CHALMERS
UNIVERSITY OF TECHNOLOGY

# Semi-automatic Annotation for 3D Object Detection using Bayesian Smoothing

Master's thesis in Computer Science – Algorithms, Languages and Logic & Systems, Control and Mechatronics

JOHAN HYREFELDT & BJÖRN VIBERG

Department of Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2019

Master's thesis 2019

# Semi-automatic Annotation for 3D Object Detection using Bayesian Smoothing

Johan Hyrefeldt & Björn Viberg


CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
*Division of Signal Processing and Biomedical Engineering*
Chalmers University of Technology
Gothenburg, Sweden 2019

Semi-automatic Annotation for 3D Object Detection using Bayesian Smoothing

JOHAN HYREFELDT BJÖRN VIBERG



Supervisors:
Lennart Svensson, Department of Electrical Engineering
Daniel Langkilde, Annotell
Examiner:
Lennart Svensson, Department of Electrical Engineering

Master's Thesis 2019
Department of Electrical Engineering
Division of Signal Processing and Biomedical Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Gothenburg, Sweden 2019

Semi-automatic Annotation for 3D Object Detection using Bayesian Smoothing
JOHAN HYREFELDT
BJÖRN VIBERG
Department of Electrical Engineering
Chalmers University of Technology

# Abstract

In the field of autonomous driving, it is important to localize surrounding objects such as cars, pedestrians and cyclists using sensor data, such as camera or LiDAR data. To this end, one common technique is to use neural networks to perform object detection, where the purpose is to classify all objects and produce bounding boxes around them. To train neural networks for object detection, large datasets annotated with bounding boxes are needed. The extensive amount of annotations needed in these datasets normally involve many hours of manual labor, making them very expensive.

In this thesis, a method to increase the level of automation in the annotation process for 3D bounding boxes is presented. The method uses an iterated posterior linearization smoother – a smoothing algorithm based on Bayesian inference – to estimate a trajectory for a particular object sequence. The trajectory can then be used to propose bounding boxes at intermediate time instances. In this project we focus on trajectory estimation for moving cars.

The method was tested using four motion models that describe the kinematics of the cars in different ways: constant velocity, constant acceleration, coordinated turn and the bicycle model. Results for these models were compared and the non-linear models – coordinated turn and the bicycle model – outperform the simpler linear models – constant velocity and constant acceleration. Furthermore, coordinated turn yields slightly better results than the bicycle model, in the sense that it allows for a higher level of automation while at the same time being simpler in its design.

Keywords: Semi-automatic annotation, Trajectory estimation, Bayesian inference, Smoothing, IPLS, Square-root smoothing

# Acknowledgements

Firstly, we would like to express our gratitude examiner and supervisor Prof. Lennart Svensson, from the department of Electrical Engineering, for his support, great interest and many ideas that helped us a lot in this thesis project. We would also like to thank our company supervisor Daniel Langkilde, co-founder and CPO at Annotell, for his commitment and weekly meetings that gave us useful insights. We would also like to thank Dr. Ángel García-Fernández for his guidance in square-root implementations of smoothing algorithms. Finally, we want to say that we have had a great time with the people at Annotell and we have really appreciated their good company.

Johan Hyrefeldt & Björn Viberg, Gothenburg, June 2019

# Contents

# List of Figures



[1] light detection and ranging

# List of Tables

# Abbreviations and Acronyms

**AHS** average heading similarity

**BM** bicycle model

**CA** constant acceleration

**CT** coordinated turn

**CV** constant velocity

**IoU** intersection over union

**IPLS** iterated posterior linearization smoother

**KF** Kalman filter

**KLD** Kullback-Leibler divergence

**LiDAR** light detection and ranging

**MAP** maximum a posteriori

**MSE** mean square error

**pdf** probability density function

**RADAR** radio detection and ranging

**RMSE** root-mean-square error

**RTS** Rauch-Tung-Striebel smoother

**SLR** statistical linear regression

**SSM** state space model

# 1

# Introduction

The emerging field of self-driving vehicles has the potential to create a safer environment for everyone on and near the road network. For a car to drive autonomously it needs to be able to localize and classify surrounding objects using sensor data, such as camera images or point clouds from a LiDAR. This problem is known as *object detection*, where the goal is to produce labeled bounding boxes around the objects and it can be done both in 2D and 3D. The most common technique to perform object detection is to train neural networks for this task which requires large datasets with annotated bounding boxes.

