

Institutionen för systemteknik

Department of Electrical Engineering

Examensarbete

Localization algorithms for indoor UAVs

Examensarbete utfört i Reglerteknik
vid Tekniska högskolan vid Linköpings universitet
av

Daniel Barac

LiTH-ISY-EX--11/4526--SE

Linköping 2011



Linköpings universitet
TEKNISKA HÖGSKOLAN

Localization algorithms for indoor UAVs

Examensarbete utfört i Reglerteknik
vid Tekniska högskolan i Linköping
av


Daniel Barac

LiTH-ISY-EX--11/4526--SE

Handledare: **Martin Skoglund**
isy, Linköpings universitet
Piotr Rudol
IDA, Linköpings universitet
Mariusz Wzorek
IDA, Linköpings universitet

Examinator: **Thomas Schön**
isy, Linköpings universitet

Linköping, 28 October, 2011

	Avdelning, Institution Division, Department Division of Automatic Control Department of Electrical Engineering Linköpings universitet SE-581 83 Linköping, Sweden	Datum Date 2011-10-28
Språk Language <input type="checkbox"/> Svenska/Swedish <input checked="" type="checkbox"/> Engelska/English <input type="checkbox"/> _____	Rapporttyp Report category <input type="checkbox"/> Licentiatavhandling <input checked="" type="checkbox"/> Examensarbete <input type="checkbox"/> C-uppsats <input type="checkbox"/> D-uppsats <input type="checkbox"/> Övrig rapport <input type="checkbox"/> _____	ISBN _____ ISRN LiTH-isy-ex--11/4526--SE Serietitel och serienummer ISSN Title of series, numbering _____
URL för elektronisk version http://www.control.isy.liu.se http://www.ep.liu.se		
Titel Lokaliseringsalgoritmer för inomhus UAv Title Localization algorithms for indoor UAVs Författare Daniel Barac Author		
Sammanfattning Abstract <p>The increased market for navigation, localization and mapping system has encouraged the research to dig deeper into these new and challenging areas. The remarkable development of computer soft- and hardware have also opened up many new doors. Things which more or less where impossible ten years ago are now reality.</p> <p>The possibilities of using a mathematical approach to compensate for the need of expensive sensors has been one of the main objectives in this thesis. Here you will find the basic principles of localization of indoor UAVs using particle filter (PF) and Octomaps, but also the procedures of implementing 2D scanmatching algorithms and quaternions. The performance of the algorithms is evaluated using a high precision motion capture system. The UAV which forms the basis for this thesis is equipped with a 2D laser and an inertial measurement unit (IMU). The results show that it is possible to perform localization in 2D with centimetre precision only by using information from a laser and a predefined Octomap.</p>		
Nyckelord Keywords UAV, particle filter, Octomap, localization		

Abstract

The increased market for navigation, localization and mapping system has encouraged the research to dig deeper into these new and challenging areas. The remarkable development of computer soft- and hardware have also opened up many new doors. Things which more or less where impossible ten years ago are now reality.

The possibilities of using a mathematical approach to compensate for the need of expensive sensors has been one of the main objectives in this thesis. Here you will find the basic principles of localization of indoor UAVs using particle filter (PF) and Octomaps, but also the procedures of implementing 2D scanmatching algorithms and quaternions. The performance of the algorithms is evaluated using a high precision motion capture system. The UAV which forms the basis for this thesis is equipped with a 2D laser and an inertial measurement unit (IMU). The results show that it is possible to perform localization in 2D with centimetre precision only by using information from a laser and a predefined Octomap.

Sammanfattning

Den ökade marknaden för navigering, lokalisering och kartläggningssystem har uppmuntrat forskningen att gräva djupare inom detta nya utmanande område. Den uppseendeväckande utvecklingen inom mjuk- och hårdvara inom datorteknik har också bidragit till att öppna upp många nya dörrar. Saker som mer eller mindre var omöjliga för tio år sedan är nu verklighet.

Möjligheterna att använda en matematisk metod för att kompensera för behovet av dyra sensorer har varit ett av huvudmålen i examensarbetet. I denna rapport finner du de grundläggande principerna för lokalisering av inomhus-UAVer med hjälp av partikelfilter (PF) och Octomaps, men också metoder för att implementera 2D scanmatching och quaternioner. Prestandan på algoritmerna kommer att verifieras med hjälp av ett motion capture system. UAV:n som ligger till grund för detta examensarbete är utrustad med en 2D laser och en inertial measurement unit (IMU). Resultaten visar att det är möjligt att utföra lokalisering i 2D med centimeter-noggrannhet enbart med information från en laser och en fördefinierad Octomap.

Acknowledgments

First of all I would like to thank my examiner and mentor Dr Thomas Schön for introducing me to this master's thesis but also for being a source of inspiration when stuff was getting tricky and time-consuming. My supervisor Lic Martin Skoglund deserves a special thank for always being enthusiastic, supportive and available for discussion regarding my sometimes stupid questions. Thank you for proofreading the thesis. I would also like to give a special thanks to Lic Piotr Rudol who always found time to support me with experimental sessions in the VICON lab although he was quite busy.

Lic Mariusz Wzorek, Lic Karl Granström and Lic Fredrik Lindsten thank you for helping me with both practical and theoretical matters. MSc Niklas Forslöv, thanks for hours of interesting discussions regarding your and especially my master's thesis.

To all my classmates from "Elektronikdesign", thank you for a very good time together and for the superb cooperation we always had. I have learned a lot of things from all of you. Keep it up guys and I hope we can get together once in a while although we are quite spread out right now.

My best friends, Åsengänget, you know who you are and there are no one like us. Thank you for being my friends. My five years far away from you have been tough and I have discovered how special you really are. During my studies and throughout the pain in the ass situations, I have thought of all the "Åsendamp" we have done together (Sälen, Konstvandring, Kyrkansgård, etc.) and what we will do next, which always made my day. There are no words to describe what we have together and our friendship will last forever.

Caroline, things did not go as expected in the end, nevertheless we had a good time together and experienced amazing stuff. You have really supported and encouraged me throughout these five years. Thank you very much for being there and you will always be my friend.

Finally, everything I have accomplished throughout my life has only been possible because of my wonderful family who always stood by my side, in both good and bad times. Many appreciating hours of phone-support from my brother, re-

garding everything from simple algebra to the physical interpretation of the real part in the laplace operation but also fun stuff as what shall we do when we caught up next time. My sisters' always positive attitude to life has touched me deeply and my father who does not hesitate a second to help out. I remember with with gratefulness all the times he drove me almost four hours single way to Linköping before he turn around and drive four more hours back to Åsensbruk. Add to this my amazing mother who always helps me with stuff before I even realize it is a problem and all the summer jobs she has introduced me to. You are all invaluable to me and I can not explain how much I love you. Being the youngest child has been a privilege which I have realized in recent years.

Contents

1	Introduction	1
1.1	Background	1
1.2	Problem specification	2
1.3	Related work	2
1.4	Objectives	3
2	State estimation	5
2.1	State space models	5
2.2	The recursive Bayesian solution	7
2.3	The particle filter	9
3	Motion and sensor models	13
3.1	Motion models	13
3.1.1	1D motion	14
3.1.2	Motion in 2D	14
3.1.3	Motion in 3D	15
3.2	Measurement equations	16
3.2.1	The Laser range finder	17
3.2.2	The Gaussian distribution	19
3.2.3	Outliers	20
3.2.4	The student's t-distribution	20
3.2.5	Inertial Measurement Unit	22
4	Mapping and localization	23
4.1	Maps	23
4.2	Grid maps	24
4.3	Binary Bayes filter	25
4.4	The occupancy grid mapping algorithm	27
4.5	Octomap	28
4.5.1	Pointers	30
4.5.2	Quantization	31
4.6	Scanmatching	31
4.6.1	The general cost function	32
4.6.2	The iterative closest point algorithm	33

5	Experimental Results	35
5.1	Simulations	37
5.1.1	Localization in the 1.5D PF	37
5.1.2	Localization in the 2D and 2.5D PF	39
5.2	Evaluation of real data	46
6	Conclusion and future work	51
	Bibliography	53

Chapter 1

Introduction

1.1 Background

April 26, 1986, one of the reactors at Chernobyl Nuclear Power Plant exploded and left a black mark in our history. Employees and people from the rescue team suffered radiation injuries and some of them died because of this. Now, almost 25 years later we all remember the Fukushima nuclear disaster which could have cost many lives since no one had the possibility to enter the building and confirm the situation because of the radiation level. Within a couple of years there will hopefully be a solution to this problem where we can take control over the situation without risking someone's life. The solution could be intelligent robots ready to operate within hazardous or inaccessible indoor environments with limited human interaction but still able to generate useful and important information on the situation, also known as unmanned aerial vehicles (UAVs), see Figure 1.1.

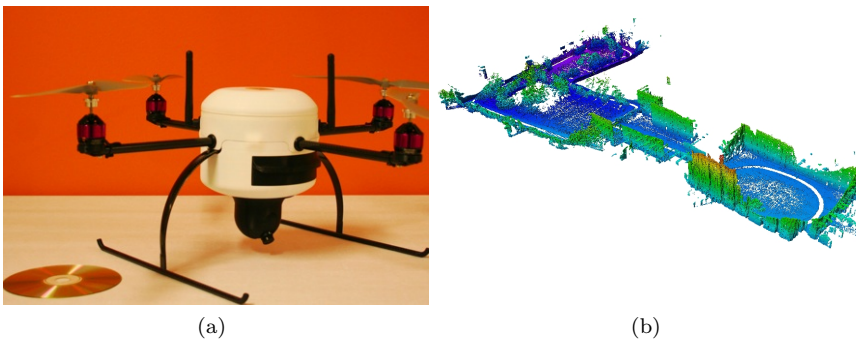


Figure 1.1: A Linkquad developed by UAS Technologies Sweden AB is shown in (a) while an Octomap can be seen in (b).

After many successful years of research in the field of autonomous ground vehicles, the area of UAVs or unmanned aerial system (UAS) have increased worldwide in

a remarkable way.

The possibility of creating and navigating within 3D-maps of hazardous or inaccessible indoor environments can be used in many fields, such as surveillance, search and rescue, etc.

1.2 Problem specification

The development of UAVs has reached far but there are still unexplored areas, especially when it comes to indoor UAVs. Indoor UAVs are often referred to as micro aerial vehicle or MAVs. They are typically very small with low a weight which makes them useful in many different areas. The drawback of using MAVs are their limited payload. Often one has to make use of limited power sources which brings a demand on the energy consumption for the sensors, CPU and other important hardware. Indoor UAVs always operate in GPS-denied and much more restricted environments compared to outdoor UAVs. Manoeuvrability and flexibility will therefore be key properties for indoor UAVs and one can roughly summarize the problems of indoor UAVs in two parts:

- Estimating the position and orientation of the UAV in a reliable way.
- Having full control and manoeuvrability of the UAV.

To fulfill the second statement, one have to know the position of the UAV which implies that the first statements is the main problem to solve for these kind of situations. Estimating the position and orientation does not always has to be a difficult problem when there is useful information available, such as a predefined map. Of course this is not always the case and this is where things get very complex. Think of this; to estimate the position of something you must first have some kind of reference, a map for instance. But to create a reliable map you must know your position, it is like the old saying "which came first, the chicken or the egg?". In state estimation this situation is known as Simultaneous Localization And Mapping (SLAM). To summarize one can simple say that performing SLAM for indoor UAVs in a 3D framework is really state-of-the-art research.

1.3 Related work

In the last ten years there has been a major increase of research in the area of UAVs. Many authors have concentrated on the stability and manoeuvrability of UAVs [2, 8–10] which is a quite huge research area with a lot of improvement possibilities.

Point clouds addresses the problem of representing a map in a three-dimensional coordinate system. In [3, 18] point clouds have been used for ground vehicles to perform 3D-SLAM in outdoor environments. [19] present an other approach for

modeling 3D environments based on octrees using probabilistic occupancy estimation which is known as Octomaps. A similar technique (multi-volume occupancy grid) has also been utilized for indoor UAVs [4], worth to mention is that the authors used a depth camera which the UAV actually did not manage to carry by itself.

Thrun et al. [13] have successfully demonstrated the possibility of using scan-matching algorithms for outdoor 3D Surface Modeling.

Indoor UAVs have a lot of new challenges compared to their outdoor counterpart. Manoeuvrability and flexibility are key properties for indoor UAVs and some publications [1, 5] have successfully proved that there are solutions to these kind of problems, but there is still a lack of research regarding the possibility of creating 3D-maps for indoor environments using UAVs without cameras.

1.4 Objectives

As already mentioned, there are a lot of complex issues to consider when developing indoor UAVs. Unfortunately the time in this master's thesis project is quite limited and a lot of interesting things must be less prioritized or not considered at all.

The UAV which forms the base in this work will be equipped with a 2D laser which can be attached to the UAV in several different ways. To get a complete 3D profile of some environment one can use the 2D laser and move it up- and downwards along a global z -axis if the laser only is sweeping in the xy -plane. Another interesting approach would be to rotate the laser so it will point downwards along the negative z -axis. If the altitude of the laser is high enough, one could simply rotate the laser 180 degrees around the z -axis and a 3D profile of the environment can be distinguished, see Section 5.1.2. Three dimensional scanmatching algorithms have been studied to evaluate the possibility of performing SLAM in 3D. Past experience indicates that a lot of time must be spent on the scanmatching algorithms otherwise there is a high risk it will end up with poor results.

To avoid a situation where the time is running out and the outcome fails, the main purpose will be to investigate the possibility of estimating the position of a UAV for indoor environments by only using a 2D laser. This thesis will therefore only cover the first problem described in the previous section. To further limit the problem, a predefined map is assumed to be available. For this reason, we only have a localization problem and not a full SLAM problem. The objectives of the thesis can be summarized in following way:

- Finding a suitable mathematical model for the UAV.
- Developing a simulation environment to evaluate the algorithms.
- Evaluating the performance of the algorithm using real data.

In the first stage, things will be kept as simple as possible. Hence, the algorithms will first be tested and evaluated in a 2D framework but the 3D situation will always be kept in mind and evaluated if time permits.

Chapter 2

State estimation

All the time in our daily life, humans have to make decision depending on the situation and the environment. For instance, when someone choose to open a closed door there is a lot of information to be processed and questions to be answered before the door can be opened, for instance; Where am I? How far is it to the door? Can I reach the door? The most amazing part in this situation is how the human brain handles everything automatically and translates information from our minds so we can start acting. A robot in the same situation must rely on information from its sensors and make use of mathematical equations to answer all questions before it can interact with the world. Still the robot is never fully aware of the situation because of measurement errors in the sensors, but we can make useful models of how the robot is behaving which is called state estimation.

2.1 State space models

A state space model can be seen as a mathematical representation of a real world system. A general non-linear state space model is given by

$$x_{t+1} = f(x_t, u_t, v_t), \quad (2.1a)$$

$$y_t = h(x_t, u_t, e_t), \quad (2.1b)$$

where x_t denotes the states, y_t denotes the measurements, u_t denotes known inputs to the system (which often are omitted for clarity), v_t and e_t denotes the stochastic process and measurement noise and represents the uncertainty in the dynamic model and the measurements. A special case of (2.1) is often used where the noise is assumed to be additive,

$$x_{t+1} = f(x_t) + v_t, \quad (2.2a)$$

$$y_t = h(x_t) + e_t. \quad (2.2b)$$

Since the process and measurement noise are stochastic, it is convenient to use a probability density function (PDF) when formulating the state space model. The

PDF is a function which describes the probability for a stochastic variable to occur at a given point. The noise is often assumed to be Gaussian (normal) distributed with zero mean and some covariance,

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}, \quad (2.3)$$

where μ is the mean i.e. the location of the peak, while σ^2 denotes the variance which defines the width of the peak, see Figure 2.1. With this in mind a more

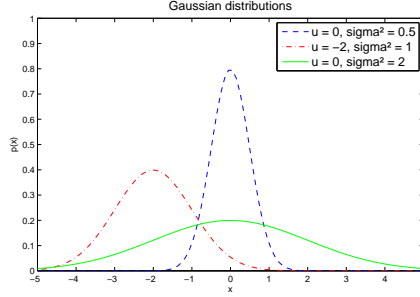


Figure 2.1: A visualization of three different Gaussian distributions.

general formulation of the stochastic state space model can be expressed as a probability density function,

$$x_{t+1} \sim p(x_{t+1}|x_t), \quad (2.4a)$$

$$y_t \sim p(y_t|x_t), \quad (2.4b)$$

which means that the states and measurements have some uncertainty in their model. This is quite intuitive since there is always some noise in the measurements from the sensors and the dynamic model is often a simplification of a much more complex model, but the process noise can also be seen as an unknown input which can change the states of the model.

