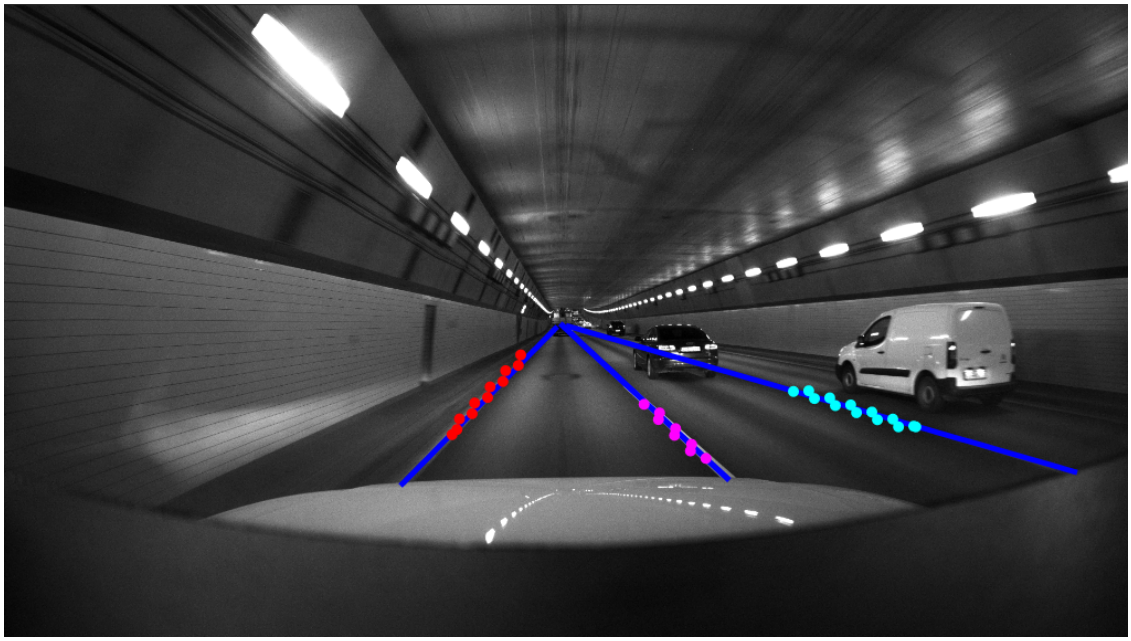

Monocular ego-vehicle localization using high definition map

A comparison of data association techniques in the map-based localization scenario

Master's thesis in Systems, Control and Mechatronics

Adrian Fahlström-Svärd & Zacharias Hultman

Department of Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2021

Master's thesis 2021

Monocular ego-vehicle localization using high definition map

A comparison of data association techniques in the map-based localization scenario

Adrian Fahlström-Svärd & Zacharias Hultman

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden 2021

Monocular ego-vehicle localization using high definition map
A comparison of data association techniques in the map-based localization scenario
Adrian Fahlström-Svärd & Zacharias Hultman

Supervisor: Lars Hammarstrand, Electrical Engineering
Advisor: Axel Beauvisage, Junsheng Fu and Erik Stenborg. Zenseact
Examiner: Lars Hammarstrand, Electrical Engineering

Master's Thesis 2021
Department of Electrical Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Camera image with an illustration of the HD map projected into the image where the blue lines are the HD map, and an illustration of three different road surface markers detected by the camera represented by red, purple and cyan dots. Note that this is not the actual HD map nor the actual detected RSMs, but illustrations showing the concept.

Typeset in LaTeX
Gothenburg, Sweden 2021

Monocular ego-vehicle localization using high definition map
A comparison of data association techniques in the map-based localization scenario
Adrian Fahlström-Svärd & Zacharias Hultman
Department of Electrical Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

Autonomous vehicles and advanced driver assistance-systems have become a heavily researched field of study in the recent years. A vital part of such systems is localization. A common way of implementing a localization system for autonomous vehicles is to make use of cameras together with a prerecorded map. The cameras detect road surface markers (RSMs), which then can be compared with the corresponding RSMs in the map. However, the problem of associating detected RSMs from the camera with the correct RSMs from the map is far from trivial.

In this thesis a localization system is presented using a yaw rate sensor, ego vehicle velocity and acceleration sensors both laterally and longitudinally, as well as a monocular camera paired with a prerecorded HD map. A Cubature Kalman Filter is utilized for fusing the sensors measurements. The camera and HD map measurement update is performed in the image frame. The localization system is tested with three different implementations of data association methods for the camera and map measurements, which are Nearest Neighbour, Individual Compatability Nearest Neighbour and Joint Compatability Branch and Bound. Furthermore, a method of handling roads with bridges above or beneath is implemented and tested. This method is referred to as Multi Layer Road Removal.

The different association methods and the Multi Layer Road Removal are evaluated on 37 sequences of real driving data, where each sequence is one minute long. It is shown that for the localization system, the best association method is Nearest Neigbour, which has a positioning error below one meter for 94.6 % of the sequences. The more complex association method Joint Compatibility Branch and Bound has however shown to be more robust in handling multi layered roads compared to the other methods.

Keywords: Electrical engineering, engineering, map-based localization, localization, AD, ADAS, filtering, computer vision, CKF, data association

Acknowledgements

We would like to thank our supervisors at Zenseact who guided us throughout this project and contributed with their highly valuable expertise. Their patience, interest and eagerness to help has been crucial for this thesis.

We would also like to extend our gratitude to Lars that has helped us as much as he was legally allowed to.

Thanks to Zenseact for giving us the opportunity to work on this interesting and challenging problem. They have provided us with the necessary data and tools for completing this thesis which we are grateful for.

Zacharias och Adrian, Gothenburg, June 2021

Contents

List of Figures

List of Tables

List of Algorithms

1
Introduction

Autonomous vehicles and advanced driver assistance-systems have become a heavily researched field of study in the recent years. In order for an autonomous vehicle to make good decisions it must have a high level of perception, both regarding the surrounding environment as well as its own position in that environment. By using sensor fusion, multiple sensors can be used together to yield an estimate of the vehicle position as well as the surrounding environment, but what types of sensors to be used together is a subject of research. Even methods of associating the sensors data to the property of interest can have a significant effect on the reliability of the vehicle position estimate. Previous work on estimating a vehicles pose using sensor fusion techniques with sensors on the vehicle together with a prerecorded map has been showing promising results with a sub-meter error in pose estimation [1, 2, 3, 4, 5, 6].

This project aims to investigate the problem of designing a sensor fusion filter that estimates the pose and velocity when using a mono camera, a partial IMU and wheel speed sensors together with a high definition 3D map, and is in collaboration with the company Zenseact. The available measurements from the IMU is the ego vehicle acceleration in lateral and longitudinal direction as well as the yaw rate. The high definition map is prerecorded and is provided by the company. The map consists of lines describing where Road Surface Markings (RSMs) are located in a three dimensional global coordinate system. RSMs are the white lines on a road indicating the driving lanes. This thesis does not differentiate between continuous lines or dashed lines but treat them the same way. An algorithm for detection of RSM in the camera images is provided by Zenseact, which gives detections represented by polylines.

To be able to estimate a vehicles pose by using a camera detection of RSMs, a model is needed that describes the relationship between the detections and the map in the RSM of the vehicle pose. More specifically, a metric for determining the error in the vehicle pose by using the distance between a detected RSM and the corresponding RSM in the map is needed. To achieve this, it is possible to either project the image into the map or the other way around. Projecting the map into the image gives a simple representation of the detected RSMs and the prerecorded map with no need of any additional information, but it comes with drawbacks. For example, objects that are occluded in the image might be projected into it from the map, which will cause uncertainties. Another drawback is that it is less intuitive how to calculate the error in the vehicle pose from the image with the projected map. Projecting the image into the map would yield more intuitive vehicle pose errors, but does on the other hand need additional information about the depth to

detections of RSMs from the camera.

In both of the previously described approaches regarding projection, the problem of associating a detected RSM with the corresponding RSM from the map must be considered, both from having multiple lanes in the map as well as having a left RSM and a right RSM for every lane. The problem of determining which RSM from the map a detection corresponds to is investigated.

1.1 Related work

There are various approaches to relate the detected features with a prerecorded 3D map. One approach, presented by Lu et al. in [6], uses a mono camera to detect features and projects the map into the camera image by using the current estimate of the vehicle pose. In another approach presented by Schreiber et al. in [5], the detected features are projected into 3D by using depth data from stereo cameras, where it then can be compared with the map. With other map configurations other solutions are possible. In [7, 2] the map consists of a birds eye view image containing RSMs, and the feature detections from the cameras are therefore transformed to birds eye view perspective. While localization with only use a GNSS receiver is possible for simple navigation maps, it is not sufficient for lane-level accurate maps [7, 4]. GNSS is not accurate enough in its measurements [4] and the GNSS signal is not always accessible, for say when driving in a tunnel or similar [7].

1.1.1 Occlusion

When projecting a prerecorded map into the camera image, a problem that can occur is occlusion. Features that are in the map is not always detectable by the camera since they might be out of sight. Problematic objects are generally roads beneath or above the road which the vehicle is currently driving on. This is because these roads are present in the map but not always visible to the camera. Other problematic situations can occur when RSMs are blocked by objects on the road, such as road barriers. An attempt to solve this is presented by Poggenhans et al. in [4] where a maximum distance is used, such that all map features beyond that distance is discarded. Unfortunately, this only narrows down the problem to occlusion happening near the vehicle. Lu et al. presents a solution in [6] for handling occlusion that occurs due to barriers blocking the sight of RSMs on the other side of the road, which is to only project RSMs for the relevant lane into the image. This solution does however not work if the estimated position has drifted far to another lane or another road, making it less robust and more dependent on the previous estimated position of the car.

1.1.2 Association

A driving lane normally consists of at least two RSMs. To be able to match detected RSM feature with the correct RSM from the map, a measurement has to be associated to the corresponding feature in the map. Schreiber et al. presents in [5]

a solution to this called Nearest Neighbour association relative to the current estimated pose. A detection is simply matched to its closest corresponding feature from the map. This however, does not verify if a position proposed by a measurement is reachable from the previous known position.

When using a particle filter (PF), it is possible to avoid the problem of associating detected RSMs to the corresponding RSMs in the map. This is shown by Jo et al. in [7], where a likelihood field for each particle in the filter can be constructed by taking RSMs gathered from the map using the different proposed positions of the particles. The RSMs from the map is then modeled to have some noise which creates a likelihood field that is compared to the detected RSMs, which then describes the likelihood of the particle that corresponds to that likelihood field.

Poggenhans et al. presents a method of investigating if the detected feature really is matched with the correct corresponding feature from the map in [4]. The detected feature is used to calculate a vehicle pose hypothesis based on the current estimate of the position, which is used to determine the support of that pose among the other detected features. This approach has the ability to distinguish the left RSM from the right RSM, since faulty matches would not generate much support from other detections. A similar solution it presented by Xiao et al. in [8] where a RANSAC algorithm is used to both associate RSM detections with RSMs from the map, and to remove outlier detections. This is done by associating a detected RSM with a map RSM randomly, and calculate how this association would update the estimate of the position. The proposed new position is used to evaluate how well other RSM detections can be associated with map RSMs through a Nearest Neighbour approach. Outliers are removed by only regarding RSM detections that have a near enough correspondence in the map, and the final solution is found by choosing the random association that proposes the new position with the least amount of error for all inlier pairs of detected RSMs and map RSMs.

A solution used in simultaneous localization and mapping (SLAM) implementations is the Joint Compatibility Branch and Bound algorithm presented by Neira in [9]. It makes use of the extended Kalman filter (EKF) equations to compute a property called the normalized innovation square, which is used in two ways. First, the normalized innovation square is computed for an individual association, which is evaluated through an Individual Compatibility test. Furthermore, the algorithm also makes use of a joint normalized innovation square value, which determines the fitness of an entire hypothesis of multiple associations between detections and map RSMs. This is evaluated in a Joint Compatibility test. The method uses a branch and bound search algorithm in order to search for the largest jointly compatible set of detections associated to corresponding map RSMs.

1.1.3 Map lanes representation

The representation of detected RSMs and RSMs from the map determine to some extent the possible approaches for associating them, and also how to relate it to

the filter states. The approach presented by Lu et al. in [6] uses a point-to-point description, where each detected RSM is described as multiple points. Each detected point is compared to a sampled point from the map and the shortest distance is searched to find the best match. A problem with this approach is that the points on the map are sampled, thus making the distance from a detected RSM to the matched map not necessarily the true shortest distance but the shortest distance to the sampled point. This approach is changed to a point-to-polyline solution by Poggenhans et al. in [4], which eliminates the need to sample the elements from the map to match the points.

1.1.4 Filter types

To estimate the vehicle pose based on the visible road markers and odometry sensors a filter is needed. There are various different filters implemented in this field, where for example EKF, Uncented Kalman Filter (UKF) and PF are used frequently [7, 4, 10, 11]. The cubature kalman filter (CKF) has not been as widely documented in this field as the other filter types, but implemented successfully in [12, 13]. These filters have many similarities but there are some major differences. The PF can handle multi modal distributions but is more computational heavy. Jo et al. presents that the PF is better suited for their implementation in [7], due to the possibility to model it as a multi modal distributions.

Poggenhans et al. discards the PF in [4] due to that an oscillation of the most likely position between several equal positions can occur, thus making an UKF better suited for their problems. It is stated by Harr et al. in [10], where PF and EKF is compared, that the problem with the EKF is the lack of robust handling of greater nonlinearities. The conclusion of this comparison is that both have similar and good results but PF is more computational heavy and EKF can not handle ambiguities.

1.2 Objective

The aim of this thesis is to develop a filter that sequentially estimates the 6 degree of freedom pose of a vehicle. The problem of localization using sensors, camera and map contains multiple sub problems. Thus we have divided the relevant problems into 4 parts that this thesis will aim to answer.

- How can the camera and map sensor model be designed such that it correctly describes the measurement uncertainty and handles static occlusion in a robust way?
- How can a filter be designed that sequentially approximates posterior density of the vehicles 6 degree of freedom pose, velocity and acceleration based on the model?
- What is the performance of the proposed filter design? Estimation error and the reliability of the filters confidence of an estimate will be evaluated.
- Which scenarios are difficult and makes this proposed filter setup fail? Why does it fail?

1.3 Scope

A localization system based on a sensor fusion filter is developed. The filter will make use of a prerecorded high definition 3D map containing lane geometry, lane geometry measurements from a front looking camera, a partial IMU and wheel speed measurements. When implementing the sequential filter, different approaches of data association between the camera and the prerecorded map will be evaluated. Furthermore, a method of preprocessing the map to handle multi layer roads will be developed and evaluated.

1.4 Demarcation

To narrow down the scope of the master thesis this section introduces some demarcations. Only one type of sensor fusion filter will be implemented and evaluated, and therefore complete systems with different filter types will not be evaluated against each other. Another demarcation in this project is that RSM detection will not be implemented and evaluated, instead software provided by Zenseact will be used for this functionality. These detection may however not be all correct and outliers will be present.

Furthermore, regarding the previously mentioned techniques of either projecting the map to an image, or vice versa, the thesis implements a solution projecting the map into the image as proposed by Zenseact. The only sensors used are a monocular camera, together with wheel speed sensors, acceleration sensor, yaw rate sensor and a prerecorded map. This sensor setup is chosen by Zenseact.

The thesis main focus is on highway driving environment, and not urban driving environment, since that is what the accessible data represents. For initialization of the system it is assumed that an initial pose of the vehicle is known.

2
Theory

To localize a vehicle in the provided settings of this thesis, some relevant background theory is needed. In the following sections, theory regarding mathematical modelling of a vehicle, filters, data association, different coordinate systems and transformations will be presented.

2.1 Motion models

In order to simulate a system properly, mathematical models describing the system have to be designed, often by using different state space representations. This is a broad topic and various research has been presented in different areas, since different applications have different models [11]. All linear models presented in this section can more generally be described as

$$\mathbf{x}_k = \mathbf{A}_{k-1}\mathbf{x}_{k-1} + \mathbf{q}_{\mathbf{k-1}}, \tag{2.1}$$

where $\mathbf{A_{k-1}}$ is the transition matrix at time $k-1$, $\mathbf{x}_{k-1}$ represents the state vector at time $k-1$ and $\mathbf{q}_{k-1}$ is the noise at time $k-1$. All nonlinear models can be described as

$$\mathbf{x}_k = f(\mathbf{x}_{k-1}) + \mathbf{q}_{k-1}, \tag{2.2}$$

where f is the nonlinear system function of the state values $\mathbf{x}_{k-1}$ at time $k-1$ and $\mathbf{q}_{k-1}$ is the noise at time $k-1$.

There are not only different use cases for the models, but also different levels of complexity. In vehicle applications the models of lower complexity are the constant velocity-model (CV), which assumes constant velocity as

$$\mathbf{x}_k = \begin{bmatrix} x_{k-1} + T \cdot \dot{x}_{k-1} \\ y_{k-1} + T \cdot \dot{y}_{k-1} \\ \dot{x}_{k-1} \\ \dot{y}_{k-1} \end{bmatrix} + \mathbf{q}_{k-1}, \tag{2.3}$$

where $\mathbf{q}_{k-1}$ is the noise at time $k-1$, T is the sampling time and the state space is

$$\mathbf{x}_k = \begin{bmatrix} x_k & y_k & \dot{x}_k & \dot{y}_k \end{bmatrix}^T. \tag{2.4}$$

Here the states are the position in x and y and the velocities $\dot{x}$, $\dot{y}$ at time k. The dots above the state indicates time derivative, one for each derivative. The constant acceleration-model (CA) is shown in equation (2.5).

$$\mathbf{x}_k = \begin{bmatrix} x_{k-1} + T \cdot \dot{x}_{k-1} + \frac{T^2}{2} \cdot \ddot{x}_{k-1} \\ y_{k-1} + T \cdot \dot{y}_{k-1} + \frac{T^2}{2} \cdot \ddot{y}_{k-1} \\ \dot{x}_{k-1} + T \cdot \ddot{x}_{k-1} \\ \dot{y}_{k-1} + T \cdot \ddot{y}_{k-1} \\ \ddot{x}_{k-1} \\ \ddot{y}_{k-1} \end{bmatrix} + \mathbf{q}_{k-1}, \tag{2.5}$$

$\mathbf{q}_{k-1}$ is the noise at time t and the state space is defined as

$$\mathbf{x}_k = \begin{bmatrix} x_k & y_k & \dot{x}_k & \dot{y}_k & \ddot{x}_k & \ddot{y}_k \end{bmatrix}^T.$$

The states are very similar to the CV model, but the accelerations in each direction is added and denoted as $\ddot{x}$ and $\ddot{y}$ at time k. Most often the assumption of constant velocity or acceleration is false, and therefore noise is added to these state time derivatives to model the changes.

The advantage of these simpler models is their linearity. However, the movement of the car is not always possible to describe with these simpler models, for example when there is rotation to be taken into account. These models are fairly similar but both have different advantages, for example, if the system tends to not vary a lot in velocity, the CV model works well and does not add unnecessary states. But this model will fail if the velocity varies a lot, which makes the CA model better suited [14].

To describe more complex motions of the vehicle, more complex models are needed. By adding rotations around z-axis, the heading of the vehicle can be taken into account. These models are sometimes referred to as curvilinear models. The most simple model of this complexity is the constant turn rate and velocity (CTRV) model

$$\mathbf{x}_k = \begin{bmatrix} \frac{v_{k-1}}{\dot{\theta}_{k-1}} \sin(\dot{\theta}_{k-1}\Delta T + \theta_{k-1}) - \frac{v_{k-1}}{\dot{\theta}_{k-1}} \sin(\theta_{k-1}) + x_{k-1} \\ -\frac{v_{k-1}}{\dot{\theta}_{k-1}} \cos(\dot{\theta}_{k-1}\Delta T + \theta_{k-1}) + \frac{v_{k-1}}{\dot{\theta}_{k-1}} \sin(\theta_{k-1}) + y_{k-1} \\ \dot{\theta}_{k-1}\Delta T + \theta_{k-1} \\ v_{k-1} \\ \dot{\theta}_{k-1} \end{bmatrix} + \mathbf{q_{k-1}}, \tag{2.6}$$

$\mathbf{q_{k-1}}$ is the noise at time $k-1$. The state for this model is

$$\mathbf{x}_k = \begin{bmatrix} x_k & y_k & \theta_k & v_k & \dot{\theta}_k \end{bmatrix}^T,$$

where x is the position in x-axis at time k; y is the position in y-axis at time k; θ is the angle around z-axis referred to as yaw-angle at time k; v is the velocity at time k and $\dot{\theta}$ is the yaw rate at time k [14].