The annotated datasets are normally created by humans using a software tool designed for the annotation task. Because of the huge number of annotations needed to train the neural networks, a lot of manual labor is needed. A completely manual annotation process can incur a high cost which makes it necessary to find methods that reduce the need for human labor or alternatively, support the human annotator to work efficiently.

This thesis has been conducted in cooperation with Annotell, which is a company that aim to build an annotation software that allows for a consistent and efficient annotation process. Those features are enabled by different components such as an intelligently designed graphical user interface and an increased level of automation in the annotation process. This project is intended to address the latter component – the level of automation – which is best explained by the amount of annotation work that the software can take over from the human annotator that operates the software. This thesis specifically focuses on raising the level of automation when annotating 3D boxes around moving cars.

## 1.1 Quality and Human Intervention

It is clearly desirable to create an annotation software tool that reduces the need for human labor as much as possible since human working hours are expensive compared to that of computers. Ideally the software would perform all the annotation work without the need for human labor. But if that were the case the annotation software would already perform the task that we are trying to teach neural networks to solve.

Therefore humans are indispensable in the task of generate ground truth annotations and the best we can hope for is to find ways to support human annotators that allows them to work more efficiently.

When a human makes an annotation we call that a human intervention. The assumption is that a human intervention results in an annotation of high quality that is good enough to be considered as ground truth. The annotations that the software creates automatically without human intervention will most likely have a lower quality due to the lack of human supervision. The ratio between human intervention and automatically created annotations will have an impact on the overall quality of the annotated dataset.

## 1.2 Annotation of Sequences

As mentioned above this thesis concerns annotation of training data for 3D object detection. Such training data usually consists of sequences of images, collected by a camera, and point clouds, collected by a LiDAR or RADAR[1]. Since nearby cars and pedestrians will be captured in multiple consecutive time instances, the task at hand is to annotate sequences of these objects. If a human annotator provides annotations at some time instances, then a reasonable hypothesis is that those annotations could be used to estimate the other annotations in the sequence.

## 1.3 Aim

The overarching aim of this thesis is to allow for a more sparse human annotation, which would lead to faster annotation of large training datasets for 3D object detection. To achieve that goal we have implemented an algorithm that can estimate object trajectories by combining information from (i) a sparse set of annotated 3D bounding boxes, and (ii) kinematic models of object motion. Limitations will be discussed in detail in Section 1.7, but one important limitation is that we will only estimate the trajectories of moving cars.

The specific research questions that this project will try to answer are the following:

- Which motion model yields the most accurate trajectory estimations for cars?
- How does the level of human intervention affect the quality of the automatically generated annotations?

[1] radio detection and ranging

## 1.4 Dataset

In order to evaluate the performance of the algorithm, an annotated dataset is needed. We have chosen the Nuscenes dataset [1] that features 1000 scenes of approximately 20 s each and a full sensor suite that captures 360° of the surrounding for all modalities. The scenes have been recorded in Boston and Singapore. In total, there are almost 1.5 million camera images and 400 000 LiDAR sweeps available with 1.1 million 3D bounding boxes annotated at a frequency of 2 Hz.

Since this project does not involve training of neural networks we have chosen to use the Nuscenes teaser dataset, which contains 100 scenes. This is sufficiently large to be used for evaluation. The sensor setup is:

- one spinning LiDAR sensor – capture frequency 20 Hz
- five RADAR sensors – capture frequency 13 Hz
- six cameras – capture frequency 12 Hz
- inertial measurement unit (IMU)
- global positioning system (GPS)

## 1.5 Problem Formulation

The way the challenge of increasing the level of automation has been addressed in this thesis is to solve a trajectory estimation problem which "interpolates" the human annotations with suggested annotations. The trajectory estimation problem has been solved by employing a Bayesian fixed-interval smoother approach. By using a smoothing technique, the estimated annotations could be based on past, present, and future human annotations, thus using the information from all human annotations to make each annotation. The outcome is that the human does not have to provide all annotations which thereby increases the level of automation. In addition to this, different motion models will be compared based on their ability to estimate trajectories accurately.

The idea behind smoothing is to combine information about each state in an optimal way. In essence there are three different sources of information available to make each estimate: (i) knowledge about the state in adjacent points, (ii) knowledge about how the state tends to evolve over time (i.e. motion models, see Section 2.1), and (iii) a measurement of the current state. Chapter 2 describes how these sources of information can be fused optimally under some assumptions.

