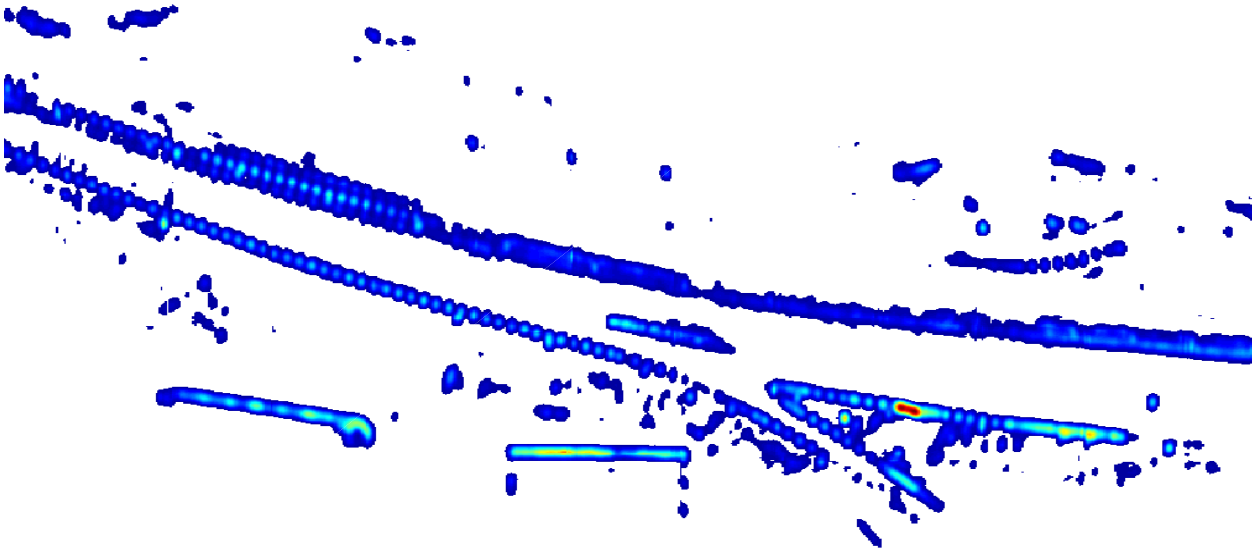


Crowd Sourcing of Sensor Maps

For Autonomous Driving Localization

Master's thesis in System, Control and Mechatronics

ANDREAS LÖFMAN
LINLIN GUO

Department of Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2018

Master's thesis EX018/2018

Crowd Sourcing of Sensor Maps

For Autonomous Driving Localization

ANDREAS LÖFMAN
LINLIN GUO

Department of Electrical Engineering
Division of Signals Processing and Biomedical Engineering
Signal Processing Group
Chalmers University of Technology
Gothenburg, Sweden 2018

Crowd Sourcing of Sensor Maps
For Autonomous Driving Localization
ANDREAS LÖFMAN
LINLIN GUO

Supervisor: Roza Maghsood, Zenuity
Supervisor: Erik Stenborg, Zenuity
Examiner: Carl Olsson, Department of Electrical Engineering, Chalmers

Master's Thesis EX018/2018
Department of Electrical Engineering
Division of Signals Processing and Biomedical Engineering
Signal Processing Group
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Sensor map constructed in Matlab showing radar-detections from a road segment along the DriveMe route around Gothenburg.

Typeset in LaTeX
Printed by [Name of printing company]
Gothenburg, Sweden 2018

Crowd Sourcing of Sensor Maps
For Autonomous Driving
ANDREAS LÖFMAN
LINLIN GUO
Department of Electrical Engineering
Chalmers University of Technology

Abstract

Autonomous driving is on the rise, as can be seen from the various efforts taken by companies worldwide to make the technology a realization in a near future. A driverless car needs to be able to determine its own position much more so than a human driver, which can be achieved by using sensor maps that describe stationary object in a certain area, as seen according to radars and cameras. These maps have traditionally been created using specialized probing vehicles with high-precision equipment, particularly expensive high-precision global positioning system (GPS) sensors with centimeter-accuracy.

However, the road network is constantly changing. When maps are needed to be quickly and continuously updated, this approach is no longer viable. Thus, this thesis answers the question whether or not it is feasible to instead generate said maps through crowd-sourcing data from a fleet of vehicles using consumer-grade GPS. This is done by following a process of firstly smoothing the GPS poses, then creating individual maps as seen from each car and lastly aligning and merging the separate maps using a Graph-SLAM approach. In order to benchmark-test the map, simulations of vehicle-localization are used in order to determine if the generated maps have sufficient precision to yield satisfying results.

Based on the results shown in this thesis, the authors conclude that a Graph-SLAM approach to solve the map-alignment problem yields maps that by performance is comparable to maps created from high-precision probing, in which localization is possible.

Keywords: Consumer GPS, Camera, Radar, Grid maps, SLAM, Crowd sourcing, Sensor maps, Localization, Autonomous Driving.

Acknowledgements

We would like to acknowledge the following persons, all of which have aided us in performing and writing this thesis.

- Roza Maghsood, our supervisor at Zenuity, who helped us not only with getting started and learning how things worked at Zenuity, but also continuosly made an effort of making sure we stayed on track and guiding us through the immense amount of code that was available.
- Erik Stenborg, our supervisor at Zenuity, who with his expertize helped us to understand the problems in more technical terms and helped us out when we got stuck.
- Carl Olsson, our thesis examiner at Chalmers, which we would like to thank for taking time to grade this thesis and making sure it obliges to academic standards.
- Tony Gustafsson, who helped us a lot at Zenuity by supplying and explaining the code regarding map-building and localization.
- The whole Magellan team at Zenuity, who provided everyday-aid of various issues and good advice, as well as good conversations over the fika-breaks.

Andreas Löfman, Linlin Guo, Gothenburg, June 2018

Contents

List of Figures

List of Tables

1
Introduction

The following section serves to give the reader an introduction to the background and purpose of the thesis and motivates the need of thereof, also comparing to previous works and thus identifying scientific contributions resulting from this thesis.

1.1 Background

Autonomous driving (AD) is currently one of the largest engineering challenges in the automotive industry. With the help of digitized solutions road safety can be greatly improved, where 1.2 million people died by traffic related accidents alone in 2013 [1]. Furthermore, traffic efficiency, driving comfort and most notably environmental impact can be improved with the help of AD by reducing traffic jams causing increased emissions of carbon dioxide compared to normal driving scenarios [2].

In order to safely guide vehicles on the road, it is paramount to have access to precise positioning in the world, with the help of sensor maps created with Global Positioning System (GPS) and Advanced Driving Assistance System (ADAS) sensors such as radar [3], camera and more recently also LiDAR [4]. Previously, these maps have been created by survey vehicles using high-precision equipment, similarly to the equipment used in *Use of Vision Sensors and Lane Maps to Aid GPS-INS Navigation* [4]. However, this costly process does not scale well with the increased need for continuous up-to-date maps of the environment for AD purposes.

A better alternative for future applications would be to crowd-source maps through a fleet of vehicles using consumer-graded GPS available in most cars. This will thus introduce additional uncertainty with regards to the actual position of each vehicle, which should be taken into account when creating (or updating) the collective sensor map. This can be achieved by aligning individual maps created as seen from each vehicle to achieve overlapping observations, thus yielding a consistent map.

1.2 Purpose

This thesis aims to test the feasibility of crowd sourced sensor maps created by using consumer GPS, radar and camera sensors commonly found in today's cars. In order to correctly align the crowd-sourced maps, the alignment will be formulated and solved as a Graph-SLAM problem [5].

In the future, the usage of cameras in cars in order to obtain visual information might be more common than using radar or both sensors. Thus reaching a conclusion regarding the ability to align maps that relies as little as possible on radar is also of interest in this thesis. Being able to create a map relying only on camera-information would be the ideal case, if possible.

Smoothing is a statistical technique that can remove irregularities of a sequence of data and improve the accuracy of data. Thus smoothing will be performed on raw GPS-poses in order to find the matching trajectory regarding the actual movement of the vehicle. The implementation and evaluation of this filter will thus also be of interest in this thesis.

Localization based on particle filtering is one example of a use-case of the sourced map, where the trajectory of the vehicle can be estimated based on observations of the world and comparing these to the map. This method will therefore be implemented for evaluating the performance of the map, by simulating a vehicle localizing itself in the map and compare the estimated trajectory to the true trajectory and thus obtain an estimate of the trajectory drift.

1.3 Previous works

The usage of consumer-grade sensors for autonomous cars has previously yielded promising results, as can be seen in the paper published by Z. Tao et al. [6], which used consumer GPS and camera sensors for performing real-time localization, an approach that has many similarities to this thesis. However, the localization required an accurate map which was pre-constructed using high-precision GPS, meaning the approach is not entirely independent of high-accuracy equipment.

L. Cao and J. Krumm's paper [7] provides an example of using consumer-grade sensors for map creation from data collected from several vehicles. Although the sourced map is represented as a road-network, more fitting for vehicle route planning, rather than for localization, the motivation behind the approach is similar.

The map-representation and SLAM-problem formulation presented in this thesis have similarities to previous works, such as the submap-stitching using Atlas-SLAM approach suggested by E. Rehder and A. Albrecht [8]. However the odometry measurements in their paper was supplied by wheel encoders, gyroscope and accelerometer rather than GPS measurements. Another notable difference is that the purpose of their SLAM-formulation is to perform stitching of adjacent submaps generated from the motion of a single vehicle, rather than aligning overlapping submaps from several vehicles passing the same area.

Previous in-house work at Zenuity regarding this subject has mainly revolved around smoothing of GPS-poses and the usage of radar information when aligning the maps. In this project, camera information will also be considered in the Graph-SLAM formulation. The fusion of radar and camera information in order to align the map

can be motivated by the argument that while radar detections provide good information in the longitudinal direction far away from the vehicle, detections in the lateral direction are lacking in precision. Meanwhile for cameras the reverse is true, since cameras perform best at estimating lanes close to the vehicle, but lack in depth-estimation. Thus using information from both sensors should arguably yield an improved alignment in both directions.

1.4 Research questions

The thesis aims to give answers to the following questions;

- Which map representation will be used? How to extract and transform features from the sensors to this map representation?
- Which form of metric should be used to evaluate if the performance of the GPS-smoothing is acceptable?
- How can the Graph-SLAM problem be formulated in order to perform map-alignment of maps created from separate data-collections? Is it a viable method?
- Which metric should be used to evaluate the results from using a particle filter on the map?
- How does the crowd-sourced map perform comparing to a map created using high-precision GPS?

1.5 Contributions

Based on the presented previous work in the area together with the purpose and problem formulation of this thesis, the following major contributions of this thesis to the field can be highlighted;

- Implementing a method which completely removes all dependence of high-precision GPS in the map-creation process.
- Fuse information from both radar and camera in the map-alignment process.
- Testing the viability of using a Graph-SLAM approach to solve map-alignment problems.

To our knowledge, these contributions have not been previously attempted.

1.6 Limitations

The thesis will limit its scope with regards to the following;

- The project will not include any LiDAR sensor data. Although LiDAR usage as a sensor type with regards to AD implementation is expected to rise in the future, as stated by J. Allen in [4], it is deemed as out of bounds for this project. This project will mainly focus on the usage of cameras, radars and GPS.

- The map being built will not be implemented in a 3D-representation, it is deemed as sufficient to have a 2D-representation of the map, as the road will be assumed to be flat.
- Since the purpose of this thesis is to test the concept of crowd sourced maps, optimizing the process with regards to speed or memory will not be a main focus.
- The major focus of this thesis will be the alignment of the map and evaluation, thus improving the smoothing and particle filter algorithms already present at Zenuity will not be in focus. Although needs for future work regarding those might be proposed.

1.7 Definitions

Table 1.1: Definitions of expressions commonly used in the report

Hits	Actual amount of detections being returned from a location
Looks	Maximum amount of detection returns that could potentially be achieved
Ground Truth	The assumed true trajectory of the vehicle
Ego	Ego meaning "self" and is often used in conjunctions to describe behaviour of the current vehicle, for example ego-poses, ego-heading and ego-speed
Segment	Element in the grid-environment dividing the world into local areas
RODS	Collective name for front and rear radar of the vehicle
FLC	Forward Looking Camera

1.8 Thesis layout

This thesis report will have the following structure; Chapter 2 gives the reader sufficient knowledge of the underlying theory as to be able to follow the rest of the report. Chapter 3 describes the implementation of the project, initially giving an overview of the work flow and subsequently a more detailed description. In Chapter 4 the results obtained from the project are presented and discussed with regards to the purpose and problem formulation. Chapter 5 presents the conclusions which can be drawn from the project and whether or not the purpose of the project can be considered achieved, along with suggestions for further work following the thesis.

2
Theory

In this section, underlying theory will be presented in order to equip the reader with the necessary knowledge of the subject serving as the background of this thesis. Theory regarding various sensors, filtering and smoothing concepts, coordinate frames and transformations, occupancy-grids, 2D cross-correlation and SLAM will be covered.

2.1 Crowd Sourcing

Crowd sourcing [9] is a concept mainly referring to problem solving through workload distribution over the Internet, where internet users each solve a small part of the problem, thus collaboratively reaching a total solution. There are several variations of this concept. One such variation related to, and thus the title of this thesis, would be a fleet of vehicles collectively creating/updating a map based on inaccurate observations from each vehicle, which will be compensated for as more information is added.

2.2 Sensors

Sensors are devices that can sense their surroundings and handle various input signals. Those inputs could be light, colors, distance, angles or any other environmental phenomena. The usual outputs of sensors are generally signals that can be understood by computers, with the potential of visualization for human comprehension. There are several types of sensors that are frequently used in daily life, for example pressure sensors, temperature sensors, position sensors, speed sensor and so on.

2.2.1 GPS

The Global Positioning System (GPS) is a space-based radionavigation system that provides geolocation (latitude, longitude and height) and time information to a GPS receiver anywhere on or near the Earth [10, 11]. Most GPS systems operate using three of more satellites in orbit around Earth, using timed signals to calculate the distance to each and thus triangulate the receiver's position on the surface, see Figure 2.1. GPS has numerous applications in different areas, for example navigation and tracking. In this project, the usage of consumer grade navigational GPS and high-precision real-time kinematics (RTK) GPS [12] for map creation will be discussed below.

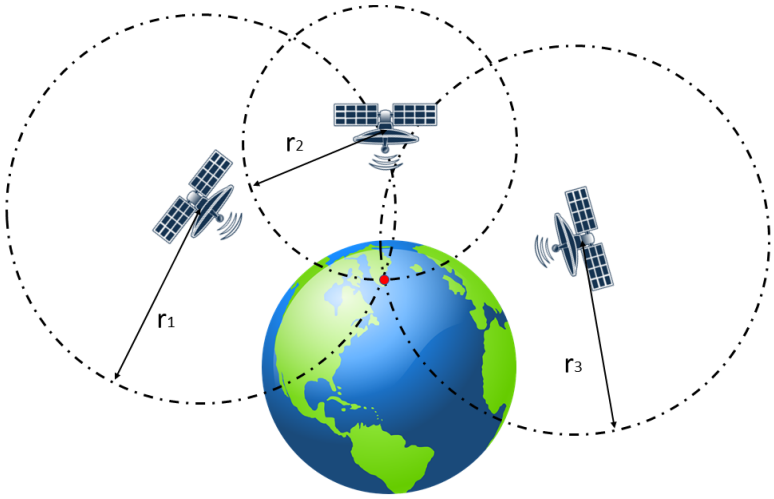


Figure 2.1: Using the distance to each satellite, the receiver's position on Earth can be determined.

Consumer-grade GPS

Consumer-grade GPS is currently available in most modern cars with the purpose of navigation. The positional accuracy that is available differs depending on manufacturer, but most literature describes the typical error as around 10 meters [13–16]. This error is acceptable for navigation, however not for lane-positioning of autonomous cars. It is possible to post-smooth the noisy GPS measurements in order to yield an offsetted approximation to the true trajectory of the vehicle (see Section 2.3), which can then be used to describe the relative motion of the current vehicle.

Real time kinematic GPS

Real time kinematic (RTK) GPS is able to provide global position with very high accuracy, the errors obtained by the RTK-GPS are at the centimeter-level [12]. The basic principle of RTK positioning is having one or several base-stations with previously determined positions (using several stations is called Network-RTK [17]) for error corrections at a rover-receiver. Thus RTK GPS can serve as a good "absolute ground truth" when evaluating the smoothed SPA GPS. However, the RTK receiver is much more expensive compared to the consumer-grade GPS, with current prices ranging in thousands of US dollars [18], meaning it can be considered as unfitting for usage in mass-produced products.

2.2.2 Inertial measurement unit

Inertial measurement unit (IMU) is a sensor mass that contain precision gyroscopes, accelerometers, and magnetometers sensors [19]. Thus these inertial sensors can measure not only angular velocity and linear acceleration, but also magnetic field strength. A common IMU has six degree of freedom, three orthogonal gyroscope axes and three orthogonal accelerometer axes. When a low cost IMU is used in conjunction with consumer GPS and/or other cheap sensors, the localization and

mapping performance can be improved with increased accuracy [20].

2.2.3 Radar

A radar operates through a fairly simple concept [21]; The radar transmitter emits waveform-signals of electromagnetic radiation with known frequency towards an area, and a receiver detects echos of the signal being reflected back from objects. By analyzing the time between transmitting and receiving the signal, the range to the object can be determined. The angle to the object can also be determined by detecting the return angle of the signal. Based on the range and angle, the position of a detection relative to the radar device can thus be obtained. By also analyzing the shift in frequency of the returned signal (caused by the Doppler-effect [22]), a relative velocity of the detected object can be determined, thus allowing the distinction of stationary objects from moving targets. The usage of radars in the automotive industry is becoming increasingly more common, mainly used for driving-assistance and safety applications such as the Volvo Auto-Break [23] or Adaptive Cruise Control (ACC) [24].

2.2.4 Forward looking camera

Forward looking camera (FLC) is an image sensor using visible light as input, which aims to identify objects in front of the car, especially lanes [25]. Using well-known computer vision concepts (such as Hugh-transform and thresholding), the left and right markings of the current lane can be detected as seen from the camera, and by transforming the detected lanes from the projection frame to bird's eye vision using a top-view transformation [26], as illustrated in Figure 2.2, the lanes can thereafter be expressed with positions relative a local vehicle coordinate system.

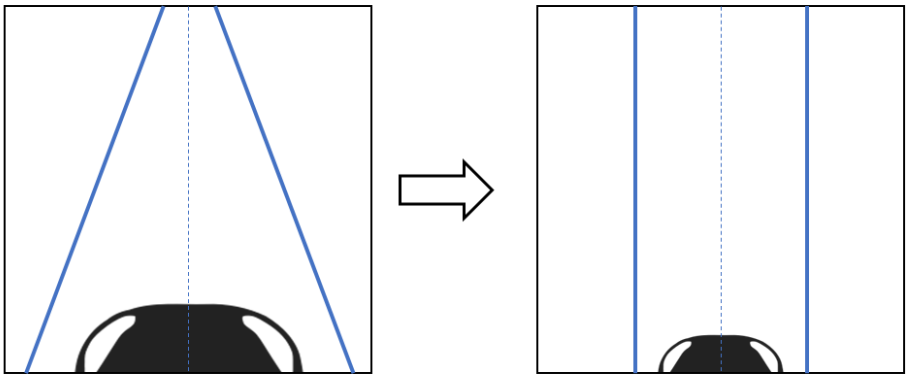


Figure 2.2: Transformation from the cameras projection frame to top-view frame.

2.3 Gaussian Kalman filtering and smoothing

Gaussian Kalman filtering [27] is a very well-established filter technique to statistically estimate the state of a process based on (noisy) output measurements.Kalman filters are commonly used for estimation, prediction and smoothing problems, as most practical problems are nonlinear by nature. There are many variations, such as extended Kalman filters (EKF), unscented Kalman filters (UKF), cubature quadrature Kalman filters (CQKF) and cubature Kalman filters (CKF). [28]

2.3.1 Motion models and measurement models

In order to use Kalman filtering, the problems need to be stated in state-space form. State-space representation is the concept of describing the evolution of a process in the form of a state-vector. For a car this state could for example consist of position, speed, heading and yaw rate. For discrete time, let x_k be said variable vector at time instance k, then the following state-space model represents a given process:

$$\begin{aligned} x_{k+1} &= f(x_k, u_k) + n_k, \\ y_k &= h(x_k, u_k) + m_k, \end{aligned} \tag{2.1}$$

where x_{k+1} is the value of the state at time instance k+1, u_k and y_k are the input and measured output respectively to the system at time k, and n_k and m_k are the process and measurement noise respectively affecting the system, if any are present. $f(.)$ is called the motion model for the state, and gives a prediction of the next value of the state as a function of the previous state and input. $h(.)$ is the measurement model, and describes y_k as a function of the current state and input.

A general Markov motion model and measurement model can also be written as

$$\begin{aligned} x_{k+1} &\sim p(x_{k+1}|x_k), \\ y_k &\sim p(y_k|x_k), \end{aligned} \tag{2.2}$$

where $p(.)$ is a term for the probability density function (PDF). Of particular interest for performing Kalman filtering and smoothing is to have a correct motion model. For describing vehicle dynamics, there are several well-defined motion models describing vehicle behaviour such as the constant velocity (CV), constant acceleration (CA) and coordinated turn (CT) models [29].