Another similar model is the constant turn rate and acceleration (CTRA) model

$$\mathbf{x}_k = \mathbf{x}_{k-1} + \begin{bmatrix} \Delta x_{k-1} \\ \Delta y_{k-1} \\ \dot{\theta}_{k-1}\Delta T \\ a_{k-1}\Delta T \\ 0 \\ 0 \end{bmatrix} + \mathbf{q_{k-1}}, \tag{2.7}$$

$\mathbf{q}_{k-1}$ is the noise parameter at time $k-1$. The states for CTRA is

$$\mathbf{x}_k = \begin{bmatrix} x_k & y_k & \theta_k & v_k & a_k & \dot{\theta}_k \end{bmatrix}^T,$$

which are very similar to the states for CTRV. The acceleration a at time k is added as a state, since the velocity is not constant in this model. Δx_{k-1} and Δy_{k-1} are given by equation (2.8) and (2.9).

$$\begin{aligned} \Delta x_{k-1} =& \frac{1}{\dot{\theta}_{k-1}^2}[(v_{k-1}\dot{\theta}_{k-1} + a_{k-1}\dot{\theta}_{k-1}\Delta T)\sin(\theta_{k-1} + \dot{\theta}_{k-1}\Delta T) \\ &+ a_{k-1}\cos(\theta_{k-1} + \dot{\theta}_{k-1}\Delta T) \\ &- v_{k-1}\dot{\theta}_{k-1}\sin\theta_{k-1} - a_{k-1}\cos\theta_{k-1}] \end{aligned} \tag{2.8}$$

$$\begin{aligned} \Delta y_{k-1} =& \frac{1}{\dot{\theta}_{k-1}^2}[(-v_{k-1}\dot{\theta}_{k-1} - a_{k-1}\dot{\theta}_{k-1}\Delta T)\cos(\theta_{k-1} + \dot{\theta}_{k-1}\Delta T) \\ &+ a_{k-1}\sin(\theta_{k-1} + \dot{\theta}_{k-1}\Delta T) \\ &+ v_{k-1}\dot{\theta}_{k-1}\cos\theta_{k-1} - a_{k-1}\sin\theta_{k-1}] \end{aligned} \tag{2.9}$$

CTRV and CTRA assumes no correlation between the velocity v_{k-1} and the yaw rate ω_{k-1}. This makes it possible for the vehicle to change heading when the vehicle is not moving. The CTRV model assumes that the vehicle moves on a circular trajectory, whilst CTRA model models a linear variation of the curvature, making the assumed trajectory a clothoid [14].

2.2 Observation models

An observation model describes how measurements relate to the states of the system. This is highly depending on the state space and the sensors of the system. These models can be both linear or nonlinear depending on the states of the system and what is observed. A general description of observation model is shown in Equation (2.10), which describes how to predict the measurements $\mathbf{y}_k$ by using the states $\mathbf{x}_k$ with the observation function $h(\mathbf{x}_k)$ with added noise $\mathbf{r}_k$.

$$\mathbf{y}_k = h\left(\mathbf{x}_k\right) + \mathbf{r}_k, \tag{2.10}$$

In equation, (2.10) $\mathbf{y}_k$ describes the predicted measurement using the current state estimate.

2.3 Bayesian filters

The well-known Bayes theorem, see equation (2.11), has become one of the important branches in statistics [15]. Bayes theorem is used to express a property of interest, the posterior, based on something known.

$$p(\mathbf{x} \mid \mathbf{y}) = \frac{p(\mathbf{y} \mid \mathbf{x})p(\mathbf{x})}{p(\mathbf{y})} \tag{2.11}$$

In equation (2.11), $\mathbf{x}$ represents a state vector, $\mathbf{y}$ represents an observation vector, $p(\mathbf{x} \mid \mathbf{y})$ is the **posterior**, $p(\mathbf{y} \mid \mathbf{x})$ is a **likelihood** of an observation $\mathbf{y}$, $p(\mathbf{x})$ is the **prior** for $\mathbf{x}$ and $p(\mathbf{y})$ is the **prior** for $\mathbf{y}$. Since y is observed, $p(\mathbf{y} \mid \mathbf{x})$ is often viewed as the likelihood function of x as

$$l(\mathbf{x} \mid \mathbf{y}) = p(\mathbf{y} \mid \mathbf{x}). \tag{2.12}$$

A simplified version of the Bayes theorem is presented in Equation (2.13).

$$\text{posterior} \propto \text{likelihood} \times \text{prior} \tag{2.13}$$

Bayesian statistic can be used in filtering applications, referred as Bayesian filters. Bayesian filter is under the assumption that the true states can be represented by a Markov model. That is, the state and observations at time steps prior to the known state x_k will not have an explicit influence on the future state. However, the previous states and observations is represented through the state $\mathbf{x}_k$. Bayesian filters are recursive and, the posterior distribution at time k is $p(\mathbf{x}_k \mid \mathbf{y}_{1:k})$, where $\mathbf{y}_{1:k}$ is all previous observations. This is derived from $p(\mathbf{x}_{k-1} \mid \mathbf{y}_{1:k-1})$, see algorithm 1 where one iteration is shown.

Data: In : $p(\mathbf{x}_k \mid \mathbf{x}_{k-1})$, $p(\mathbf{x}_{k-1} \mid \mathbf{y}_{1:k-1})$, $p(\mathbf{y}_k \mid \mathbf{x}_k)$
Result: Out : $p(\mathbf{x}_k \mid \mathbf{y}_{1:k})$
Predict
1 $p(\mathbf{x}_k \mid \mathbf{y}_{1:k-1}) = \int p(\mathbf{x}_k \mid \mathbf{x}_{k-1})\, p(\mathbf{x}_{k-1} \mid \mathbf{y}_{1:k-1})\, d\mathbf{x}_{k-1}$
Update measurement
2 $p(\mathbf{x}_k \mid \mathbf{y}_{1:k}) \propto p(\mathbf{y}_k \mid \mathbf{x}_k)\, p(\mathbf{x}_k \mid \mathbf{y}_{1:k-1})$

Algorithm 1: Bayesian filter algorithm

Line 1 in algorithm 1 is often referred to as the prediction step and line 2 is referred as the update step.

One type of Bayesian filters is the Kalman filter (KF), which there are multiple versions of depending on the usage and setting [15, 16, 17]. The standard KF can only handle linear systems, the EKF, UKF and CKF all handle both linear and nonlinear systems. The difference between EKF, UKF and CKF is that both

UKF and CKF can handle greater nonlinearities but have a higher computational complexity [16, 17].

The KF calculates a distribution and to get the actual state value the minimum mean square error (MMSE) of the distribution is the state value. Since this thesis will handle both linear and more nonlinear applications, the regular KF will not be suitable, thus more emphasis will be on the nonlinear Kalman filter types.

2.3.1 Extended Kalman Filter

Many useful motion and measurement models in vehicle applications are non linear, as the CTRV and CTRA models shown in section 2.1. To filter these models nonlinear KF is needed, such as the EKF. The EKF if very similar to the standard KF but the idea is to linearize $f(\mathbf{x}_{k-1})$ and $h(\mathbf{x}_k)$ and then apply the Kalman filter on the linearized system. $f(\mathbf{x}_{k-1})$ and $h(\mathbf{x}_k)$ are linerized by first order Taylor expansion as

$$\mathbf{x}_k = f\left(\mathbf{x}_{k-1}\right) + \mathbf{q}_{k-1} \tag{2.14}$$

$$\mathbf{x}_k \approx f\left(\hat{\mathbf{x}}_{k-1|k-1}\right) + f'\left(\hat{\mathbf{x}}_{k-1|k-1}\right)\left(\mathbf{x}_{k-1} - \hat{\mathbf{x}}_{k-1|k-1}\right) + \mathbf{q}_{k-1}, \tag{2.15}$$

and

$$\mathbf{y}_k = h\left(\mathbf{x}_k\right) + \mathbf{r}_k \tag{2.16}$$

$$\mathbf{y}_k \approx h\left(\hat{\mathbf{x}}_{k|k-1}\right) + h'\left(\hat{\mathbf{x}}_{k|k-1}\right)\left(\mathbf{x}_k - \hat{\mathbf{x}}_{k|k-1}\right) + \mathbf{r}_{k-1}. \tag{2.17}$$

The EKF algorithm is shown in algorithm 2, where one iteration is shown. Compared to the other presented nonlinear filter types EKF has the lowest computational complexity. Since KF calculates a distribution and then the MMSE is chosen from that distribution, the updated state values is denoted as the mean $\hat{\mathbf{x}}_k$.

Data: In : $\hat{\mathbf{x}}_{k-1|k-1}$, $\mathbf{P}_{k-1|k-1}$, $\mathbf{Q}_{k-1}$, $\mathbf{R}_k$, f, h, $\mathbf{y}_k$
Result: Out : $\hat{\mathbf{x}}_{k|k}$,$\mathbf{P}_{k|k}$
Prediction step
1 $\hat{\mathbf{x}}_{k|k-1} = f\left(\hat{\mathbf{x}}_{k-1|k-1}\right)$
2 $\mathbf{P}_{k|k-1} = f'\left(\hat{\mathbf{x}}_{k-1|k-1}\right)\mathbf{P}_{k-1|k-1}f'\left(\hat{\mathbf{x}}_{k-1|k-1}\right)^T + \mathbf{Q}_{k-1}t$
Update measurement step
3 $\mathbf{S}_k = h'\left(\hat{\mathbf{x}}_{k|k-1}\right)\mathbf{P}_{k|k-1}h'\left(\hat{\mathbf{x}}_{k|k-1}\right)^T + \mathbf{R}_k$
4 $\mathbf{K}_k = \mathbf{P}_{k|k-1}h'\left(\hat{\mathbf{x}}_{k|k-1}\right)^T\mathbf{S}_k^{-1}$
5 $\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k\left(\mathbf{y}_k - h\left(\hat{\mathbf{x}}_{k|k-1}\right)\right)$
6 $\mathbf{P}_{k|k} = \mathbf{P}_{k|k-1} - \mathbf{K}_k\mathbf{S}_k\mathbf{K}^T$

Algorithm 2: Extended Kalman filter algorithm

Lines 1 through 2 are the prediction step in algorithm 2. On line 2 is the predicted states calculated through the motion model f and the states from the previous time

step. Similar on line 3 but for predicting the covariance using the previous covariance $\mathbf{P}_{k-1|k-1}$ and measurement noise covariance $\mathbf{Q}_{k-1}$ Lines 3 through 6 is the update step. Line 3 calculates the innovation covariance $\mathbf{S}_k$. The Kalman gain is calculated on line 4, this parameter indicates how much the measurement should be incorporated into the new state estimate [18]. Line 5 updates the mean by adjusting it to the Kalman gain and the deviation of the actual measurement and the measurement predicted according to the measurement model, also referred to as the innovation. Lastly is the new covariance of the posterior calculated in Line 6, adjusting for the information gain resulting from the measurement [18].

2.3.2 Uncented Kalman Filter

The idea with both UKF and CKF is to use σ-points, which creates a probability distribution, to better approximate an arbitrary nonlinear function. For UKF the σ-points, $\mathcal{X}^i$, are generated as

$$\begin{aligned}
\mathcal{X}^{(0)} &= \hat{\mathbf{x}} \\
\mathcal{X}^{(i)} &= \hat{\mathbf{x}} + \sqrt{\frac{n}{1-W_0}}\mathbf{P}_i^{1/2}, \quad i = 1, 2, \ldots, n \\
\mathcal{X}^{(i+n)} &= \hat{\mathbf{x}} - \sqrt{\frac{n}{1-W_0}}\mathbf{P}_i^{1/2}, \quad i = 1, 2, \ldots, n \\
\mathbf{W}_i &= \frac{1-W_0}{2n},
\end{aligned} \tag{2.18}$$

where $\mathbf{P}_i^{1/2}$ is the i-th column of $\mathbf{P}^{1/2}$ and n is the number of states. In a UKF, $2n+1$ σ-points are generated.

In equation (2.18) W_i is the weights, where a common choice of initial weights are $W_0 = 1 - \frac{n}{3}$. The σ-points are used in algorithm 3, where one iteration is shown. This algorithm is similar to the EKF algorithm, see algorithm 2, the major difference is the use of σ-points for approximating the state distribution.
P_{xy} on line 8 is the cross covariance between the states and the measurements. Otherwise the notations are similar as in the EKF case.

2.3.3 Cubature Kalman Filter

CKF is a very similar to UKF, but varies in σ-points calculations. For CKF the σ-points is calculated as

$$\begin{aligned}
\mathcal{X}^{(i)} &= \hat{\mathbf{x}} + \sqrt{n}\mathbf{P}_i^{1/2}, \quad i = 1, 2, \ldots, n \\
\mathcal{X}^{(i+n)} &= \hat{\mathbf{x}} - \sqrt{n}\mathbf{P}_i^{1/2}, \quad i = 1, 2, \ldots, n \\
W_i &= \frac{1}{2n}
\end{aligned} \tag{2.19}$$

This is very similar to the UKF σ-points equation (2.18), but $W_0 = 0$ and $\mathcal{X}^0$ is not used thus making the total number of sigma points $2n$.

Data: In : $\hat{\mathbf{x}}_{k-1|k-1}$, $\mathbf{P}_{k-1|k-1}$, $\mathbf{Q}_{k-1}$, $\mathbf{R}_k$, f, h, $\mathbf{y}_k$
Result: Out : $\hat{\mathbf{x}}_{k|k}$,$\mathbf{P}_{k|k}$
Form a set of σ-points, $\mathcal{X}_{k-1}$
Prediction step
$\hat{\mathbf{x}}_{k|k-1} \approx \sum_{i=0}^{2n} f\left(\mathcal{X}_{k-1}^{(i)}\right) W_i$
$\mathbf{P}_{k|k-1} \approx a_{k-1} + \sum_{i=0}^{2n+1} \left(f\left(\mathcal{X}_{k-1}^{(i)}\right) - \hat{\mathbf{x}}_{k|k-1}\right)(\cdot)^T W_i$
Update measurement step
Form a set of σ-points, $\mathcal{X}_k$
$\hat{\mathbf{y}}_{k|k-1} \approx \sum_{i}^{2n+1} h\left(\mathcal{X}_k^{(i)}\right) W_i$
$\mathbf{P}_{xy} \approx \sum_{i}^{2n+1} \left(\mathcal{X}_k^{(i)} - \hat{\mathbf{x}}_{k|k-1}\right)\left(h\left(\mathcal{X}_k^{(i)}\right) - \hat{\mathbf{y}}_{k|k-1}\right)^T W_i$
$\mathbf{S}_k \approx \mathbf{R}_k + \sum_{i}^{2n+1} \left(h\left(\mathcal{X}_k^{(i)}\right) - \hat{\mathbf{y}}_{k|k-1}\right)(\cdot)^T W_i$
$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{P}_{xy}\mathbf{S}_k^{-1}\left(\mathbf{y}_k - h\left(\hat{\mathbf{x}}_{k|k-1}\right)\right)$
$\mathbf{P}_{k|k} = \mathbf{P}_{k|k-1} - \mathbf{P}_{xy}\mathbf{S}_k^{-1}\mathbf{P}_{xy}^T$

Algorithm 3: Uncented Kalman filter algorithm

The algorithm for this filter is the same as for the UKF shown in algorithm 3, but with one less sigma point as mentioned before. The performance is similar for CKF and UKF, due to the similar algorithms. CKF does however not suffer from the curse of dimensionality nor experience divergence as EKF and UKF does [17].

2.3.4 Particle filter

The particle filter is an alternative nonparametric implementation of Bayes filter [19]. The PF estimates the posterior distribution by a finite number of parameters. The idea of the PF is to represent the posterior $p(\mathbf{x}_k \mid \mathbf{y}_{1:k})$ by a set of random state samples, referred to as particles, drawn from the posterior. Every particle is a hypothesis of the true state at time k. The likelihood for a state hypothesis $\mathbf{x}_k^{(i)}$ to be drawn as a sample should ideally be proportional to the posterior. The PF is better to use when the models are highly nonlinear and/or the posterior distribution is significantly non-Gaussian, for example a multimodal density. Unfortunately PF does suffer from the curse of dimensionality and are intractable in higher dimensions [19].

2.4 Data association

In general, the problem of data association is determining to which property a measurement corresponds to. This can be associating a measurement to a certain object when having multiple objects from which measurements can be observed from.

In this thesis the problem of data association stems from the need of finding the correct map RSM for a detected RSM from the camera, in order to give information about how the pose should be updated. A detected RSM can theoretically

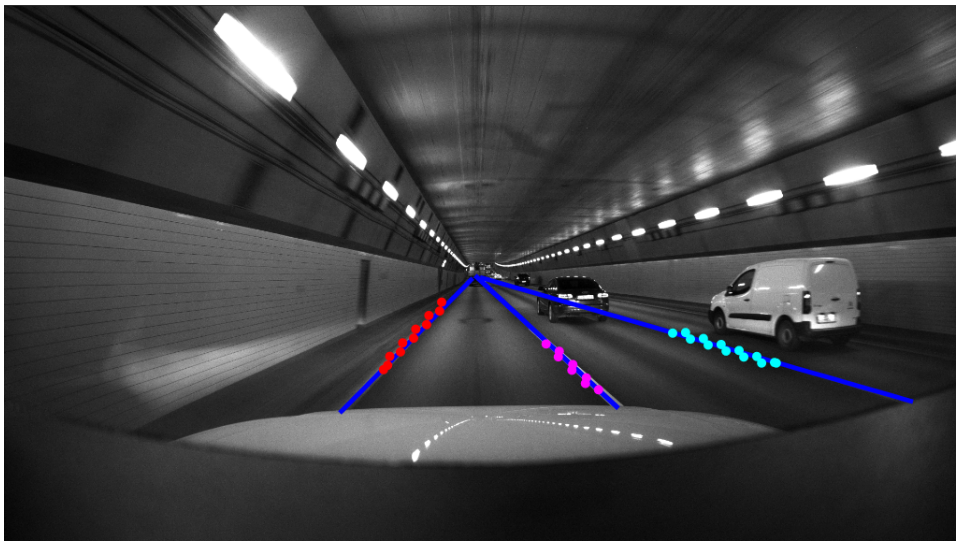

Figure 2.1: Camera image with an illustration of the HD map projected into the image where the blue lines are the HD map, and an illustration of three different RSMs detected by the camera represented by red, purple and cyan dots. Note that this is not the actual HD map nor the actual detected RSMs, but illustrations showing the concept.

belong to any of the RSMs present on the current driving road, but to be used as a measurement it is necessary to determine which RSM on the road that is detected. Figure 2.1 illustrates the data association setting in this thesis, where it is shown how the map is transformed in to the image frame and the detection of RSMs by the front facing mono camera.

The red dots in Figure 2.1 are measurements from an RSM on the road detected in the image frame. The red dots can belong to either one of the blue lines, the map RSMs. In order to pair the red dots to the correct map RMS, which in this case is the leftmost blue line, a data association technique is necessary. In this section some general methods for data association are described.

2.4.1 Nearest Neighbour

A very simple association technique can in some cases be both useful and effective, which is shown in [20, 21] where the Nearest Neighbour algorithm is implemented. The algorithm associates by simply comparing a measurement with the predicted measurement of all possible state distributions. The state distribution that has the closest predicted measurement in relation to the actual measurement is chosen as the best association.

In the setting of this thesis, the Nearest Neighbour algorithm compares the distance in pixels between a detected RSM and all of the map RSM. The map RSM that has the shortest distance in pixels to the detected RSM is chosen as the best association.

2.4.2 Normalized Innovation Squared

A property that can be used for determining fitness of a data association is the normalized innovation squared [22], which is a property used in Gaussian filters. In this thesis, the normalized innovation squared is calculated with equations used in an EKF. A low normalized innovation squared value indicates a correct association. The innovation function is defined as

$$\mathbf{v} = \mathbf{y} - h(\mathbf{x}), \tag{2.20}$$

where v is the innovation, y is the measurement and $h(\mathbf{x})$ is the predicted measurement. To compute the normalized innovation squared, firstly the innovation covariance must be computed as

$$\mathbf{S}_{i,j_i} = \mathbf{H}_{i,j_i}\mathbf{P}\mathbf{H}_{i,j_i}^T + \mathbf{R}_i \tag{2.21}$$

where $\mathbf{S}_{i,j_i}$ is the innovation covariance for measurement i associated to state j_i, $\mathbf{H}_{i,j_i}$ is the jacobian of the innovation function v, $\mathbf{P}$ is the state covariance matrix and $\mathbf{R}_i$ is the measurement covariance matrix. The normalized innovation squared is then computed as

$$D_{i,j_i}^2 = \mathbf{v}_{i,j_i}^T \mathbf{S}_{i,j_i}^{-1} \mathbf{v}_{i,j_i} \tag{2.22}$$

where D_{i,j_i}^2 is the normalized innovation squared value for measurement i associated with state j_i and $\mathbf{v}_{i,j_i}$ is the innovation for measurement i associated with state j_i. The normalized innovation squared is in literature sometimes referred to as the *Mahalanobis distance* [23]. In the setting of this thesis a state j_i corresponds to the position of a certain map RSM.

