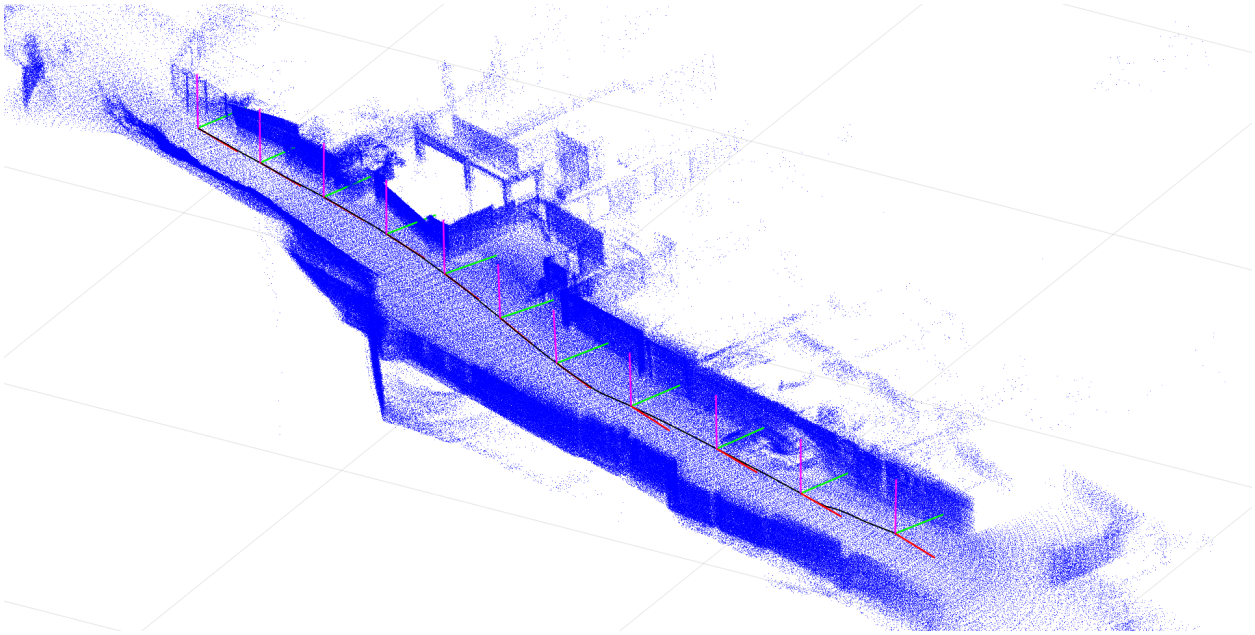
CHALMERS
UNIVERSITY OF TECHNOLOGY



Localisation using LiDAR and Camera

Localisation in low visibility road conditions

Master's thesis in Systems, Control and Mechatronics

THANG HOANG
VIKTOR PALMQVIST BERNTSSON

Department of Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2017

Master's thesis 2017:EX054

Localisation using LiDAR and Camera

Localisation in low visibility road conditions

THANG HOANG
VIKTOR PALMQVIST BERNTSSON

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Division of Systems and Control
Chalmers University of Technology
Gothenburg, Sweden 2017

Localisation using LiDAR and Camera
THANG HOANG

VIKTOR PALMQVIST BERNTSSON

Supervisor: Balázs Adam Kulcsár, Department of Electrical Engineering
Supervisor: Anne Piegsa, Semcon Sweden AB
Examiner: Yiannis Karayiannidis, Department of Electrical Engineering

Master's Thesis 2017:EX054
Department of Electrical Engineering
Division of Systems and Control
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: 3D LiDAR map with position and orientation of vehicle

Localisation using LiDAR and Camera
THANG HOANG
VIKTOR PALMQVIST BERNTSSON
Department of Electrical Engineering
Chalmers University of Technology

Abstract

Localisation of the vehicle is one of the most important part of autonomous driving and can be achieved in many different ways. Most autonomous vehicles today are not able to navigate in complex environments, for instance snow covered roads or heavy rain. This thesis proposes a method for localising a vehicle using LiDAR and Camera fused together with INS and GNSS. The method is intended to work in the complex environments mentioned earlier. The system consists of a vehicle equipped with a 3D scanning LiDAR, a forward looking camera, a combined INS/GNSS unit and a 3D map over the area of interest. The LiDAR and camera are used together to measure distances to points of interest in the 3D map which are then used to triangulate the position of the vehicle. The position from the triangulation is then fused with INS and GNSS in an Ensemble Kalman Filter along with a motion model of the vehicle to estimate the position of the vehicle. It is concluded that the system is promising with a mean translational error of about 40 centimeters which is a good result considering that DGPS has about the same accuracy.

Keywords: LiDAR, Camera Features, Ensemble Kalman Filter, 3D Point Cloud, Localisation

Acknowledgements

We would like to thank all the people who have helped us during this thesis. Firstly we would like to thank our examiner and supervisor at Chalmers University Yiannis Karayiannidis and Balázs Adam Kulcsár for their enthusiasm in the thesis and helpful guidance during pleasant meetings. We would also like to thank our supervisors at Semcon AB Anne Piegsa, Jens Henriksson and Carl Johan Aldén for the support and technical expertise. Furthermore we would like to thank the people at the department of Systems and Controls at Semcon AB for helpful advice and pleasant company.
The project would not have been possible without data to test the proposed algorithm and we would therefore like to thank the people at Karlsruhe Institute of Technology for the KITTI dataset, the people at Oxford University for the RobotCar dataset and the people at University of Michigan for the Ford Campus Vision dataset.

Thang Hoang and Viktor Palmqvist Berntsson
Gothenburg, June 2017

Contents

List of Figures

List of Tables

List of Acronyms

DGPS	Differential Global Positioning System
DoG	Difference of Gaussians
EKF	Extended Kalman Filter
EnKF	Ensemble Kalman Filter
FAST	Feature from Accelerated Segment Test
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
IMU	Inertial Measurement Unit
INS	Inertial Navigation System
KITTI	Karlsruhe Institute of Technology and Toyota Technological Institute
LiDAR	Light Detection And Ranging
MEMS	Micro-Machined Electromechanical System
MOEMS	Micro-Optical Machined Electromechanical System
MSER	Maximally Stable Extremal Regions
NVDB	Nationella Vägdatabasen
RTK	Real Time Kinematics
SIFT	Scale Invariant Feature Transform
SLAM	Simultaneous Localisation And Mapping
SURF	Speeded Up Robust Features
TOF	Time of Flight

1
Introduction

In the wake of the rapid development of technology more and more things in society and our lives becomes autonomous. A topical subject of today is self-driving cars which currently already are on the roads in real life testing. In the world's first large-scale autonomous project, by Volvo Car Group, 100 autonomous driving cars will be used by real-world customers this year, 2017 [4]. Autonomous household appliances, such as robotic vacuum cleaner and robotic lawn mower, are becoming more common for taking care of our everyday work. For a mobile autonomous agent that can move freely it is required that it knows where to move and how to do it. It must have the ability to move from one place to another and handle all the unexpected scenarios that may appear along the path. To achieve this information about the surrounding environment and the current position is essential. Exteroceptive sensors can be used to provide knowledge of the surrounding environment, but in complex environments, for instance snow covered roads, rain and fog, the information may be insufficient. This places higher demands on the accuracy of the current position information.

1.1 Background

Most car companies of today are developing self-driving vehicles but most of them have not yet solved the problem how to navigate when the visibility of the road edges are limited, for instance when there are no lane markings on the road. This can be caused by many factors, in colder environment one cause can be snow which erases lot of visible information. There are also many roads which are not as standardized as car roads for instance bicycle paths which can have different or no lane markings depending on which type of path it is.

Most autonomous vehicles today have not been tested in the snow according to [5]. The tests have mostly been carried out in sunny environments. According to Ford Motor Company the future of autonomous vehicles cannot depend on ideal weather and lighting conditions since more than 70% of the US citizens lives in snowy regions [6].

One possible way to determine the position of the vehicle is knowing the initial position and from that position use dead reckoning from wheel encoders and an Inertial Measurement Unit (IMU). In theory, this works well if all sensor inputs are noise free. Since this is not the case this method will only work for small distances since the noise from the sensors will accumulate and the error will grow over time.

There are many ways to locate a vehicle to get an absolute location. The first and most common is GPS which is fitted to every smart phone and also in most modern vehicles. GPS is a well developed system which works over the whole world 24 hours per day [7] with an accuracy of about 5 meters which is sufficient in most cases. However when positioning vehicles for autonomous driving this accuracy is not enough, it can not be more than 30-50 cm similar to the Differential Global Positioning System (DGPS). The accuracy of the GPS can be greatly improved using two different methods, DGPS and Real Time Kinematics (RTK), both of these systems utilises reference stations placed in the vicinity of the vehicle. With these systems an accuracy of a few centimeters can be achieved, which is sufficient for autonomous driving and also used for many projects in controlled environment. The downside to this however is that these base stations have to be placed a few hundred meters apart which is costly to implement in a large scale such as a whole city.

Autonomous vehicles today utilise camera to determine the local position and where on the road the vehicle is. This is done by looking for lane markings in the road, which is easy for the camera to detect, and some type of GPS for the global positioning. To be as efficient as possible most autonomous vehicles use some kind of Simultaneous Localisation And Mapping (SLAM) algorithm which means that the vehicle maps the environment at the same time as it locates itself. This is done by fusing data from multiple sensors and a map of the environment which is updated as the vehicle travels along its path.

There are many types of SLAM algorithms, the static approach utilises Kalman Filters, Particle filters and matching of range data from for instance Light Detection And Ranging (LiDAR). The SLAM algorithms can be divided into two different types [8], on-line and full SLAM. Filtering algorithms estimate the variables every time step using only previous data, this is called on-line SLAM. Full SLAM, or smoothing algorithms utilises the complete data set to estimate the variables.

1.2 Purpose and Scope of the Thesis

The goal of this thesis is to develop an algorithm for localisation of a vehicle in conditions when the road surface, lane markings and curbs are not visible with a similar accuracy as DGPS (30-50 cm). This will be done by using LiDAR, camera, IMU and GPS. A 3D map from LiDAR measurements shall be created and used as a baseline for comparison with feature matching from an image. The goal for the system is to be able to run it in real time and since the LiDAR operates at 10 Hz that is the goal for the algorithm.

The method proposed for localisation of a vehicle involves camera feature matching together with LiDAR measurements in a 3D map fused together with GPS and IMU using an Ensemble Kalman Filter (EnKF).

The purpose of this thesis is to test if the proposed solution for the localisation

problem is feasible.

1.3 Delimitations

There is small amounts of data with camera, LiDAR and Inertial Measurement Unit (IMU) for both snowy weather and and ideal weather conditions on the same road and therefore the project is delimited to use open source data with almost ideal weather conditions.

The project is delimited to only test and evaluate the algorithms in simulation environments due to the fact that there is no working prototype of the vehicle. There is also no available similar vehicle with the correct sensor setup and with access to all the data.

1.4 Method and Solution Overview

As mentioned, the goal of the thesis is to localise a vehicle in complex road conditions. In this section the methods used to do this is briefly explained. Firstly, the vehicle needs a couple of sensors: a camera, a LiDAR and a combined GNSS/INS unit. Then a 3D map over the area of interest must be ground truthed in advance of using the proposed algorithm.

Having the senors and the map, the vehicle triangulate its position in the point cloud using ranges from the LiDAR to determine distances to points of interest. To get a better and more reliable estimate of the position the output from the algorithm is fused together with the a motion model of the vehicle and the input data from the GNSS/INS unit.

1.5 Related Work

A. Y. Hata et al. [9] uses a 3D LiDAR to localise a vehicle in urban environment to enhance the position from a Global Navigation Satellite System (GNSS). To do the localisation, curbs and road signs are detected using LiDAR and are matched to a pre built occupancy grid using Monte Carlo Localisation. To build the occupancy grid a 3D LiDAR equipped to the car measures the distance to the curbs and signs and place them in the occupancy grid using GNSS and RTK. This method gives a absolute accuracy of about 0.2 meters

T. Caselitz et al. [10] proposed an algorithm for localisation using a monocular camera and a pre build 3D map from LiDAR. The proposed algorithm uses visual odometry that uses bundle adjustments to create a sparse set of 3D points which are then aligned to the 3D map using Iterative Closest Point. To eliminate drift of the visual odometry the local reconstruction is continuously aligned to the 3D map.

Using this algorithm the resulting translational error is about 0.3 m.

J. Levinson et al. [11] proposed an localisation algorithm using GPS, IMU and LiDAR to generate high-resolution environment maps. Offline relaxation is used to align the maps at intersections. The map is then reduced to the flat road to remove other vehicles from the map. To localise a moving vehicle in the map a particle filter is used for matching the LiDAR measurements to the existing map. Using this method Levinson et al. reaches an real time accuracy within 10 centimeters.

K. Jo et al. [12] presented a solution using Monte Carlo localisation of an autonomous vehicle. The algorithm uses a GPS, a precise digital road map with lane markings and multiple cameras. The cameras are used to match the current road markings to the map and in that way localise the vehicle. The measurements from the multiple cameras are then fused in a particle filter. The mean translational accuracy of this algorithm is around 0.5 meters.

Ford [5] claims to drive autonomously in snowy conditions by creating high resolution digital 3D maps. These maps are created from autonomous vehicles driving in ideal weather conditions equipped with four LiDAR sensors which generates 2.8 millions laser points per second. These LiDAR maps are then used as a baseline for the vehicle driving in snowy conditions to match with objects that are not covered with snow. To handle snowflakes and raindrops Ford uses powerful sensors and an algorithm to detect if the measurement is from a false reading or not.

All of the related works except the one from Ford uses the road and lane markings to localise the vehicle. Ford uses LiDAR to match a prebuilt 3D map for its localisation. The difference between these solutions and the proposed solution in this thesis is that in this thesis the road and lane markings are excluded in the localisation algorithm. The reason for this is for the algorithm to be able to work in complex weather and road conditions for instance snow.

1.6 Thesis Layout

This thesis consists of four main parts excluding the introduction. The first part describes the background theory used for the proposed solution. Next part is the methods used to create the proposed algorithm. The result chapter presents the results from experiments and tests of the algorithm. The thesis is wrapped up with a discussion and conclusion.

2

Background Theory

This chapter contains the background theory that is essential to understand the methods used in this thesis. It describes the way positioning and orientation is done with coordinate transformations. It also describes the different sensors used, the way they work and how the sensor readings are fused together using Bayesian filtering and Kalman filter.

2.1 Position and Orientation

A rigid body can be described in space if having the pose of its coordinate frame with respect to a reference coordinate frame. The pose consists of the position of the coordinate frame, which can be seen as a displacement from the reference frame, and its orientation. The background theory presented in this section is obtained from [13].

2.1.1 Representation in 2D

In Figure 2.1 the reference frame $\{W\}$, the body frame $\{B\}$ and the relative pose of the body frame with respect to the reference frame ${}^{W}\xi_{B}$ are depicted. The relative pose is represented by the displacement (x, y) and the angle θ, that the coordinate frame is rotated. A point $\mathbf{P}$ in 2D space can be described by the bounded vector

$$\mathbf{P} = x\hat{\mathbf{x}} + y\hat{\mathbf{y}} \tag{2.1}$$

where (x, y) are the coordinates in a specific coordinate frame and $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$ are unit vectors parallel to the axes in the same coordinate frame.

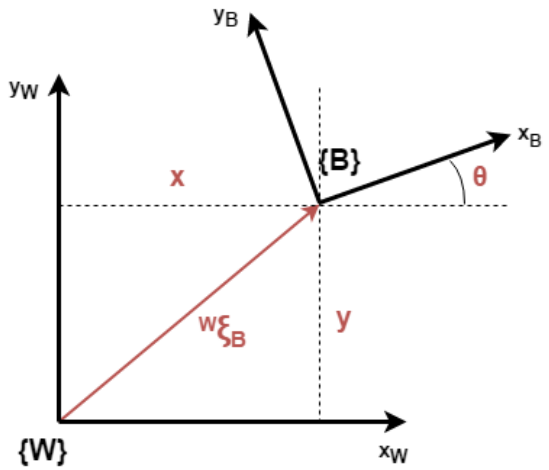


Figure 2.1: The reference frame $\{W\}$, the body frame $\{B\}$ and the relative pose ${}^{W}\xi_{B}$.

2.1.2 Representation in 3D

When representing the pose in three dimensions one more coordinate axis is introduced, the z-axis, which is orthogonal to both the x-axis and y-axis. A point $\mathbf{P}$ in 3D space can then be described by the bounded vector

$$\mathbf{P} = x\hat{\mathbf{x}} + y\hat{\mathbf{y}} + z\hat{\mathbf{z}} \tag{2.2}$$

where (x, y, z) are the coordinates in a specific coordinate frame and $\hat{\mathbf{x}}$, $\hat{\mathbf{y}}$ and $\hat{\mathbf{z}}$ are unit vectors parallel to the axes in the same coordinate frame.