2.3.2 Prediction and measurement update

Kalman filters are recursive, so new measurements would be used when they are available. Each recursion contains two main parts, one is prediction which is also called time update; another is measurement update. The corresponding distribution of predictive and updated state are computed by solving several integrals shown as following:

Prediction:

$$\hat{x}_{k|k-1} = \int f(x_{k-1})\mathcal{N}(x_{k-1}; \hat{x}_{k-1|k-1}, P_{k-1|k-1})dx_{k-1}, \tag{2.3a}$$

$$P_{k|k-1} = \int (f(x_{k-1}) - \hat{x}_{k|k-1})(f(x_{k-1}) - \hat{x}_{k|k-1})^T \mathcal{N}(x_{k-1}; \hat{x}_{k-1|k-1}, P_{k-1|k-1})dx_{k-1} + Q_{k-1}, \tag{2.3b}$$

$$\hat{y}_{k|k-1} = \int h(x_k)\mathcal{N}(x_k; \hat{x}_{k|k-1}, P_{k|k-1})dx_k, \tag{2.3c}$$

where f(.) is a non-linear function, $\mathcal{N}(.)$ is the corresponding Gaussian probability density and Q_k is the covariance noise at time k. In this step, the predictive distribution $\hat{x}_{k|k-1}$ is estimated by the Chapman-Kolmogorov equation and the covariance $P_{k|k-1}$ is computed based on a quadratic loss function.

Update:

$$P_{xy} = \int (x_k - \hat{x}_{k|k-1})(h(x_k) - \hat{y}_{k|k-1})\mathcal{N}(x_k; \hat{x}_{k|k-1}, P_{k|k-1})dx_k, \tag{2.4a}$$

$$S_k = \int (h(x_k) - \hat{y}_{k|k-1})(h(x_k) - \hat{y}_{k|k-1})^T \mathcal{N}(x_k; \hat{x}_{k|k-1}, P_{k|k-1})dx_k + R_k, \tag{2.4b}$$

where h(.) is a non-linear functions and R_k is the measurement noise. The measurement update $\hat{y}_{k|k-1}$, filter gain P_{xy} and measurement prediction covariance S_k have been calculated before updating state and covariance. All those above will be formed in order to calculate the estimated state vector $\hat{x}_{k|k}$ and covariance $P_{k|k}$:

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + P_{xy}S_k^{-1}(y_k - \hat{y}_{k|k-1}), \tag{2.5a}$$

$$P_{k|k} = P_{k|k-1} - P_{xy}S_k^{-1}P_{xy}^T. \tag{2.5b}$$

2.3.3 Smoothing

Smoothing is the process of creating an approximating function that can generate a set of smooth data by removing noise and detail. If Gaussian filters can be regarded as 'forward' time filters, smoothing would be considered as 'backward' filters, which can improve the accuracy of current state by using data from both previous and later times [28]. Rauch–Tung–Striebel (RTS) smoothing is one of generally used smoothing methods and the basic description of this algorithm is as follows:

- Running a Gaussian filter for k=1,2,...,T and store the moments $\hat{x}_{k|k}$, $P_{k|k}$, $\hat{x}_{k+1|k}$ and $P_{k+1|k}$.
- The smoothing recursive starts with the last time step T; for k=T-1, T-2,...,1 compute

$$G_k = P_{k|k}A_k^T P_{k|k}^T, \tag{2.6}$$

$$\hat{x}_{k|T} = \hat{x}_{k|k} + G_k(\hat{x}_{k+1|T} - \hat{x}_{k+1|k}), \tag{2.7}$$

$$P_{k|T} = P_{k|k} - G_k[P_{k+1|k} - P_{k+1|T}]G_k^T, \tag{2.8}$$

where A_k is the transition matrix for state prediction $\hat{x}_{k+1|k} = A_k\hat{x}_{k|k}$.

In this thesis, the Rauch–Tung–Striebel (RTS) smoothing, or to be more precise, the square-root unscented Rauch-Tung-Striebel smoother will be used in implementation.

2.3.4 The Square-Root Unscented Kalman filter and smoothing

The Unscented Kalman filters (UKF) is a variation of the general Kalman filter and used for approximating nonlinear distributions for Gaussian models. The Square-root Unscented Kalman filter (SR-UKF) is another approach for estimating the non-linearity in the distribution. Compared with UKF, SR-UKF can not only keep the stability of a time series data, but also guarantee the state covariances staying positive semi-definite [30].

A general Square-Root UKF algorithm for state-estimation can be described as according to Algorithm 1. To initialize SR-UKF, the zero'th estimation of the state $\hat{X}_0$ corresponds to the expected value of prior X_0 and the zero'th Cholesky factor S_0 is the matrix square-root (qr) of expected covariance (see Equations (2.9) and (2.10)). When calculating the weight scalar in Equations (2.12), (2.13) and (2.14), n is the dimension of the state vector, λ is a scaling parameter, α is a constant that denotes the spread of sigma points χ_k around $\hat{X}_k$, β is a parameter that incorporate the distribution of previous state and j=1,2,....2n. In order to generate a time update and measurement update, the functions f(.) and h(.) are used to compute sigma points prediction and measurement sigma points where u_k is exogenous input at time k, cholupdate (available in Matlab) [31] is the Cholesky update [32] which is an effective rank 1 update algorithm to compute the new Cholesky factor. This algorithm could consecutively provide update results by using specific columns of U.

The Square-Root Unscented Rauch-Tung-Striebel smoother follows similar steps as the Square-Root Unscented Kalman filter; the difference is that the estimated states $\hat{X}_{k|k}$, predicted states $\hat{X}_{k|k-1}$, predicted Cholesky factor $S_{k|k-1}$ and updated Cholesky factor S_y are stored after performing Square-Root UFK. These are used in the RTS smoother algorithm (see Section 2.3.3).

Algorithm 1 Square-Root Unscented Kalman filter (SR-UKF)

1: State initialization:

$$\hat{X}_0 = E[X_0] \tag{2.9}$$

$$S_0 = E[(X_0 - \hat{X}_0)(X_0 - \hat{X}_0)^T] \tag{2.10}$$

2: Sigma points:

$$\chi_{k-1|k-1} = [\hat{X}_{k-1|k-1,0} \quad \hat{X}_{k-1|k-1} + \gamma S_{k-1|k-1,j} \quad \hat{X}_{k-1|k-1} - \gamma S_{k-1|k-1,j}] \tag{2.11}$$

3: Corresponding weights:

$$W_0^{(m)} = \frac{\lambda}{\lambda + n} \tag{2.12}$$

$$W_0^{(c)} = \frac{\lambda}{\lambda + n} + (1 - \alpha^2 + \beta) \tag{2.13}$$

$$W_j^{(m)} = W_j^{(c)} = \frac{1}{2(\lambda + n)} \tag{2.14}$$

4: Prediction mean and covariance:

$$\chi_{k|k-1,j} = f(\chi_{k-1|k-1,j}, u_{k-1|k-1}) \tag{2.15}$$

$$\hat{X}_{k|k-1} = \sum_{j=0}^{2n} W_j^{(m)} \chi_{k|k-1,j} \tag{2.16}$$

$$S^-_{k|k-1} = qr([\sqrt{W^{(c)}_{1:2n}}(\chi_{k|k-1,1:2n} - \hat{X}_{k|k-1}R]) \tag{2.17}$$

$$S_{k|k-1} = cholupdate([S^-_{k|k-1}, \chi_{k|k-1,0} - \hat{X}_{k|k-1}, W_0^{(c)}]) \tag{2.18}$$

5: Measurement sigma-points:

$$Y_{k|k-1,j} = h(\hat{X}_{k|k-1,j}, v_{k-1|k-1}), j = 1, 2, 3...2n \tag{2.19}$$

$$\hat{Y}_{k|k-1} = \sum_{j=0}^{2n} W_j^{(m)} Y_{k|k-1,j} \tag{2.20}$$

6: Measurement update:

$$S_y^- = qr([diag(\sqrt{W^{(c)}_{1:2n}})(Y_{k|k-1,1:2n} - \hat{Y}_{k|k-1}), S_R]) \tag{2.21}$$

$$S_y = cholupdate([S_y^-, X_{k|k-1,0} - \hat{X}_{k|k-1}, W_0^{(c)}]) \tag{2.22}$$

$$p_{xy} = \sum_{j=0}^{2n} W_j^{(m)} (\chi_{k|k-1,j} - \hat{X}_{k|k-1})(Y_{k|k-1,j} - \hat{Y}_{k|k-1})^2 \tag{2.23}$$

$$K = (p_{xy}/S_y^T)/S_y \tag{2.24}$$

$$X_{k|k} = \hat{X}_{k|k-1} + K\hat{Y}_{k|k-1}) \tag{2.25}$$

$$S_{k|k} = cholupdate(S_{k|k-1}, U, -1), \quad U = KS_y \tag{2.26}$$

7: Repeat steps 2 to 6

2.4 Particle filtering

Particle filtering is a popular technique to estimate a state sequence, especially when the posterior distribution is non-Gaussian [33]. The primary application for particle filtering is to estimate the trajectory of observations, which can be seen as "tracking" the current position of interests. Monte Carlo methods are important techniques behind particle filters [28], which can be used to approximate a posterior distribution through a set of distributed samples. The basis of Monte Carlo methods can be formed by importance sampling (IS). As can be seen in Algorithm 2, samples are drawn from the approximate distribution $q(x)$ which is also called the importance distribution. q(x) is "similar" to p(x) which is computed in Equation (2.27). N is the number of particles, $\delta(.)$ is the Dirac delta function, x is the mean of the state vector, $x^{(i)}$ are the particles and $\omega^{(i)}$ are the associated weights.

Algorithm 2 Importance Sampling

Draw samples, $x^1, x^2, x^3, ..., x^N$, from $q(x)$ and compute

$$p(x) \approx \sum_{i=1}^{N} \omega^{(i)} \delta(x - x^{(i)}) \tag{2.27}$$

where

$$\omega^{(i)} = \frac{\tilde{\omega}^{(i)}}{\sum_{i=1}^{N} \tilde{\omega}^{(i)}}, \quad \tilde{\omega}^{(i)} = \frac{p(x^{(i)})}{q(x^{(i)})} \tag{2.28}$$

Sequential importance sampling (SIS) is a version of importance sampling and used for filter distribution approximation. Furthermore, sequential importance resampling (SIR) can be considered as a necessary step when the number of effective samples are too small. Without resampling there will be no feedback from observations, thus sample degeneracy will occur, causing a poor estimation of the state sequence. SIS and SIR are given by Algorithm 3. Firstly, samples are generated from the importance distribution and weights are normalized. Then, a mass of particles with associated weights represent different distributions in the areas of interest according to measurement update. In addition, each weight contributes to the corresponding estimation. Finally, new samples would be drawn from current distribution with normalized weights.

SIS and SIR are two main parts for implementing a particle filter. It can be generally described as a series of movements that are taken, causing a mount of particles to gather in high probability regions with weights that update over time. However the implementation of PF is very computationally expensive, since a large number of particles have to be taken into account.

Algorithm 3 Particle filtering (PF)

Sequential importance sampling:

- Draw N samples from prior

$$x_0^{(i)} \sim p(x_0), i = 1, 2, 3.....N \tag{2.29}$$

- The corresponding weights
$\omega_0^{(i)} = \frac{1}{N}$, i=1,2,3.....N
- At each time K, draw samples $x_k^{(i)}$ and compute new weights

$$x_k^{(i)} \sim q(x_k|x_{k-1}^{(i)}, y_{1:k}), i = 1, 2, 3.....N \tag{2.30}$$

$$\omega_k^{(i)} \propto \omega_{k-1}^{(i)} \frac{p(y_k|x_k^{(i)})p(x_k^{(i)}|x_{k-1}^{(i)})}{\pi(x_k^{(i)}|x_{k-1}^{(i)}, y_{1:k})} \tag{2.31}$$

then normalize weights to sum of unity.

Sequential importance resampling:

- Interpret each weight $w_k^{(i)}$ as the probability of obtaining the sample k from the particle index i of states.
- Draw N samples from that discrete distribution and replace the old samples set with new ones.
- Set all weights to the constant value $\frac{1}{N}$.

2.5 Coordinate frames and transforms

A coordinate system or frame is a reference system used to represent coordinates with respect to a specified origin. There are several types of frames, on a global and local scale.

2.5.1 Global geodetic coordinate systems

Global geodetic coordinate systems are used to express large-scale coordinates, where the curvature of the Earth has to be taken into account. These systems are thus expressed in three dimensions. One such coordinate frame is the global geodetic coordinate representation WGS 84 [34], an Earth-centered, Earth-fixed (ECEF) frame with origin in the mass-centre of the Earth. In WGS 84, coordinates are expressed as the geodetic angles longitude, latitude and height above the reference ellipsoid which serves as an approximation of the Earth surface. GPS measurements, which needs to be expressed similarly across the globe, are expressed according to WGS 84. Another example of a coordinate system is SWEREF 99 (Swedish reference frame 1999) [35], which is a Swedish reference system based on 21 stations spread around Sweden. SWEREF 99 can be used to express local coordinates (in the case of this thesis, the area of Gothenburg) more accurately since the frame is relative to Sweden.

2.5.2 Local Cartesian coordinate systems

On a smaller scale, where the effects of the curvature of the Earth is negligible, expressing coordinates using local Cartesian coordinate frames are much more preferred, i.e. a frame with axis pointing in x,y, and possibly z-direction. These local frames are usually projections of coordinates from a global frame upon a 2D-surface coinciding with the surface of the Earth around the origin, meaning the errors grow the farther the coordinates are translated from the origin. A common frame being the East-North-Up (ENU) frame, with the x-axis in eastwards direction, y axis in northward direction and a (potential) z-axis in the direction orthogonal to the surface, see Figure 2.3 for an illustration.

It is possible, and not uncommon, to have multiple local coordinate systems in the same small vicinity. One such example would be sensor-readings that results in detection-positions expressed according to the ego-vehicle frame. It can then be preferred to transform these into another frame in which the vehicle itself is positioned in. Expressing local coordinates given in one local frame as seen from another can be represented as using a rotational matrix together with a translation vector. For the 2-dimensional euclidean space, a counter-clockwise rotation of angle θ and translation of a frame can be expressed as

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}, \tag{2.32}$$

where (x',y') corresponds to the old coordinates (x,y) but expressed in the new frame.

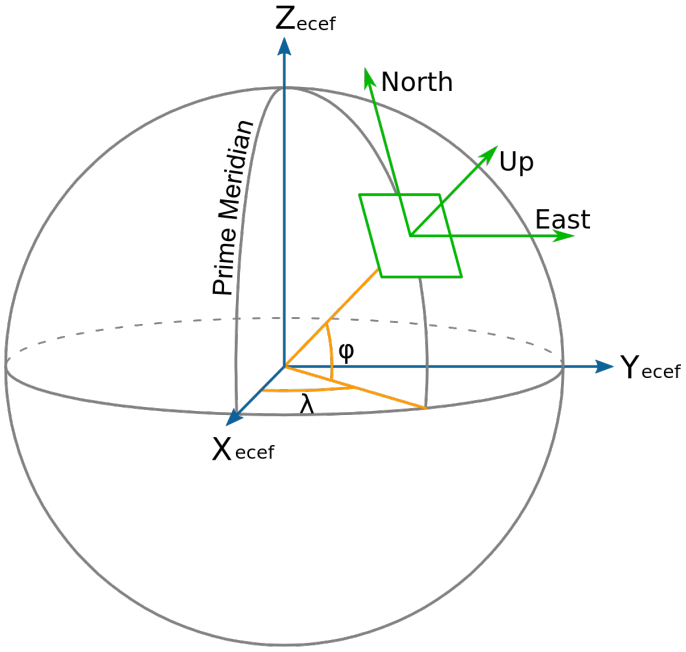


Figure 2.3: Illustration of the ENU local frame in relation to the geodetic ECEF-frame. Image fetched from [36] (public domain).

2.5.3 Transformation from geodetic to Cartesian

Since vehicles are traveling along the surface of the Earth and in relatively small areas, it is preferred to express their positions according to a local Cartesian frame. However, since GPS-coordinates are given in the geodetic frame WGS 84, in order to work in a local Cartesian frame there is a need to transform these using a projection-transformation. There are many approaches to this projection, one commonly used is the Gauss-Krüger projection [37]. The transformation is performed by solving a set of standard equations, albeit not completely effortlessly, and will thus not be expressed in detail in this thesis. For a detailed description of these equations refer to [38], Chapter 8.2.

2.6 Occupancy-grid submaps

Occupancy grid maps [39] represents the world as divided into grids (see Figure 2.4 for an example) where each grid consists of a block of cells, each cell either occupied by an object with a certain likelihood or unoccupied. This likelihood can be approximated by using the amount of detections returned from an object (hits) divided by the maximum amount of possible detection returns (looks). After post-processing the raw data collected from sensors, features of interest will be mapped into corresponding grids, where each grid can be checked independently from others. If the sensors used to create the map have low accuracy, features of interest will be incorrectly mapped into grids which will result in invalid grid maps. Hence, how to compensate for this measurement inaccuracy is an important problem that needs to be taken into account.

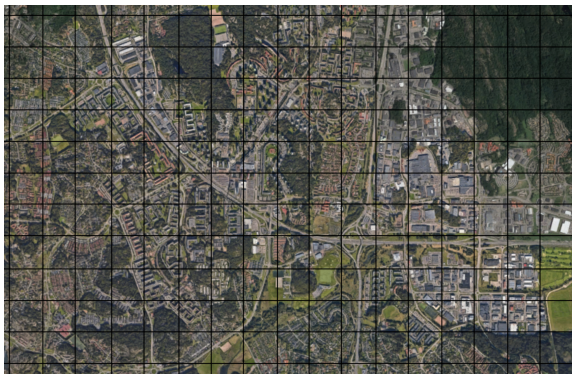


Figure 2.4: An example of how a local area of Gothenburg could be divided into grid maps. The satellite image is fetched from Google Maps, 2018 [40].

2.7 2D cross-correlation

Cross-correlation is widely used as a similarity measurement in signal processing. Similarly, 2D cross-correlation is used for image pattern matching [41], where one image serve as a template to be matched in the other, and is a way to find the positions where two images (or matrices) most resemble each other. This is done

by choosing one of the images as the template, and iteratively shift this image over the other and calculate the overlapping sum for each element in the cross-correlation matrix. Consider two images A and B represented as matrices, where A is of dimension (N, M) and B is of dimension (V, W). The resulting cross-correlation matrix will thus be of dimensions $(N + V - 1)$ by $(M + W - 1)$. The value in the (i, j):th element of the cross-correlation matrix R can thus be calculated as

$$R(i,j) = \sum_{(n,m),(v,w)\in C(i,j)} A(n,m)B(v,w), \tag{2.33}$$

where $C(i, j)$ is the set of overlapping elements in A and B when calculating the element (i, j) of R. Based on the position of the maximum value in R, it is then possible to determine the relative x and y shift between the template and the main image.

2.8 Simultaneous localization and mapping

In robotic navigation, simultaneous localization and mapping (SLAM) [42] is the process of iteratively constructing a map of the robot's unknown surroundings while simultaneously finding the robot's unknown position in said map, hence the name. For this project offline SLAM will be considered, i.e. data will have already been collected and the total map will be built à posteriori. Compare this to online SLAM, where mostly the current position and immediate environment of the robot is of interest. The SLAM-problem formulation is given as estimating the posterior probability of the robot's trajectory and the map of the environment. For offline SLAM, this can be formulated as

$$p(x_{1:T}, m|z_{1:T}, u_{1:T}, x_0), \tag{2.34}$$

where $x_{1:T}$ is the robot's trajectory over the time-span $(1, ...T)$ where T is the final time, m is the map, $z_{1:T}$ are measurements, $u_{1:T}$ are control inputs and x_0 is a given initial position. Compare this with $p(x_k, m|z_{1:k}, u_{1:k}, x_0)$, which is the formulation for online SLAM regarding the robot's pose at time k.

2.8.1 Graph-SLAM

One representation of the SLAM-problem for offline mapping can take the form of a graph, with nodes corresponding to the robot's poses and edges representing spatial constraints between each node-pair from measurements. This representation of the SLAM-problem is called Graph-based SLAM [5]. Figure 2.5 provides a visual representation of how the nodes in the graph are connected. The predicted measurement $\hat{z}_{k,k+1}$, usually called a virtual measurement, from node x_k to x_{k+1} express the position of x_{k+1} as seen from x_k based on the current node-configuration. $z^{1,2}_{k,k+1}$ denotes actual measurements of x_{k+1} as observed from x_k, which results in multiple errors $e^{1,2}_{k,k+1} = z^{1,2}_{k,k+1} - \hat{z}_{k,k+1}$. Thus the solution to the Graph-SLAM problem is the node configuration that minimizes the sum of all errors squared, also known as the log-likelihood of all observations, alternatively meaning the configuration that

maximizes the consistency of measurements. This can be expressed as finding the optimal x^* that minimizes the least-squares error

$$F(x) = \sum_{(k,k+1)\in M} e_{k,k+1}^T \Omega_{k,k+1} e_{k,k+1}, \tag{2.35}$$

where $\Omega_{k,k+1}$ is the information matrix, meaning the inverse of the covariance matrix, that expresses the uncertainty of the measurement and M is the set of measurements resulting in errors in the graph-nodes.

