

Institutionen för systemteknik

Department of Electrical Engineering

Examensarbete

Airborne mapping using LIDAR

Examensarbete utfört i Reglerteknik
vid Tekniska högskolan i Linköping
av

Erik Almqvist

LiTH-ISY-EX--10/4374--SE

Linköping 2010



Linköpings universitet
TEKNISKA HÖGSKOLAN

Airborne mapping using LIDAR

Examensarbete utfört i Reglerteknik
vid Tekniska högskolan i Linköping
av


Erik Almqvist

LiTH-ISY-EX--10/4374--SE

Handledare: **Karl Granström**
isy, Linköpings universitet
Magnus Sethson
CybAero AB

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

Linköping, 20 August, 2010

	Avdelning, Institution Division, Department Division of Automatic Control Department of Electrical Engineering Linköpings universitet SE-581 83 Linköping, Sweden	Datum Date 2010-08-20				
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--10/4374--SE Serietitel och serienummer ISSN Title of series, numbering _____				
URL för elektronisk version http://www.control.isy.liu.se http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-58866						
<table style="width: 100%;"> <tr> <td style="width: 15%;">Titel</td> <td>Luftburen kartering med LIDAR</td> </tr> <tr> <td>Title</td> <td>Airborne mapping using LIDAR</td> </tr> </table> Författare Erik Almqvist Author			Titel	Luftburen kartering med LIDAR	Title	Airborne mapping using LIDAR
Titel	Luftburen kartering med LIDAR					
Title	Airborne mapping using LIDAR					
Sammanfattning Abstract <p>Mapping is a central and common task in robotics research. Building an accurate map without human assistance provides several applications such as space missions, search and rescue, surveillance and can be used in dangerous areas. One application for robotic mapping is to measure changes in terrain volume. In Sweden there are over a hundred landfills that are regulated by laws that says that the growth of the landfill has to be measured at least once a year.</p> <p>In this thesis, a preliminary study of methods for measuring terrain volume by the use of an Unmanned Aerial Vehicle (UAV) and a Light Detection And Ranging (LIDAR) sensor is done. Different techniques are tested, including data-merging strategies and regression techniques by the use of Gaussian Processes. In the absence of real flight scenario data, an industrial robot has been used for data acquisition. The result of the experiment was successful in measuring the volume difference between scenarios in relation to the resolution of the LIDAR. However, for more accurate volume measurements and better evaluation of the algorithms, a better LIDAR is needed.</p>						
Nyckelord Keywords LIDAR, Gaussian Processes, non-stationary Gaussian Processes, 2.5D-mapping, UAV, known poses						

Abstract

Mapping is a central and common task in robotics research. Building an accurate map without human assistance provides several applications such as space missions, search and rescue, surveillance and can be used in dangerous areas. One application for robotic mapping is to measure changes in terrain volume. In Sweden there are over a hundred landfills that are regulated by laws that says that the growth of the landfill has to be measured at least once a year.

In this thesis, a preliminary study of methods for measuring terrain volume by the use of an Unmanned Aerial Vehicle (UAV) and a Light Detection And Ranging (LIDAR) sensor is done. Different techniques are tested, including data-merging strategies and regression techniques by the use of Gaussian Processes. In the absence of real flight scenario data, an industrial robot has been used for data acquisition. The result of the experiment was successful in measuring the volume difference between scenarios in relation to the resolution of the LIDAR. However, for more accurate volume measurements and better evaluation of the algorithms, a better LIDAR is needed.

Sammanfattning

Kartering är ett centralt och vanligt förekommande problem inom robotik. Att bygga en korrekt karta av en robots omgivning utan mänsklig hjälp har en mängd tänkbara användningsområden. Exempel på sådana är rymduppdrag, räddningsoperationer, övervakning och användning i områden som är farliga för människor. En tillämpning för robotkartering är att mäta volymökning hos terräng över tiden. I Sverige finns det över hundra soptippar, och dessa soptippar är reglerade av lagar som säger att man måste mäta soptippens volymökning minst en gång om året.

I detta exjobb görs en undersökning av möjligheterna att göra dessa volymberäkningar med hjälp av obemannade helikoptrar utrustade med en Light Detection and Ranging (LIDAR) sensor. Olika tekniker har testats, både tekniker som slår ihop LIDAR data till en karta och regressionstekniker baserade på Gauss Processer. I avsaknad av data inspelad med riktig helikopter har ett experiment med en industrirobot genomförts för att samla in data. Resultaten av volymmätningarna var goda i förhållande till LIDAR-sensors upplösning. För att få bättre volymmätningar och bättre utvärderingar av de olika algoritmerna är en bättre LIDAR sensor nödvändig.

Acknowledgments

I would like to thank Magnus Sethson, Andreas Gising, Johan Mårtensson and the others at CybAero AB in Linköping for the opportunity of doing my thesis on a technically inspiring subject. I would also like to thank Thomas Schön and Karl Granström at ISY for helping with both technical answers as well as the report. Further, Henrik Ohlsson at ISY for answering questions about Gaussian Processes, and Tobias Lang for sharing his knowledge about the non-stationary implementation. Lars Andersson and Peter Nordin at IEI have provided great help with the experimental setup and hardware. Sebastian Ljungberg for good critique on my written work and presentation. Fredrik Bodesund for long conversations during late hours and nice coffee breaks. In general, thanks to everyone showing a happy face during this work!

Last, but not least. The biggest Thanks goes to my wife Liseth, my family and my friends for supporting me throughout the years at LiU.

Contents

1	Introduction	1
1.1	Problem formulation	1
1.2	Thesis objective	2
1.3	Related work	2
1.4	CybAero	3
1.5	Outline of thesis	4
2	Sensors and Pre-Processing	7
2.1	Pose information	7
2.2	Geometry	8
2.3	The LIDAR sensor	9
2.3.1	Sensor uncertainties	10
3	Laser mapping	13
3.1	2.5D and grid mapping	13
3.2	Initial mapping methods	13
3.3	Gaussian Processes	16
3.3.1	Background	16
3.3.2	Training a Gaussian Process	19
3.3.3	Applying a Gaussian Process	20
3.3.4	Visualisation	21
3.3.5	Local windowing	22
3.4	Non-stationary Gaussian Processes	23
3.4.1	The elevation structure tensor	25
3.4.2	Finding the local kernels	27
3.4.3	Estimating the kernel at unseen locations	29
4	Experimental results	31
4.1	Artificial data set	31
4.2	Experiments using an industrial robot	33
4.2.1	Data collection	33
4.2.2	LIDAR calibration experiments	35
4.3	Scenario one - blocks	38
4.4	Scenario two - natural terrain	41
4.5	Discussion	44

5	Concluding remarks	47
5.1	Conclusions	47
5.2	Future work	48
	Bibliography	51
A	Quaternions	53
A.1	Rotation using quaternions	54
B	Abbreviations	55

Chapter 1

Introduction

Mapping is a central and common task in robotics research. Building an accurate maps without human assistance provides for several applications such as space missions, search and rescue, surveillance and can be used in dangerous areas. This work focuses on the application of measuring volume difference using an airborne robot. This first chapter provides an introduction to the problem and the purpose and structure of the thesis.

1.1 Problem formulation



Figure 1.1: An Unmanned Aerial Vehicle measuring volume at a landfill.

There are hundreds of landfills in Sweden. Landfills are currently regulated by restrictions from Naturvårdsverket that says that all landfills in Sweden have to measure their growth at least once a year for as long as the landfill is active. The growth should be measured by volume or height [11]. Today this kind of landfill measurement is performed once a year by a person manually placing a GPS at known positions on the landfill. This job is far from safe, and accidents have happened in the past, making this a question of safety. There is also the question of accuracy: the resolution of the measurement made by foot is around ten GPS measurements per hectare. One of the questions that this thesis is trying to answer is whether this can be done with higher accuracy using an Unmanned Aerial Vehicle (UAV). The idea is to use sensor fusion to solve this problem, placing the required sensors on the UAV and merging the acquired sensor information into an accurate map. This work evaluates different methods for creating maps from raw data acquired

by a LIDAR. The position of the UAV is assumed to be known in this work.

Since all sensors are more or less affected by noise, so also laser range finders, the need for methods that handles uncertainties and sensor incompleteness is desirable. In search for such a method this work has focused on the use of Gaussian Processes (GP). Gaussian Processes provides a probabilistic approach to mapping, which is the dominant approach for mapping using robotics as it provides ways of dealing with uncertainties. As a preliminary study before the flights with a real UAV an experimental setup using an industrial robot is used for data collection.

1.2 Thesis objective

The objective of this work is to find methods and algorithms for merging information from a LIDAR sensor with known sensor poses into a topological map, and to evaluate the performance of these algorithm for the purpose of measuring changes in terrain volume.

1.3 Related work

The problem of mapping has previously been solved using a variety of algorithms and with several different types of maps. Difficulties in robotic mapping include how to handle uncertainties and sensor incompleteness in a good way. The amount of probabilistic approaches to the mapping problem has grown significantly as computational power has increased the last decades. Probabilistic methods provide ways of handling the sensor limitations, hence their popularity [18]. Along with the choice of mapping method, there is also a number of methods for storing and representing the terrain, all contributing to the wide array of methods available [16].

Triangulated Irregular Networks (TIN) is commonly used in computer games to create terrain, but is also used for mapping purposes. Leal, Scheding et al. uses TINs and a stochastic mapping approach to model terrain [8]. TIN provides a meshed model of the terrain, and does not require an equidistant map. Therefore TIN maps are theoretically able express very fine grained terrain models in comparison to many other methods. Approaches using TIN do have problems with scalability as the number of so called network nodes grows significantly when the mapped areas gets larger. In computer gaming such issues can be handled by dividing the TIN network using binary space partitioning that reduces the amount of active TIN nodes, but that is not a valid simplification in a mapping application.

There is a great deal of applications that utilizes various kinds of Kalman filters to handle the uncertainties of the sensors. Examples includes Kleiner and Dornhege who use a Kalman like filter together with a convolution kernel to reduce the effect of missing data and erroneous range information [4]. Cremean and Murray implements a Simoultaneous Localization and Mapping (SLAM) algorithm using a Digital Elevation Map and a Kalman filter for mapping using a ground vehicle

[1]. Cremean and Murray have real time requirements on their implementation as well as requirements on scalability (as it is a high speed outdoor application), and make different simplifications in order to meet these requirements. For example, a 2.5D relaxation is used in order to limit computations.

Another popular mapping method when mapping is Occupancy Grid Mapping (OGM). OGM provides a probabilistic approach to the problem of mapping using Bayesian filtering. The map is divided into cells and the probability of each cell being occupied is calculated [16]. The sum of the probability of each cell is given as a measurement for how probable the whole map is. The number of variables (cells) that need to be evaluated in an OGM approach increases significantly when going from two dimensions to three dimensions, hence OGM traditionally has not been well suited for large scale 3D mapping. Recently, Wurm, Hornung et al. presented an extension to the OGM called Octomaps [23]. The Octomap approach builds a tree based structure which leads to good memory representation even in three dimensions. The result of Octomaps have looked promising both in simulations and real-world experiments.

Gaussian processes (GP) builds on the same Bayesian framework as OGM. GPs have been studied for mapping purposes by Vasudevan, Ramos et al. [22] who uses Gaussian processes together with KD-trees for increased scalability to produce elevation maps with uncertainty information, primarily by using data provided by the mining industry. They found the use of a non-stationary neural network kernel to be a good way of mapping large areas with rough terrain. Gaussian processes are also used by Lange, Plagemann et al [7], who use the same basic approach of non-stationary kernels, but instead look into the computer vision community to find ways of creating local kernels. A very good overview of the theory of GP for machine learning is given by Carl Rasmussen who covers the basic theory as well as kernel choices and more [14].

Aerial mapping has had a later development due to the increased complexity (increased number of freedoms, payload limitations and tougher dynamics) of flying vehicles compared to ground vehicles. Despite this, several solutions have been proposed. Thrun, Hahnel and Diel implemented a method for 3D-modeling using Bayes filtering and a fast optimization technique to map using a manually controlled non traversing helicopter [20]. Grzonka, Grisetti and Burgard [2] performed indoor mapping using a quad-rotor flying UAV using particle filters. A node based approach was used for solving the mapping part of the SLAM problem. As indoor flying implies heavy restrictions on payloads for the helicopter, the same laser was used for both mapping and altitude estimation by using a mirror.

1.4 CybAero

CybAero AB develops and manufactures Vertical Take Off and Landing (VTOL) Unmanned Aerial Vehicles (UAV), i.e. unmanned helicopters. One of their VTOL

UAVs can be seen in Figure 1.2. A VTOL UAV has several applications, both civil and military. One such application is to survey the ground below the UAV and produce a topographic map. Military use for a mapping application includes surveillance missions or mine detection. Civil applications includes good ways to get an overview of disaster areas or to measure the volume of landfills. One of CybAeros possible partners for the application of measuring volume differences is Tekniska Verken. Tekniska Verken is a regional company which aims to create a community which is sustainable in the long term. Waste management is a part of their care.

CybAero started the process of measuring landfill volume using UAV's during the spring of 2010. As a part of this two master theses have been carried out at CybAero during the spring of 2010. One focusing on estimating the position and orientation of the helicopter, and this thesis that focuses on laser mapping methods.



Figure 1.2: One of the Vertical Take Off and Landing (VTOL) Unmanned Aerial Vehicles (UAV) developed by CybAero AB.

1.5 Outline of thesis

The remainder of this work is outlined as follows.

Chapter 2 Includes necessary theoretical background information. This includes information on the laser range finder sensor and how to transform data from the sensor into a usable raw data set.

Chapter 3 Provides information about the laser mapping algorithms that were implemented.

Chapter 4 Presents the results of experiments and the performance of the implementations.

Chapter 5 Presents conclusions of the work, and possible future work.

Appendix A Contains information on quaternion rotation, and a section with abbreviations and technical terms.

Chapter 2

Sensors and Pre-Processing

This chapter introduces the state vector and covers the basic relations for merging the data from the sensors into raw elevation data. It also contains a description of the sensors used, specially focusing on the laser range scanner as it is the primary sensor for this work.

2.1 Pose information

To describe a moving body two coordinate systems, which can be seen in Figure 2.1, is used. The first coordinate system is fixed in the world (earth) and is referred to as \mathbf{x}_{wc} (*wc* for world coordinate). The second coordinate system moves with the vehicle system, and is referred to as \mathbf{x}_{sp} (*sp* for sensor platform). The position and orientation of the moving coordinate system, \mathbf{x}_{sp} , is assumed to be provided by some external estimation procedure. The state vector consists of the following states:

$$[\hat{\mathbf{x}} \quad \mathbf{q}]^T = [x \quad y \quad z \quad q_0 \quad q_1 \quad q_2 \quad q_3]^T, \quad (2.1)$$

where $\hat{\mathbf{x}} = [x \ y \ z]$ denotes the position of a known point in the moving frame \mathbf{x}_{sp} from which the mounting position of the LIDAR is known, and $\mathbf{q} = [q_0, \dots, q_3]$ is a quaternion that denotes the orientation of this known point. Quaternions is a generalization of the complex plane to three dimensions that is frequently used to describe orientation. It is superior to the Euler angle representation as it does not suffer from the problems of singularities in the same way as Euler angles do (this is one of the reasons that the quaternion representation has been chosen as interface between the two parallel master theses).