As for the 2D case the relative pose between two different reference frames can be divided into two parts, the translation and the rotation. The translation in 3D is a displacement by the coordinates (x, y, z), which is similar to the 2D case. The orientation part in 3D is however much more complex than the orientation in 2D. When rotating a reference frame in 2D we only consider rotation about the z-axis, but for the 3D case we have to consider the rotation about all three axis.

2.1.2.1 Rotation matrix

According to Euler's rotation theorem any rotation is a sequence of rotations around different coordinate axes. Assuming that two vectors have the same origin, a vector defined with respect to a coordinate frame $\{B\}$ can be rotated to a vector in coordinate frame $\{W\}$ by multiplying with the rotation matrix ${}^{W}\mathbf{R}_{B}$, see Equation (2.3).

$$\begin{bmatrix} {}^{W}x \\ {}^{W}y \\ {}^{W}z \end{bmatrix} = {}^{W}\mathbf{R}_{B} \begin{bmatrix} {}^{B}x \\ {}^{B}y \\ {}^{B}z \end{bmatrix} \tag{2.3}$$

The columns of ${}^{W}\mathbf{R}_{B}$ is formed by the unit vectors of frame $\{B\}$ projected on frame $\{W\}$, hence the rotation matrix is orthonormal and has the following property.

$$\begin{bmatrix} {}^{B}x \\ {}^{B}y \\ {}^{B}z \end{bmatrix} = ({}^{W}\mathbf{R}_{B})^{-1} \begin{bmatrix} {}^{W}x \\ {}^{W}y \\ {}^{W}z \end{bmatrix} = ({}^{W}\mathbf{R}_{B})^{T} \begin{bmatrix} {}^{W}x \\ {}^{W}y \\ {}^{W}z \end{bmatrix} = {}^{B}\mathbf{R}_{W} \begin{bmatrix} {}^{W}x \\ {}^{W}y \\ {}^{W}z \end{bmatrix} \tag{2.4}$$

This means that the inverse is the same as the transpose and the rotated vector with respect to frame $\{W\}$ can be rotated back to frame $\{B\}$ by multiplying with the inverse of the rotation matrix ${}^{W}\mathbf{R}_{B}$. The orthonormal rotation matrices for rotation around the coordinate frame axes by angle θ are given bellow.

$$\mathbf{R}_{\mathbf{x}}(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \tag{2.5a}$$

$$\mathbf{R}_{\mathbf{y}}(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \tag{2.5b}$$

$$\mathbf{R}_{\mathbf{z}}(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.5c}$$

2.1.2.2 Pose matrix

As mentioned before the relative pose consists of a change in position and orientation. The orientation of a coordinate frame can be written as a linear combination of its unit vectors and the orthogonal axes of the reference frame. The orientation can then be combined with the translation to find the relative pose between the coordinate frames. One of the most practical representation of the relative pose is the 4 x 4 homogeneous transformation matrix

$$\begin{bmatrix} {}^{W}x \\ {}^{W}y \\ {}^{W}z \\ 1 \end{bmatrix} = \begin{bmatrix} {}^{W}\mathbf{R}_{B} & \mathbf{t} \\ \mathbf{0}_{1x3} & 1 \end{bmatrix} \begin{bmatrix} {}^{B}x \\ {}^{B}y \\ {}^{B}z \\ 1 \end{bmatrix} \tag{2.6}$$

where ${}^{W}\mathbf{R}_{B}$ is the orthonormal rotation matrix between the coordinate reference frame $\{W\}$ and frame $\{B\}$ and $\mathbf{t}$ is the Cartesian translation vector between the origin of the two coordinate frames.

2.2 Sensors

Sensors in its broadest definition is an electrical component for detecting changes in the environment. There are many types of sensors for different purposes. Sensors relative to this thesis are sensors that can detect motion, measure distances to the environment and estimate the position in space.

2.2.1 Light Detection And Ranging

Light Detection And Ranging (LiDAR) sensors work by sending out laser light beams which bounces of the environment and return to the sensor, the Time of Flight (TOF) is measured and by knowing the speed of light the distance to the environment can be calculated. The advantage over many other ranging sensors is that it can target a wide range of materials. LiDAR uses ultraviolet, visible or near infrared light to image objects.

There are two main types of LiDAR sensors, high-functionality three dimensional scanning LiDAR and low-functionality two dimensional scanning LiDAR. Velodyne's [14] high-functionality sensors uses multiple laser beams to detect the distance to the environment. One example of this type is a Velodyne HDL-64E which has a 360° field of view, a rotation rate of between 5-20 Hz and 64 laser beams vertically spaced with 0.4° between each beam. This gives roughly 2.2 million points per second with an accuracy of less than 2 cm [15]. A scan from this sensor can be seen in Figure 2.2. The low-functionality 2D sensors uses a single beam of light which is rotated in one plane. This gives distance measurements in one plane. A example of this type of sensor is the Sick LMS 151 [16] which has a 270 °field of view.

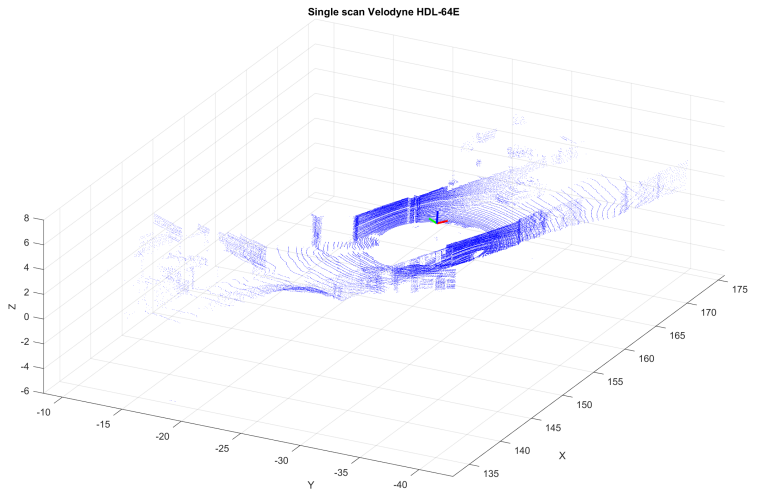


Figure 2.2: 3D map using Velodyne HDL-64E

2.2.2 Inertial Navigation System

An Inertial Navigation System (INS) consists of an Inertial Measurement Unit (IMU) which is used to determine the orientation, heading and motion of the body. The IMU consists of inertial sensors that measures acceleration, rotation and magnetic field. The acceleration of the body is measured using linear accelerometers. An IMU usually contains three accelerometers which are mounted perpendicular to each other to measure accelerations in the X, Y, Z direction of the IMU [17]. Gyroscopes can measure both rotational velocity and rotational angle. By knowing the acceleration

and orientation of a body in space is it possible to calculate the relative position from start and velocity by integrating the sensor values over time.

The INS can be based on many different technologies for instance Micro-Machined Electromechanical System (MEMS) and Micro-Optical Machined Electromechanical System (MOEMS). The most common technology is MEMS since the low price and in recent development the accuracy of the MEMS Gyroscope is as good as 0.01 degrees per hour and the MEMS acceleromter is around 0.00981 m/s^2 [17]. There are four types of errors acting upon an INS: fixed and repeatable terms, temperature induced variations, switch-on to switch-on variations, in-run variations. For instance a bias which is predictable and the same every time the sensors is switched-on and is therefore easy to compensate for. A temperature induced error is also easy to compensate for using a predetermined calibration. A random bias which differs from the switch-on to switch-on but stay the same during the whole run is a example of switch-on to switch-on variations. A random bias which changes during the run is different from sensor to sensor and is difficult to compensate for [17].

2.2.3 Global Navigation Satellite System

Global Navigation Satellite System (GNSS) is a navigation method based on satellite signals sent from orbiting satellites. The signals from the satellites can be interpreted to information about position, velocity and heading. The most common GNSS is NAVSTAR Global Positioning System (GPS) which is a space based navigation system that works in all weather conditions which was developed by the US. Department of Defence for military use, but later promoted for civil use [7]. There are other GNSS financed by other nations, GLONASS is Russias, COMPASS is Chinas and the European Union has a GNSS called Galileo which is still under development. GNSS uses satellites to determine the position and velocity of an object which is done by measuring the time of flight for the broadcasted signals. This technique is dependent on the internal clock of the receiver and the transmitter which are never perfectly synchronised. To overcome this problem four or more satellites are necessary to determine the three point coordinate and the clock error.

The difference between the different GNSS is the number of satellites, GPS have 24 satellites and manages to have global coverage. The GLONASS have 18 satellites which is not enough to have global covreage. The Galileo project will have 27 satellites which will give global coverage and better accuracy at the polar regions [18].

2.2.4 Camera

A camera is a light sensing device for capturing images. The name camera comes from the word Camera Obscura which in Latin means dark chamber. The Camera Obscura is a dark room with a small window which results in that the outside world is projected on the wall opposite to the window. This is the basic function of the simplest camera model called a pinhole camera which is a box with a small hole in

one side and a photographic plate on the opposite wall. The cameras of today are more sophisticated with multiple lenses and image sensors but it still using the basic principle of the pinhole camera. Cameras have since the early 1970s been used in computer vision, which is the visual perception of the outside world for a computer [19]. There are many fields of application for computer vision today for instance object recognition and object measurements.

Cameras have different parameters such as focal length f, image sensor size and number of pixels in the sensor called camera intrinsic parameters. These properties are important for computer vision to be able to determine distances in an image. The image captured by a camera is calibrated using the parameters mentioned earlier by a calibration matrix K, see equation (2.7), where p_x, p_y is the principal point which is the center pixel of the image sensor and q is the size of the pixels.

$$\mathbf{K^{-1}} = \begin{pmatrix} 1 & 0 & -p_x \\ 0 & 1 & -p_y \\ 0 & 0 & f/q \end{pmatrix} \tag{2.7}$$

All image models assume that straight lines in the world are also straight lines in an image which is not the case. With most camera lenses radial distortion occurs and thus need to be compensated for. For compensation of radial distortion in most lenses it is sufficient to use distortion models of low-order polynomials [19], see equation (2.8) where x_c, y_c is the pixel coordinate after perspective division,

$$\begin{aligned} \hat{x}_c &= x_c(1 + k_1 r_c^2 + k_2 r_c^4) \\ \hat{y}_c &= y_c(1 + k_1 r_c^2 + k_2 r_c^4) \end{aligned} \tag{2.8}$$

$\hat{x}_c$, $\hat{x}_c$ is the pixel coordinate after distortion compensation, $r_c = x_c^2 + y_c^2$ and k_1, k_2 is the so called radial distortion parameters. The difference between an image with radial distortion and the same image after compensation for the radial distortion can be seen in Figure 2.3.

(a) Raw image with distortion

(b) Undistorted image after compensation for radial distortion

Figure 2.3: Distorted and undistorted image from Point Grey Ladybug3 omnidirectional camera.

2.3 Camera features

Image features are used to find references in the environment. Image features are small segments of pixels in the image that diverge a lot from their vicinity. The features can for example be blobs, edges and corners whose pixels differs in colour and intensity from their neighbourhood pixels. There are different image feature detectors that can detect different kinds of features. In this section the features that will be evaluated in the thesis are presented.

SIFT features - Scale Invariant Feature Transform. The SIFT feature detector is a scale and rotation invariant algorithm. In the SIFT algorithm Difference of Gaussians (DoG) are utilised to find image features. All the images are smoothed with Gaussian filters in order to reduce the details in the images. The level of blurriness is determined by the standard deviation σ in the Gaussian function. The smoothing process is repeated using different standard deviations in order to compute the DoG.

$$DoG(x, y, \sigma) = I(x, y, k\sigma) - I(x, y, \sigma) \tag{2.9}$$

where $I(x, y, \sigma_i)$ is an image blurred by a Gaussian function with standard deviation σ and $k = 1, 2, ..., n$ where n is an arbitrary number. Local extrema are then found by comparing each pixel with their eight neighbour pixels in the same DoG and the nine neighbour pixels in the closest DoG levels. The local extrema are then examined using a 3D quadratic fitting function to find the SIFT features. [20]

SURF features - Speeded Up Robust Features. The SURF algorithm is inspired by the SIFT algorithm and the algorithms are build up with the same steps, however the different steps are carried out differently. In the SIFT algorithm all the images are smoothed with Gaussian filters while in the SURF algorithm all the images are filtered with square-shaped filters as an approximation of Gaussian smoothing. This makes the SURF algorithm much faster than the SIFT algorithm. [20]

Min Eigen Features - Minimum Eigenvalue Features. The MinEigen algorithm is a method for detecting corner features. The algorithm calculates the eigenvalues of a window in an image. Two small eigenvalues means that the intensity is roughly the same in the image. Two large eigenvalues can represent a corner within the window [21].

MSER - Maximally Stable Extremal Regions. The MSER regions are defined by the extremal property of the intensity at the region and outer boundary. The regions depend on the threshold for the intensity, and all pixels below a threshold in the same area are grouped together to a MSER region, which can be described by a feature descriptor [22]. Parameters can be set to choose max and min size of the regions and the threshold of intensity can be changed.

2.4 Map projection

A map projection is a way to convert latitude and longitude of a sphere into locations on a plane. There are many ways to project the essentially spherical shape of the earth onto a non-spherical paper. There is no way to do this without distortions [23]. It is very important for autonomous vehicles to be able to convert longitude and latitude into meters since most of the other measurements from the other sensors are given in meters. It is also of high importance that this conversion from latitude and longitude is very accurate to get a good estimation of the vehicles position.

2.4.1 Mercator projection

Mercator projection was introduced in 1569 by Gerardus Mercator and is a so called cylindrical projection and must be derived mathematically. The Mercator projection is widely used for navigation purposes since any straight line in a Mercator projection is a constant true bearing which enables easy navigation. However the Mercator Projection is not great for projecting a map over the whole world since further away from the equator the distances appears bigger than they are, for instance Greenland appears bigger than South America [24]. In most web maps a projection called Web Mercator projection or WGS84 is used since Google set the standard 2005 when Google Map was released. The Web Mercator projection is based on the Cylindrical Mercator projection but the big difference is that the Web Mercator projection uses a spherical equation for the forward equations with an Earth radius of 6,378,137.0 m [25]. These maps are made for the user to be able to zoom and pan around the map freely. Since it is a cylindrical projection it has the same issue as the standard Mercator projection with scale distortion at the poles. In the web maps this is done by adjusting the graphic scale bar as the user pans north or south [25].

2.5 Bayesian filtering

Bayesian filtering is used to estimate a state of a time-varying system which is observed with different measurements. Here the Bayesian filtering equations and the Kalman filter equations is going to be derived. The background theory presented in this section is obtained from [26].

A system is often represented as a state space model where the noise is assumed to be additive and Gaussian which means that it is normally distributed, in equation (2.10) x_k is the state at time k and y_k is the measurement at the same time.

$$\begin{aligned} x_k &= f(x_{k-1}, u_k) + q_{k-1} \\ y_k &= h(x_k, u_k) + r_k \end{aligned} \tag{2.10}$$

f and h is referred to as the motion model and the measurement model. u_k is a known input to the system, q_{k-1} is the noise of the motion and r_k is the measurement noise.

For Bayesian filtering the state space is assumed to be a probabilistic state space model which is a sequence of conditional probability distributions, see equation (2.11) where x_k is the state at time k and y_k is the measurement at time k.

$$\begin{aligned} x_k &\sim p(x_k|x_{k-1}, u_k) \\ y_k &\sim p(y_k|x_k) \end{aligned} \tag{2.11}$$

$p(x_k|x_{k-1})$ is called the motion model and is the probability of the state x_k given the previous state x_{k-1}. The $p(y_k|x_k)$ is called the measurement model and is the probability of the measurement y_k given the state x_k. The state space model is Markovian and the states are a Markov chains which means that all states $x_k, x_{k+1}, ...$ given x_{k-1} are independent.