2.4.3 Individual Compatibility Nearest Neighbour

An algorithm that is using the normalized innovation squared for association is Individual Compatibility Nearest Neighbour presented in [24]. The level of fitness for a measurement belonging to a certain state is evaluated by calculating the normalized innovation square for each measurement and each state. The computed value is then subject to a gating test called Individual Compatibility test. It determines if a measurement and state pair should be viewed as an outlier or not based on the fact the the normalized innovation squared value has a χ^2 distribution. To pass the Individual Compatibility test the following inequality must hold

$$D_{i,j_i}^2 < \chi_{d,\alpha}^2 \tag{2.23}$$

where $\chi_{d,\alpha}^2$ is an upper bound for which a random sample of the χ^2 distribution should be lower than with a probability of α. The input d is referred to as the degrees of freedom for the χ^2 distribution, which here is the dimension of the innovation function. For each measurement, the association which yields both a passed test and the lowest value for the normalized innovation squared normalized with the gating value, $\frac{D_{i,j_i}^2}{\chi_{d,\alpha}^2}$, is chosen. If no association passes the test the measurement is classified as an outlier.

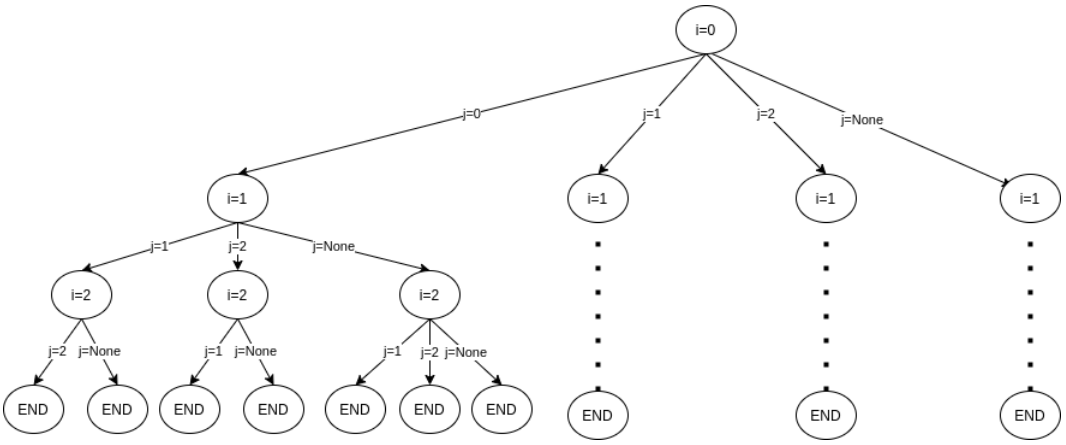

Figure 2.2: Illustration of the tree structure in the Joint Compatibility Branch and Bound algorithm

When used in this thesis, the measurements are not associated to different states but instead to different map RSMs. However, exactly the same procedure is used with the only difference being that instead of using different states in the innovation function, different map RSMs is used.

2.4.4 Joint Compatibility Branch and Bound

The Joint Compatibility Branch and Bound algorithm presented in [24] builds on top of the association method described in Section 2.4.3, but instead of finding the best match individually for each measurement, it aims to find the optimal associations jointly for all measurements. This implementation is often treated as the gold standard in association implementations [25]. The algorithm views the data association problem as an interpretation tree, where each leaf of the tree corresponds to one measurement paired with a state. Multiple measurements can not be associated to the same state.

An illustration of the tree can be seen in Figure 2.2, where the nodes $i = 0, 1, .., n$ represents n measurements, and the edges coming out from these nodes represents associating the measurements to m states $j = 0, 1, .., m$. At every level of the tree the possibility of not assigning a measurement to a state exists, which is equal to seeing the measurement as an outlier. To match a measurement anywhere in the tree with a state, the pair must first pass the Individual Compatibility test. If the pairing passed the first test, it is evaluated in a Joint Compatibility test which evaluates the compatibility of the new pairing with all previous accepted measurement and state pairs in the current branch of the tree. The algorithm aims to find the association hypothesis with the largest number of jointly compatible measurements. A hypothesis is in Figure 2.2 a path from $i = 0$ to any "END"-node.

To find the largest set of jointly compatible measurements with corresponding states, a hypothesis is constructed. Starting with one single measurement and state pair

the test is equal to Individual Compatibility test, and if the pair passed it is added to the hypothesis. Once the hypothesis has at least one pairing, new measurements that passed the individual test is evaluated together with the current hypothesis. The joint innovation function is constructed as

$$\mathbf{v}_{\mathcal{H}_i} = \begin{bmatrix} \mathbf{v}_{\mathcal{H}_{i-1}} \\ \mathbf{v}_{i,j_i} \end{bmatrix} = \begin{bmatrix} \mathbf{v}_{1,j_1} \\ \vdots \\ \mathbf{v}_{ij_i} \end{bmatrix} \tag{2.24}$$

where $\mathbf{v}_{\mathcal{H}_i}$ is the joint innovation for hypothesis $\mathcal{H}$ up to measurement i. The joint jacobian of the innovation function is constructed using the same method.

$$\mathbf{H}_{\mathcal{H}_i} = \begin{bmatrix} \mathbf{H}_{\mathcal{H}_{i-1}} \\ \mathbf{H}_{ij_i} \end{bmatrix} = \begin{bmatrix} \mathbf{H}_{1j_1} \\ \vdots \\ \mathbf{H}_{ij_i} \end{bmatrix} \tag{2.25}$$

As can be seen in equations (2.24) and (2.25) both moments are constructed by stacking the individual moments vertically.

$$\mathbf{S}_{\mathcal{H}_i} = \begin{bmatrix} \mathbf{H}_{\mathcal{H}_{i-1}} \\ \mathbf{H}_{ij_i} \end{bmatrix} \mathbf{P} \begin{bmatrix} \mathbf{H}_{\mathcal{H}_{i-1}}^T & \mathbf{H}_{ij_i}^T \end{bmatrix} + \begin{bmatrix} \mathbf{R}_{\mathbf{H}_{i-1}} & 0 \\ 0 & \mathbf{R}_i \end{bmatrix} \tag{2.26}$$

The joint innovation covariance can be computed as shown in equation (2.26), where $\mathbf{R}_{\mathcal{H}_{i-1}}$ is the measurements covariance matrix for all measurements in hypothesis $\mathcal{H}$ up to measurement $i-1$.
The normalized innovation squared value for all measurements with associated map elements in hypothesis $\mathcal{H}_i$ is computed as

$$D^2_{\mathcal{H}_i} = \mathbf{v}^T_{\mathcal{H}_i} \mathbf{S}^{-1}_{\mathcal{H}_i} \mathbf{v}_{\mathcal{H}_i}. \tag{2.27}$$

Much like the Individual Compatibility test the jointly computed normalized innovation squared value is evaluated against a threshold from the χ^2 distribution,

$$D^2_{\mathcal{H}_i} < \chi^2_{d,\alpha}. \tag{2.28}$$

To find the largest set of jointly compatible associations the search algorithm Branch and Bound is used. It searches the interpretation tree by using the number of current jointly compatible associations plus the future possibly usable points as heuristics.

The Branch and Bound search is initialized with two empty containers, hypothesis $\mathcal{H}$ and the best hypothesis Best_$\mathcal{H}$. Theses are fed to the algorithm together with the first measurement, and the measurement is evaluated against the map elements to find associations that is both individually compatible and jointly compatible with the current hypothesis. When a compatible association is found, a recursive call to the algorithm is done with hypothesis $\mathcal{H}$ extended with the new association, and also the next measurement in line. If no compatible association was found for the current measurement, the measurement is treated as an outlier for the current hypothesis. A recursive call to the algorithm is done if the hypothesis containing the outlier still has a possibility to be the best hypothesis, that is it has the possibility to have the highest number of jointly compatible associations.

Input: hypothesis $\mathcal{H}$, detection number i, number of detections m, number of map RSMs n
Result: Best_$\mathcal{H}$

```
if i > m then
    if num_pairings(H) > num_pairings(Best_H) then
        Best_H = H
        return Best_H
    end
else
    for j = 1 to n do
        if individually_compatible(i, j) as equation (2.23) then
            if jointly_compatible(H, i, j) as equation (2.28) then
                JCBB([H j], i+1,m,n)
            end
        end
    end
    if num_pairings(H) + m - i > num_pairings(Best_H) then
        JCBB([H 0], i+1,m,n)
    end
end
return Best_H
```

Algorithm 4: Joint compatibility branch and bound

2.5 Coordinate system

In this thesis different coordinate systems are used, which is explained in the sections below. However, to give an overview of what purposes the different coordinate systems serve, a brief overview will be presented. It can be said that the global geographical system is used for initializing the localization system. In the global Cartesian system the position of the vehicle is tracked, which means that all states in the localization filter operates in this frame. The ego-vehicle frame is a frame in which all of the vehicles inertial sensors give their measurements. Finally, the camera coordinate system is a coordinate frame used when projecting into the image plane and where the camera measurements operate in.

2.5.1 Global geographical system

The global geographical system is a widely used coordinate system for describing position on the entire globe. A position in this coordinate frame consists of three values, longitude, latitude and altitude. Longitude is the angle between the Greenwich Meridian and the position, where an angle to the west of the Greenwich Meridian is between 0 and -180 degrees and an angle to the east is between 0 and 180 degrees. The latitude is defined as the angle between the position and the equator, along the north-south axis. The latitude angle ranges from -90 to 90 degrees. The altitude is the height to an ellipsoid approximating the surface of the earth, in this case the

WGS84 standard is used.

2.5.2 Local Cartesian system

A local Cartesian system is a conversion of the previously mentioned global geographical system. Instead of having the the coordinate system expressed in spherical coordinates as in the Global Geographical system, it is expressed with an x-, y- and z-coordinate.

In this thesis global geographical coordinates is transformed to a local Cartesian system, in order to simplify calculations. The local Cartesian system is an East-North-Up (ENU) system, which means that is has its x-axis pointing to east, y-axis pointing to north, and z-axis pointing up from the earth's surface.

2.5.3 Ego-vehicle frame

Since the vehicle contains inertial sensors giving measurements in a so called ego-vehicle coordinate system, the properties of this frame must be known as well. The ego-vehicle coordinate system is defined according to the standard ISO 8855 [26]. The x-axis is directed in the forward direction of the vehicle, the y-axis points to the left of the vehicle and the z-axis point upwards.

2.5.4 Camera coordinate system

The camera coordinate system is a coordinate frame used in calculations regarding the camera. It has its z-axis pointing forward in the camera direction, x-axis to the right of the camera, and y-axis points downwards.

2.5.5 Homogeneous coordinates

In projective geometry homogeneous coordinates is a method of transforming between a three dimensional coordinate system and a two dimensional projective space. A homogeneous coordinate is a coordinate that represents a projection line in space, and if multiplied with a nonzero scalar it still represents the same projection line. The nonzero scalar does however determine the position along the projection line.

To convert a coordinate from a Cartesian coordinate system to homogeneous coordinates, the Cartesian coordinate is extended with a 1 in the fourth dimension, which works as a scaling factor. Given a point in a three dimensional Cartesian space, it can be converted to homogeneous coordinates as

$$p = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \implies p_h = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}, \tag{2.29}$$

where p_h is the point p expressed in homogeneous coordinates. The point p_h can be multiplied with a nonzero scalar as

$$\lambda p_h = \begin{bmatrix} \lambda x \\ \lambda y \\ \lambda z \\ \lambda \end{bmatrix}, \tag{2.30}$$

where it can be seen that after being multiplied with the scalar λ, the homogeneous coordinate p_h still represents the same point but with a scaling factor. The scaling factor can easily be removed by normalizing with the value in the fourth dimension. Homogeneous coordinates are also easy to use when handling transformations between different coordinate systems, since the transformation can be executed by using one single matrix multiplication. A general transformation matrix M containing rotation and translation is constructed as

$$M = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \mathbf{0} & 1 \end{bmatrix}, \tag{2.31}$$

where $\boldsymbol{R}$ is a rotational matrix and $\boldsymbol{t}$ is the translation in space expressed as a column vector.

As mentioned, homogeneous coordinates can be used for projecting three dimensional objects into a projective space, such as an image. When projecting a 3D point into an image, the point is extended with a 1 in the fourth dimension, and then multiplied with the camera matrix.

$$\begin{bmatrix} a \\ b \\ \lambda \end{bmatrix} = \mathbf{K} \begin{bmatrix} \mathbf{R}_c & \mathbf{t}_c \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}. \tag{2.32}$$

In equation (2.32) a and b are projected coordinates in image space, $\mathbf{K}$ is the intrinsic camera matrix, R_c is the rotation matrix from the coordinate system where the 3D point is expressed to the cameras own coordinate system and t_c is the translation between the two coordinate frames. To retrieve the pixel values for a 3D point that has been projected into image space, the left hand side in equation (2.32) must be divided with its third coordinate, as

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{a}{\lambda} \\ \frac{b}{\lambda} \\ 1 \end{bmatrix}. \tag{2.33}$$

3
Method

In this chapter the developed system is described in detail. The system is referred to as *The localization framework*. First an overview of the system is presented in Section 3.1, including a description of the sensors that is used. In Section 3.2 implementation details regarding the localization filter is presented. Lastly, in Section 3.3 three solutions to the data association problem is described in detail.

3.1 System overview

In Figure 3.1 the concept of *The localization framework* is shown. The sensor fusion filter sequentially estimates the position and orientation of the vehicle by making use of a motion model together with various measurements.

The localization framework has to find a safe place to initialize on, before it can start, denoted as *Initialization* in Figure 3.1. The safe initialization spot is defined as that there is no roads above or beneath the current driving road. This makes it possible for *The localization framework* to initialize on the correct lane before the filter is started, denoted as *Initial state*. The initial state is given by a highly accurate GNSS/INS, namely an OXTS RT3000. An initial uncertainty matrix is also used to initialize the system, which is treated as a tuning parameter. The OXTS RT3000 is more accurate than an ordinary GPS normally found in cars [27], but is still prone to errors especially in altitude. The OXTS RT3000 is also used as ground truth data later on in the evaluation of the system.

When the system is initialized with an initial distribution of the vehicles position and orientation the filter recursively estimates the states as algorithm 1. Next step is to predict the state distribution for the next time step, through the *Prediction step* in Figure 3.1. *The localization framework* has three separate update steps. The first update step, denoted as *Update step - ego vehicle* in Figure 3.1, makes use of the ego vehicle sensors to update the speed, acceleration and yaw rate. With the updated state information the map is ran through a preprocessing step where unnecessary map RSMs are removed, denoted as *Map preprocessing*. The preprocessed map is used to get measurements of the roll, pitch and height in the *Map measurement*, which is then used to update those state values in the *Update step - map measurement*. This same map is then used together with the detected RSMs from the camera and the current estimated state to associate the detected RSMs to the corresponding map RSMs, in the *Association* step. This relationship is used to

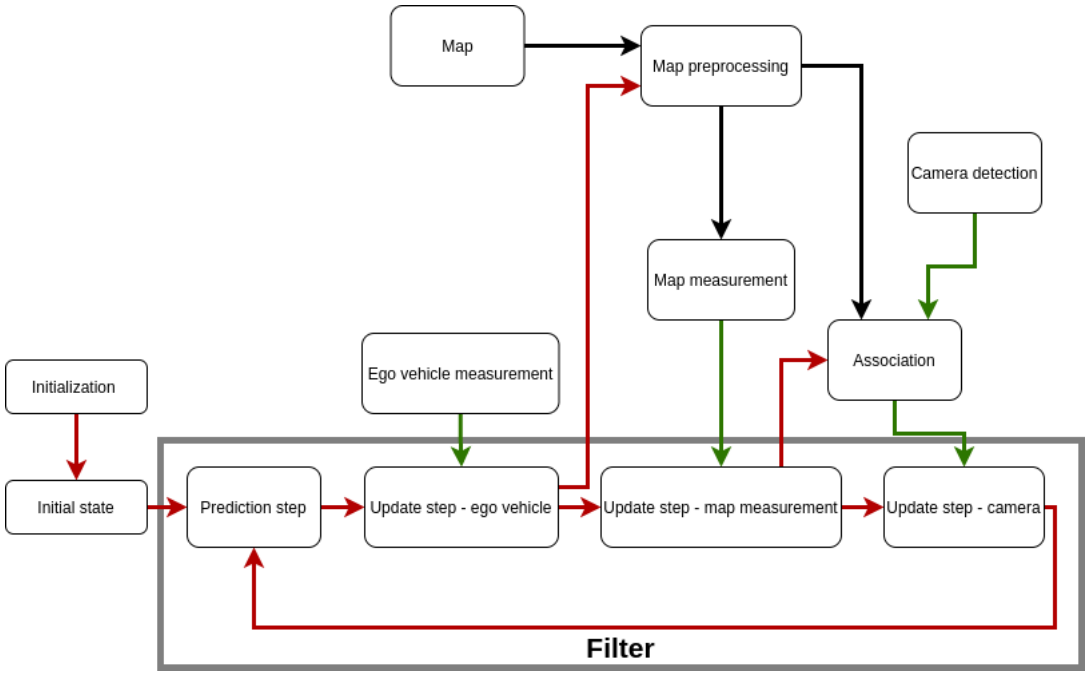

Figure 3.1: Flowchart of *The localization framework*. The filter notation shows where the typical filter techniques are used. The red arrows shows the flow of the current state values. The green arrows indicates what is treated as measurements. Black arrows are the flow of map information. Numbers show the order of the events in the flow

update the states in *Update step - camera*. The updated state is fed back to *The localization framework*s *Prediction step* in order to produce a prior distribution for the next time step.

3.1.1 Sensor setup

The sensors are a vital part of *The localization framework*. They are the base for how a system can perceive the world and set to some extent the limit for the achievable accuracy. *The localization framework* uses a set of different sensors: wheel speed and acceleration sensors that measures speed and acceleration in longitudinal and lateral direction, a yaw rate sensor, a prerecorded 3D map and a front looking monocular camera combined with the 3D map. All of these sensors have their strengths and weaknesses, and by fusing all the measurements a more robust estimate of the position is possible. All sensors have different sampling frequencies, but this is not a problem handled in the thesis. The data provided by the company has been sampled accordingly by the company.

3.1.2 HD map

The HD map used in this thesis contains a three dimensional map consisting of polylines representing the RSMs. The map is not developed in this project, but an already existing map is used instead. Since working with a map that cover large

areas is not efficient, it is possible to retrieve a smaller area around some geodetic coordinate. When the localization algorithm is initialized, a smaller map segment is retrieved around the starting point of the vehicle. The starting point is a geodetic coordinate, and the map returns its map RSMs in a local Cartesian coordinate system with its origin in the starting point. The retrieving and rotation of the map is not a part of this thesis, and was implemented in advance. This local system in the first iteration of the algorithm is set to be the local Cartesian coordinate system that the filter operates in, and when new small segments of the map is retrieved at other iterations of the algorithm, they are transformed to this initial coordinate frame. The rate of fetching a new map is chosen such that the vehicle is not able to move outside the current available map.

3.1.3 Front facing mono camera

The camera and the map can be used as a sensor when combined. A front facing camera is utilized in *The localization framework* for detecting RSMs on the road. The camera images are processed by a RSM detection algorithm based on deep learning techniques. The detected RSMs are prone to outliers as well as multiple detections of the same map RSM. The RSMs in the smaller area retrieved by the map is compared with the RSMs detected by the camera to produce a measurement that can be related to the state space.

3.2 Localization Filter

The heart of *the localization framework* is its sensor fusion filter, namely a CKF. Implementation details regarding the localization filter is described in this section, together with a short motivation regarding the filter choice.

3.2.1 State space

The state space used in the filter is chosen based on the available sensors and the 6 degrees of freedom of a car.

$$\mathbf{x}_k = \begin{bmatrix} x_k & y_k & z_k & \phi_k & \theta_k & \psi_k & \dot{x}_k & \dot{y}_k & \dot{\psi}_k & \ddot{x}_k & \ddot{y}_k \end{bmatrix}^T \tag{3.1}$$

x_k, y_k and z_k describe the position in the local Cartesian coordinate system, ϕ_k, θ_k and ψ_k are roll, pitch and yaw angles describe in the east-north-up standard. The variables with dots above them are first and second order time derivatives of previously explained quantities.