The stochastic variables in (2.2) are the process and measurement noise, by simple algebra one can express their PDF as:

$$p(x_{t+1}|x_t) = p_{v_t}(x_{t+1} - f(x_t)), \quad (2.5a)$$

$$p(y_t|x_t) = p_{e_t}(y_t - h(x_t)). \quad (2.5b)$$

There are three different problems to consider when the posterior distribution (see explanation below) of the states given the observations are computed or estimated.

- The posterior distribution of the filter solution is denoted $p(x_t|y_{1:t})$ and consider the situation where the states at time t are estimated given the observations from time= 1... t .

- The second problem is called prediction and the posterior distribution is denoted $p(x_{t+m}|y_{1:t})$, $m > 0$. As the name already indicates, prediction handles the situation where the states are estimated what to be in the future.
- Smoothing is the final situation where the posterior is denoted $p(x_{t-m}|y_{1:t})$, $m > 0$, which is often used in off-line application when the future observations are known in advance.

Summarized, one can say that the posterior distribution is defined as the PDF of the states after the information from the last observation have been included in estimation, this step is also known as a measurement update. Throughout this section, the terms distribution and density PDF will be used interchangeably. All three problems can be solved using the Bayesian approach together with the Markov property and marginalization which will be introduced below.

2.2 The recursive Bayesian solution

The filter density $p(x_t|y_{1:t})$ allows us to estimate the position of the UAV in real time. The mathematical solution to this distribution can be derived quite easy when Bayes' rule shown in Theorem 2.1 is applied.

Theorem 2.1 (Bayes' rule) Assume there are three different events A, B and C . The conditional probability for the events are then given by

$$p(A|B, C) = \frac{p(B|A, C) p(A, C)}{p(B|C)}, \quad (2.6a)$$

$$p(A, B|C) = p(A|B, C) p(B|C). \quad (2.6b)$$

where $p(A, B)$ is the same as $p(A \cap B)$ or $p(AB)$, which is the probability that both A and B are satisfied, also known as the joint probability.

Another well-known and useful feature is the Markov property which refers to the situation where future information only depends on the present state and not any past states. For instance, if the present states are known we can discard the old states since the information is already included in the present state

$$p(y_t|x_t, y_{1:t-1}) = p(y_t|x_t). \quad (2.7)$$

Apply Bayes' rule (2.6a) to the posterior filter density

$$p(x_t|y_{1:t}) = p(x_t|y_t, y_{1:t-1}), \quad (2.8)$$

and identify the terms $A = x_t$, $B = y_t$ and $C = y_{1:t-1}$ results in

$$p(x_t|y_{1:t}) = \frac{p(y_t|x_t, y_{1:t-1}) p(x_t|y_{1:t-1})}{p(y_t|y_{1:t-1})}. \quad (2.9)$$

Finally we can make use of the Markov property and the information from $y_{1:t-1}$ can be removed if the state x_t is known,

$$\frac{p(y_t|x_t, y_{1:t-1}) p(x_t|y_{1:t-1})}{p(y_t|y_{1:t-1})} = \frac{p(y_t|x_t) p(x_t|y_{1:t-1})}{p(y_t|y_{1:t-1})}. \quad (2.10)$$

The second expression is known as the measurement update in the Bayesian recursion, where the prediction term $p(x_t|y_{1:t-1})$ is referred to as the one step ahead prediction or the time update. In order to handle the denominator $p(y_t|y_{1:t-1})$ and the one step ahead prediction in (2.10) we have to express them with known density functions which can be accomplished with marginalization.

The joint probability, as previously mentioned, is the probability of two events being satisfied which is expressed as $p(A, B)$. With marginalization it is possible to obtain the probability of A regardless of the outcome from event B , i.e it does not matter if event B did or did not occur. To illustrate this situation in an intuitive way, one can see B as a stochastic variable X with two possible outcomes B and B' . The marginal probability of A is denoted $p(A)$ and can be obtained by summing or integrating the joint probabilities over all outcomes for X , which is known as marginalization [15],

$$p(A) = p(A \cap B) + p(A \cap B') \quad (2.11)$$

To derive an useful expression for the time update, one can apply Bayes' rule given from (2.6b) and identify the terms $A = x_{t+1}$, $B = x_t$ and $C = y_{1:t}$,

$$p(x_{t+1}, x_t|y_{1:t}) = p(x_{t+1}|x_t, y_{1:t}) p(x_t|y_{1:t}) = p(x_{t+1}|x_t) p(x_t|y_{1:t}), \quad (2.12)$$

once again, the Markov property has been used in the last equality. Next step is the marginalization which is done by integrating both sides with respect to x_t ,

$$p(x_{t+1}|y_{1:t}) = \int_{\mathbb{R}^n} p(x_{t+1}|x_t) p(x_t|y_{1:t}) dx_t, \quad (2.13)$$

where \mathbb{R}^n defines the state vector while n is the number of states. You may notice a difference in time notation but remember that $p(x_{t+1}|y_{1:t})$ is equal to $p(x_t|y_{1:t-1})$. The same procedure with marginalization can be employed for the denominator in (2.10),

$$p(y_t|y_{1:t-1}) = \int_{\mathbb{R}^n} p(y_t|x_t) p(x_t|y_{1:t-1}) dx_t. \quad (2.14)$$

All these equations are very important and together they form the recursive Bayesian solution which will be used in the particle filter but also when deriving the occupancy grid map algorithm.

Theorem 2.2 (The Bayesian recursion) *If the state space model is given by*

$$x_{t+1} \sim p(x_{t+1}|x_t), \quad (2.15a)$$

$$y_t \sim p(y_t|x_t), \quad (2.15b)$$

then the filter density and the one step ahead prediction density are given by

$$p(x_t|y_{1:t}) = \frac{p(y_t|x_t) p(x_t|y_{1:t-1})}{p(y_t|y_{1:t-1})}, \quad (2.16a)$$

where

$$p(x_{t+1}|y_{1:t}) = \int_{\mathbb{R}^n} p(x_{t+1}|x_t) p(x_t|y_{1:t}) dx_t, \quad (2.16b)$$

$$p(y_t|y_{1:t-1}) = \int_{\mathbb{R}^n} p(y_t|x_t) p(x_t|y_{1:t-1}) dx_t. \quad (2.16c)$$

2.3 The particle filter

This section will only give a brief introduction to the particle filter (PF) because there is already a lot of work published in this field and good literature is easy to find for the interested reader, see e.g. [6]. A mathematical representation of the PF will be derived and summarized later, but first a more intuitive explanation of the PF.

Example 2.1: The principles behind a PF

Imagine a situation where you are blindfolded and located inside some room, but you have already memorized the plan arrangement of the room. The task is now to figure out your position in the room.

Before you move, what would the first guess about your position in the room be? Since you can not see anything your initial guess would probably be; I can be anywhere in the room (step 1, Initialization) ¹. Start to move around and try to remember your movement (step 2, Time update). Put the hands in front of you and move until something touches the finger tips (step 3, measurement update). Now you can make some new assumptions regarding the position in the room (step 4, state estimation), e.g. I am next to a wall and all the previous thoughts of being in the middle of the room is not valid any more and will be discarded (step 5, resampling). Continue to move in any direction (iterate from step 2) and soon new features of the wall will give you more information of your position. The principle of the PF for the UAV works in the same way, see Algorithm 1.

Algorithm 1: The principle of a particle filter for UAV localization

1. Spread out the particles in the predefined map. If the pose of UAV is totally unknown, spread the particles all over the area.
 2. Move the particles according to the motion model.
 3. For each particle, compare the measurements from the laser with simulated measurements from the map. Particles which matches the real measurements best is more likely to be true and will therefore also get a higher weight.
 4. Calculate an average value of all the particles, which will also be the estimation of the true state.
 5. Discard the particles with low weight since they are not very likely to be true and replace them with particles which have higher weights.
 6. Start over from step 2.
-

Now to the basic derivation of the PF. The problem with solving the Bayesian recursive filter from Theorem (2.2) is to mathematically calculate all the non-linearities in the density functions. The fundamental concept of the PF is to approximate all the density functions as a set of samples

$$p(x_t|y_{1:t}) \approx \hat{p}(x_t|y_{1:t}) = \sum_{i=1}^N w_t^i \delta(x_t - x_t^i), \quad \sum_{i=1}^N w_t^i = 1, \quad w_t^i \geq 0, \forall i, \quad (2.17)$$

where $\hat{p}(\cdot)$ denotes an approximation of the distribution, $\delta(\cdot)$ is the Dirac delta and w_t^i denotes the weight which is associated with the sample x_t^i also referred to as particles. The Dirac delta $\delta(x - a)$ can be seen as a function which has zero value everywhere except at $x = a$ where it is infinitely large,

$$\int_{-\infty}^{\infty} f(x) \delta(x - a) dx = f(a), \quad \int_{-\infty}^{\infty} \delta(x) dx = 1. \quad (2.18)$$

The key idea behind the PF is to generate random numbers which represents the filter density in a reliable way. The major problem is to know how these random numbers can be generated from a density function which we will refer to as the target density $t(x_t)$. In most situations it is impossible to generate samples directly from $t(x_t)$, instead another density function which is easier to generate samples from will be used. This function is known as the proposal density $q(x_t)$, although one cannot generate sample from this distribution it is possible to calculate the

¹The statements within brackets is referring to the mathematical representation of the PF.

probability that the sample $\tilde{x}_t \sim q(x_t)$ actually is drawn from the target function,

$$t(\tilde{x}_t) \propto w(\tilde{x}_t) q(\tilde{x}_t), \quad (2.19)$$

where $w(\cdot)$ is referred to as the importance weight. The filter density is the one we are interested in, therefore it is also convenient to let the target density be given by

$$t(x_t) = p(x_t|y_{1:t}). \quad (2.20)$$

Furthermore, we know that

$$p(x_t|y_{1:t}) = \frac{p(y_t|x_t) p(x_t|y_{1:t-1})}{p(y_t|y_{1:t-1})}, \quad (2.21)$$

where $p(y_t|y_{1:t-1})$ only depends on the measurements and can be seen as a normalization factor which does not have to be calculated if the distribution is numerically normed,

$$\underbrace{p(x_t|y_{1:t})}_{t(x_t)} \propto \underbrace{p(y_t|x_t)}_{w(x_t)} \underbrace{p(x_t|y_{1:t-1})}_{q(x_t)}. \quad (2.22)$$

The proposal density for the particle filter is given by,

$$\begin{aligned} q(x_t) &= p(x_{t+1}|y_{1:t}) = \int_{\mathbb{R}^n} p(x_{t+1}|x_t) p(x_t|y_{1:t}) dx_t, \\ &\approx \int_{\mathbb{R}^n} p(x_{t+1}|x_t) \sum_{i=1}^N w_{t|t}^i \delta(x_t - x_t^i) dx_t, \\ &= \sum_{i=1}^N w_t^i \int_{\mathbb{R}^n} p(x_{t+1}|x_t) \delta(x_t - x_t^i) dx_t, \\ &= \sum_{i=1}^N w_t^i p(x_{t+1}|x_t^i). \end{aligned} \quad (2.23)$$

The proposal density for one particle can therefore be chosen as

$$q(x_t^i) = p(x_{t+1}|x_t^i). \quad (2.24)$$

Notice that $p(x_{t+1}|x_t^i)$ still is continuous and needs to be sampled. This can be done by passing particles from the previous time instance through the motion model,

$$\tilde{x}_t^i \sim p(x_{t+1}|x_t^i), \quad (2.25)$$

where the notation \tilde{x}_t^i indicates a sampled particle. From (2.22), one can see that the importance weights are given by

$$\tilde{w}_t^i = p(y_t|\tilde{x}_t^i) \quad (2.26)$$

The acceptance probability is calculated by normalizing the importance weights,

$$\begin{aligned} w_t^i &= \frac{\tilde{w}_t^i}{\sum_{j=1}^N \tilde{w}_t^j} \\ &= \frac{p(y_t|\tilde{x}_t^i)}{\sum_{j=1}^N p(y_t|\tilde{x}_t^j)} \end{aligned} \quad (2.27)$$

An approximation of the filter density can be calculated by inserting the sampled particles from (2.25) into the expression in (2.17),

$$\hat{p}(x_t|y_{1:t}) = \sum_{i=1}^N w_t^i \delta(x_t - \tilde{x}_t^i) \quad (2.28)$$

Algorithm 2 summarizes the basic principles of the PF.

Algorithm 2: A particle filter

1. Initialize the particles, $x_0 \sim p(x_0)$ and set appropriate weights, $w_{0|0}^i$ for all $i = 1, \dots, N$ and let $t := 1$.

2. Time update: Generate N new particles by drawing from the proposal density

$$\tilde{x}_t^i \sim p(x_t|x_{t-1}^i), \quad i = 1, \dots, N. \quad (2.29)$$

3. Measurement update: Compute the importance weights

$$\tilde{w}_t^i = p(y_t|\tilde{x}_t^i), \quad i = 1, \dots, N. \quad (2.30)$$

and normalize $w_t^i = \tilde{w}_t^i / \sum_{j=1}^N \tilde{w}_t^j$

4. Compute an approximation of the estimate

$$\hat{x}_t = \sum_{i=1}^N w_t^i \tilde{x}_t^i \quad (2.31)$$

5. Resampling: For each $i = 1, \dots, N$ draw a new particle x_i with replacement according to

$$P(x_t^i = \tilde{x}_t^j) = w_t^j, \quad j = 1, \dots, N. \quad (2.32)$$

6. Set $t := t + 1$ and iterate from step 2.
-

Chapter 3

Motion and sensor models

Handling the information from the sensors in a robust and reliable way can be difficult but nevertheless very important in state estimation. The general formulation of a stochastic state space model has already been mentioned in section 2 and it is given by,

$$x_{t+1} \sim p(x_{t+1}|x_t), \quad (3.1a)$$

$$y_t \sim p(y_t|x_t). \quad (3.1b)$$

The implementation of this expression can be accomplished by the nonlinear discrete-time state-space model with additive noise given by (2.2),

$$x_{t+1} = f(x_t) + v_t, \quad (3.2a)$$

$$y_t = h(x_t) + e_t. \quad (3.2b)$$

Outliers are a recurrent problem and there are several ways to handle them. Two different approaches have been evaluated:

- A Gaussian distribution with a pre-filtering approach
- A student's t-distribution

while some other methods will briefly be discussed in Chapter 6.

3.1 Motion models

In both navigation and tracking it is important to have a good model describing the motion of an object. In this section a motion model in one, two and three dimensional will be introduced.

3.1.1 1D motion

Consider some unknown motion in one dimension where position and velocity are the states in the state vector. To understand the basic principles of the model one can start with the model in continuous time and then transform it to discrete time. As already proposed, the state vector is given by

$$x_t = \begin{pmatrix} p_t \\ v_t \end{pmatrix} \quad (3.3)$$

where p_t denotes the position and v_t denotes the velocity. The model illustrates the situation when the acceleration is unknown and randomly changes the velocity of the object. To overcome this problem one can model the acceleration as process noise. This is also known as random walk since there is some unknown input interaction on the acceleration and the best we can do is to model it as a random signal. The model in continuous time is given by

$$\dot{x}_t = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} x_t + \begin{pmatrix} 0 \\ 1 \end{pmatrix} w_t, \quad (3.4)$$

where w_t denotes the process noise which randomly can change the velocity. This state space model is known as the constant velocity (CV) model. Applying the Euler discretization will result in,

$$x_{t+1} = \begin{pmatrix} 1 & T \\ 0 & 1 \end{pmatrix} x_t + \begin{pmatrix} \frac{T^2}{2} \\ T \end{pmatrix} w_t. \quad (3.5)$$

Of course there are some other possible ways to discretize the expression in (3.4), but this one is the simplest and most straightforward and will therefore be applied here.