Since the states are Markovian and using the State space model the joint prior distribution for $x_{0:T}$ and the joint likelihood of the measurements $y_{1:T}$ is described in equation (2.12).

$$\begin{aligned} p(x_{0:T}) &= p(x_0) \prod_{k=1}^{T} p(x_k|x_{k-1}, u_k) \\ p(y_{1:T}|x_{0:T}) &= \prod_{k=1}^{T} p(y_k|x_k) \end{aligned} \tag{2.12}$$

By applying Bayes' Rule the posterior distribution for a given time instance T can be calculated using equation (2.13).

$$\begin{aligned} p(x_{0:T}|y_{1:T}, u_{1:T}) &= \frac{p(y_{1:T}|x_{0:T}, u_{1:T})p(x_{0:T})}{p(y_{1:T}, u_{1:T})} \\ &\propto p(y_{1:T}|x_{0:T})p(x_{0:T}, u_{1:T}) \end{aligned} \tag{2.13}$$

The problem of using Bayes' Rule is that the number of computations increases with every time step and will therefor only work for small datasets. To get around this problem the Bayesian filtering equations is used which calculates the marginal posterior distribution or the filtering distribution of the state given the history of measurements. The Bayesian filtering equations are done in two steps, prediction step and update step. The prediction step can be computed using the Chapman-Kolmogorov equation:

$$p(x_k|y_{1:k-1}, u_{1:k}) = \int p(x_k|x_{k-1}, u_{1:k})p(x_{k-1}|y_{1:k-1}, u_{1:k-1})dx_{k-1} \tag{2.14}$$

The update step given the measurement is computed using Bayes' Rule :

$$p(x_k|y_{1:k}) = \frac{1}{Z_k} p(y_k|x_k)p(x_k|y_{1:k-1}, u_{1:k}) \tag{2.15}$$

where Z_k is a normalisation constant and is given by the following equation

$$Z_k = \int p(y_k|x_k)p(x_k|y_{1:k-1}, u_{1:k})dx_k \tag{2.16}$$

2.5.1 Kalman Filter

Kalman Filter is a solution to a special case of the Bayesian filtering equations when the model is Gaussian and linear. The linear state space is in equation (2.17).

$$\begin{aligned} x_k &= A_{k-1}x_{k-1} + B_{k-1}u_{k-1} + q_{k-1} \\ y_k &= H_k x_k + D_k u_k + r_k \end{aligned} \tag{2.17}$$

As mentioned earlier the noise is assumed to be Gaussian additive with zero mean:

$$\begin{aligned} q_k &\sim \mathcal{N}(0, Q) \\ r_k &\sim \mathcal{N}(0, R) \end{aligned} \tag{2.18}$$

The prediction step of the Kalman filter is described in the following equations

$$\hat{x}_{k|k-1} = A_{k-1}x_{k-1|k-1} + B_k u_k \tag{2.19}$$

$$P_{k|k-1} = A_{k-1}P_{k-1|k-1}A_{k-1}^T + Q_{k-1} \tag{2.20}$$

With the prediction calculated, the update step is calculated using the following five equations where K_k is the Kalman gain, v_k is the innovation and S_k is the innovation covariance:

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k v_k \tag{2.21}$$

$$P_{k|k} = P_{k|k-1} - K_k S_k K_k^T \tag{2.22}$$

$$K_k = P_{k|k-1}H_k^T S_k^{-1} \tag{2.23}$$

$$v_k = y_k - H_k \hat{x}_{k|k-1} \tag{2.24}$$

$$S_k = H_k P_{k|k-1} H_k^T + R_k \tag{2.25}$$

2.5.2 Ensemble Kalman Filter

The Kalman Filter assumes that all part of the system is linear which often is not the case. There are different ways to approach the problem, for example Extended Kalman Filter (EKF) where the system is linearised at every iteration of the filter. This is computationally heavy since a Jacobian have to be calculated in each iteration. To avoid this, a method called Ensemble Kalman Filter (EnKF) is used. By sampling the errors of the prediction and update the EnKF gives a statistical approximation to the EKF. The background theory presented in this section is gathered from [27] and [28].

The derivation of the EnKF is quite similar to the Kalman filter, firstly an initial prior is made by generation ensembles of size N given the initial guess x^b and η which is $\mathcal{N}(0, B)$ where B is the covariance for the initial guess:

$$\hat{x}_0 = x^b + \eta \tag{2.26}$$

The prediction step of the EnKF consists of the motion model $f(x_{k-1})$ and the sample time Δt and ζ is randomly drawn from $\mathcal{N}(0, Q)$ where Q is the motion covariance, this is done for each ensemble j:

$$\hat{x}_{j_{k|k-1}} = f(x_{j_{k-1}}) + \sqrt{\Delta t}\zeta \tag{2.27}$$

The update step is done by first calculate the Kalman gain:

$$K = PH(HPH^T + R)^{-1} \tag{2.28}$$

where R is the measurement covariance and PH and HPH^T is approximated by:

$$\begin{aligned} PH &= \frac{1}{N-1}\sum_{j=1}^{N}(x_{j_{k|k-1}} - \overline{x})\left(\sum_{j=1}^{N}(\hat{y}_{j_{k|k-1}} - \overline{y})\right)^T \\ HPH^T &= \frac{1}{N-1}\sum_{j=1}^{N}(\hat{y}_{j_{k|k-1}} - \overline{y})\left(\sum_{j=1}^{N}(\hat{y}_{j_{k|k-1}} - \overline{y})\right)^T \end{aligned} \tag{2.29}$$

Where $\hat{y}_{j_{k|k-1}}$ is the measurement model and $\overline{x}, \overline{y}$ is the mean of all the ensembles. Using the Kalman gain the the update step is calculated using the following equation:

$$\hat{x}_{j_{k|k}} = \hat{x}_{j_{k|k-1}} + K(y_k + \epsilon - \hat{y}_{j_{k|k-1}}) \tag{2.30}$$

where $\epsilon \sim \mathcal{N}(0, R)$ and R is the measurement covariance. The update step, $\hat{x}^a_{k|k}$, gives a estimation for each ensemble and is calculated using the following equation:

$$\hat{x}^a_{k|k} = \frac{1}{N-1}\sum \hat{x}_{j_{k|k}} \tag{2.31}$$

3
Methods

In this chapter the methods used and the algorithm developed are described. The chapter starts with a step by step explanation of the proposed algorithm. Following that, an in depth explanation of each step in the algorithm is given. Finally the experiments to test the algorithm are described.

3.1 Localisation System

The system proposed in this thesis is for localising a vehicle on a known road using LiDAR and camera fused together with IMU and GPS, which is visualised in a flow chart in Figure 3.1. For the algorithm to work a couple of sensors needs to be fitted to a vehicle, a rotating 3D LiDAR with at least 60 beams, a forward looking camera with wide field of view, a GPS and IMU unit. The distance and rotation between the sensors have to be precisely measured and all measurements from the sensors must be transformed into the same coordinate system.

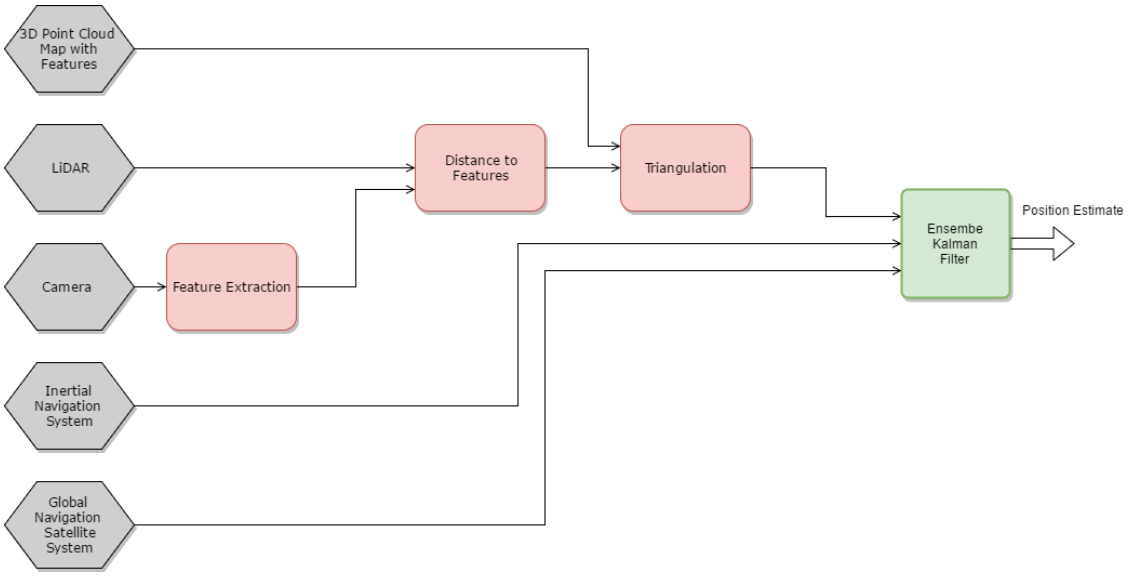


Figure 3.1: Flow chart showing the localisation system from sensors to position estimate

With the sensors on the vehicle and the sensor data translated to the correct frame the first part of the algorithm is to build a ground truth map over the areas of

interest, where the vehicle will be driving. This ground truth map consists of a 3D point cloud from the LiDAR together with interest points, called features, from the camera placed in the correct location in the 3D point cloud. This 3D map with interest points is stored locally in the vehicle.

With the 3D map as a base for the localisation of the vehicle the GPS is used to get a rough location of the vehicle which corresponds to a point in the 3D cloud. To refine the position from the GPS the algorithm developed in this thesis is used. The first step is to take a current LiDAR scan and an image from the camera at the corresponding time to the LiDAR scan, the image is then rectified by the camera calibration parameters and features are extracted. The LiDAR scan is then projected onto the image to get a depth map in the image and the LiDAR measurements closest to the features are stored for later use. The features from the current scan are matched to the features stored in the ground truthed 3D point cloud. By combining the known position of the features in the point cloud and the distances measured in the current scan to the matched features, the vehicle position is triangulated. To get an even more refined position the position estimate from the triangulation is fused together with a motion model of the vehicle and the input from the INS unit.

3.2 Datasets

In this project three different open source data sets are used. One called KITTI which is gathered by Karlsruhe Institute of Technology and Toyota Technological Institute [1], one called Oxford Robotcar Dataset which is gathered by Oxford University [2] and one called Ford Campus Vision gathered by University of Michigan and Ford Motor Company [3].

3.2.1 KITTI dataset

The KITTI data is gathered in a mid-sized town called Karlsruhe, Germany during September 2011 by using a Volkswagen Passat station wagon 3.2 with a sensor platform. The dataset contains both urban and suburbs environments. The sensor setup used in the vehicle is listed below and can be seen in Figure 3.2 [1]:

- 2x PointGray Flea2 grayscale cameras [29] (FL2-14S3M-C), 1.4 Megapixels, 1/2” Sony ICX267 CCD, global shutter
- 2x PointGray Flea2 color cameras [30] (FL2-14S3C-C), 1.4 Megapixels, 1/2” Sony ICX267 CCD, global shutter
- 4x Edmund Optics lenses, 4mm, opening angle ~90°, vertical opening angle of region of interest (ROI) ~35°
- 1x Velodyne HDL-64E rotating 3D laser scanner [15], 10 Hz, 64 beams, 0.09 degree angular resolution, 2 cm distance accuracy, collecting ~1.3 million points/second, field of view: 360°horizontal, 26.8°vertical, range: 120 m
- 1 × OXTS RT3003 inertial and GPS navigation system [31], 6 axis, 100 Hz, L1/L2 RTK, resolution: 0.02m / 0.1°

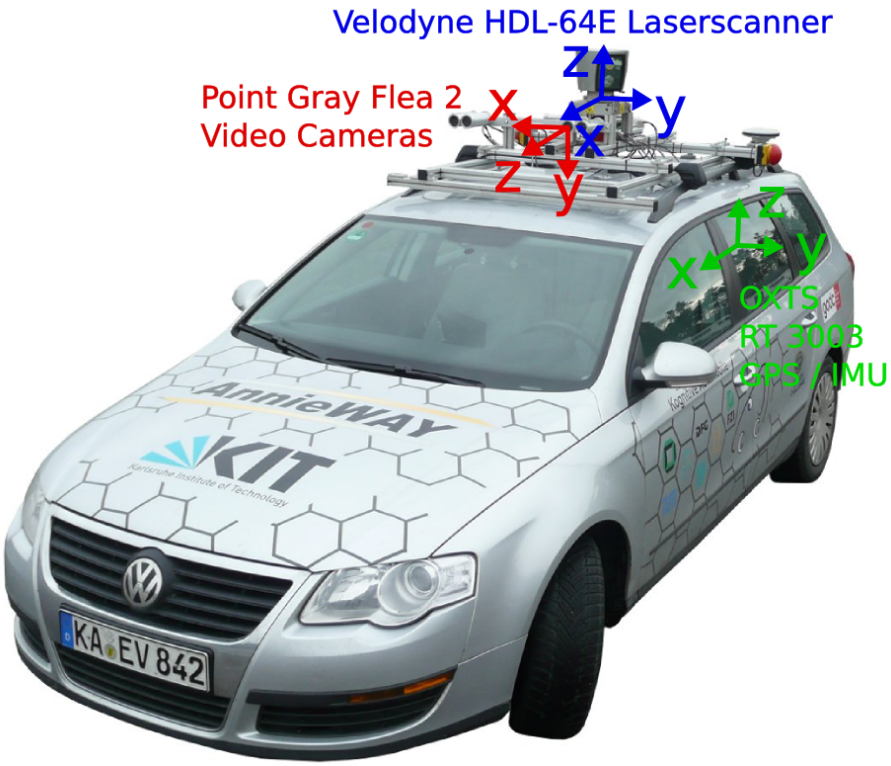


Figure 3.2: Volkswagen Passat with sensors (vehicle for collecting KITTI data). Image from [1].

The incoming data from all sensors is stored by a PC with two six-core Intel XEON X5650 processors and a shock-absorbed RAID 5 hard disk storage with a capacity of 4 Terabytes. The PC runs a 64-bit Ubuntu Linux operating system and real-time database. The placement of the sensors can be seen in Figure 3.3.

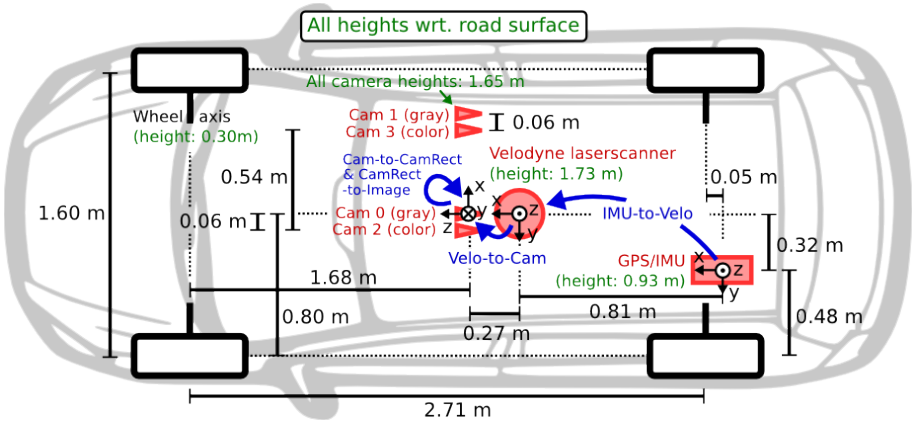


Figure 3.3: Sensor location for the KITTI dataset (Measurements in the figure are approximate, the exact values are included in the development kit). Image from [1].

3.2.2 Oxford Robotcar dataset

The Oxford data is gathered in a period from May 2014 to December 2015 on a route through the center of Oxford, England, twice a week. The vehicle used to collect the data is the Oxford RobotCar platform which is an autonomous-capable Nissan LEAF, see Figure 3.4, with the following sensors attached [2]:

- 1 x Point Grey Bumblebee XB3 [32] (BBX3-13S2C-38) trinocular stereo camera, 1280 × 960 × 3, 16Hz, 1/3" Sony ICX445 CCD, global shutter, 3.8mm lens, 66°HFoV, 12/24cm baseline
- 3 x Point Grey Grasshopper2 [33] (GS2-FW-14S5C-C) monocular camera, 1024 × 1024, 11.1Hz, 2/3" Sony ICX285 CCD, global shutter, 2.67mm fisheye lens (Sunex DSL315B-650-F2.3), 180°HFoV
- 2 x SICK LMS-151 2D LIDAR [16], 270 ◦ FoV, 50Hz, 50m range, 0.5°resolution
- 1 x SICK LD-MRS 3D LIDAR [34], 85°HFoV, 3.2°VFoV, 4 planes, 12.5Hz, 50m range, 0.125°resolution
- 1 x NovAtel SPAN-CPT ALIGN inertial and GPS navigation system [35], 6 axis, 50Hz, GPS/GLONASS, dual antenna

Figure 3.4: Sensor locations and the RobotCar. Image from [2].