To clarify the angular states further, the definition of how they are used is presented. ϕ, θ and ψ represent the orientation of the vehicle, which are used to compute the rotation matrix from the East-North-Up local Cartesian system and the ego-vehicle system. This rotation matrix is built by multiplying three rotation matrices, each representing a rotation around one of the coordinate systems axes. The first rotation

matrix is computed using ϕ as

$$\mathbf{R}_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\phi) & -sin(\phi) \\ 0 & sin(\phi) & cos(\phi) \end{bmatrix}, \tag{3.2}$$

and represents a rotation around the x-axis. The second rotation matrix is computed using θ as

$$\mathbf{R}_y = \begin{bmatrix} cos(\theta) & 0 & sin(\theta) \\ 0 & 1 & 0 \\ -sin(\theta) & 0 & cos(\theta) \end{bmatrix}, \tag{3.3}$$

which represents a rotation around the y-axis. Finally, the third rotation matrix is computed using ψ as

$$\mathbf{R}_z = \begin{bmatrix} cos(\psi) & -sin(\psi) & 0 \\ sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{3.4}$$

and represents a rotation around the z-axis. The rotations are applied in the order z, y, x to compute the final rotation as

$$R_{ego,ENU} = \mathbf{R}_z\mathbf{R}_y\mathbf{R}_x, \tag{3.5}$$

which describes the rotation from the local Cartesian coordinate system to the ego-vehicle coordinate system.

3.2.2 Vehicle dynamics

To allow the filter to transform from one time instance to the next a prediction step is used to calculate a prior distribution of the state space. This is referred to as the *Prediction step* in algorithm 3. A mathematical model is used to predict how the states have changed from one time step to the next, which gives a prediction of the states. There are various different models used in vehicle applications, presented in Section 2.1. In [14], the best model for urban and highway scenarios was concluded to be the CTRA model. However, the model used in these applications tends to vary [13, 28, 7, 29]. To choose the most suitable model for this thesis, testing was performed. Since this is not a central part of the thesis, testing results will not be presented showing how the different models affected the algorithm. The different motion models tuning parameters was tuned using a small subset of the dataset, and evaluated on the entire dataset. Based on the evaluation the model Constant Acceleration was chosen since it produces the best results, see appendix A.

3.2.3 Ego-vehicle measurement

The first update step of the localization filter is integrating the measurements from the ego-vehicle sensors. The ego-vehicle measurement consists of velocity and acceleration, in both longitudinal and lateral direction in ego-vehicle frame, as well as a measurement of the yaw rate. The yaw rate does not need to be transformed to any other coordinate frame since the z-axis is assumed the same in both the local

Cartesian system and the ego-vehicle coordinate system in this setup. For the velocity and acceleration however, it is necessary to have an observation model that relates the state vector to the measurements by rotating it from the local Cartesian system to the ego-vehicle coordinate system. The velocity is rotated as

$$\begin{bmatrix} \dot{x}_{ego} \\ \dot{y}_{ego} \\ 0 \end{bmatrix} = \mathbf{R}_{ego,ENU} \cdot \begin{bmatrix} \dot{x} \\ \dot{y} \\ 0 \end{bmatrix}. \tag{3.6}$$

where $\dot{x}_{ego}$ is the x position in ego-vehicle frame, $\dot{y}_{ego}$ is the y position in ego-vehicle frame and $\mathbf{R}_{ego,ENU}$ is the rotation matrix from local Cartesian coordinate system to ego-vehicle coordinate system. $\mathbf{R}_{ego,ENU}$ is calculated by using the current estimate of the position and orientation. The acceleration states $\ddot{x}$ and $\ddot{y}$ is rotated exactly the same way. The observation function for the ego-vehicle sensors is constructed by three different observation functions, one for each sensor. The velocity observation function is

$$h_{vel}(\dot{x}, \dot{y}) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \cdot \mathbf{R}_{ego,ENU} \cdot \begin{bmatrix} \dot{x} \\ \dot{y} \\ 0 \end{bmatrix}, \tag{3.7}$$

the observation function for the acceleration sensor is

$$h_{acc}(\ddot{x}, \ddot{y}) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \cdot \mathbf{R}_{ego,ENU} \cdot \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ 0 \end{bmatrix}, \tag{3.8}$$

and finally the observation function for the yaw rate is simply

$$h_{yawrate}(\psi) = \dot{\psi}. \tag{3.9}$$

The final observation function for all ego-vehicle sensors is

$$h_{ego}(\dot{x}, \dot{y}, \ddot{x}, \ddot{y}, \psi) = \begin{bmatrix} h_{vel}(\dot{x}, \dot{y}) \\ h_{acc}(\ddot{x}, \ddot{y}) \\ h_{yawrate}(\psi) \end{bmatrix}. \tag{3.10}$$

The ego-vehicle sensors are assumed to be affected by Gaussian noise. This is modeled by further assuming that all measurement noise is uncorrelated, which allows for describing the uncertainty with a single parameter for each measurement, namely the standard deviation of the noise. The measurement uncertainty matrix is expressed by

$$R = \begin{bmatrix} \sigma^2_{vel,lon} & 0 & 0 & 0 & 0 \\ 0 & \sigma^2_{vel,lat} & 0 & 0 & 0 \\ 0 & 0 & \sigma^2_{acc,lon} & 0 & 0 \\ 0 & 0 & 0 & \sigma^2_{acc,lat} & 0 \\ 0 & 0 & 0 & 0 & \sigma^2_{yawrate} \end{bmatrix}, \tag{3.11}$$

where $\sigma_{vel,lon}$ represents the standard deviation for the velocity in longitudinal direction, $\sigma_{vel,lat}$ represents the standard deviation for the velocity in lateral direction,

$\sigma_{acc,lon}$ represents the standard deviation for the acceleration in longitudinal direction, $\sigma_{acc,lat}$ represents the standard deviation for the acceleration in lateral direction, and finally $\sigma_{yawrate}$ represents the standard deviation for the yawrate. All of the standard deviations described above are tuning parameters but often the tuning parameters can be set with information from the sensor manufacturer.

3.2.4 Map measurement

Since there is no complete IMU in the sensor setup, and the roll and pitch angles are important states in *The localization framework*, these states have to be estimated in other ways. This is also the case with the z value. There are no measurements of the acceleration or velocity in the z-direction, this state have to be estimated in other ways. The proposed solution to this problem in this system is to use the map for gathering measurements for these state variables. To extract the measurements, the left and right RSMs of the current driving lane need to be found. To find the left and right RSMs in relation to the current position of the car the position is orthogonally projected to all RSM segments in all map RSMs, using

$$t = \frac{(a-p)\cdot(b-a)}{(b-a)\cdot(b-a)} \tag{3.12a}$$

$$\hat{p} = a + t\cdot(a-b). \tag{3.12b}$$

Here p is the point to be projected to the map RSM. a is the start coordinates of the RSM segment and b is the end coordinate of the RSM segment. $\hat{p}$ is the point p projected on to the map RSM.

Since the position is projected to all the RSM segments in all map RSMs, the closest point from each map RSM is chosen, to find the closest point on each map RSM. The two closest RSMs on each side of the vehicle is chosen by

$$\mathbf{a}\cdot\mathbf{b} < 0, \tag{3.13}$$

where $\mathbf{a}$ is the vector from the vehicle position to the closest point on the map RSM and $\mathbf{b}$ is the vector from the vehicle position to the second closest map RSM. If this is not the fulfilled the next closest point is tested in the same way. When these closest RSMs have been found the mean of the height of the RSMs plus the vehicle height offset is considered as the height measurement. These two closest map RSMs is also used to calculate the roll ϕ of the road, which is assumed to be the same as to the roll of the vehicle, as illustrated in Figure 3.2.

The end points of the black line in Figure 3.2 is the two closest map RSMs, the black line illustrates the road between those map RSMs. When these distances are calculated simple trigonometry is used to calculate the roll. The pitch θ is calculated similarly, see Figure 3.3.

When calculating the pitch both RSMs are used individually and the mean of these

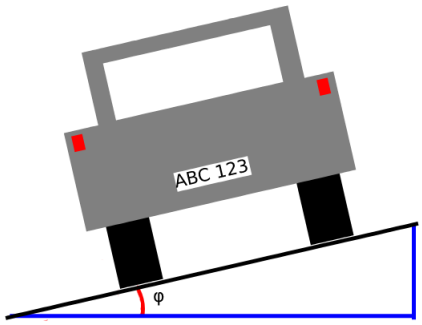

Figure 3.2: Illustration of how the roll is calculated for the map measurements update step

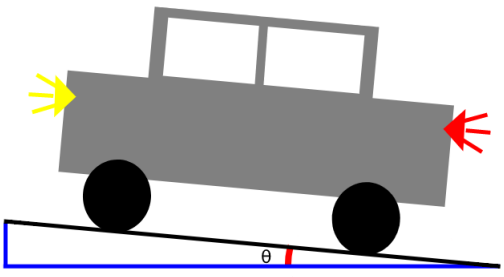

Figure 3.3: Illustration of how the pitch is calculated for the map measurements update step. The car is heading to the left in the image.

two pitches is then considered as the measurement of the pitch of the vehicle. The measurement function is then as

$$h_{map}(z_k, \phi_k, \theta_k) = \begin{bmatrix} z_k \\ \phi_k \\ \theta_k \end{bmatrix}. \tag{3.14}$$

The map measurements are assumed to be affected by Gaussian noise. This is modeled by assuming that all measurement noise is uncorrelated, which allows for describing the uncertainty with a single parameter for each measurement, namely the standard deviation of the noise. The measurement uncertainty matrix is constructed as

$$R = \begin{bmatrix} \sigma_z^2 & 0 & 0 \\ 0 & \sigma_\phi^2 & 0 \\ 0 & 0 & \sigma_\theta^2 \end{bmatrix}, \tag{3.15}$$

where σ_z is the standard deviation for the noise in z, σ_ϕ is the standard deviation for the noise in ϕ, and σ_θ is the standard deviation for the noise in θ. Each parameter is considered a tuning parameter.

3.2.5 Camera measurement

For the update step used when the filter is given a camera sensor measurement, it is a bit less intuitive than with the ego-vehicle sensors. To retrieve information about the current pose of the vehicle, the detected RSMs from the camera is matched with the corresponding RSMs from the map. Before doing this, the detected RSMs must be associated with corresponding RSMs from the map. Association is performed by employing an association algorithm, which finds the most suitable match from the map for each detected RSM. By projecting the map RSMs in front of the current filter estimate of the position into the image, the association can be performed in the image frame. The map RSM that is associated to a certain camera detection is viewed as the detection's expected measurement. The different association techniques are further described in section 3.3.

Given a detected RSM from the camera that is associated to a map RSM, information about the current vehicle position and orientation is retrieved through an observation model. The observation model for the camera consists of several steps. The associated map RSM consists of two 3D points or more, and each point is projected into the image in the first step of the camera observation model. The projection is computed as

$$\begin{bmatrix} a_{map} \\ b_{map} \\ \lambda \end{bmatrix} = \mathbf{K} \cdot \mathbf{T}_{c,ego} \cdot \mathbf{T}_{ego,ENU} \cdot \begin{bmatrix} x_{map} \\ y_{map} \\ z_{map} \\ 1 \end{bmatrix}, \tag{3.16}$$

where a_{map} and b_{map} is unnormalized image coordinates, λ is the projection scaling factor, $\mathbf{K}$ is the camera intrinsic matrix, $\mathbf{T_{c,ego}}$ is a transformation matrix from ego-vehicle frame to the camera coordinate frame retrieved through calibration of

the camera setup, $\mathbf{T}_{\mathbf{ego,ENU}}$ is a transformation matrix from the local Cartesian coordinate frame to the ego-vehicle frame which is computed using the current estimate of the position and orientation, and x_{map}, y_{map} and z_{map} is a 3D point from the map RSM. The unnormalized image coordinates a_{map} and b_{map} is then normalized with the factor λ to retrieve the pixel values for the map point as

$$\begin{bmatrix} u_{map} \\ v_{map} \\ 1 \end{bmatrix} = \frac{1}{\lambda} \begin{bmatrix} a_{map} \\ b_{map} \\ \lambda \end{bmatrix}, \tag{3.17}$$

where u_{map} and v_{map} is the pixel values for the map point. The detected RSM from the camera consists of points in the image which when connected forms a 2D polyline. The projected map RSM in the image also consists of points which forms a 2D polyline, but in order to relate the two, every point in the detected RSM is orthogonally projected onto the 2D map polyline. Since the 2D map polyline can consist of multiple segments, the closest segment is chosen for each point in the detected RSM. The orthogonal projection for one detected RSM point onto a 2D map segment is computed as

$$t = \frac{(p_{map1} - p_{det}) \cdot (p_{map2} - p_{map1})}{(p_{map2} - p_{map1}) \cdot (p_{map2} - p_{map1})}, \tag{3.18a}$$

$$\hat{p}_{map} = p_{map1} + t \cdot (p_{map1} - p_{map2}). \tag{3.18b}$$

Here p_{det} is the point to be projected to the map RSM segment. p_{map1} is the start coordinates of the RSM segment and p_{map2} is the end coordinate of the RSM segment. $\hat{p}_{map}$ is the point p_{det} projected on to the map RSM. The expected measurement for a point from a detected RSM is $\hat{p}_{map}$, and the observation function for the camera computes $\hat{p}_{map}$ for each point in the detected RSM which is stacked in a vector.

The final observation function for a detection of arbitrary length is computed as

$$h_{cam}(\mathbf{M}, \mathbf{x}_k) = \begin{bmatrix} \hat{p}_{map,1}(\mathbf{M}, \mathbf{x}_k) \\ \vdots \\ \hat{p}_{map,n}(\mathbf{M}, \mathbf{x}_k) \end{bmatrix}, \tag{3.19}$$

where $\mathbf{M}$ is the 3D map and n is the number of points in the detected RSM. The final step of the camera observation function is illustrated in Figure 3.4. Not all points will be able to orthogonally project to the lane, and this is handled by simply removing these points.

For the camera sensor, the noise modeling is a bit complicated because of the fact that it involves the uncertainty in the RSM detection algorithm, the uncertainty of the map as well as the uncertainty in the camera itself. To simplify the noise modeling for the camera sensor, all points in one detected RSM are assumed to be affected by uncorrelated noise. Therefore, the measurement uncertainty matrix for a measurement consisting of only one point can be described as

$$R = \begin{bmatrix} \sigma_{cam}^2 & 0 \\ 0 & \sigma_{cam}^2 \end{bmatrix}, \tag{3.20}$$

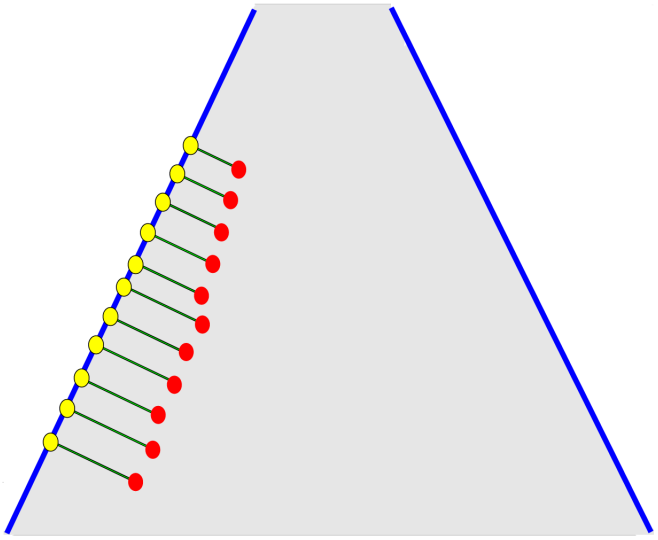

Figure 3.4: Illustration of the road, grey, and the reference map RSMs on each side of the road, blue, seen in camera frame. The red dots are the measurement from the front looking mono camera. The yellow dots are the orthogonally projected detections on to the closest map RSM. The green lines between the dots are what is considered as the innovation in the CKF.

where σ_{cam} is a tuning parameter representing the standard deviation of the camera noise.

3.2.6 Filter choice

For estimating the position and heading a CKF is implemented, due to its ability to handle nonlinearities as well as its computational efficiency as presented in section 2.3.3. Since the observation model for the camera and map combined is highly nonlinear the Extended kalman filter is deemed unfit. Both the PF and UKF has been proven to work in similar setups [7, 29, 4, 30]. However the CKF filter has not been as widely tested nor implemented in this field. This may be due to the fact that it is a fairly new filter technique, first presented in 2009 [17]. Due to the similarities to the UKF, a CKF implementation to this problem should be possible, while make use of the benefits of a CKF compared to an UKF as mentioned section 2.3.3.

3.3 Data association

Three methods for association is implemented in *The localization framework*, described in further detail in sections 2.4.1, 2.4.3 and 2.4.4. The Nearest Neighbour method is in its simplicity varying slightly between implementations and needs to be adapted to this system, described in Section 3.3.1. Both compatibility methods are originally developed to be used in a Simultaneous Localization And Mapping system, which has its differences from the system developed in this project. Therefore both methods needs to be adapted to fit the current problem, which is described in Sections 3.3.2 and 3.3.3.

3.3.1 Nearest Neighbour

This simplest association algorithm implemented in *The localization framework* is called Nearest Neighbour which is widely implemented in literature [4, 5, 31], described in section 2.4.1. The implementation is shown in algorithm 5.

By calculating the mean distance from all points in a detected RSM to a map RSM, a score for this particular association is retrieved. The distance from a detected point and a map RSM is calculated by projecting the point onto the map RSM, and then calculating the norm of the difference between the two points. The detected RSM and map RSM pair with the lowest score is considered the nearest association. This is done for all detections and all map RSMs. If one detected point can not be orthogonally projected onto a map RSM the score will be given a penalty of σ_{cam}, which is the standard deviation of the noise parameter for the camera. To make it a bit more robust it has a simple outlier rejection threshold, given in pixels, which will remove associations with a larger score than that threshold.

Data: In: image_map_RSMs, detected_RSMs, σ_cam
Result: Nearest associations

```
for det_RSM in detected_RSMs do
    lowest_score = infinity
    for map_RSM in image_map_RSM do
        temp_score = 0
        for det_point in det_RSM do
            point_on_line = orthogonally projection of detection point onto
             the map_RSM
            if points could be projected then
                temp_score = temp_score +
                 norm(point_on_line-det_point)
            else
                temp_score = temp_score + σ_cam
            end
        end
        score = temp_score / amount of projected points
        if score < lowest_score then
            lowest_score = score
            best_association = map_RSM
        end
    end
    if lowest_score > outlier threshold then
        Nearest associations[detection] = None
    else
        Nearest associations[detection]=best_association
    end
end
```

Algorithm 5: Nearest neighbour

3.3.2 Individual Compatibility Nearest Neighbour

The algorithm described in 2.4.3 considers measurements consisting of only one point in space. In the application presented in this thesis, the measurements are RSM detections consisting of multiple points. Therefore, the Individual Compatibility Test was tweaked to be able to handle multi point measurements. By following the same method as is proposed in the Joint Compatibility Test in section 2.4.4, the innovation function is constructed as

$$\mathbf{v} = \begin{bmatrix} \mathbf{v}_{1,j_1} \\ \vdots \\ \mathbf{v}_{n,j_n} \end{bmatrix} \tag{3.21}$$

where $\mathbf{v}$ is the innovation for a measurement that contains n points, where for example $\mathbf{v}_{1,j_1}$ is the innovation for detection point 1. The jacobian is constructed using the same method

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_{1,j_1} \\ \vdots \\ \mathbf{H}_{n,j_n}. \end{bmatrix} \tag{3.22}$$

Here $\mathbf{H}$ is the jacobian that contains n points, for example $\mathbf{H}_{1,j_1}$ is the jacobian for detection point 1. Using Equations 3.21 and (3.22) the innovation covariance is computed as

$$\mathbf{S} = \mathbf{H}\mathbf{P}\mathbf{H}^T + \mathbf{R}, \tag{3.23}$$

where R is the measurement uncertainty for the camera sensor. It is described in further detail in Section 3.2.5. Note that the innovation covariance used in association is computed using the EKF framework. Finally, the normalized innovation squared value for the detected RSM and map RSM pair can be computed as

$$D^2 = \mathbf{v}^T\mathbf{S}^{-1}\mathbf{v}. \tag{3.24}$$

An additional change was implemented to handle the case when a single point belonging to a detected RSM is an outlier in relation to the rest of the points in the detected RSM. If the innovation for a certain point is larger than $2\sigma_{cam}$, the point is discarded.

With these changes the Individual Compatibility Nearest Neighbour was implemented in *The localization framework*. The algorithm is very similar to the association algorithm described in algorithm 5, but with two differences. The first difference is that in Individual Compatibility Nearest Neighbour, the score of which an association is evaluated is the normalized innovation squared value instead of the projection distance. The second difference is that outlier rejection is performed by comparing the normalized innovation squared value with a χ^2 distribution threshold as

$$D_i^2 < \chi_{d,\alpha}^2, \tag{3.25}$$

instead of having a fixed threshold. d is the degrees of freedom and α is a tuning parameter.