The first unit of the quaternion, q_0 , describes the magnitude of the rotation around an axis. The axis is described by the remaining units of the quaternion, $q_1 - q_3$. The quaternion is normalized so that its absolute value always sums up to one. See Appendix A for more information about quaternions, and how the quaternions is used to describe orientation. The robot is assumed to be a rigid body, and thus



Figure 2.1: Two coordinate systems are used to describe a moving vehicle. The coordinate system \mathbf{x}_{sp} moves with the vehicle, while \mathbf{x}_{wc} is fixed in the world.

the orientation of the known point on the body is the same as the orientation of the LIDAR. The states given in (2.1) are provided at approximately 100 HZ.

2.2 Geometry

The data collected by a LIDAR sensor placed on a moving vehicle forms a geometrical problem. The gathered raw data needs to be transformed into information about the world around it. In order to project the LIDAR measurements into the world forward kinematics can be used [4]. The following section explains how linear algebra can be used to describe the relationship between the moving coordinate system, \mathbf{x}_{sp} , placed at a known location at the moving vehicle, and the world fixed coordinate system \mathbf{x}_{wc} .

Let O denote the origin of \mathbf{x}_{wc} , and R be the origin of \mathbf{x}_{sp} . Further, let L be the placement of the LIDAR sensor within \mathbf{x}_{sp} , and S be a range measurement made by the LIDAR in \mathbf{x}_{sp} according to Figure 2.2. The measurements made by the LIDAR in \mathbf{x}_{wc} can then be expressed as the vector

$$\overline{OS}_{wc} = \overline{OR}_{wc} + \overline{RS}_{wc}. \quad (2.2)$$

The relations in (2.2) have to be expressed in the same coordinate system, the fixed world coordinate system \mathbf{x}_{wc} . The vector \overline{RS} between the origin of \mathbf{x}_{sp} and the LIDAR measurement can be seen as an addition between the vectors \overline{RL} and \overline{LS} in \mathbf{x}_{sp} . This can be related to world coordinates by applying a rotation matrix $R(\mathbf{q})$ to (2.2) that expresses how \mathbf{x}_{sp} is rotated with respect to \mathbf{x}_{wc} . The rotation matrix is provided by the quaternion \mathbf{q} (See Appendix A). Using $R(\mathbf{q})$ a range measurement made by the LIDAR expressed in world coordinates is given by (2.3). The vector \overline{RL} is a translation between the known origin of the moving frame and the mounting point of the LIDAR. This distance is measured by hand. The vector \overline{LS} is the measurement made by the LIDAR as explained in the next section.

$$\overline{OS}_{wc} = \overline{OR}_{wc} + R(\mathbf{q}) (\overline{RL}_{sp} + \overline{LS}_{sp}) \quad (2.3)$$

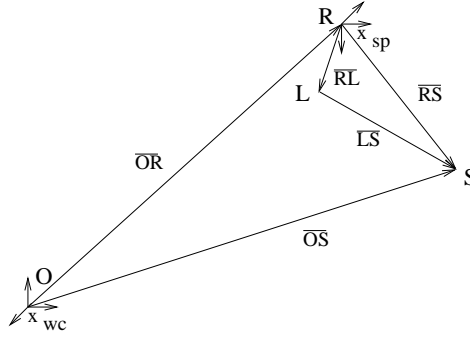


Figure 2.2: Principles of the geometry. By the use of translations and a rotation matrix the measurements made by the LIDAR in the moving coordinate system \mathbf{x}_{sp} can be expressed in the fixed world coordinate system, \mathbf{x}_{wc} .

2.3 The LIDAR sensor

A LIDAR sensor is used to measure the environment in this work, and this section contains basic information about the sensor. The LIDAR used is a Hokuyo URG-04-LX, see Figure 2.3. The LIDAR provides range measurements by emitting



Figure 2.3: The Hokuyo URG-04-LX sensor.

laser pulses and detecting the pulses reflected from nearby objects. Like a radar, distance to the object is determined by measuring the time delay between the transmission of a pulse and the detection of the reflected signal. The URG-04-LX has a 240° field of view and has an angular resolution of about 0.36° . It has an operation range of zero to four meters. The width of the laser cone grows with increasing range, and the observed range corresponds to an arbitrary range within the width of the cone [4]. The LIDAR operates at 10 HZ and collects laser data in sweeps, each with N laser measurements. The URG-04-LX does not provide information on intensity of the pulses. A laser measurement, L_i , each consists of a range measurement r_i and an angular measurement ϕ_i as

$$L_i = \begin{bmatrix} r_i \\ \phi_i \end{bmatrix}, \quad i = 1, \dots, N. \quad (2.4)$$

For the particular mapping application in this work, the whole field of view is not significant as measurements only provides information when hitting the ground. Hence the field of view has been limited to $[\phi_{\min}, \phi_{\max}]$. The range and bearing measurements from the LIDAR can be transformed into Cartesian coordinates by the following transform,

$$\begin{bmatrix} x_i \\ y_i \end{bmatrix} = \begin{bmatrix} r_i \cos(\phi_i) \\ r_i \sin(\phi_i) \end{bmatrix}, \quad i = 1, \dots, N. \quad (2.5)$$

2.3.1 Sensor uncertainties

An ideal LIDAR would always measure the correct range to the nearest object within the measurement field. However, even if the LIDAR would measure the range to the closest object correct it would still be subject of errors, due to limited resolution, atmospheric effects on the measurements etc [19]. Therefore, measurement noise and uncertainties always have to be taken into account when working with LIDARs. There are also other sources of noise connected to LIDARs, including missed measurements, unexpected objects and sensor failures [19]. These also have to be taken into account when developing algorithms using measurements from a LIDAR. In order to provide information about the uncertainties of the measurements, the model in (2.5) was first expanded into three dimensions by the use of spherical coordinates as

$$\begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} = \begin{bmatrix} r_i \cos(\phi_i) \sin(\theta) \\ r_i \sin(\phi_i) \sin(\theta) \\ r_i \cos(\theta) \end{bmatrix} \quad i = 1, \dots, N, \quad (2.6)$$

where r_i is the measured range, ϕ the angle against the x axis and θ the angle against the z axis in the moving coordinate system \mathbf{x}_{sp} . The angle that describes the pitch, θ , is always set to $\pi/2$ since the LIDAR is acquiring all measurements in the same plane. The uncertainties in range and angular measurements can be transformed into uncertainties in spherical coordinates using the Jacobians,

$$\begin{aligned} J_i &= \frac{\partial(x_i, y_i, z_i)}{\partial(r_i, \phi_i, \theta_i)} = \\ &= \begin{pmatrix} \cos(\phi_i) \sin(\theta) & -r_i \sin(\phi_i) \sin(\theta) & r_i \cos(\phi_i) \cos(\theta) \\ \sin(\phi_i) \sin(\theta) & r_i \cos(\phi_i) \sin(\theta) & r_i \sin(\phi_i) \cos(\theta) \\ \cos(\theta) & 0 & -r_i \sin(\theta) \end{pmatrix}. \end{aligned} \quad (2.7)$$

Setting $\theta = \frac{\pi}{2}$ and calculating the Jacobian of (2.6) results in

$$J_i = \begin{pmatrix} \cos(\phi) & -r \sin(\phi) & 0 \\ \sin(\phi) & r \cos(\phi) & 0 \\ 0 & 0 & -r \end{pmatrix} \quad (2.8)$$

The uncertainty of a range scan increases with the detected range, and the uncertainty in range is dominating. There are also uncertainties in the angles ϕ_i and θ_i . The standard deviation in angular accuracy describes the spreading of the LIDAR. Setting the standard deviation to zero would imply a beam with no area, which is clearly not the truth. The covariance matrix for the LIDAR measurements is modelled as

$$\Sigma_i = \begin{pmatrix} \left(\sigma_r \left(1 + \frac{r_i}{r_{\max}} \right) \right)^2 & 0 & 0 \\ 0 & \sigma_\phi^2 & 0 \\ 0 & 0 & \sigma_\theta^2 \end{pmatrix}, \quad i = 1, \dots, N, \quad (2.9)$$

where $\sigma_r, \sigma_\phi, \sigma_\theta$ denote the standard deviation of r, ϕ and θ respectively. The standard deviations in the angles are the same since the LIDAR beam is circular. The standard deviations for the angles are set to

$$\sigma_\phi = \sigma_\theta = \sqrt{\frac{1}{2} \frac{2\pi}{1024}}. \quad (2.10)$$

Given the above relationships the covariance of each measurement point can be expressed in the world coordinate system \mathbf{x}_{wc} by first changing base from the spherical coordinates and then rotating by a rotational matrix, $R(\mathbf{q})$, that describes how the local coordinate system \mathbf{x}_{sp} is rotated in the world (see Section 2.2). The translation does not affect the covariance matrix as the translations are linear without any uncertainties being added [15]. This results in a covariance matrix in the fixed world coordinate frame given by

$$\Sigma_{sp} = J \Sigma J^T, \quad (2.11a)$$

$$\Sigma_{wc} = R(\mathbf{q}) \Sigma_{sp} R(\mathbf{q})^T = R(\mathbf{q}) J \Sigma J^T R(\mathbf{q})^T. \quad (2.11b)$$

Chapter 3

Laser mapping

This section covers the different mapping methods that have been implemented. The approach has been to try the simplest things first, hence starting off with averaging and median filtering for each cell. Focus was then shifted towards models that handle uncertainties, including a Kalman like filter that weights the measurements against their respective covariances, and then on to Gaussian processes (GPs). All methods have been implemented in MatlabTM. The initial methods are covered in Section 3.2. An introduction to the stationary GP can be found in Section 3.3. The stationary GP is generalized to non-stationary GP in Section 3.4.

3.1 2.5D and grid mapping

An important assumption during the course of this work has been the 2.5D assumption. The difference between 2.5D mapping and full 3D mapping problem is that gaps in the terrain as seen in Figure 3.1a are not modelled in the 2.5D case. This makes the mapping problem easier in many ways, as a 2D grid map can be used for storing information instead of a full scale 3D grid map. This reduces the amount of grids needed to represent the map by one dimension, and thus fewer calculations are needed than in the full 3D case. In a 2D elevation map the value of each grid cell corresponds to the height of the corresponding grid. A grid $h_{k,l}$ with a resolution of r cells in the first dimension (k) and s cells in the second dimension (l), makes it a total of rs cells in the grid. Such grid is illustrated in Figure 3.1b.

3.2 Initial mapping methods

Converting LIDAR data into world coordinates gives a data-set of n independent measurement points, $\mathcal{D} = (\mathbf{x}_i, y_i)_{i=1}^n$, where $\mathbf{x} = [x_1, x_2]$ corresponds to the coor-

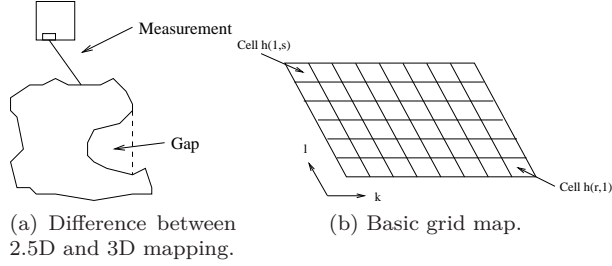


Figure 3.1: Illustration of 2.5D LIDAR mapping and the corresponding grid map

dinates in the plane and y is the terrain elevation at the corresponding location¹. The goal of the following mapping procedures is to filter and structure the data in this data set into a valid map. Initial methods are grid based approaches that focuses on merging the neighbouring raw data points into a elevation grid map. Such methods all have the advantage of being easy to understand and use, but suffer from the drawback of poor handling of incomplete data. Incomplete data occurs when the LIDAR have measured erroneous data, or simply have not collected any data in a region. Therefore merging methods needs to be used together with a interpolation technique in order to provide full maps. The three different methods for merging the data in \mathcal{D} into a map is explained below.

A first approach to determining the map is to use averaging. The area that is to be mapped is divided into a grid of wanted resolution, $h_{k,l}$. The average of the LIDAR measurements within each cell is calculated and set as the output value of the cell as

$$h_{k,l} = \frac{\sum_{j=1}^p y_{k,l,j}}{p}, \quad k \in [1, r], l \in [1, s], \quad (3.1)$$

where $h_{k,l}$ is the corresponding elevation estimate of cell k, l , $y_{k,l,j}$ is the height value of a LIDAR measurement within cell k, l , and p is the number of measurements within cell k, l . The number of hits within a cell, p , will naturally vary for all cells. The grid map h equals to the output elevation map.

The second approach is almost the same as the first one, but instead of calculating the average for each cell the median of all LIDAR hits within each cell is calculated and set to the corresponding grid elevation estimate

$$h_{k,l} = \text{med}(y_{k,l,1}, \dots, y_{k,l,p}), \quad k \in [1, r], l \in [1, s], \quad (3.2)$$

where $h_{k,l}$ is the corresponding elevation estimate of cell k, l , $y_{k,l,j}$ is the height value of a LIDAR measurement within cell k, l , and p is the number of measurements within cell k, l in the same way as for the averaging filter.

¹To clarify, $\mathbf{x} = [x_1, x_2]$ here corresponds to the x and y coordinates and y to the z coordinate in a traditional Cartesian coordinate system.

The target value of each grid cell in the elevation map can also be set by a filter that weights the measurements against their corresponding accuracy, where the the elevation of each grid cell is updated given all the observations within the grid cell in the past and the uncertainty of the measurement [4]. Observations are modelled as a normal distribution $N(y_j, \sigma_{y_j}^2)$ with σ_{y_j} being the standard deviation of the LIDAR measurement as explained in Section 2.3.1. The elevation of cell k, l given measurement j is modelled as a normal distribution with $N(h_{k,l,j}, \sigma_{h_{k,l,j}}^2)$, see Figure 3.2. The elevation of a grid cell then updated against the measurement uncertainty for each measurement in the cell and the cells uncertainty given all observations within the cell in the past. This results in a Kalman-like weighting filter where each measurement is weighted against the accuracy of the LIDAR as

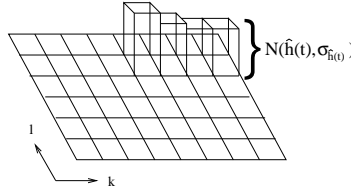


Figure 3.2: When weighting against uncertainty, each cell in the elevation map is assumed to be normally distributed.

$$h_{k,l,j} = \frac{1}{\sigma_{y_j}^2 + \sigma_{h_{k,l,j-1}}^2} (\sigma_{y_j}^2 h_{k,l,j-1} + \sigma_{h_{k,l,j-1}}^2 y_j), \quad (3.3a)$$

$$\sigma_{h_{k,l,j}}^2 = \frac{1}{\frac{1}{\sigma_{h_{k,l,j-1}}^2} + \frac{1}{\sigma_{y_j}^2}}, \quad (3.3b)$$

where in the same way as before $j \in [1, p]$ indicates the number of measurements within each grid cell. The initial uncertainty of each grid cell, $\sigma_{h_{k,l,0}}$, was set high as no information is known about the cell beforehand.