3.1.2 Motion in 2D

When the motion model increases with another dimension, it is convenient to extend the state vector with the position and the velocity in the new dimension but also to add a new state called the heading angle ψ (yaw). The yaw angle can be modeled as random walk in angular velocity in the same way as the acceleration affects the velocity in (3.4),

$$\dot{\psi}_t = w_t, \quad (3.6)$$

where w_t denotes the process noise and should not be mistaken by the angular velocity which usually is associated with ω . The state vector becomes

$$x_t = \begin{pmatrix} p_{x,t} \\ p_{y,t} \\ v_{x,t} \\ v_{y,t} \\ \psi_t \end{pmatrix}, \quad (3.7)$$

and the CV model is given by

$$x_{t+1} = \begin{pmatrix} I & TI & 0 \\ 0 & I & 0 \\ 0 & 0 & I \end{pmatrix} x_t + \begin{pmatrix} \frac{T^2}{2}I & 0 \\ TI & 0 \\ 0 & TI \end{pmatrix} \begin{pmatrix} w_{1,t} \\ w_{2,t} \end{pmatrix}, \quad (3.8)$$

where 0 and I are of appropriate dimensions, furthermore $w_{1,t}$ and $w_{2,t}$ are two different types of process noise illustrating unknown inputs.

3.1.3 Motion in 3D

Going from 1D to the 2D motion is quite intuitive and easy, unfortunately this is not the case when the motion model is updated to 3D. A full 3D PF has not been developed and evaluated, but a lot of effort has been put in the basic understanding and this area is highly relevant for future work and will briefly be introduced here. The major difference between the 2D and the 3D motion is that there will be 6 degrees of freedom (DOF) instead of 3DOF. The unit quaternions which are numerically much more robust than the regular Euler angle will be used for the orientation representation. A full derivation of the quaternions can be found in [14]. The state vector for a 6DOF motion is given by

$$x_t = \begin{pmatrix} p_{x,t} \\ p_{y,t} \\ p_{z,t} \\ v_{x,t} \\ v_{y,t} \\ v_{z,t} \\ q_{0,t} \\ q_{1,t} \\ q_{2,t} \\ q_{3,t} \end{pmatrix}, \quad (3.9)$$

where p and v , denotes position and velocity, while q is the unit quaternions which represent the orientation. We assume an IMU to be available and it is possible to model the motion model in the following way,

$$x_{t+1} = \begin{pmatrix} I & TI & 0 \\ 0 & I & 0 \\ 0 & 0 & I + \frac{1}{2}S(w_{xyz}) \end{pmatrix} \begin{pmatrix} p_t \\ v_t \\ q_t \end{pmatrix} + \begin{pmatrix} \frac{T^2}{2}I & 0 \\ TI & 0 \\ 0 & \frac{1}{2}S'(q_n) \end{pmatrix} \begin{pmatrix} e_{1,t} \\ e_{2,t} \end{pmatrix} \quad (3.10)$$

where

$$S(w_{xyz}) = \begin{pmatrix} 0 & -\omega_x & -\omega_y & -\omega_z \\ \omega_x & 0 & \omega_z & -\omega_y \\ \omega_y & -\omega_z & 0 & \omega_x \\ \omega_z & \omega_y & -\omega_x & 0 \end{pmatrix} \quad (3.11)$$

and

$$S'(q_n) = \begin{pmatrix} -q_1 & -q_2 & -q_3 \\ q_0 & -q_3 & q_2 \\ q_3 & q_0 & -q_1 \\ -q_2 & q_1 & q_0 \end{pmatrix}. \quad (3.12)$$

$\omega_{xyz} = (\omega_x \ \omega_y \ \omega_z)^T$ denotes the angular velocity measured by the gyroscope and $q_n = (q_0 \ q_1 \ q_2 \ q_3)^T$ has the unit quaternion and describes the orientation.

Notice that the angular velocities from the gyroscope are modeled as inputs in the motion model. It is quite straightforward to calculate the rotation matrix from the given quaternion,

$$R(q_n) = \begin{pmatrix} (q_0^2 + q_1^2 - q_2^2 - q_3^2) & 2(q_1q_2 - q_0q_3) & 2(q_1q_3 + q_0q_2) \\ 2(q_1q_2 + q_0q_3) & (q_0^2 - q_1^2 + q_2^2 - q_3^2) & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_3 - q_0q_2) & 2(q_2q_3 + q_0q_1) & -(q_0^2 - q_1^2 - q_2^2 + q_3^2) \end{pmatrix}. \quad (3.13)$$

An illustration of a rotation with quaternions can be seen in Figure 3.1 where the angular velocities are assumed to be known.

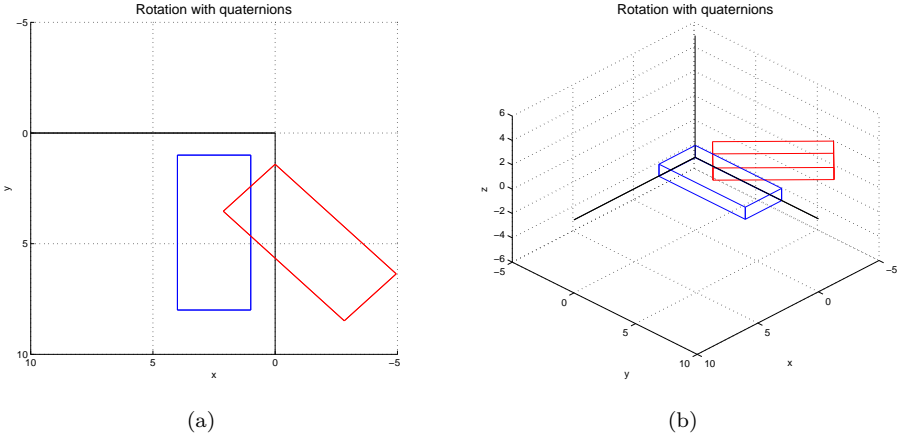


Figure 3.1: An illustration of a rotational motion where the orientation is given by quaternions, i.e. $q_{t+1} = (I + \frac{1}{2}S(\omega_{xyz}))q_t + \frac{1}{2}S'(q_n)$. A simulation has been run for two seconds and the measurements from the gyro is given by the fixed value of $\omega_{xyz} = (0 \ 0 \ \pi/8)^T rad$ while the measurement noise has been neglected. In (a) a top view of the rotation can be seen where the two rectangles illustrates the same object before and after the rotation. A 3D overview of the simulation can be seen in (b). The unit of the axis in the figures is meter.

3.2 Measurement equations

Although the UAV is equipped with both laser and IMU, only the laser will be used in a first stage. The reason for not using the IMU is because things will be kept as simple as possible in the beginning.

3.2.1 The Laser range finder

A two dimensional sweeping laser is very convenient to work with because of its simple nature. These kind of lasers employ a quite basic principle and measures the distance to the closest obstacle at different angles, see Figure 3.2. The direction of each laser beam can be represented by a mathematical function $\alpha(\cdot)$. Let XY_t be a coordinate system attached to the range finder at time t and assume that the X axis is aligned with the laser beam at $\alpha(K/2) = 0^\circ$, see Figure 3.3. The direction of the laser is then given by

$$\alpha(k) = \rho k - \rho K/2, \quad (3.14)$$

where ρ denotes the angular resolution and k is the laser index which is a number between 0 and K , where $K + 1$ corresponds to the number of measurements in one scan. The scaling factor $\rho K/2$ has been introduced to make it easier to convert the measurements from the laser into correct angles.

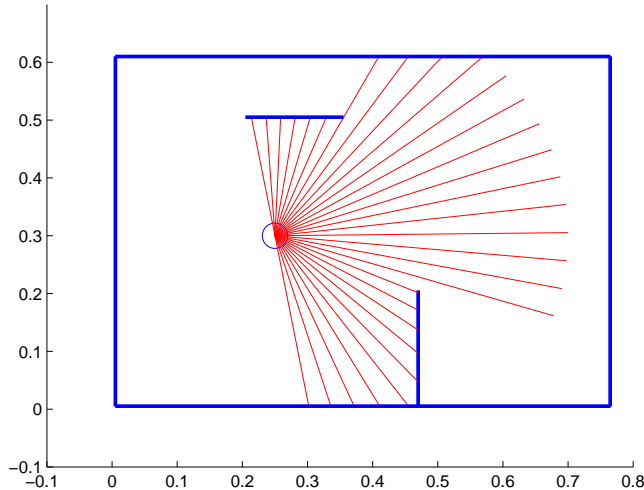


Figure 3.2: A basic principle of a laser range finder which measures the distance to the closest obstacle at different angles. The arc of laser beams which ends up before they reach the wall occurs due to the limited maximum range of the laser. The laser in this example only has a field-of-view of 180 degrees while the real laser used throughout the experiments can detect objects within 240 degrees.

The Hokuyo laser shown in Figure 3.4 will be used throughout the experiments. The Laser has a field-of-view of 240 degrees with 682 measurements for each scan with a max range of 4 m. The benefit with using the Hokuyo Laser is the relative low weight and power consumption, while the drawback is the quite limited max range.

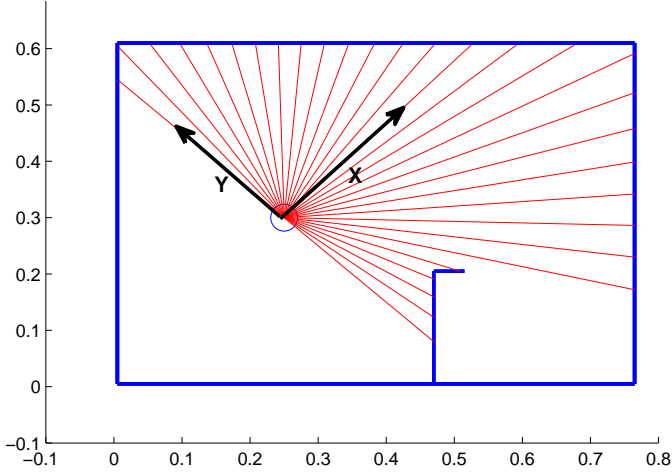


Figure 3.3: A local coordinate system has been attached to the laser.

A sweeping laser generates a sequence of measurements in a predefined pattern, e.g. the laser starts to sample from its far right side relative the forward direction and move to the left side step by step until it reaches the end and start all over again. A laser scan at time t is denoted y_t and the individual measurements will be referred to as y_t^k ,

$$y_t = \{y_t^0, \dots, y_t^K\} \quad (3.15)$$

The measurement y_t^k corresponds to a distance generated from the direction given by (3.14). The measurements within the same scan or scans from different times will have measurement errors compared to each other. For instance, a laser measuring the distance to a static obstacle will generate different measurements each time because of physical and mechanical phenomena. Also, it has a limited max range and detection problems with different materials. These problems can be handled by modelling a mixture of different density functions.



Figure 3.4: The Hokuyo URG-04LX laser sensor.

3.2.2 The Gaussian distribution

As recently mention, simple things will be tried first and therefore an intuitive and easy approach is applied. Each laser measurement is assumed to be a stochastic Gaussian distribution $p(y_t|x_t)$ with the "true" distance as the mean together with some known variance σ_{laser} depending on the laser device. The total probability for each scan is obtained as the product of the individual likelihoods for the measurements

$$p(y_t|x_t) = \prod_{k=0}^K p(y_t^k|x_t) \quad (3.16)$$

The implementation of (3.16) can be done by applying the PDF from (2.5).

$$p(y_t|x_t) = p_{e_t}(y_t - h(x_t)) = \prod_{k=0}^K p_{e_t}(y_t^k - h(x_t)), \quad (3.17)$$

where $p_{e_t}(\cdot)$ is a Gaussian distribution with zero mean and the variance σ_{laser} ,

$$\begin{aligned} p_{e_t}(y_t - h(x_t)) &= \prod_{k=0}^K \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_t^k - h(x_t))^2}{2\sigma^2}\right), \\ &= \frac{1}{\sqrt{2\pi\sigma^2}} \prod_{k=0}^K \exp\left(-\frac{(y_t^k - h(x_t))^2}{2\sigma^2}\right). \end{aligned} \quad (3.18)$$

Inserting this expression in 2.27 gives

$$\begin{aligned} w_t^i &= \frac{\frac{1}{\sqrt{2\pi\sigma^2}} \prod_{k=0}^K \exp\left(-\frac{(y_t^k - h(\hat{x}_t^i))^2}{2\sigma^2}\right)}{\sum_{j=1}^N \frac{1}{\sqrt{2\pi\sigma^2}} \prod_{k=0}^K \exp\left(-\frac{(y_t^k - h(\hat{x}_t^j))^2}{2\sigma^2}\right)} \\ &= \frac{\prod_{k=0}^K \exp\left(-\frac{(y_t^k - h(\hat{x}_t^i))^2}{2\sigma^2}\right)}{\sum_{j=1}^N \prod_{k=0}^K \exp\left(-\frac{(y_t^k - h(\hat{x}_t^j))^2}{2\sigma^2}\right)} \end{aligned} \quad (3.19)$$

Because of the normalization of the weights, one can successfully get rid of the $\frac{1}{\sqrt{2\pi\sigma^2}}$ factor. It might not seem very useful to to get rid of the factor because it is quite easy to calculate the value. But we will later see how the normalization can be used when another distribution is introduced.

The Gaussian function will have its peak at $(y_t^k - h(x_t)) = 0$, which corresponds to a perfect match between the measured value and the measurement function $h(\cdot)$ of the true state. Now, the question is how the function $h(\cdot)$ should look? The measurement function can be different depending on the situation and which problem we want to solve. Here, we have a localization problem and the task is to estimate the position of the UAV inside a predefined map. In this case, $h(\cdot)$ is a non-linear function representing a predefined map which is a model of the real

world. The task is now to compare the distance between the UAV and obstacles in the real world with simulated measurements from the map. In the real world the range measurements are generated by the laser, while the simulated measurements from the map is calculated by using linear algebra or some iterative algorithms.

When there is a match between the simulated measurements and the real measurements, it is with high probability possible to estimate which position in the map that corresponds to the real world position of the UAV.

3.2.3 Outliers

The drawback with a likelihood function calculated as a product of probabilities is how sensitive it can be to outliers. For instance, a set of almost perfect measurements with high probability can easily be destroyed if there is only one outlier with very low probability, this might end up in a low likelihood function because of the product between all the probabilities. There are several ways to interpret outliers and usually an outlier is defined as some random behaviour of the sensor which may result in false detections of objects that does not exist. Here an outlier will also be defined as a situation where small deviation in the angle of the emitted laser results in a huge difference of the measured distance, i.e. one can hit the corner of an object or just miss it by a centimetre and detect something far behind, see Figure 3.5. The same situation can occur if the map does not fully match the real world. When this occurs for a particle in the PF it will imply that the particle will disappear after the resampling step, which is not a desirable behavior. There are several ways to overcome this problem and some of them will be implemented and tested on real data. The first approach is an ad hoc solution where some of the worst matching measurements are removed. To decrease the computational time, 62 of the 682 measurements generated by the Laser for each scan are used. The 5 measurements which have the biggest difference between simulated and real measurements are removed. The second approach is more sophisticated and the purpose is to use a student's t-distribution.

3.2.4 The student's t-distribution

The characteristic features of student's t-distribution compared to the regular Gaussian is the heavier tails which makes the distribution more permissive for outliers far away from the mean. The student's t-distribution is defined in the following way

$$p(x|\mu, \lambda, \nu) = \frac{\Gamma(\frac{\nu+1}{2})}{\Gamma(\frac{\nu}{2})} \left(\frac{\lambda}{\pi\nu} \right)^{\frac{1}{2}} \left[1 + \frac{\lambda(x-\mu)^2}{\nu} \right]^{-\frac{\nu+1}{2}}, \quad (3.20a)$$

$$E(X) = \mu \quad \text{for } \nu > 1, \quad (3.20b)$$

$$\text{var}(X) = \frac{1}{\lambda} \frac{\nu}{\nu-2} \quad \text{for } \nu > 2, \quad (3.20c)$$

where

$$\Gamma(n) = (n-1)! \quad \text{for } n > 0. \quad (3.21)$$

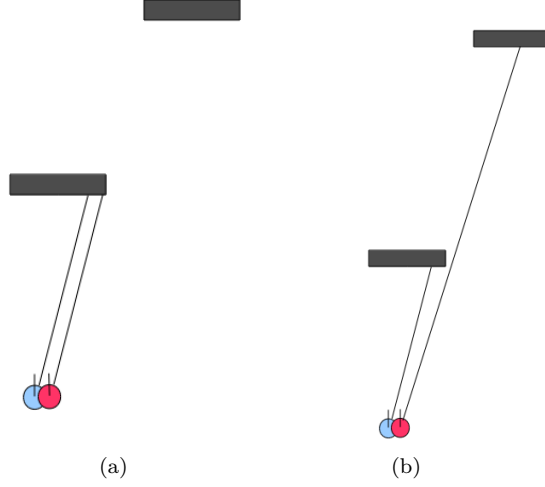


Figure 3.5: Usually an outlier is defined as some random behaviour in the measurements, for instance false detections of objects which does not exist. In (a) one can see a normal situation where the true (blue) and the simulated (red) measurement is quite similar to each other. When there are small deviations in the orientation, it can result in huge differences between the measured distances as shown in (b) and these situation will also be defined as outliers.