The smoothing problem is normally modeled by a state space model (SSM) and the graphical representation of an SSM is illustrated in Figure 1.1 where the subscripts denote different time instances. The blue nodes represent the state vectors $\mathbf{x}_k$ and

the yellow nodes hold the observation vectors $\mathbf{y}_k$, where the latter one always has the configuration

$$\mathbf{y}_k = \begin{bmatrix} x_k & y_k & z_k & \phi_k & w_k & l_k & h_k \end{bmatrix}^\top. \tag{1.1}$$

These values constitute a bounding box: the center 3D coordinates, $x_k$, $y_k$, $z_k$, the yaw angle, $\phi_k$, and the width, length and height of the box, $w_k$, $l_k$, $h_k$. The state vector $\mathbf{x}_k$ will also contain the elements in Equation (1.1) along with some more variables depending on the choice of motion model.

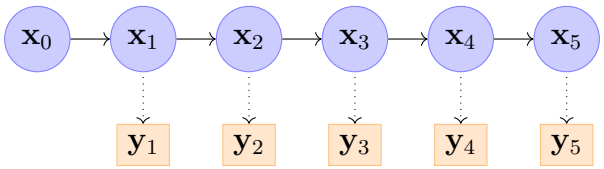



**Figure 1.1:** The trajectory estimation problem can be represented graphically as a state space model. The hidden states, $\mathbf{x}_k$, are the states that are subject to estimation and the observed states, $\mathbf{y}_k$, are the states which in this thesis will be the human annotations. The observed vectors will also be referred to as measurements.

The SSM in Figure 1.1 has 5 time instances (the initial state $\mathbf{x}_0$ is not counted here). For an SSM with $T$ time instances the smoothing problem is formulated by the maximization

$$\hat{\mathbf{x}}_k = \underset{\mathbf{x}_k}{\operatorname{argmax}}\, p(\mathbf{x}_k|\mathbf{y}_1, \ldots, \mathbf{y}_T) \quad \forall k = 0, \ldots, T. \tag{1.2}$$

The sequence $\hat{\mathbf{x}}_{0:T} = (\hat{\mathbf{x}}_0, \ldots, \hat{\mathbf{x}}_T)$ is the estimated trajectory and there are several different smoothing algorithms available that solve this maximization. The algorithm used in this thesis is described in Chapter 2.

To simulate the change of human intervention, different numbers of measurements were removed from the dense measurement sequence represented in Figure 1.1. To evaluate the performance of the estimates, they were simply compared to the removed annotations using different metrics.

## 1.6 Related Work and Contributions

Attempts to build a semi-automatic annotation system have been made before. In the making of the Berkeley Deep Drive dataset, Yu et al. [2] created a labelling system with the ability to support annotation of bounding boxes, semantic and instance segmentation. They used a Fast-RCNN model to generate object detections as a means to suggest bounding box annotations to the annotators. This results in a more efficient annotation process which reduces the total cost.

This thesis is concerned with the annotations of 3D objects and the main contributions of the thesis are:

- The motion model coordinated turn is established to be the model that best describes the motion of cars in city traffic.
- A smoothing algorithm that reduces the need for human annotations without introducing large errors to the estimated annotations.
- A square-root version of statistical linear regression has been derived allowing for more numerically stable statistical predictors, filters, and smoothers.

## 1.7 Limitations

For this project we have chosen to use the dataset Nuscenes which is a publicly available autonomous driving dataset [1]. The annotations in Nuscenes occur at a frequency of 2 Hz. Since we will consider these annotations as ground truth in this study, we cannot evaluate the error for the estimated trajectories at a frequency higher than 2 Hz. Additionally, we limit the quality of the ground truth to the quality provided by Scale – the annotating partner to Nuscenes.

The annotations in Nuscenes [1] have identities such that annotations for the same object can be distinguished between for different annotated time instances. This project will be limited to the case when such IDs for the annotations exist. For Annotell this means that the human annotators must provide these IDs.

Kinematic motion models usually describe the movement of vehicles in real world 3D coordinates. This requires access to 3D coordinates for the surrounding vehicles and for the ego vehicle. In some cases, that is not what Annotell has access to and thereby it is a restriction to assume that 3D coordinates for the aforementioned objects are available. This project will be limited to the cases when 3D world coordinates for the surrounding and ego vehicle are accessible.