Since the three methods mentioned above all suffer from not handling incompleteness, the methods need to be augmented with interpolation methods. The method for interpolation used in this work determines the height of an incomplete cell by linear interpolation with height values from neighbouring cells. This was done by utilizing a MatlabTM command that uses Delaunay triangulation to find the height of the missing grid cells.

The LIDAR may also suffer from artefacts occurring when the LIDAR beam hits edges of objects, which may results in the returned range being a mixture of ranges to different objects. This may lead to phantom peaks in the map [24]. To reduce such effects a convolution kernel with a smoothing effect can be applied to the elevation map [4]. The convolution kernel is a simplified version of the Certainty

Assisted Spatial filter (CAS) implemented by Yee and Borenstein [24]. In the convolution kernel the height of a grid cell is filtered against the cells uncertainty $\sigma_{h_{k,l}}$ and the distance to the center of the kernel. A three by three cell convolution kernel is defined as follows. Let $h_{k+i_1,l+i_2}$ denote a height value related to the kernel center at map location k,l , with $i_1, i_2 \in \{-1, 0, 1\}$. Then weights, w , are calculated according to (3.4a) and the height of the given cell $h_{k,l}$ is updated according to (3.4b).

$$w_{i_1, i_2} = \begin{cases} \frac{1}{\sigma_{h_{k+i_1, l+i_2}}^2} & \text{if } |i_1| + |i_2| = 0 \\ \frac{1}{2\sigma_{h_{k+i_1, l+i_2}}^2} & \text{if } |i_1| + |i_2| = 1 \\ \frac{1}{4\sigma_{h_{k+i_1, l+i_2}}^2} & \text{if } |i_1| + |i_2| = 2 \end{cases} \quad i_1, i_2 \in \{-1, 0, 1\}. \quad (3.4a)$$

$$h_{k,l} = \frac{1}{\sum_{i_1, i_2} w_{i_1, i_2}} \sum_{i_1, i_2} h_{k+i_1, l+i_2} w_{i_1, i_2} \quad i_1, i_2 \in \{-1, 0, 1\}. \quad (3.4b)$$

3.3 Gaussian Processes

As previously mentioned autonomous mapping includes difficulties in several forms. Sensors are affected by measurement noise (Section 2.3.1), and the acquired data may be incomplete. Gaussian Processes (GPs) provide a way to overcome these obstacles by replacing missing information with best unbiased estimates while considering sensor uncertainty. This section covers the basics of Gaussian process regression, a method that utilizes probabilistic theory to form a map. The implementation is based on a combination of two works on GPs. Vasudevan, Ramos et al. finds a scalable solution to the mapping problem by a windowing approach [22]. Lang, Plagemann et al develops the concept of local kernels for modelling terrain and introduces the Elevation Structure Tensor (EST) for this purpose [7]. The theoretical cornerstones of GP regression theory for machine learning purposes are well summarized by Rasmussen and Williams [14]. GPs have been chosen as the method for investigation as it is a probabilistic approach that was believed to utilize the 2.5D assumption in a good way compared to other algorithms (for example Occupancy Grid Mapping (OGM)).

3.3.1 Background

The initial mapping methods discussed in the previous section all focuses on merging gathered data to gain an accurate view of the surrounding environment. Section 1.3 briefly mentioned an alternative ways to merging data strategies based on probabilistic theory. The popularity of probabilistic methods within robotic mapping stems from their ability to handle the sensor uncertainties. Probabilistic mapping methods explicitly model the sources of noise and their effect on the measurements, and does so with the whole framework of the mathematical probability

theory behind them [18]. The cornerstone of probabilistic mapping is Bayes rule. Using Bayes rule the probability of a map a given some observed data b is given by

$$p(a|b) = \frac{p(b|a)p(a)}{p(b)}, \quad (3.5)$$

where $p(a)$ is called the prior, $p(a|b)$ the posterior, and $p(b|a)$ the likelihood. The cornerstone of Bayesian filters is to maximize the probability of a map by the use of information learned about the world prior to observations and then combine the information learned beforehand with observations to make the best prediction about the observed data. Gaussian Processes (GP) is one of many methods that utilizes the Bayesian framework. GPs are a non-parametric model in the sense that they do not absorb the data used to gain the prior information, which is a benefit with GP [14].

Mapping using GPs differs against the previously mentioned algorithms in Section 3.2 in that it is based on regression. Regression based approaches does not associate measurements into a certain cell, but instead uses the measurements to create a valid model which explains the gathered data. The goal is to recover a function f

$$y_i = f(\mathbf{x}_i) + \epsilon_i, \quad \epsilon_i \sim \mathcal{N}(0, \sigma_n^2), \quad (3.6)$$

where $\mathbf{x}_i \in \mathbb{R}^d$ denotes input (location samples) with dimension d and $y_i \in \mathbb{R}$ targets (corresponding to terrain elevation). The variance σ_n^2 is the same for all n points. As the name implies GPs have strong connections with Gaussian distributions, and can be seen as an generalization of the Gaussian distribution. While Gaussian distributions handles random variables over vectors, GPs are over functions. The definition of a GP is [14]:

Definition 3.1 *A Gaussian process is a collection of random variables, any finite number of which have a joint Gaussian distribution.*

Using this definition the regression problem for terrain modelling purposes can be formalized as follows. Given a data-set $\mathcal{D} = (\mathbf{x}_i, y_i)_{i=1}^n$ with n observations of terrain elevations y at locations \mathbf{x} , find a model for $p(y^*|\mathbf{x}^*, \mathcal{D})$, where y^* is the elevations of the terrain at new locations in a test grid \mathbf{X}^* .

The key behind the Gaussian process framework is that the samples y_i from the finite data-set \mathcal{D} can be seen as being jointly normally distributed, where the inference is made in function space. Viewing the set of samples from \mathcal{D} as being normally distributed the predictive distribution for the targets is

$$p(y_1, \dots, y_n | \mathbf{x}_1, \dots, \mathbf{x}_n) \sim \mathcal{N}(\mu, K) \quad (3.7)$$

where $\mu \in \mathbb{R}^D$ denotes the mean, and K is a covariance matrix. The mean μ is usually assumed to be zero. K is specified by a covariance function k with an additional global noise term σ_n . Element (i, j) of K is defined as

$$K_{i,j} = \text{cov}(y_i, y_j) = \text{cov}(f(\mathbf{x}_i), f(\mathbf{x}_j)) = k(\mathbf{x}_i, \mathbf{x}_j) + \sigma_n^2 \delta_{ij}. \quad (3.8)$$

where δ_{ij} is called Kronecker's delta, and $\delta_{ij} = 1$ iff $i = j$, as the measurements are assumed to be independent. In other words a function f distributed as a GP is fully specified by a mean function μ which is assumed to be zero, and a covariance function k .

The covariance function is perhaps the most central part of the GP framework. A covariance function $k(\mathbf{x}_i, \mathbf{x}_j)$ represents the prior knowledge about the targets and models the relationship between measurements taken at \mathbf{x}_i and \mathbf{x}_j . Notice that the covariance function in (3.8) only depends on the input locations \mathbf{x} and not on targets y . In theory, any covariance function could be used as long as it is a positive definite function, which is a demand of the underlying Gaussian distribution theory. One choice of covariance function is the squared exponential covariance function, k_{SE} :

$$k_{SE}(\mathbf{x}_i, \mathbf{x}_j) = \sigma_f^2 \exp \left(-\frac{1}{2} (\mathbf{x}_i - \mathbf{x}_j)^T \Sigma (\mathbf{x}_i - \mathbf{x}_j) \right). \quad (3.9)$$

$\mathbf{x}_i, \mathbf{x}_j \in \mathbb{R}^d$, σ_f^2 denotes the amplitude of the function and the *kernel* Σ is a symmetric $d \times d$ matrix that models the dependencies among the dimensions d . Setting the number of dimensions d to two which is the natural choice for terrain modelling, and using a diagonal kernel matrix $\Sigma = \text{diag}(\ell_1 \ell_2)$ (3.9) can be rewritten as

$$k_{SE}(\mathbf{x}_i, \mathbf{x}_j) = \sigma_f^2 \exp \left(-\frac{1}{2} \sum_{k=1}^2 \frac{(\mathbf{x}_{i,k} - \mathbf{x}_{j,k})^2}{\ell_k^2} \right). \quad (3.10)$$

Here, the length scale parameters, ℓ_i , models how quickly the function changes along the two dimensions, and tells us how far we can move in either direction before measurements become uncorrelated. If the kernel matrix Σ is not diagonal it is possible to rotate the axes and get oriented kernels (See Section 3.3.4). Parameters that specify the covariance function, such as ℓ_1, ℓ_2, σ_f , are called *hyperparameters* for the covariance function and are denoted by Θ . Finding the right hyperparameters is of great importance and is further discussed in Section 3.3.2.

Notice that the k_{SE} covariance function in (3.10) only depends on relative distance between data points, $|\mathbf{x}_i - \mathbf{x}_j|$. It is invariant to translations and says that the rate at which points are correlated with each other decreases with the euclidean distance between the points. A covariance function that only depends on relative distance and thus is the same all over input space, such as the k_{SE} , is called a stationary covariance function.

The process of finding the hyperparameters Θ for a specific kernel is called *training* the Gaussian process. This procedure is described in the next section. The found hyperparameters Θ together with the training data set \mathcal{D} is then used to *apply* the GP to a set of points in a test grid \mathbf{X}^* . In practice, the amount of data provided in the data-set \mathcal{D} is often fairly large. Therefore, a sampling step is often included when working with GPs. The sampling of data points could in theory be random,

uniform or taken from high gradient areas in the terrain. Random sampling have been used in this work. The overall process of using the Gaussian processes can be seen in Figure 3.3.

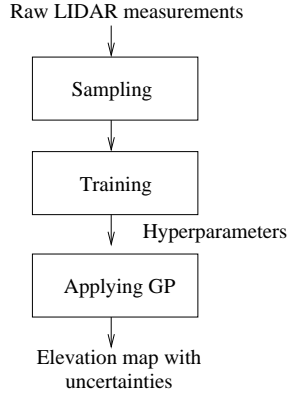


Figure 3.3: The steps of mapping using GPs can be divided into sampling, training and applying.

3.3.2 Training a Gaussian Process

The covariance function of a GP is not fully specified without values for the set of hyperparameters Θ . For the squared exponential covariance function introduced in the previous section, the set of hyperparameters includes ℓ_1, ℓ_2, σ_f as well as the global noise parameter σ_n . These hyperparameters depend both on the choice of covariance function and the data-set, \mathcal{D} . Therefore, training the GP is the same thing as optimizing the hyperparameters in the light of the data. The importance of finding the right length scales is illustrated in Figure 3.4 for a one dimensional problem. Having too short length scales will result in uncertainty growing significantly away from the measurements points, Figure 3.4b, while having too long length scales yields a smoother function but at the cost of higher uncertainty overall, Figure 3.4c. The optimal choice is a trade-off, Figure 3.4d.

Training the GP equals to finding the optimal solution to the log marginal likelihood problem

$$\log p(\mathbf{y}|\mathbf{X}, \Theta) = -\frac{1}{2}\mathbf{y}^T \mathbf{K}^{-1} \mathbf{y} - \frac{1}{2} \log |\mathbf{K}| - \frac{n}{2} \log(2\pi), \quad (3.11)$$

where \mathbf{y} denotes a vector of y measurements, \mathbf{X} holds the corresponding \mathbf{x} measurements and \mathbf{K} is the covariance function for the corresponding noisy targets, $K_{i,j} = k(\mathbf{x}_i, \mathbf{x}_j) + \sigma_n^2 I$ from (3.8) and (3.10). This expression is the result of marginalization over the function values, integrating the prior times the likelihood in (3.5). The log marginal likelihood problem in (3.11) is non-linear, so the optimization of the hyperparameters is not a convex problem. From (3.11) itself it is

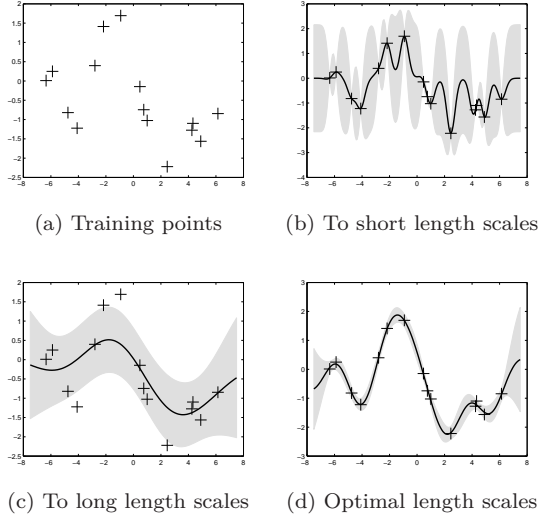


Figure 3.4: Illustration of the importance of finding the correct length scale, ℓ . Grey areas indicate confidence regions.

possible to derive the partial derivatives of the hyperparameters analytically, but in practice gradient based approaches are the choice. In this thesis, an implementation that uses Polack-Ribiere flavour of conjugate gradients to find the optimal values have been used².

Note that (3.11) contains three different terms. The first term is describing the data fit, the second term is penalizing complexity and the third term is a normalizing factor. Having the penalizing term in the expression makes sure the parameters does not suffer from overfit. Thus, Occam's razor which says that "entities must not be multiplied beyond necessity", is built into the optimization procedure.