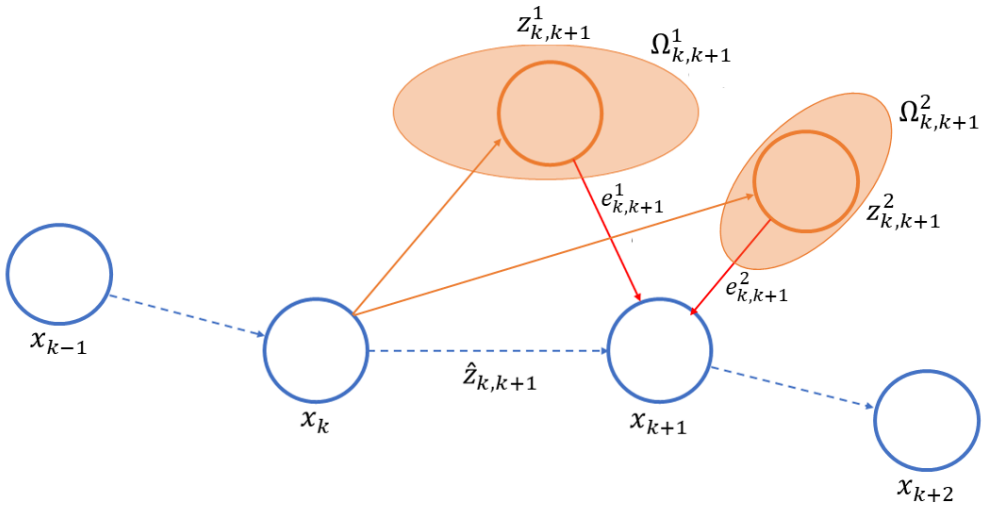


Figure 2.5: Visual illustration of the Graph-SLAM principle. Nodes correspond to poses of the robot, and edges refer to measurements. Inconsistencies in the measurements z^1 and z^2 will produce different errors to be minimized.

3
Crowd Sourcing Implementation

In this chapter the implemented process of Crowd Sourcing will be presented, starting at the initial smoothing of GPS poses and ending at the evaluation of the generated sensor map.

3.1 Process overview

An overview of the whole process is illustrated in Figure 3.1, which also includes a comparison versus the original RTK map-generation.

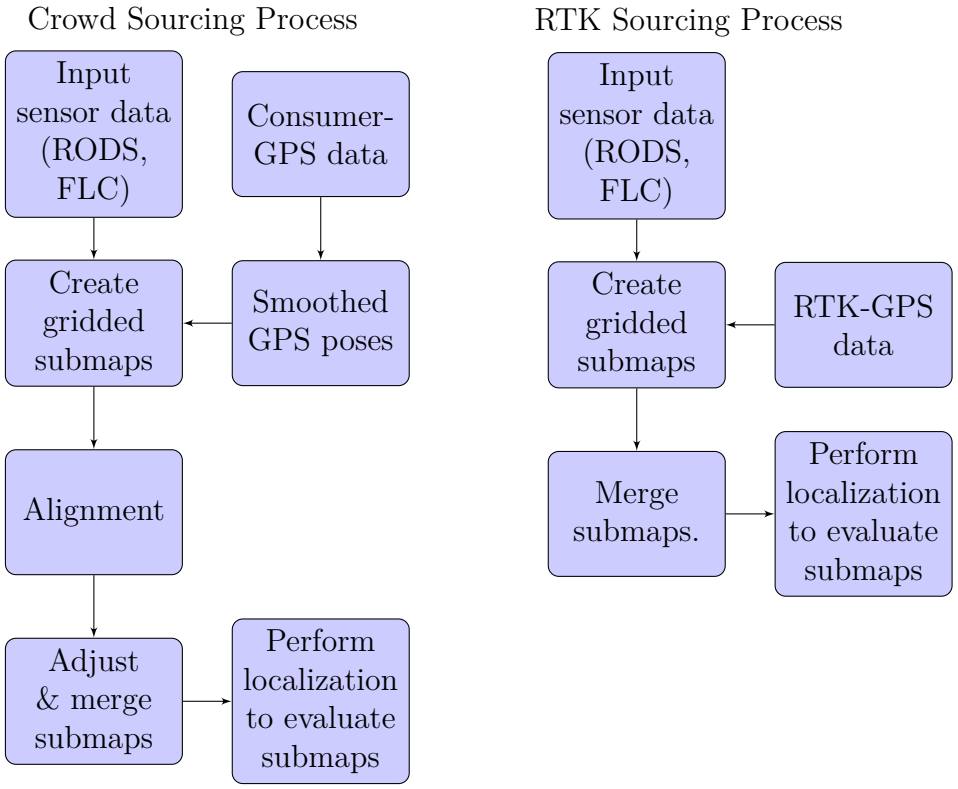


Figure 3.1: Process flow of the Crowd sourced map-creation compared to the original RTK map-creation. The smoothing and alignment are not necessary when using RTK.

The process could be considered to consist of three major parts; Firstly, consumer GPS poses are smoothed, and are used together with radar (RODS) and camera in-

formation (FLC) to create separate submaps from several logfiles each corresponding to roughly one minute of driving. Afterwards, due to the fact that the smoothed trajectories do not correspond to the true trajectories of the cars, submap-alignment is performed in order to achieve correct overlap. Lastly, localization using particle-filtering is performed on the generated map in order to evaluate performance. Each major part can furthermore be divided into smaller subtasks, which will be further discussed in Sections 3.2, 3.3 and 3.4.

3.2 Creation of submaps per logfile

The Square-Root Unscented Rauch-Tung-Striebel (URTS) smoothing algorithm is used to smooth the consumer GPS poses to better follow the shape of the RTK trajectory, however with a certain (unknown) offset compared to the true value of each pose, which will be compensated for in the alignment of the submaps. These smoothed poses are then used as groundtruth for the map-creation algorithm, which creates gridded submaps from RODS and FLC detections. This transforms the detections from the initial coordinate system relative to the axis of the vehicle to a local coordinate system centered around Gothenburg. The submaps are created from available sensor data from several vehicles gathered from the Drive Me [43] route.

3.2.1 Smoothing of consumer GPS

As data for the whole time-span for a trajectory is known, smoothing of consumer GPS will improve the accuracy of estimation considerably compared to only performing filtering. This is done in order to reduce the noise in the GPS data and estimate a consistent trajectory. This section will in detail describe the implementation of the URTS smoother.

Initial state

The state vector

$$X = [X_{dist} \quad Y_{dist} \quad \theta \quad offset_x \quad offset_y \quad w \quad b]^T \tag{3.1}$$

consists of distance in x direction from a fixed position in GPS, distance in y direction from the fixed position in GPS, heading angle, offset of GPS in x direction, offset of GPS in y direction, yaw rate bias and wheel speed scaling factor respectively. The first five states can be visually illustrated according to Figure 3.2.

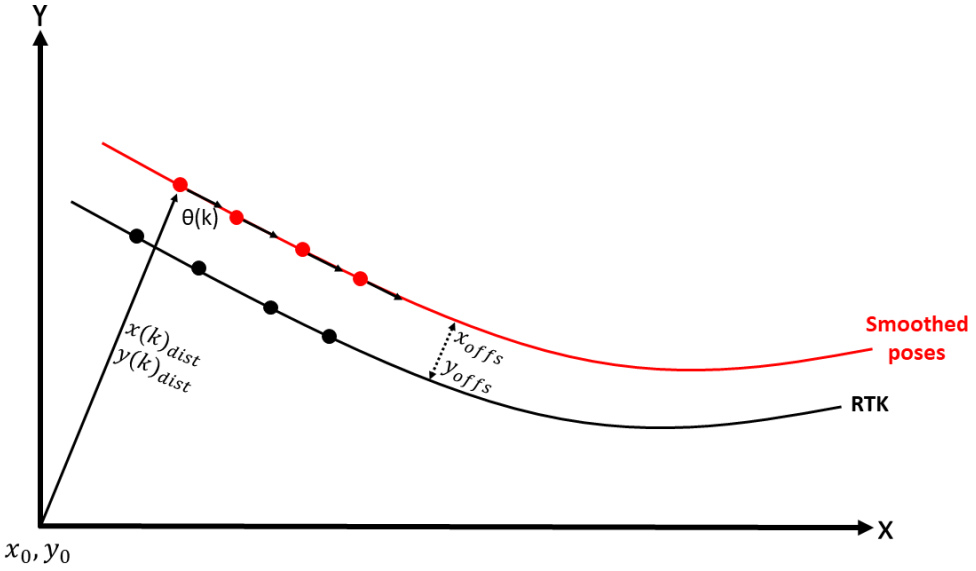


Figure 3.2: Illustration of the first five smoothing-states, where the red line corresponds to GPS poses to be smoothed and the black the RTK trajectory (not used for estimation).

The initial state of the smoothing is given according to the first GPS sample and first valid yaw, so X_{dist}, Y_{dist}, $offset_x$, $offset_y$ and yaw rate bias are set to 0, θ is given as yaw_0=$tan(\frac{Y_{dist}(valid)}{X_{dist}(valid)}$ and the wheel speed scaling factor is set to 1, thus X_0 can be written as:

$$\hat{X}_{0|0} = [0 \quad 0 \quad yaw_0 \quad 0 \quad 0 \quad 0 \quad 1.0]^T. \tag{3.2}$$

The initial square-root covariance S_0 is chosen as

$$S_0 = \sqrt{P_0} = diag([5 \quad 5 \quad 0.3 \quad 20 \quad 20 \quad 0.01 \quad 0.01]). \tag{3.3}$$

Sigma points and weights

Since the dimension of state is $n = 7$, 14(2n+1) sigma points in SR-UKF would be calculated according to Equation (2.11) in order to generate the measurement prediction at time $k \in \{1, 2, 3..., N\}$. $S_{k-1,j}$ is set as the j:th column of the matrix S_{k-1}, and γ which determines the spread of the sigma points is set as $\gamma = \sqrt{\frac{n}{1-\frac{1}{n}}}$.

The corresponding weights are computed according to Equations (2.12), (2.13) and (2.14), thus the scalar weights $W_j^{(m)}$ and $W_j^{(c)}$ are given as

$$W_0^{(c)} = W_0^{(m)} = \frac{1}{n}, \tag{3.4}$$

$$W_j^{(m)} = W_j^{(c)} = \frac{1 - W_0}{2n}. \tag{3.5}$$

.

Time update and measurement update

As sigma points and their weight are obtained, the predicted states $\hat{X}_{k|k-1}$ can be computed according to Equations (2.15) and (2.16). The noise R is set to 0, thus the predicted Cholesky factor $S_{k|k-1}$ and $\hat{Y}_{k|k-1}$ can be obtained according to Equations (2.17), (2.18), (2.19) and (2.20).

There are two measurement models that are used in this project. One is using consumer GPS measurements when a new sample is available, the other is using a yaw rate bias measurement if the car is standing still. When consumer GPS measurement is available, the measurement vector can be written as

$$Y_{k|k-1} = \begin{bmatrix} x_{gps} - x_0^{gps} \\ y_{gps} - y_0^{gps} \end{bmatrix}, \tag{3.6}$$

where x_{gps} and y_{gps} represents the position in x and y direction with respect to GPS. x_0^{gps} and y_0^{gps} are the initial point of the vehicle with respect to GPS. When the vehicle is standing still, the measurement will instead be the yaw rate bias which can simply be written as $Y_{k|k-1} = \omega$.

Once either of those measurements is available, the estimated state sequence is calculated and the smoothing is generated as according to SR-UKF and smoothing algorithm (see Algorithm1 and Section 2.3.3). The processing noise S_R is set to 0.

3.2.2 Smoothing evaluation

After smoothing, it is easy to get the estimation samples and the corresponding RTK samples. Since the number of samples can be reorganized to be the same for smoothed GPS and RTK GPS the x,y and heading-errors and x,y-drifts can be computed as in Equations (3.7) and (3.8) respectively. It is worth noting how the drift is defined, which will serve as a metric of how well the smoothed trajectory follows the shape of RTK, without considering actual offset-error. Here X_{SGPS} stands for the estimated sequence of smoothed GPS poses, X_{RTK} is the sequence of RTK GPS poses, θ_{SGPS} and θ_{RTK} represent headings of smoothed GPS and RTK GPS respectively, N is the length of samples and N_A is a chosen number of samples ahead in order to locally determine the drift over several seconds. For planning purposes, it is usually of importance to locally determine the vehicle's position around 10 seconds in advance, thus N_A is chosen to correspond to 10 seconds in this implementation.

$$X_{err} = X_{SGPS} - X_{RTK}, \tag{3.7a}$$

$$Y_{err} = Y_{SGPS} - Y_{RTK}, \tag{3.7b}$$

$$\theta_{err} = \theta_{SGPS} - \theta_{RTK}. \tag{3.7c}$$

$$X_{drift} = (X_{SGPS} - X_{RTK})_{(1+N_A):N} - (X_{SGPS} - X_{RTK})_{1:(N-N_A)}, \tag{3.8a}$$

$$Y_{drift} = (Y_{SGPS} - Y_{RTK})_{(1+N_A):N} - (Y_{SGPS} - Y_{RTK})_{1:(N-N_A)}. \tag{3.8b}$$

It is of interest to find out under which circumstances the smoothing performs well and when the performance is unsatisfactory. Using the previously mentioned metrics, it is possible to define four scenarios where the smoothing can be considered as invalid. The first takes the x,y-errors between smoothed GPS and RTK GPS into consideration. The smoothed trajectory should not be more than a couple of meters away from the actual trajectory. If either X_{error} or Y_{error} is above a certain value, the logfile can thus be considered as containing invalid smoothing.

The second and third scenarios considers the drifts between smoothed GPS and RTK GPS. By calculating the absolute mean of the drifts for a certain logfile, it is possible to determine if the smoothing is continuously diverging from the actual trajectory. It is also possible to detect peaks in the drifts, meaning large positional jumps which also suggests invalid smoothing performance.

The fourth scenario is to consider the heading-error between smoothed GPS and RTK GPS. When these errors are greater than a certain value, the corresponding logfile will be regarded as containing invalid smoothing.

3.2.3 Generating local occupancy grids

The process of fusing sensor information from RODS or FLC together with GPS to create individual grid-submaps per logfile is identical for both crowd-sourced maps and RTK maps. The only difference comes in the form of using different pose-estimations, as well as how merging of said maps is performed (and offset-adjustment in the case of crowd-sourced maps). For each area, the submaps will be separated according to detections originating from the three sensors; front RODS, rear RODS and FLC. The reason for separating RODS detections is due to the Doppler-shift effect commonly encountered in radars, meaning detections from each directions will look different and thus harder to combine.

The first step in this process is to transform the vehicle's ego-poses in the groundtruth-data to a local Cartesian coordinate system. The raw data from the GPS has been post-processed to be given in SWEREF99 latitude and longitude instead of the global coordinate system, which is then transformed to the local Cartesian system using the local meridian and origin, resulting in a coordinate system with origin in Slottskogen, Gothenburg, and x-axis aligned towards east and y-axis along north.

The RODS and FLC detection data (hits) have already been pre-processed to be given as x and y coordinates as seen from the ego-vehicle frame with origin at the rear-axis of the car. Thus, for each sample of the groundtruth, using Equation (2.32) together with the current ego-heading of the vehicle it is simple to express these detections in the local coordinate system. Thus these detections can be assigned to the corresponding cells in the grid-segment(s) describing the area in which detections are found, Figure 3.3 gives an illustration of this process. Since the cells in the grid have a certain resolution, it is not possible to make out shapes of objects that are

below this resolution when building the map. If the detections span an area that has not yet been mapped, a new segment is initialized. If not the detections are added to the pre-existing segment.

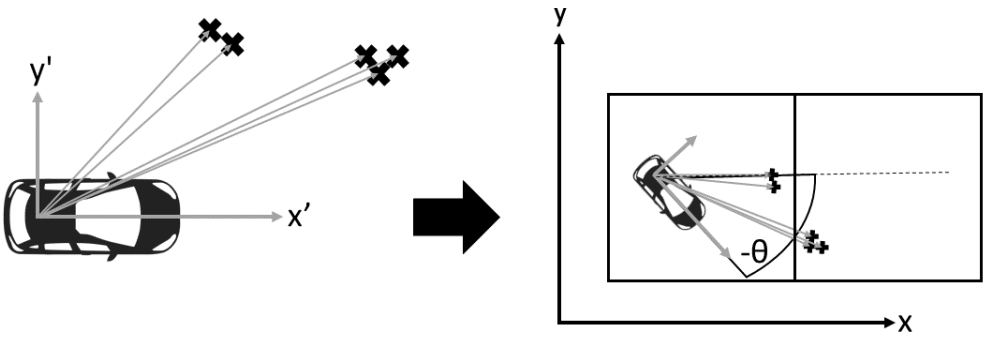


Figure 3.3: When assigning detections to submaps, the detections are transformed from the vehicle frame to the local frame and assigned to a segment.

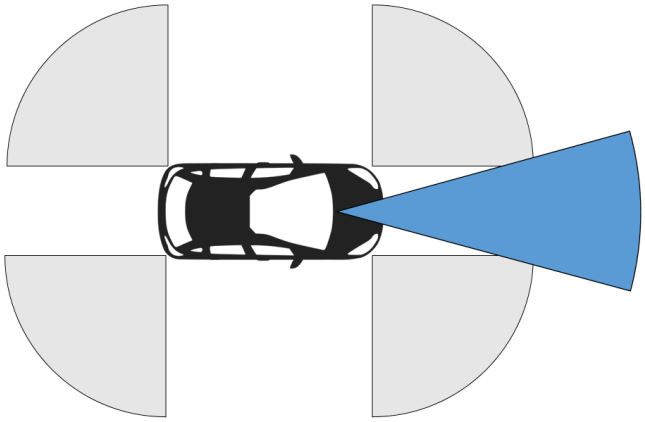


Figure 3.4: In order to determine which areas that are affected by a sensor sweep (a look), a sensor model is used.

When gathering the looks of each segment, a sensor model is used in order to determine which area that is spanned by the reach of the sensors, see Figure 3.4 for an illustration of this model. Thus for each sample, the positions of the cells that have been "looked at" can be found relative to the ego-vehicle frame, and can be transformed into the local coordinate system and assigned to a segment using the same principle as above. At the next sample, the car is at a new position and thus covers new cells in the sensor-span as well as previously seen cells, as can be illustrated in Figure 3.5.

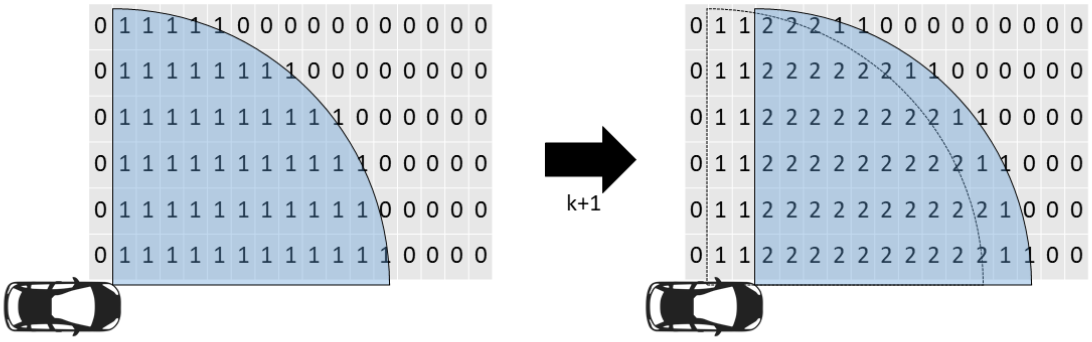


Figure 3.5: Illustration of how the looks of a segment are counted. At the next sample, the car has moved and is thus seeing other cells.

3.3 Alignment using Graph-SLAM

The alignment is implemented through a Graph-SLAM approach, as previously mentioned. Instead of representing nodes as a robot's poses, the nodes instead symbolize the sought alignment for each submap in order to achieve overlap. The problem can be formulated as setting up and solving a set of linear equations, i.e. a linear system, which will be further explained in the following subsections.

3.3.1 Graph formulation

The problem of finding the correct offset for each submap can be represented and illustrated as a graph with the structure according to Figure 3.6. Each node corresponds to the offset for a submap generated from a certain logfile, and is connected to other nodes by several edges. These consist of adjacency constraints fixing neighbouring submaps generated from the same logfiles to each other, as well as constraints expressing relative offsets between submaps of the same segment from different logfiles.

Expressed according to probability theory, the problem can be formulated as finding the maximum probability of observing a certain offset-configuration given measurements, i.e.

$$\arg\max_{x_{1:N}} \; p(x_{1:N}|y_{1:M}), \tag{3.9}$$

where $x_{1:N}$ are the sought offsets for N submaps, and $y_{1:M}$ are the observed offset measurements from M submap-pairs as well as adjacency measurements. How these measurements are obtained is explained further in Section 3.3.2.
According to Bayes' Theorem, the probability above can instead be expressed as

$$\arg\max_{x_{1:N}} \; p(x_{1:N}|y_{1:M}) \;\; = \;\; \arg\max_{x_{1:N}} \; \frac{p(y_{1:M}|x_{1:N})p(x_{1:N})}{p(y_{1:M})}. \tag{3.10}$$

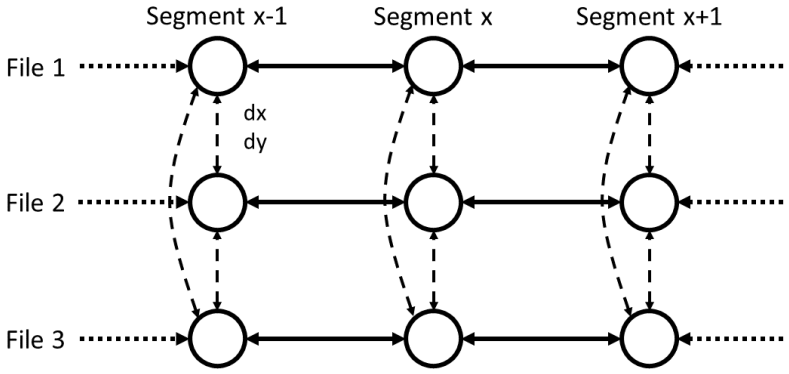


Figure 3.6: Graph-representation of the SLAM-problem of this thesis. Each submap is connected to their neighbours (here illustrated in 1D, in reality the neighbourhood is in 2D) by adjacency constraints (solid arrows) as well as constraints correlating submaps of the same segment (dashed arrows).