3.3.3 Joint Compatibility Branch and Bound

Since the Joint Compatibility Branch and Bound also makes use of the Individual Compatibility Test, the same changes as described in Section 3.3.2 was applied. Furthermore, since the Joint Compatibility Test described in Section 2.4.4 already was made for evaluating multiple points, no changes was needed when implementing it on *The localization framework*.

Regarding the Branch and Bound search algorithm, some smaller changes were made when implementing it on *The localization framework*, these changes are marked with red text comparing to algorithm 4. As can be seen in Algorithm 6, an additional

Input: hypothesis $\mathcal{H}$, detection number i, number of detections m, number of map RSMs n
Result: Best_$\mathcal{H}$

```
1  if i > m then
2      if num_pairings(H) > num_pairings(Best_H) then
3          Best_H = H
4          return Best_H
5      end
6      if num_pairings(H) == num_pairings(Best_H) and joint_NIS(H) as
         equation (2.27) < joint_NIS(Best_H) as equation (2.27) then
7          Best_H = H
8          return Best_H
9      end
10 else
11     for j = 1 to n do
12         if individually_compatible(i, j) as equation (3.25) then
13             if jointly_compatible(H, i, j) as equation (2.28) then
14                 JCBB([H j], i+1,m,n)
15             end
16         end
17     end
18     if num_pairings(H) +num_remaining_possible_pairings >
         num_pairings(Best_H) then
19         JCBB([H 0], i+1,m,n)
20     end
21 end
22 return Best_H
```

Algorithm 6: Joint Compatibility Branch and Bound reworked

statement that prioritizes solutions with lower joint normalized innovation squared value score in case of equal number of pairings was added. Also, the statement that evaluates if a branch should be pruned or not when a detection has been classified an outlier was modified to handle detections containing multiple points. The *num_remaining_possible_pairings* on line 18 is calculated as the number of points remaining in the detections that yet not has been associated.

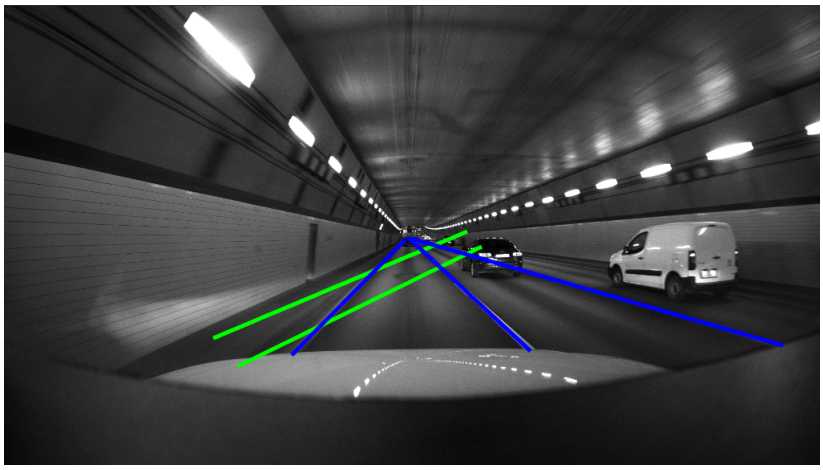

Figure 3.5: Camera image with an illustration of the HD map projected into the image. The blue lines are the HD map lanes of the current road the vehicle is driving on and the green lines are a the HD map lanes of a bridge beneath the current driving road. Note that this is not the actual HD map but illustrations showing the concept.

3.4 Map preprocessing

In order to make the problem of data association easier some preprocessing of the map is necessary. The map contains a lot of information, and this can be narrowed down to help the association algorithm associate the correct measurements with the corresponding map lane. As previously mentioned the map is initially preprocessed by only fetching a square of 200x200 meters around the vehicle. This limits the amount of map lanes the association algorithm has to evaluate drastically. To further narrow down the search of correct associations, map RSMs that are behind the vehicle and therefore not in the cameras field of view are removed. This removes not only map RSMs that are behind the vehicle but also entire RSMs that are far to the sides. The motivation for performing this additional preprocessing is simply because association is performed in the image frame, and since the camera looks straight ahead there is no need to consider map RSMs that lie behind the camera.

Furthermore, since the association is carried out in the image frame, there is a risk that faulty associations can be done when the map consists of multiple layers. For example, if there is a bridge on top of the road which the vehicle is currently driving on, both the bridge and the road is projected into the image as illustrated in Figure 3.5. If the bridge and the road are somewhat parallel, association is very hard since the bridge and the road would be indistinguishable from each other in the image plane.

To further remove useless and possible harmful parts of the map more preprocessing of the map is possible, now in terms of static occlusion. Static objects are in this thesis referring to multi layered roads, for example when there is a bridge beneath the road which the vehicle is driving on. The proposed method for doing this is

called Multi Layer Road Removal. These objects are removed in the map preprocessing stage in *The localization framework*, see Figure 3.1.

To remove these possible harmful roads the Multi Layer Road Removal algorithm first transforms the map into the ego-vehicle frame. This is done to be able to evaluate the height of the map lanes such that the z-direction represents up as the vehicle sees it. The vehicles position is then projected orthogonally onto all present map lanes. The magnitude of the values in the z-direction for these points are then individually evaluated against a predefined threshold. If the magnitude of the value in the z-direction for a point is larger than the threshold, the map lane it belongs to is removed from the map.

4
Results

All methods developed in this project are in this section evaluated using *The localization framework*. The evaluation is described in more detail in Section 4.1.

Regarding the results, firstly the effects of the Multi Layer Road Removal, from here on referred to as the MLRR method, are evaluated on each data association method separately. Secondly, the best versions of the different methods for data association is evaluated against each other. The results shown in the figures are each individual sequence RMSE value for the different methods. The results in the tables are the percentage of *good*, *ok* or *bad* bad scenarios for the different methods. The Nearest Neighbour is now be referred to as NN, the Individual Compatibility Nearest Neighbour is referred to as ICNN and the Joint Compatibility Branch and Bound is referred to as JCBB. Lastly, an evaluation of the uncertainty estimation for a sequence is presented.

4.1 Evaluation

To investigate how well the different features and solutions to this system performs the positional error is evaluated. The dataset that is used in testing consists of 37 sequences, each of which consisting of one minute real life driving data. The provided dataset contains more sequences but the vehicle exits the region where the map is available for two sequences, and therefore these are removed.

This dataset is prerecorded through driving a test vehicle on roads in Gothenburg where the prerecorded map is available. While recording the driving data, ground truth data is collected simultaneously, by an OXTS RT3000 [27]. The data from the OXTS RT3000 is said to be highly accurate [26], but errors have been found especially in height, roll and pitch by Zenseact. Thus the estimated height, roll and pitch from the filter is not be evaluated since there is no reliable ground truth. However, height, roll and pitch is indirectly evaluated since the 2D position estimate of the vehicle from the filter is depending on these states, and if the 2D position is correct then these states have to be correct as well.

The evaluation metrics of the filter is the root mean square error (RMSE) in ego-vehicle lateral and longitudinal position, the normed 2D position and in yaw angle for each sequence. This shows how the filter performs on each individual sequence.

Due to the challenging nature of certain scenarios, it is possible that a single or a few iterations perform badly and affect the state estimate in a way it cannot be recovered. This will then have repercussions on the whole sequence and most likely lead to very large final errors. Because of this, for each association method, every scenario is classified as either *good*, *ok* or *bad*. Scenarios is considered *good* if they have a RSME for the absolute position below one meter. Scenarios with a RMSE for the absolute position between one and four meters is considered *ok*, and the rest is considered *bad*. The final metric to decide which association method that performs the best is the percentage of *good* scenarios out of all scenarios, for each association method. The thresholds for the classification of sequences was chosen through discussions with Zenseact. The reasoning was simply that by having an error less than one meter, the position estimate is probably in the correct lane which is considered *good*. By having an error between one and four meters, the position estimate should still be on the road, which can be interesting from an evaluation perspective.

To evaluate the occlusion technique the same metrics as described above is used. The result is shown with and without the occlusion technique active on all different association techniques.

To show how the filters uncertainty estimate behaves, the 2σ-contours is examined. The 2σ-contours are ellipses with two multiplied with the filters estimation of the standard deviation as its semi-axes. The ellipses shows the uncertainty in the 2D position. Note however that the state of the filter contains more than just the position in x and y, and the filter of course estimates the uncertainty in all states. The reason for only visualizing the uncertainty in x and y is because these states have corresponding ground truth and are easily interpreted. The 2σ-contours will be shown for a scenario together with the ground truth and the filters estimate of the position.

4.2 Multi Layer Road Removal

In this section the different association techniques are evaluated against each other with and without MLRR activated. Starting with the NN comparison followed by the ICNN and lastly the JCBB.

4.2.1 Nearest neighbour

The different RMSE values for the scenarios is shown in Figure 4.1 - 4.4 for the NN association technique. As seen in figure 4.1 the lateral RMSE decreases drastically for sequence 3, 8, 9 when MLRR is active. The longitudinal position decreases for sequence 8 with MLRR activated along some minor fluctuations see Figure 4.2. The yaw error presented in Figure 4.3 is fairly low overall for most of the sequences but seems to be large when the longitudinal and lateral error is large. The absolute position varies for the different sequences and are more easily compared with table 4.1, where the *good*, *ok* and *bad* rate are shown. As shown in table 4.1 it is an improvement in *good*, *ok* and *bad*rate when the MLRR is active.

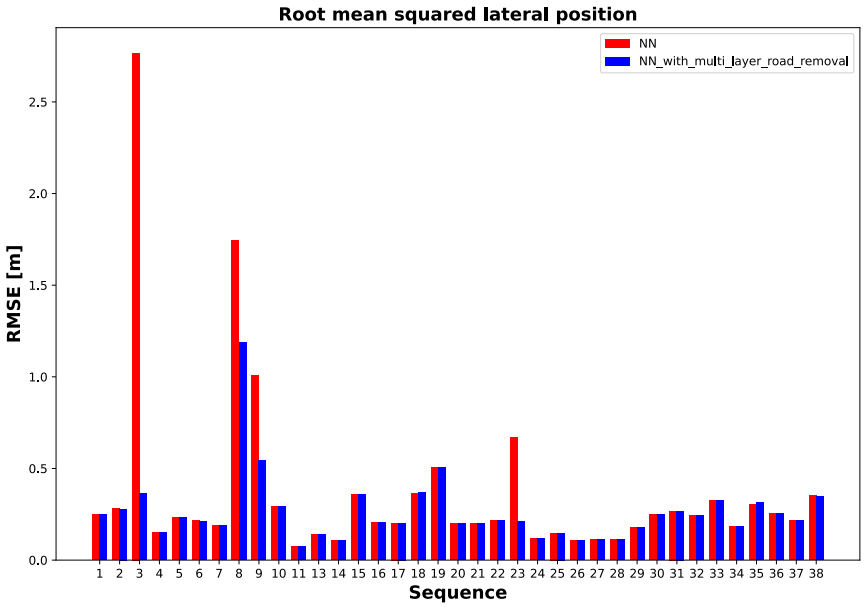

Figure 4.1: RMSE for each scenario in lateral direction with nearest NN with and without MLRR

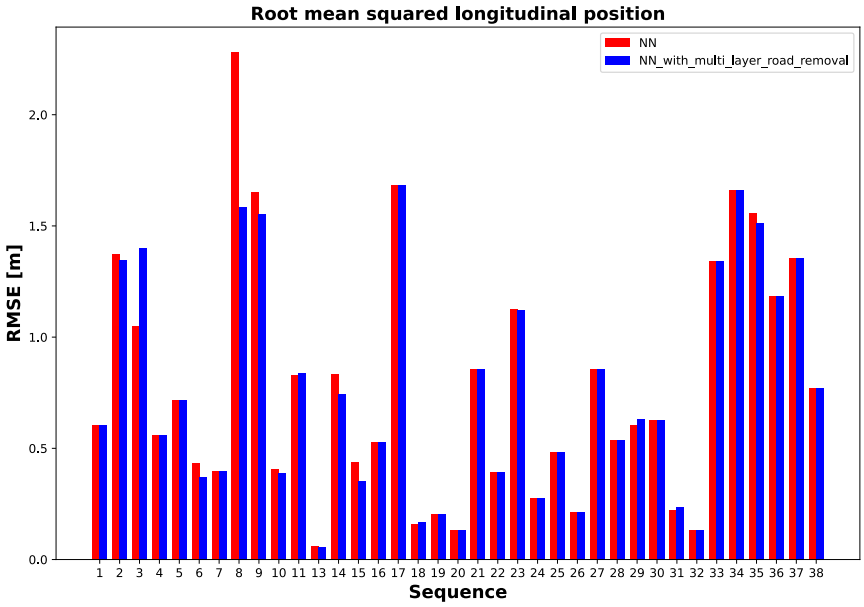

Figure 4.2: RMSE for each scenario in longitudinal direction with NN association with and without MLRR

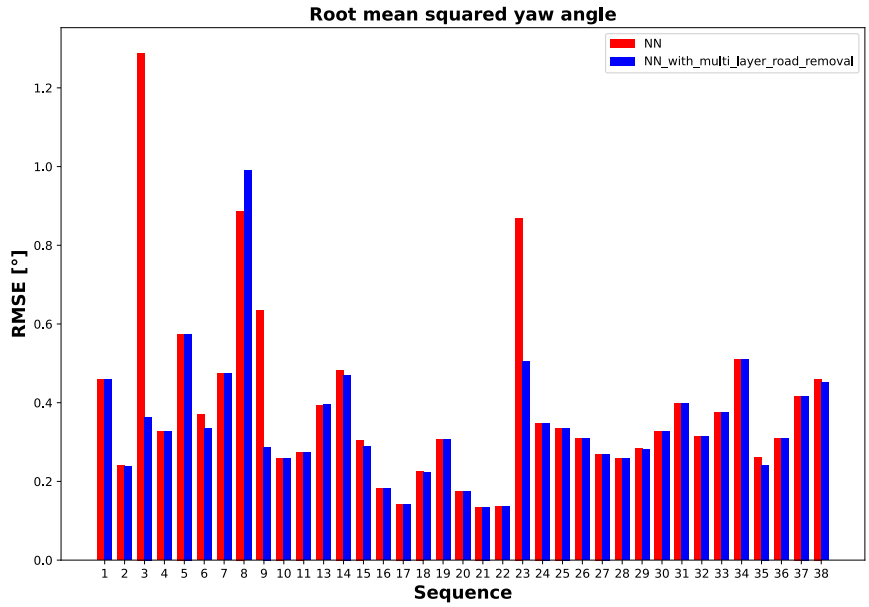

Figure 4.3: RMSE for each scenario in yaw angle with NN association with and without MLRR

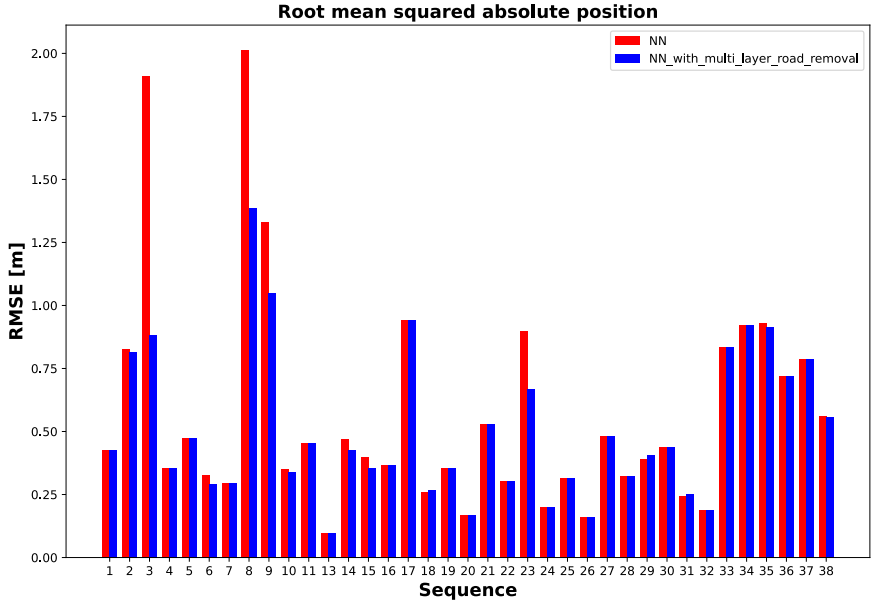

Figure 4.4: RMSE for each scenario in absolute position with NN association with and without MLRR

NN	Good [%]	OK [%]	Bad [%]
Without multi layer road removal	91.9	8.1	0
With multi layer road removal	94.6	5.4	0

Table 4.1: Table of the percentiles for NN with and without MLRR.

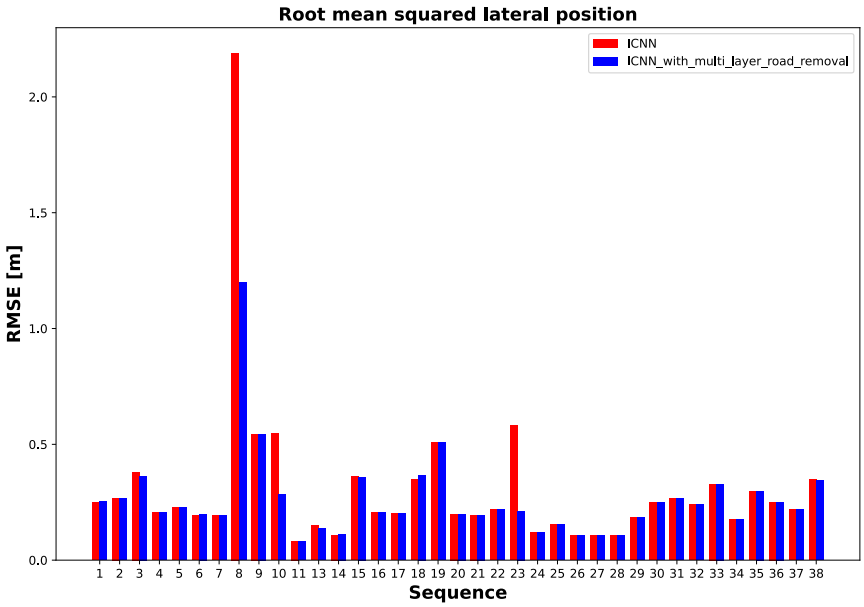

Figure 4.5: RMSE for each scenario in lateral direction with ICNN association with and without MLRR

4.2.2 Individual Compatibility Nearest Neighbour

The different RMSE values with the ICNN association technique for the scenarios is shown in Figure 4.5 - 4.8. The lateral RMSE decreases for sequence 8, 10 and 23 when enabling MLRR. The MLRR is however increasing it slightly for sequence 18, see Figure 4.5. The longitudinal RMSE decreases for sequence 3, 8, 9 and 35 among others, when the MLRR is active. It does however increase it a bit for sequence 5, see Figure 4.6. The yaw RMSE in figure 4.7 decreases for sequences 8, 10 and 23 with MLRR activated. The absolute position RMSE is presented in 4.8 and is best compared by looking at Table 4.2. It can be seen in Table 4.2 that the MLRR improves the performance in general compared to when it is not activated.

ICNN	Good [%]	OK [%]	Bad [%]
Without multi layer road removal	89.2	10.8	0.0
With multi layer road removal	91.9	8.1	0.0

Table 4.2: Table of the percentiles for ICNN with and without MLRR.