3.3.3 Applying a Gaussian Process

After the optimal hyperparameters have been found, the GP can be applied to a set of query points, the test grid \mathbf{X}^* that consists of m regression points \mathbf{x}^* . The result of the process of applying the GP is an elevation grid map with given uncertainties, see Figure 3.3. As the joint distribution³ of any finite number of random variables of a GP is Gaussian, the measurements from the data-set \mathcal{D} and the query points \mathbf{X}^* can be seen as samples taken from the same Gaussian distribution [22]. Therefore it is possible to specify a one dimensional normal

²Implementation by Carl Rasmussen available at <http://www.gaussianprocess.org/gpml/>

³For a joint Gaussian $\begin{bmatrix} x \\ y \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} a \\ b \end{bmatrix}, \begin{bmatrix} A & C \\ C^\top & B \end{bmatrix}\right)$ the conditional probability is given as $x|y \sim \mathcal{N}\left(a + CB^{-1}(y - b), A - CB^{-1}C^\top\right)$

distribution that expresses the relationship between the measurements and the test grid. By denoting all \mathbf{x} measurements in \mathcal{D} by \mathbf{X} the relationship between the heights of the corresponding measurements \mathbf{y} and the height of the regression points \mathbf{y}^* is given as

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{y}^* \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} k(\mathbf{X}, \mathbf{X}) + \sigma_n^2 I & k(\mathbf{X}, \mathbf{X}^*) \\ k(\mathbf{X}^*, \mathbf{X}) & k(\mathbf{X}^*, \mathbf{X}^*) \end{bmatrix}\right), \quad (3.12)$$

where for n training points in the data-set \mathcal{D} and n^* points in the test grid \mathbf{X}^* , $k(\mathbf{X}^*, \mathbf{X})$ denotes a $n \times n^*$ matrix evaluated between all pairs of training and test points. By denoting $K = k(\mathbf{X}, \mathbf{X})$, $\mathbf{k} = k(\mathbf{X}, \mathbf{X}^*)$ and $k_* = k(\mathbf{X}^*, \mathbf{X}^*)$ the following important equations calculate a one dimensional normal distribution for the targets of the test grid

$$f^* = \mathcal{N}(\mu^*, v^*) \quad (3.13a)$$

$$\mu^* = E\{f^*\} = \mathbf{k}^T (K + \sigma_n^2 I)^{-1} \mathbf{y}, \quad (3.13b)$$

$$v^* = V\{f^*\} = k_* + \sigma_n^2 - \mathbf{k}^T (K + \sigma_n^2 I)^{-1} \mathbf{k}, \quad (3.13c)$$

where dimensions of these properties given n points of training data is: $K \in \mathbb{R}^{n \times n}$, $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$, $\mathbf{k} \in \mathbb{R}^{n \times n^*}$, $k_{i,j} = k(\mathbf{x}_i^*, \mathbf{x}_j)$, $k^* = k(\mathbf{X}^*, \mathbf{X}^*) \in \mathbb{R}^{n^* \times n^*}$, and $\mathbf{y} \in \mathbb{R}^n$. The uncertainty for a point in the test grid, \mathbf{x}^* , is given by the covariance $k^* + \sigma_n^2$ minus a positive term that depends on the data from the training inputs, $\mathbf{k}^T (K + \sigma_n^2 I)^{-1} \mathbf{k}$. The posterior covariance will therefore be smaller than the prior covariance all the time. Or in other words, the information from the training data-set is used to lower the uncertainties of the estimates.

3.3.4 Visualisation

The perhaps most important part of the k_{SE} covariance function is the *kernel* Σ . As mentioned before, if Σ is a diagonal matrix it holds the length scales for each dimension. The length scale parameters, ℓ_i , models how quickly the function changes along the two dimensions, and tells us how far we can move in either direction before measurements become uncorrelated. They can be seen as a covariance matrix for the Gaussian kernel. If Σ is a non-diagonal matrix the axes are rotated and describes how the covariance structure of the input space is oriented. Luckily this has a good visualization possibility, as kernel matrices can be seen as ellipses in the case of two dimensional input space.

Any positive semi-definite matrix⁴ is a valid kernel matrix, as it meets the criterions for the underlying multivariate normal distribution. Looking into the matrix decomposition rules in linear algebra, the spectral theorem says that any positive definite matrix Σ can be divided into two matrices $\Sigma = RAR^T$, where A is a

⁴A symmetric $n \times n$ matrix M is positive definite if $z^T M z > 0$ holds for all non-zero vectors z , $z \in \mathbb{R}^n$.

diagonal matrix containing the eigenvalues of Σ and R contains the corresponding eigenvectors for these eigenvalues. The matrix Σ can thus be divided into a rotation matrix R and a diagonal matrix A as:

$$\Sigma = RAR^T = R \begin{pmatrix} \ell_1^2 & 0 \\ 0 & \ell_2^2 \end{pmatrix} R^T = \quad (3.14)$$

Σ can be visualized as an ellipse, where the eigenvalues determines the length along the axes of the ellipse and the angle of the orientation matrix R determines the orientation of the ellipse. Let α be the orientation angle of the rotation matrix $R = \begin{pmatrix} \cos \alpha & -\sin \alpha \\ \sin \alpha & \cos \alpha \end{pmatrix}$ in (3.14), then the kernel matrix Σ can be visualized according to Figure 3.5. The ellipse can also be seen as statistical limits of a 2D- Gaussian distribution, where the length scale corresponds to one standard deviation of the Gaussian.

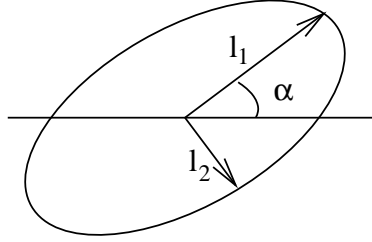


Figure 3.5: Visualization of kernels, Σ , is possible by drawing ellipses with length-scales ℓ_i defining the length of the axes and orientation specified by an angle α .

3.3.5 Local windowing

The equations for estimating the terrain elevation and the corresponding uncertainty given in (3.13) includes an inversion of the covariance matrix of the training data $(K + \sigma_n^2 I)$. This matrix is of the dimension $n \times n$, making the matrix inversion very costly for a large scale terrain modelling problem as the inversion operation is of cubic complexity $O(n^3)$. There are several more or less complicated approximation methods reducing the complexity of this inversion, several of them are listed by Rasmussen and Williams [14]. An initial method is to divide the area into several subregions in order to decrease the size of matrix that is to be inverted. This has been performed by Lang in order to speed up computations of a large scale mapping scenario [7]. A natural extension to this method is to apply a local moving window technique. A local approximation method like the moving window technique is based on the idea that a point is correlated the most with its immediate neighbourhood. Hence the m spatially closest points is chosen when applying the GP model to the terrain, which speeds-up the matrix inversion of $(K + \sigma_n^2 I)$. In this work $m = 50$ have been used. A drawback of using the moving window approximation is that it introduces the need for good sorting methods in order to find the nearest neighbours of a point. Finding closest neighbours is a

subject that has been well analysed throughout the years. Vasudevan, Ramos et al proposed the use of a k-dimensional tree (KD-tree) for sorting the measurement points and finding the nearest neighbours [22]. A MatlabTMKD-tree implementation was briefly tested during this work, but was not the optimized and was outperformed by sorting the matrices after euclidean distance.

3.4 Non-stationary Gaussian Processes

In Section 3.3.1 the squared exponential covariance function k_{SE} was introduced. The k_{SE} is a frequently used covariance function in GPs as it is fairly easy to understand since the correlation between the points in input space decreases with euclidian distance in the same way across the whole input space. However, this property of the k_{SE} does not fit very well with our goals of modeling terrain using GPs. It is not realistic to assume that the terrain varies as much in all areas, some areas may be flat while others may be very rough. The k_{SE} also gives a smoothing effect on map which is good in flat areas but not wanted in regions with rough obstacles. One of the difficulties in mapping is to find a good trade-off between smoothness and preserving discontinuities. A natural thought would be to vary the length scales, and thus the covariance function, depending on the local terrain. This introduces the principle of non-stationary covariance functions and Non-stationary Gaussian Processes (NGP). Whereas a stationary covariance function is static all over the input space, a non-stationary covariance function varies over the input space and enables the capturing of local terrain properties. There are several possible non-stationary covariance functions. For example Vasudevan, Ramos et al. successfully used a dot product neural covariance function kernel to model large scale terrain [22]. In this work a slightly different approach, first introduced by Paciorek and Schervish [12] is attempted.

Paciorek and Schervish introduced a generalized version of the k_{SE} covariance function by assigning an individual kernel matrix Σ_i to each measurement point \mathbf{x}_i in the data-set \mathcal{D} . This kernel matrix holds information about the local environment of \mathbf{x}_i , and provides a way to adapt the length-scales depending on the input location. The principle is illustrated in Figure 3.6. The idea is that in areas with flat terrain the kernel matrix is circular and holds large length scales in both dimensions, while areas of rough terrain yields thinner kernels with length scales adapted to the terrain. Another way of putting this would be that in a point in rough terrain, one does not have to go far away from the measurement point to become uncorrelated, while flat areas have higher correlation with measurements far away. The stationary squared exponential covariance function k_{SE} is generalized into the non-stationary squared exponential, k_{NSE} as

$$k_{NSE}(\mathbf{x}_i, \mathbf{x}_j) = \sigma_f^2 |\Sigma_i|^{\frac{1}{4}} |\Sigma_j|^{\frac{1}{4}} \left| \frac{\Sigma_i + \Sigma_j}{2} \right|^{-\frac{1}{2}} \cdot \exp \left[-(\mathbf{x}_i - \mathbf{x}_j)^T \left(\frac{\Sigma_i + \Sigma_j}{2} \right)^{-1} (\mathbf{x}_i - \mathbf{x}_j) \right]. \quad (3.15)$$

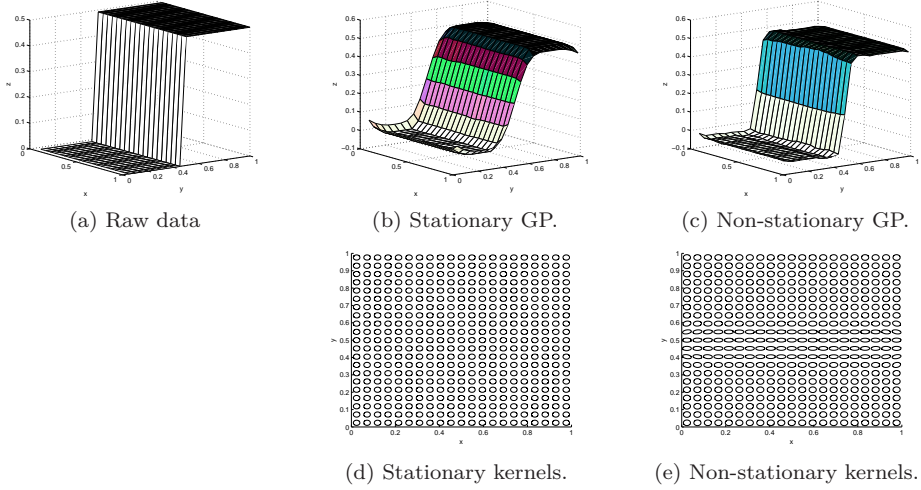


Figure 3.6: For the k_{SE} covariance function the same kernels is used for all input locations which leads to oversmoothing of discontinuities. The non-stationary covariance function, k_{NSE} adapts individual kernels that makes it possible to handle discontinuities better, and at the same time gain smoothness in flat areas.

where Σ_i and Σ_j correspond to the kernels associated with \mathbf{x}_i and \mathbf{x}_j respectively. Breaking down k_{NSE} and comparing it with k_{SE} in (3.10) first notice that setting $\Sigma_i = \Sigma_j$ in (3.15) gives the stationary k_{SE} covariance function. This indicates that the k_{NSE} is indeed a generalization of the k_{SE} . k_{NSE} can be divided into three parts;

$$k_{NSE} = \sigma_f^2 \cdot p \cdot e \quad (3.16)$$

The properties of these parts is fully explained in [6], but is briefly summarized here as well. The first part is recognized from k_{SE} as the amplitude scale factor. The second and third term differs from k_{SE} . The exponential part e measures the Mahalanobis distance between input locations, a weighted average of the individual kernels at input locations i and j . As is the case with k_{SE} , the correlation between data points decreases with distance, but in k_{NSE} even two distant input locations may be correlated if the averaged kernel is large.

The addition of the last term, the prefactor p , might look confusing. It is added to make k_{NSE} positive definite, which is a requirement on a covariance function in the Gaussian framework [14]. However, the prefactor also gives rise to some unexpected behaviours noted in [13] and analysed in [6]. Summarized, the prefactor penalizes kernels that differ in shape. That is, given two equally shaped kernels the prefactor will get its largest value at $p = 1$, independent of the kernel size. The prefactor p then starts to decrease towards zero when the kernels start to differ in shape. This leads to larger kernels having smaller covariances if they are different in shape in comparison to two small kernels that are equally shaped.

Looking closer at (3.16) there are risks of data overfitting that can be derived from both the exponential part and prefactor. Overfitting occurs if one relies too much on the close local surroundings, which in the end may result in correlation with a single measurement point alone. If the kernels become very small the exponential part will add to the risk of overfitting. On the other hand, the prefactor will lead to overfitting if the shapes of the neighbouring kernels are too different. Due to these behaviours, there are in practice two constraints on the kernels that need to be met in order to avoid overfitting:

- Kernels shall *not be too small*, due to the behaviour of the exponential part.
- Kernels shall *vary smoothly*, due to the behaviour of the prefactor.

Note that there is a difference between saying that the shape of the kernel shall vary smoothly, and that the terrain itself shall vary smoothly. For example, the kernel could look the same at both sides of a rough step in the terrain, with the kernels being shaped along the edges.

3.4.1 The elevation structure tensor

The previous section introduced the principle of covariance functions with local kernels, Σ_i . Finding the local kernels can be done in multiple ways, but is not a trivial task. In addition to the hyperparameters σ_f and σ_n three additional parameters, ℓ_1, ℓ_2 and α , have to be set for all n kernels. This results in a total number of $3n + 2$ parameters to set. When Paciorek and Schervish introduced the principle of the non-stationary covariance function they proposed a hierarchical method for finding the local kernels by using a second level GP and Markov-Chain Monte Carlo sampling [12]. As stated by the authors this provides a good and elegant solution by using the same framework to find Σ_i as to make the actual elevation predictions. However, the method introduces several additional parameters per local kernel, and hence loses the simplicity of the stationary GP model. This also leads to very slow computations as the number of parameters grows, computationally limiting the amounts of kernels to around one thousand measurement points. Lang introduces two methods of finding the kernels in [6]. The first method uses a gradient search method for finding the optimal values for each local kernel. In order to avoid local minima a neural network adaptation procedure called RProp is used, and an extra regulation procedure is applied to ensure that the local kernels vary smoothly. Even if special precautions were taken to avoid local minima, the method is still sensitive to local minima as the amount of parameters ($3n + 2$) quickly gets very large.

The second method introduced by Lang in [6] is the method of choice in this work. The values for Σ_i are found by adapting the kernels against the local gradients of the terrain. The hyperparameters found by the optimization procedure in Section 3.3.2 are influenced by the properties of the local terrain structure in an iterative manner. The method of using local gradient information is inspired

by work done in the computer vision community by Middendorf and Nagel [10]. They used an adaptive approach to find the grey-value structural tensor to capture the local image structure. The method also has similarities with the principles of the Harris detector [3], used in the computer vision community to detect edges. The difference between detecting edges in an image and terrain modelling using LIDAR is that the pixels in images consists of equidistant information, whereas data gathered by a LIDAR does not. The principle of adapting kernels against local terrain is built upon the definition of a Elevation Structure Tensor (EST). The EST provides a locally weighted average of the gradients in an area and are estimated directly from the elevation samples in the neighbourhood of the point. The EST for a point $\mathbf{x}_i = [x_{i,1}, x_{i,2}]$ is defined as

$$EST(\mathbf{x}_i) = \sum_{k=1}^m w(\mathbf{x}_k, \mathbf{x}_i) \begin{bmatrix} I_{x_1}^2 & I_{x_1}I_{x_2} \\ I_{x_2}I_{x_1} & I_{x_2}^2 \end{bmatrix} = \begin{bmatrix} \langle I_{x_1}^2 \rangle & \langle I_{x_1}I_{x_2} \rangle \\ \langle I_{x_2}I_{x_1} \rangle & \langle I_{x_2}^2 \rangle \end{bmatrix}, \quad (3.17)$$

where $I_{x_1} = \left(\frac{\partial y}{\partial x_1} \right)$ and $I_{x_2} = \left(\frac{\partial y}{\partial x_2} \right)$ is the first order derivatives with respect to height and $I_{x_1}^2$ and $I_{x_2}^2$ denotes second order derivatives in the same manner. The window w consists of normalized Gaussian weights with standard deviation σ_E

$$w(\mathbf{x}_k, \mathbf{x}_i) = \frac{1}{2\pi\sigma_E^2} e^{-|\mathbf{x}_k - \mathbf{x}_i|/2\sigma_E^2}. \quad (3.18)$$

where k indexes the m points closest to \mathbf{x}_i and therefore determines the size of the window.