Since this thesis aims to aid annotators in their annotation process we will assume that the data is recorded beforehand and the algorithm developed in this project will be run offline. This allows us to use future data points and remove the computational time restriction inherent to real-time systems to make our estimations.

We will also assume that the world is flat and that objects have zero roll and pitch. Hence the only rotation of the bounding boxes that we model is described by the yaw angle.

# 2

# Theory

This thesis aims to estimate trajectories of cars. This can be achieved by using Bayesian inference, the relevant details of which will be explained in this chapter. Since this thesis only considers annotations, the data have been collected beforehand which makes future[1] information available when estimating the variables and hence a smoothing approach that can accommodate non-linear state mappings will be the focus of this chapter.

## 2.1 Probabilistic State Space Models

The modeling of state transitions is integral to solving a smoothing problem since they model the "trends" in how states are likely to evolve over time. This information is referred to as the prior in Bayesian inference.

This thesis will use motion models that describe how the current state is transformed to the next state

$$\mathbf{x}_{k+1} = f_k(\mathbf{x}_k) + \mathbf{q}_k, \quad \mathbf{q}_k \sim \mathcal{N}(\mathbf{0}, Q_k) \tag{2.1}$$

and a measurement model that describes how the observed measurement is mapped from the underlying state

$$\mathbf{y}_k = h_k(\mathbf{x}_k) + \mathbf{r}_k, \quad \mathbf{r}_k \sim \mathcal{N}(\mathbf{0}, R_k) \tag{2.2}$$

where $k$ is the time index and $f_k(\cdot)$ and $h_k(\cdot)$ are possibly non-linear mappings. The terms $\mathbf{q}_k$ and $\mathbf{r}_k$ represent statistical noise from a multivariate normal (Gaussian) distribution $\mathcal{N}(\cdot, \cdot)$ where the first argument is the mean vector and the second is the covariance matrix. Both $\mathbf{q}_k$ and $\mathbf{r}_k$ have zero mean and their corresponding covariance matrices are $Q_k$ and $R_k$.

The motion models are simplifications and can not describe the dynamics of the physical motion perfectly. The Gaussian noise, $\mathbf{q}_k$, introduces an uncertainty in the model which can be seen as accommodating the model errors of the chosen motion model. When using this class of motion models the trajectory forms a Markov

[1] In this context the future is relative to the current measurement. E.g. if the smoother estimates the state at time $k = 3$ information from time $k = 4, 5, 6, \ldots$ would be in the future.

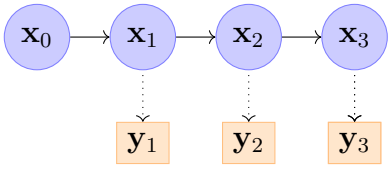



**Figure 2.1:** The state evolution can be modeled as a state space model. The blue nodes, $\mathbf{x}_i$ represent the cuboid at different time instances and the orange nodes represent the measurements of the state. The horizontal arrows represent the motion model and the vertical arrows represent the measurement model.

sequence. This means that the states $\mathbf{x}_{k+1:T}$ are conditionally independent of $\mathbf{x}_{0:k-1}$ given $\mathbf{x}_k$ [3, p.52]. In Figure 2.1 the Markov sequence has been visualized as an SSM. The motion model (2.1) is represented by the horizontal arrows and the measurement model (2.2) is represented by the dotted, vertical arrows.

## 2.2 Posterior Approximation

To make the trajectory estimations we seek to estimate the state sequence $\mathbf{x}_{0:T} = (\mathbf{x}_0, \ldots, \mathbf{x}_T)$ given the measurement sequence $\mathbf{y}_{1:T} = (\mathbf{y}_1, \ldots, \mathbf{y}_T)$. The posterior probability $p(\mathbf{x}_{0:T}|\mathbf{y}_{1:T})$ holds all information about the state sequence $\mathbf{x}_{0:T}$. By using the Markovian property of the process, $p\left(\mathbf{x}_{0:T}|\mathbf{y}_{1:T}\right)$ can be factorized as

$$p(\mathbf{x}_{0:T}|\mathbf{y}_{1:T}) \propto \prod_{k=1}^{T} [p(\mathbf{y}_k|\mathbf{x}_k)p(\mathbf{x}_k|\mathbf{x}_{k-1})]p(\mathbf{x}_0) \tag{2.3}$$