A visual representation of the student's t-distribution can be seen in Figure 3.6. Apply the student's t-distribution to the expression in equation 3.17,

$$\begin{aligned}
 p(y_t|x_t) &= p_{e_t}(y_t - h(x_t)) = \\
 &= \frac{\Gamma(\frac{\nu+1}{2})}{\Gamma(\frac{\nu}{2})} \left(\frac{\lambda}{\pi\nu} \right)^{\frac{1}{2}} \left[1 + \frac{\lambda(y_t - h(x_t))^2}{\nu} \right]^{-\frac{\nu+1}{2}}, \\
 &= \prod_{k=0}^K \frac{\Gamma(\frac{\nu+1}{2})}{\Gamma(\frac{\nu}{2})} \left(\frac{\lambda}{\pi\nu} \right)^{\frac{1}{2}} \left[1 + \frac{\lambda(y_t^k - h(x_t))^2}{\nu} \right]^{-\frac{\nu+1}{2}},
 \end{aligned} \tag{3.22}$$

Notice that the expression

$$\frac{\Gamma(\frac{\nu+1}{2})}{\Gamma(\frac{\nu}{2})} \left(\frac{\lambda}{\pi\nu} \right)^{\frac{1}{2}}, \tag{3.23}$$

does not depend on x_t and will therefore disappear after the normalization of the weights,

$$w_t^i = \frac{\prod_{k=0}^K \left[1 + \frac{\lambda(y_t^k - h(\hat{x}_t^i))^2}{\nu} \right]^{-\frac{\nu+1}{2}}}{\sum_{j=1}^N \prod_{k=0}^K \left[1 + \frac{\lambda(y_t^k - h(\hat{x}_t^j))^2}{\nu} \right]^{-\frac{\nu+1}{2}}}. \tag{3.24}$$

This will of course make it easier to implement the student's t-distribution since the Γ function does not have to be considered at all. A comparison between the

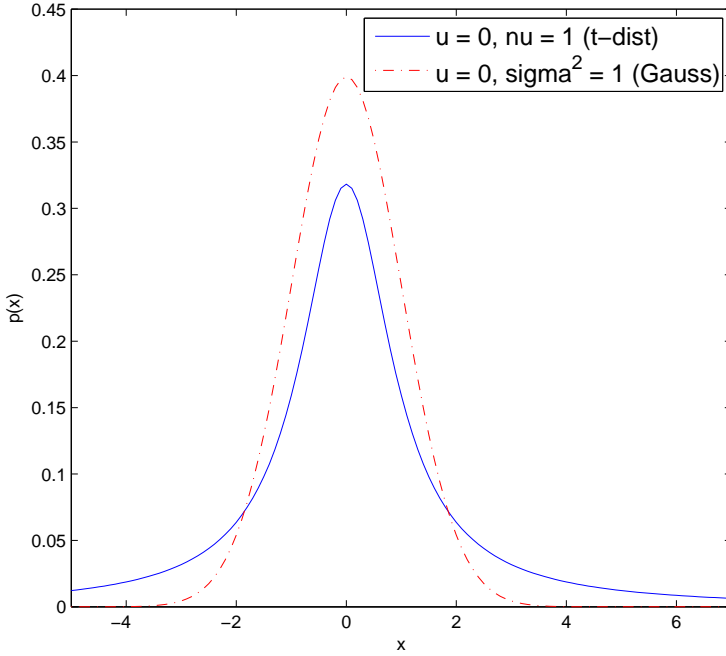


Figure 3.6: A comparison between a Gaussian and a student's t-distribution. Unfortunately Matlab does not provide the three-parameter version of the student's t-distribution which includes μ, λ, ν , but the definition of the two-parameter version is very similar to the expressions in (3.20).

result using real data the Gaussian distribution and the student's t-distribution can be seen in Chapter 5.

3.2.5 Inertial Measurement Unit

The accelerometer inside the IMU measures the acceleration in all three dimensions while the gyroscope measures the angular velocities in the three dimensions. The angular velocities have already been handled in the motion model, see section 3.1.3. The measured acceleration is given by,

$$y_{a,t} = R(q_n)a_t - R(q_n)g_{xyz} + e_{a,t}, \quad (3.25)$$

where $g_{xyz} = (g_x \ g_y \ g_z)^T$ denotes the gravity force which often is assumed to be $(0 \ 0 \ -9.82)^T$ while $R(q_n)$ is the rotation matrix defined in (3.13), furthermore $e_{a,t}$ describes the measurement noise in acceleration from the IMU.

Chapter 4

Mapping and localization

A very popular and useful map representation for dynamic environments is the occupancy grid map. As the name indicates, occupancy grid mapping addresses the problem of creating maps based on the probability that the grids are occupied or not. A grid is defined as a discrete representation of some 2D or 3D area. The basic concept of occupancy grid mapping is to represent the features of a real world environment as binary variables which are randomly spread out in a map, Figure 4.1. The occupancy grid map is suitable to use when the position and the orientation of the robot is known, i.e. when the SLAM problem is solved. This could sound very confusing since SLAM already addresses the mapping problem. But there is actually a major benefit with the grid map because it can generate a map suitable for path planning and navigation which usually is not the case with the map achieved from SLAM.

4.1 Maps

A map m is defined as a list of objects and their corresponding properties from some environment,

$$m = \{m_1, m_2, \dots, m_N\}, \quad (4.1)$$

where N is the number of objects and m_n is the property for each object. There are usually two kinds of maps, feature-based and location-based. Location-based maps have the advantage to easily identify the presence of objects, where each element m_n specifies a unique location in the world but also the corresponding status in this location. It is common to replace the label m_n with $m_{x,y,z}$ to specify which world coordinate the label is referring to. Occupancy grid maps are location-based and will be discussed soon. Feature-based maps are quite different and contains a list of the properties for each feature (shape, average distance, etc.) and the location of the feature.

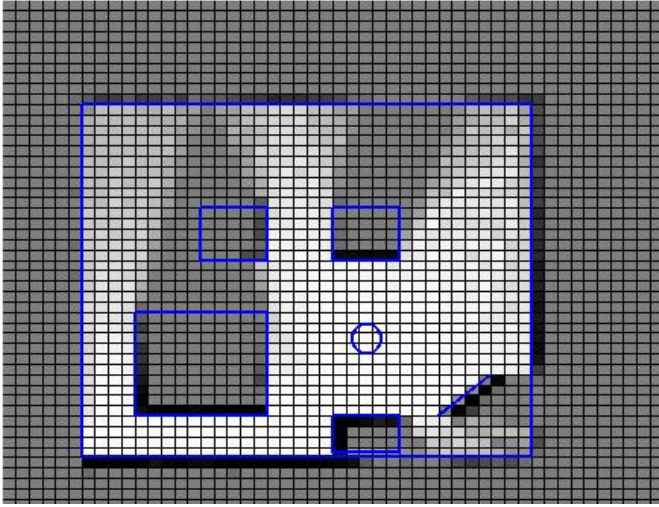


Figure 4.1: A logarithmic gridmap created in Matlab. The color of each grid cell corresponds to the probability for a grid being occupied (white = low probability, dark = high probability).

4.2 Grid maps

The main problem with occupancy grid maps is the estimation of the posterior distribution for a map given the data,

$$p(m|y_{1:t}, x_{1:t}), \quad (4.2)$$

where m is the map, $y_{1:t}$ the set of measurements and $x_{1:t}$ is the robot's position and orientation (also known as a pose) from which the measurements were taken.

Example 4.1: Problems with grid maps

The number of possible distributions for a map grows exponentially when the number of grid cells are increased. Take for instance a 2D map which is 5 x 7 meters with a grid size of 0.1 meter, this situation can generate $2^{50 \times 70} = 6.35 \times 10^{526}$ different maps and it would not be possible to calculate a posterior distributions for each map. Remember, this is a very small map and sometimes not even interesting because of its limited size.

To get rid of the more or less impossible situation shown in the example, one can break down the problem into smaller problems of estimating

$$p(m_i|y_{1:t}, x_{1:t}), \quad (4.3)$$

where m_i denotes a grid cell with index i and $p(m_i)$ represents the probability of the cell being occupied [11, 12]. This approach is quite convenient but it does

not take dependencies between neighbouring cells into account which is a major drawback. The posterior distribution of the map is approximated as the product of each grid cell,

$$p(m|y_{1:t}, x_{1:t}) = \prod_i p(m_i|y_{1:t}, x_{1:t}). \quad (4.4)$$

4.3 Binary Bayes filter

Because of the factorization we now have a binary estimation problem for every grid cell. According to [11], occupancy grid maps is an example of binary Bayes filters with the static state m_i which does not change. The author of this document interpret the situation of seeing the states in the grid map as static can be done because each cell is either occupied or free, which mean that we can use the static state occupied for all grids and then just look at the probability of the state in order to determine if the grid is occupied or not. A high probability of the state corresponds to an occupied grid while a low probability corresponds for a free cell. In this way the state does not change but the probability of the state being true could.

To make the derivation of the occupancy grid mapping algorithm easier to follow a substitution will be done where the information from $y_{1:t}$ and $x_{1:t}$ are replaced by the single parameter $z_{1:t}$. Start by defining the odds of a state as the probability for an event divided by the probability of its negate,

$$\frac{p(m_i)}{p(\neg m_i)} = \frac{p(m_i)}{1 - p(m_i)}. \quad (4.5)$$

Notice the lack of time index (index i still represents a specific grid cell and does not have anything to do with time) which indicates a static state. Apply the logarithm to this expression and we can define the log odds ration as

$$l(m_i) = \log \left(\frac{p(m_i)}{1 - p(m_i)} \right). \quad (4.6)$$

Consider Bayes filter algorithm which already has been derived in section (2.2),

$$p(m_i|z_{1:t}) = \frac{p(z_t|m_i) p(m_i|z_{1:t-1})}{p(z_t|z_{1:t-1})}. \quad (4.7)$$

The expression $p(z_t|m_i)$ can be very difficult to calculate and therefore an inverse measurement model $p(m_i|z_t)$ will be used. Imagine a situation with a door being either opened or closed, i.e. we have a binary state X and would like to estimate the state of the door using only a camera image y_t . It is easier to find algorithms for determining if the door is opened or closed from a camera image $p(x|y_t)$, than trying to describe all possible camera images showing a closed door $p(y_t|x)$.

Bayes rule gives the following expression for the measurement model

$$p(z_t|m_i) = \frac{p(m_i|z_t) p(z_t)}{p(m_i)}, \quad (4.8)$$

which can be inserted in (4.7)

$$p(m_i|z_{1:t}) = \frac{p(m_i|z_t) p(z_t) p(m_i|z_{1:t-1})}{p(m_i) p(z_t|z_{1:t-1})}. \quad (4.9)$$

The analytical solution to this posterior distribution is hard to determine because $p(z_t)$ and $p(z_t|z_{1:t})$ are difficult to calculate. To overcome this problem, one can derive a similar equation but for the free cell in the map, $\neg m_i$,

$$p(\neg m_i|z_{1:t}) = \frac{p(\neg m_i|z_t) p(z_t) p(\neg m_i|z_{1:t-1})}{p(\neg m_i) p(z_t|z_{1:t-1})}. \quad (4.10)$$

Dividing the probability of the state with its negation leads to the opportunity of cancelling out complex probability distributions:

$$\begin{aligned} \frac{p(m_i|z_{1:t})}{p(\neg m_i|z_{1:t})} &= \frac{\frac{p(m_i|z_t) p(z_t) p(m_i|z_{1:t-1})}{p(m_i) p(z_t|z_{1:t-1})}}{\frac{p(\neg m_i|z_t) p(z_t) p(\neg m_i|z_{1:t-1})}{p(\neg m_i) p(z_t|z_{1:t-1})}}, \\ &= \frac{p(m_i|z_t)}{p(\neg m_i|z_t)} \frac{p(m_i|z_{1:t-1})}{p(\neg m_i|z_{1:t-1})} \frac{p(\neg m_i)}{p(m_i)}. \end{aligned} \quad (4.11)$$

Replace $p(\neg m_i|z_{1:t})$ with $1 - p(m_i|z_{1:t})$,

$$\frac{p(m_i|z_{1:t})}{1 - p(m_i|z_{1:t})} = \frac{p(m_i|z_t)}{1 - p(m_i|z_t)} \frac{p(m_i|z_{1:t-1})}{1 - p(m_i|z_{1:t-1})} \frac{1 - p(m_i)}{p(m_i)}. \quad (4.12)$$

The posterior distribution, $p(m_i|z_{1:t})$, holds the interesting information about the grid cell m_i being occupied or not. The expression in (4.12) can be rearranged and the posterior distribution for each grid is calculated as

$$p(m_i|z_{1:t}) = 1 + \left[\frac{p(m_i|z_t)}{1 - p(m_i|z_t)} \frac{p(m_i|z_{1:t-1})}{1 - p(m_i|z_{1:t-1})} \frac{1 - p(m_i)}{p(m_i)} \right]^{-1}, \quad (4.13)$$

where $p(m_i|z_{1:t-1})$ is the probability of the grid in the previous timestep. Furthermore, $p(m_i)$ is the prior distribution of each grid cell m_i and defines the probability of a cell being occupied before any measurements have been utilized. The most intuitive value of the prior probability for a cell is apparently 0.5 which indicates that initially we don't know anything about the map and each cell could be either occupied or not. The last distribution in (4.13) is the already mentioned inverse measurement model $p(m_i|z_t)$. There are several ways of estimating this distribution and we will get back to it later.

The reason why it was possible to end up in the relatively straightforward equation defined in (4.13), was because of the simplicity and the strength of using binary states. This would not have been the case if we had a situation with three or more

plausible outcomes. If the distribution for the state indicates a low probability for the first outcome, one can not use the simple exclusion method as in the binary case to distinguish the probability for all three states. Instead one have to introduce some complex distribution function which makes all the calculations more complicated.

The probability representation in (4.13) suffers from numerical instabilities when the probability is in the region of zero and one. This can be handled with the log odds representation which is derived by the logarithm of the expression in (4.12). This expression is also known as binary Bayes filter,

$$\begin{aligned} l_{t,i}(m_i) &= \log \frac{p(m_i|z_t)}{1 - p(m_i|z_t)} + \log \frac{p(m_i|z_{1:t-1})}{1 - p(m_i|z_{1:t-1})} + \log \frac{1 - p(m_i)}{p(m_i)}, \\ &= \log \frac{p(m_i|z_t)}{1 - p(m_i|z_t)} + l_{t-1,i}(m_i) - l_0(m_i), \end{aligned} \quad (4.14)$$

where

$$l_0 = \log \frac{p(m_i)}{1 - p(m_i)}, \quad (4.15)$$

and

$$l_{t-1,i}(m_i) = \log \frac{p(m_i|z_{1:t-1})}{1 - p(m_i|z_{1:t-1})}. \quad (4.16)$$

4.4 The occupancy grid mapping algorithm

To complete the derivation of the occupancy grid mapping algorithm, one have to remember the substitution of $z_t = (y_t, x_t)$ made in the beginning,

$$\begin{aligned} l_{t,i} &= \log \frac{p(m_i|y_t, x_t)}{1 - p(m_i|y_t, x_t)} + \log \frac{p(m_i|y_{1:t-1}, x_{1:t-1})}{1 - p(m_i|y_{1:t-1}, x_{1:t-1})} + \log \frac{1 - p(m_i)}{p(m_i)}, \\ &= \log \frac{p(m_i|y_t, x_t)}{1 - p(m_i|y_t, x_t)} + l_{t-1,i} - l_0(m_i), \end{aligned} \quad (4.17)$$

$$\text{where } l_0 = \log \frac{p(m_i)}{1 - p(m_i)} \text{ and } l_{t-1,i}(m_i) = \log \frac{p(m_i|y_{1:t-1}, x_{1:t-1})}{1 - p(m_i|y_{1:t-1}, x_{1:t-1})}.$$

A very neat property of the log odds representation is how easily the posterior distribution is recovered:

$$p(m_i|y_{1:t}, x_{1:t}) = 1 - \frac{1}{1 + \exp(l_{t,i})}. \quad (4.18)$$

Algorithm 3 summarizes the basic principles of the occupancy grid mapping. The most tricky part with the occupancy grid mapping algorithm is to find a suitable inverse sensor/measurement model. The strength of using a laser for mapping is the relative high precision in range and bearing, therefore a quite simple but nevertheless powerful method will be applied, see Algorithm 4.