The EST is a 2 by 2 matrix which can be represented by two eigenvalues λ_1 and λ_2 as well as an orientation parameter β (See Section 3.3.4). The first eigenvalue of the EST is pointing in the direction where the terrain gradient is strongest, while the second eigenvalue will be perpendicular to this direction. These properties are consistent with those of the Harris detector [3] and the following inference can be made by considering the magnitude of λ_1 and λ_2

- If both $\lambda_1 \approx 0$ and $\lambda_2 \approx 0$ the measurement point is located in a flat area.
- If $\lambda_1 \gg 0$ and $\lambda_2 \approx 0$ an edge in the terrain is found.
- If both $\lambda_1 \gg 0$ and $\lambda_2 \gg 0$ a corner, terrain top or a bottom of a hole is found.

The basic relationship between the EST tensor and the local kernels Σ_i is that Σ_i should be adapted against the inverse of the corresponding EST tensor. In flat areas where the EST is small there is correlation between measurements over large areas. In the proximity of edges or in steep terrain the EST is oriented towards the strongest ascent, and correlation between measurements can be found at each side of the edge. This means that the kernels Σ_i should be shifted 90° against the largest eigenvalue of the EST. For an edge, the kernels will then be oriented along the edge. For corner structures two large eigenvalues will result in smaller circular Σ_i . The principles of the EST adaptation can be seen in Figure 3.7.

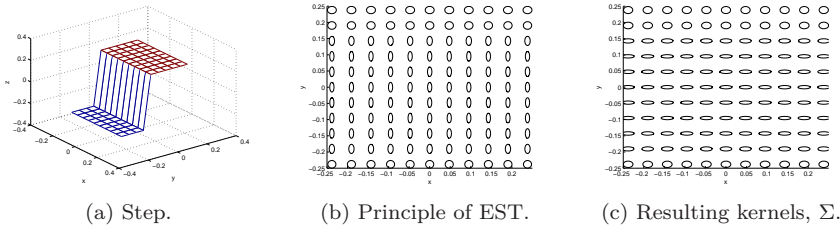


Figure 3.7: The underlying idea behind the Elevation Structure Tensor (EST) is that the kernels should be rotated 90° against the strongest elevation.

3.4.2 Finding the local kernels

Finding the local kernels with the properties presented in Section 3.4.1 is done iteratively. The reason for utilizing an iterative process is to ensure that overfitting is avoided. Kernels shall not be too small to avoid extreme correlation with the closest neighbourhood, and kernels should vary smoothly across the input space. The second requirement is hard to verify in a strict way, and also the hardest requirement to meet implementation wise. In order to prevent the kernels from being too large, limitations on the kernel eigenvalues are set to ensure that kernels lie between a smallest and largest value σ_{min} and σ_{max} [10]. Different methods for limiting the kernels have been proposed. In this work choice has been Bounded Linear adaptation as it has proven to produce well balanced kernels [6]. Bounded Linear adaptation first introduces $\bar{\lambda}_k = \lambda_k / (\lambda_1 + \lambda_2)$, $k = 1, 2$ and uses spectral theorem to separate Σ into an orientation matrix and a diagonal matrix containing the eigenvalues of Σ as described in Section 3.3.4. Limitations on the length-scales of ℓ_k can then be set as

$$\ell_k^2 = \bar{\lambda}_k \sigma_{min}^2 + (1 - \bar{\lambda}_k) \sigma_{max}^2, \quad k = 1, 2. \quad (3.19)$$

The process of finding the local kernels Σ_i can thus be made in an iterative manner using the dataset \mathcal{D} , σ_{min} and σ_{max} as inputs according to Algorithm 1. The local kernels are initialized with the lengthscales of the stationary GP, and then adapted against the inverse of the local values of the EST tensor, Σ'_i . The parameters σ_{min} and σ_{max} are tuned depending on the size of the optimal length scale of the stationary GP.

Algorithm 1 also introduces the local learning rate μ_i which is used to speed-up the iterations. The learning rate is set in relation to the local data-fit $\text{df}(\mathbf{x}_i)$ and the kernel complexity c_i . The data-fit increases the learning rate if the model is poor within an area, while the complexity term penalizes learning if the kernel is starting to become too small. The data-fit is given by

$$\text{df}(\mathbf{x}_i) = \frac{p(y_i|x_i)}{\max_y p(z|x_i)}. \quad (3.20)$$

This equation can be evaluated as (3.6) states that $y_i = f(\mathbf{x}_i) + \epsilon_i$ where the

noise is given as $\epsilon_i \in \mathcal{N}(0, \sigma_n^2)$. Hence Equation 3.20 yields a number between zero and one, and can be calculated as

$$\text{df}(\mathbf{x}_i) = \frac{\mathcal{N}(y_i - f(\mathbf{x}_i), \sigma_n^2)}{\mathcal{N}(0, \sigma_n^2)}. \quad (3.21)$$

The kernel complexity c_i is calculated from the length scales as

$$c_i = \frac{1}{|\Sigma_i|} = \frac{1}{l_{1,i}^2 l_{2,i}^2}. \quad (3.22)$$

The resulting learning rate is estimated by a sigmoid (3.23), see Figure 3.23. The sigmoid is empirically tuned and holds two parameters, a and b , used to modify the sigmoid. The parameter b is needed as the complexity term is general quite large and needs to be scaled down. Parameter a is needed as the normal sigmoid is defined between $[-\infty, \infty]$, but as the the data-fit term is defined only between $[0, 1]$ it only enables half of this input space.

$$f(\mathbf{x}_i) = \frac{1}{1 + \exp\left(\frac{\text{df}(\mathbf{x}_i) \cdot c_i - a}{b}\right)}. \quad (3.23)$$

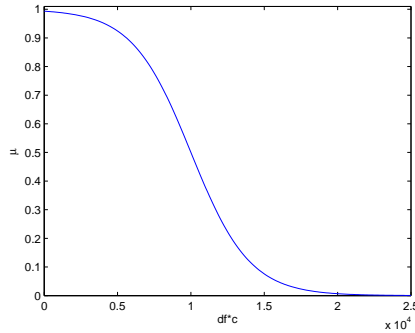


Figure 3.8: A sigmoid depending on the data fit, $\text{df}(\mathbf{x}_i)$ and kernel complexity c_i is used to increase the learning rate.

Algorithm 1 Adaptation of local kernels

Learn global parameters Θ using stationary GPs.
Initialize local kernels $\Sigma_i = \Theta$
while not converged
 for all Σ_i
 Estimate the local learning rate μ_i
 Estimate $\text{EST}(\mathbf{x}_i)$ according to Σ_i
 Adapt Σ'_i according to $\text{EST}(\mathbf{x}_i)$
 $\Sigma_i \leftarrow \mu_i \Sigma'_i + (1 - \mu_i) \Sigma_i$
 end for
end while

3.4.3 Estimating the kernel at unseen locations

The k_{NSE} covariance function introduced above requires a local kernel to be assigned to each data point in the training set \mathcal{D} . When applying the GP using (3.13b) and (3.13c), evaluation of the covariance function at the new input locations in the test grid \mathbf{x}^* , and the evaluation of $k(\mathbf{x}^*, \mathbf{x}^*)$ and $k(\mathbf{x}_i, \mathbf{x}^*)$ is required. As there is naturally no terrain measurement available at these locations, the kernel Σ^* can not be estimated by the EST tensor. First thoughts of solving this problem includes either choosing the closest kernel from the data set \mathcal{D} to represent the kernel Σ^* , or to use the stationary GP kernel for all predicted kernels Σ^* . Choosing the closest kernel will however exclude the penalty of the prefactor from the calculations, and choosing the stationary kernel for all prediction points will not guarantee the local smoothness criterion mentioned in Section 3.4. Instead a weighted average of the kernels of the closest neighbours of \mathbf{x}^* is made to estimate the corresponding kernel Σ^* . The kernels are weighted against a 2D Gaussian with standard deviation of 1, so that kernels in \mathcal{D} closer to \mathbf{x}^* is weighted higher than kernels associated with points further away. The weights are normalized so that they sums up to one. In this way the local kernels can be estimated also for points in the test grid, \mathbf{X}^* .

Chapter 4

Experimental results

There are several different aspects to keep in mind when evaluating the implementations presented in the previous chapter. The results depend on what one prioritizes in terms of accuracy, robustness, scalability and to some extent speed and complexity. While evaluating the results from the experiments it is also important to have a real flight scenario in mind. This chapter first compares the two methods based on Gaussian Processes based on an artificial data-set. Then an experimental setup is presented, and focus is placed on the LIDAR sensor. Two experiments are made using this setup. The first one with terrain changes where the true volume change is known accurately, and the second one with more natural terrain. A discussion about the results is provided at the end of this chapter.

4.1 Artificial data set

To illustrate the principles of the non-stationary kernel an artificial data set is used. An artificial data set is a good way of showing the different properties of the Gaussian Processes while eliminating several problems of working with sensors. The artificial data in Figure 4.1 contains both flat regions and several discontinuities that are hard to adapt to. It contains three plateaus with different edges. To simulate a realistic environment random white noise of magnitude $\sigma = 0.03$ was added to the raw data. The artificial data-set consists of 441 measurement points.

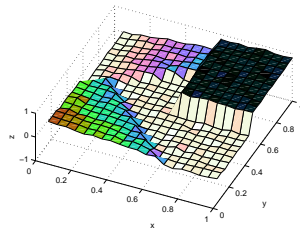


Figure 4.1: Artificial dataset with noise.

The results of applying the GPs to the data are shown in Figure 4.2. Before the adaptive iterations, the NGP is the same as a stationary GP. This can be seen in Figure 4.2b, which shows that the local kernels are the same all over the input space. After the iterations, the kernels have adapted to the local environment. The benefit of adapting local kernels is clearly seen at the discontinuous steps between the plateaus. For example the smoothing effect between the highest plateau and the ground level is not as visible as for the GP. Looking at the local errors in Figure 4.2e and 4.2h it is also clear that the steps looks hardest to adapt to and they also contains the regions with largest error.

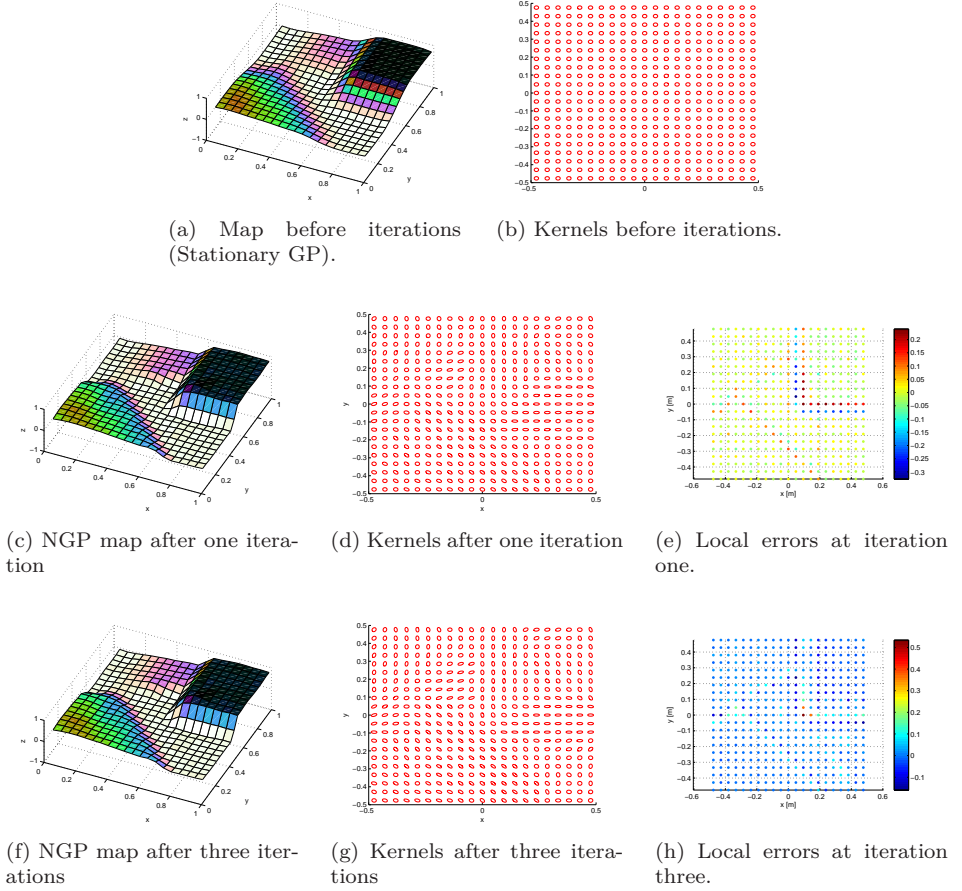


Figure 4.2: Artificial data set showing the progress of the NGP.

Working with the artificial data-set showed that it is possible to get a better trade-off between smoothing flat regions and step discontinuities. It also showed the increasing amount of complexity involved in the NGP compared to GP. The main problem found with NGP in the artificial data-set has been the tendency to overfit data. The requirement that says kernels of the k_{NSE} should not get to

small is easy to control. However, the requirement that states kernels shall vary smoothly is hard to meet. This includes tuning of the learning-rate sigmoid, which is a crucial part in meeting this requirement.

4.2 Experiments using an industrial robot

Flight time for an UAV is rather expensive and also surrounded by many security regulations. Therefore an UAV flight was simulated by the use of an industrial robot throughout the thesis. The robot provides the same degrees of freedom as a real helicopter and also provides the necessary pose information. The same combination of sensors that is to be used in real flight was attached to a sensor platform that is held by the arm of the industrial robot, as shown in Figure 4.3.