From prior experience regarding the offsets, $p(x_{1:N})$ is assumed to be normally distributed with an average offset of zero. Furthermore, $p(y_{1:M})$ does not depend on $x_{1:N}$ and can thus be omitted when estimating the maximum. This means that Equation (3.10) can be simplified to

$$p(y_{1:M}|x_{1:N})p(x_{1:N}), \tag{3.11}$$

resulting in an maximum likelihood estimation. In text this can be expressed as maximizing the probability of observing the measurements given a certain configuration, which intuitively is equivalent to maximizing the probability of observing the configuration given the measurements. The former can also be expressed as equal to maximizing the probability for observing the errors $e_{1:M}$ between the measurements and the correct configuration. Furthermore, if it is assumed the observed errors are normally distributed and conditionally independent, also assuming independent prior distribution, the probability to be maximized can be further expanded as

$$\begin{aligned} p(e_{1:M}|x_{1:N})p(x_{1:N}) &= \prod_{m=1}^{M} p(e_m|x_{1:N}) \prod_{n=1}^{N} p(x_n) = \\ \prod_{m=1}^{M} |2\pi\Sigma_m|^{-\frac{1}{2}} exp(-\frac{1}{2} e_m(x_{1:N})^T \Sigma_m^{-1} e_m(x_{1:N})) &\prod_{n=1}^{N} |2\pi\Sigma_n|^{-\frac{1}{2}} exp(-\frac{1}{2} x_n^T \Sigma_n^{-1} x_n), \end{aligned} \tag{3.12}$$

where Σ_m is the covariance of e_m and similarly Σ_n for x_n. This expression can be further simplified by assigning an error function $e'_n = x_n = x_n - 0$ (i.e. treat the prior-distribution as "measurements") and include the product of N prior-distributions together with the set of M measurements error to form the new measurement set M'. This means they can be expressed in the same product-sum, where only the error-functions differ. Maximizing this probability is equal to maximizing the logarithm of the expression, thus after taking the logarithm of Equation (3.12) and taking the previously mentioned simplification into account, the following expression is acquired:

$$\sum_{m=1}^{M'} log(|2\pi\Sigma_m|^{-\frac{1}{2}}) - \frac{1}{2}\sum_{m=1}^{M'} e_m(x_{1:N})^T\Sigma_m^{-1}e_m(x_{1:N}). \tag{3.13}$$

Since the first term is constant and not dependent on x_{1_N}, the maximum of Equation (3.13) with respect to the configuration will thus be acquired by minimizing the right-most term in the expression. This results in being able to express the problem as a least-squares problem, i.e. minimizing the sum S where

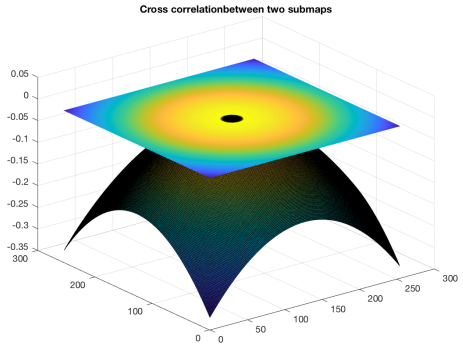
$$S = \sum_{m=1}^{M'} E_m^T E_m. \tag{3.14}$$

Setting $E_m = \Sigma_m^{-\frac{1}{2}} e_m$ would result in the sum expressed in Equation (3.13), where $\Sigma_m^{\frac{1}{2}}$ is defined as the matrix square-root of the inverse covariance matrix. Since the covariance matrix is symmetrical the transpose corresponds to itself, thus $\Sigma_m^{-\frac{1}{2}T}\Sigma_m^{-\frac{1}{2}} = \Sigma_m^{-1}$. Being able to formulate the problem as a least-squares forms the basis of the Graph-SLAM method used in this thesis. As will be explained further on in Section 3.3.3, the resulting least-squares problem will also be linear, meaning it can easily be solved by setting up and solving a linear system using three different measurements representing the spatial constraints mentioned at the start of this section.

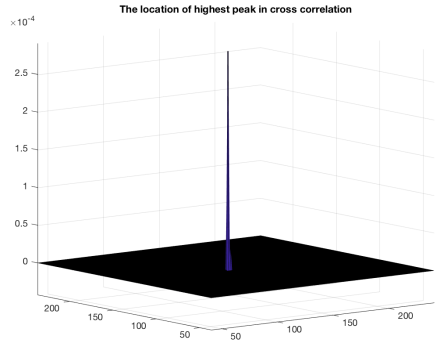
3.3.2 Submaps cross-correlation

Before solving the linear system, the measurements have to be obtained. The generated submaps are first ordered according to which logfile the submaps were generated from. This allows the algorithm to easily find matches for a certain segment describing the same area over logfiles. Then cross-correlation is performed for each submap-pair sharing the same segment. The cross-correlation represents the degree of similarity between two submap histograms, where the maximum intensity peak of the cross-correlation would refer to maximal overlap of the histograms. Thus, by finding the position of the peak the relative offset between the two submaps can be acquired.

One major question is how a valid peak can be found and defined. One cross-correlation may yield several peaks, should there be areas that partly overlap. To find each peak, the Matlab function *imregionalmax* [44] is used. This results in a logical matrix specifying the locations of peaks, which is multiplied element-wise to the cross-correlation matrix to yield the values of the peaks, Figure 3.7 shows an illustration of this process. In order to be certain that the offset generated from the cross-correlation is correct, cross-correlations can thereafter be considered valid or invalid according to the illustrations in Figure 3.8, which shows different 2D-representations of peaks found in the used data.

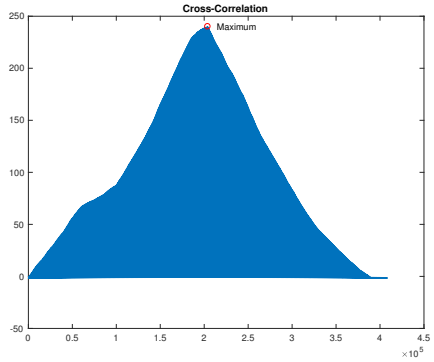


(a) Ideal cross correlation between two submaps.

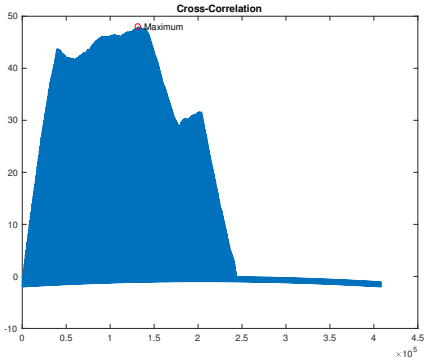


(b) The location of the highest peak in cross correlation.

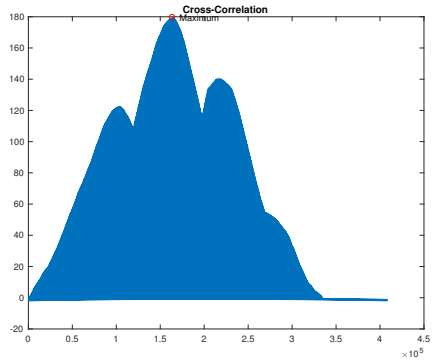
Figure 3.7: The maximum cross-correlation is used for deciding the offset between two submaps for the same region



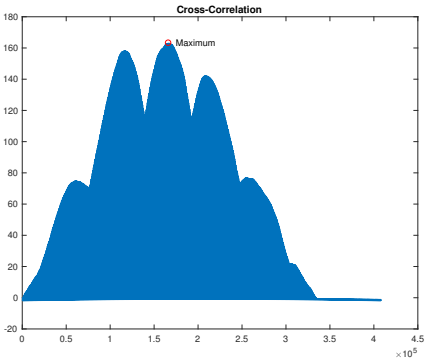
(a) Cross-correlation which only has one peak, which is valid.



(b) Cross-correlation with valid peak, but the information spread is low.



(c) Cross-correlation has several peaks, but only one valid peak.



(d) Cross-correlation has several peaks, but no peak can be considered as uniquely valid.

Figure 3.8: Subfigures (a) and (c) have practical cross-correlation peaks, whereas (b) has a less practical peak and (d) has no practical peak.

Peak hessian

As mentioned in Section 3.3.1, Equation (3.12), estimating the covariance of the measurement is of interest. Thus after a valid cross-correlation peak has been found, the hessian of the peak is calculated. The hessian is defined as

$$H = \begin{bmatrix} \frac{\partial^2 F_{x0,y0}}{\partial x^2} & \frac{\partial^2 F_{x0,y0}}{\partial x \partial y} \\ \frac{\partial^2 F_{x0,y0}}{\partial x \partial y} & \frac{\partial^2 F_{x0,y0}}{\partial y^2} \end{bmatrix}, \tag{3.15}$$

where $F_{x0,y0}$ is the intensity-value of the cross-correlation at the peak located at $(x0, y0)$. For maximum-likelihood estimations, the covariance matrix can then be estimated as the inverse of the negative hessian, i.e. $\Sigma = (-H)^{-1}$.

Special case for FLC

When cross-correlating detections from FLC maps, there is a need to take extra precaution when selecting the correct peak. The cross-correlation of FLC-submaps contains 3 peaks, as shown in Figure 3.9. The maximum peak for a FLC cross-correlation would correspond to an offset such that both lane-markings will overlap, whereas the two others corresponds to only 1 of the lanes overlapping. However, consider the case where the submaps were generated from cars driving in separate lanes. This means that immediately choosing the highest peak would result in the two lanes merging into one, resulting in an incorrect map as seen in Figure 3.10. Instead one would like to chose one of the lesser peaks that correspond to only one of the lane-markings overlapping. Or in the cases where three of more lanes are present, performing a cross-correlation between a left and right lane (with middle-lanes in between) would provide invalid offsets. This issue is not present when considering radar-submaps, since those detections span a larger area around the road and thus do not depend in which lane the car is driving.

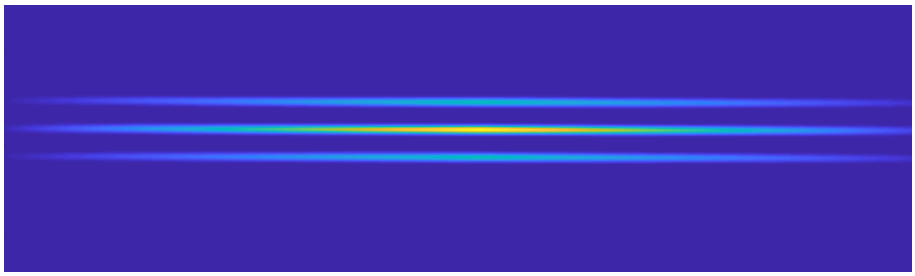
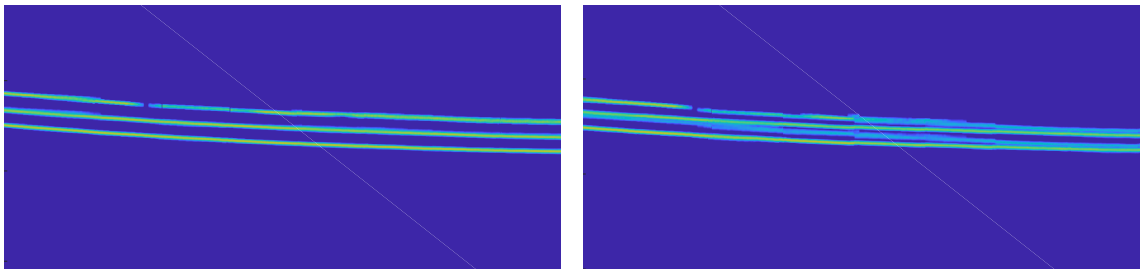


Figure 3.9: Cross-correlation of the FLC submaps, showing multiple peaks corresponding to the multiple lane-overlaps.



(a) For a certain stretch of road, there should be two lanes.

(b) Without using RODS measurements as prior information, the lanes attempt to merge into one.

Figure 3.10: Impact of not using RODS as prior information when selecting FLC peak.

To be able to completely solve the problem of accidentally merging lanes, there would be a need to assign in which lane the probing vehicle was driving when creating the submap. Currently the FLC submaps themselves do not contain any information that could be used to solve this independently from other information. Attempts such as assigning a lane-index depending on the type of lane-markings that the camera detects on either side of the car (dashed and solid lines) and only cross-correlate submaps with the same lane-index yielded unsatisfactory results, and did not always remove the lane-merging behaviour.

Instead, another approach is to use the offsets acquired from radar as prior information in order to choose the correct peak for the FLC cross-correlation. This has yielded more correct results, and is thus the preferred approach in this implementation. The procedure is as follows; First the corresponding cross-correlation for two submaps from the front or rear radar is found. If none is found, or if the found match from either radar-sensor is invalid, the FLC cross-correlation is immediately set to invalid due to lack of reliable prior information. Should valid radar-information be found, the FLC cross-correlation is smoothed using a Gaussian mask in order to reduce the amount of "false" intermediate peaks. After performing *imreagionalmax()* on the smoothed cross-correlation, all non-zero elements are kept and checked. The peak corresponding to the closest match when comparing its offset to the offset acquired from radar is chosen as the correct peak. If this peak is still not within 1 meter in either x or y direction, the cross-correlation is deemed as invalid.

This approach means, naturally, that a FLC map cannot be created without using radar information. Thus the ability to create a FLC map completely independent of radar can not be achieved using this implementation.

3.3.3 Linear system setup

Since the error in heading caused by the smoothing is assumed to be very low, all submaps can be regarded as approximately aligned with each other in terms of rotation. This means that the least-squares Graph-SLAM formulation can be solved as a weighted linear system, as opposed to a nonlinear system should the heading have a considerable error and thus needed to be rotationally adjusted. This results in a vastly simplified process of setting up the linear system and solving it. This system will be overdetermined, since there will be more measurements than submaps, meaning the linear system has to be solved according to Equation (3.16), where $\hat{x}$ denotes the estimation of the "best fit" solution to the system.

$$Ax \approx y \iff \min_x ||Ax - y||^2 \Rightarrow \hat{x} = (A^T A)^{-1} A^T y. \tag{3.16}$$

The general structure of the linear system is as follows. The element in the row m and column n of the matrix can be expressed as

$$A_{mn} = \frac{\partial E_m}{\partial x_n}, \tag{3.17}$$

meaning the partial derivative of the m:th error with regards to the offset x_n. The m:th element in the y-vector is the measurement y_m. There are three different measurements to consider, described further in the sections below. Thus the linear system could be illustrated as consisting of three different blocks each using separate expressions for the error E_m, of which the first consists in turn of three different sub-blocks depending on the sensor used to acquire the measurements (front radar, rear radar and FLC), as illustrated in Figure 3.11.

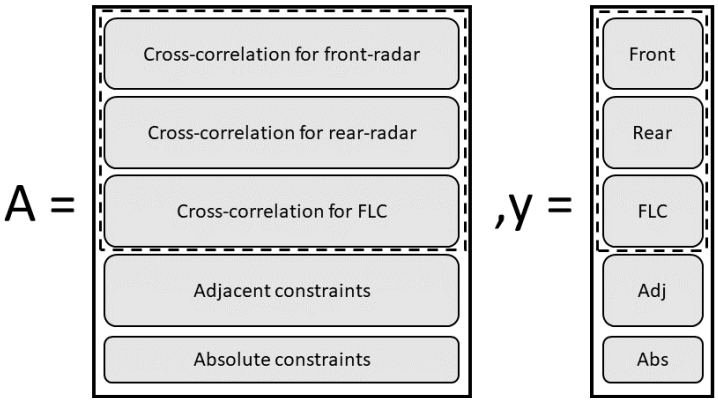


Figure 3.11: Visual representation of the linear system divided into blocks.

Cross-correlation offsets

For each sensor, the cross-correlation for the submap-pair i, j which is deemed as valid is included as an observation of the measurement m in the Graph-SLAM linear system. The error can be expressed as

$$E_m = \Sigma_m^{-\frac{1}{2}}(y_m - (x_i - x_j)), \tag{3.18}$$

which leads to matrix-elements on the format

$$A_{mn} = \frac{\partial E_m}{\partial x_n} \quad \Rightarrow \quad A_{mi} = -\Sigma_m^{-\frac{1}{2}} \ , \ A_{mj} = \Sigma_m^{-\frac{1}{2}}. \tag{3.19}$$

This means that each row of the system matrix will only contain two non-zero entries corresponding to x_i and x_j, resulting in a very sparse matrix.

In order to favor rows that contain observations that can be considered as more reliable, weights are introduced as expressed in Equation (3.20), which gives cross-correlation peaks with higher intensity $V_{xc,max,ij}$ (i.e. containing more information) a higher weight value, which is applied to all elements in the row. This is scaled relatively with the maximum intensity $V_{xc,max}$ observed for each sensor (since the intensity values differ between sensors), resulting in a scaling between 0 and 1. This is then scaled again with a design parameter D_n, where $n = 1, 2, 3$ for each sensor, that can be chosen depending on how observations from any of the three sensors should be considered as more reliable than the others. Setting this design parameter to zero would result in observations from this sensor being disregarded when finding the correct configuration of the offsets.

$$w_m = \frac{V_{xc,max,ij}}{V_{xc,max}} D_n, \quad n = 1, 2, 3. \tag{3.20}$$

Adjacency constraints

Apart from offset observations, additional constraints can be formulated to relate submaps which are describing neighbouring segments. The motivation is that submaps generated from the same logfile, and thus using the same estimated trajectory, should locally have the same GPS error meaning the offsets for submaps in the same neighbourhood should be roughly equal. For this, virtual measurements can be used, where all $y_{1:M} = 0$. Furthermore, since the measurements are assumed to be independent with equal spread in both x and y dimensions, the covariance matrix is assigned as the identity matrix. Thus, for two adjacent submaps (i,j), the linear system setup can be expressed as

$$E_m = \Sigma_m^{-\frac{1}{2}}(y_m - (x_i - x_j)) \ = \ (x_i - x_j), \tag{3.21}$$

which leads to matrix-elements on the format

$$A_{mn} = \frac{\partial E_m}{\partial x_n} \quad \Rightarrow \quad A_{mi} = -I \ , \ A_{mj} = I. \tag{3.22}$$

These constraints can also be scaled with a weighting factor to have a larger or smaller impact on the solution of the system, as in Equation (3.23). In this case, this weight is only expressed as a chosen design parameter D_4. Setting this parameter to zero would result in submaps being able to "float" around when aligning themselves according to observations, without regarding their positions according to their neighbours, which can yield discontinuous and undesirable results. However, setting this parameter to a very high value will result in a blurry map in segments further away from the areas with highest cross-correlation intensity. This is caused

by the adjacency constraints fixing these submaps according to the offset generated by the higher cross-correlation through a chain of connected neighbours, resulting in an incorrect offsetting since the GPS error might not be consistent from one end of a neighbour-chain to the other.

$$w_m = D_4. \tag{3.23}$$

Absolute constraint

In order to prevent the linear system from becoming rank-deficient, i.e. the having several configurations that serves as solutions to the linear system, the solution needs to be spatially fixed. Preferably, these positions should be as close to the true positions of the submaps as possible. This can be achieved by using the previously mentioned knowledge of the prior distribution. What this means in practice is averaging the positions of the submaps by adding a low-weighted constraint that each submaps should not be moved. The error equation, assuming the information spread is equal in x and y direction resulting in an identity covariance matrix, is thus simply

$$E_m = x_i, \tag{3.24}$$

which leads to

$$A_{mn} = \frac{\partial E_m}{\partial x_n} \quad \Rightarrow \quad A_{mi} = I. \tag{3.25}$$

The weight for these system-rows is again chosen as a design parameter D_5 according to Equation (3.26), which is assigned a very low value.

$$w_m = D_5. \tag{3.26}$$

An alternative method for fixing the position of the submaps, without assuming a prior distribution, would be to instead fix an arbitrarily chosen submap (i.e. same error-function as above but for one row) and put the corresponding weight as a very high value. As far as the resulting map itself is concerned, these two methods do not differ significantly. The only difference is that the whole map might be offset-shifted compared to each-other, especially if the submap that is arbitrary chosen should be originating from a smoothed trajectory further away from the other trajectories. In theory this offset of the map is not an issue, since if all vehicles in an area localize based on the same map this offset will influence all vehicles equally. However, should the map have a too large offset compared to the real map, in practice this might pose a problem when initializing the particle filter used in this thesis for localization, causing it to have an increased time until convergence to a proper trajectory. Thus the authors argue that using an averaging absolute constraint is preferable, abiding by the assumption of normally distributed offsets.

3.3.4 Merging the map

After the correct offset for each submap has been found, the only thing remaining before the final map is completed is to merge the various submaps into one submap for each grid. The process is as follows; For each submap that is to be

offsetted, a local and empty neighbourhood of adjacent segments is created around the submap-segment. Should the hits/looks of the submap be moved outside of the segment-boundary after offsetting, it will be assigned as hits/looks of the neighbouring segment. Figure 3.12 shows an illustration of this process. After the offset, the neighbourhood is split up, and the hits and looks of non-empty segments are added to one submap that contains the summation of all separate submap-segments. This summation of total hits and looks serves to separate temporary detections that are not observable in the majority of the submaps. For example, should a car be parked along the road, it will be included as a stationary detection for each car that passes that area at that time. At any other day, the parked car might not be present, thus the passing vehicles do not detect an object at said location. These kind of false positives thus results in a lower occupancy-likelihood by the hits and looks summation.

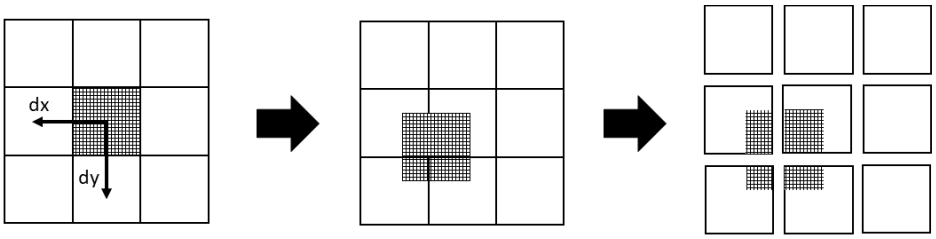


Figure 3.12: Illustration of the neighbourhood that the detection-grid is moved within. After a grid has been moved, the neighbourhood is split up into separate submaps.