Algorithm 3: Occupancy grid mapping

1. Input: $l_{(t-1,i)}, x_t, z_t$
2. **for** all grid cells m_i **do**
3. **if** the cell m_i is covered by the measurement y_t **then**
4. Update the log odds of the grid cell accordingly

$$l_{(t,i)} = l_{(t-1,i)} + \log \frac{p(m_i|y_t, x_t)}{1 - p(m_i|y_t, x_t)} - l_0 \quad (4.19)$$

5. **else**
6. The log odds of the grid will not change,

$$l_{(t,i)} = l_{(t-1,i)} \quad (4.20)$$

7. **end if**
 8. **end for**
 9. Output: $l_{(t,i)}$
-

Sometimes it can be useful to define a max and min threshold for $l_{t,i}$ to prevent instabilities for probabilities near zero and one. Changing these threshold levels closer to zero will make the map more dynamic and thereby more able to switch a grid from free to occupied or vice versa.

Algorithm 4: A simple version of the InverseSensorModel

$$\text{InverseSensorModel} = \begin{cases} l_{\text{occu}} & \text{if the beam ends up inside the grid} \\ l_{\text{free}} & \text{if the beam traverses the grid} \end{cases} \quad (4.21)$$

where l_{occu} is a positive value which corresponds to a probability greater than 0.5 while l_{free} is a negative value corresponding to a probability less than 0.5. These statements can easily be verified by the formula in equation (4.18).

4.5 Octomap

Octomap [19] is a three dimensional occupancy grid map which addresses the principles shown in Algorithm 3 and 4. The benefit of using an Octomap compared to a straight-forward version of a grid map lies in the effective way of allocating memory for each cell. The easy way would be to create a memory cell for each grid in the map.

Example 4.2: Memory cell consumption

If we go back to the simple example with a map of size 5×7 meter, expand it with further a dimension in height (3 m) and keep the grid resolution of 0.1 meter. This is quite an ordinary room size and the memory consumption will finally end up in $50 \times 70 \times 30 = 10500$ memory cells.

Of course it is possible to implement a grid map in this way, but usually one has to create maps of entire buildings, or even worse an outdoor area, which will make the map very inefficient regarding the memory. Octomap provides a solution to the memory problem by utilizing intelligent algorithms to effectively decrease the memory usage. The basic idea behind Octomap is an algorithm called octree which is a tree data structure where each node can be divided into eight subnodes (called children of a node) until a minimum size is reached, see Figure 4.2. Octomap has

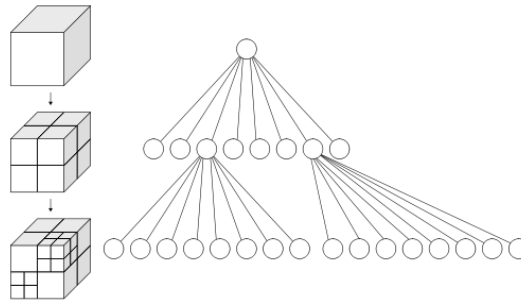


Figure 4.2: A visual illustration of the octree structure, [17]. Some of the nodes have been divided into smaller nodes.

adapted this method and each node in the octree structure is represented by a grid in the map. Imagine a situation with some $0.8 \times 0.8 \times 0.8$ m area and a grid resolution of 0.1 meter being totally free from any obstacles. This area could then be represented by only one node and it would not gain any further information if the area was divided into smaller subregions, because all of them would still be free. If a grid becomes occupied it will be divided into smaller regions until the grid size or some lower bound is reached.

To make the illustrations easier to follow, 2D figures have been used. From Figure 4.3 one can see the benefits of using the Octomap representation instead of the straight-forward approach where each grid cell has an allocated memory cell connected to it. The map consists of $0.8/0.1 \times 0.8/0.1 \times 0.8/0.1 = 512$ grids and each cell is either occupied = 1_2 or free = 0_2 and will be represented by a digital bit (indicated by the index 2). For this example it would mean that we need 512 bits = 64 bytes (1 byte = 8 bits) to represent a regular grid map. With Octomap one have to define three situation for each node; it is either free, occupied or a new subnodes. If we go back to the digital representation, we need at least two bits to represent three different modes; 01_2 = occupied, 10_2 = free and 11_2 = subnode. Notice that the binary representation 11_2 actually is a pointer to a new node (at

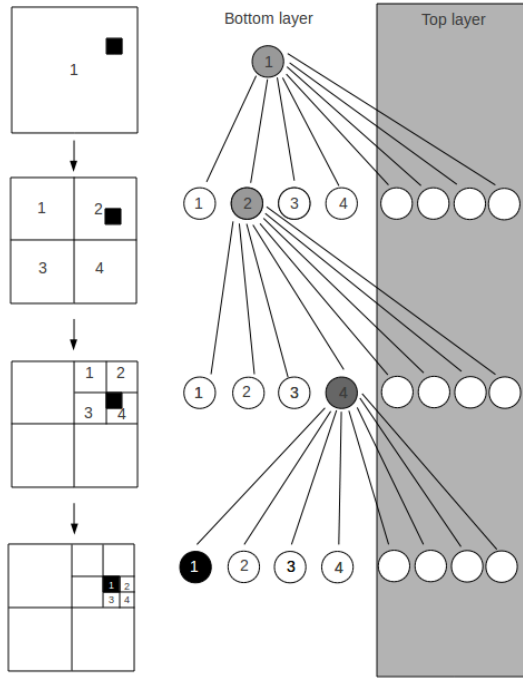


Figure 4.3: The white squares to the left illustrates the bottom layer of a cube, furthermore assume that the top layer is totally free. The black circle corresponds to an occupied area while the white circles represents free regions. The gray circle indicates when a node has been divided into new subnodes. The area is divided into smaller regions until the interesting grid or some lower bound is reached.

least 4 bytes are required to represent a pointer) but will now be seen as an indication of new nodes. This assumption has been made to simplify the understanding of the strengths with using Octomaps, we will get back to the pointer later. When all children of a node have the same state (free or occupied) one can cut away the children and later re-generate them if future measurements indicates a new state for the for the node.

Since there are three different levels in this example and each of them consists of eight nodes which have a binary representation of two bits, it will finally end up in a memory consumption of $3 \times 8 \times 2 = 48$ bits = 6 bytes. Compared to 64 bytes, this is a quite good improvement on a relatively limited area.

4.5.1 Pointers

The real size of a pointer depends on the hardware structure inside the computer, either you run a 32- or 64-bit architecture. For instance, the 32-bit architecture requires 32 address bits to access a specific cell in the memory, the pointer will

therefore be 32 bits or 4 bytes.

The octree structure for each node consists of sorted lists of its children [19]. Therefore it is possible to use eight pointer to represent the eight children of a node and will require a memory usage of $8 \times 4 \text{ byte} = 32 \text{ byte}$, which is a quite huge memory consumption. To overcome this problem one can create a pointer to an array of eight pointers, but the array will only be allocated when the children of a node needs to be initialized. Notice that the pointer approach would actually generate a larger memory usage compared with the straight-forward method in the previous example since we need three levels of arrays which is equal to $3 \times 32 \text{ byte} = 96 \text{ bytes}$. This could seem strange, but what will happen with the straight-forward approach if the grid resolution decreases to 0.05m. The memory usage will increase to $0.8/0.05 \times 0.8/0.05 \times 0.8/0.05 = 512 \text{ bytes}$ while the Octomap approach is not even close to this value since the only thing one have to do is to introduce an extra array which corresponds to $4 \times 32 \text{ byte} = 129 \text{ byte}$.

4.5.2 Quantization

There are always quantization problems when creating discrete models of a real system. This is also the case for Octomap because the map consists of grid cells with a certain resolution. Imagine a situation where the UAV is moving parallel along a straight wall and measuring the distance to it at a specific angle. If there is no measurement error for the laser, one should measure the same distance to the wall along the whole path. Unfortunately this is not the case when measurements are simulated in the Octomap because each measurement always refer to the center of a grid and the measurements can therefore differ a lot depending of the grid resolution and the angle of incidence. It is possible to overcome this problem by increasing the variance of the laser in the sensor model.

4.6 Scanmatching

Ground vehicles are often equipped with odometers which can give some indication of the movement and the equivalent sensor for aerial vehicles is the IMU. Unfortunately IMU's are hard to use in the same way as odometers because of the fact that IMU's can drift resulting in corrupt data when calculating the position. As already mentioned IMU's measures both angular velocity and acceleration, integrating the acceleration twice will give the position. The measurement error will also be integrated and the estimation of the position is therefore corrupted.

Since the UAV is equipped with a laser, it is possible to estimate the relative displacement between two laser scans [7]. A very powerful algorithm for these kinds of problems is the Iterative Closest Point (ICP). The key idea of ICP is to match two sets of data and then calculate the displacement between these. The ICP algorithm is summarized in Algorithm 5 below.

4.6.1 The general cost function

Define a set of points $\{q_t\}$ at time t , then a single point $q_t(j)$ is given by:

$$\begin{aligned} x_t(k) &= d_t^k \cos(\alpha_t(k)), \\ y_t(k) &= d_t^k \sin(\alpha_t(k)), \end{aligned} \quad (4.22)$$

where $\alpha_t(k)$ is the function defined in equation (3.14) and d_t^k is a distance measured by the laser.

When there is a motion, two consecutive scans from time t and $t + 1$ have gathered data from two different positions. The scans are referred to as $\{q_t\}$ and $\{q_{t+1}\}$ which is given in their local coordinate system. To find the correspondence between the scans, one have to project the points from $\{q_{t+1}\}$ into $\{q_t\}$'s coordinate system according to some transformation T_t . The transformed points will be known as $\{\hat{q}_t\}$ and the key idea with the ICP algorithm is to estimate T_t as good as possible. The transformation is a combination of the relative displacements and the rotation,

$$T_t = \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta \psi \end{bmatrix}. \quad (4.23)$$

The coordinates for $\{\hat{q}_t\}$ are given by

$$\begin{bmatrix} \hat{x}_t(k) \\ \hat{y}_t(k) \end{bmatrix} = \begin{bmatrix} \cos(\Delta \psi) & -\sin(\Delta \psi) \\ \sin(\Delta \psi) & \cos(\Delta \psi) \end{bmatrix} \begin{bmatrix} x_{t+1}(k) \\ y_{t+1}(k) \end{bmatrix} + \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}. \quad (4.24)$$

Next step is to define a correspondence index function as $J(k)$ when the projection $\hat{q}_t(k)$ of the point $q_{t+1}(k)$ corresponds with the earlier observed point $q_t(J(k))$, i.e. k is the index of projected point from time $t + 1$ while $J(k)$ is the index of the previous observed point at time t (see equation (4.31) for an example of how the function can be defined). An error function is introduced to get some indication of the error between the matched points. The error can be calculated as a function of the projected point and the observed point,

$$e_{T_t}(k) = e_{T_t}(\hat{q}_t(k), q_t(J(k))). \quad (4.25)$$

Outliers and unmatched points can be detected by defining the following function,

$$p_{T_t} = \begin{cases} 1 & \text{if } |e_{T_t}(j)| \geq E \\ 0 & \text{otherwise} \end{cases}. \quad (4.26)$$

where E is the threshold level and the number of valid correspondence is given by

$$n_{T_t} = \sum_{k=0}^K p_{T_t}(k). \quad (4.27)$$

Futhermore, the ratio between the number of points that actually have been used and total number of points is calculated as

$$P_{T_t} = \frac{n_{T_t}}{K + 1}. \quad (4.28)$$

There are many things which makes it impossible to find an exact correspondence between the points, such as measurement noise, moving objects, occluded areas, among others. The situation can be seen as an optimization problem for determining a 2D transformation T_t by minimizing a matching criterion I_{T_t} . By accumulating the matching errors for all valid points and then dividing the sum by both n_{T_t} to normalize and P_{T_t} to penalize low correspondence rates, a general matching index can be formulated,

$$I_{T_t} = \frac{\sum_{k=0}^K p_{T_t}(k) \cdot e_{T_t}(k)}{n_{T_t} P_{T_t}}, \quad (4.29)$$

which is the cost function in the optimization problem.

4.6.2 The iterative closest point algorithm

The first step in the ICP algorithm is to calculate the coordinates \hat{q}_t according (4.24) where T_t is initialized by the motion estimated by the odometers which will be a problem because the UAV does not have any odometers. Next step is to calculate the squared distance for all possible combinations of the point from \hat{q}_t and q_t ,

$$e(i, k) = (x_t(i) - \hat{x}(k))^2 + (y_t(i) - \hat{y}(k))^2. \quad (4.30)$$

The third step includes the estimation of the correspondence index function $J(k)$ by minimizing the squared distance between the points,

$$J(k) = m, \text{ if } e(m, k) = \min_{i=0}^K [e(i, k)]. \quad (4.31)$$

Finally the match error function is given by

$$e_{T_t} = (x_t(J(k)) - \hat{x}(k))^2 + (y_t(J(k)) - \hat{y}(k))^2. \quad (4.32)$$

Now it is possible to analytically solve the optimization problem by minimizing the cost function in (4.29), the solution can be found in [7]. The relative displacement which is estimated by the ICP algorithm can be used in the PF as an indication of how the particles should be moved.

Algorithm 5: Iterative Closest Point

1. Define two sets of points $\{q_t, q_{t+1}\}$ and calculate $\{\hat{q}_t\}$ as the projection of $\{q_{t+1}\}$.
 2. Find the closest point from each set by minimizing the squared distance for every possible combination in $\{\hat{q}_t\}$ and q_t , see (4.30)-(4.32).
 3. Discard points which are further away than a certain threshold, see (4.26).
 4. Calculate the transformation vector (displacement and rotation) by analytically solve a optimization problem, see (4.29).
 5. Move and rotate the points from set q_t according the transformation vector.
 6. Go back to step 1 and iterate the procedure a couple of times.
-

The number of points within each set does not have to be equal which makes the algorithm very flexible. The drawback with the algorithm is the fact that it can only guarantee to find a local minima and not a global.

Chapter 5

Experimental Results

The VICON lab located at the University of Linköping is equipped with a motion capture system which makes it possible to get ground truth position for all the experiments. The VICON lab has been very useful when evaluation the performance of the PF, but it has also been used when creating Octomaps. The motion capture system track reflective balls which are attached to different objects by using ten infrared cameras and infrared lamps, see Figure 5.1.

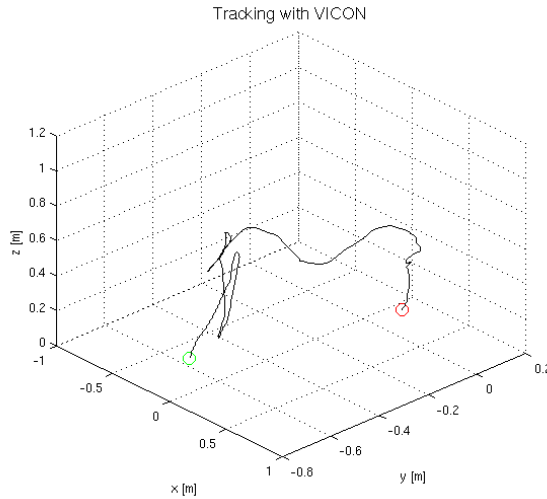


Figure 5.1: A visualization of the motion capture system tracking an object. The beginning and the end of the motion is marked with two circles.