There is of course differences between the experimental environment and real flight that is important to be aware of. The environment in which the industrial experiment is carried out is a lot more controlled than a real flight. Even if the robot itself can be programmed with great precision it is very hard to fully simulate the vibrations of the UAV. Furthermore the sensor platform can not rotate more than 180 degrees in either direction, so a fully natural flight path can not be programmed. Further, the LIDAR used within the experiment is not the same as the one used during real flight scenarios. The resolution and other properties of the LIDAR data will therefore not be the same during experiment and flight scenarios. The data collected by the experiment is to be considered quite uniform and the grid cells are hit by the LIDAR beam in a more evenly distributed way in the experiment compared to real flight. The mounting of the sensors will also differ between the experiment and the real flight, as well as the speed in which the sensors is moving above the ground. A flight using a real helicopter will also yield significantly more measurement data as the intended LIDAR sensor will be operating at higher frequency and the flight scenario will be longer. This calls for a scalable mapping solution.

4.2.1 Data collection

There are three sensors attached to the sensor platform on the industrial robot in Figure 4.3. An Inertial Measurement Unit (IMU), a camera and a LIDAR. The IMU, a Crossbow IMU400CC-200, measures angular velocity and specific force using accelerometers and gyroscopes. In this thesis the IMU is mainly used for synchronization between the sensors mounted on the sensor platform and the industrial robot, as they provide measurements with different timestamps. The camera is an Allied Marlin F080B, which takes black and white images of the environment. These images are used for state estimation, for example for loop closure and to estimate the orientation of the industrial robot. The industrial robot, an ABB IRB-1400, is a frequent tool within the process industry for efficient automation of processes. In this work the industrial robot is both used as a "sensor" providing the states introduced in Section 2.1 and to simulate the UAV movement. In a real flight scenario a GPS will also be mounted on the UAV to

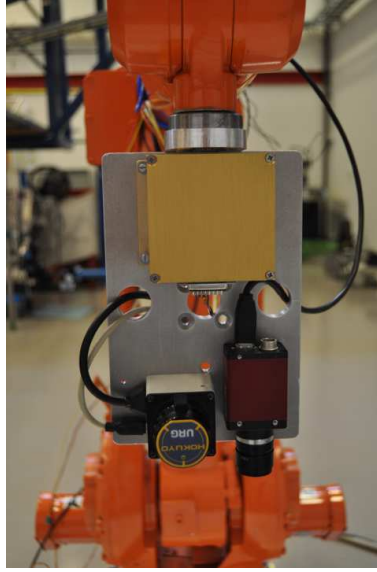


Figure 4.3: The industrial robot and the sensor platform.

replace the states provided by the robot.

An Intel P4, with 1GB RAM and Ubuntu was used in the data collection. To handle the data from the sensors a software suite called Player¹ was used. Player is a project that gathers different Linux sensor drivers to make it easier to collect and visualize data. Player was used as it provided drivers that worked well out of the box for all sensors except for the IMU.

There are two different sets of timestamps within the experiment that are in need of synchronization. On the one hand there is the industrial robot with its own timestamps on the data, produced by a background process in the industrial robot that stores the position and rotation of the sensor platform with a sample frequency of 100 Hz. On the other hand the sensors placed at the sensor platform receive their timestamps by Player. Only the mounted IMU provided an internal time stamp for the measurements. A time stamp is therefore placed on each measurement by the computer when it arrives in Player. In order to synchronize these timestamps the acceleration data stored by the IMU and the positions stored by the industrial robot is used. The robot moves in a predefined pattern, a synchronization sequence, at the start of every data collection. The position stored by the industrial robot is derived two times to get comparable units. These accelerations are then correlated to find the time delay between the data sets. A typical synchronization sequence can be seen in Figure 4.4.

¹Visit the Player Stage Project at playerstage.sourceforge.net/

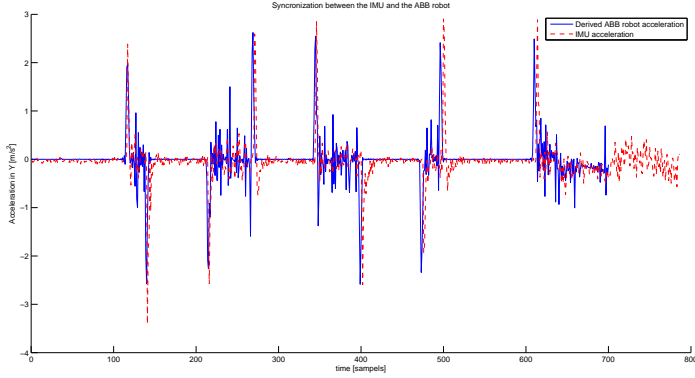
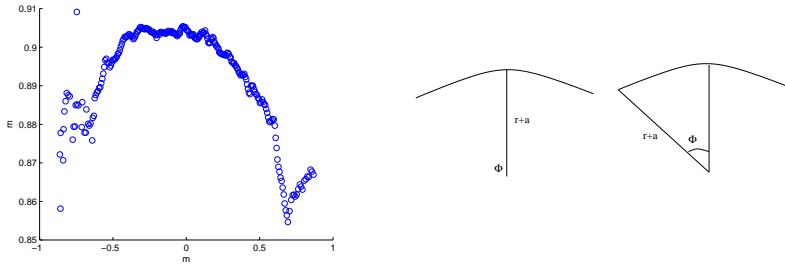


Figure 4.4: Synchronization sequence between robot and sensors, followed by a shorter circular path.

4.2.2 LIDAR calibration experiments

Simple experiments with the LIDAR showed that the LIDAR was effected by several different sources of errors and was in need of calibration. The LIDAR was slowly rotated 180° on the spot while taking measurements over a planar surface. Rotating the LIDAR was done to make sure the ground was planar, yet the LIDAR measurements looked convex as well as somewhat skew. This can be seen in Figure 4.5a. The LIDAR was therefore assumed to have both



(a) Average of sweeps made by LIDAR on planar surface while rotating the LIDAR 180° .

(b) Cancelling the convex look of the LIDAR sweep by adding a small constant to all the range measurements r .

Figure 4.5: LIDAR sweep field of view limited to $[2\phi_{\min}, 2\phi_{\max}]$.

a small bias error in range and a small angle (mounting) error. The skewness was compensated for with an empirically tuned rotation matrix, applied when projecting the LIDAR data into the world. The convex look of LIDAR sweeps was cancelled by adding a small constant to the range measurements as illustrated in Figure 4.5b. Adding a small constant to the measurements cancels out the convex effect as measurements far out in the viewing field will be affected less by the

constant than measurements at zero degrees. While the effect of the sweeps being convex was somewhat reduced as the field of view was limited to $[\phi_{\min}, \phi_{\max}]$ (and the effect is clearer for larger angles), it was still significant when doing longer experiments. The result of cancelling the convex behaviour and compensating by a rotational matrix can be seen in Figure 4.6. By taking measurements at different ranges within the area of work the conclusion that the LIDAR properties would differ greatly at different distances to the target was neglected, see Figure 4.7. Figure 4.5 and the rest of the LIDAR sweep illustrations in this section (except Figure 4.5a) was made by averaging over approximately 150 LIDAR sweeps while the sensor was not moving.

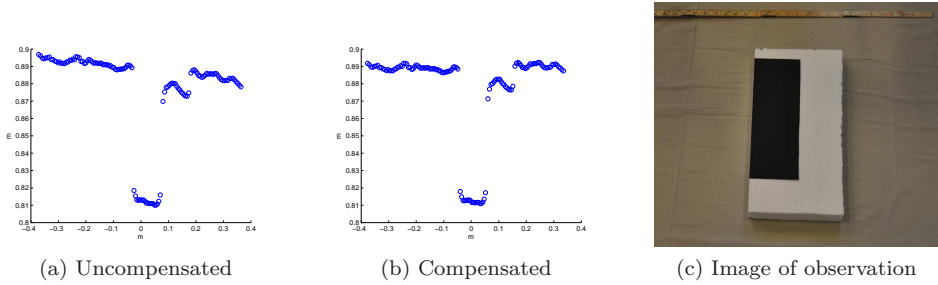


Figure 4.6: Partly covered styrofoam block (5 cm) placed on a sheet.

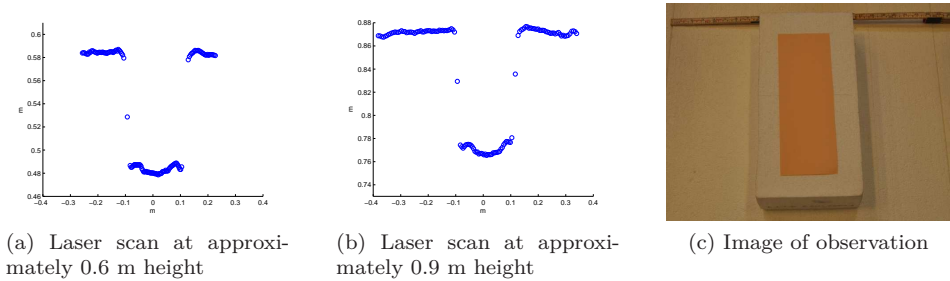
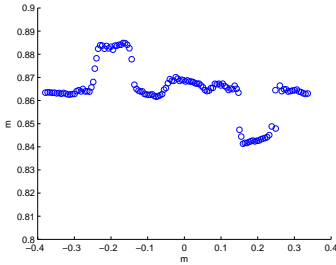


Figure 4.7: 10 cm styrofoam block with orange identifier measured at different distances from ground.

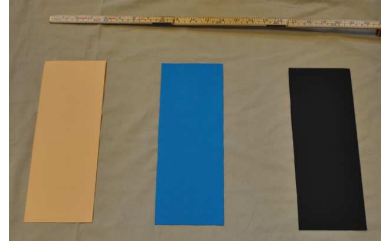
Reflectivity issues

During the data acquisition it was important to have identifiers on the ground for a camera to be able to track features. The choice of color for these features proved to be very important. The returned range from a LIDAR depends on the reflection property of the material the beam hits [4]. In Figure 4.6 a five cm height styrofoam chunk was placed on a planar light-grey sheet. A small black paper sheet was also placed on the styrofoam. In Figure 4.6 the part of the block that was not covered by the black paper was measured to about two cm, while the

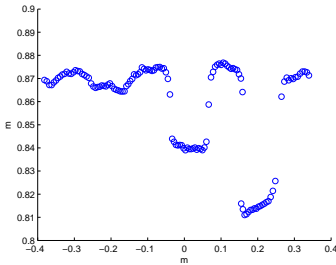
block covered by the black paper sheet was indicating an object height of about ten cm. This shows that the material reflectivity, density and colour indeed has great impact on the resulting range measurements. In some experiments, it was very hard to even distinguish the styrofoam from the ground level using the setup in Figure 4.6. To investigate the importance of reflectivity and the accuracy of the



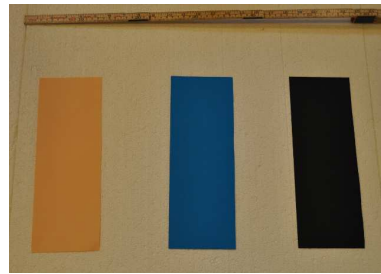
(a) Light-grey sheet background



(b) Light-grey sheet background - Photo



(c) Styrofoam background.



(d) Styrofoam background - Photo

Figure 4.8: Material reflectivity importance. Comparison between a light-gray sheet and styrofoam as background material, with orange, blue and black paper sheets placed on them as identifiers.

LIDAR further and to find good identifiers a second experiment was carried out. Paper sheets of three different colours were placed on two different background materials, light grey sheet and styrofoam. The paper had the colours of orange (apricot), blue and black. The experimental setup can be seen Figure 4.8. When placed on the sheet, the orange paper gave a negative contribution to the range measurements in comparison to the background material. The blue paper sheet was hardly visible in the sweep, while the black paper gave a positive contribution to the range measurements (Figure 4.8a and 4.8b). If placed on styrofoam, the result is somewhat the opposite. The black paper was still very easy to distinguish from the background sheet. The blue paper also gave a large positive contribution to the range measurements. The orange paper blended in best of the three colours (Figure 4.8c and 4.8d).

As a result of these experiments, the remaining data acquisitions made in this

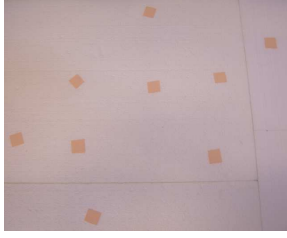
work included orange paper, as the identifiers for the camera was orange paper. Styrofoam was used both as ground material and terrain material. It should also be noted that other things such as lighting and shadows also affects the LIDAR. The findings of the experiments made here are backed up by [9], who does an extensive comparison of two LIDARS, a SICK LMS-200 and the Hokuyo URG-04-LX. They find the same importance of reflectivity properties for the URG-04-LX. There is another version of the URG-04-LX sensor available on the market, called UTM-30LX. This LIDAR provides intensity information from the received signals that can be used to compensate for material properties.

4.3 Scenario one - blocks

As a first scenario, a terrain made of square block of known volume and a relatively kind "flight route" was chosen. A kind flight route in this case means that the sensor platform orientation was not tilted back and forth much. Square blocks provide discontinuities that are generally hard to model, but it is easy to measure the true volume. The initial scenario was done by acquiring two data sets. The first one without any terrain placed in the scene and the other one with a block with known volume of 4dm^3 . An introduction to the scenario can be seen in Figure 4.9. The figure shows the raw data gathered by the LIDAR. The data consists of a total of 11000 measurement points and was divided into an estimation data set and an validation data set to make it possible to cross validate. Each set therefore consisted of about 5500 measurement points. The resolution of the elevation grid map was set to 2cm^2 . The methods introduced in Chapter 3 are compared in Table 4.1 and in Figure 4.10 and Figure 4.11.

One way of comparing the performance of the implemented algorithms is to see how much data is needed to produce reliable maps. In Table 4.1 the amount of LIDAR data available for the iterations is varied. It is a natural thought that the volume difference would be better if more data is available. The experiments showed that a minimum of 10-15% of the data-set of 5500 points (i.e 500-750 points) was needed in order to produce maps that accurately modelled the shape of the terrain. If fewer data-points are used the volume difference is approximately the same, however the look of the volume measured started to look deformed. The sharp edges of the squared block were blurred more and more regardless of method used.

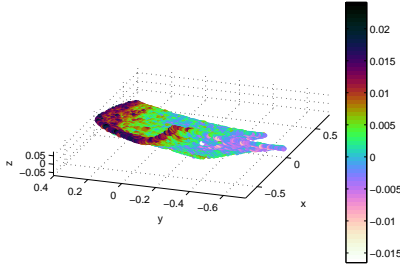
The reason that the volume estimates stay somewhat constant is the central limit theorem, which states that the result of randomly adding and removing points will eventually sum up to zero. When randomly altering the number of used data, some contributions will add to the volume difference, while others will reduce the volume difference. Figure 4.10 and Figure 4.11 shows the resulting maps for two different amounts of data-points are used. It shows that the three initial methods are basically the same if fewer data-points is used. This is because



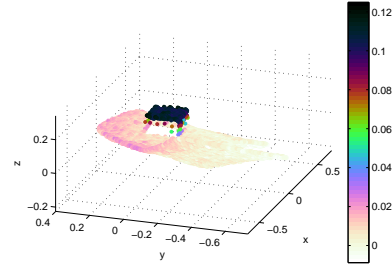
(a) Flight one. Image.



