{\varepsilon} \sim \mathcal{N}(\boldsymbol{\varepsilon}; \boldsymbol{0}; R), \tag{4.23}$$

where the covariance matrix R is defined as

$$R = \begin{bmatrix} \sigma_x^2 & 0 & 0 \\ 0 & \sigma_y^2 & 0 \\ 0 & 0 & \sigma_\theta^2 \end{bmatrix}. \tag{4.24}$$

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

The next step is to linearise the non-linear vector function $\boldsymbol{g}$ by means of Taylor expansion. Equation 4.16 and Equation 4.18 can be used to calculate the jacobian matrix G_t of $\boldsymbol{g}$:

$$G_t = \boldsymbol{g}'(\boldsymbol{u}_t, \boldsymbol{x}_{t-1}) = \begin{bmatrix} 1 & 0 & -\frac{v_t}{\omega_t}\cos\theta + \frac{v_t}{\omega_t}\cos(\theta + \omega_t\Delta t) \\ 0 & 1 & -\frac{v_t}{\omega_t}\sin\theta + \frac{v_t}{\omega_t}\sin(\theta + \omega_t\Delta t) \\ 0 & 0 & 1 \end{bmatrix}. \tag{4.25}$$

Using Equation 4.17 and the jacobian matrix G_t calculated in Equation 4.25, the vector function $\boldsymbol{g}$ can be linearised around $\boldsymbol{u_{t-1}}$, thus

$$\boldsymbol{g}(\boldsymbol{u_t}, \boldsymbol{x}_{t-1}) \approx \boldsymbol{g}(\boldsymbol{u_t}, \boldsymbol{\mu}_{t-1}) + G_t(\boldsymbol{x}_{t-1} - \boldsymbol{\mu}_{t-1}). \tag{4.26}$$

Using the result in Equation 4.26, the state transition in equation can be approximated as

$$\boldsymbol{x}_t \approx \boldsymbol{g}(\boldsymbol{u}_t, \boldsymbol{\mu}_{t-1}) + G_t(\boldsymbol{x}_{t-1} - \boldsymbol{\mu}_{t-1}) + \boldsymbol{\varepsilon}. \tag{4.27}$$

Linear measurements were assumed for the implementation and therefore

$$\boldsymbol{z}_t = C_t\boldsymbol{x}_t + \boldsymbol{\zeta}_t. \tag{4.28}$$

For the simulation the robot was given ten different control inputs $\boldsymbol{u}_t$ and Δt was set to one.

4.5 Unscented Kalman Filter

The linearisation used in the EKF is rudimentary, as non-linear functions are linearised only around the mean which, in some scenarios, can lead to very inaccurate approximations. The unscented Kalman filter (UKF) is an alternative approach which in many cases deliver better results than that of the EKF, especially when $\boldsymbol{g}$ and $\boldsymbol{h}$ is highly non-linear.

As mentioned before, a Gaussian distribution propagated through a non-linear function results in a distribution that is non-Gaussian. The principle the UKF follows is to draw carefully chosen points called *sigma points* around the mean from an existing Gaussian distribution and transform it through a non-linear function. The transformed sigma points can then be used to approximate the resultant distribution as a Gaussian distribution.

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

$$\mathcal{X}^{[i]},\ i\ =\ 1,\ ...,\ 2n, \tag{4.29}$$

where i indicates the index of the sigma point.

Sigma points are determined as follows:

$$\mathcal{X}^{[0]} = \boldsymbol{\mu}, \tag{4.30}$$

$$\mathcal{X}^{[i]} = \boldsymbol{\mu} + \left(\sqrt{(n+\lambda)\Sigma}\right) \quad \text{for} \quad i = 1, ..., n, \tag{4.31}$$

$$\mathcal{X}^{[i]} = \boldsymbol{\mu} - \left(\sqrt{(n+\lambda)\Sigma}\right) \quad \text{for} \quad i = n+1, ..., 2n, \tag{4.32}$$

where the scaling parameter λ is defined as

$$\lambda = \alpha^2(n + \kappa) - n. \tag{4.33}$$

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 $w_c^{[i]}$, and is respectively used to recover the mean $\boldsymbol{\mu}_t$ and covariance Σ_t. Weights corresponding to $\mathcal{X}^{[0]}$, the mean $\boldsymbol{\mu}$ are determined as

$$w_m^{[0]} = \frac{\lambda}{n+\lambda}, \tag{4.34}$$

$$w_c^{[0]} = \frac{\lambda}{n+\lambda} + (1 - \alpha^2 + \beta). \tag{4.35}$$

The parameter the optimal choice for the parameter β is usually 2. The rest of the weights where $i = 1$, ..., $2n$ are determined as

$$w_m^{[i]} = w_c^{[i]} = \frac{1}{2(n+\lambda)} \quad \text{for } i = 1, \ ..., \ 2n. \tag{4.36}$$