An example of the information generated by the motion capture system can be seen in Figure 5.2. In the VICON lab there is a global coordinate system which defines origo, i.e. the position and orientation for all the objects of interest are given relative the global coordinate system.

```

1 Joints Kinematics
2 250
3 ,,World_LRF_110216,,,,,
4 Frame,Sub_Frame,RX,RY,RZ, TX,TY,TZ,deg,deg,deg,mm,mm,mm
5 1,0,0.43533,0.272921,-1.89824,-231.584,-1028.19,76.7924
6 2,0,0.43533,0.272921,-1.89824,-231.584,-1028.19,76.7924
7 3,0,0.43533,0.272921,-1.89824,-231.584,-1028.19,76.7924
8 4,0,0.43533,0.272921,-1.89824,-231.584,-1028.19,76.7924
9 5,0,0.43533,0.272921,-1.89824,-231.584,-1028.19,76.7924
10 6,0,0.43533,0.272921,-1.89824,-231.584,-1028.19,76.7924
11 7,0,0.43533,0.272921,-1.89824,-231.584,-1028.19,76.7924
12 8,0,0.43533,0.272921,-1.89824,-231.584,-1028.19,76.7924
13 9,0,0.43533,0.272921,-1.89824,-231.584,-1028.19,76.7924
14 10,0,0.43533,0.272921,-1.89824,-231.584,-1028.19,76.7924
15 11,0,0.43533,0.272921,-1.89824,-231.584,-1028.19,76.7924
16 12,0,0.43533,0.272921,-1.89824,-231.584,-1028.19,76.7924
17 13,0,0.420803,0.294849,-1.87807,-231.641,-1028.03,76.66
18 14,0,0.420803,0.294849,-1.87807,-231.641,-1028.03,76.66
19 15,0,0.422241,0.267254,-1.91863,-231.723,-1028.12,76.7159
20 16,0,0.422241,0.267254,-1.91863,-231.723,-1028.12,76.7159
21 17,0,0.422241,0.267254,-1.91863,-231.723,-1028.12,76.7159
22 18,0,0.422241,0.267254,-1.91863,-231.723,-1028.12,76.7159
23 19,0,0.422241,0.267254,-1.91863,-231.723,-1028.12,76.7159
24 20,0,0.422241,0.267254,-1.91863,-231.723,-1028.12,76.7159
25 21,0,0.422241,0.267254,-1.91863,-231.723,-1028.12,76.7159
26 22,0,0.422241,0.267254,-1.91863,-231.723,-1028.12,76.7159
27 23,0,0.422241,0.267254,-1.91863,-231.723,-1028.12,76.7159
28 24,0,0.422241,0.267254,-1.91863,-231.723,-1028.12,76.7159
29 25,0,0.422241,0.267254,-1.91863,-231.723,-1028.12,76.7159
30 26,0,0.422241,0.267254,-1.91863,-231.723,-1028.12,76.7159

```

Figure 5.2: The first column indicates which frame the data is associated with, the system runs with a speed of 250 Hz which means that the time between each frame is 0.004s. The second column is irrelevant but the remaining six columns are useful and defines the pose of the object relative the global coordinate system in the following order; rotation around X, Y, Z and then the translation for X, Y, Z.

Both simulated and real data have been evaluated and some of the results will be summarized in this section. The simulations are performed by using Octomap together with Matlab, while the VICON lab has been used when evaluating the real data.

5.1 Simulations

The possibility of simulating algorithms and important properties before they are tested in reality can be a very powerful tool. This section will give a brief overview of the simulation environment used throughout this thesis. The software used for the simulations is both Matlab and C++.

5.1.1 Localization in the 1.5D PF

To illustrate the basic principles and the simplicity of the particle filter, a 1.5 dimensional example will be used.

Example 5.1: Illustration of the 1.5D PF

Imagine an airplane flying at constant altitude over some known terrain and the task is to estimate the position of the plane. The airplane is equipped with a sonar measuring the distance to the ground and the altitude is known and constant.

The reason why this situation is stated as a 1.5 dimensional examples is because the plane moves along a straight line in one dimension while the sonar is mounted downwards and measuring the distance to the ground in a different dimension compared to the movement direction. The motion model in this example is the one dimensional CV model, see equation (3.5). Initially the particles are spread out all over the terrain since we do not know anything about the situation (step 1 in Algorithm 2, initialization), see Figure 5.3a. Move the particles according to the motion model (step 2 in Algorithm 2, time update). Sample data from the sonar and update the weights of particles by comparing the real measurements with the simulated ones (step 3 in Algorithm 2, measurement update). Compute an estimation of position by using the weights (step 4 in Algorithm 2, estimation). Already after the first measurement there is enough information available to discard a lot of particles (step 5 in Algorithm 2, resampling), see Figure 5.3c. The procedure is repeated and new estimations can be done.

To make the situation more realistic, some measurement error has been added to the sonar. The measurement error is represented by a Gaussian distribution with zero mean and a covariance σ_{sonar}^2 . Although the sonar measures a distance of Z meter, the probability of the detected object being anywhere between $Z \pm \sigma_{sonar}$ m is very high. Figure 5.4 shows how the measurement error affects the particle filter.

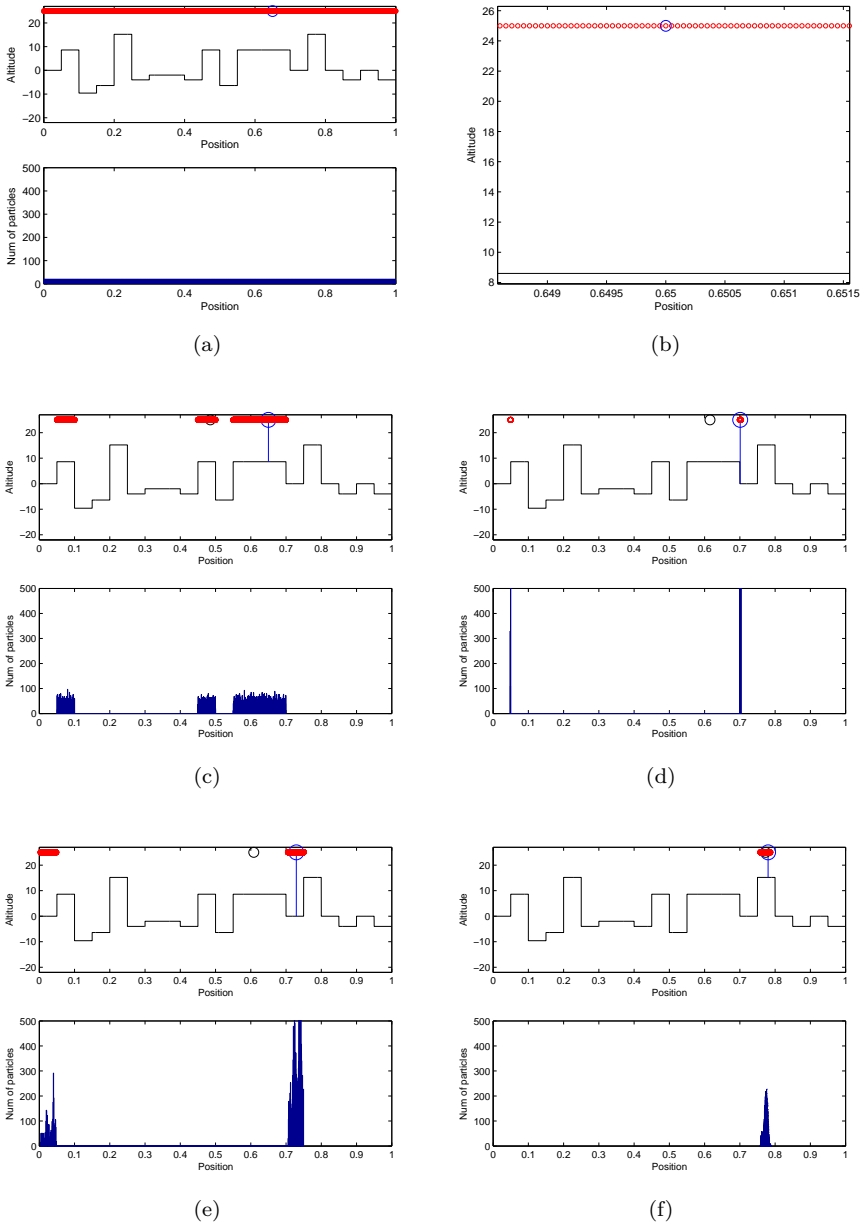


Figure 5.3: The procedure of the 1.5D PF. The big blue circle indicates the true position and the smaller red circles are the particles while the black medium size circle is the estimated position. The initialization step of the PF where the particles are spread out all over the area can be seen in (a). The lower figure in each sub figure is the histogram of the particle position, i.e an indication of how many particles there are in each region. The enlarged area around the true position is shown in (b). In each subfigure (c-f), the whole procedure of the PF algorithm (time update, measurement update, estimation, resampling) is included. The unit of the altitude and the position in the figure is meter.

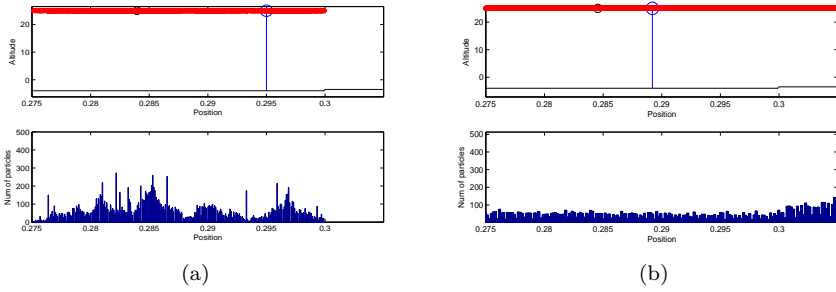


Figure 5.4: The ground level to the left of the knee at the position 0.3 is -4 m and -3.5 m to the right. An ideal sonar without any measurement noise has been used in (a) and one can clearly see how the particles do not propagate further than the knee at 0.3 m. In reality there is no ideal sonar which can measure the correct distance every time, although the sonar indicates a specific ground level it is not appropriate to fully trust this measurement and therefore one also have to include other levels within the measurement error of the sonar. In (b) the sonar has a standard deviation of 1 m and now there is also some probability of being at the ground level of -3.5 m when the sonar measures -4 m. The unit of the altitude and the position in the figure is meter

5.1.2 Localization in the 2D and 2.5D PF

A good and reliable simulation model can take a very long time to develop. Since the time of the thesis is quite limited and the simulations are not the main topic, the simplest solutions have once again been prioritized. In a first stage, the simulation environment is developed in Matlab and later on the simulations have been converted into C++ where Octomap is used. Since a two dimensional situation is a special case in the three dimensional room, a simulation environment with three dimensions has been developed which can be used for both the 2D and 2.5D PF when the altitude of the UAV is constant. The 2D case is the situation where the laser is attached to the UAV in such a way so it is measuring the distance to different objects in the same plane as the movement of the UAV, see Figure 5.5. In the 2.5D situation the laser has been rotated 90 degrees and is pointing downwards, Figure 5.6.

Line-plane intersection

Since the UAV is equipped with a laser, the line-plane intersection [16] will be a main part of the simulations since we assume the environment to be built up by flat objects. The parametric line equation can be represented as

$$\mathbf{l}_a + (\mathbf{l}_b - \mathbf{l}_a)t, \quad t \in \mathbb{R}, \quad (5.1)$$

where $\mathbf{l}_a = (x_a, y_a, z_a)$ and $\mathbf{l}_b = (x_b, y_b, z_b)$ are two points along the line. In the same way a plane can be represented as

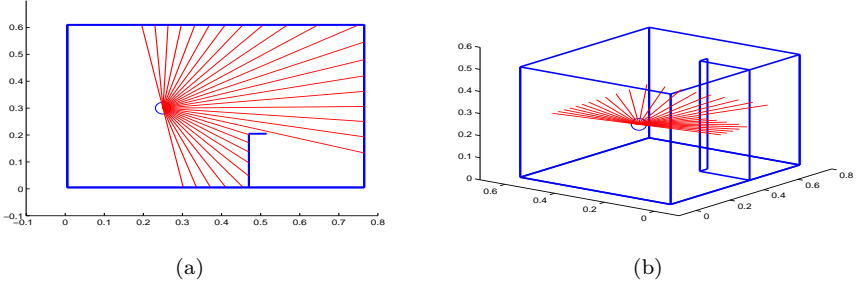


Figure 5.5: The 2D situation, one can see a top view in (a) while a 3D overview is shown in (b). The unit for all the axis is meter.

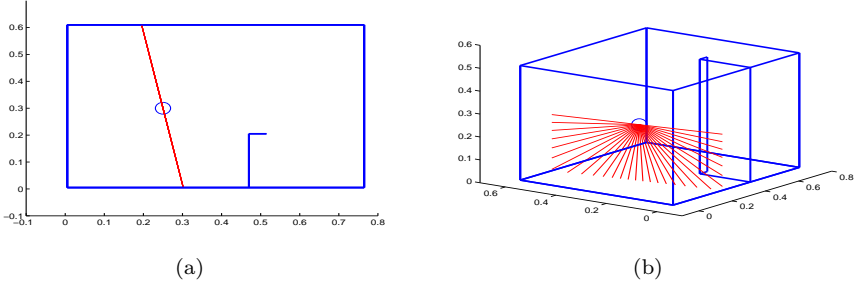


Figure 5.6: The 2.5D situation, one can see a top view in (a) while a 3D overview is shown in (b). The unit for all the axis is meter.

$$\mathbf{p}_0 + (\mathbf{p}_1 - \mathbf{p}_0)u + (\mathbf{p}_2 - \mathbf{p}_0)v, \quad u, v \in \mathbb{R} \quad (5.2)$$

where $\mathbf{p}_k = (x_k, y_k, z_k)$, $k = 0, 1, 2$ are three points in the plane which are linearly independent. Finding an intersection between the line and the plane can be done by setting the equations shown in (5.1) and (5.2) equal to each other,

$$\mathbf{l}_a + (\mathbf{l}_b - \mathbf{l}_a)t = \mathbf{p}_0 + (\mathbf{p}_1 - \mathbf{p}_0)u + (\mathbf{p}_2 - \mathbf{p}_0)v. \quad (5.3)$$

Simple algebra gives

$$\mathbf{l}_a - \mathbf{p}_0 = (\mathbf{l}_b - \mathbf{l}_a)t + (\mathbf{p}_1 - \mathbf{p}_0)u + (\mathbf{p}_2 - \mathbf{p}_0)v, \quad (5.4)$$

which can be expressed in matrix form as

$$\begin{bmatrix} x_a - x_0 \\ y_a - y_0 \\ z_a - z_0 \end{bmatrix} = \begin{bmatrix} x_a - x_b & x_1 - x_0 & x_2 - x_0 \\ y_a - y_b & y_1 - y_0 & y_2 - y_0 \\ z_a - z_b & z_1 - z_0 & z_2 - z_0 \end{bmatrix} \begin{bmatrix} t \\ u \\ v \end{bmatrix}. \quad (5.5)$$

Finally, the equations can be solved by inverting the matrix:

$$\begin{bmatrix} t \\ u \\ v \end{bmatrix} = \begin{bmatrix} x_a - x_b & x_1 - x_0 & x_2 - x_0 \\ y_a - y_b & y_1 - y_0 & y_2 - y_0 \\ z_a - z_b & z_1 - z_0 & z_2 - z_0 \end{bmatrix}^{-1} \begin{bmatrix} x_a - x_0 \\ y_a - y_0 \\ z_a - z_0 \end{bmatrix}. \quad (5.6)$$

From here it is possible to find the intersection between the line and the plane. If the line is parallel to the plane, the matrix in 5.6 will be singular and not invertible. The same situation will occur when the plane lies inside the plane.

Otherwise an intersection will be found, when the solution satisfies the condition $t \in [0, 1]$, the intersection point is located on the line between \mathbf{l}_a and \mathbf{l}_b . If the solution satisfies $u, v \in [0, 1]$, $(u + v) \leq 1$, then the intersection point is in the plane inside the triangle spanned by the three points $\mathbf{p}_0, \mathbf{p}_1$ and \mathbf{p}_2 .

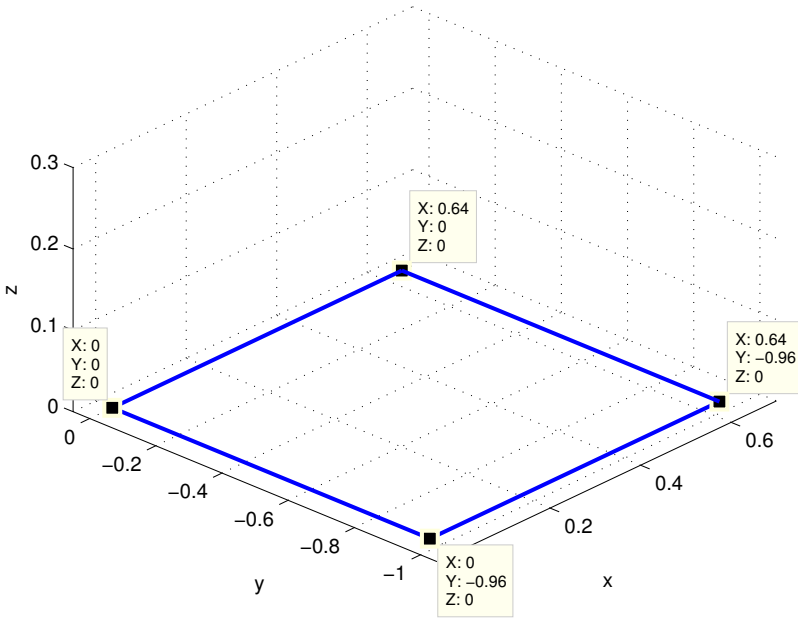


Figure 5.7: Every plane is defined by the points in each corner of the plane.