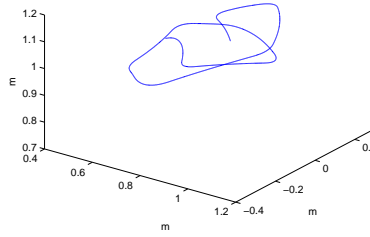
(b) Flight two. Image.



(c) 100 % Estimation LIDAR data flight one. Note the scale.



(d) 100 % Estimation LIDAR data flight two.



(e) Flight route.

Figure 4.9: Scenario one.

the importance of choosing averaging, median or weighting algorithms is not as important if there is only a few points available in each cell. Looking at the GP based methods, the NGP can arguably be said to model the shape of the surface better. If 100% of the data is used it is harder to determine which method that is the best. Adding more measurements theoretically means that the resolution of the grid can be lowered. Lowering the resolution to 1cm^2 did not change the volume difference errors in Table 4.1 significantly. For the GP-based methods, the result of adding more measurement points is that the length scales decrease. Such

Volume difference error (dm^3)					
Data used:	5%	10%	20%	50%	100%
Average	0.8	0.5	0.6	0.6	0.6
Median	0.4	0.6	0.6	0.6	0.6
Weighted	0.5	0.7	0.8	0.8	0.8
GP	-0.1	0.4	0.7	0.6	0.5
NGP	0.4	0.5	0.6	0.6	0.5

Table 4.1: Volume difference error of scenario one. Average over ten runs.

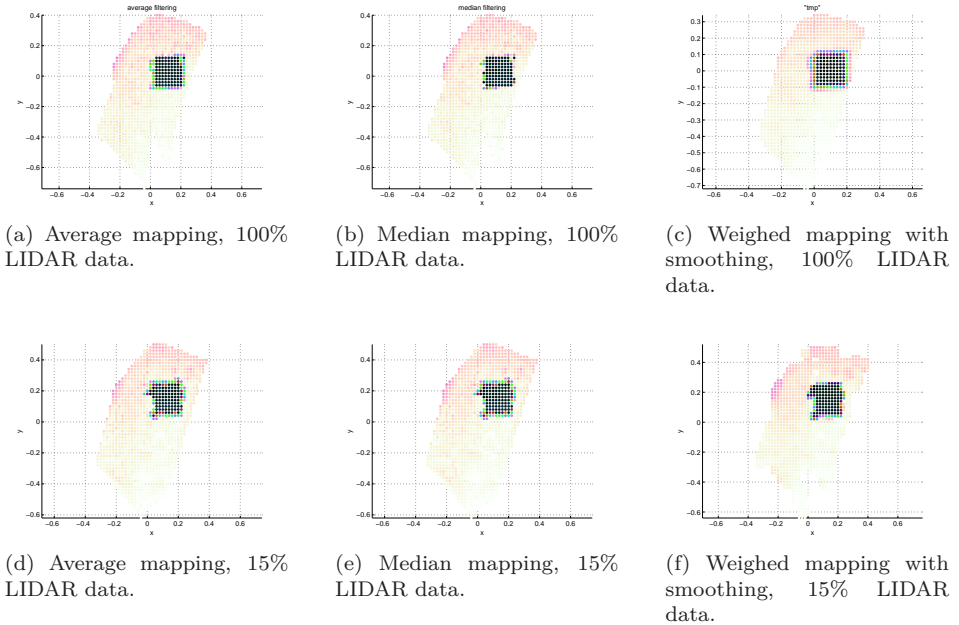


Figure 4.10: Scenario one, initial methods for different LIDAR data density.

optimal choice of resolution versus length-scales for the NGP was not searched for due to the need for retuning the algorithm. The volume differences estimated in Table 4.1 are generally converging towards a volume difference larger than the true value. Even if a part of this offset may relate to the shape of the squared block being hard to model and the edges being smoothed out over the step, it also indicates that the LIDAR may have an offset despite the calibration. An average of the volume difference error in Table 4.1 is somewhere around 0.6dm^3 . This offset needs to be taken into consideration together with the accuracy of the LIDAR, the observations about LIDAR sensitivity and the resolution of the grid map. In the manual for the Hokuyo URG LIDAR sensor the accuracy for range measurements made at distances where these measurements are taken, around one meter above the ground, is about one percent. To get a feeling for it, an error of one centimetre on a twenty by twenty styrofoam block gives rise to a volume of

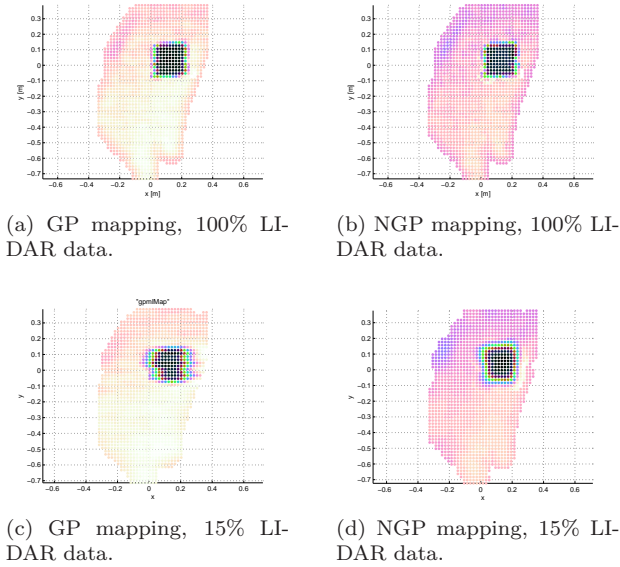


Figure 4.11: Scenario one, GP based methods for different LIDAR data density.

$0.2\text{m} \cdot 0.2\text{m} \cdot 0.01\text{m} = 0.0004\text{m}^3 = 0.4\text{dm}^3$. As the results of Table 4.1 are averages over multiple measurements, all of the volume difference error can not be blamed on the LIDAR. However, assuming that the specifications made in the LIDAR manual are optimistic, as they often are, together with the observations in Section 2.3.1, it is reasonable to say that it is hard to get more accurate volume difference measurements using the Hokuyo URG-04-LX LIDAR.

4.4 Scenario two - natural terrain

A second experiment containing more naturally shaped terrain was also performed. The industrial robot was used to perform two flights over the terrain shown in Figure 4.12. The raw data set consisted of a total of 49000 measurement points, divided into estimation and validation sets this gives around 24500 points per set. The true volume differences between the two flights were measured by hand to about 14.5dm^3 . This true volume was measured by pouring water into a container and measuring how much the terrain altered the water level, hence the measured volume is uncertain. A comparison between the methods is performed in Table 4.2 and illustrated in Figure 4.13 and Figure 4.14.

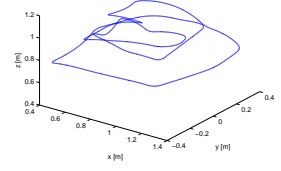
In Table 4.2 the amount of measurement points used is varied and compared to the estimated volume difference in the same way as in Scenario one. As the true volume of 14.5dm^3 was measured with low accuracy, it is hard to determine which method is best from the results in Table 4.2. It can also be noted that



(a) Scenario two, flight one. Image.



(b) Scenario two, flight two. Image.



(c) Flight route.

Figure 4.12: Scenario two.

the offset on volume-error is not always positive, which indicates that some of the offset in Scenario one is related to discontinuities of the squared block and not just the LIDAR. All methods proved to produce fairly accurate maps if all data was used for the estimates. However, the computational time for the NGP method started to become very large when the data increased. This is due to the additional operations needed to find nearest neighbours in order to determine the EST tensor. When 2500 data points were used the total computation time for the NGP algorithm was around 18 minutes on a standard laptop computer. When all 22500 points were used the time used increased to around 5 hours. Figure 4.14 and 4.13 shows the appearance after applying the average mapping, GP and NGP mapping in the case where 5% data was used. This produced the same amount of measurements per grid as in Scenario one. From Figure 4.14 and Figure 4.13 it is hard to visually determine which method produces the best result.

Volume difference error (dm^3)						
Data used:	1%	5%	10%	20%	50%	100%
Average	-0.5	-0.2	-0.1	-0.1	0.0	0.0
Median	-0.7	-0.1	-0.1	0.0	0.0	0.1
Weighted	-1.3	-0.3	-0.3	-0.1	0.2	0.2
GP	0.2	0.4	0.4	0.3	0.2	0.2
NGP	-0.4	0.5	0.2	0.2	0.1	0.4

Table 4.2: Volume difference error of scenario two. Averages over multiple runs (4-10).

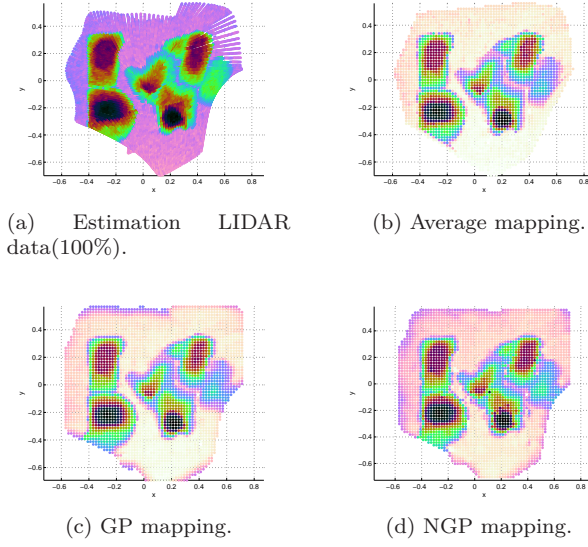


Figure 4.13: Scenario two, comparison experiment made over Figure 4.12b. 100% data used.

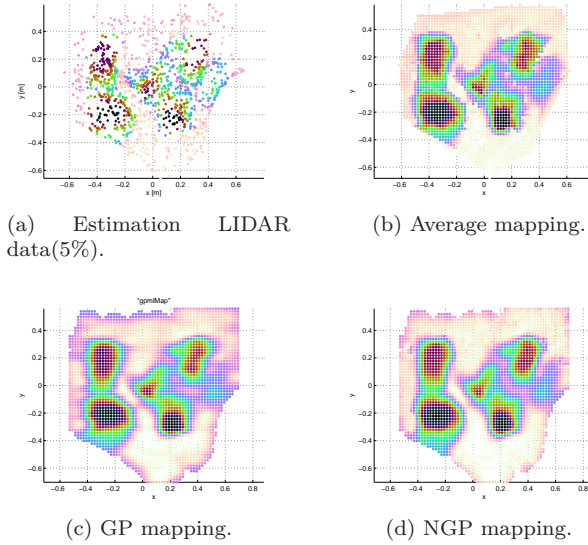


Figure 4.14: Scenario two, comparison experiment made over Figure 4.12b. 5% data used.

4.5 Discussion

Considering the results from these preliminary scenarios using an industrial robot it is clear that measuring terrain volume difference by the use of an UAV equipped with a LIDAR should be possible. The two scenarios studied in this work measured changes in volume to the same magnitude as the true known volume, both in scenarios with simple blocks as terrain and in scenarios with more naturally shaped terrain. In the first scenario there was a positive offset on the volume measurements regardless of the method used for mapping. This offset has to be taken in consideration together with the resolution of the LIDAR, and that the discontinuities in the terrain are hard to model. Section 4.2.2 showed that the Hokouyo URG-04-LX had several artefacts related to reflectivity, which may also contribute to the error in volume difference. In the experiments the LIDAR is also operating only on a very small part of this working area. This makes the experiments made slightly out of scale compared to a real flight scenario. The volume errors was around 0.5dm^3 in these small scaled experiments, and the Hokouyo LIDAR sensor has got an operating range of $0 - 4\text{m}$. Further the terrain models was about ten centimetres high. These numbers does not directly scale up to a real flight scenario, with hundreds of cubics of terrain and a LIDAR that operates at $0 - 80\text{m}$ range. Therefore, the offset on the volume measurements could very well be connected to the choice of LIDAR, experiment setup and the scale of the experiment.

Further, the experiments like the ones made in this work can be carried out on many different levels hardware wise. The experiments introduce an amount of uncertainties by default. For example using a non real time operating system as Ubuntu has disadvantages. Only the IMU provided an internal time stamp for the measurements. Therefore, a time stamp was placed on sensor measurements when they arrived in Player. Using timestamps made by the computer, one has to be aware of the possible time delays in the different ports used (USB, fire-wire and serial ports) and that Player is not the only process running on the computer. The synchronization of the two different timestamps, the Player timestamps and the industrial robots, is another uncertainty introduced in the experiment.

Looking closer at the different methods for mapping, they all provided similar measurements in the case of dense data. With the number of measurements decreasing the GP and NGP method provided visually more accurate maps, compared to the other mapping methods, due to the inherent ability of GPs to better handle sensor incompleteness. The visual appearance of the three initial methods was almost the same. The weighting filter suffers from the scaling problem mentioned above. As the LIDAR is operating in a small part of its working range, the difference in variance for the measurements will be small compared to an experiment where the whole LIDAR operating range is used. It should also be noted that the computational cost of adding more measurements increased especially for the NGP where computations took long time for dense data, even when applying the locally moving window. This is due to the need for sorting of the measurements points

in order to calculate the local gradients of the EST tensor.

Further, it should also be noted that the constraint on NGP which says that the kernels should vary smoothly across the input space have been hard to meet implementation wise. Even with additional averaging steps taken when searching for the local-learning rate the method was still sensitive to data overfit. A major criticism to many machine learning algorithms is also the extensive need for tuning of the algorithm. So also for the NGP implementation, where especially tuning of the learning-rate sigmoid was hard.

Chapter 5

Concluding remarks

In this work five possible LIDAR mapping methods have been implemented and evaluated with the goal of finding volume differences between different data sets. Experiments using an industrial robot were made to gather data, as the data from real flight scenarios were not available. This chapter summarizes the results in Section 5.1, and takes a look at possible future development in Section 5.2.

5.1 Conclusions

During this work the approach has been to try the simplest mapping solutions first. Three initial mapping methods were tested, using simple averaging and median methods as well as a slightly more complicated method that weights measurements against their corresponding uncertainty in a Kalman like way. In combination with linear interpolation these methods proved to produce quite good results in the case of very dense data-sets. For sparse data-sets the measured volume difference remained about the same, however the resulting maps were starting to get deformed visually.