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

$$\mathcal{Y}^{[i]} = \boldsymbol{f}\left(\mathcal{X}^{[i]}\right). \tag{4.37}$$

The distribution as a result of the non-linear transformation can be approximated as Gaussian distribution by estimating the mean $\boldsymbol{u}'$ and covariance Σ' with

$$\boldsymbol{\mu}' = \sum_{i=0}^{2n} w_m^{[i]} \mathcal{Y}^{[i]}, \tag{4.38}$$

$$\Sigma' = \sum_{i=0}^{2n} w_c^{[i]} \left(\mathcal{Y}^{[i]}\right) \tag{4.39}$$

The general theory of the UKF was covered in this section, the next section will discuss the implementation of the UKF.

4.5.1 Application of UKF

Algorithm 4 Unscented Kalman Filter

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

Chapter 5

Probabilistic Graphical Models

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

5.1 Overview

Reasoning about systems with uncertainty can become very complex and sometimes it is completely unmanageable. The PGM is a graphical technique which make problems tractable by modelling probabilistic systems in a logical and compact manner. PGMs are thus used to describe the relationship between variables and allows to reason about it. This action of querying 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 making medical diagnosis to localising robots.

PGMs used for modelling can be divided in to two categories. The first is Markov networks, which are used for non-causal systems. The second is Bayesian networks, where relationships between random variables are causal and specified as CPDs. As the localisation problem is in most cases 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. Markov networks are undirected graphs, thus all edges are undirected. Bayesian networks are directed graphs, thus all 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 5.1, 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 5.2, a is a parent of b and b is a child of a. [7]

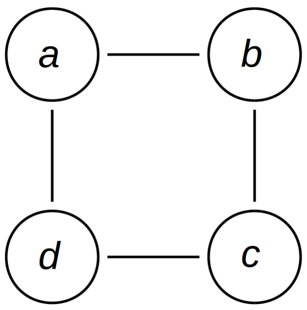

Figure 5.1: Directed graph

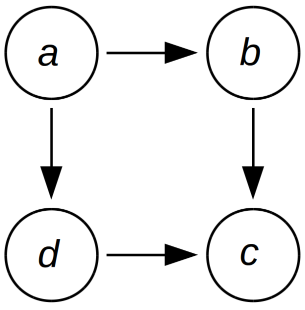

Figure 5.2: Undirected graph

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.

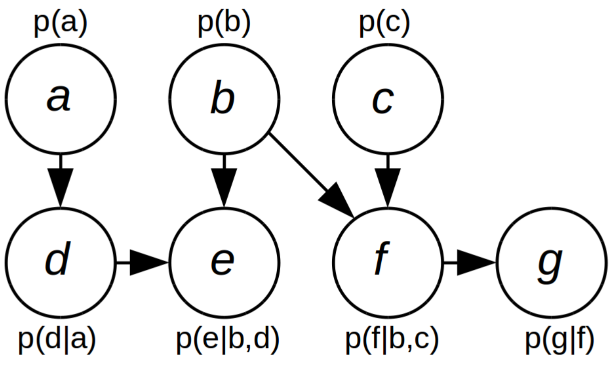

Figure 5.3: Bayesian network

The chain rule specifies that the joint probability density distribution of all the variables in a Bayesian network can be found by finding the product of all the CPDs associated with nodes [2] and 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 found 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 inference a Bayesian network, but it can be very tedious. The next section covers an alternative method to inference Bayesian networks 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 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 with a subset of variables that is 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 ϕ_k in a Bayesian network with a set of factors Φ, there exists a cluster C_i which accommodates ϕ_k

$$\text{Scope}[\phi_k] \subseteq C_i. \tag{5.9}$$

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 in 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.

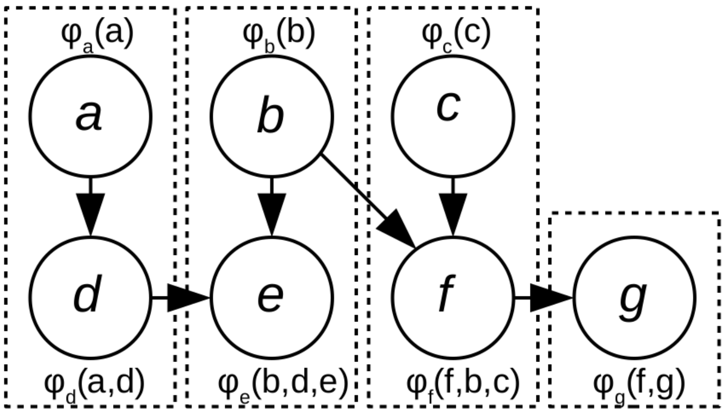

Figure 5.4: Bayesian network with cluster boundaries

The initial belief of a cluster can be calculated by finding the product of all the factors inside the cluster [2]. The initial belief of a cluster is defined as

$$\psi_i(C_i) = \prod_k \phi_k. \tag{5.10}$$

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.