In the simulation model a plane is defined as the area between four coordinates in a three-dimensional space, Figure 5.7. The only solution used from equation (5.6) is the one given by t , i.e. u, v is not used since they will only indicate if the intersection point is located inside the area spanned by the three points $\mathbf{p}_0, \mathbf{p}_1$ and \mathbf{p}_2 . Using all three solutions for the matrix (t, u, v) would mean that every plane must consist of two triangles. Equation (5.6) could then be extended or used twice for each plane. Instead of this method an other approach has been used where the planes are defined by the coordinate in each corner of the plane. The intersection is still calculated using three points from the plane, but the map

which includes the planes have been expanded with further information about the length, width and height for each plane. This information makes it possible to see if the intersection between the line and plane ends up inside the valid area defined by the four coordinates.

Matlab simulations

As already mentioned, there are several plausible motion models to apply when estimating the dynamics of aerial vehicles. Without the IMU, it is reasonable to use the 2D CV model which already has been described in Section 3.1.2.

The fact that each particle has to simulate a couple of measurements and solve the line-plane intersection equation for every laser beam makes the simulations very time consuming. It is hard to evaluate the performance of the PF when the simulation time is long. Matlab is quite slow when running nested loops which more or less are impossible to avoid in a PF. To reduce the simulation time one can decrease the number of particles and the corresponding measurements but this will also create problems, because the PF will not converge to the true state if the number of particles is too small.

Simulations in C++

As already stated, simulations can be very time consuming if heavy systems are being tested and evaluated. Sometimes it is hard to do anything about the computational time because of the complex nature of the simulation environment. Past experience indicates that the simulation time of the PF can be decreased substantially when running the algorithms in C++ compared to Matlab. The structure and the idea behind the Matlab code from the simulations can be adapted and utilised in the real UAV platform at a later stage. For these two reasons, all the code from the Matlab simulations has been converted into C++. But Matlab will still be used to create measurements which is converted to an Octomap, see , Figure 5.8. The possibility to easily modify the code for the simulation model into real running algorithms will always be kept in mind.

Simulation results

Due to time restrictions, only evaluation of the 2D PF has been conducted. A small number of tests have been made on the 2.5D PF, but there is some more debugging needed before it can be evaluated for real and therefore it has not been analysed any further. The scanmatching algorithm has been tested with both simulated and real data to see if it is possible to use the information as odometer data. The results have been quite successful but the implementation of the scanmatching algorithm replacing an odometer in the PF has not been fully tested because of lack of time.

It is always hard to fully reproduce realistic motions and measurements. To make it easier, a CV model has been applied for simulating interesting movements and

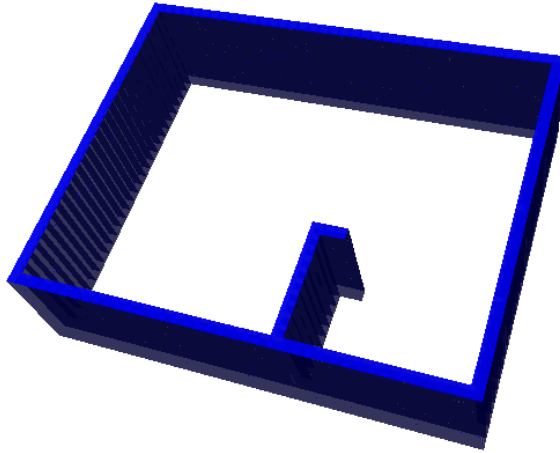


Figure 5.8: A visual representation of an Octomap. This map has been created under perfect circumstances by simulating measurements in Matlab from the map shown in Figure 5.5 and 5.5.

therefore illustrate the object which should be localized. In some simulations, measurement noise has been added to make the situation more realistic. The Gaussian distribution with the ad hoc solution can be seen in Figure 5.9 and 5.10 while the student's t-distribution is applied in Figure 5.11 and 5.12. By comparing the root mean square error (RMSE) for the position and the orientation, one can get a good indication of the performance. Matlab has only been used to visualise the results and have nothing to do with the simulations.

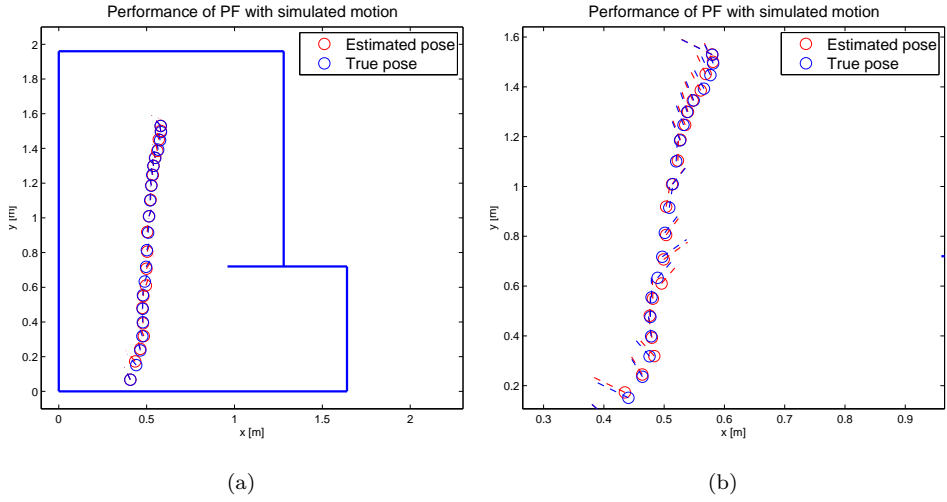


Figure 5.9: A simulation without measurement noise. Number of particles = 2000, position RMSE = 0.0070 m, orientation RMSE = 0.0068 rad. An overview of the simulation can be seen in (a) while a closer view of the trajectory is shown in (b).

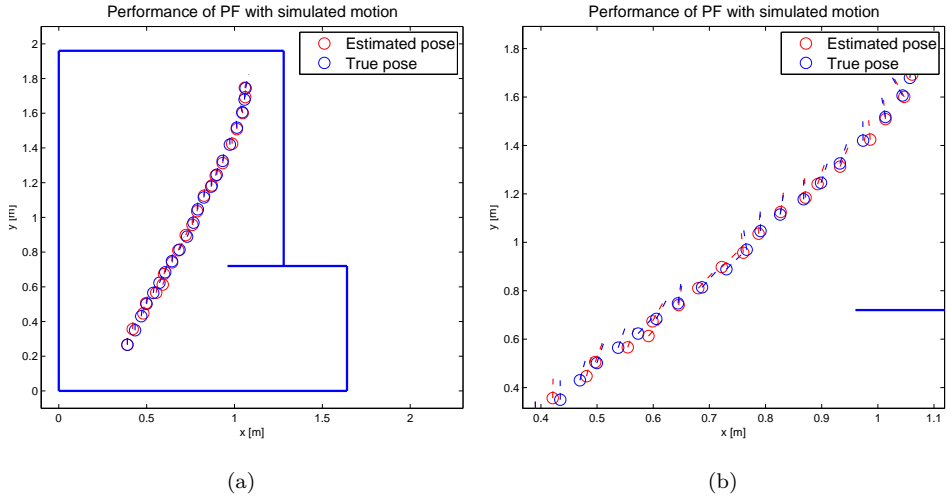


Figure 5.10: A simulation without measurement noise where the measurement model is Gaussian distributed. Number of particles = 2000, position RMSE = 0.0109 m, orientation RMSE = 0.0162 rad. An overview of the simulation can be seen in (a) while a closer view of the trajectory is shown in (b).

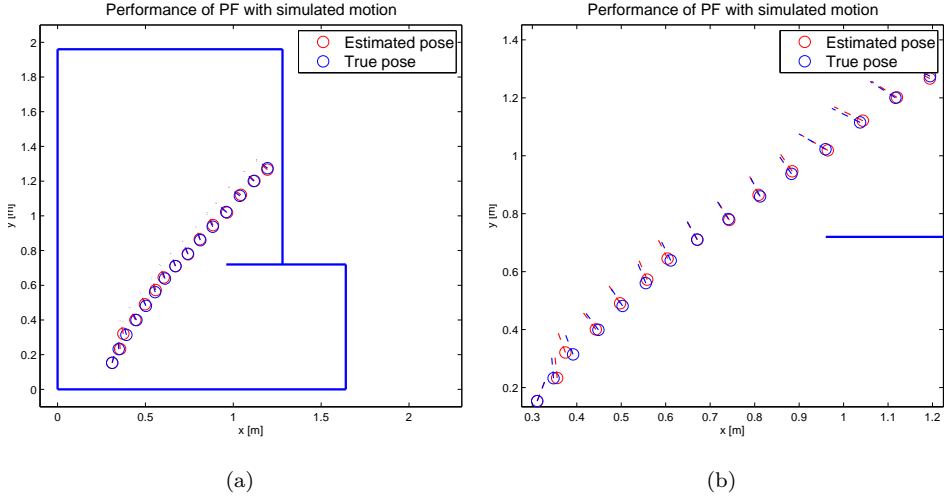


Figure 5.11: A simulation without measurement noise where the measurement model is student's t-distributed. Number of particles = 2000, position RMSE = 0.0075 m, orientation RMSE = 0.0070 rad. An overview of the simulation can be seen in (a) while a closer view of the trajectory is shown in (b).

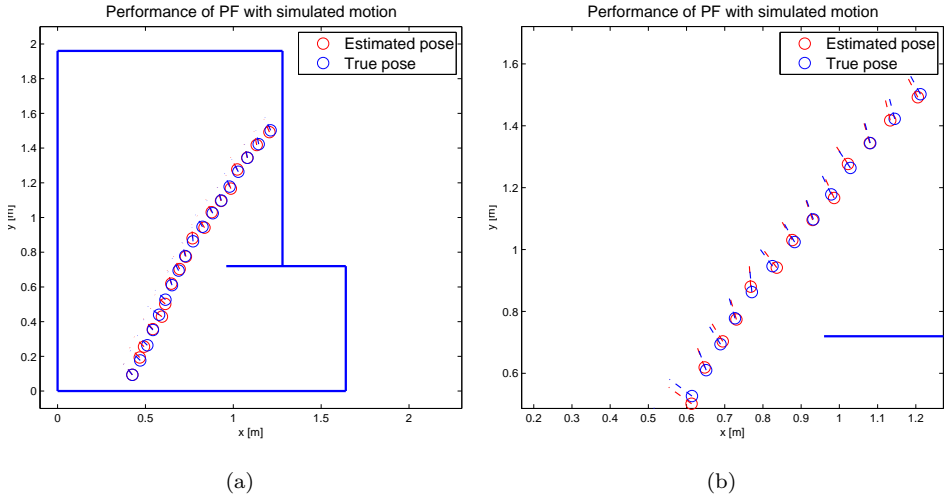


Figure 5.12: A simulation where measurement noise have been added to the laser, the noise is Gaussian distributed with zero mean and a standard deviation of 0.05 m. The measurement model is student's t-distributed. Number of particles = 2000, position RMSE = 0.0116 m, orientation RMSE = 0.0100 rad. An overview of the simulation can be seen in (a) while a closer view of the trajectory is shown in (b).

5.2 Evaluation of real data

Working with real data always implies challenges compared to simulated data where the circumstances, more or less, can be perfect. In all the test sessions, only the laser has been used and moved around in a 2D pattern, which means that the 3D situation with the complete UAV platform has not been evaluated. It has already been stated that there are some quantization problems with the Octomap. A real dataset from the laser and the corresponding Octomap with a grid resolution of 0.04m is shown in Figure 5.13. A comparison between real and simulated data can be seen in Figure 5.14.

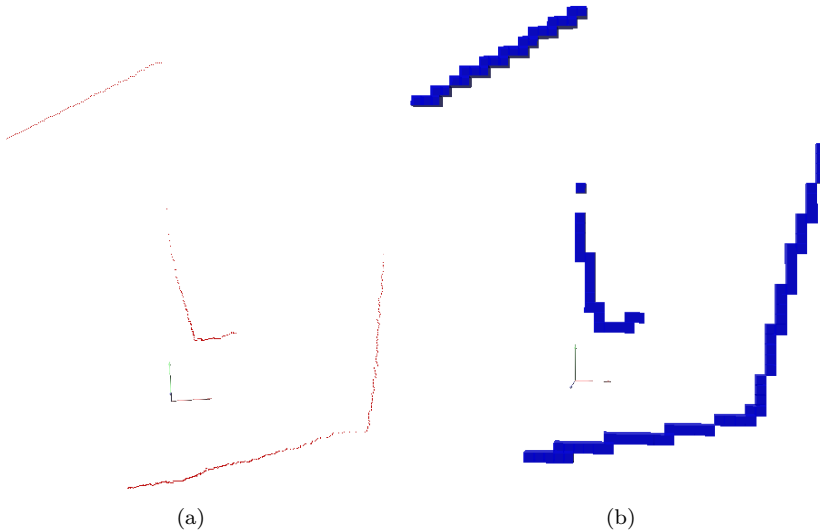


Figure 5.13: (a) Real measurements (682 points) from the laser and the corresponding Octomap is shown in (b). A small coordinate system is also included in both figures to represent the origin.

Although the measurements are simulated from exactly the same position as the one the real measurements were gathered from and later used for creating the octomap, there will be some differences between the datasets. The Hokuyo laser which has been used throughout the experimental sessions is very reliable in both precision and repeatability. Figure 5.13a has already shown the result from a stationary Hokuyo laser which has collected data for a couple of revolutions. By moving the laser and using the information from the motion capture system makes it possible to create an Octomap of a real world environment. The measurements from the motion capture system and the moving laser does not generate as good data as in the situation with a stationary laser. From Figure 5.15a one can clearly see the difference of the thick and blurred walls compared to the straight and thin walls generated by the stationary laser from Figure 5.13a.

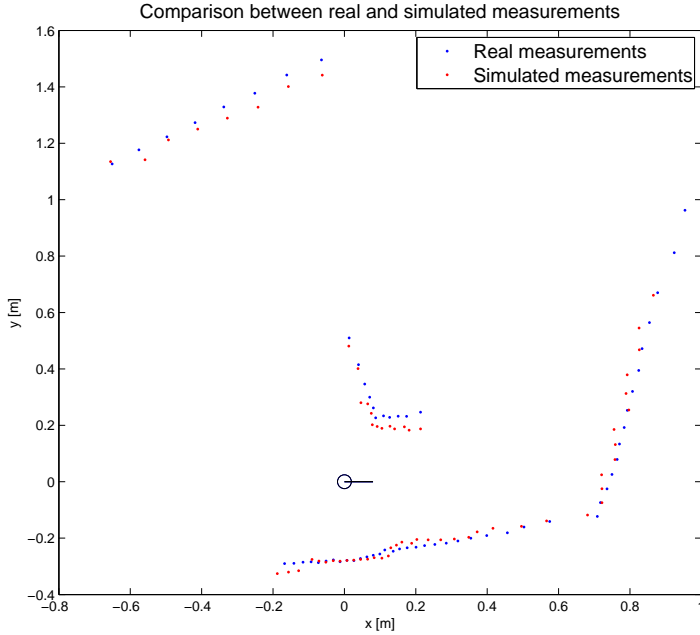


Figure 5.14: An illustration of the quantization problem which can occur in the Octomap. The measurements are simulated from exactly the same position as the one the real measurements were gathered from.

The problem with thick walls originates from the fact that the laser does not gather data fast enough. As already mentioned it is almost 67 ms between the first and the last measurement in the same scan. This might sound quite small but imagine the laser turning with an angular velocity of $\pi/4$ rad/s. Which is not unusual and it would mean that the laser has enough time to rotate almost $\pi/4 \text{ rad/s} \times 0.067\text{s} = 0.0526$ rad from its position where the first laser measurement was taken from to the last position. Still this can sound negligible, but remember that it can have a huge affect when the measured distance is quite large. A measured distance of three meter could for instance result in an error of $3 \sin(0.0526) = 0.157$ meters along the y-axis.

All these problems are something one have to consider when developing algorithms for localisation because it can be difficult to create reliable maps without any quantization errors or other assumptions.