As a complement to the initial methods two probabilistic Bayesian methods based on Gaussian Processes were implemented. GP are, in contradiction to the previously mentioned algorithms, based on regression. Regression based approaches do not associate the measurements into a certain cell, but instead uses the data to create a valid model that mathematically explains the acquired data. This increases the ability to handle incomplete data, and hence the GP-based methods should theoretically perform better in case of very sparse data sets. The experiments partly supported this. In this work, a moving average window was applied making the method of GP more scalable. GP based methods also provides uncertainty information, which is a benefit with this regression based approach. Theoretically, GP has a built in smoothing effect that rimes well with modelling flowing landscapes but tends to over-smooth steps. This lead to the implementation of a non-stationary GP (NGP) model using a covariance function that adapts individual kernels to measurements by the use of a local EST tensor. The NGP manage

to produce good maps even with sparse data sets, and considering an artificial data set it also provided the wanted trade-off between handling of discontinuities and handling of flat regions. Introducing the NGP model adds another level of complexity to the mapping problem, and while doing the experiments, problems with overfitting occurred due to the need for smoothly varying kernels. Computational time also increased as a search for nearest neighbours is needed. GP provides a good smoother, which handles incompleteness and provides uncertainty information, and is fairly easy to use and implement. The next step, introducing NGP by adapting local kernels through an EST tensor, do not quite pay for itself in terms of complexity versus performance.

The experimental setup affected the results to a high degree. There are built in uncertainties in the experiment from the start, for example with timestamps on data being set in a non ideal way. Further, the LIDAR used is very sensitive to reflectivity, and was only operating on a very small part of its working area, which makes the experiments slightly out of scale. Getting to know the LIDAR and making sure that gathered data is as good as possible is therefore important.

The experiments showed that to create topographical maps for volume estimation purposes, simple methods may very well be accurate enough to produce valid estimates. For CybAero AB to succeed in making volume measurements with a higher resolution than the previously mentioned ten GPS measurements per hectare is therefore very likely. The determining factor for a real case flight is the density and accuracy of the LIDAR data. In case of sparse LIDAR data regression based methods like GP and NGP might be better. The need for methods based on regression, such as GPs decreases with the density and accuracy of the LIDAR data. Focusing on knowing *where* a measurements is taken, and getting a good raw-data set is therefore of great importance.

5.2 Future work

There are two ways of looking at possible future work following this thesis. One based on the hardware and improvements possible there, and the other one looking at the different algorithms and possible improvements in that area.

During both the experiment and during real flight a camera is used. This camera can provide information for adding textures to the map. This will require some work as ways of merging several images into one texture is needed. One of the problems during the thesis has been that the LIDAR was very sensitive to material reflectivity properties and densities. In this thesis these problems have been solved by altering the experimental environment. Theoretically, information from a camera could provide ways of cancelling these unwanted effect of the LIDAR. Testing another LIDAR model that provides intensity information would also be interesting in the experimental environment, for example the Hokuyo UTM-30LX.

Looking at the developed algorithms the method of combining a local windowing approach suggested in [22] with the non-stationary family of Gaussian Processes introduced in [12] theoretically provides a good way of handling incomplete data and discontinuities. There are several areas where there is room for improvement. First, the non-stationary approach includes a search for nearest neighbours, for example when calculating the local elevation tensor (EST). On an implementation basis it is therefore important to sort and organize the LIDAR point measurements in order to quickly find the nearest neighbours. Vasudevan, Ramos et al suggested a KD-tree for sorting and storing the LIDAR point clouds [22]. This is likely to be a good method for structuring the data.

Second, the EST tensor and the algorithm for finding the local kernels showed a great need for tuning and smoothing to avoid overfitting, which is a critique for many machine learning algorithms. It would therefore be interesting to further investigate the possibilities of finding the local kernels using probabilistic approaches. The methods for estimating kernels at unseen locations can also be improved. A very appealing idea would be to learn a second "hyperprocess" over the kernel parameters, and try to estimate the kernels at the unknown location in a more principled way.

More work can also be done on using the uncertainty map provided by the GP framework in order to improve the accuracy of the estimates. For example, if the sampling stage is included iterations of GPs are possible. New sample points can be picked from the data set where uncertainties are high in order to increase performance. This was briefly tested during the thesis, and while time-consuming it worked fairly well.

There is also the possibility of searching for other covariance functions than the one used in this work. Vasudevan, Ramos et al used a covariance function from another family based on a neural-network covariance function [22]. Their particular implementation was patented, but it shows that several ways of utilizing the GP framework for solving the robotic mapping problem are possible.

Lastly, there are other probabilistic mapping methods well worth looking into. During the work with this thesis, the use of Occupancy Grid Maps (OGM) was neglected due to the authors belief that it would not utilize the 2.5D assumption in a good way. Recently, Wurm et. al presented Octomaps as an extension to the well documented OGM [23]. The Octomap builds upon a tree based structure which leads to good memory representation even in three dimensions, providing good scalability and at the same time ensuring good accuracy. Octomaps may very well be the best implementation available at the time this thesis is summarized, and it is strongly suggested that CybAero AB looks into it in the case easier mapping techniques do not provide the desirable accuracy in measuring volume difference.

Bibliography

- [1] Lars B. Cremean and Richard M. Murray. Uncertainty-based Sensor Fusion of Range Data for Real-time Digital Elevation Mapping (RTDEM). In *Proceedings of the IEEE International Conference on Robotics and Automation*, Barcelona, Spain, April 18 - 22 2005.
- [2] Slawomir Grzonka, Giorgio Grisetti, and Wolfram Burgard. Towards a navigation system for autonomous indoor flying. In *Proceedings of the IEEE International Conference on Robotics and Automation*, Kobe, Japan, May 12 - 17 2009.
- [3] C. Harris and M. Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, page 50. Manchester, UK, 1988.
- [4] Alexander Kleiner and Christian Dornhege. Building Elevation maps from laser range data. Rescue Robotics camp - Rome, Italy 2006.
- [5] Jack B. Kuipers. *Quaternions and rotation sequences*. Princeton University Press, 1999. ISBN 0-691-05872-5.
- [6] Tobias Lang. Gaussian processes for terrain modeling. Master's thesis, Albert-Ludwigs-University, Department of Computer Science, Freiburg, 2007.
- [7] Tobias Lang, Christian Plagemann, and Wolfram Burgard. Adaptive non-stationary kernel regression for terrain modeling. In *Proceedings of Robotics: Science and Systems*, Georgia, USA, June 27-30 2007.
- [8] Jeff Leal, Steve Scheduling, and Gamini Dissanayake. 3D terrain mapping: A stochastic approach. In *Proceedings of the Australian Conference on Robotics and Automation*, pages 135–140, Sydney, Australia, November 14 - 15 2001.
- [9] K.H. Lee and R. Ehsani. Comparison of two 2D laser scanners for sensing object distances, shapes, and surface patterns. *Computers and electronics in agriculture*, 60(2):250–262, 2008.
- [10] Markus Middendorf and Hans-Hellmut Nagel. Empirically convergent adaptive estimation of grayvalue structure tensors. *Pattern Recognition*, pages 66–74, 2002.

- [11] Naturvårdsverkets författningssamling 2004. Paragraph 14. Available May 11 2010 from www.naturvardsverket.se/Documents/foreskrifter/nfs2004/NFS2004-4.pdf.
- [12] Christopher .J Paciorek and Mark .J Schervish. Nonstationary covariance functions for Gaussian process regression. In *Proceedings of the 2003 conference of Advances in Neural Information Processing Systems 16*, page 273. The MIT Press, 2004. ISBN 0262201526.
- [13] C.J. Paciorek. *Nonstationary gaussian processes for regression and spatial modelling*. PhD thesis, Carnegie Mellon University, Pittsburgh USA, 2003.
- [14] Carl E Rasmussen and Chrisopher K.I Williams. *Gaussian processes in machine learning*. The MIT Press, 2006. ISBN 978-262-18253-9.
- [15] Randall Smith, Matthew Self, and Peter Cheeseman. Estimating uncertain spatial relationships in robotics. *Autonomous robot vehicles*, 1:167–193, 1990.
- [16] Cyrill Stachniss. *Robotic Mapping and Exploration*. Springer, 1 edition, 2009. ISBN 978-3-642-01096-5.
- [17] Brian L Stevens and Frank Lewis. *Aircraft control and simulation*. John Wiley & Sons Inc, 2 edition, 2003. ISBN 0-471-37145-9.
- [18] Sebastian Thrun. Robotic mapping: A survey. *Exploring artificial intelligence in the new millennium*, pages 3,17–18, 2002.
- [19] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. MIT Press, 1 edition, 2006. ISBN 978-0-262-20162-9.
- [20] Sebastian. Thrun, Mark Diel, and Dirk Hähnel. Scan alignment and 3-D surface modeling with a helicopter platform. In *Field and Service Robotics*, pages 287–297. Springer, 2006.
- [21] David Törnqvist. *Estimation and detection with applications to navigation*. PhD thesis, Department of Electrical Engineering, Linköping University, Sweden, 2008.
- [22] Shrihari Vasudevan, Fabio Ramos, Eric Nettleton, and Hugh Durrant-Whyte. Gaussian process modeling of large-scale terrain. *Journal of Field Robotics*, 26(10), 2009.
- [23] 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*, Alaska, USA, May 3 2010.
- [24] Cang Ye and Johann Borenstein. A new terrain mapping method for mobile robots obstacle negotiation. In *Proceedings of the UGV Technology Conference at the 2003 SPIE AeroSense Symposium*, pages 21–25, Orlando, USA, April 21-25 2003.

Appendix A

Quaternions

This appendix presents the basic equations needed to understand rotations using quaternions. There are several properties that are vital and interesting when working with quaternions. However, this section only intends to cover the basics needed for this work. A thorough explanation of quaternions and how its used for rotations is made by Kupiers [5]. David Törnqvist also provides a good explanation of quaternion rotation [21]. Quaternions is an attempt to generalize the complex plane to three dimensions that was introduced by W.R Hamilton [17]. A quaternion is a four-touple of elements defined as:

$$q = q_0 + q_1i + q_2j + q_3k = \begin{pmatrix} q_0 \\ \mathbf{q} \end{pmatrix} \quad (\text{A.1})$$

where

$$i^2 = j^2 = k^2 = ijk = -1, ij = k, jk = i, ki = j = -ik$$

The quaternion can be divided into a scalar part, q_0 , and a vector part, \mathbf{q} . Working with quaternions follows the same basic algebraic rules as vectors, with the exception that multiplication is not commutative. That is, the order of the quaternions matters when multiplying quaternions. Multiplication of quaternions p and q is defined as:

$$p \odot q = \begin{pmatrix} p_0 \\ \mathbf{p} \end{pmatrix} \odot \begin{pmatrix} q_0 \\ \mathbf{q} \end{pmatrix} = \begin{pmatrix} p_0q_0 - \mathbf{p} \cdot \mathbf{q} \\ p_0\mathbf{q} + p_0\mathbf{p} + \mathbf{p} \times \mathbf{q} \end{pmatrix}, \quad (\text{A.2})$$

where \times is the standard cross product for vectors. Further, the inverse of a quaternion is defined as

$$q^{-1} \odot q = q \odot q^{-1} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}^T \quad (\text{A.3})$$

and can be calculated as

$$q^{-1} = \begin{pmatrix} q_0 \\ \mathbf{q} \end{pmatrix}^{-1} = \begin{pmatrix} q_0 \\ -\mathbf{q} \end{pmatrix} / \sum_{i=0}^3 q_i^2 \quad (\text{A.4})$$

A.1 Rotation using quaternions

There are several properties that need to hold in order for a rotation to be valid. A valid rotation should preserve the length of the rotated vector, and an inverse rotation should bring the vector back to its original shape for example. Working with quaternions there are two ways of doing rotations in a way that these properties holds. Either a vector is rotated about another vector, or a coordinate frame is rotated around another, fixed coordinate frame. The only thing that differs between these two representations is the sign of the scalar part of the quaternion. For the application in this work, rotation around coordinate frames is used, see Figure A.1. If q is a unit quaternion ($\text{norm}(q) = 1$) such rotation can be described as

$$v = q^{-1} \odot u \odot q, \quad (\text{A.5})$$

where v is the rotated output vector and u is the vector to be rotated,

$$v = \begin{pmatrix} 0 \\ \mathbf{v} \end{pmatrix} \text{ and } u = \begin{pmatrix} 0 \\ \mathbf{u} \end{pmatrix}. \quad (\text{A.6})$$

If one extends the expression in (A.5) using the formula for quaternion multiplication given by Equation A.2 and quaternion inversion given by Equation A.4 the result of the quaternion rotation is

$$v = \begin{pmatrix} \mathbf{q}\mathbf{u}q_0 - (q_0\mathbf{u} - \mathbf{q} \times \mathbf{u}) \cdot \mathbf{q} \\ (\mathbf{q} \cdot \mathbf{u})\mathbf{q} + q_0(q_0\mathbf{u} - \mathbf{q} \times \mathbf{u}) + (q_0\mathbf{u} - \mathbf{q} \times \mathbf{u}) \times \mathbf{q} \end{pmatrix} \quad (\text{A.7})$$

Doing the calculations in Equation A.7, the rotation can be rewritten to a matrix multiplication where $R(q)$ is the rotation matrix that allows for the change of coordinate frame,

$$\begin{pmatrix} 0 \\ \mathbf{v} \end{pmatrix} = \begin{pmatrix} 0 \\ R(q)\mathbf{u} \end{pmatrix} \quad (\text{A.8})$$

$$R(q) = \begin{pmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 + q_0q_3) & 2(q_1q_3 - q_0q_2) \\ 2(q_1q_2 - q_0q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2q_3 + q_0q_1) \\ 2(q_1q_3 + q_0q_2) & 2(q_2q_3 - q_0q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{pmatrix} \quad (\text{A.9})$$

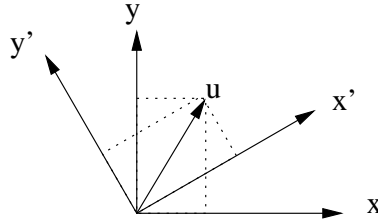


Figure A.1: Rotation made around a fixed coordinate system can also be seen as a rotation about a vector, u .

Appendix B

Abbreviations

Abbreviations:

EST	Elevation Structure Tensor
GP	Gaussian Process
GPS	Global Positioning System
LIDAR	Light Detection And Ranging (a laser range finder)
KD-Tree	K-dimensional tree (storage structure)
NGP	Non-stationary Gaussian Process
OGM	Occupancy Grid Mapping
SLAM	Simultaneous Localization and Mapping
TIN	Triangulated Irregular Network
UAV	Unmanned Aerial Vehicle

Technical terms:

\mathcal{D}	Data set $\mathcal{D} = \{\mathbf{x}_i, y_i i = 1, \dots, n\}$
$h_{k,l}$	Grid map at position k, l
k_{SE}	Squared exponential covariance function
k_{NSE}	Non stationary squared exponential covariance function
\mathbf{q}	Quaternion
Σ_i	Kernel matrix for training input i .
\mathbf{x}_i	The i th training input (x and y coordinates)
\mathbf{X}	Vector of \mathbf{x} measurements (vector of x and y coordinates)
\mathbf{x}^*	A regression point (x and y coordinates)
\mathbf{X}^*	A test grid (multiple \mathbf{x}^*)
y_i	Target at input \mathbf{x}_i (z coordinate)
\mathbf{y}	Vector of targets (z coordinates)
y^*	Predicted target (z coordinate).