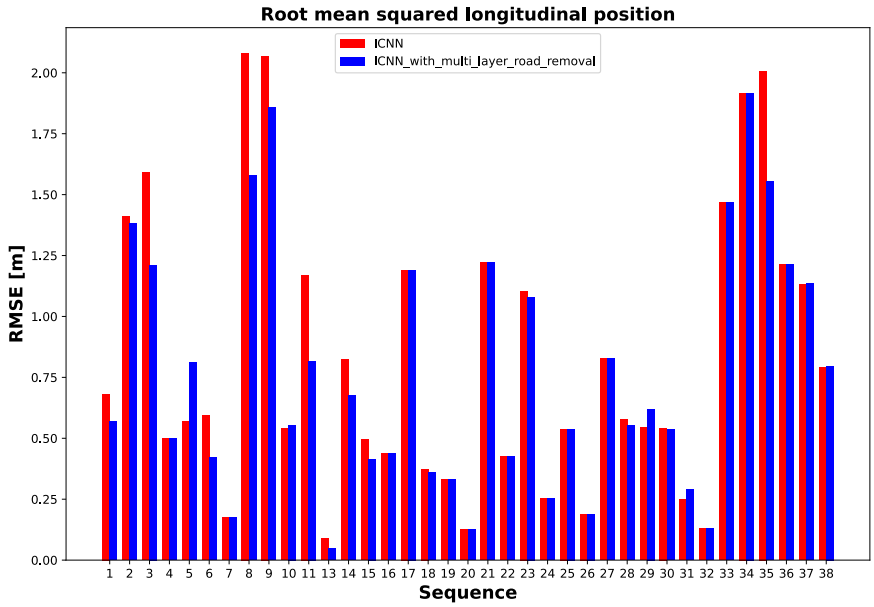

Figure 4.6: RMSE for each scenario in longitudinal direction with ICNN association with and without MLRR

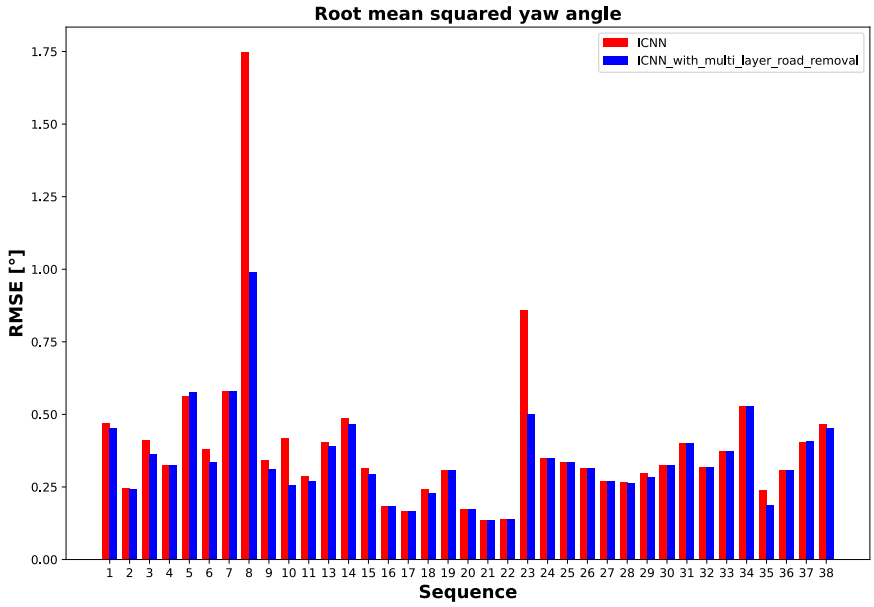

Figure 4.7: RMSE for each scenario in yaw with ICNN association with and without MLRR

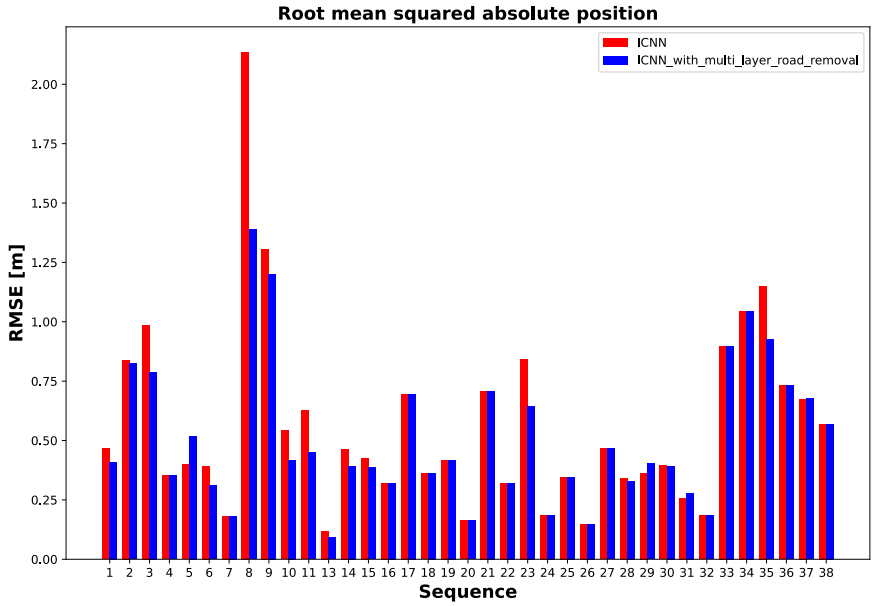

Figure 4.8: RMSE for each scenario in absolute position with ICNN association with and without MLRR

JCBB	Good [%]	OK [%]	Bad [%]
Without multi layer road removal	89.2	10.8	0
With multi layer road removal	91.9	8.1	0

Table 4.3: Table of the percentiles for JCBB with and without MLRR

4.2.3 Joint compatibility branch and bound

The different RMSE values with the JCBB association technique for the scenarios is shown in Figure 4.9 - 4.12. The lateral error decreases slightly in sequence 3. However it is increased in sequences 8, see Figure 4.9. The longitudinal error decreases in sequence 3 but increases for sequence 8 as shown in Figure 4.10, similar sequences as for the lateral error. The yaw RMSE in Figure 4.11 increases drastically in sequence 8, with some smaller changes in other sequences. The absolute position RMSE is shown in Figure 4.12 but easier evaluated with table 4.3. As presented in table 4.3 the *good* and *ok* rate is improved by the MLRR.

4.3 Association method comparison

The best versions of the association techniques are compared to each other in Figure 4.13 - 4.16 and in table 4.4. The best versions for each association technique was NN with MLRR, ICNN with MLRR, and JCBB with MLRR. The lateral RMSE is bigger for JCBB with MLRR in scenario 8 compared to NN with MLRR and ICNN with MLRR, as seen in Figure 4.13. It is shown in Figure 4.14 that the lateral

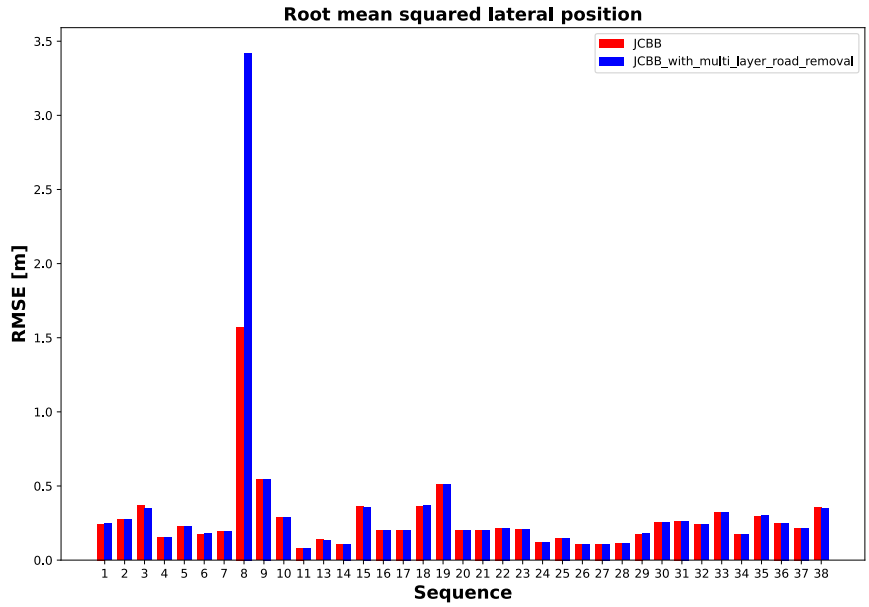

Figure 4.9: RMSE in lateral direction for each scenario with JCBB association, with and without MLRR.

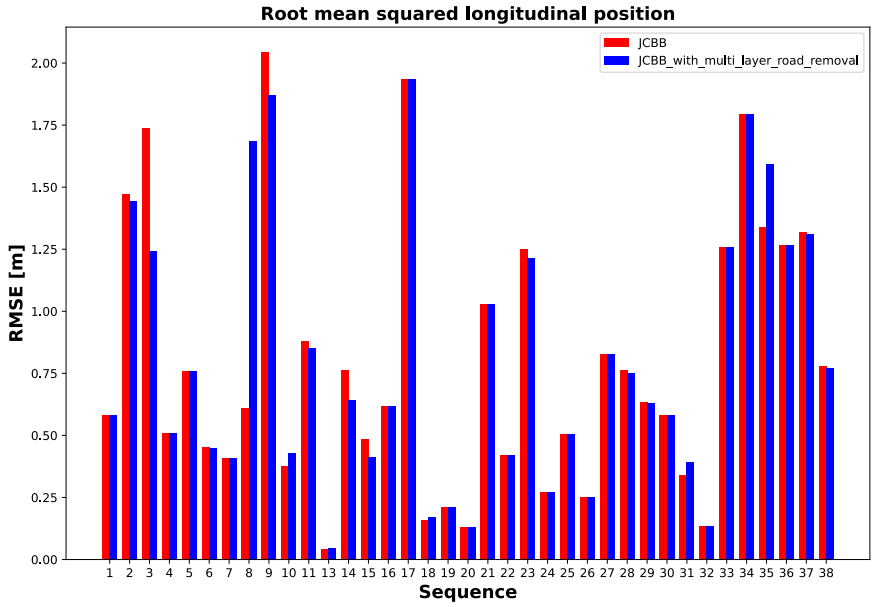

Figure 4.10: RMSE in longitudinal direction for each scenario with JCBB association, with and without MLRR.

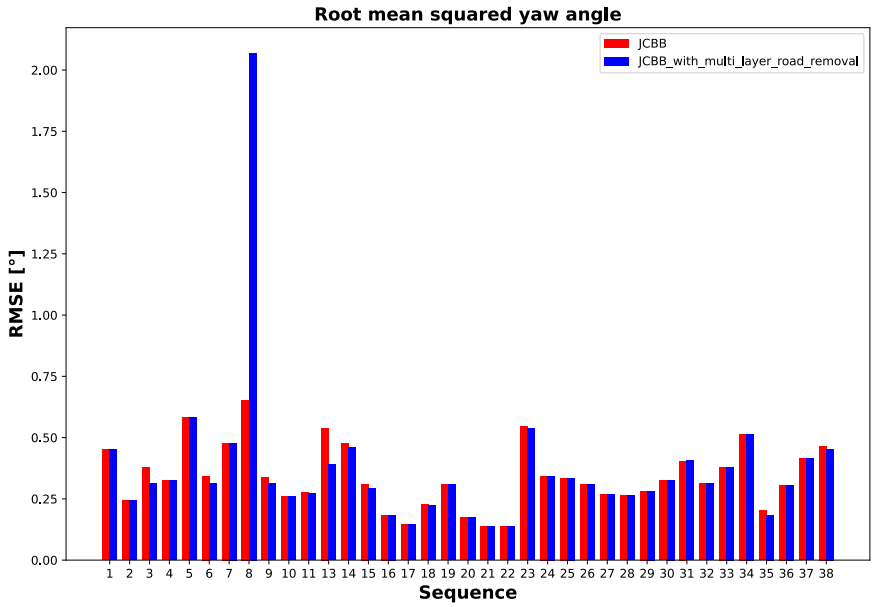

Figure 4.11: RMSE in yaw for each scenario with JCBB association, with and without MLRR.

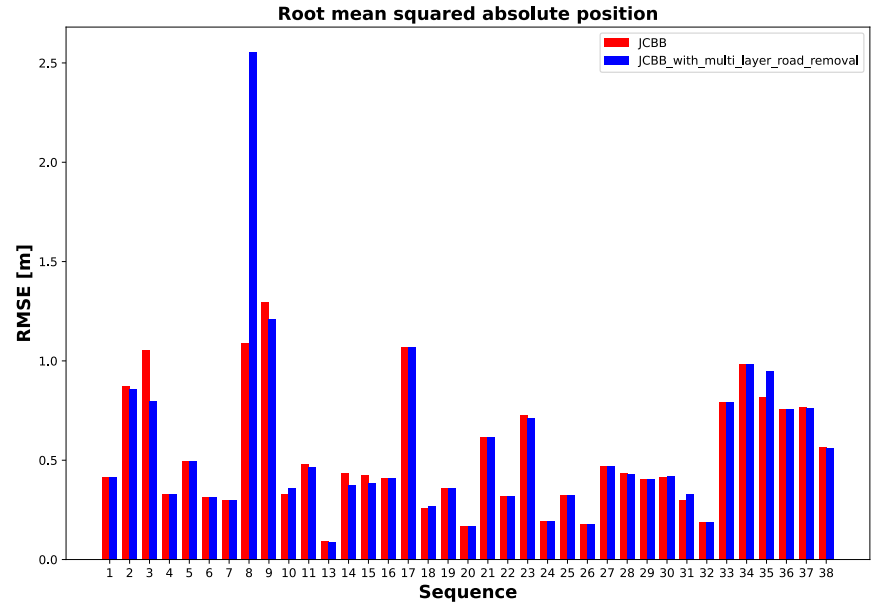

Figure 4.12: RMSE in absolute position for each scenario with JCBB association, with and without MLRR.

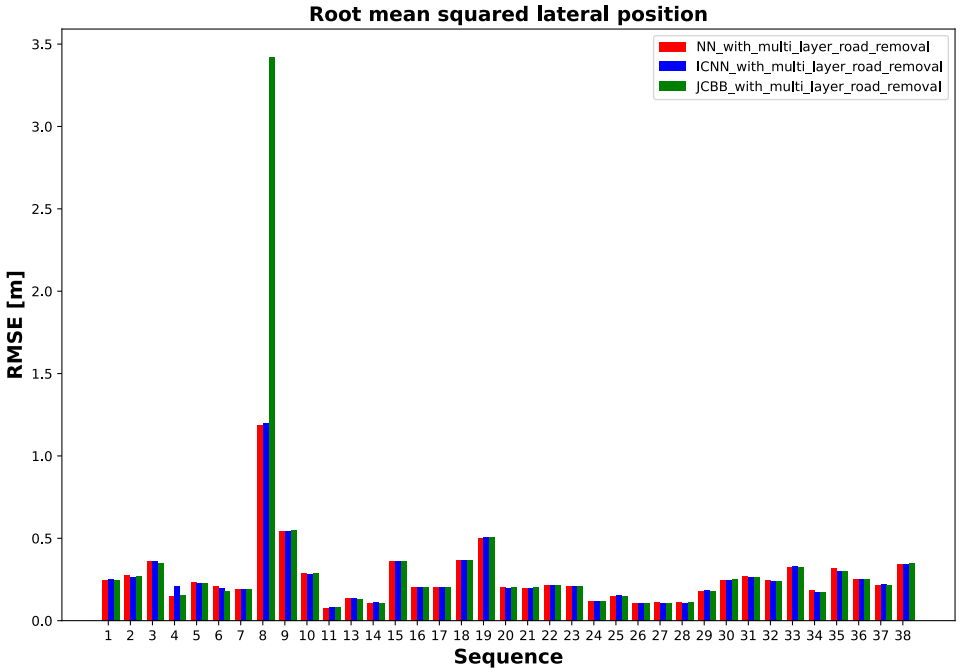

Figure 4.13: RMSE in lateral direction for each scenario with all association methods.

Association technique	Good [%]	OK [%]	Bad [%]
NN with MLRR	94.6	5.4	0
ICNN with MLRR	91.9	8.1	0
JCBB with MLRR	91.9	8.1	0

Table 4.4: Table of mean RMSE for all scenarios for different association techniques

error fluctuates for the different scenarios with the different association techniques. ICNN with MLRR has the best performance on sequence 17, but the worst on other sequences. The yaw RMSE is worst for JCBB with MLRR for sequence 8 compared to the others, see Figure 4.15. This is similar sequences as for the lateral error. The position RMSE is showed in Figure 4.16 and easiest evaluated with table 4.4. It is presented in Table 4.4 that the NN with MLRR has the same best performance regarding the *good*, *ok* and *bad* rate. JCBB with MLRR has the same performance as ICNN with MLRR.

4.4 Uncertainty evaluation

To evaluate the performance of *The localization framework* more than the positional error has to be considered. The uncertainty of the filter is of interest as well, to see if it estimates its own uncertainty in a reasonable manor. Three smaller parts of a sequence will be shown, and the entire sequence is presented in Figure 4.17.

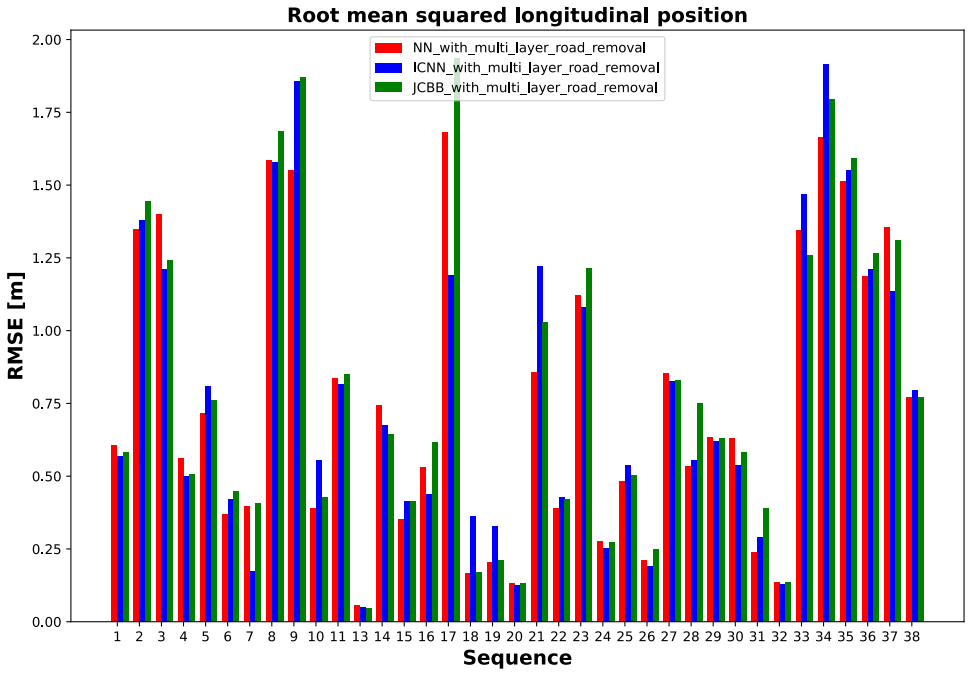

Figure 4.14: RMSE in longitudinal direction for each scenario with all association methods.

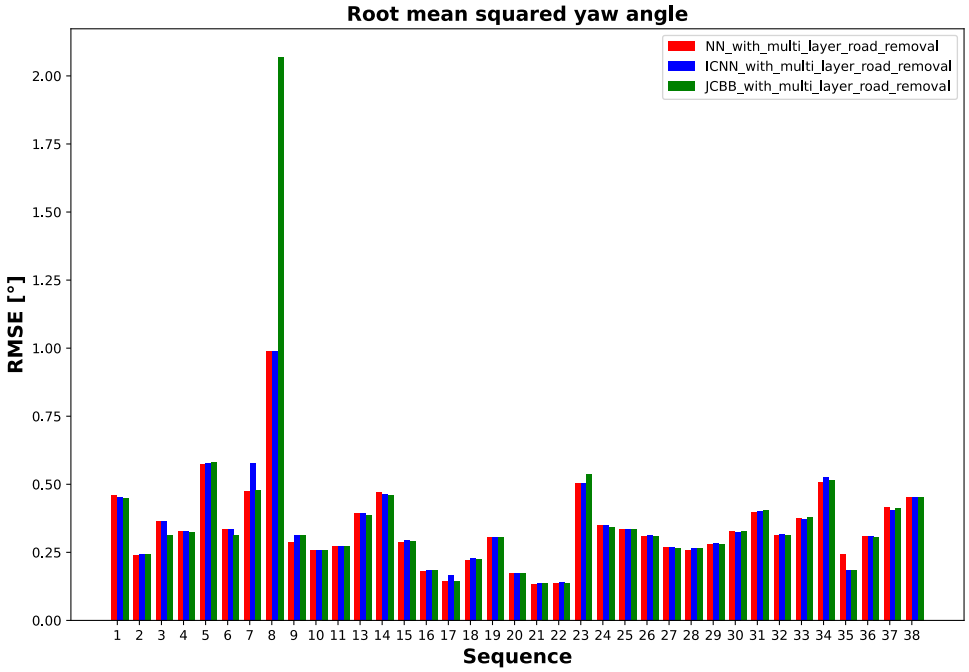

Figure 4.15: RMSE in yaw angle for each scenario with all association methods.