After merging has been performed, each segment is filtered by only keeping occupancy cells that contain more than a minimum amount of looks, minimum amount of hits and minimum occupancy likelihood. The total map is then saved in a sparse format where each segment is represented using three vectors; x-indexes of the kept cells, y-indexes and the likelihood of said cells.

3.4 Map evaluation using particle filtering

After the map has been generated, and without further work, it is only possible to evaluate the map visually through observing the resulting sensor maps. This is not a fitting evaluation method for comparison. Thus a localization algorithm using particle filtering is implemented, to yield an evaluation score based on the resulting estimation of the trajectory of cars driving in the map. The following sections will describe the implementation of said particle filter.

3.4.1 Particle initialization

The state-vector of particles is given as

$$X = [X^{(i)} \quad Y^{(i)} \quad \phi^{(i)}]^T, \tag{3.27}$$

where $X^{(i)}$ and $Y^{(i)}$ denotes the longitudinal position and lateral position of particles with respect to the consumer-GPS respectively, ϕ is the heading of particles and i=1,2,.....,N. The number of particles N is set to 2000, since a larger amount of particles result in a more accurate estimation.

The initial particles are obtained by drawing N samples from the first distribution of the state according to SIS algorithm (see Algorithm 3). The Initial weights are set to $W_0^{(i)} = \frac{1}{N} = 0.0005$.

3.4.2 Particle prediction

The motion of the particles is estimated using a discrete coordinated turn model and the Euler method. Thus, the predicted state of particles can be obtained as

$$\begin{bmatrix} X_{k|k-1}^{(i)} \\ Y_{k|k-1}^{(i)} \\ \phi_k^{(i)} \end{bmatrix} = \begin{bmatrix} X_{k-1}^{(i)} + T_s(V_{lon,k-1}cos(\phi_{k-1}^{(i)}) + V_{lat,k-1}sin(\phi_{k-1}^{(i)})) \\ Y_{k-1}^{(i)} + T_s(V_{lon,k-1}sin(\phi_{k-1}^{(i)}) + V_{lat,k-1}cos(\phi_{k-1}^{(i)})) \\ \phi_{k|k-1}^{(i)} + T_s\omega_{k-1}^{(i)} \end{bmatrix} + q_k, \tag{3.28}$$

where $V_{lon,k}$ is the longitudinal speed of the vehicle, $V_{lat,k}$ is the lateral speed, $\omega_k^{(i)}$ is the yaw rate, T_s is the sampling time and q_k is randomly generated noise for the motion.

3.4.3 Measurement likelihood

The measurements used for updating are longitudinal position (m_{Lon}) and latitudinal position (m_{Lat}) of detections in the sensor map. In addition, the weights of uncertainty about detection W_{uc} are computed, where the distance to detections, velocity and amplitude are three main factors for determining uncertainty and the value of those weights. Thus, the longitudinal positions ($P_{lon}^{(j)}$) and lateral positions ($P_{lat}^{(j)}$) used for the measurement update are calculated as

$$\begin{bmatrix} P_{lon}^{(j)} \\ P_{lat}^{(j)} \end{bmatrix} = \begin{bmatrix} m_{on}^{(j)} W_{uc}^{(j)} \\ m_{Lat}^{(j)} W_{uc}^{(j)} \end{bmatrix}, \tag{3.29}$$

where $j = 1, 2, ..., M$ and M is the number of detections that are given by the sensor map. Then, by using these measurements from the map and the predicted particles, the measurement update will be computed as

$$\begin{bmatrix} X_{k|k}^{(i)} \\ Y_{k|k}^{(i)} \end{bmatrix} = \begin{bmatrix} X_{k|k-1}^{(i)} \\ Y_{k|k-1}^{(i)} \end{bmatrix} + R \begin{bmatrix} P_{lon}^{(j)} \\ P_{lat}^{(j)} \end{bmatrix}, \tag{3.30}$$

$$R = \begin{bmatrix} cos(\phi_k^{(i)}) & sin(\phi_k^{(i)}) \\ -sin(\phi_k^{(i)}) & cos(\phi_k^{(i)}) \end{bmatrix}. \tag{3.31}$$

Thus the measurement likelihood $p(y_k|x_k^{(i)})$ is determined by these particle poses according to the SIS algorithm.

3.4.4 Weight update and normalization

Weights will be updated as

$$W_k^{(i)} \propto W_{k-1}^{(i)} p(y_k|x_k^{(i)}), \tag{3.32}$$

where i=1,2....N at time k and $p(y_k|x_k^{(i)})$ is the likelihood of current measurement. Then the weights are normalized as

$$W_k^{(i)} = \frac{W_k^{(i)}}{\sum_i^N (W_k^{(i)})}. \tag{3.33}$$

3.4.5 Resampling

As previously mentioned, current measurement likelihood and weights would be obtained after implementing the SIS algorithm (see Algorithm 3). However, most weights $W_k^{(i)}$ have a very small value. It is therefore necessary to perform the SIR algorithm (see Algorithm 3). New independent samples are drawn and all the weights are set to $W_0^{(i)} = \frac{1}{N}$. Finally, multiple copies of high probability particles would be used for next iteration.

3.4.6 Evaluation of localization estimates

When evaluating the map, the crowd-sourced map might not necessarily be aligned with the true RTK map, but instead shifted with a certain offset, as briefly discussed in Section 3.3.3. This offset might not necessarily be the same for all areas of the map, but should locally coincide. Thus an evaluation metric that does not concern itself with the actual offset between the estimated trajectory and the true trajectory, but rather compares the shapes between the two is desirable. This means that the evaluation approach used for evaluating the smoothing results used in Section 3.2.2 can again be used to also evaluate the localization results in terms of drift and heading-error. This drift is calculated much like in Equation (3.8), also here an ahead-time of 10 seconds is used to determine local drift.

However, in this case, the equation needs to be slightly adjusted to account for allowing the estimation to converge to steady state after initialization. This convergence time is depending on how far from the map the first pose of the estimation is initialized. When localizing based on RTK maps and RTK groundtruth, the convergence time is close to 0 since the initial pose is the correct position in the map. This is not true when instead initializing with consumer-GPS poses, and more so when using the crowd-sourced map (unless the initial pose by small chance appear in the correct position in the map). Since this convergence is not a product of the map performance itself, but rather the localization, it might be favorable to only calculate the drift after a chosen start-sample N_S. Thus Equation (3.8) can be modified to Equation (3.34), where X_{Est} and Y_{Est} are the x respective y coordinates of the estimated trajectory. The absolute errors are still calculated as in Equation (3.7), where the heading-error is still of interest.

$$X_{drift} = (X_{Est} - X_{RTK})_{(N_S+N_A):N} - (X_{Est} - X_{RTK})_{N_S:(N-N_A)}, \tag{3.34a}$$

$$Y_{drift} = (Y_{Est} - Y_{RTK})_{(N_S+N_A):N} - (Y_{Est} - Y_{RTK})_{N_S:(N-N_A)}. \tag{3.34b}$$

When using several logfiles to evaluate the map, it gets quickly difficult to illustrate the results in a viable way. Preferably, one would like to instead evaluate the results based on a single metric rather than plots of the errors and drift for each logfile. One way to do this is to simply take the mean of the absolute drift (to avoid cancellation) for each logfile l, and then again the mean among all logfiles. Mathematically this can be expressed as

$$\overline{X}_d = \frac{1}{L}\sum_{l}^{L}\left(\frac{1}{N}\sum_{n}^{N}|X_{drift}(n)|\right),, \tag{3.35}$$

where L is the number of logfiles used, and N is the length of the drift-vector calculated using Equation (3.34).

Since the particle filtering has some randomness in the localization estimates (due to the generation and weighting of random particles), it might furthermore be desirable to run several evaluations based on identical logfiles in order to find the correct mean. This is however very time consuming, and thus most results presented regarding localization results in the following section have been created using the mean of 18 identical drift-evaluations on 17 logfiles, resulting in 306 samples of the mean.

4

Results

This chapter will present the results obtained throughout the project regarding smoothing performance, alignment performance and localization evaluation.

4.1 Smoothing and RTK comparison

In this section the results achieved from the evaluation of the smoothing performance will be presented, both visually and through the computed x,y-drifts and heading-error.

4.1.1 Visual comparison

Figure 4.1 shows a plot of the resulting trajectories for the whole AD route. A small area within the black circle of the AD route is picked randomly to demonstrate the poses of consumer GPS, smoothed consumer GPS and RTK GPS in more detail. There are furthermore four noticeable areas within solid blue circles and dashed red circles in the whole AD route, two underground tunnels named Lundbytunneln and Gnistängstunneln, and two complex road conditions called "Mark_1" and "Mark_2".

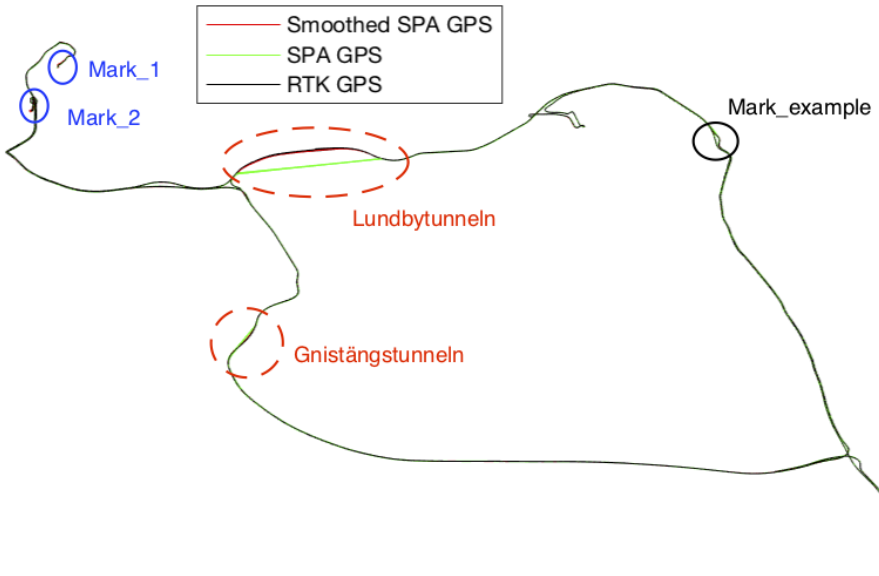


Figure 4.1: Trajectories for the whole Gothenburg-AD route. The red lines represents the smoothed poses, green raw consumer GPS poses and black RTK poses.

It can be seen in Figure 4.2 that the smoothed trajectory follows the shape of the RTK and is more precise than consumer GPS in the small area mentioned above. From an overall perspective, the smoothed trajectory will indeed follow the shape of the RTK also in the rest of the route.

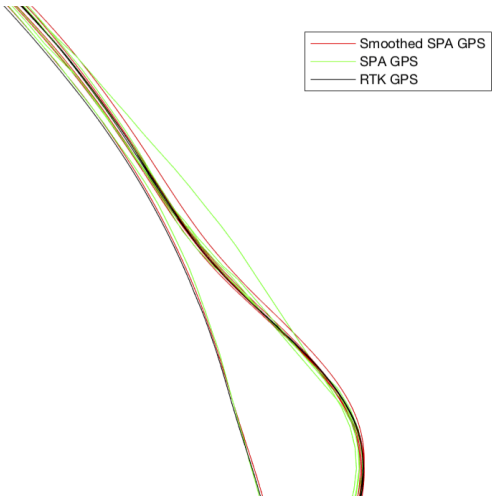
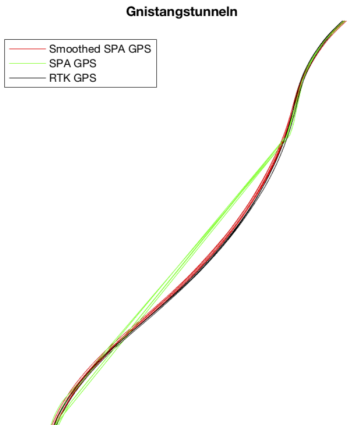
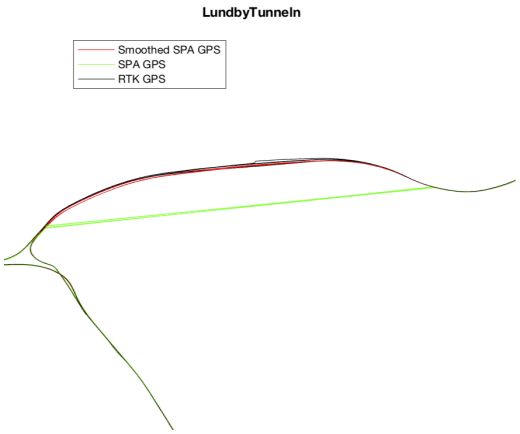


Figure 4.2: Trajectories at the area called "Mark_example". The red line represents the smoothed consumer GPS poses, green raw consumer GPS poses and black RTK GPS poses.

For the case of the underground tunnels, as shown in Figure 4.3, the trajectories have a similar behavior. Figure 4.3a shows smoothed poses for a road section inside Gnistängstunneln and Figure 4.3b for Lundbytunneln. As can be seen, the raw consumer GPS is unable to provide any measurements inside the tunnels, but the smoothed trajectory is still able to follow the shape of the RTK poses, although with higher offsets.



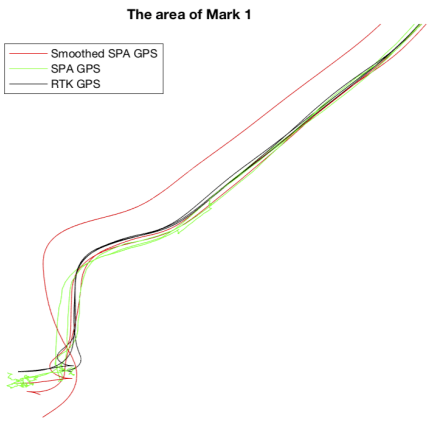
(a) Trajectories in Gnistängstunneln along the AD route.



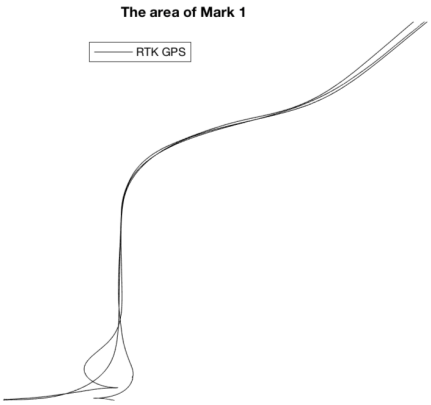
(b) Trajectories in Lundbytunneln along the AD route.

Figure 4.3: Trajectories of consumer-GPS, RTK and smoothed consumer inside two underground tunnels along the AD route.

Figure 4.4 and 4.5 shows two areas where it has been discovered that the smoothing regularly performs badly. However, those areas also have poor RTK performance, where the trajectories provide different shapes for different runs. It is also noteworthy that the cars are driving with a low speed in these areas, which could help to explain the behaviour. These areas can thus be detected and filtered away when the car is driving at speeds below a certain limit.

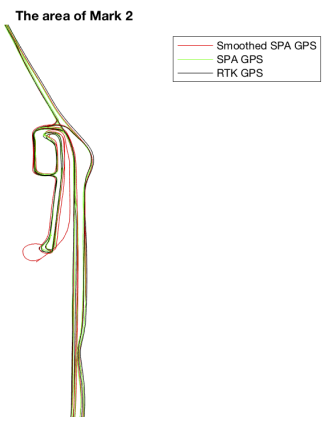


(a) Smoothed consumer poses of the area "Mark_1".

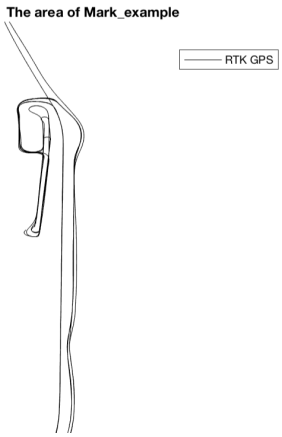


(b) RTK GPS poses of the area "Mark_1".

Figure 4.4: Plots of poor smoothing performance for the area "Mark_1".



(a) Smoothed consumer poses of the area "Mark_2".



(b) RTK GPS poses of the area "Mark_2".

Figure 4.5: Plots of poor smoothing performance for the area "Mark_2".

4.1.2 Evaluation results

As mentioned in Section 3.2.2, the error drifts and heading errors between smoothed consumer GPS and RTK GPS are computed after smoothing 2567 logfiles, which was the total amount of DriveMe data available at the time of this thesis. As shown in

Figure 4.6, the error drifts in X and Y directions from 19 example-logfiles range from -2.5 meters to 2 meters. Even though the error drifts are greater at the beginning, the mean is around 0 meters and tend towards stability. This means that poses in smoothed consumer GPS does indeed have a (roughly constant) local offset to RTK GPS. It is however interesting to note that error drifts from a specific date, 2016-09-14, shown in Figure 4.7 provide a very strange and "jumpy" behaviour.

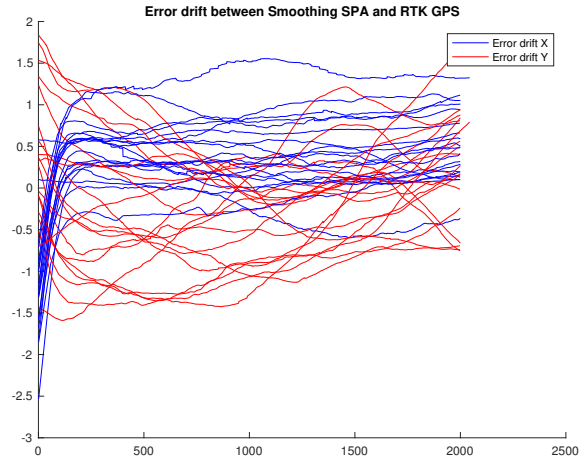


Figure 4.6: Error drifts between smoothed and RTK GPS from 19 example-logfiles.

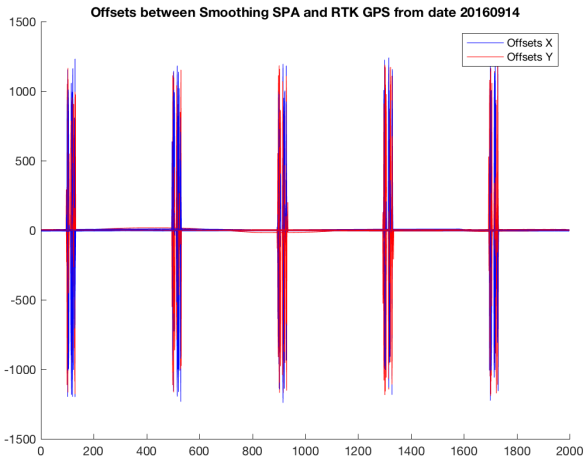


Figure 4.7: Error drifts between smoothed and RTK GPS from the date 2016-09-14.

In Figure 4.8 is is possible to see that the heading errors from 19 example-logfiles fluctuate slightly and is roughly zero-mean. Thus heading angles of the smoothed consumer GPS are indeed comparable to that of RTK GPS. As mentioned previously, heading errors from the date 2016-09-14 shown in Figure 4.9 again show a strange behavior.

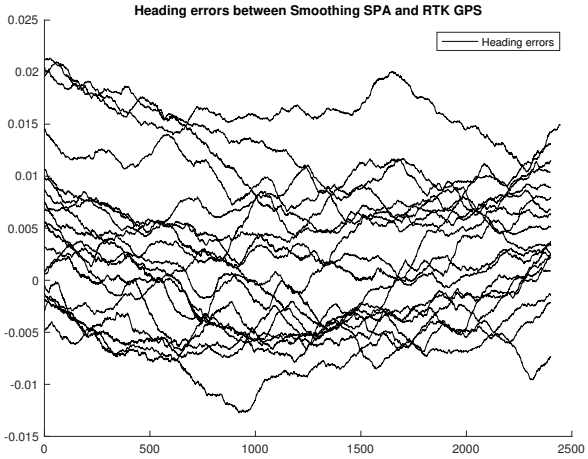


Figure 4.8: Heading errors between smoothed and RTK GPS from 19 example-logfiles.

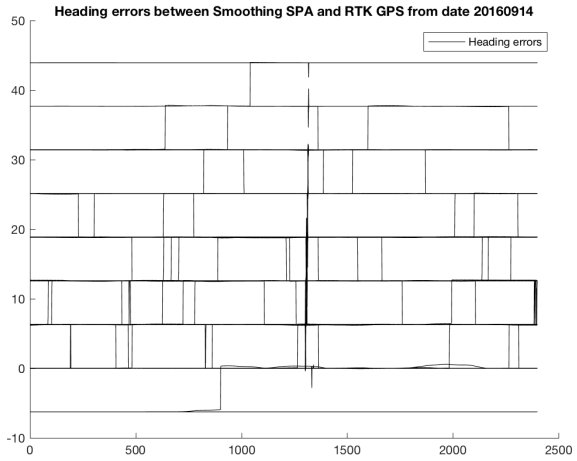
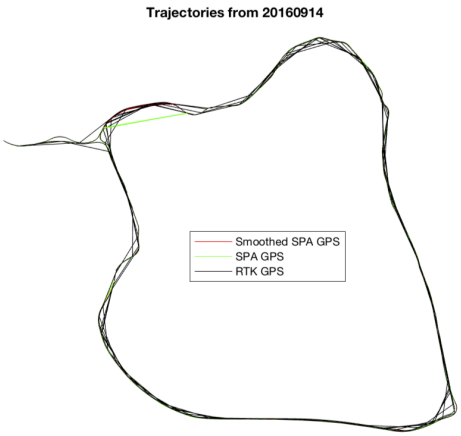
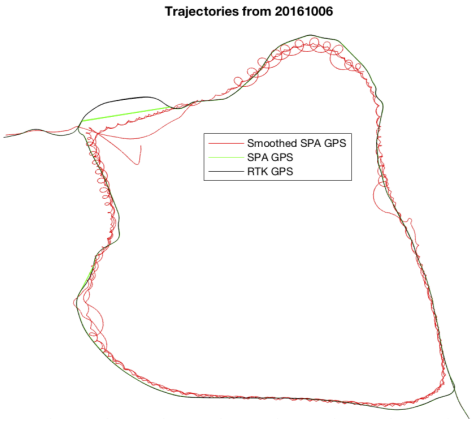


Figure 4.9: Heading errors between smoothed and RTK GPS from the date 2016-09-14.