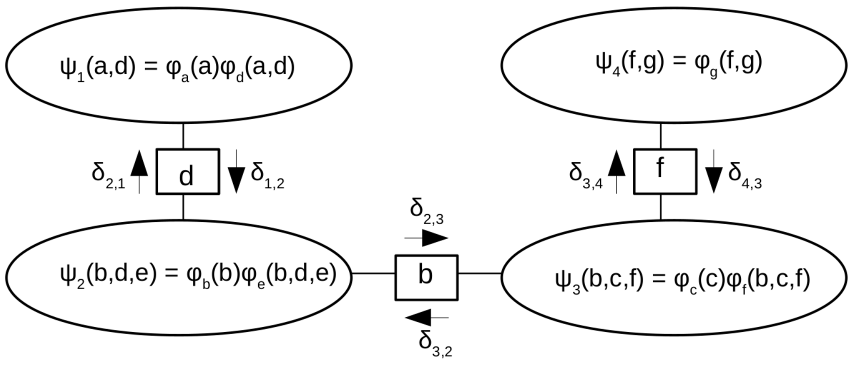

Figure 5.5: Cluster graph showing sepsets and clusters with initial beliefs

This section showed 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 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 initial cluster belief ψ_i with all the incoming messages $\delta_{k,j}$ from other clusters C_k and then marginalising onto the variables contained in the sepset $S_{i,j}$, thus integrating over variables not in the sepset ie. $C_i - S_{i,j}$. This procedure is defined as [2]

$$\delta_{i,j}(S_{i,j}) = \int_{C_i - S_{i,j}} \psi_i \times \prod_{k \neq j} \delta_{k,i}(S_{k,i}). \tag{5.11}$$

Equation 5.11 can be demonstrated by calculating arbitrary messages $\delta_{2,3}$ and in $\delta_{3,4}$ Figure 5.5, therefore

$$\delta_{2,3}(b) = \int_d \int_e \delta_{1,2}(d)\psi_2(b, d, e)dd\ de \tag{5.12}$$

and

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

Sometimes the value of a 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 any more. 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.

In Equation 5.14, an outgoing message $\delta_{i,j}(y)$ is calculated. Evidence of x is available and is indicated as X. The evidence of x is used to reduce the cluster ψ_i before multiplying it with all incoming messages and integrating over z.

$$\delta_{i,j}(y) = \int_z \psi_i(x = X, y, z) \times \prod_{k \neq j} \delta_{k,i}(S_{k,i}) \tag{5.14}$$

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

$$\beta_i(C_i) \propto \psi_i \times \prod_k \delta_{k,i}(S_{k,i}). \tag{5.15}$$

This chapter covered the properties of the Bayesian network and how one can reason about a Bayesian network by constructing a cluster graph. The Bayesian network will be used in the next chapter to model the uncertainty of the location of a robot.

Chapter 6

Localisation using Probabilistic Graphical Models

In this chapter the localisation problem will be solved using PGMs. There is a

6.1 Modelling the Localisation problem

In this section the localisation problem is modelled using a Bayesian network. The network is then transformed into a cluster graph in order to reason about the belief of the pose $\boldsymbol{x}_t$ at time t.

The localisation problem is modelled with the Bayesian network and is shown in Figure. The belief of the robot's initial state is given to the network as $p(\boldsymbol{x}_0)$. The state of the robots 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 current state $\boldsymbol{x}_t$ of the robot.

As discussed in Section, the CPD associated with each node can be written as a factor. The Bayesian network in Figure is converted to a cluster graph shown in Figure. There are a few ways to group the factors to form clusters. In this cluster graph factors ϕ_{u_t}, ϕ_{x_t} and ϕ_{z_t} were grouped together in cluster C_t. The initial belief of a cluster C_t can be determined as

$$\psi_t(x_t, x_{t-1}, u_t, z_t) = \phi_{x_t}(x_t, u_t, x_{t-1})\phi_{z_t}(z_t, x_t)\phi_{u_t}(u_t). \tag{6.1}$$

Sepsets with common variables between cluster are indicated with rectangles.

Only messages in the "forward" direction were calculated and the technique called *smoothing* was not used. Smoothing is a technique where messages are sent back in time to get better estimates of previous beliefs.

6.2 Non-linear localisation

6.2.1 Taylor series expansion

6.2.2 Unscented transform

6.2.3 Monte Carlo integration

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.