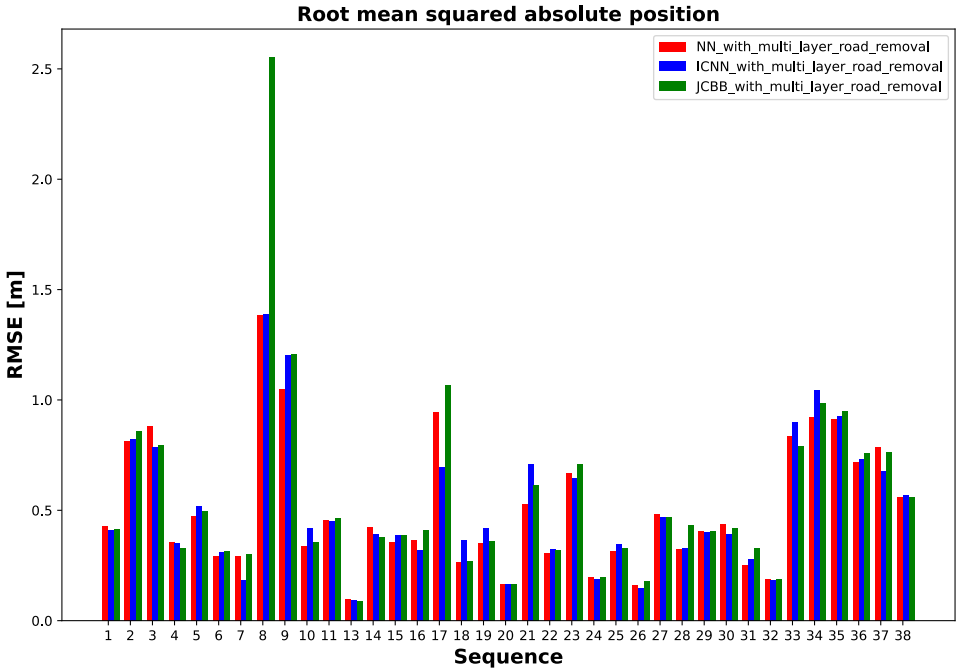

Figure 4.16: RMSE in absolute position for each scenario with all association methods.

The uncertainty covariance matrix from different steps in the filter will be showcased, where P_pred is extracted after the prediction step and P_final is extracted after the Camera update step. To showcase the uncertainty matrices ellipses are made with two standard deviations as its semi-axes.

In Figure 4.18 the very beginning of the sequence is shown, together with the 2σ-contours. The P_pred for the first step is very large since it has not yet gotten any measurements at all, but after being updated with the various sensor readings the uncertainty shrinks quite a bit. Figure 4.19 shows when the vehicle is in the middle of a turn. Here it can be seen that the longitudinal uncertainty is larger than the longitudinal uncertainty in Figure 4.18. Towards the end of the sequence in Figure 4.20, the filter shows less and less uncertainty in the longitudinal direction compared to the case showed in Figure ch 4.19. The lateral uncertainty is smaller towards -655 on the x-axis in Figure 4.20 compared to Figure 4.19, showing that the filter gets more certain about its position over time.

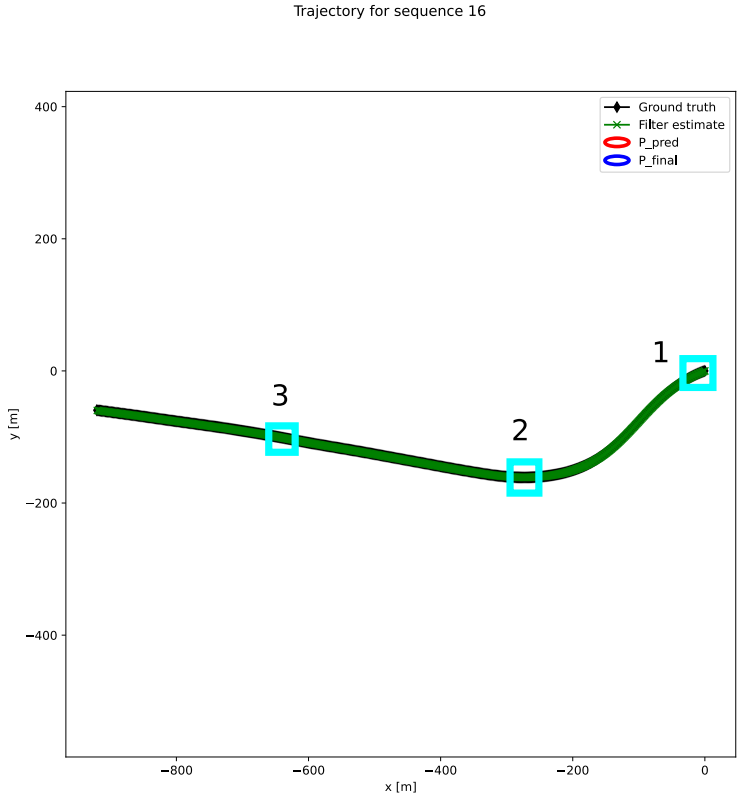

Figure 4.17: Trajectory of sequence 16 using NN association method. The cyan squares indicates where the smaller parts of the trajectory are located, see zoomed in Figure 4.18 - 4.20.

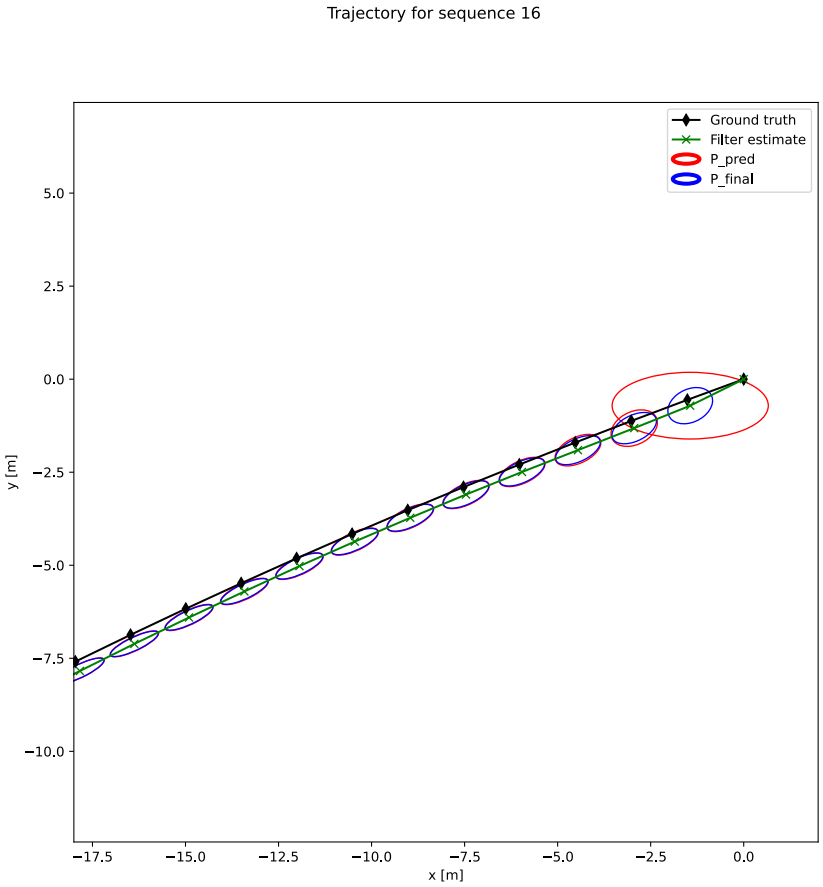

Figure 4.18: Trajectory of sequence 16 zoomed in around square 1 in figure 4.17.

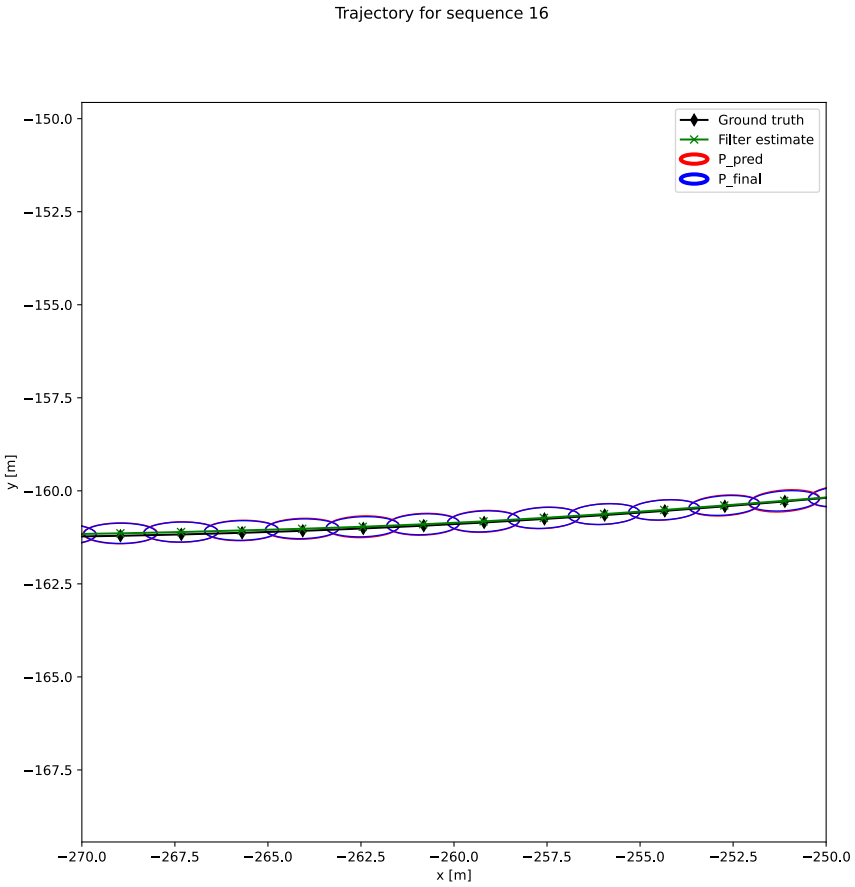

Figure 4.19: Trajectory of sequence 16 zoomed in around square 2 in figure 4.17.

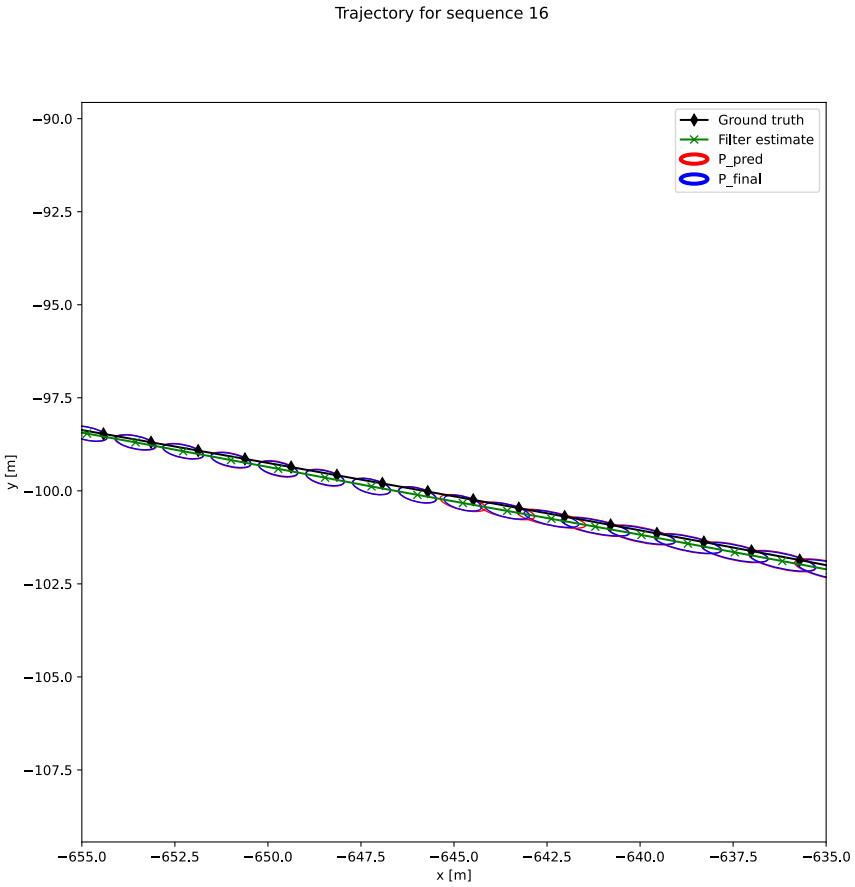

Figure 4.20: Trajectory of sequence 16 zoomed in around square 3 in figure 4.17

5
Discussion

In this chapter the results presented previously is discussed. General comments about the entire system is presented as well as discussions regarding methods, implementation difficulties and evaluation drawbacks.

5.1 General comments about the entire system

The localization framework is shown to be effective in general for the given setting, regardless of the association method. However, the best performing system, as shown in table 4.4, is with NN having MLRR activated. Worth noting though is that it only has one more sequences classified as *good* compared to both ICNN and JCBB. The ICNN has sequence 34 classified as *ok* instead of *good* as for the other association techniques, however the RMSE in absolute position for this sequence is slightly over one meter. This is the same for JCBB but with sequence 17, where the RMSE in absolute position is slightly over one meter, see Figure 4.16.

The localization framework was incrementally improved during the work with this thesis, in order to handle the different situations that occurs in a highway driving scenario. NN was the first association method implemented, and as a result this method highlighted different problems that needed to be solved early on. An example of this is the MLRR. Since the more complex association methods did not have as much trouble with multi layer roads, it is possible that if we would have started with implementing those methods in the system first other problems specific to those methods would have been solved instead, possibly leading to even better results with the more complex methods. It is worth noting that even though the system was developed using the NN as association method, and most of the time spent on this project was spent improving the system with NN as association method, ICNN and JCBB still managed to have similar performance as NN.

Though, since no association method has an RMSE below one meter for 100 % of the sequences, there are still improvements to be made. The ICNN and JCBB data association algorithm does not always associate entirely correct, and most likely it would be possible to achieve better results with some additional tuning. The localization system could possibly achieve better results by additional tuning of the motion model noise and measurement uncertainty covariances as well. Though, a larger dataset with longer sequences would be needed in order to avoid over fitting the system to the available scenarios.

The localization framework has shown to track the position of the ego-vehicle accurately given a sufficiently accurate initial position. However, it lacks a way to handle large corrections in position. If the estimate of the ego-vehicle position drifted away and ended up in an adjacent lane, it is very unlikely that it could find its way back. This is because all lanes look very similar, especially in highway scenarios with many lanes. This will make the filter to most likely stay in the wrong lane if it has drifted to that lane. Since the localization filter matches its measurements with what seems as the best fit given the current filter estimate of the position, it is unable to correct itself in situations where measurements might fit multiple positions. A possible solution to this is presented in section 6.1

A problem observed in the evaluation of *The localization framework* is that it has difficulties estimating drastic changes in z-direction and pitch angle. Also, if the Map measurement fails to deliver a few timesteps in a row the estimation of the height and pitch becomes unreliable. This eventually leads to other errors, such that the MLRR removes useful parts of the map. The poor height and pitch estimation when the changes are drastic is the reason that *The localization framework* performs poorly in sequence 8, see Figure 4.16. The idea behind having roll, pitch and height as states in the localization filter is that by having this the camera sensor could help with the estimation of these states. However, it has been observed through the work with this project that the camera struggles to capture the dynamics of these states. Solutions to the problem of estimating roll, pitch and z is discussed in 6.1.

5.1.1 Map measurement update

The system performs well in most of the given scenarios, and one of the reasons it works is the map measurement update. Since the system does not have explicit sensors for height, roll and pitch it would probably not be as accurate without the measurements extracted from the map. However, this method of extracting measurements from the map relies on the map being very precise, having an accurate current estimate of the position and choosing the correct closest map RSMs. This is a flaw in the system since it lacks robustness in the estimation of the height, roll and pitch. It would however be fairly easy to extend the suite of sensors with a full gyroscope capable of estimating roll rate and pitch rate, and also with a full accelerometer capable of estimating the acceleration in the z-direction. These sensor upgrades would probably improve *The localization framework* to have a more robust estimation of height, roll and pitch.

5.1.2 Camera noise modeling

In this project all measurement noise has been modeled as uncorrelated Gaussian noise, with a constant standard deviation term describing the noise from each sen-

sor. This model is shown to work since the localization filter is able to accurately estimate the position of the ego vehicle. However, it is reasonable to think that a more complex modeling of the noise could improve the results, especially for the camera sensor. Measurements from the camera that come from the bottom center of the image are very close to the camera, and will therefore appear large in the image. Measurements that come from the center of the image are very far away, and appear very small in the image. It is reasonable to think that the noise of each measurement should reflect its resolution, such that a measurement that is very large in the image frame should have a slightly larger noise than something small.

Having a variable noise where measurements farther away has a lower noise could also help in the association step, since the noise to some extent decides the threshold for how far away associated detected RSMs and map RSMs can be. A lower noise for detections close to the center of the image would lower this threshold, which is reasonable since the corresponding map elements appear closer near the center of the image.

The noise could be scaled by calculating how far away the pixels in the image are from the principal point, making those points closer to the principal point have lower noise compared to those far away.

5.2 Multi layer road removal

The MLRR improves the performance of *The localization framework* for all association methods, as shown in Tables 4.1, 4.2 and 4.3. It is most clearly shown in scenario 3 where it has been seen throughout the testing that some detected RSMs associate to a bridge beneath the road it is currently driving on, see Figures 4.4,4.8 and 4.12. This causes great error in that sequence and the positional error is smaller when the road beneath is removed. This performance improvement is however not the case for all sequences for JCBB, see Figure 4.12, where sequence 8 increases when the MLRR is activated. The reason for this is that the MLRR is highly depending on the current estimate. If the current estimate have drifted far enough in height MLRR will remove the current road the vehicle is driving on and the estimate can continue to drift. It can be concluded that removing objects from the map must be done very carefully, since useful information might be removed as well.

MLRR is shown to increase the performance more for NN compared to the impact on the ICNN and JCBB. This is due to that the NN simply chooses the closest point in image frame as the correct association and taking no other information into account. Both ICNN and JCBB uses the normalized innovation squared value instead, where the dynamics of the map and camera are considered as well. This makes these more complex association techniques more robust than the simpler technique.

Furthermore, the MLRR is highly dependent on the current estimated height. If the current estimated height have a large error the correct RSMs might be removed,

and the camera and map will then be useless. In conclusion, the MLRR helps the localization system as long as the positional estimate is good enough, otherwise it can make the localization worse.

5.3 Normalized innovation squared and the χ^2 gating test for association

Using the normalized innovation squared as the metric for judging associations, used in ICNN and JCBB, seems as shown in the results to be better than using plain projection distance as used in NN. This is most likely due to the fact that the normalized innovation squared takes the current uncertainty in the filter into account, such that it can allow associations where the detected RSM and map RSM are farther away from each other if the uncertainty in the filter is high. Furthermore, since the jacobian of the innovation function is involved in the computation of the normalized innovation squared value, it also judges the association on how large the update of the state would be due to that association. Because the association is done in the image frame, the projection distance in the image frame can translate to different distances in the 3D cartesian frame, since it matters where in the image the distance is measured.

The gating test presented in Equation (2.23), which is used both in ICNN and JCBB effectively rejects bad associations. This can be concluded from the results of JCBB, since the branch and bound algorithm only tries to find the largest set of compatible points regardless of the normalized innovation squared value. Because of the fact that JCBB is able to localize most of the time, the gating test must be rejecting bad associations because otherwise wrongful associations would be made consistently.

5.4 Association comparison

Different association techniques have been tested and evaluated in this thesis. There are some similarities between them that are worth mentioning. For example are NN and ICNN basically the same algorithm, but with different metrics of determining a good fit. Both methods chooses the association for a detected RSM by taking the map RSM with the lowest score, and therefore both methods allows for multiple detected RSMs to be associated to the same map RSM. Even though this has the capability to handle the corner case where multiple detected RSMs in fact come from the same RSM, it fails to utilize the information that detected RSMs in most cases should not be associated to the same map RSM.

The JCBB on the other hand, utilizes this since mutual exclusion is a part of the algorithm. It tries to find the best set of associations jointly, which in this implementation unfortunately comes at a high computational cost. A great advantage of the simpler data association techniques is the low time complexity, but the Branch

and Bound algorithm has potentially much higher time complexity since it has to traverse an hypothesis tree. This tree has shown to grow large for cases where there are a lot of both detected RSMs and map RSMs. This is not further investigated in this thesis since the goal is not to have the solution running in real time, however if these association algorithms where to be implemented in real time this aspect would be of greater importance.

The three different methods of association used in this project differ in complexity, where NN is the simplest, ICNN is some middleground and JCBB is the most complex. It is shown in the results that ICNN and JCBB performs equally well, while NN performed the best. It is reasonable to say that the expected result would be that the most complex method would perform the best, but the results presented does not support this. This could be because of the previously mentioned fact that the entire system was first developed using NN as association method.