The total logfiles mentioned above are collected from different days of various months. What is noteworthy is that logfiles with large error drifts in x and y-directions are mostly from the same dates, for example the previously mentioned 2016-09-14, but also 2016-10-06 and 2016-10-12. However, dates 2016-09-14 and 2016-09-27 contain most of the logfiles which have large heading errors. Thus bad performance of smoothing is not only caused by the low accuracy of consumer GPS, but also arise from the noise introduced when collecting raw data. As shown in Figure 4.10a, the poses of RTK GPS varies for different runs, while the poses of smoothed consumer GPS are quite stable. This means that unstable RTK poses may lead to bad smoothing evaluation, when the smoothing in reality has satisfying performance. Figure 4.10b shows strange smoothing poses which may be caused by "weird" measurement data.



(a) Trajectories from the date 2016-09-14



(b) Trajectories from the date 2016-10-06

Figure 4.10: Plots of poor smoothing performance from the dates 2016-09-14 and 2016-10-06.

Table 4.1 shows results of smoothing performance for all available logfiles, including the special dates of poor performance. The mean of the error drifts in x-direction is about 0.0440 meters and in y-direction is 0.3440 meters. These error drifts could somewhat already be considered as acceptable. However, the mean of the error drifts decrease greatly when these strange dates are not included. It can be seen from Table 4.1 that the mean of error drifts in x-direction drops to 0.0180 meters and in y-direction to 0.0548 meters. When considering all logfiles, the mean of heading errors is -32.8921 degrees which is very large. Similarly, heading errors reduce dramatically to 0.6018 degrees when the strange dates mentioned above are removed.

Smoothing poses comparing with RTK	Error drifts in X direction	Error drifts in Y direction	Heading errors
Mean of errors or error drifts	0.0180m	0.0548m	0.6018°

Table 4.1: Smoothing performance without logfiles from the special dates

Overall, taking all factors mentioned above into account, about 83.72% of all logfiles provide valid smoothing. In the future, if the errors or noise introduced by collecting data can be reduced, practical poses resulting from the smoothing would obviously rise. According to the current data, if logfiles from special dates (2016-09-14, 2016-10-06,....) are removed when considering the number of effective logfiles, the percentage of smoothing performance that can be considered as valid increase to 93.39%.

4.2 Visualization of resulting maps

The following section will visually present the resulting maps after alignment has been performed. For testing and validation of the map alignment, logfiles originating from a test area localized on the southern parts of the Gothenburg DriveMe route, shown in Figure 4.11, were mainly used to produce the following results. Due to non-disclosure revolving the sensors used by the probing vehicles, the following maps will lack coordinate axis, as well as have undergone image blurring.

Figure 4.11: Satellite image showing highway where the test-data was gathered from. The satellite image is fetched from Google Maps, 2018 [45].

4.2.1 Sourced map and RTK map comparison

Should the submaps be visualized prior to performing offset-alignment, the resulting radar map would look according to Figure 4.12. As can be seen, the shape of the road is outlined, but the map is shifted as expected. After Graph-SLAM is performed, the resulting map can be viewed in Figure 4.14. Here it can be seen that the proper shape of the road has been achieved, comparable to the RTK map seen in Figure 4.16. Similarly, Figure 4.13, 4.15 and 4.17 shows the corresponding FLC map for the same three cases. Thus it can visually be deduced that the Graph-SLAM alignment is working as expected.

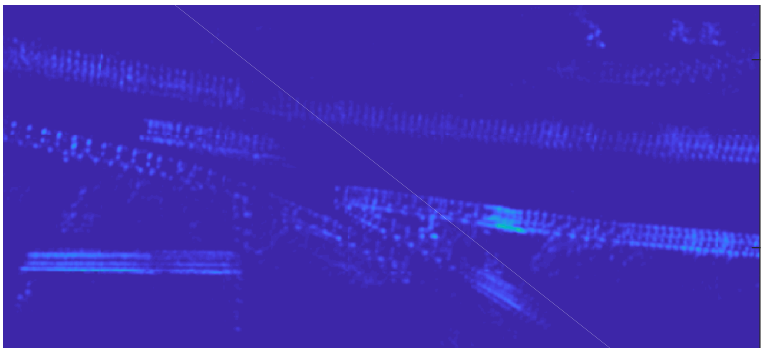


Figure 4.12: Radar map of the test-area before alignment, where the non-adjusted submaps results in a very blurry map.

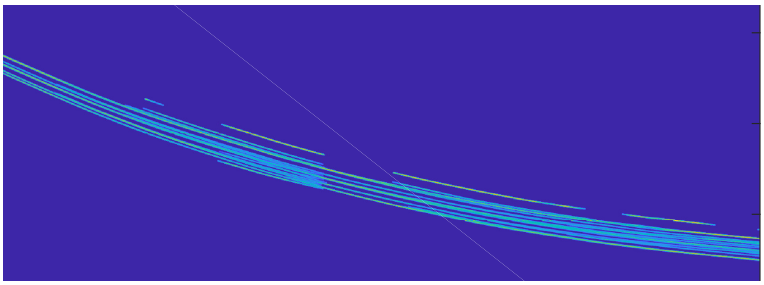


Figure 4.13: FLC map of the test-area before alignment, also resulting in a blurry map.

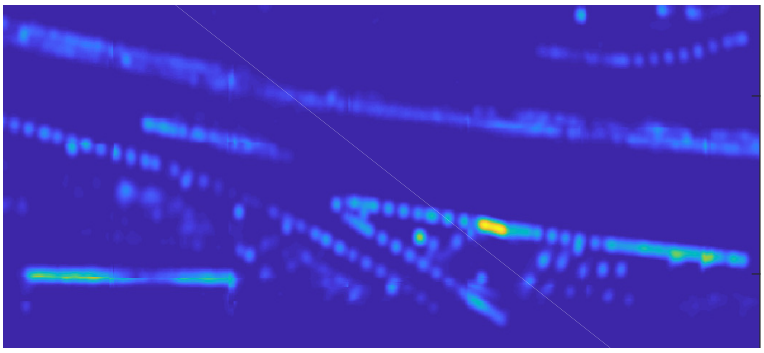


Figure 4.14: Radar map after offset-alignment, now comparable to the RTK map in Figure 4.16. Note that a different blurring was used compared to Figure 4.12.

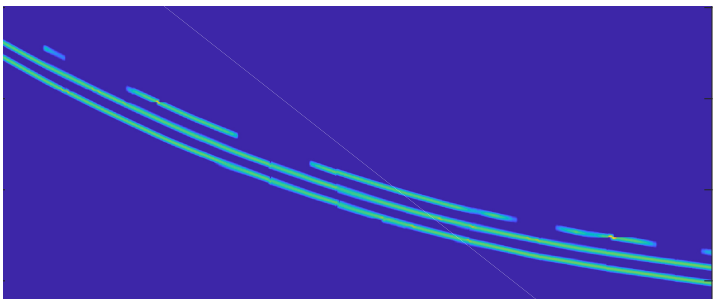


Figure 4.15: FLC map after offset-alignment, comparable with Figure 4.17.

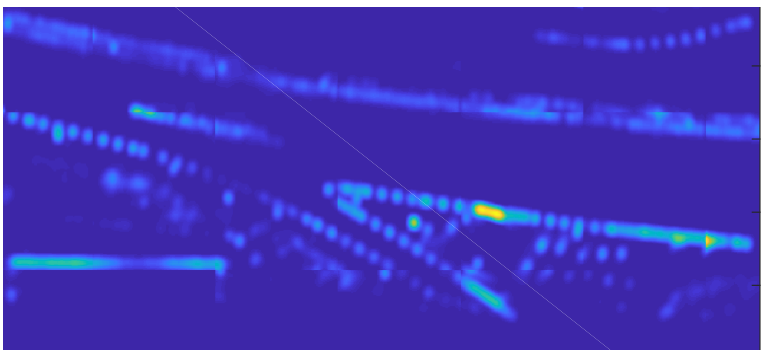


Figure 4.16: RTK radar map for the same area as Figure 4.12 and 4.14.

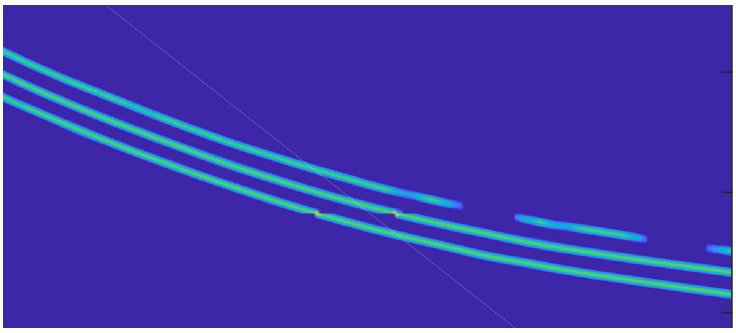


Figure 4.17: RTK FLC map for the same area as Figure 4.13 and 4.15.

4.2.2 Impact of increased amount of data

As explained in Section 3.3.4, increasing the number of submaps available for a certain area helps to separate stationary detections from temporary ones, thus creating a more proper map to use for localization. Another advantage of using more data is the ability to supplement for areas that are missing or occluded when running few logfiles. Figure 4.18 shows one such example, where the map was created using only 5 logfiles. As can be seen, some areas are missing in this map, causing inconsistencies. Figure 4.19 on the other hand was created using 30 logfiles, and it can be seen that the increased amount of data has mended the previous gap.

Figure 4.18: Generating a map using a low amount of logfiles may result in gaps where data is missing.

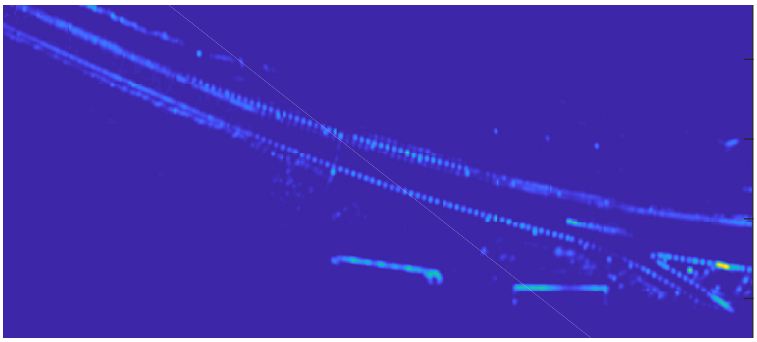


Figure 4.19: Generating the same map using more logfiles, it can be seen that the previously present discontinuities have been mended.

4.2.3 Impact of probing from different lanes

Another phenomenon that might not be immediately visible by observing the radar map, but can be observed from the FLC map, is the impact of which lane a probing vehicle is traveling in. For the construction of radar maps this has no impact, for similar reason as previously stated in Section 3.3.2, but not so for the FLC maps. This means that if only a few or none of the vehicles traverse in either lane, this lane will lack consistency in the map. One such case can be observed in Figure 4.20, a map generated using 30 logfiles where the majority of the probes traveled in the right lane and only 1 in the left. This results in the left lane being underdefined, compared to the true RTK map shown in Figure 4.21.

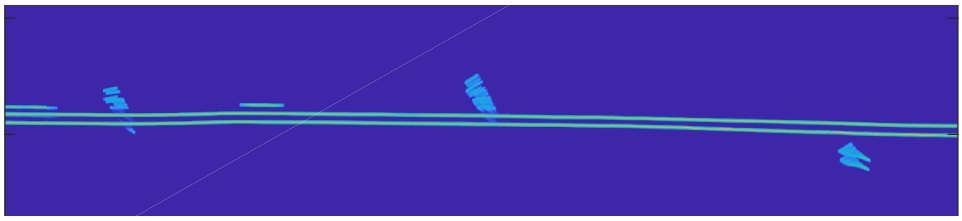


Figure 4.20: Example of an under-defined left lane, while still using 30 logfiles to generate the map. Outliers are also visible.

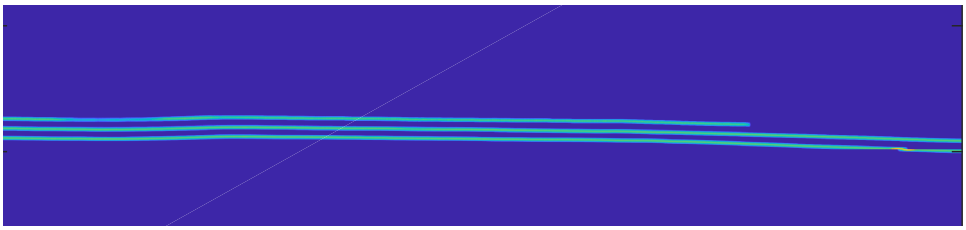


Figure 4.21: Illustration of the RTK map for the same area for comparison, where the left lane is visible.

It can also be observed that some of these logfiles contain submap-outliers, i.e. submaps generated from bad smoothing trajectories. These remain in the FLC map due to the nature of the FLC detection area, where the camera only looks in a narrow area where the lane markings serves as edges of the look-boundary, meaning none of the vehicles in the right lane sees the area containing outliers when adding up the total amount of looks. These are not observable in the same way in the RODS-map shown in Figure 4.22, since the span of the look-area is much larger, meaning outliers will result in a much lower detection probability as long as they remain close to the road. If they are far away from the road, it might be visible, however then it will be of no issue in the localization. Thus the RODS maps can be seen as much more robust compared to the FLC maps when using more data.
Compare this to Figure 4.23, also created using mostly the same 30 logfiles, but 5 of these have been replaced with specifically chosen logfiles where the probing vehicles are driving in the left lane. As can be seen, the left lane is now appearing to be

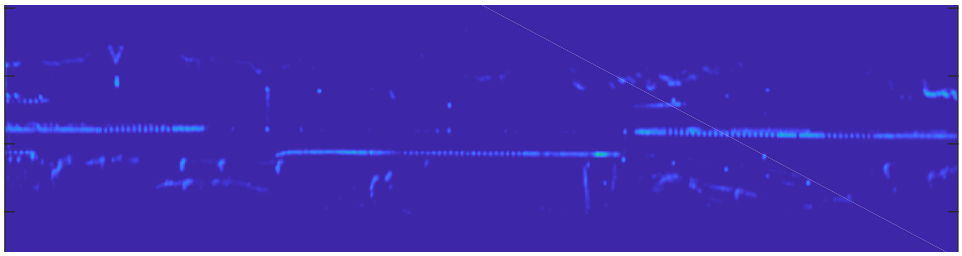


Figure 4.22: Radar map for the same area and generated from the same logfiles as Figure 4.20. The outliers found in the FLC map is not observable in the radar map.

properly visible and comparable to the RTK map in Figure 4.21. Another note is that the outliers previously present have now been subdued by the appearance of other probe vehicles traveling the same area. This means that increasing the amount of data does not necessarily mean an improvement for the FLC map, should the data not contain more vehicles traveling in the less traveled lane. However, increasing the amount of data should intuitively result in an increased amount of probes traveling in the other lane.

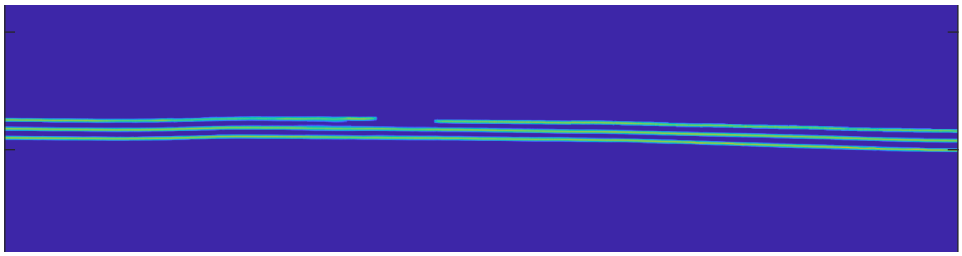
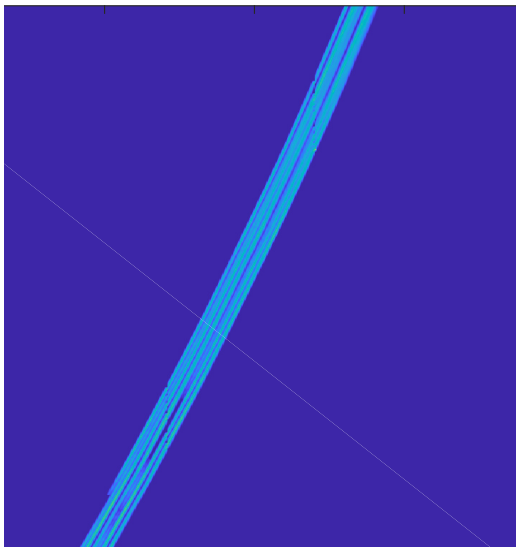


Figure 4.23: When using more logfiles originating from driving in the left lane, said lane is now much more consistent.

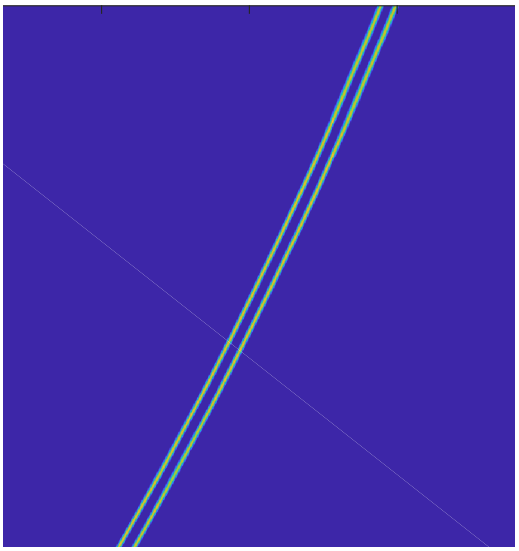
4.2.4 Graph-SLAM inside tunnels

One especially interesting case is the ability to create maps of tunnels, since being able to localize in areas where no valid GPS data is available is of significance. Another reason is to validate how lacking radar reflections affect the resulting map, as is usually the case inside tunnels. Thus the algorithm was tested using logfiles probed from Gnistängstunneln, a 500-meter long tunnel along the DriveMe-route shown in Figure 4.24. The resulting FLC map can be viewed in Figure 4.25b. The RODS map is not illustrated due to difficulties in making out the detections inside the tunnel. As can be seen, the alignment works even inside the tunnel.

Figure 4.24: Satellite imagery of Gnistängstunneln, one of three underground tunnels along the DriveMe-route. The satellite image is fetched from Google Maps, 2018 [46].



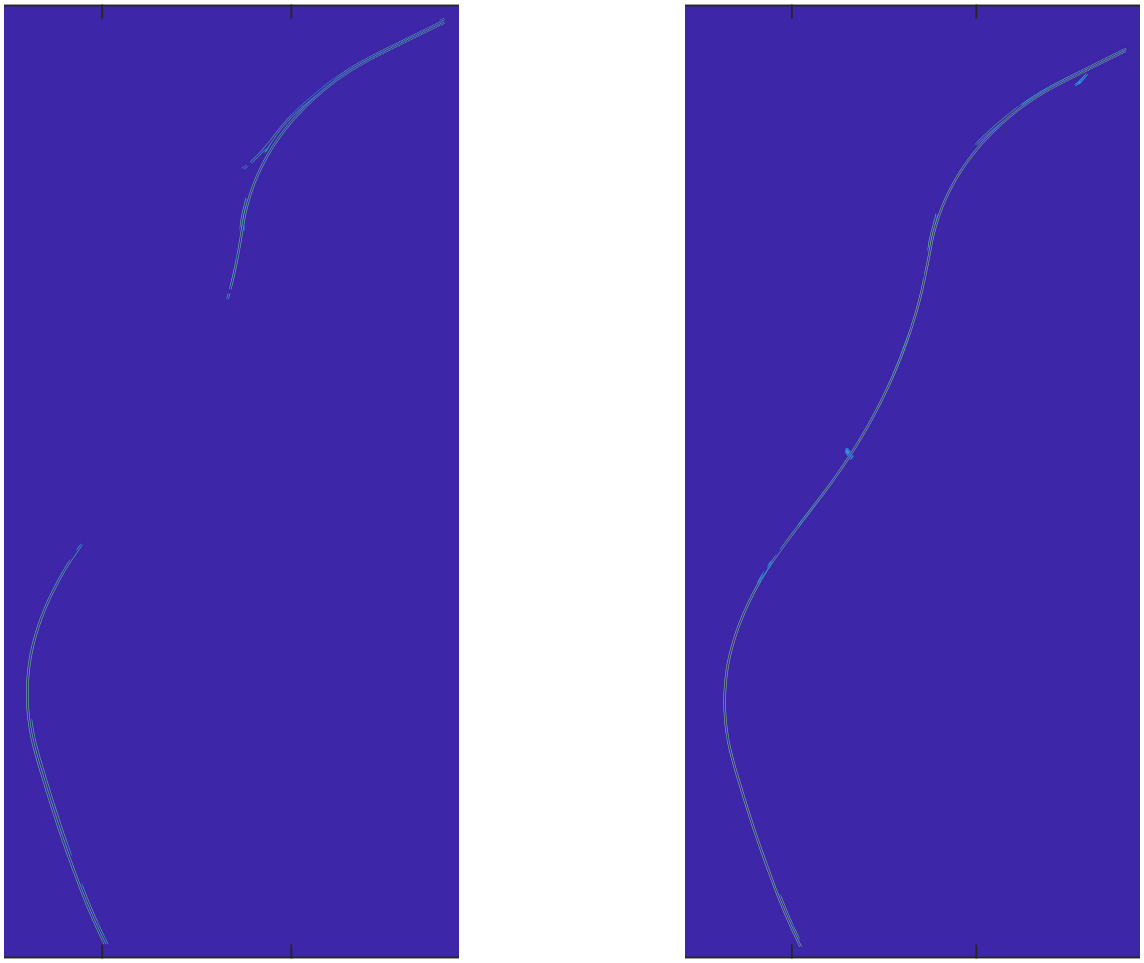
(a) FLC map showing the inside of Gnistängstunneln before alignment.