The sensor location and orientation can be seen in Figure 3.5. The data was logged using a PC running Ubuntu Linux on two eigh-core Intel Xeon E5-2670 processors, 96GB quad-channel DDR3 memory and a RAID 0 (striped) array of eight 512GB SSDs, for a total capacity of 4 terabytes [2].

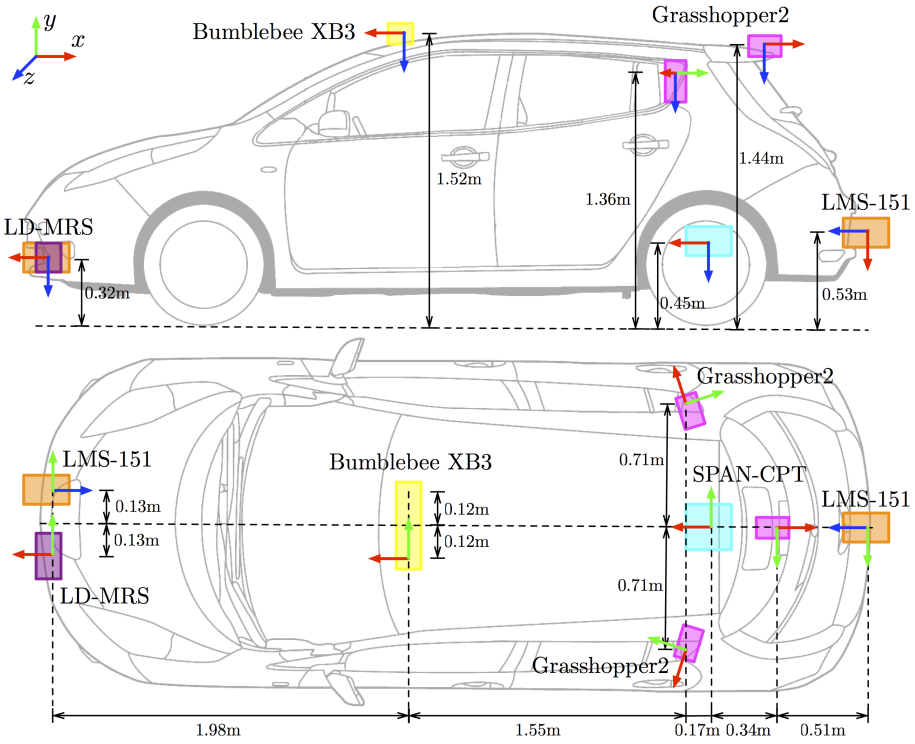


Figure 3.5: Sensor location for the Robotcar Platform (Measurements in the figure are approximate, the exact values are included in the development kit). Image from [2].

3.2.3 Ford dataset

The data from Ford Campus Vision is collected around Ford Research Campus and downtown Dearborn, Michigan, during November-December 2009 using a modified Ford F-250 pickup truck, see Figure 3.6, with the following sensors:

- 1x Velodyne HDL-64E rotating 3D laser scanner [15], 10 Hz, 64 beams, 0.09 degree angular resolution, 2 cm distance accuracy, collecting ~1.3 million points/second, field of view: 360°horizontal, 26.8°vertical, range: 120 m
- 1x Point Grey Ladybug3 omnidirectional camera [36],1600×1200, 12 MP (2 MP x 6 sensors), Global shutter, 1/1.8" Sony ICX274
- 2x Riegl LMS-Q120 LiDAR, 80°FoV, 150m range, 0.2°resolution
- 1x Applanix POS-LV 420 INS with Trimble GPS [37], 6 axis, 1024-count wheel encoder, 100Hz,
- 1x Xsens MTi-G [38], 6 axis, 100Hz

Figure 3.6: Modified Ford F-250 pickup truck with the sensor rig attached. Image from [3].

The location of each sensor can be seen in Figure 3.7. The data was collected using four 2U quad-core processors mounted in the back of the pickup [3].

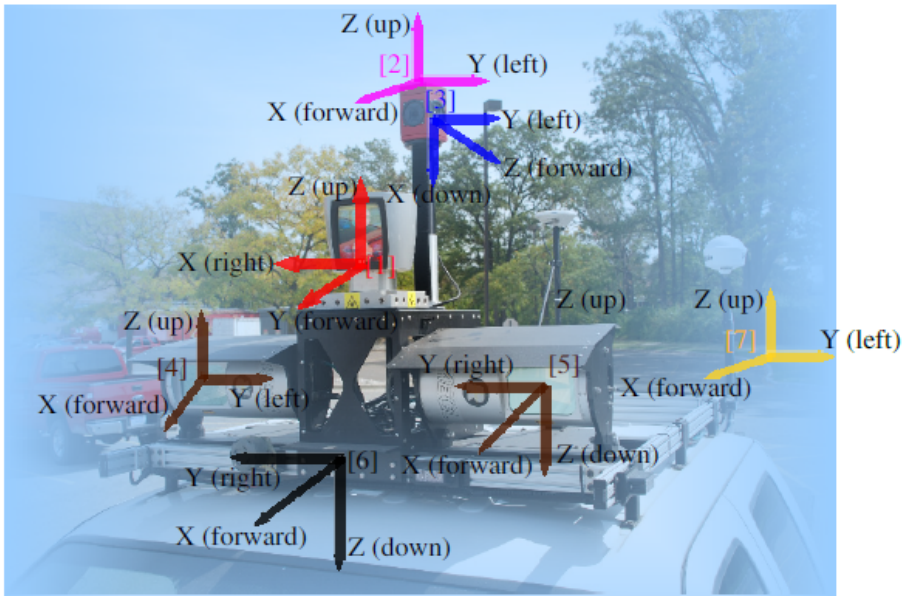


Figure 3.7: Sensor positions and orientation, image from [3],
(1)Velodyne , (2)Laydybug3, (3)Laydybug3 (Camera 5), (4, 5)Riegel,
(6)Body frame (actual location: center of rear axle),
(7)Local frame (Angle between X-axis and East is known)

3.2.4 Preparing data

The KITTI dataset is available in two different versions. One where the raw data is given and one where KITTI have rectified and synced the data. The synced and rectified data is used in this thesis which means that all the images are correlated with a LiDAR scan and a set of IMU values and all the sensors are updated at 10 Hz. In the other two datasets the raw data is only provided which means that the sensors are not synced. In this case the Unix timestamp for each sensor input is given and have to be matched with each other. The Unix time stamp provided is in microseconds and start from 1 Jan 1970.

As can be seen in Figure 3.3, 3.5 and 3.7 the sensors are mounted in different locations and different directions in the vehicle. To be able to get correct readings from all the sensors it is important to transform all values to the same location and orientation in the car. Since the datasets are different, the coordinate frame that is best to transform the sensors to is different for the datasets. For the KITTI dataset it is most convenient to work in the body frame of the vehicle which coincides with the position of the IMU. For the RobotCar dataset it is most convenient to use the frame of the Bumblebee and in the Ford dataset it is most convenient to use the Local Frame. The transformation of the sensors to the correct frame is done by calculating the homogeneous transformation matrix. By multiplying the rotation matrices from equation (2.5) as $\mathbf{R_z}(\alpha)\mathbf{R_y}(\beta)\mathbf{R_x}(\gamma)$ where the angles are yaw, pitch and roll represented as α, β, γ gives the complete rotation matrix $\mathbf{R}$ around the three axis, see equation (3.1).

$$\mathbf{R} = \begin{bmatrix} \cos\gamma\cos\beta & \cos\gamma\sin\beta\sin\alpha - \cos\alpha\cos\gamma & \cos\gamma\sin\beta\cos\alpha + \sin\gamma\sin\alpha \\ \cos\beta\sin\gamma & \cos\gamma\cos\alpha + \sin\gamma\sin\beta\sin\alpha & \cos\alpha\sin\gamma\sin\beta - \cos\gamma\sin\alpha \\ -\sin\beta & \cos\beta\sin\alpha & \cos\beta\cos\alpha \end{bmatrix} \tag{3.1}$$

The translation of the sensors is described by a vector $\mathbf{t}$ of the distance, in X, Y, Z coordinates, between the sensor that is going to be translated and the frame the sensor is being translated into, see equation (3.2).

$$\mathbf{t} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{3.2}$$

The rotation matrix and the translation vector is then combined into the homogeneous transformation matrix $\mathbf{T}$ by making the translation vector into homogeneous coordinate, see equation (3.3).

$$\mathbf{T} = \begin{bmatrix} \cos\gamma\cos\beta & \cos\gamma\sin\beta\sin\alpha - \cos\alpha\cos\gamma & \cos\gamma\sin\beta\cos\alpha + \sin\gamma\sin\alpha & X \\ \cos\beta\sin\gamma & \cos\gamma\cos\alpha + \sin\gamma\sin\beta\sin\alpha & \cos\alpha\sin\gamma\sin\beta - \cos\gamma\sin\alpha & Y \\ -\sin\beta & \cos\beta\sin\alpha & \cos\beta\cos\alpha & Z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

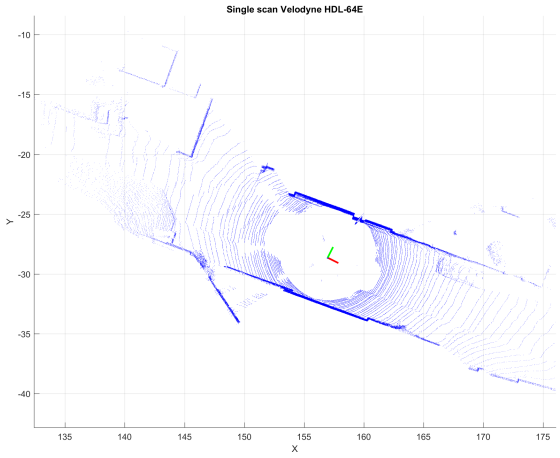
3.3 Localisation

The localisation of the vehicle is done in steps, to first get a rough position which then is refined. The first step is to use the GPS to get an absolute position in longitude and latitude with an uncertainty radius. The area of interest is segmented into smaller parts to gain better accuracy and the correct segment where the vehicle is at the moment is selected by the position measurement from the GPS. All the roads in the area of interest are previously ground truthed with LiDAR to create a 3D map and simultaneously the 3D map is mapped to a real wold map. To refine the position from the GPS the 3D map is used as a reference which is matched using camera and LiDAR. From this measurement the position is triangulated in order to refine the position of the vehicle. The position from the GPS and the triangulated position is fused together with motion measurements of the vehicle from the IMU and the motion model using the Ensemble Kalman Filter (EnKF).

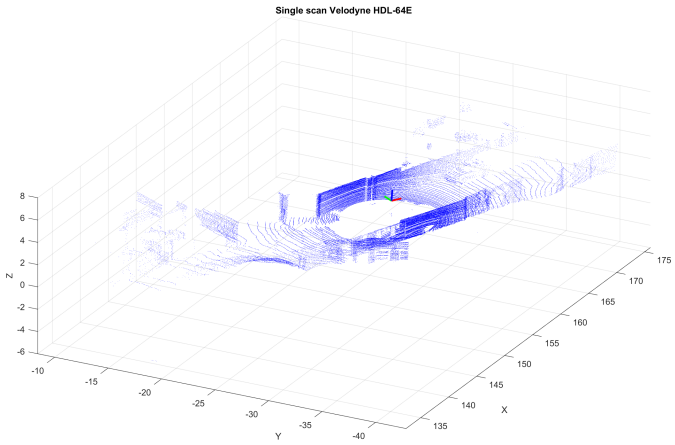
3.3.1 Map

Real world maps are available in different qualities in databases for example OpenStreetMap or in Sweden, Nationella Vägdatabasen (NVDB). These maps are printed in Mercator projection and are measured in degrees and minutes in longitudinal and latitudinal direction which is the same as the coordinates given by the GPS. This map is used to match the rough position of the vehicle in the world to the position given by the GPS.

The reference 3D map is built in two ways depending on the type of LiDAR used. In the KITTI and Ford dataset the Velodyne 3D LiDAR is used which gives a point cloud with roughly 200.000 points per 360° revolution around the vehicle, see Figure 3.8a and 3.8b. In the RobotCar dataset a SICK LMS 151 is used, which is a 2D LiDAR that gives points in one plane that is perpendicular to the front of the vehicle, see Figure 3.9a and 3.9b.

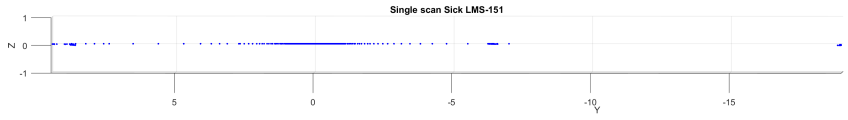


(a) Top down view using Velodyne HDL-64E

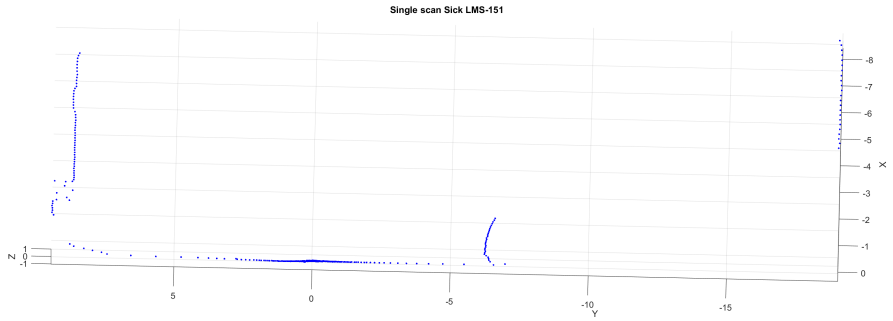


(b) Side view using Velodyne HDL-64E

Figure 3.8: Single LiDAR scan with Velodyne HDL-64E 3D LiDAR



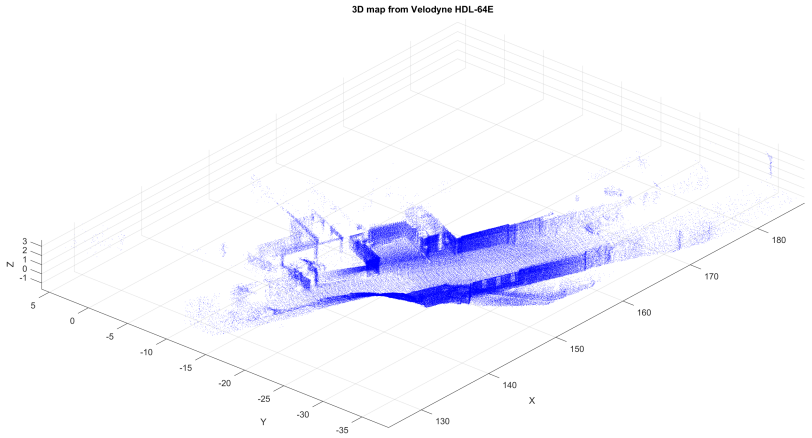
(a) Top down view using Sick LMS 151



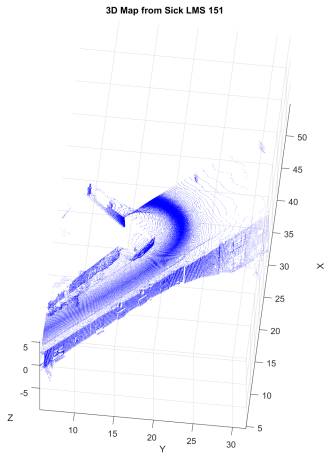
(b) Side view using Sick LMS 151

Figure 3.9: Single LiDAR scan with Sick LMS 151 2D LiDAR

To build the point cloud for the KITTI and Ford data, each scan from a ground truth run is combined with the exact position of the vehicle at the corresponding time. This gives an extremely dens point cloud(about 1 million points per second), see Figure 3.10a. For the RobotCar dataset the approach is similar to the other datasets but the resulting cloud is different in the way that is it much sparser and have bigger shadow areas, see Figure 3.10b.