It is however worth noting that all association methods perform really well in lateral direction, shown in Figure 4.16. This shows that the only reason that NN gets a better *good*-rate is because it manages to get a better estimation of the longitudinal position. An explanation for this could be because NN associates more measurements in general, and some of them could impose a longitudinal state update which the other association methods rejects as unlikely measurements. The way that the observation from the camera sensor is calculated does not grant any reliable information in longitudinal direction, except during perfect circumstances when the road is turning. Since *the localization system* does not have any other feedback for the longitudinal direction, the longitudinal position error is bound to drift.

Furthermore, when looking at the lateral error for the association methods with and without MLRR activated, it can be seen that JCBB basically is not affected by the MLRR at all. Even without MLRR it manages to perform as well as the other methods with MLRR activated in lateral error. This can be seen in Figures 4.1, 4.5 and 4.9. A conclusion of this is that the JCBB association method is more robust against multi layered roads than both ICNN and NN.

5.5 Ground truth correctness and filter uncertainty

The ground truth used in this thesis are measurements from an OXTS RT3000, and as mentioned before this a highly accurate GNSS. Since this is what the proposed solutions are compared to, *The localization framework* is evaluated on how similar it is to the OXTS RT3000 measurements. It is reasonable to believe that this usually is not an issue in vehicle tracking, since the OXTS RT3000 often is the most accurate data available. However, in this project there is a highly accurate HD map available that is obtained through a different way than the OXTS RT3000 measurements are. The HD map and the OXTS RT3000 are not aligned with each other which might cause greater errors in the localization filter estimate evaluation than it actually is. *The localization framework* tries to align itself with the map using the camera detections, and given that the HD map is highly accurate this should produce a

highly accurate pose. The error shown in this thesis might therefore be larger than it actually is. However, it should be noted that since no evidence of the accuracy of the map is presented, this is pure speculation.

Regarding the uncertainty modeling in the localization filter, it sometimes fails to capture the ground truth inside the 2σ-contours. This might be because of the previously mentioned problem, namely that the filter could be more accurate that the ground truth. However, it is reasonable to believe that the filter does not capture the uncertainty perfectly and perhaps should have been tuned in a different way in order to yield a more accurate estimate of the uncertainty.

6
Conclusion

This thesis has provided a system, *The localization framework*, for estimating a vehicle pose using sensor fusion with a monocular camera together with an HD map, wheel speed sensors, a partial IMU and the HD map on its own. The provided method has an RMSE below one meter for 94.6 % of the tested real world driving scenarios. *The localization framework* is proven to be most effective with the easier Nearest Neighbour data association technique for the camera and HD map measurements.

Three different methods for associating a camera detection with the corresponding map element has been implemented and tested. The simple NN, the slightly more complex ICNN and the even more complex JCBB. Through testing these different algorithms on the same set of data, with the same filter tuning, it is shown that the NN algorithm with MLRR is the most accurate association method on the provided real world driving scenarios. However, it can also be seen that all methods perform equally well in lateral error when using MLRR. It is also shown that JCBB is more robust against multi layered roads when not using MLRR compared to NN or ICNN without MLRR.

It is shown that the NIS is a good scoring tool, showed by the good results of the ICNN method. The χ^2-gating test is shown to effectively remove faulty associations, proven by the good performance of the JCBB method.

It is also shown that by preprocessing the HD map through the provided Multi Layer Road Removal method effectively removes possibly harmful parts of the map as long as the current estimate of the position is good enough. The possible harmful parts refer to other layers of the road which the vehicle is currently driving on, such as bridges above or roads beneath. However, the Multi Layer Road Removal sometimes also removes potentially useful parts of the map which worsens positional estimate.

The localization system is not able to handle all of the sequences in the testing with satisfactory accuracy, and therefore further work with improving both the tuning of the localization filter itself, as well as the data association technique would be needed.

6.1 Future work

The localization framework has some flaws as discussed previously in this chapter and some additional work could potentially increase the localization accuracy.

To make the estimate of the height, roll and pitch more robust additional sensors would be needed. For example, the acceleration in z-axis could be measured, similar as the acceleration in x-axis and y-axis is measured in the current system, with a complete accelerometer. To better estimate the roll and pitch a complete gyroscope is needed, and not only yaw rate measurements which is what the current setup has. Having access to these sensors together with the current solution would make the estimation of the height, roll and pitch more robust and less reliant upon the map and the current estimate of the position. Another possible solution would be to omit roll, pitch and height from the states in the filter, and instead retrieving these from the map when needed. This could possibly lead to a better positional estimate from the camera sensor since it would not be able to change roll, pitch and height in order to fit the measurements with the map. Instead it would only update the positional states together with the yaw, which most likely is better since the current observation function mainly allows for correction in position and yaw.

Another sensor that is needed in order to get a robust localization system is a GPS. No feedback in longitudinal position is currently used in the system which causes it to drift. The presented system is shown promising results for localizing laterally but fails in the longitudinal direction.

To be able to correct the estimated states when it has drifted from the correct position a multiple filter solution could be used. It could for example be having one localization filter per driving lane on the current road, and continuously entertaining a hypothesis of being in each lane. In each time step the hypothesis matching the current measurements the best would be chosen as the current positional estimate. A solution like this would have the ability to find its way back to the correct position after a potential drift to another lane.

The implemented noise model for the camera is in *The localization framework* rather simple and could be more advanced. As discussed should the noise for an RSM far away, that is near the center of the image, have a lower noise compared to RSMs in the bottom of the image. This could be a subject for future work and might improve performance of *The localization framework*.

7

Bibliography

[1] K. Jo, K. Chu, and M. Sunwoo, "Erratum: Interacting multiple model filter-based sensor fusion of GPS with in-vehicle sensors for real-time vehicle positioning (IEEE Transactions on Intelligent Transportation Systems (2012) 13: 1 (329-343))," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 2, p. 973, 2012.

[2] J. Min Kang, T. S. Yoon, E. Kim, and J. B. Park, "Lane-level map-matching method for vehicle localization using GPS and camera on a high-definition map," *Sensors (Switzerland)*, vol. 20, no. 8, pp. 1–22, 2020.

[3] W. Lu, E. Seignez, F. S. A. Rodriguez, and R. Reynaud, "Lane marking based vehicle localization using particle filter and multi-kernel estimation," in *2014 13th International Conference on Control Automation Robotics Vision (ICARCV)*, pp. 601–606, 2014.

[4] F. Poggenhans, N. O. Salscheider, and C. Stiller, "Precise Localization in High-Definition Road Maps for Urban Regions," *IEEE International Conference on Intelligent Robots and Systems*, pp. 2167–2174, 2018.

[5] M. Schreiber, C. Knöppel, and U. Franke, "LaneLoc: Lane marking based localization using highly accurate maps," *IEEE Intelligent Vehicles Symposium, Proceedings*, no. Iv, pp. 449–454, 2013.

[6] Y. Lu, J. Huang, Y. T. Chen, and B. Heisele, "Monocular localization in urban environments using road markings," *IEEE Intelligent Vehicles Symposium, Proceedings*, no. Iv, pp. 468–474, 2017.

[7] K. Jo, Y. Jo, J. K. Suhr, H. G. Jung, and M. Sunwoo, "Precise Localization of an Autonomous Car Based on Probabilistic Noise Models of Road Surface Marker Features Using Multiple Cameras," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 6, pp. 3377–3392, 2015.

[8] Z. Xiao, D. Yang, T. Wen, K. Jiang, and R. Yan, "Monocular localization with vector HD map (MLVHM): A low-cost method for commercial IVs," *Sensors (Switzerland)*, vol. 20, no. 7, pp. 1–24, 2020.

[9] J. D. Tard, "Joint Compatibility Test," *IEEE Transactions on Robotics*, vol. 17, no. 6, pp. 890–897, 2001.

[10] M. Harr, K. D. Müller, A. M. Hellmund, and N. Wagner, "Robust localization on highways using low-cost GNSS, front/rear mono camera and digital maps," *AmE 2018: Automotive meets Electronics - 9. GMM-Fachtagung*, pp. 20–26, 2018.

[11] I. Skog, P. Händel, and S. Member, "In-Car Positioning and Navigation Technologies — A Survey," vol. 10, no. 1, pp. 4–21, 2009.

[12] X. Xin, J. Chen, and J. Zou, "Vehicle State Estimation Using Cubature Kalman Filter," in *2014 IEEE 17th International Conference on Computational Science and Engineering*, no. 1, pp. 44–48, IEEE, 2014.

[13] B. Zhang, S. Chen, and X. Zhu, "A Robust Cubature Kalman Filter for GPS Vector Tracking Loop," *International Journal of Engineering and Applied Sciences (IJEAS)*, vol. 5, no. 10, pp. 43–47, 2018.

[14] R. Schubert, E. Richter, and G. Wanielik, "Comparison and evaluation of advanced motion models for vehicle tracking," *Proceedings of the 11th International Conference on Information Fusion, FUSION 2008*, no. January, 2008.

[15] Z. H. E. Chen, "Bayesian Filtering : From Kalman Filters to Particle Filters , and Beyond VI Sequential Monte Carlo Estimation : Particle Filters," vol. 182, no. January, 2003.

[16] E. A. Wan and R. Van Der Merwe, "The unscented Kalman filter for nonlinear estimation," *IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium, AS-SPCC 2000*, pp. 153–158, 2000.

[17] I. Arasaratnam and S. Haykin, "Cubature kalman filters," *IEEE Transactions on Automatic Control*, vol. 54, no. 6, pp. 1254–1269, 2009.

[18] S. Thrun, "Probabilistic robotics," *Communications of the ACM*, vol. 45, no. 3, p. 43, 2002.

[19] S. Thrun, "Probabilistic robotics," *Communications of the ACM*, vol. 45, no. 3, p. 96, 2002.

[20] D. Bernstein and A. Kornhauser, "An Introduction to Map Matching for Personal Navigation Assistants," *Technology*, vol. 24064, no. August, pp. 587–602, 1996.

[21] C. E. White, D. Bernstein, and A. L. Kornhauser, "Some map matching algorithms for personal navigation assistants," *Transportation Research Part C: Emerging Technologies*, vol. 8, no. 1-6, pp. 91–108, 2000.

[22] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan, *Estimation with applications to tracking and navigation: theory algorithms and software*. John Wiley & Sons, 2004.

[23] P. E. H. Richard O. Duda, *Pattern classification and scene analysis*, vol. 3. John Wiley and Sons, 1973.

[24] J. Neira and J. D. Tardós, "Data association in stochastic mapping using the joint compatibility test," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 6, pp. 890–897, 2001.

[25] Y. Li, S. Li, Q. Song, H. Liu, and M. Q. Meng, "Based Approximate Joint Compatibility Test," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 1, pp. 331–339, 2014.

[26] "Road vehicles — Vehicle dynamics and road-holding ability — Vocabulary," standard, International Organization for Standardization, Geneva, CH, Dec. 2011.

[27] OXTS, "Rt3000 v3," 2021.

[28] M. Roth, G. Hendeby, and F. Gustafsson, "EKF/UKF maneuvering target tracking using coordinated turn models with polar/Cartesian velocity," in *17th International Conference on Information Fusion (FUSION)*, pp. 1–8, 2014.

[29] L. Weng, M. Yang, L. Guo, B. Wang, and C. Wang, "Pole-based real-time localization for autonomous driving in congested urban scenarios," *2018 IEEE International Conference on Real-Time Computing and Robotics, RCAR 2018*, pp. 96–101, 2019.

[30] V. Bistrovs and A. Kluga, "The analysis of the UKF-based navigation algorithm during GPS outage," *Elektronika ir Elektrotechnika*, vol. 19, no. 10, pp. 13–16, 2013.

[31] M. Schreiber, A. M. Hellmund, and C. Stiller, "Multi-drive feature association for automated map generation using low-cost sensor data," *IEEE Intelligent Vehicles Symposium, Proceedings*, vol. 2015-August, no. Iv, pp. 1140–1147, 2015.

A

Appendix 1

A.1 Motion model evaluation

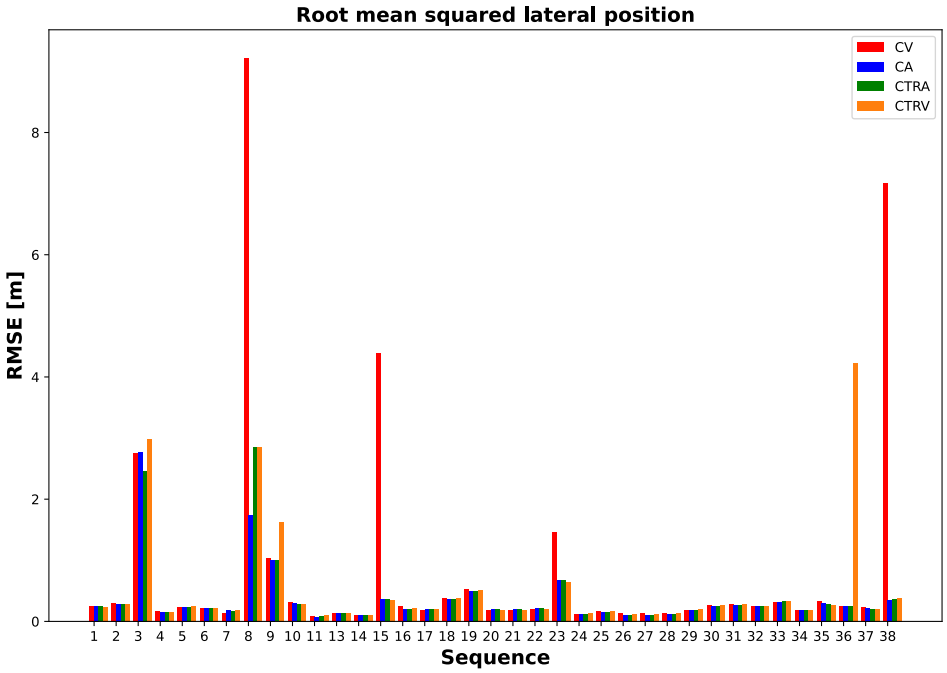

Figure A.1: RMSE for each scenario in lateral direction for different motion models implemented in *The localization framework* with Nearest Neighbour association

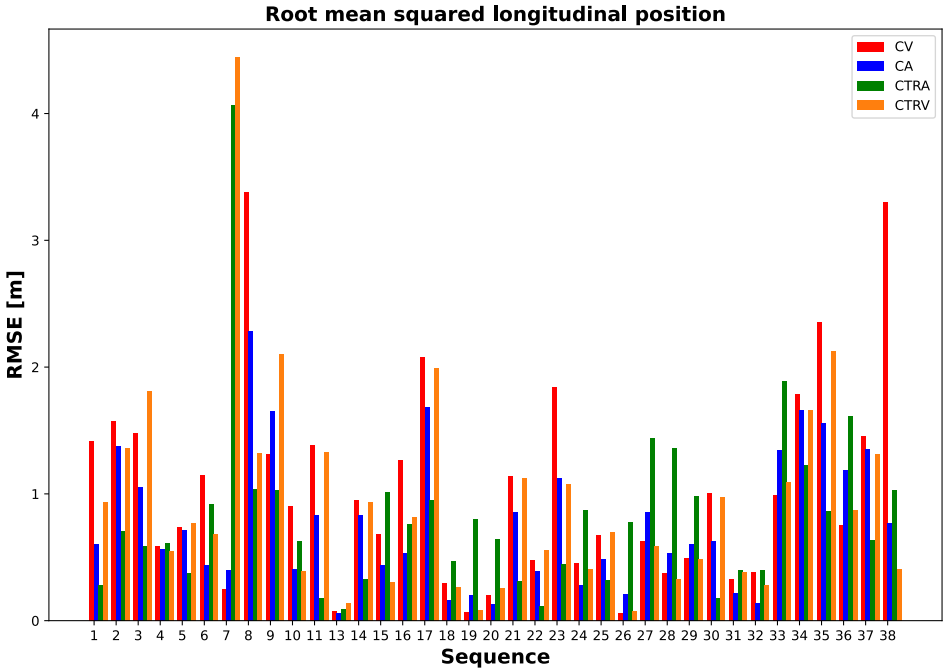

Figure A.2: RMSE for each scenario in longitudinal direction for different motion models implemented in *The localization framework* with Nearest Neighbour association

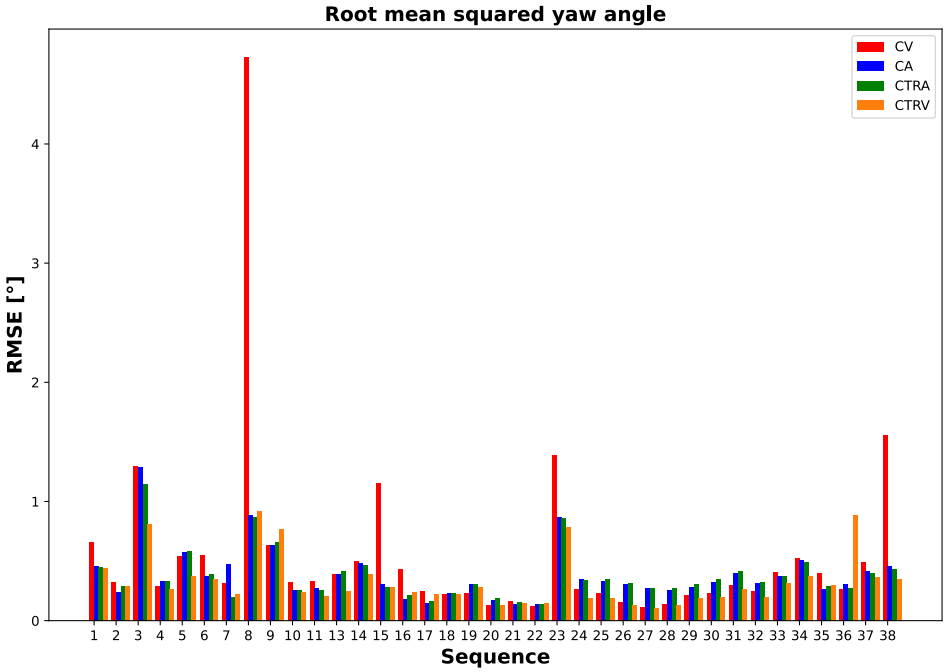

Figure A.3: RMSE for each scenario in yaw for different motion models implemented in *The localization framework* with Nearest Neighbour association

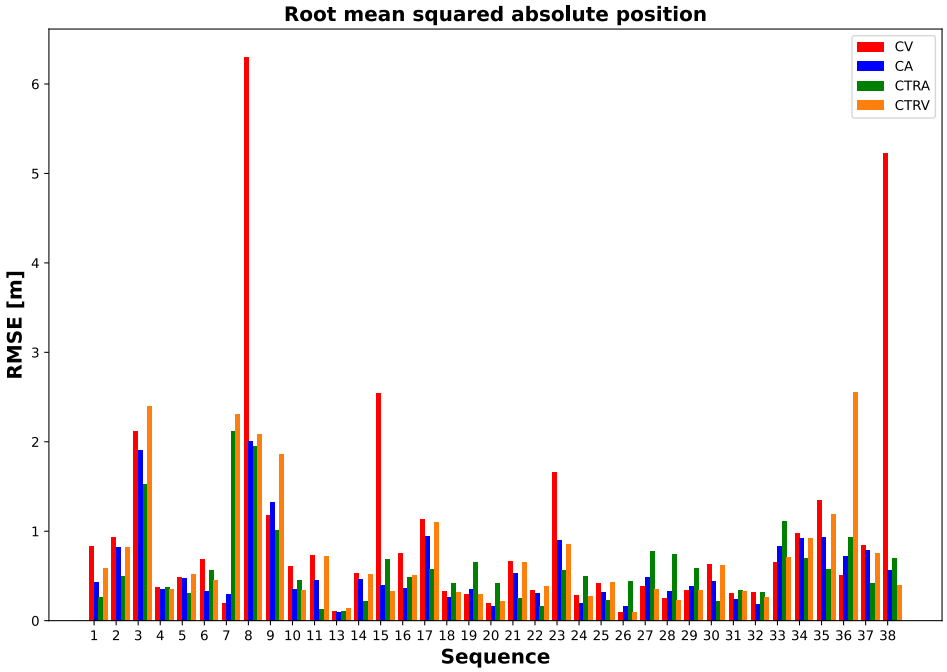

Figure A.4: RMSE for each scenario in position for different motion models implemented in *The localization framework* with Nearest Neighbour association

A.2 Mean RMSE for motion model

Motion model	Good rate [%]	OK rate [%]	Bad rate [%]
CV	78.4	16.2	5.4
CA	91.9	8.1	0
CTRV	81.1	18.9	12.8
CTRA	86.5	13.5	0

Table A.1: Table of mean RMSE for all scenarios for different motion models