Results

Three different sessions have been preformed in the VICON lab. The outcome from the sessions are quite similar to the one in Figure 5.15. A comparison of the performance has been done between a Gaussian distributed and a student's t-distributed measurement model. In each session, every tenth sample has been

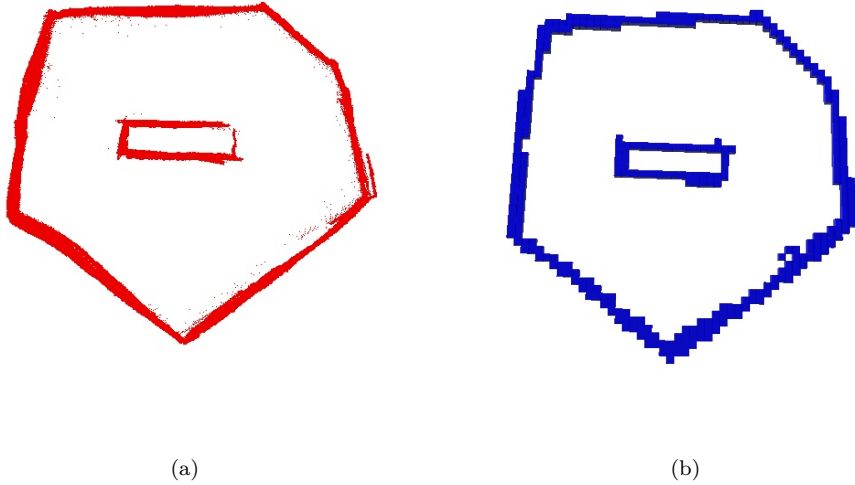


Figure 5.15: (a) is real measurements (682 points) from the laser and the corresponding Octomap is shown in (b).

used which corresponds to a sample time of approximately 1 s. It can be difficult to make any reliable conclusions regarding the RMSE value from only three sessions. To compensate for this, each session has been iterated 50 times to calculate a more reliable RMSE value. The results can be seen in Figure 5.16-5.18 and summarised in Table 5.1 and 5.2.

Parameters	Test 1	Test 2	Test 3
Distribution	Gaussian	Gaussian	Students's t
x_0	True state ¹	True state ¹	True state ¹
p_0	$(0.05 \ 0.05 \ 0 \ 0 \ 0.1)^T$	$(0.05 \ 0.05 \ 0 \ 0 \ 0.1)^T$	$(0.05 \ 0.05 \ 0 \ 0 \ 0.1)^T$
Q_t	$(0.04 \ 0.04 \ 0.25)^T$	$(0.04 \ 0.04 \ 0.25)^T$	$(0.04 \ 0.04 \ 0.25)^T$
R_t	0.01	0.01	-
λ	-	-	100
ν	-	-	1
N particles	2000	2000	2000
N meas.	62	62	62
Removed meas.	0	5	-

Table 5.1: A summary of the parameters which have been used in the evaluation of the PF on real data.

	Test 1	Test 2	Test 3
pRMSE [m], session 1	0.0890	0.0246	0.0274
oRMSE [rad], session 1	0.1392	0.0265	0.0320
pRMSE [m], session 2	0.0925	0.0317	0.0425
oRMSE [rad], session 2	0.1225	0.0370	0.0466
pRMSE [m], session 3	0.1371	0.0505	0.0448
oRMSE [rad], session 3	0.2294	0.0870	0.0878
pRMSE [m], all sessions	0.1062	0.0356	0.0382
oRMSE [rad], all sessions	0.1637	0.0484	0.0555

Table 5.2: The results from the test sessions. The best result in each session has been highlighted.

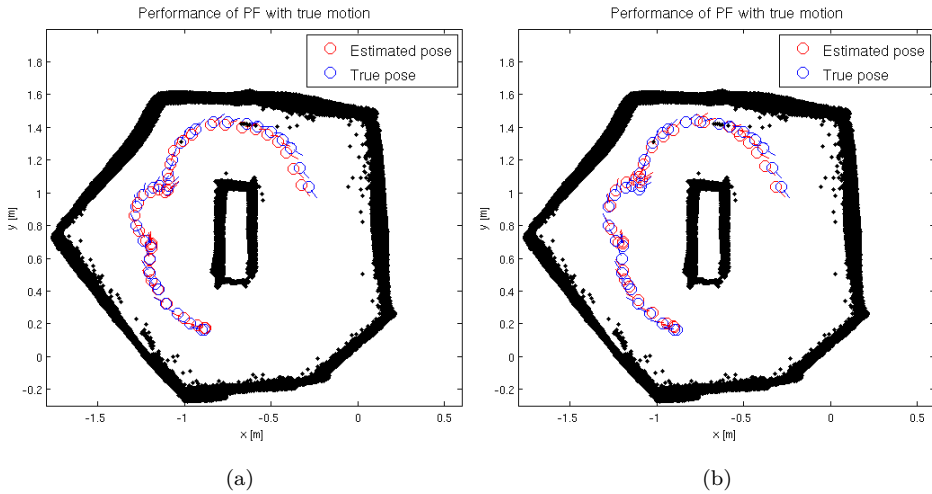


Figure 5.16: In (a) a Gaussian distributed measurement model has been used while the student's t-distribution has been applied in (b). The RMSE for the position and the orientation are 0.0246 m and 0.0265 rad for the Gaussian. The corresponding values for the student's t-distribution are 0.0274 m and 0.0321 rad.

¹It can seem very strange that x_0 is equal to the true state. If one does not have the possibility to choose x_0 close to the true state, one have to spread out the particles all over the area and increasing both p_0 and the number of particles to cover up all important states. In order to keep down the simulation time and the run time, x_0 has been chosen to the true state instead of increasing p_0 and the number of particles.

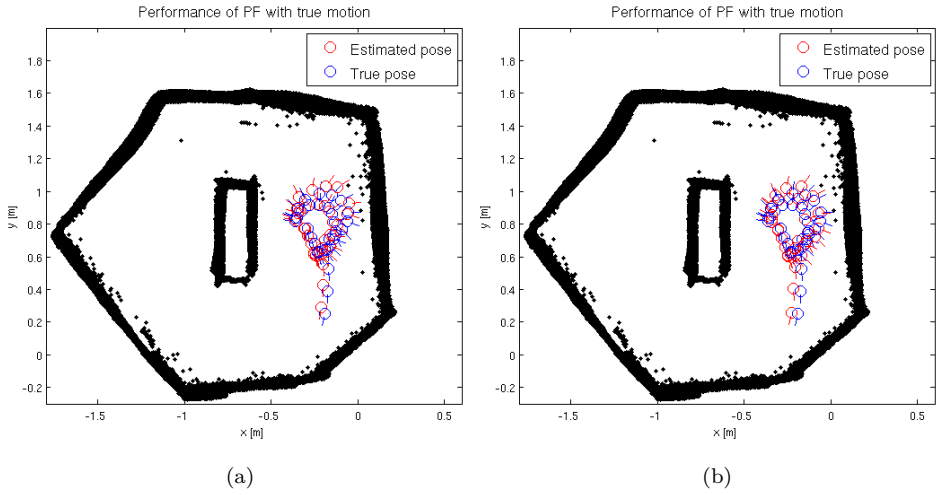


Figure 5.17: The Gaussian distribution applied in (a) and the student's t-distribution in (b). The position RMSE is 0.0317 m and the orientation RMSE is 0.0370 rad for the Gaussian distribution and. The corresponding values for the student's t-distribution are 0.0425 m and 0.0466 rad.

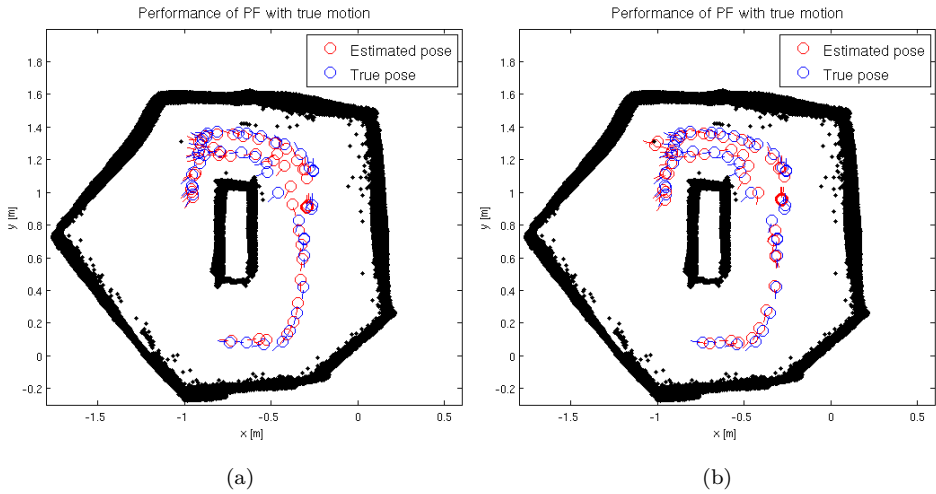


Figure 5.18: The third session does not generate as good results as the other two. The Gaussian distribution is shown in (a) and the student's t-distribution in (b). The RMSE for the orientation is quite poor compared to the other results. The explanation to this behaviour is unknown. The RMSE for the position and the orientation is 0.0505 m and 0.0870 rad for Gaussian distribution. The corresponding values for the student's t-distribution are 0.0448 m and 0.0878 rad.

Chapter 6

Conclusion and future work

The research field for UAVs is enormous and there are a lot of things to consider when working within this area. The main goal of the thesis has been to develop reliable algorithms in a 3D framework. To reach this goal, one should first have full understanding of the 2D case and then adopt these algorithms and principles when increasing the degree of freedom with another dimension. Unfortunately a lot of time has been spent on the 2D case resulting in lack of time when evaluating the more interesting 3D case. It has been quite obvious that the measurement equation and the corresponding distribution is very crucial in these kind of situations. According to the experimental results, it seems like the most reliable measurement model is the one where the Gaussian distribution together with the ad hoc solution has been used. It can be possible to improve the results for the student's t-distribution by tuning its parameters.

As already mention in Section 1.2-1.4, few have reported the possibility of performing SLAM in 3D for indoor UAV's only using IMU and a 2D laser. The initial objective of this thesis was to develop some kind of offline-SLAM algorithm based on scanmatching to create a 3D map of the environment. After a couple of weeks, it was clear that it would take a lot of time to generate robust scanmatching algorithms. I still think this would be possible to do and very useful in the future development of UAVs. You may wonder why bother with a 2D laser and complex algorithms at all when a camera can simplify everything? Obviously there are situations where the camera could work perfectly but there is also situations where the camera does not work at all, for instance in dark rooms. From the very beginning of this thesis, I have always thought of how useful a UAV with a laser could be for a search and rescue session inside a burning building filled with thick smoke which makes it impossible to use a regular camera.

Both the Gaussian and the student's t-distribution would probably work better if one could guarantee an Octomap which represents the real world with high accuracy. In reality this is not possible, instead one has to compensate with robust algorithms which can handle the problem of using an non-perfect Octomap. A solution which could work much better for these kinds of situations is probably a feature detection algorithm. Feature algorithms are not very sensitive to outliers and model errors which descends from the Octomap. Although the map is not perfect, one can still detect lines, corners and other features in a reliable way and use this information for localisation. The major reason for not evaluating feature detection is because of its complex nature which probably would require a lot of time before effective algorithms can be developed. Another interesting approach would be to use multimodal Gaussian distribution. The key idea is to include some uncertainty in the angle of the emitted laser beam and then make use of some multimodal distribution which can cover up the situation when the laser detects an object while the simulated measurement misses the object with some centimetres and detects something behind, like the situation in Figure 3.5. In this way it would be possible to make use of the information to improve the estimations instead of seeing the measurement as an outlier and removing it.

There are also a lot of things to improve regarding the PF. First of all it could be mentioned that the optimal proposal density is given by $q(x_t|x_{t-1}^i, y_t)$ and would of course produce better result because the current measurement y_t is now considered when new particles are generated. The PF can be very time consuming to calculate because when the dimension of the states space vector increases, the number of particles will also increase as a consequence of this. A solution to this problem is the Rao-Blackwellized Particle Filter (also known as the Marginalized Particle Filter) which utilizes a method where the states are separated in a linear and a non-linear part.

Bibliography

- [1] M. Achtelika, A. Bachrachb, R. Heb, S. Prenticeb, and R. Royb. Autonomous navigation and exploration of a quadrotor helicopter in GPS-denied indoor environments. In *First Symposium on Indoor Flight Issues (IARC)*, pages 1–12, University of Puerto Rico, Mayagüez, Puerto Rico, 2009.
- [2] S. Bouabdallah and R. Siegwart. Full control of a quadrotor. In *Proceedings of the 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 153–158, Kobe, Japan, October/November 2007.
- [3] C. Brenneke, O. Wulf, and B. Wagner. Using 3D laser range data for slam in outdoor environments. In *Proceedings of the IEEE/RSJ international conference on Intelligent Robots and Systems*, volume 1 of *IROS'03*, pages 188 – 193 vol.1, Las Vegas, Nevada USA, October 2003.
- [4] I. Dryanovski, W. Morris, and J. Xiao. Multi-volume occupancy grids: An efficient probabilistic 3d mapping model for micro aerial vehicles. In *Proceedings of 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1553 –1559, Taipei, Taiwan, October 2010.
- [5] S. Grzonka, G. Grisetti, and W. Burgard. Towards a navigation system for autonomous indoor flying. In *Proceedings of the 2009 IEEE international conference on Robotics and Automation, ICRA'09*, pages 1679–1684, Piscataway, NJ, USA, 2009.
- [6] F. Gustafsson. *Statistical Sensor Fusion*. Studentlitteratur AB, 2010. ISBN 978-91-44-05489-6.
- [7] J. L. Martinez, J. Gonzalez, J. Morales, A. Mandow, and A. J. Garcia-cerezo. Genetic and ICP laser point matching for 2d mobile robot motion estimation.
- [8] B. C. Min, C. H. Cho, K. M. Choi, and D. H. Kim. Development of a micro quad-rotor uav for monitoring an indoor environment. In *Proceedings of the FIRA RoboWorld Congress 2009 on Advances in Robotics*, pages 262–271, Berlin, Heidelberg, 2009.
- [9] P. Pounds, R. Mahony, and P. Corke. Modelling and control of a quad-rotor robot. In *Proceedings Australasian Conference on Robotics and Automation 2006*, Canberra, Australia, December 2006.

- [10] J. F. Roberts, T. Stirling, J.C. Zufferey, and D. Floreano. Quadrotor Using Minimal Sensing For Autonomous Indoor Flight. In *European Micro Air Vehicle Conference and Flight Competition (EMAV2007)*, Lausanne, Switzerland, September 2007.
- [11] W. Burgard D. Fox S. Thrun. *Probabilistic Robotics*. MIT press, 2006. pp. 94-96, 281-305.
- [12] H. Choset K. M. Lynch S. Hutchinson G. Kantor W. Burgard L. E. Kavraki S. Thrun. *Principles of Robot Motion*. MIT press, 2005. pp. 322-337.
- [13] S. Thrun, Mark Diel, and D. Hähnel. Scan alignment and 3D surface modeling with a helicopter platform. In *Proceedings of the International Conference on Field and Service Robotics*, Lake Yamanaka, Japan, 2003.
- [14] D. Törnqvist. *Estimation and Detection with Applications to Navigation*. Linköping studies in science and technology. dissertations. no. 1216, Linköping University, Nov 2008.
- [15] Wikipedia. Conditional probability. http://en.wikipedia.org/wiki/Conditional_probability. [Online; accessed June 2011].
- [16] Wikipedia. Line-plane intersection. http://en.wikipedia.org/wiki/Line-plane_intersection. [Online; accessed June 2011].
- [17] Wikipedia. Octree. <http://en.wikipedia.org/wiki/File:Octree2.svg>. [Online; accessed August 2011].
- [18] O. Wulf, K.O. Arras, H.I. Christensen, and B. Wagner. 2d mapping of cluttered indoor environments by means of 3d perception. In *Proceedings of the ICRA '04. 2004 IEEE International Conference on Robotics and Automation*, pages 4204 – 4209 Vol.4, New Orleans, LA, USA, May 2004.
- [19] K. M. Wurm, A. Hornung, M. Bennewitz, C. Stachniss, and W. Burgard. OctoMap: A probabilistic, flexible, and compact 3D map representation for robotic systems. In *Proceedings of the ICRA 2010 Workshop on Best Practice in 3D Perception and Modeling for Mobile Manipulation*, Anchorage, AK, USA, May 2010.