(a) 3D map using Velodyne HDL-64E



(b) 3D map using Sick LMS 151

Figure 3.10: 3D maps created with Velodyne HDL-64E 3D and Sick LMS 151 2D LiDAR

The LiDAR gives its measurements in meters for x, y, z coordinates and therefore the points in the 3D map have distances measured in meters. To combine the 3D map to the real world map it has to be transformed into meters from the Mercator projection. Since the world is not a perfect sphere it is modeled as en ellipsoid to get the correct ratio of objects anywhere in the world. The earth is a elliptical shape which is described by the major and minor axis (R, r) which can be seen in equation (3.5) and by knowing the longitudinal position λ, the x position m_x in meters can be derived from equation (3.4).

$$m_x = R \cdot \lambda \cdot \frac{\pi}{180} \tag{3.4}$$

where the values of R and r are

$$\begin{aligned} R &= 6378137 \\ r &= 6356752.3142 \end{aligned} \tag{3.5}$$

The y position m_y is more complicated to calculate since the radius of the ellipsoid changes with the longitude φ and is given by equation (3.6). Eccentricity $\mathcal{E}$ is a measure of the ratio between the semi-major axis and semi-minor axis, see equation (3.7). For a ellipse the eccentricity is in the range of $[0, 1]$ where zero is a special case which represents a circle.

$$m_y = R \ln \left[\tan \left(\frac{\pi}{4} + \frac{\varphi}{2} \right) \left(\frac{1 - \mathcal{E} \sin \varphi}{1 + \mathcal{E} \sin \varphi} \right)^{\frac{\mathcal{E}}{2}} \right] \tag{3.6}$$

$$\mathcal{E} = \sqrt{1 - \left(\frac{r}{R} \right)^2} \tag{3.7}$$

The Mercator projection in meter in latitude starts at the equator and gives a positive measurement to the north and a negative to the south. In the longitudinal direction the scale starts at zero degrees which is at time zone zero in Greenwich Mean time.

3.3.2 Camera features

There are several techniques to distinguish points and features in an image as discussed earlier. The one used in this project is called Maximally Stable Extremal Regions (MSER) which captures areas of intensity in an image. This is done using the `Matlab` function called `detectMSERFeatures` [39] which returns MSER object. The MSER are used in this thesis due to their stability in different lighting conditions. The MSER descriptors and points in the image are extracted using the `Matlab` function called `extractFeatures` [40] which takes the image and the MSER as input and returns the feature descriptors and the position of the features in the image.

The LiDAR is used to get the distance to the features detected by the MSER algorithm. To know which of the LiDAR measurement corresponds to the features in the image the LiDAR measurements are projected onto the image, which is done by multiplying the LiDAR X, Y, Z coordinates, which are transformed into the camera frame, with the camera matrix $\mathbf{K}$ which converts the points into pixel coordinates in x, y, z as follows:

$$pixels = \mathbf{K} * \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{3.8}$$

Since images are two dimensional the z axis of the pixel coordinates is eliminated by dividing x and y with z in equation (3.9) to get the pixel coordinates for each LiDAR point u, v.

$$u = \frac{x}{z}$$
$$v = \frac{y}{z} \tag{3.9}$$

These LiDAR pixel coordinates are then matched to the position of the features by finding the closest LiDAR pixel to the features which have to be within 5 pixels from the feature position. The features are then positioned at the closest real X, Y, Z LiDAR coordinate in the 3D point cloud.

3.3.3 Positioning in the map

With the ground truthed map over the area of interest the vehicle is able to position itself using the camera and LiDAR. The camera and the LiDAR takes a snapshot at the current moment and features are extracted from the camera image as described in the previous section. The extracted features from the image are matched to all features in the 3D point cloud. By knowing the positions of the features in the point cloud and the distances from the vehicle to the features the location of the vehicle is triangulated. The matching of the features is done by a `Matlab` function called `matchFeatures` which takes two set of features and match them with a set threshold returning the index of the matched features in the two sets.

3.3.4 Triangulation of estimated position

Each of the features from the camera have a known position in the 3D map. The LiDAR gives each measurement in XYZ coordinates in the vehicles frame. To estimate the position in the map frame the point from the feature points have to be transformed. The transformation is done by rotating the car frame into the map frame and then calculate the position from the distance measurements. The normal vectors of a coordinate system can be described by:

$$\mathbf{e_x} = \begin{bmatrix} a \\ b \end{bmatrix} \tag{3.10}$$

$$\mathbf{e_y} = \begin{bmatrix} -b \\ a \end{bmatrix} \tag{3.11}$$

The Equation for the position of the vehicle, X and Y can be seen in Equation (3.12), where Δx and Δy are the measurement in X and Y coordinates from the LiDAR, x_1 and y_1 is the position in the map where the LiDAR measured to.

$$(X, Y) + (\begin{bmatrix} \Delta x & \Delta y \end{bmatrix} \mathbf{e_x}, \begin{bmatrix} \Delta x & \Delta y \end{bmatrix} \mathbf{e_y}) = (x_1, y_1) \tag{3.12}$$

Since there are four unknown variables in Equation (3.12) two measurements from the LiDAR is needed to calculate the position of the vehicle which is described in Equation (3.13).

$$\begin{cases} y_1 = \Delta y_1 a - \Delta x_1 b + y \\ y_2 = \Delta y_2 a - \Delta x_2 b + y \\ x_1 = \Delta x_1 a + \Delta y_1 b + x \\ x_2 = \Delta x_2 + \Delta y_2 b + x \end{cases} \tag{3.13}$$

The equation system above can be rewritten into state space form which can be seen in Equation (3.14) to Equation (3.16).

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & \Delta x_1 & \Delta y_1 \\ 0 & 1 & \Delta y_1 & -\Delta x_1 \\ 1 & 0 & \Delta x_2 & \Delta y_2 \\ 0 & 1 & \Delta y_2 & -\Delta x_2 \end{bmatrix} \tag{3.14}$$

$$\mathbf{X} = \begin{bmatrix} x \\ y \\ a \\ b \end{bmatrix} \tag{3.15}$$

$$\mathbf{B} = \begin{bmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \end{bmatrix} \tag{3.16}$$

Equation (3.17) solves the equation system which gives the position of the car. An illustration of the triangulation can be seen in Figure 3.11

$$\mathbf{X} = \mathbf{A}^{-1}\mathbf{B}. \tag{3.17}$$

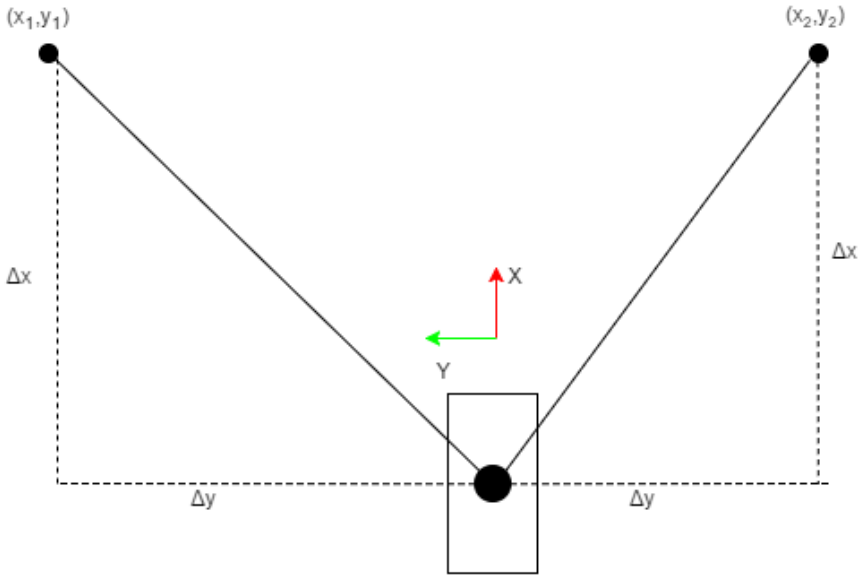


Figure 3.11: Triangulation of the vehicles pose using two feature points

To get a good estimate of the position and remove any false readings, all of the matched LiDAR measurements are randomly selected in pairs and calculated to a position estimate. These positions are then sorted in x and y respectively from low to high. Coordinates which are not within a set distance from the previous set position are removed. The remaining points are then clustered together within the closest meter from the previous position. If this cluster contains less than five measurements, more measurements from the sorted list are taken into the cluster if they are within 2 meters from the previous position. After the cluster is set the mean value of the cluster is calculated which gives the estimated position.

3.4 Sensor Fusion

Ensemble Kalman Filter is used to fuse the triangulated position estimates from LiDAR and camera, described in Section 3.3.4, with GNSS and INS data. The three datasets are collected with different sensor setups and therefore the sensor data process is not the same for all datasets. In this section the motion and measurement models and the sensor data process for each datasets are presented.

3.4.1 KITTI dataset

The states in the Kalman filter can be chosen in many different ways resulting in varying complexity and performance. In order to find a good balance between model complexity and performance, the following states are chosen for the KITTI dataset.

$$\mathbf{x} = [X \; Y \; v_x \; v_y \; a_x \; a_y \; \psi \; \dot{\psi}]^T \tag{3.18}$$

where X and Y are the vehicle positions in the local coordinate frame $\{L\}$, see Figure 3.12. v_x, v_y, a_x and a_y are the velocities and accelerations in the vehicle body coordinate frame $\{B\}$. ψ is the vehicle heading (yaw) angle and $\dot{\psi}$ the yaw rate.

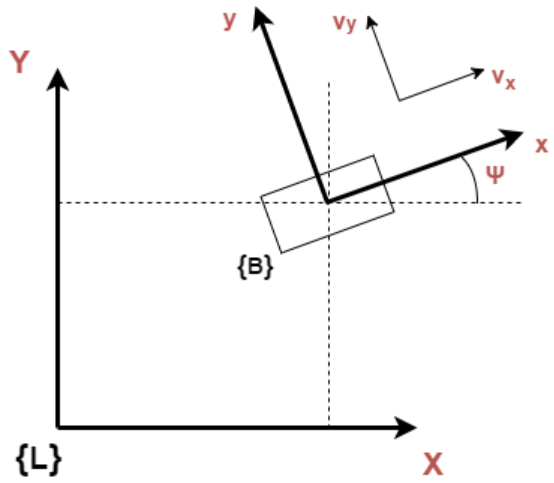


Figure 3.12: The vehicle and its body coordinate frame $\{B\}$ depict in the local coordinate frame $\{L\}$ for KITTI dataset.

The velocities and accelerations from the INS are measured in the vehicle body frame, see Figure 3.12, resulting in the motion model in equation (3.19) for the kinematic relations between the states [41]. The motion model describes the relation between the estimated states in current time step and the estimated states in previous time step.

$$\mathbf{x}_k = \begin{bmatrix} X_{k-1} + \cos\psi(v_{x_{k-1}}T_s + a_{x_{k-1}}T_s^2/2) - \sin\psi(v_{x_{k-1}}\dot{\psi}_{k-1}T_s^2/2 + 2a_{x_{k-1}}\dot{\psi}_{x_{k-1}}T_s^3/6) \\ Y_{k-1} + \sin\psi(v_{x_{k-1}}T_s + a_{x_{k-1}}T_s^2/2) + \cos\psi(v_{x_{k-1}}\dot{\psi}_{k-1}T_s^2/2 + 2a_{x_{k-1}}\dot{\psi}_{x_{k-1}}T_s^3/6) \\ v_{x_{k-1}} + a_{x_{k-1}}T_s \\ v_{y_{k-1}} + a_{y_{k-1}}T_s \\ a_{x_{k-1}} + j_{x_{k-1}}T_s \\ a_{y_{k-1}} + j_{y_{k-1}}T_s \\ \psi_{k-1} + sign(v_{x_{k-1}})\dot{\psi}_{k-1}T_s \\ \dot{\psi}_{k-1} + \ddot{\psi}_{k-1}T_s \end{bmatrix} \tag{3.19}$$

The subscript k defines current time step and $k-1$ defines previous time step. $j_{x_{k-1}}$ and $j_{y_{k-1}}$ are the jerk and $\ddot{\psi}_{k-1}$ is the yaw acceleration. These states are noisy states and are included in the motion noise q_{k-1}, see equation (2.10). As can be seen in the motion model the motion equations for three states are nonlinear. Therefore using EnKF is prefered due to the computional time advantages and to get a as short as possible computational time for the complete localisation system. The measurement model describes the relation between the sensor measurements and the states. All the states can be directly measured by the GNSS and INS except the accelerations. Therefore the measurement vector is defined as

$$\mathbf{z} = [X\ Y\ v_x\ v_y\ acc_x\ acc_y\ \psi\ \dot{\psi}]^T \tag{3.20}$$

where the states X, Y v_x, v_y, ψ and $\dot{\psi}$ are measured by the GNSS and INS. Note that acc_x and acc_y are the accelerometer measurements and not the acceleration states. By assuming that the INS is located at the center of mass of the vehicle and the fact that velocity in the z-axis is not included in the state vector, acc_x and acc_y can be formulated as functions of the states as in the following measurement model. [42]

$$\mathbf{z}_k = \begin{bmatrix} X_k \\ Y_k \\ v_{x_k} \\ v_{y_k} \\ a_{x_k} - \dot{\psi}_k v_{y_k} \\ a_{y_k} + \dot{\psi}_k v_{x_k} \\ \psi_k \\ \dot{\psi}_k \end{bmatrix} + r_k \tag{3.21}$$

where r_k is the measurement noise, see equation (2.10).

3.4.2 Ford dataset

As in the KITTI dataset the GNSS and INS used in the Ford dataset can measure all the states directly except the accelerations and consequently it is beneficial to choose the same state vector which is given in equation (3.18) above. The difference is that the measurements of the velocities and accelerations are transformed into the local coordinate frame $\{L\}$, see Figure 3.13, and are not in the vehicle body frame $\{B\}$ as in the KITTI dataset, resulting in the motion model given in equation (3.22).

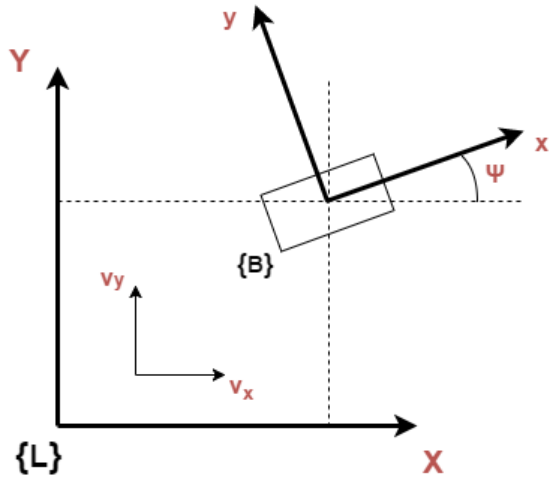


Figure 3.13: The vehicle and its body coordinate frame $\{B\}$ depict in the local coordinate frame $\{L\}$ for Ford and Oxford dataset.

$$\mathbf{x}_k = \begin{bmatrix} X_{k-1} + v_{x_{k-1}} T_s \\ Y_{k-1} + v_{y_{k-1}} T_s \\ v_{x_{k-1}} + a_{x_{k-1}} T_s \\ v_{y_{k-1}} + a_{y_{k-1}} T_s \\ a_{x_{k-1}} + j_{x_{k-1}} T_s \\ a_{y_{k-1}} + j_{y_{k-1}} T_s \\ \psi_{k-1} + sign(v_{x_{k-1}}) \dot{\psi}_{k-1} T_s \\ \dot{\psi}_{k-1} + \ddot{\psi}_{k-1} T_s \end{bmatrix} \tag{3.22}$$

The subscript k defines current time step and $k-1$ defines previous time step. $j_{x_{k-1}}$ and $j_{y_{k-1}}$ are the jerk and $\ddot{\psi}_{k-1}$ is the yaw acceleration. These states are noisy states and are included in the motion noise q_{k-1}, see equation (2.10). Due to that the measurements of the velocities and accelerations are given in the local coordinate frame there are none nonlinearities in the motion model except for one state. In real world scenarios the velocities and accelerations are transformed and are given in the body frame which would yield more nonlinearities, as for the KITTI datasets. Therefore the EnKF is used also for this dataset. The fact that the velocity and acceleration measurements are given in the local frame does not affect the measurement model, hence the measurement vector and measurement model for

the Ford dataset are formulated in the same way as for the KITTI dataset and are given in equation (3.20) and equation (3.21) respectively.

3.4.3 Oxford Robotcar dataset

The INS that is used in the Oxford dataset does not measure acceleration and yaw acceleration. Therefore the state vector is reduced to five states by removing the acceleration states and the yaw rate state, which gives the following state vector.

$$\mathbf{x} = [X \ Y \ v_x \ v_y \ \psi]^T \tag{3.23}$$

The velocities are in the same way as for the Ford dataset transformed into the local frame, see Figure 3.13. Hence the motion model for the Ford dataset can be used for the Oxford dataset but reduced.

$$\mathbf{x}_k = \begin{bmatrix} X_{k-1} + v_{x_{k-1}} T_s \\ Y_{k-1} + v_{y_{k-1}} T_s \\ v_{x_{k-1}} + a_{x_{k-1}} T_s \\ v_{y_{k-1}} + a_{y_{k-1}} T_s \\ \psi_{k-1} + sign(v_{x_{k-1}}) \dot{\psi}_{k-1} T_s \\ \dot{\psi}_{k-1} + \ddot{\psi}_{k-1} T_s \end{bmatrix} \tag{3.24}$$

where $a_{x_{k-1}}$ and $a_{y_{k-1}}$ are the acceleration states considered as noise states and are included in the motion noise q_{k-1}, see equation (2.10). As for the Ford dataset there is only one nonlinearity in this motion model. But as mentioned before the transformation of the velocities are not done in real world scenarios and therefore the EnKF is used for this dataset as well. All the states can be directly measured by the GNSS and INS and the measurement vector and measurement model can be formulated as follows.

$$\mathbf{z} = [X \ Y \ v_x \ v_y \ \psi]^T \tag{3.25}$$

$$\mathbf{z}_k = \begin{bmatrix} X_k \\ Y_k \\ v_{x_k} \\ v_{y_k} \\ \psi_k \end{bmatrix} + r_k \tag{3.26}$$

where r_k is the measurement noise, see equation (2.10).