(b) The alignment provides a valid lane even inside the tunnel.

Figure 4.25: Showing the same area inside Gnistängstunneln, before and after alignment.

The ability to generate crowd-sourced maps in underground tunnels is very interesting in another aspect, namely that it is not possible to generate a RTK map in this scenario. The reason why is that the RTK poses are invalid in the tunnel, due to missing signal, meaning the only submaps that will be generated are the areas outside the entrances of the tunnel where the RTK GPS signal will be lost/resumed. Figure 4.26a shows an overview of the FLC map generated from RTK for Gnistängstunneln. As can be seen, there are large areas of missing information. Compare this to Figure 4.26b, where said areas are appearing. Thus the crowd-sourced map can be considered to actually be better than the RTK map in this specific sense.



(a) RTK FLC map showing large portions of missing areas.

(b) When using crowd-sourced map, the missing areas appear.

Figure 4.26: Comparison between RTK and crowd-sourced maps inside Gnistängstunneln.

4.3 Localization results

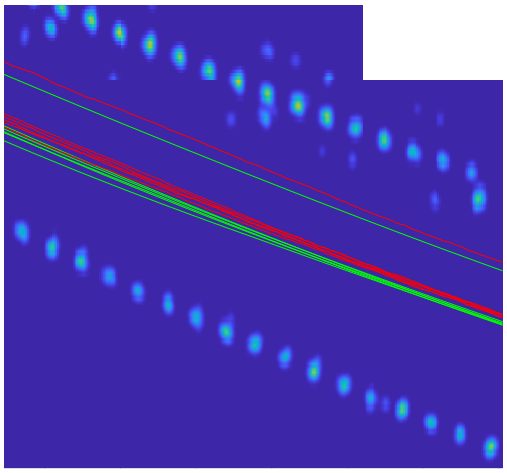
In this section the results originating from the localization will be presented, and some apparent behaviour of the algorithm will also be discussed.

4.3.1 Visual representation of estimations

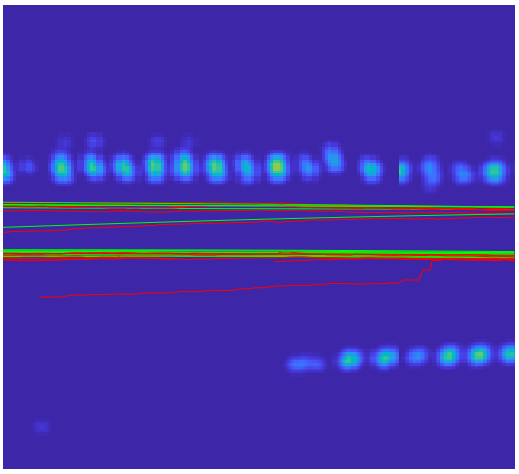
The resulting trajectory from the RODS and FLC localization along the test-route when using 17 validation-logfiles can be shown in Figure 4.27. As can first be seen in Figure 4.27a, the RODS localization does indeed give a valid estimation of the motion of the vehicle, suggesting that the alignment does provide a map sufficient for localization use. Note that the trajectories seem to be locally equally offsetted, which was a behaviour that was expected to occur in Section 3.3.3. In Figure 4.27b it is also possible to visually illustrate the convergence that was discussed in Section 3.4.6 when initiating based on consumer GPS-poses. Figure 4.27c shows the FLC estimations, where one trajectory is being initialized. One difficulty of the FLC localization that became apparent was the tendency to incorrectly align itself outside of the road or in the wrong lane should the consumer GPS-pose be positioned far from the actual lane. This is caused by the localization fitting the right lane of the road as the observations of the left lane. Eventually, the particle cloud may produce particles inside the actual lane that provide better likelihood matches, thus causing the localization to suddenly jump to the correct lane, an example shown in Figure 4.27d.

This behaviour is not a result of the performance of the map itself, rather a product of incorrect matching due to initialization, meaning this behaviour also appear when using a RTK map. Due to this jumpy behaviour, which has a huge impact on the calculated drift, currently the results from the FLC localization are not reliable when evaluating the map. Should this issue be solved, however, it would be possible to show that the crowd-sourced map would be able to provide valid FLC estimations as well, since the estimations that are correctly positioned are consistently following the true trajectory.

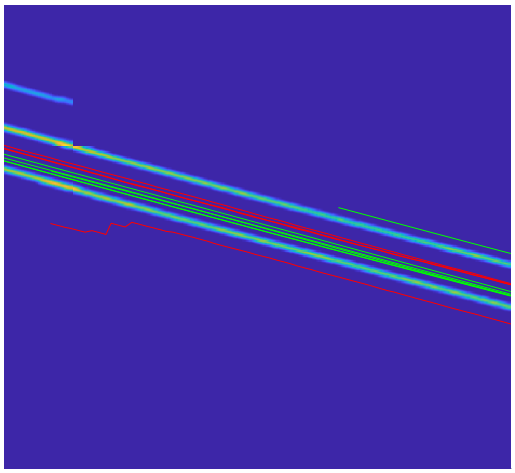
Another way to illustrate the results is to observe the resulting offset and drift plots, shown in Figure 4.28 for the same localization run as above. For this stretch of road, the x-axis (eastward direction) would correspond to the longitudinal direction of the vehicles, and y-axis (northward) as the lateral direction. As can be seen for the RODS-localization results in the left column, and as expected, the drift is centered around 0 (corresponding to a consistent local offset). The convergence-behaviour is also here visible, especially in Figure 4.28c. The heading-error shown in Figure 4.28e is small and approximately bounded between $[-1, 1]$ degrees. This is true for the heading-error resulting from the FLC localization as well.



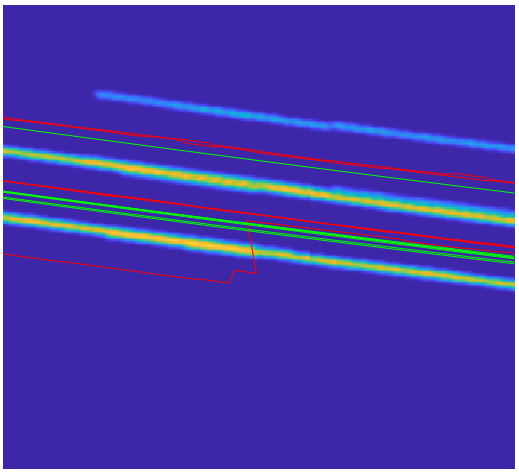
(a) Estimated trajectory of the cars according to the RODS localization.



(b) Initialization using consumer GPS will result in a certain convergence-time.



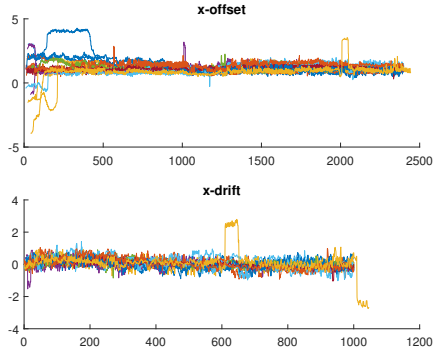
(c) For FLC localization, the initiation may produce incorrect positioning outside the lanes.



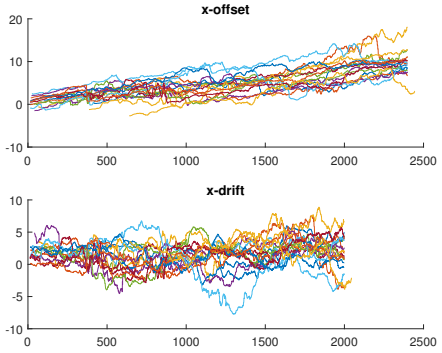
(d) Usually, after a considerable time, the FLC localization positions itself in the correct lane, causing a jump.

Figure 4.27: Visualization of different behaviours that may occur when localizing in the RODS-map (top row) and FLC-map (bottom row) respectively.

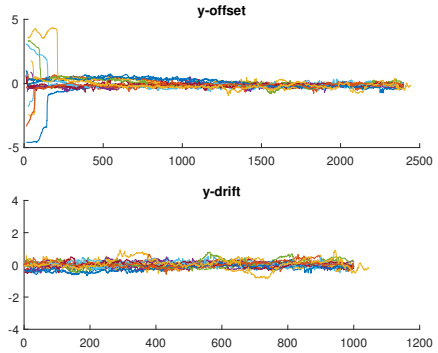
However, the x and y evaluation for the FLC, shown in Figures 4.28b and 4.28d shows the main disadvantage with the FLC localization. Notably, the estimation of position in the longitudinal direction (x) is very inaccurate in the FLC localization, which is logical due to the lack of longitudinal information in lane-markings. Thus the error is comparable to that of dead-reckoning. The previously mentioned issue of the FLC-localization incorrectly positioning itself outside of the correct lane is also evident in Figure 4.28d, which has a large impact on the drift. Without this issue, it can be speculated by observing the errors from correctly positioned estimations that the FLC localization shows promising potential of providing an even smaller drift than the RODS-localization.



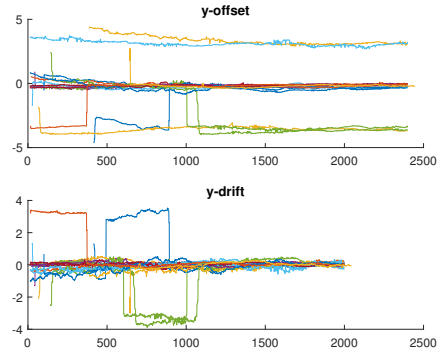
(a) RODS-localization x-offset and drift.



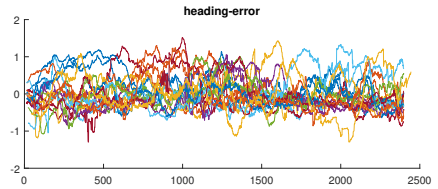
(b) FLC-localization x-offset and drift.



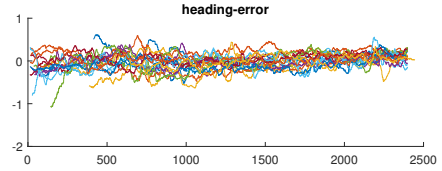
(c) RODS-localization y-offset and drift.



(d) FLC-localization y-offset and drift.



(e) RODS-localization heading-error.



(f) FLC-localization heading-error.

Figure 4.28: Offsets and drifts resulting from the RODS and FLC localization estimations, showing one run of localization using the 17 validation-logfiles.

4.3.2 Evaluating different sensor-configurations

The resulting mean evaluation metrics, calculated according to Equation (3.35), can be seen in Table 4.2. The complete histograms of which these mean-values are calculated from are attached to this report in Appendix A. For clarification, by observing the apparent non-Gaussian distribution in the mentioned histograms, the calculated means in the table are mainly used for comparing different configurations and should not be immediately used for determining the performance of the maps themselves, where the histograms would be of more correct use.

Performance over configurations						
Sensor config.	RODS x-drift	RODS y-drift	RODS θ-error	FLC x-drift	FLC y-drift	FLC θ-error
RTK	0.0314m	0.0296m	0.0932°	1.6266m	0.2196m	0.0544°
Only RODS	0.0465m	0.0664m	0.0755°	1.6738m	0.2180m	0.0792°
RODS & FLC	0.0479m	0.0612m	0.0748°	1.6101m	0.1779m	0.0737°
Only FLC	0.1149m	0.0660m	0.0950°	1.7999m	0.1644m	0.0797°

Table 4.2: Results from both RODS and FLC localizations when using different sensor-configurations, as well as the RTK results.

In order to reach a conclusion regarding the impact of which sensor-measurements that are used in the Graph-SLAM system, namely if including FLC does provide an improvement of the map-alignment, three different configurations were used to generate the crowd-sourced maps; Only include the RODS measurements (meaning $D_3 = 0$), including both RODS and FLC, and only include FLC ($D_{1,2} = 0$). By visual comparison of the maps alone, it is not possible to show any noticeable difference. The evaluation metrics from the RTK map are also included as reference.

As can be seen, the crowd-sourced maps provide enough accuracy in the RODS-estimations to be able to at least be comparable to the RTK map performance-wise, and provide localization in decimeter-accuracy. By comparing different configurations alone, whether or not both FLC and RODS information is used in the system compared to only using RODS does not pose a significant difference. However only using FLC has proved to be disadvantageous, apart for the case of FLC localization if the results could be believed. As stated before, the FLC results are not to be trusted since the drift-metric is heightened by sudden lane-jumps. However, when used from a perspective of comparison alone, it is arguable that since the FLC-behaviour should be the same for all configurations, the two latter configurations does seem to promise an improved performance of the FLC localization, which should serve as the basis for further investigation, especially the case for whether or not map-creation using only FLC information is possible. Regarding the latter, by looking at the RODS y-drift for this configuration it does appear to provide sufficient lateral alignment, however not for x-direction (longitudinal) which is to be expected. Generally, including both FLC and RODS does appear to be the best configuration of the three when looking at overall performance, also including the disputable FLC results.

4.3.3 Map localization in tunnels

Performing localization on the crowd-sourced tunnel map does provide a valid estimation of the vehicle's trajectory along the road. However, it is not possible to calculate drift for this case, since there are no valid RTK poses to compare against. Thus it is only possible, for now, to visually inspect the estimated trajectory. Figure

4.29 shows a stretch of the road inside the tunnel, where the estimated poses based on RODS localization are shown in red and FLC localization estimated shown in white. As can be seen, the RODS estimation drift away from the road much more so than the FLC estimation. This is due to the weak radar reflections inside the tunnel compared to other roads, whereas FLC detections remain more the same. This shows on one of the main motivations of localization based on FLC. However, the FLC localization inside the tunnel still suffers from the behaviour mentioned in the previous section, with estimations possibly positioned outside the road.

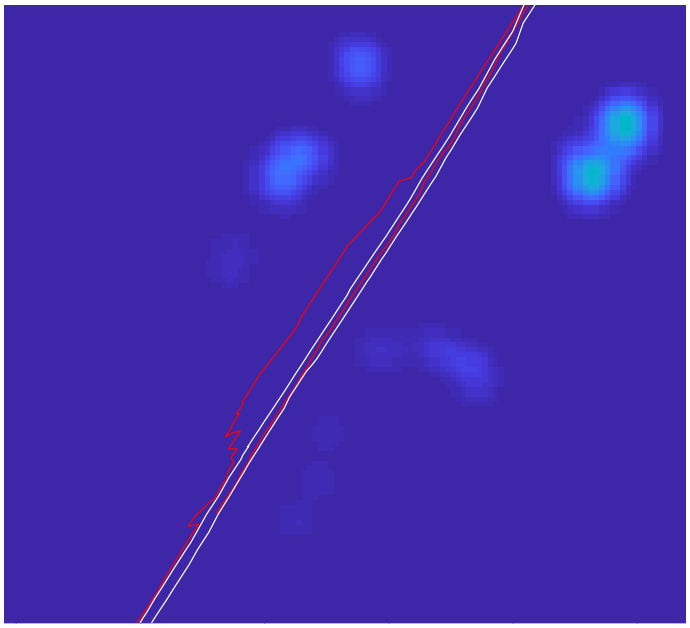


Figure 4.29: Estimated trajectories resulting from RODS (red) and FLC (white) localization.

5
Conclusion and Future work

5.1 Conclusion

Based on the results presented in this thesis, our conclusion is that using Graph-SLAM to align offsetted maps generated from consumer GPS is a valid approach in order to create a map suitable for localization within decimeter-accuracy.

However, more work is needed in order to fully benefit from the additional information that the camera sensor provides, since at the moment it is not possible to independently create a valid camera map in this approach without the use of prior radar-information to find the correct cross-correlation peaks. Should it be possible to independently choose the correct peaks, arguably it will result in the capacity to create a valid camera map based on the results acquired when not including radar-information in the solution. However this conclusion can not be proven concretely until a valid camera-localization has been implemented which does not produce the invalid behaviour present in this thesis.

Currently, we argue that the best results are achieved when using both radar and camera information in the solution, but the improvement compared with only using radar information is not as significant as initially theorized. We believe this is due to the fact that the radar appears to contain sufficient information to be able to align itself, thus the additional information is not contributing significantly. However, our theory is that the true benefit of this configuration could show itself in the case where absolutely no significant radar information is available at all, for example on a road out upon an open field.

5.2 Suggestions for future work

This section contains suggestions for following continued works, based on the experience and various observations that the authors have obtained during the implementation of this thesis.

5.2.1 Independently create FLC maps

As previously stated in the introduction to this thesis, a potentially desirable scenario would be the possibility to align maps using only camera-detections. This is however not possible at the moment, due to limitations in the implementation. For

future work using this implementation, there would be a need to track submaps according after which lane the car was driving in at moment of creation when performing cross-correlation.

Alternatively, another approach would be to not assume normal distribution in the observed errors (which in truth does not hold for FLC cross-correlations where 3 peaks are present), meaning the simplification performed in Equation 3.12 would not be possible. This would result in the problem formulation taking on a iterative-format (much like in general Graph-SLAM theory), meaning the issue of immediately choosing the correct FLC peak is no longer an issue. However, this iterative solution would result in a massive increase in computation time, which might be highly undesirable.

5.2.2 Creating large-scale maps

When generating a crowd-sourced map, there are usually two limiting factors that determines whether or not an ordinary work-computer (laptop) is able to handle the algorithm. Firstly the amount of logfiles that are used for a certain area, and secondly how large of an area the map spans. Both results in an increased number of submaps to contain within the linear system matrix. For a roughly 2x2km area, using more than 30 logfiles may result in a computer with 16GB RAM running out of memory. This means that creating a viable map of the whole DriveMe-route is not possible by only using one run of the algorithm.

However should separately sourced maps describing two areas, with certain segments overlapping, be created, it would be possible to use a similar Graph-SLAM approach to solve this problem as the one used to solve the alignment problem in this thesis. In this case, one could formulate constraints that fixes submaps in each separate map to each other with a very high weight, meaning already aligned submaps will not be moved and thus disrupting the previous alignment. Then, for the overlapping submaps, it is simply a matter of finding the relative offsets between segments using for example cross-correlation, much like the alignment problem of this thesis. This means that a final implementation for creating a large map consisting of several smaller areas could mean first solving a Graph-SLAM problem for submap-alignment within each separate area, and then solving another Graph-SLAM problem for stitching the aligned maps into one.

5.2.3 Improvements upon localization

As previously stated, the FLC-localization suffers from erroneous behaviour when the particle filter positions the car in the wrong lane or even beside the road. This behaviour does not appear when performing RODS-localization, as the RODS-data does not contain significant information regarding the road itself but rather the environment, and thus do not specifically depend upon which lane the car is currently traveling in.

Imagine the case where a new localization algorithm is implemented where RODS and FLC-detections are fused when calculating the likelihood and updating the weights for the particles. This would mean that the FLC behaviour of positioning the vehicle in the wrong lane could be suppressed by using the additional information supplied by the RODS-detections, again since they are independent of the lanes. Additionally, the overall performance of the localization in the lateral direction could be improved by using the additional lateral information contained in the FLC-detections, which appear closer to the car compared to RODS-detections of the environment.

Bibliography

[1] W. H. Organization, *Global status report on road safety 2015*. World Health Organization, 2015.

[2] M. Barth and K. Boriboonsomsin, "Traffic congestion and greenhouse gases," *Access Magazine*, vol. 1, no. 35, 2009.

[3] M. Agrawal and K. Konolige, "Real-time localization in outdoor environments using stereo vision and inexpensive gps," in *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, vol. 3, pp. 1063–1068, IEEE, 2006.

[4] J. Allen Jr, *Use of Vision Sensors and Lane Maps to Aid GPS-INS Navigation*. PhD thesis, 2011.

[5] G. Grisetti, R. Kummerle, C. Stachniss, and W. Burgard, "A tutorial on graph-based slam," *IEEE Intelligent Transportation Systems Magazine*, vol. 2, pp. 31–43, winter 2010.

[6] Z. Tao, P. Bonnifait, V. Fremont, and J. Ibanez-Guzman, "Mapping and localization using gps, lane markings and proprioceptive sensors," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pp. 406–412, IEEE, 2013.

[7] L. Cao and J. Krumm, "From gps traces to a routable road map," in *Proceedings of the 17th ACM SIGSPATIAL international conference on advances in geographic information systems*, pp. 3–12, ACM, 2009.

[8] E. Rehder and A. Albrecht, "Submap-based slam for road markings," in *Intelligent Vehicles Symposium (IV), 2015 IEEE*, pp. 1393–1398, IEEE, 2015.

[9] D. C. Brabham, "Crowdsourcing as a model for problem solving: An introduction and cases," *Convergence*, vol. 14, no. 1, pp. 75–90, 2008.