3.5 Evaluation, Testing and Experiments

To test the accuracy and performance of the proposed algorithm and verify the functionality a couple of tests is derived. Firstly the algorithm is tested for the translational accuracy by using 50-100 meters of data from each dataset and comparing the algorithm's estimated position with the true position of the vehicle. The RobotCar dataset is the only one of the three datasets which contains any snow

data and is therefore going to be used for testing the performance and translational accuracy in snowy environment.

3.5.1 Performance

Performance tests on the translational accuracy is going to be tested for all three datasets over a distance of about 50-100 meters depending on the usable data in each dataset. The first test is carried out with the proposed algorithm, GPS and IMU fused together in the EnKF. The second test is conducted with GPS only to get a comparison between GPS and the proposed system. The third test is conducted using the proposed algorithm, IMU and GPS but with loss of GPS in one part of the trajectory.

In the KITTI dataset there is only one logged dataset of each road which means that ground truthing is done on the same dataset as the algorithm runs on. To make the data set as realistic as possible, as if it was one ground truth dataset and another dataset that is used for testing, the ground truth map is built in one version for each frame in the algorithm. In each of these versions of the ground truth map the current frame in the algorithm is removed from the map and the image features from that frame is also removed. In the Ford Campus Vision dataset the logged data is from one run where the vehicle travels the same road for about 50 meters. Therefore this data set is split into two, the first part is used to build the ground truth map and the second part is used for testing the algorithm. In the RobotCar dataset the same road is logged twice a week during one year. As mentioned earlier this is the only data which have both ideal and snowy condition on the same road. A dataset with ideal weather is used to build the ground truth map and the dataset with snowy weather is used for testing the algorithm.

3.5.2 Speed versus Performance

One of the goals with the thesis is to be able to use the system in real time. Therefore experiments on performance compared to computational speed is going to be carried out. There are three main factors that affects the speed and performance of the system, how often the camera LiDAR algorithm is running, number of ensembles in the EnKF and different types of image features. The time of each iteration is going to be calculated by dividing the total time of a run with the number of estimates in that run. The tests are performed in `Matlab` which is quite slow since it is not a compiled programming language. A comparison of computational speed in `Matlab` and C++ was presented by T. Andrews [43] where Andrews claims that C++ is 500 times faster than `Matlab` for all types of `Matlab` code.

In the EnKF the different inputs does not have to have the same frequency, which provides a test of the performance versus speed of how often the camera and LiDAR algorithm are used. The number of ensembles in the EnKF will be tested in the difference of performance versus speed. As mentioned earlier there are many different

image features and the performance compared to the computational speed of SURF, MSER and Min Eigen Values is going to be evaluated.

4

Results

In this chapter the results from the tests and the experiments described in Section 3.5 is going to be presented. The chapter is divided into three sections. In the first section the test environment for all datasets will be presented. The second section the result from the performance tests are presented. And the last chapter contains the results from the performance versus speed tests for all datasets.

4.1 Test environment

As mentioned in Section 3.5 the system is tested on three datasets which are in three different locations. The parts of the data selected in each dataset is to test different aspects of the system. All the tests and experiments is done using the same computer with the following specifications:

Make	Dell
Model	XPS13 infinity
Operating system	Windows 10 Home 64-bit
Processor	I7-550U, 2.4 GHz, Dual Core
RAM	8 GB, DDR3L
Graphic processor	Intel HD Graphics 5500

Table 4.1: Specifications of computer used for testing and experiments

The KITTI dataset is logged in Karlsruhe, Germany. The data used in the testing from KITTI is in a single lane road in a residential area where there are buildings and walls on each side of the vehicle, an image from the middle of the test sequence can be seen in Figure 4.1a. The trajectory of the test sequence in a real world map can be seen in Figure 4.1b.

(a) Image showing the environment in the KITTI dataset

(b) Trajectory in real world map. Location: Wolfweg, Karlsruhe, Germany

Figure 4.1: Image from the camera and trajectory for KITTI data

In Figure 4.2 the same trajectory is plotted but now in the 3D map of the environment.

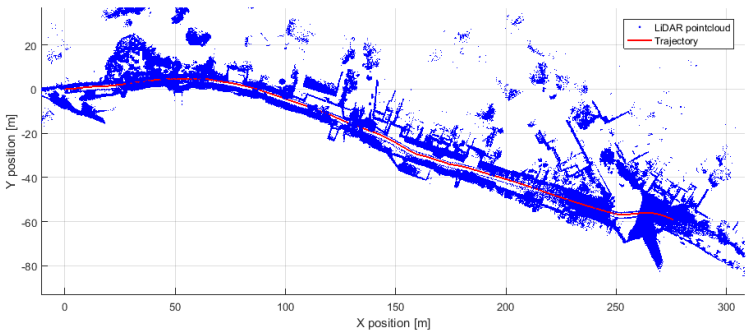


Figure 4.2: Top-down view of trajectory in the 3D map for KITTI dataset.

The RobotCar dataset is logged during a full year on the same route. The data selected from this data set is to test the performance of the algorithm in snowy conditions which only is available on a very limited part of the road which is in a parking area in the center of Oxford. The amount of snow coverage on this piece of road is estimated to be 3 cm which can be seen in Figure 4.3a and the trajectory of the vehicle in a real world map can be seen in Figure 4.3b. In Figure 4.4 the trajectory is plotted in the 3D map.

(a) Image showing the environment in the RobotCar dataset

(b) Trajectory in real world map. Location: Keble College Acland Site, Oxford, England

Figure 4.3: Camera image and trajectory for RobotCar data

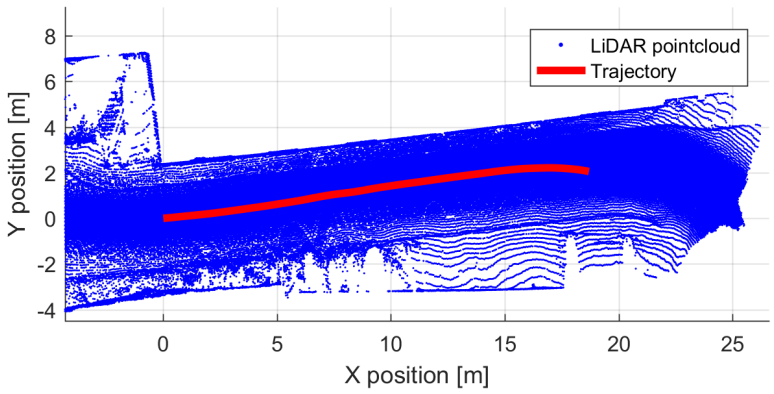
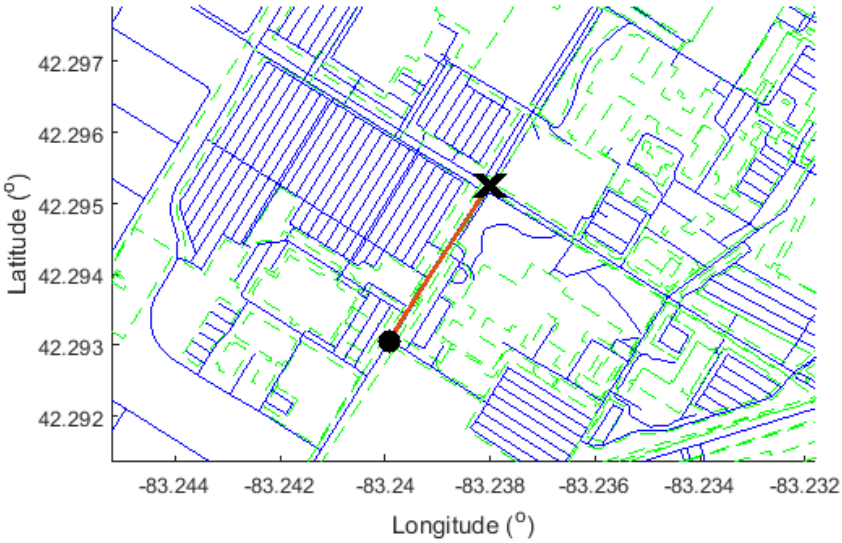


Figure 4.4: Top-down view of trajectory in the 3D map for The RobotCar dataset.

The Ford Campus Vision dataset is logged around the Ford Research Campus in Dearborn, Michigan which is a dual carriageway in an open industrial estate which can be seen in Figure 4.5a and the trajectory in a real world map can be seen in Figure 4.5b. From this dataset the algorithm is tested on how it performs when there is a long distance to the closest points of interest.

(a) Image showing the environment in the Ford Campus Vision dataset



(b) Test trajectory in real world map. Location: Village RD, Dearborn, Michigan, USA

Figure 4.5: Camera image and trajectory for Ford Campus Vision data

In Figure 4.6 the trajectory is plotted in the 3D map of the environment.

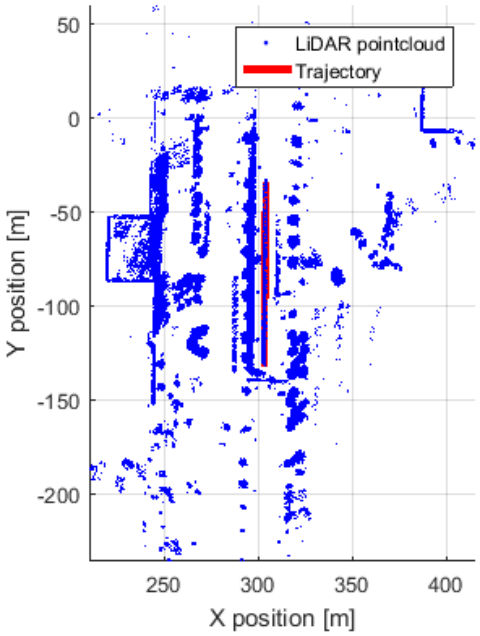


Figure 4.6: Top-down view of trajectory in the 3D map for Ford Campus Vision dataset.

4.2 Performance

As mentioned in Section 3.5.1 three different tests are going to be performed, the first using the complete system with the algorithm, IMU and GPS, the second using only GPS for localisation and the third using the complete system and simulated loss of GPS signal. The result for the three tests is going to presented for each dataset consecutively.

4.2.1 KITTI dataset

The first test is using the LiDAR and Camera algorithm, IMU and GPS. Figure 4.7 shows the true positions and the estimated positions from the localisation system for the entire run. It can be seen that the estimated positions, blue line, follows the true positions, red line, with some small deviations.

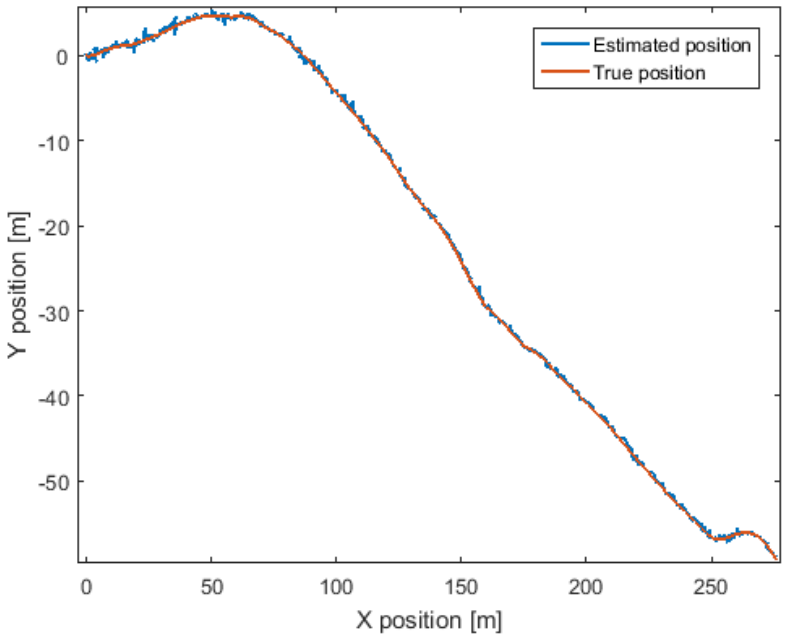


Figure 4.7: True position and estimated position for KITTI dataset

More details about the small deviations can be seen in Figure 4.8 where the absolute mean translational error for the complete algorithm are shown. The mean error is less than 0.3 m for all the parts of the trajectory except from one peak in x-coordinate.

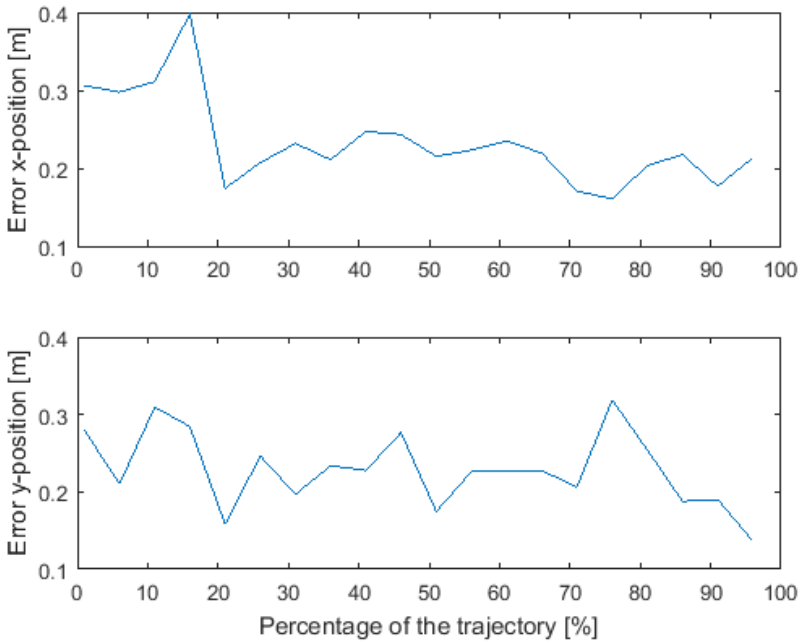


Figure 4.8: Absolute mean transalational error in x and y coordinates for the complete system for the KITTI dataset

The second test uses only GPS for the localisation on the same trajectory as test one. The result is visualised in Figure 4.9 showing a graph of the translational mean error distributed along the trajectory in x and y direction. The error in this test is around 1 m including some peaks.

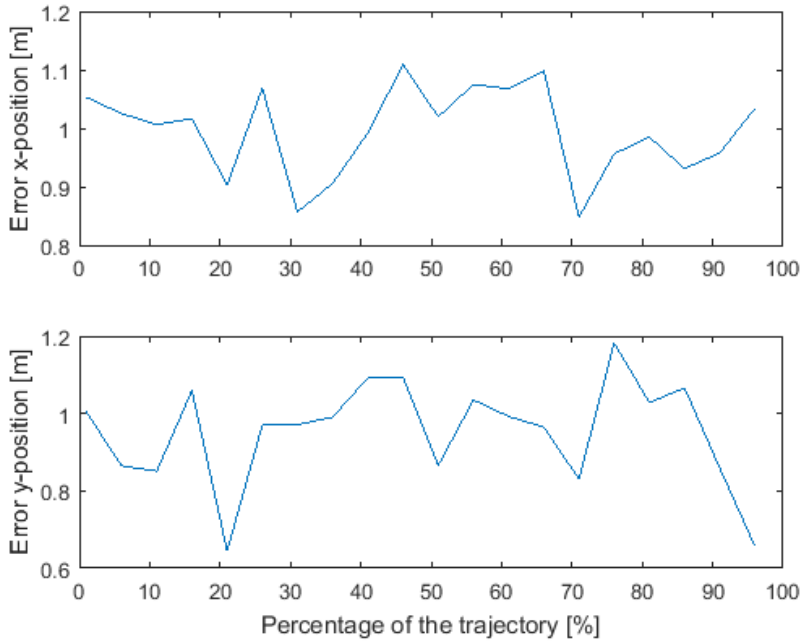


Figure 4.9: Absolute mean translational error in x and y coordinates using only GPS for the KITTI dataset

The third test uses the full system but loss of GPS signal is simulated between 40% and 55% of the trajectory. The result of this can be seen in Figure 4.10 and the error is quite the same as the result without loss of GPS signal. The mean error is around 0.3 m with some peaks.

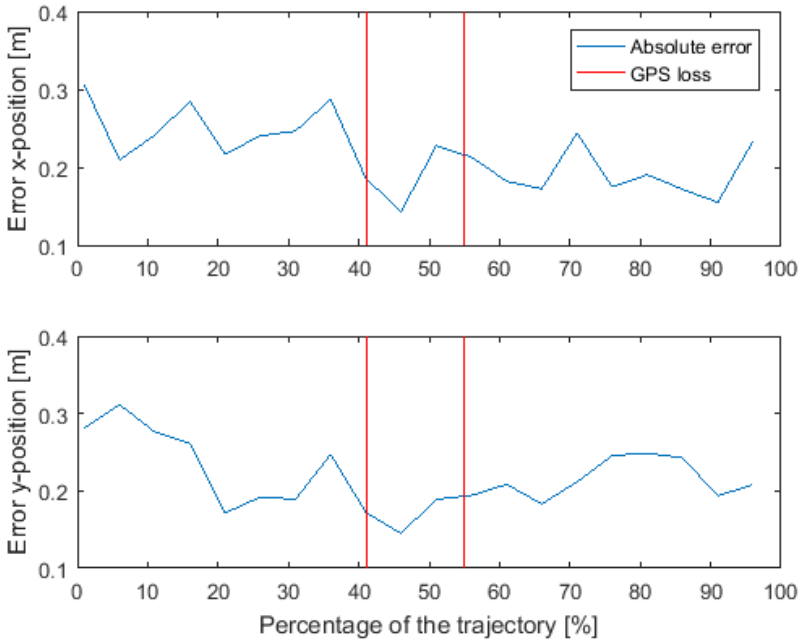


Figure 4.10: Absolute mean translational error in x and y coordinates for the complete system with loss of GPS signal between 40% and 55% of the trajectory for the KITTI dataset

4.2.2 RobotCar dataset

The first test is using the localisation system and the true positions together with the estimated positions are shown in Figure 4.7. The estimated positions follows the true positions with deviations.

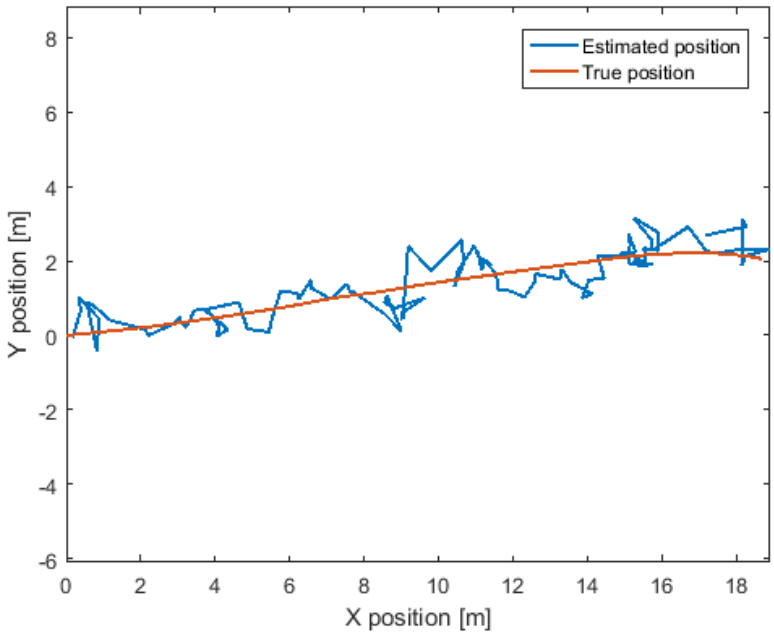


Figure 4.11: True position and estimated position for RobotCar dataset

The amplitude of the deviations for the whole trajectory can be seen in Figure 4.12 where the absolute mean error distributed along the trajectory in x and y direction are shown. The error is around 0.5 m but have a peak 1 m peak in x direction at the end of the trajectory.

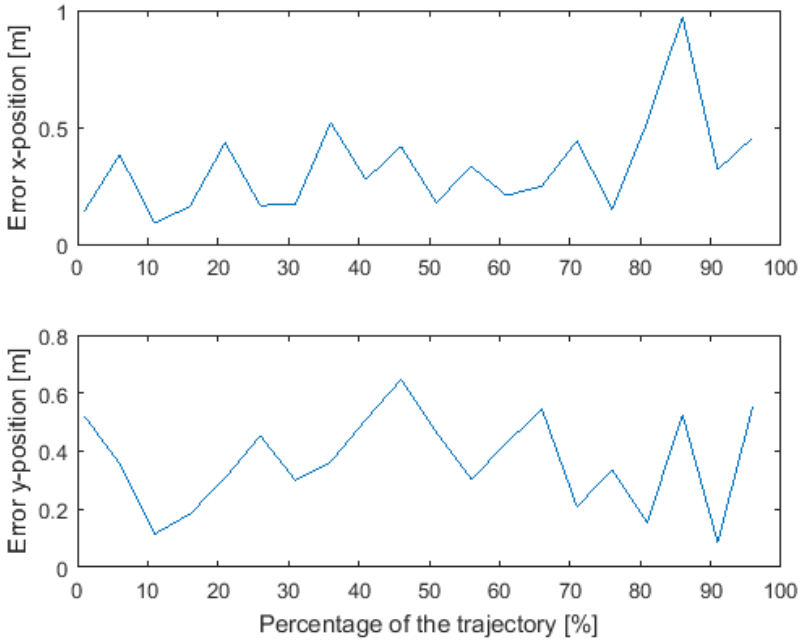


Figure 4.12: Absolute mean translational error in x and y coordinates for the complete system using RobotCar dataset

As for the KITTI dataset the second test uses only GPS for the localisation on the same trajectory. The result can be seen in Figure 4.13 where the mean is around 1 m for all parts of the trajectory.

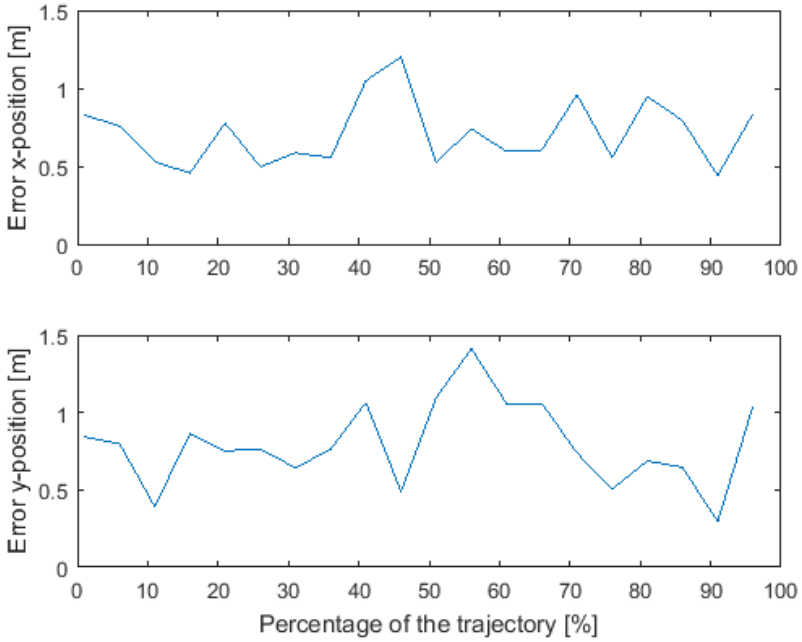


Figure 4.13: Absolute mean translational error in x and y coordinates using only GPS for the RobotCar dataset

The third test uses the full system but loss of GPS signal is simulated between 20% and 40% of the trajectory, the result of this can be seen in Figure 4.14. Also for this dataset the result from the localisation system are quite the same with and without simulated GPS loss.

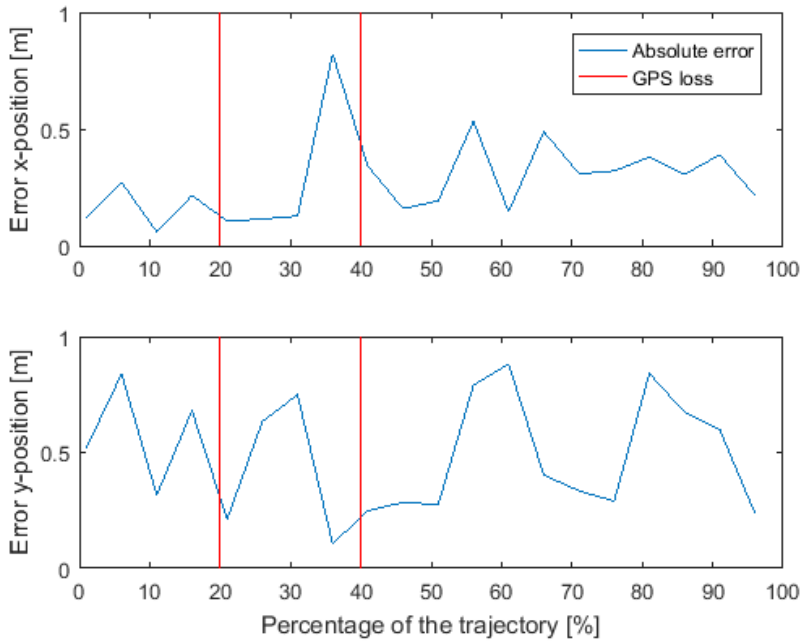


Figure 4.14: Absolute mean translational error in x and y coordinates for the complete system with loss of GPS signal between 20% and 40% of the trajectory for the RobotCar dataset

4.2.3 Ford Campus Vision dataset

The tests of the Ford Campus Vision dataset follows the same structure as the tests of the two dataset before and is going to be presented in the same way with graphs. The graphs in Figure 4.15 shows the the true positions together with the estimated positions using the complete system.

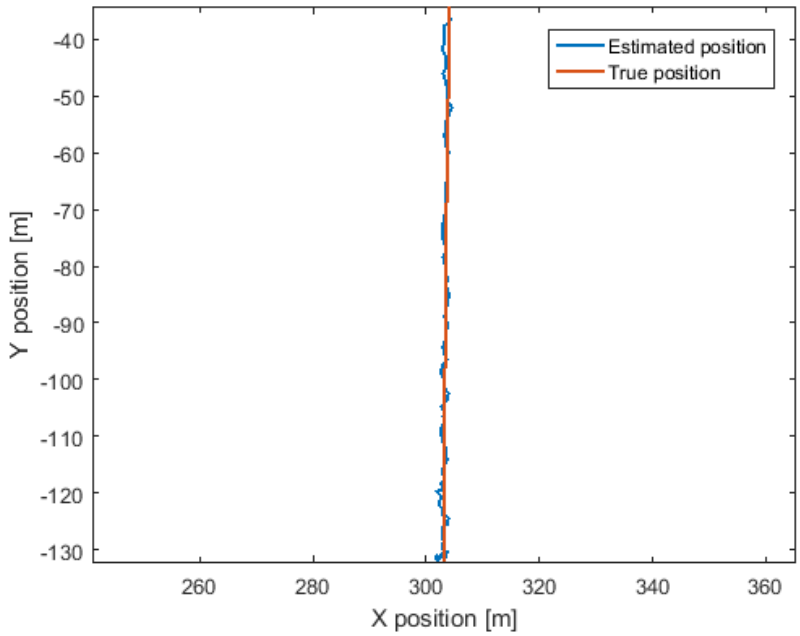


Figure 4.15: True position and estimated position for Ford Campus Vision dataset

The absolute mean translational error in x and y direction for this test can be seen in Figure 4.16. The error is around 0.4 m but have some peaks in different parts of the trajectory.

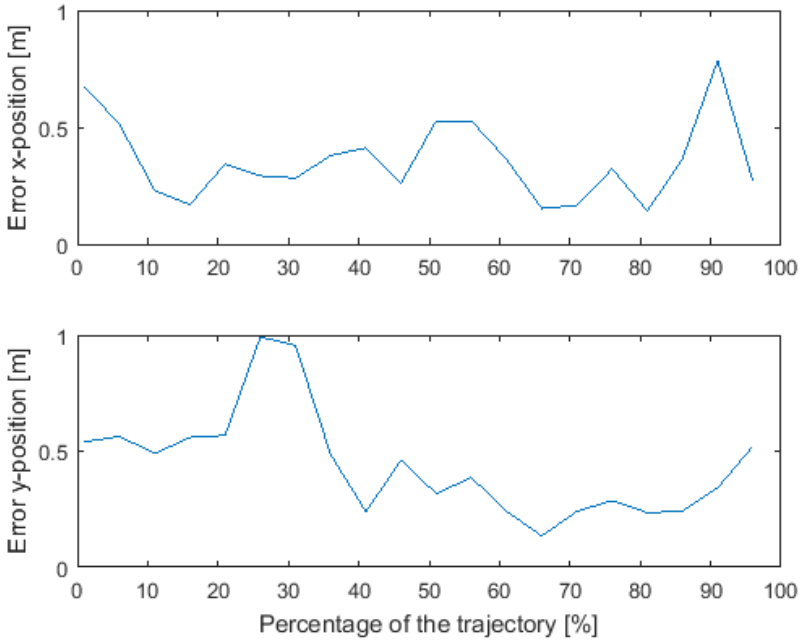


Figure 4.16: Absolute mean translational error in x and y coordinates for the complete system using Ford Campus Vision dataset

Only GPS is used for localisation in the second test. The error distributed along the trajectory is shown in Figure 4.17. The mean error is greater than 1.3 m in both x and y directions including some peaks up to 3 m.

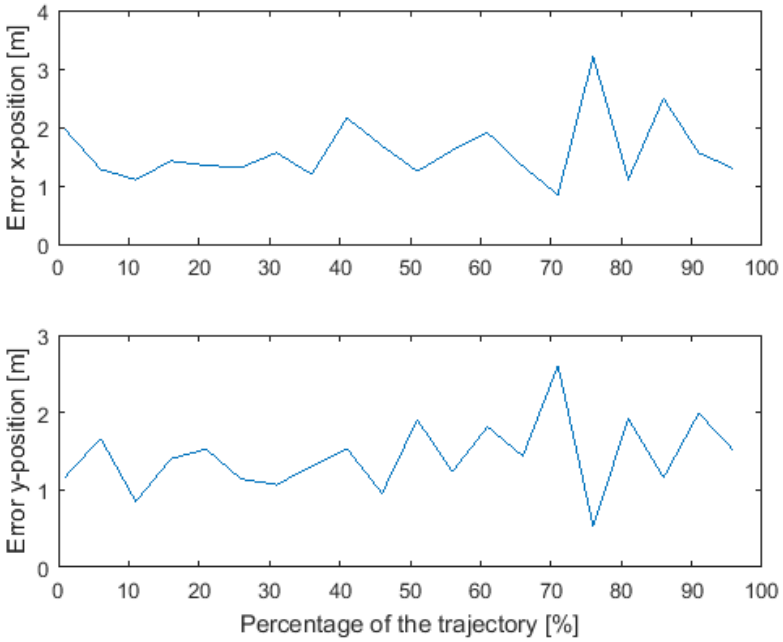


Figure 4.17: Absolute mean translational error in x and y coordinates using only GPS for the Ford Campus Vision dataset

The last test uses the full system but loss of GPS signal is simulated between 43% and 57% of the trajectory, the result of this can be seen in Figure 4.18. The mean error in the x direction is around 0.4 m which is less than the mean error in y direction, where the error is around 0.5 m including peaks up to 1 m.