[10] B. Hofmann-Wellenhof, H. Lichtenegger, and J. Collins, *Global positioning system: theory and practice*. Springer Science & Business Media, 4 ed., 2012.

[11] A. El-Rabbany, *Introduction to GPS: the global positioning system*. Artech house, 2002.

[12] R. B. Langley, "Rtk gps," *GPS World*, vol. 9, no. 9, pp. 70–76, 1998.

[13] D. K. Schrader, B.-C. Min, E. T. Matson, and J. E. Dietz, "Real-time averaging of position data from multiple gps receivers," *Measurement*, vol. 90, pp. 329 – 337, 2016.

[14] M. G. Wing, A. Eklund, and L. D. Kellogg, "Consumer-grade global positioning system (gps) accuracy and reliability," *Journal of forestry*, vol. 103, no. 4, pp. 169–173, 2005.

[15] L. L. Arnold and P. A. Zandbergen, "Positional accuracy of the wide area augmentation system in consumer-grade gps units," *Computers & Geosciences*, vol. 37, no. 7, pp. 883–892, 2011.

[16] E. Abdi, H. S. Mariv, A. Deljouei, and H. Sohrabi, "Accuracy and precision of consumer-grade gps positioning in an urban green space environment," *Forest Science and Technology*, vol. 10, no. 3, pp. 141–147, 2014.

[17] R. Emardson, P. Jarlemark, S. Bergstrand, T. Nilsson, and J. Johansson, "Measurement accuracy in network-rtk. sp report 2009: 23," tech. rep., ISBN 978-91-86319-10-6, SP Technical Research Institute of Sweden, Borås, 2009.

[18] E. Gakstatter, "Centimeter-level rtk accuracy more and more available - for less and less." `http://gpsworld.com/centimeter-level-rtk-accuracy-more-and-more-available-for-less-and-less/`. Accessed: 2018-03-19.

[19] C. M. Gregory, "Inertial measurement unit," Aug. 10 2017. US Patent App. 15/424,969.

[20] J. B. Bancroft and G. Lachapelle, "Data fusion algorithms for multiple inertial measurement units," *Sensors*, vol. 11, no. 7, pp. 6771–6798, 2011.

[21] M. I. Skolnik, "Introduction to radar," *Radar Handbook*, vol. 2, 1962.

[22] V. C. Chen, F. Li, S.-S. Ho, and H. Wechsler, "Micro-doppler effect in radar: phenomenon, model, and simulation study," *IEEE Transactions on Aerospace and electronic systems*, vol. 42, no. 1, pp. 2–21, 2006.

[23] "New collision warning with auto brake helps prevent rear-end collisions." `https://www.media.volvocars.com/global/en-gb/media/pressreleases/12129`. Accessed: 2018-03-20.

[24] "Adaptive cruise control." `https://support.volvocars.com/en-CA/cars/Pages/owners-manual.aspx?mc=v526t8hbat&my=2016&sw=15w46&article=a455580535334d17c0a801514a01f2b8`. Accessed: 2018-05-31.

[25] E. NILSSON, "3d graphics environment for verification of automotive vision system," 2012.

[26] C.-C. Lin and M.-S. Wang, "A vision based top-view transformation model for a vehicle parking assistant," *Sensors*, vol. 12, no. 4, pp. 4431–4446, 2012.

[27] K. Fujii, "Extended kalman filter. chapter 2: Principle of kalman filter," *Refernce Manual*, 2013.

[28] S. Särkkä, *Bayesian filtering and smoothing*, vol. 3. Cambridge University Press, 2013.

[29] L. Svensson, "Sensor fusion and nonlinear filtering: Lecture 5; designing a motion model, an introduction." Lecture slides, 2017. Chalmers University of Technology, Department of Electrical Engineering.

[30] R. Van Der Merwe and E. A. Wan, "The square-root unscented kalman filter for state and parameter-estimation," in *Acoustics, Speech, and Signal Processing, 2001. Proceedings.(ICASSP'01). 2001 IEEE International Conference on*, vol. 6, pp. 3461–3464, IEEE, 2001.

[31] "Mathworks: cholupdate." `https://se.mathworks.com/help/matlab/ref/cholupdate.html`. Accessed: 2018-06-21.

[32] M. Seeger, "Low rank updates for the cholesky decomposition," tech. rep., 2004.

[33] I. M. Rekleitis, "A particle filter tutorial for mobile robot localization," *Centre for Intelligent Machines, McGill University, Tech. Rep. TR-CIM-04-02*, 2004.

[34] J. A. Slater and S. Malys, "Wgs 84—past, present and future," in *Advances in Positioning and Reference Frames*, pp. 1–7, Springer, 1998.

[35] "Sweref 99, lantmäteriet." https://www.lantmateriet.se/sv/Kartor-och-geografisk-information/GPS-och-geodetisk-matning/Referenssystem/Tredimensionella-system/SWEREF-99/. Accessed: 2018-05-28.

[36] "Axes conventions." https://en.wikipedia.org/wiki/Axes_conventions. Accessed: 2018-05-29.

[37] R. Deakin, M. Hunter, and C. Karney, "The gauss-krüger projection," in *Proceedings of the 23rd Victorian regional survey conference*, 2010.

[38] B.-G. Reit, *On geodetic transformations*. Lantmäteriet, 2009.

[39] A. Milstein, "Occupancy grid maps for localization and mapping," in *Motion Planning*, InTech, 2008.

[40] "Google maps, 2018." https://www.google.com/maps/@57.6503794,11.9179313,3863m/data=!3m1!1e3. Accessed: 2018-05-03.

[41] F. Zhao, Q. Huang, and W. Gao, "Image matching by normalized cross-correlation," in *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*, vol. 2, pp. II–II, IEEE, 2006.

[42] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part i," *IEEE robotics & automation magazine*, vol. 13, no. 2, pp. 99–110, 2006.

[43] "Drive me project." https://www.volvocars.com/intl/about/our-innovation-brands/intellisafe/autonomous-driving/drive-me. Accessed: 2018-01-13.

[44] "Mathworks: imregionalmax." https://se.mathworks.com/help/images/ref/imregionalmax.html. Accessed: 2018-05-04.

[45] "Google maps, 2018." https://www.google.com/maps/@57.6463076,11.9310414,787m/data=!3m1!1e3. Accessed: 2018-05-18.

[46] "Google maps, 2018." https://www.google.com/maps/@57.6718972,11.8926052,1071m/data=!3m1!1e3. Accessed: 2018-05-18.

A

Appendix 1

A.1 RTK Drift Histograms

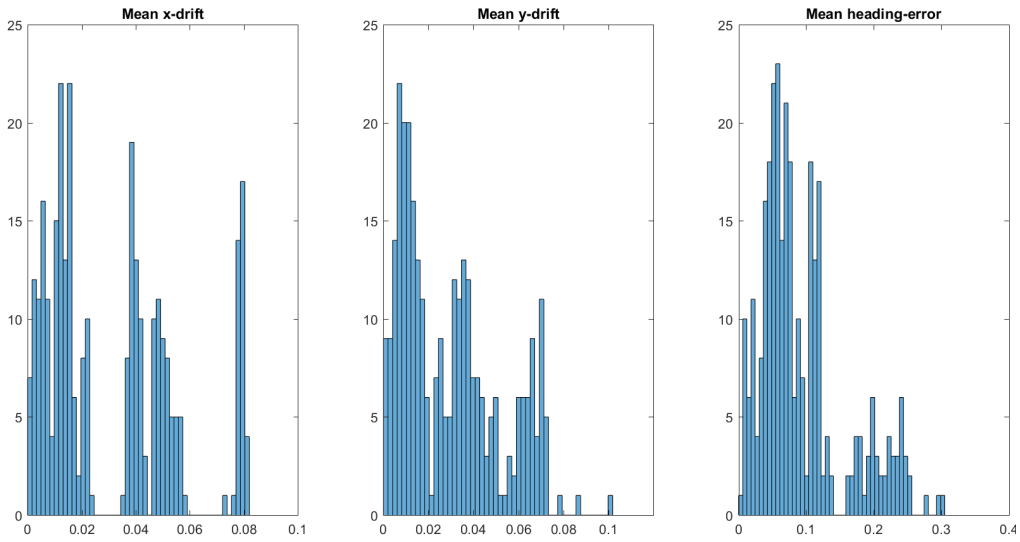


Figure A.1: Histogram of the drift-data from RODS-localization in the RTK map.

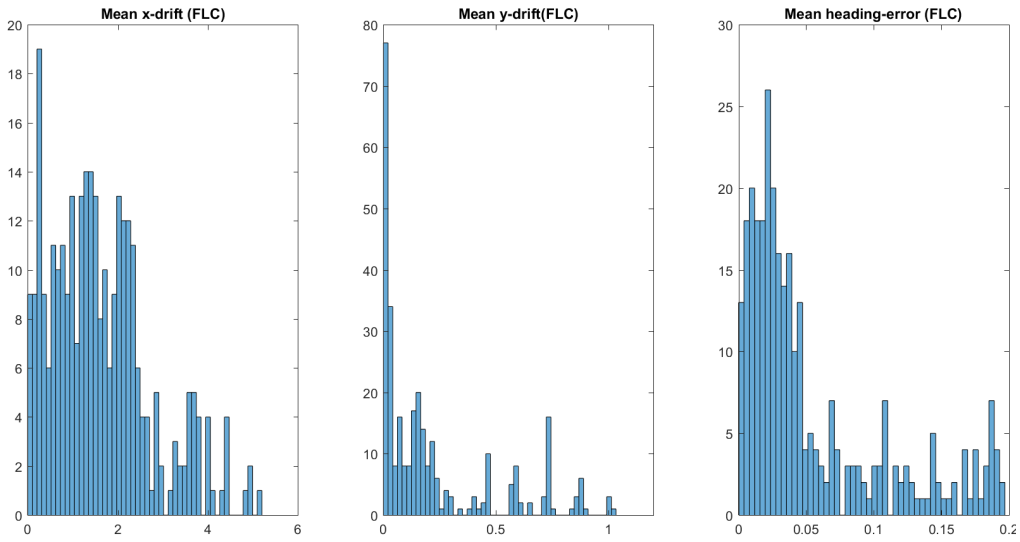


Figure A.2: Histogram of the drift-data from FLC-localization in the RTK map.

A.2 Crowd-sourced Drift Histograms (Only RODS)

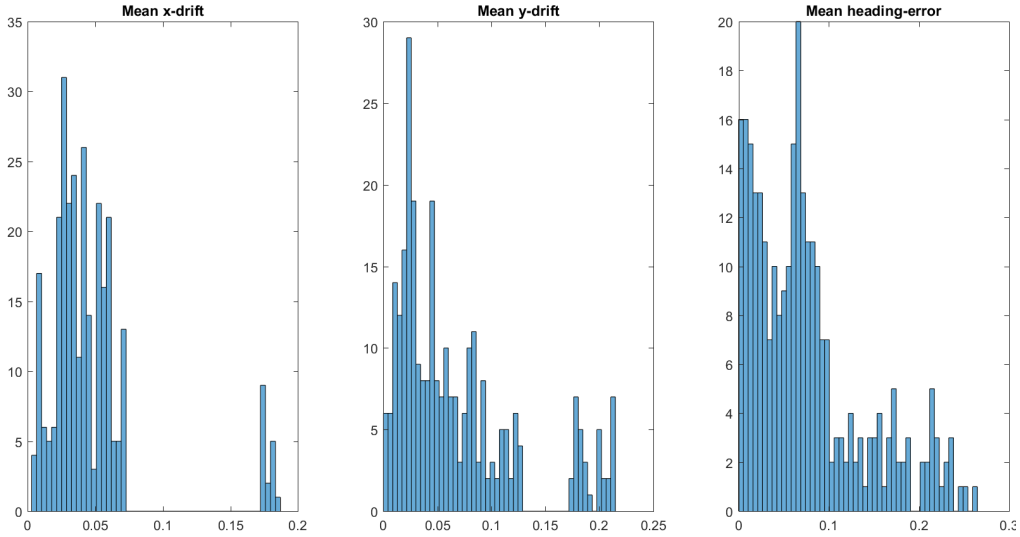


Figure A.3: Histogram of the drift-data from RODS-localization in the crowd-sourced map (using only RODS).

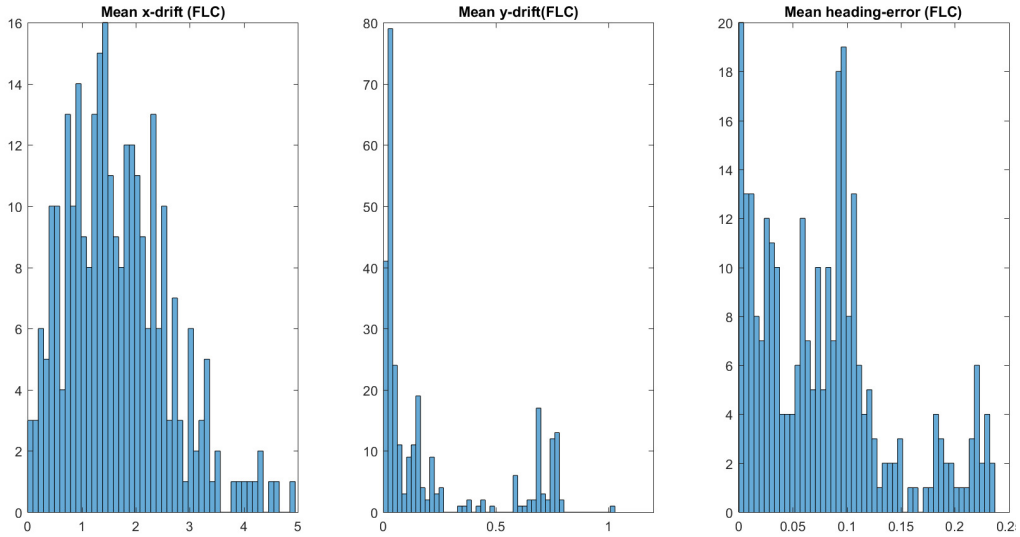


Figure A.4: Histogram of the drift-data from FLC-localization in the crowd-sourced map (using only RODS).

A.3 Crowd-sourced Drift Histograms (RODS+FLC)

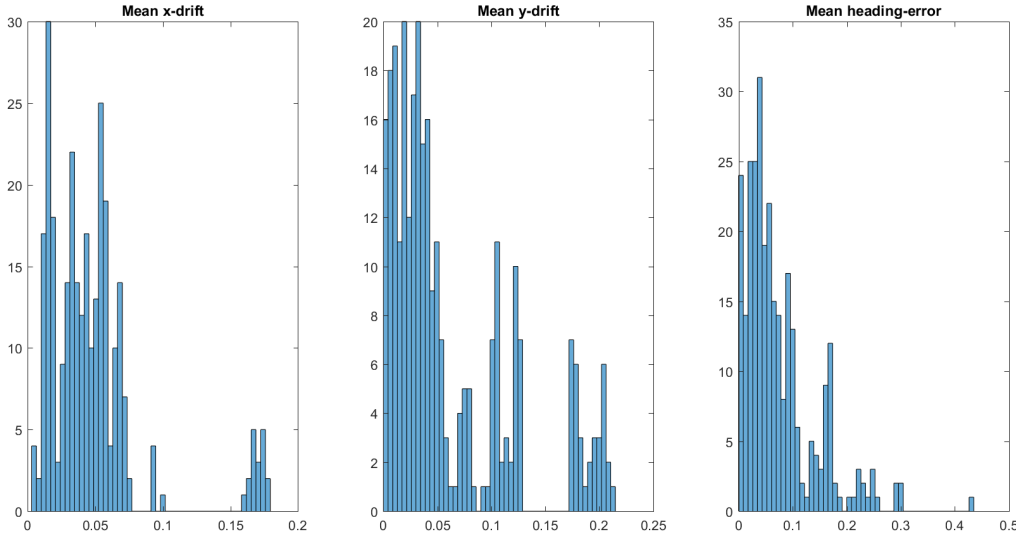


Figure A.5: Histogram of the drift-data from RODS-localization in the crowd-sourced map (using RODS+FLC).

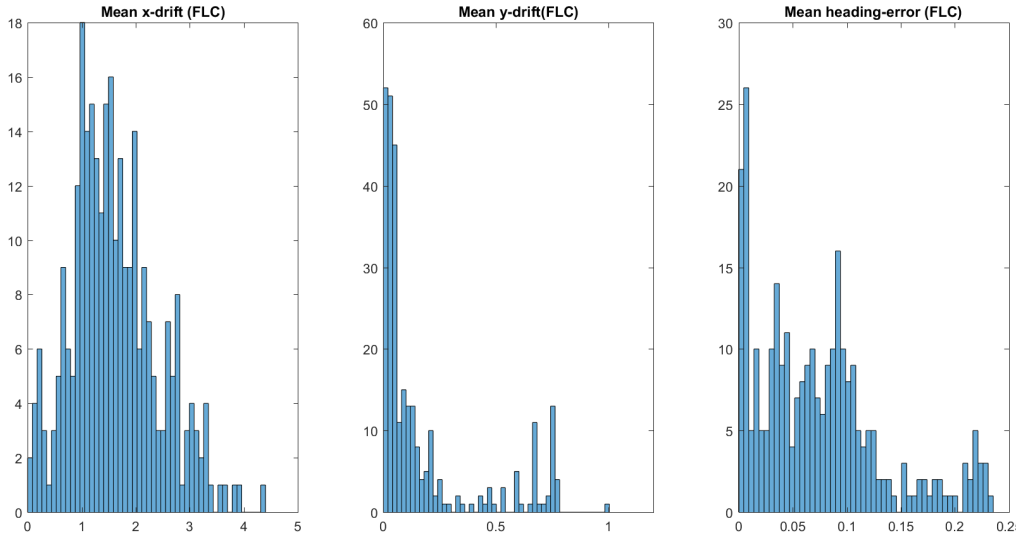


Figure A.6: Histogram of the drift-data from FLC-localization in the crowd-sourced map (using RODS+FLC).

A.4 Crowd-sourced Drift Histograms (Only FLC)

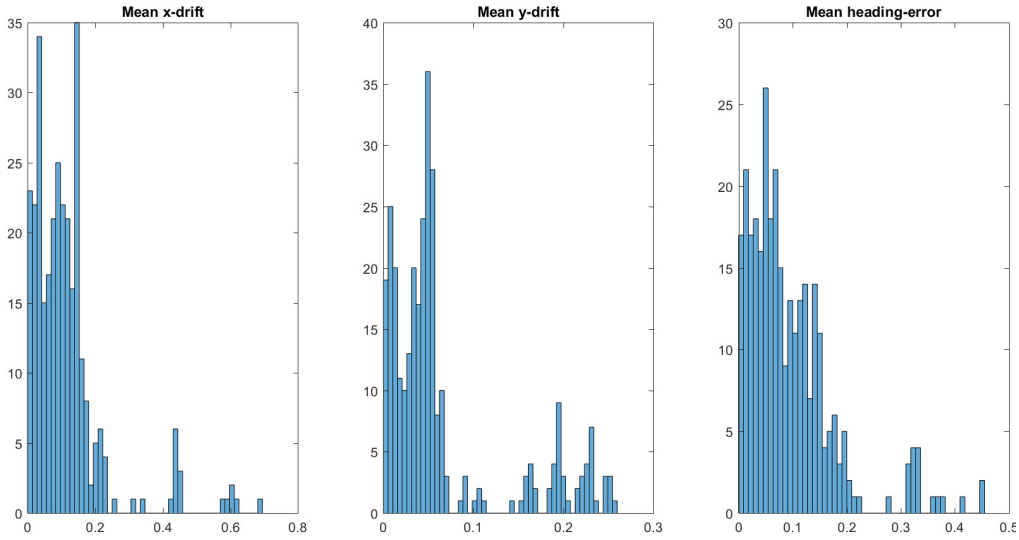


Figure A.7: Histogram of the drift-data from RODS-localization in the crowd-sourced map (using only FLC).

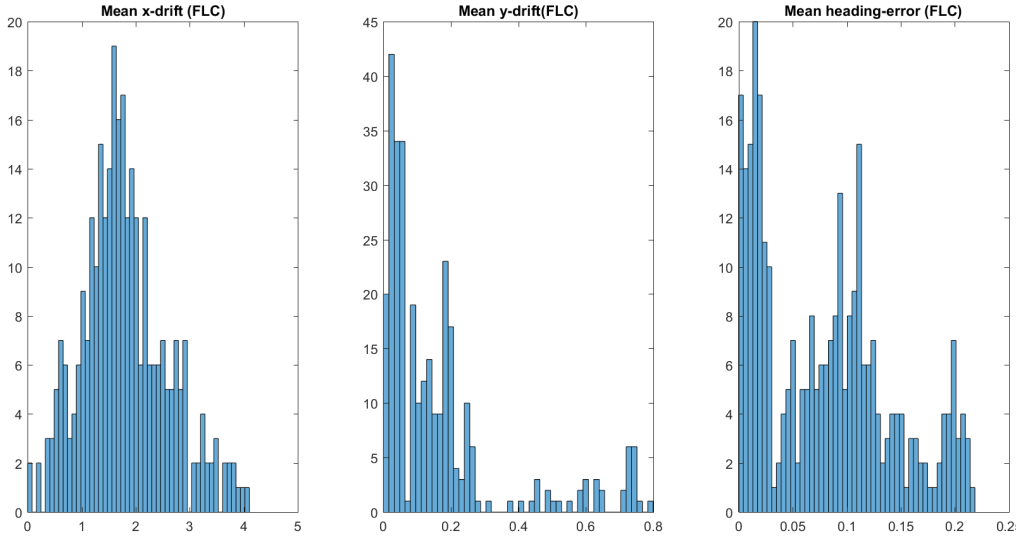


Figure A.8: Histogram of the drift-data from FLC-localization in the crowd-sourced map (using only FLC).