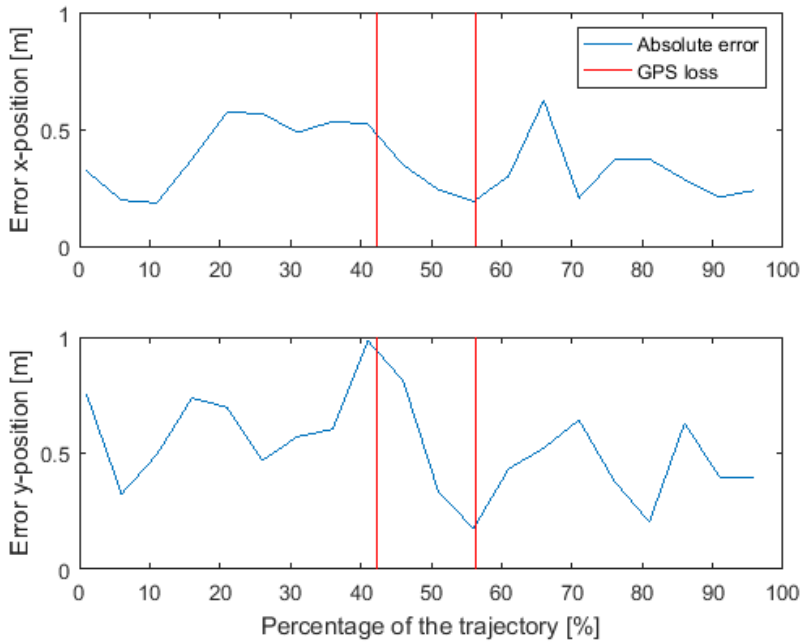


Figure 4.18: Absolute mean translational error in x and y coordinates for the complete system with loss of GPS signal between 43% and 57% of the trajectory for the Ford Campus Vision dataset

4.3 Performance versus Speed

As describes in Section 3.5.2 three tests of performance compered to computational speed is performed on the Ford Campus Vision dataset. The first experiment tests how the computational time and performance changes when decreasing the frequency of the triangulation algorithm to the EnKF. The performance of the three tests is measured in mean translational error and number of samples in percent over the threshold 0.5 meter. The results from the first experiment is shown in Table 4.2.

Algorithm frequency	Every	Every other	Every third
Mean error X [m]	0.42	0.45	0.67
Mean error Y [m]	0.54	0.58	0.86
Over threshold X [%]	32.9	38.3	50.7
Over threshold Y [%]	43.4	47.6	48.8
Time per sample [s]	0.5	0.25	0.17

Table 4.2: Table showing mean performance versus computational time over three runs with different frequency of triangulation algorithm

The second test compares the performance and speed by changing the number of ensembles in the EnKF. The result from this experiment can be seen in Table 4.3.

Number of Ensembles	10	50	100
Mean error X [m]	0.46	0.43	0.42
Mean error Y [m]	0.79	0.57	0.59
Over threshold X [%]	33.8	36.9	32.6
Over threshold Y [%]	59.9	48.4	45.8
Time per sample [s]	0.55	0.53	0.53

Table 4.3: Table showing mean performance versus computational time over three runs with 10, 50 and 100 ensembles in the EnKF.

In the third test three different types of image features have been tested for performance and speed. The results from this experiment is presented in Table 4.4.

Image Feature type	MSER	SURF	Eigen Values
Mean error X [m]	0.42	0.41	0.53
Mean error Y [m]	0.54	0.57	0.60
Over threshold X [%]	46.7	28.6	42.5
Over threshold Y [%]	61.7	46.2	48.1
Time per sample [s]	0.5	0.43	4.68

Table 4.4: Table showing mean performance versus computational time over three runs with MSER, SURF and Eigen Value image features

5
Discussion and Future Work

The overall results of the proposed algorithm looks promising regarding the limited data used for testing and evaluation. As can be seen for both the KITTI and Ford Campus Vision dataset the complete system gives better accuracy than the GPS alone. Something that is not showed in the result is that the GPS might perform with lower accuracy in cities with tall buildings where the GPS signal might be blocked. From the results when GPS loss is simulated it can be concluded that the algorithm wold be helpful in cities when the GPS looses signal.

As can be seen in the results from the KITTI dataset in Section 4.2.1 the system with the complete algorithm performs better than using only GPS. The same trend and results can be seen for the Ford Campus Vision dataset which means that the algorithm helps in estimating the position of the vehicle. The results from the RobotCar data also shows a improvement compared to only using the GPS but it does not give as good results as the other two datasets, the reason for this is discussed in Section 5.2. An advantage with the proposed algorithm, except from the improved accuracy, is that it gives an absolute estimate of the position which is not prone to drift like the INS.

As can be seen from the experiments, different parameters have a big impact on both the performance and the computational speed. In Table 4.2 both the mean error and the number values above the threshold increases when the algorithm is used less frequently but the time it takes to calculate each sample decreases a lot. In Table 4.3 there is not much difference between the the number of ensembles even though a slight trend can be seen where the accuracy increases with the number of ensembles. From the experiments with different features in Table 4.4 it is more difficult to clearly see which of the features is the most robust since they are quite different. The Eigen Values have the highest mean value but a small percentage of values over the threshold, this is due to that the Eigen Value function creates very many points and some are far way off. The MSER on the other hand produces much fewer points but they are more stable. The SIFT features are somewhere in between the MSER and the Eigen Values. The different features are good in different situations depending on the environment. It is also worth mentioning that the Eigen Values are much slower than the other two features which is due to the number of points produced.

All the speed measurements in the experiments are from `Matlab` which means that they are much slower than they would be in C++. As mentioned in Section 3.5.2

C++ code is 500 times faster than `Matlab` code which means that all of the times in the experiments would meet the 10Hz goal. The Eigen Value features which are the slowest can run at a rate of 10.8 Hz which just meets the goal, but in a real autonomous vehicle the algorithm have to be faster to allow other functions like, navigation, collision avoidance etc. to run within the 10 Hz specification.

5.1 Features

The feature detection used in this thesis is not that robust in varying light conditions which decreases the chances of good feature matching between the ground truthed point cloud and the current image, which results in lower accuracy of the position estimation. Since features from the image is extracted using `Matlab` the control over the features selected is poor which means that extracted features might be unusable. To get a better accuracy from the triangulation algorithm these features needs to be controlled in a way that they can be seen in all lighting conditions and that they land on good objects that will be there the next time the vehicle passes the same place.

Another problem with the features used is that they detects areas of intensity or corner points which may be very similar if there are identical lamp posts along a road which can give a false reading if the algorithm matches features from two different lamp posts. This creates problems in the Ford Campus Vision dataset where there are identical lamp posts on both sides of the road every ten meters.

To improve this the features need to be selected more carefully, for instance by finding structures like edges of building walls in the image. There are also a lot of universities working with finding more robust features that will be more reliable in different lighting conditions.

The Ford Campus Vision has a omnidirectional camera which means that the camera have a 360°vision. To extend the current algorithm, these images could be used for triangulation as well to get a better estimate of the position, and by doing this the algorithm will work much better when driving the opposite direction from the built point cloud. By increasing the number of images used for each estimation the accuracy should increase.

Another issue related to the features is when the projected LiDAR points in the image are matched to the features by finding the closest projected LiDAR point to the feature. If the feature is on the edge of a lamp post for instance the closest projected LiDAR might not be on the lamp post but instead measures the distance to the background, the triangulation is going to be very inaccurate.

5.2 Light Detection And Ranging

A source of error in the system is the projection of the LiDAR measurements into a image. The precision of this is extremely important for the algorithm to work properly, which can be hard to achieve without a extremely accurate calibration of the camera.

In the RobotCar data as mentioned earlier there are many shadow areas in the 3D point cloud which creates problems for the algorithm since the image features may be in the shadow area of the LiDAR and is therefor unusable which means that fewer features are used to triangulate the position of the vehicle, this leads to a less accurate estimation of the position of the vehicle.

5.3 Datasets

Two of the datasets, KITTI and Ford Campus Vision, used in the thesis have worked well since they use the same type of sensor setup as required for the algorithm to work properly. The drawback of the open source data is that they do not have the data we need to evaluate the algorithm during the correct conditions. This have restricted the testing and evaluation since it can not be stated that the algorithm works in complex road conditions.

The RobotCar dataset have some more issues than the other two datasets since the RobotCar are only equipped with a 2D LiDAR which means that for each frame in the algorithm a point cloud have to be built up from previous and future states which is very time consuming and also impossible in real world testing since the future states are unknown.

In the open source data they do not provide any exact ground truth data for the position of the vehicle, using for instance RTK, which is needed for the construction of the point cloud. Without this exact ground truth position the point cloud might not be a hundred percent accurate which decreases the performance of the proposed algorithm.

To get a better understanding for how accurate the proposed system actually is and how well it works in different complex road conditions a test vehicle is needed where the needed data can be logged with the correct sensors.

5.4 Future work

The LiDAR sensor is very expensive and therefore it would be good if in the future the proposed algorithm could work using only camera, INS and GPS. One possible way to do this is by creating the point cloud with the camera. This can be achieved by using something called Structure From Motion which basically takes points in

one image and track these points to a consecutive image. By knowing the calibration of the camera and the position of the points in the two images, 3D points can be triangulated. The position of the camera for each image can also be calculated. The main issue that have to be solved is to get a dens enough point cloud. For this algorithm to work a stereo camera, which is two cameras mounted beside each other, is needed to be able to triangulate points in the same time step.

In further development of the thesis it would be interesting to test different types of Gaussian filters. There are two filters that are the most interesting, an Extended Kalman Filter and a Particle filter. By using the particle filter it would be possible to use all measurements from the triangulation without the selection algorithm. The reason why it would be interesting to try different filters is because the EnKF is a fast but not that sophisticated as many other filters which could possibly yield a better result.

6
Conclusion

To conclude this thesis an algorithm using LiDAR and camera to localise a vehicle in complex road conditions was developed. The goal for the localisation system is to have a accuracy of about 30 to 50 centimeters. The system should also be able to run in real time and therefore a goal of 10 Hz was set. The algorithm uses a ground truth 3D map as a baseline for triangulating the current position of the vehicle. This estimated position is then fused together with measurements from a GPS and an IMU in an EnKF.

It is proven that the concept of localising a autonomous vehicle using camera and LiDAR to triangulate the position works with good accuracy. It is also shown that the result from triangulation can be improved by using an Ensemble Kalman Filter to fuse the data from the triangulation, INS and GNSS. The performance of the system in snowy conditions can not be fully tested due to the small amount of snow data and the small amount of snow in this dataset. However the mean translational accuracy from the complete system on the test data is around 0.4 meters which is within the set goal for the thesis even though there are some peaks over 0.5 meters. Over all the proposed system looks promising both with the accuracy and the computational time.

Bibliography

[1] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.

[2] Will Maddern, Geoff Pascoe, Chris Linegar, and Paul Newman. 1 Year, 1000km: The Oxford RobotCar Dataset. *The International Journal of Robotics Research (IJRR)*, 36(1):3–15, 2017.

[3] Gaurav Pandey, James R. McBride, and Ryan M. Eustice. Ford campus vision and lidar data set. *International Journal of Robotics Research*, 30(13):1543–1552, 2011.

[4] Travel calmer, safer, cleaner. `http://www.volvocars.com/au/about/innovations/intellisafe/autopilot`. Accessed: 2017-01-19.

[5] From autonomy to snowtonomy: How ford fusion hybrid autonomous research vehicle can navigate in winter.

[6] Snow and ice.

[7] B. Hofmann-Wellenhof, H. Lichtenegger, and J. Collins. *Global positioning system: theory and practice*. Springer, 2001.

[8] G Grisetti, R Kummerle, C Stachniss, and W Burgard. A tutorial on graph-based slam. *IEEE Intelligent Transportation Systems Magazine*, 2(4):31–43, 2010.

[9] A. Y. Hata and D. F. Wolf. Feature detection for vehicle localization in urban environments using a multilayer lidar. *IEEE Transactions on Intelligent Transportation Systems*, 17(2):420–429, Feb 2016.

[10] T. Caselitz, B. Steder, M. Ruhnke, and W. Burgard. Monocular camera localization in 3d lidar maps. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1926–1931, Oct 2016.

[11] J. Levinson, M. Montemerlo, and S. Thrun. Map-based precision vehicle localization in urban environments. *Robotics: Science and Systems III*, 2007. Accessed: 2017-05-28.

[12] K. Jo, Y. Jo, J. K. Suhr, H. G. Jung, and M. Sunwoo. Precise localization of an autonomous car based on probabilistic noise models of road surface marker features using multiple cameras. *IEEE Transactions on Intelligent Transportation Systems*, 16(6):3377–3392, Dec 2015.

[13] Peter I. Corke. *Robotics, vision and control fundamental algorithms in MATLAB*. Springer, 2011.

[14] Velodyne faq. `http://velodynelidar.com/faq.html`. Accessed: 2017-05-18.

[15] Velodyne hdl-64e. `http://velodynelidar.com/hdl-64e.html`. Accessed: 2017-05-18.

[16] 2d lidar sensorslms1xx / lms15x / outdoor. https://www.sick.com/de/en/detection-and-ranging-solutions/2d-lidar-sensors/lms1xx/lms151-10100/p/p141840. Accessed: 2017-05-18.

[17] David H. Titterton and John L. Weston. *Strapdown inertial navigation technology*. American Inst. of Aeronautics and Astronautics, 2009.

[18] Bernhard Hofmann-Wellenhof, Herbert Lichtenegger, and Elmar Wasle. *GNSS - Global Navigation Satellite Systems: GPS, GLONASS, Galileo and more*. Springer, 2008.

[19] Richard Szeliski. *Computer Vision: Algorithms and Applications*. Springer, 2010.

[20] H.Berg and R.Haddad. Visual odometry for road vehicles using a monocular camera. Master thesis, Department of Signals and Systems, Chalmers University of Technology, Gothenburg, Sweden, 2016. [Online]. http://publications.lib.chalmers.se/records/fulltext/246134/246134.pdf. Accessed: 2017-04-03.

[21] Jianbo Shi and C. Tomasi. Good features to track. In *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600, Jun 1994.

[22] J Matas, O Chum, M Urban, and T Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. *Image and Vision Computing*, 22(10):761 – 767, 2004. British Machine Vision Computing 2002.

[23] John P. Snyder and Philip M. Voxland. *An album of map projections*. US Government Print. Office, 1994.

[24] Mercator projection. academic.eb.com.proxy.lib.chalmers.se/levels/collegiate/article/Mercator-projection/52078#, 2017. Accessed: 2017-05-22.

[25] MILJENKO LAPAINE. *CHOOSING A MAP PROJECTION*. SPRINGER INTERNATIONAL PU, 2017.

[26] Simo Särkkä. *Bayesian filtering and smoothing*. Cambridge University Press, 2013.

[27] S. Gillijns, O.b. Mendoza, J. Chandrasekar, B.l.r. De Moor, D.s. Bernstein, and A. Ridley. What is the ensemble kalman filter and how well does it work? *2006 American Control Conference*, 2006.

[28] A. Fowler. Data assimilation tutorial on the kalman filter.

[29] Flea2 1.4 mp mono firewire 1394b (sony icx267). https://www.ptgrey.com/flea2-14-mp-mono-firewire-1394b-sony-icx267-camera. Accessed: 2017-05-18.

[30] Flea2 1.4 mp color firewire 1394b (sony icx267). https://www.ptgrey.com/flea2-14-mp-color-firewire-1394b-sony-icx267-camera. Accessed: 2017-05-18.

[31] Rt3000 v2 family. http://www.oxts.com/products/rt3000-family/, Oct 2016. Accessed: 2017-05-18.

[32] Bumblebee xb3 1.3 mp color firewire 1394b 3.8mm (sony icx445). Accessed: 2017-05-18.

[33] Grasshopper2 1.4 mp color firewire 1394b (sony icx285). https://www.ptgrey.com/grasshopper2-14-mp-color-firewire-1394b-sony-icx285-camera. Accessed: 2017-05-18.

[34] 3d lidar sensorsld-mrs / ld-mrs 4 layer / outdoor / long range. https://www.sick.com/de/en/detection-and-ranging-solutions/3d-lidar-sensors/ld-mrs/ld-mrs400001/p/p112355. Accessed: 2017-05-18.

[35] Span-cpt single enclosure gnss/ins receiver. https://www.novatel.com/products/span-gnss-inertial-systems/span-combined-systems/span-cpt/. Accessed: 2017-05-18.

[36] Ladybug3 12 mp firewire 1394b spherical digital video camera, red. https://www.ptgrey.com/ladybug3-12-mp-firewire-1394b-spherical-digital-video-camera-red. Accessed: 2017-05-18.

[37] Innovasium. Pos lv. https://www.applanix.com/products/poslv.htm, journal=Applanix. Accessed: 2017-05-18.

[38] Mti-g (legacy product) - products. https://www.xsens.com/products/mti-g/. Accessed: 2017-05-18.

[39] binaryfeatures. https://se.mathworks.com/help/vision/ref/detectmserfeatures.html.

[40] binaryfeatures. https://se.mathworks.com/help/vision/ref/extractfeatures.html. Accessed: 2017-05-19.

[41] Niklas Magnusson and Tobias Odenman. Improving absolute position estimates of an automotive vehicle using gps in sensor fusion. Master's thesis, 2012.

[42] 2.017j design of electromechanical robotic systems, fall 2009. (massachusetts institute of technology: Mit opencourseware). http://ocw.mit.edu. Accessed: 2017-05-19.

[43] Tyler Andrews. Computation time comparison between matlab and c++ using launch windows. 2012. Accessed: 2017-05-28.