where $p\left(\mathbf{x}_k|\mathbf{x}_{k-1}\right)$ and $p\left(\mathbf{y}_k|\mathbf{x}_k\right)$ in the above factorization are simply (2.1) and (2.2) re-formulated as

$$p(\mathbf{x}_k|\mathbf{x}_{k-1}) = \mathcal{N}(\mathbf{x}_k; f_k(\mathbf{x}_{k-1}), Q_{k-1}) \tag{2.4}$$

$$p(\mathbf{y}_k|\mathbf{x}_k) = \mathcal{N}(\mathbf{y}_k; h_k(\mathbf{x}_k), R_k). \tag{2.5}$$

The prior distribution for $\mathbf{x}_0$ is $p(\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_0; \hat{\mathbf{x}}_0, P_0)$. In general, the posterior cannot be expressed in closed form, which is why it must be approximated. García-Fernández et al. [4; 5] assume affine approximations of Equations (2.4) and (2.5) given by

$$f_k(\mathbf{x}_k) \approx F_k\mathbf{x}_k + \mathbf{a}_k + \mathbf{e}_k, \quad \mathbf{e}_k \sim \mathcal{N}(\mathbf{0}, \Lambda_k) \tag{2.6}$$

$$h_k(\mathbf{x}_k) \approx H_k\mathbf{x}_k + \mathbf{b}_k + \mathbf{g}_k, \quad \mathbf{g}_k \sim \mathcal{N}(\mathbf{0}, \Omega_k) \tag{2.7}$$

where $F_k$ is an $N \times N$ matrix and $H_k$ is an $M \times N$ matrix provided that the dimensions of $\mathbf{x}_k$ and $\mathbf{y}_k$ are $N$ and $M$ respectively. $\mathbf{a}_k$ and $\mathbf{b}_k$ are vectors of length $N$ and $M$ respectively. The last terms $\mathbf{e}_k$ and $\mathbf{g}_k$ are Gaussian noise vectors and their purpose is to model the non-linear behaviour of $f_k(\cdot)$ and $h_k(\cdot)$.

The equations (2.6) and (2.7) are enabling approximations that permit an approximation to $p(\mathbf{x}_{0:T}|\mathbf{y}_{1:T})$ in closed form. The parameters of these approximations are denoted by

$$\Theta = (F_{0:T-1}, \mathbf{a}_{0:T-1}, \Lambda_{0:T-1}, H_{1:T}, \mathbf{b}_{1:T}, \Omega_{1:T}) \tag{2.8}$$

which can be used to express the posterior approximation

$$p(\mathbf{x}_{0:T}|\mathbf{y}_{1:T}) \approx q^{\Theta}(\mathbf{x}_{0:T}|\mathbf{y}_{1:T}) \propto \prod_{k=1}^{T} [q^{\Theta}(\mathbf{y}_k|\mathbf{x}_k) q^{\Theta}(\mathbf{x}_k|\mathbf{x}_{k-1})] p(\mathbf{x}_0) \tag{2.9}$$

where

$$\begin{aligned} q^{\Theta}(\mathbf{x}_k|\mathbf{x}_{k-1}) &= \mathcal{N}(\mathbf{x}_k; F_{k-1}\mathbf{x}_{k-1} + \mathbf{a}_{k-1}, \Lambda_{k-1} + Q_{k-1}) \\ q^{\Theta}(\mathbf{y}_k|\mathbf{x}_k) &= \mathcal{N}(\mathbf{y}_k; H_k\mathbf{x}_k + \mathbf{b}_k, \Omega_k + R_k). \end{aligned}$$

The approximation $q^{\Theta}(\mathbf{x}_{0:T}|\mathbf{y}_{1:T})$ is Gaussian and can be computed using an affine RTS smoother [4]. More specifically, the smoother computes the first two moments, denoted by $\hat{\mathbf{x}}_{k|T}$ and $P_{k|T}$, of the marginal pdfs

$$q^{\Theta}(\mathbf{x}_k|\mathbf{y}_{1:T}) = \mathcal{N}(\mathbf{x}_k; \hat{\mathbf{x}}_{k|T}, P_{k|T}) \tag{2.10}$$

for $k = 0, \ldots, T$. In this thesis $\hat{\mathbf{x}}_{k|T}$ for $k = 1, \ldots, T$ will be used to estimate the trajectories, which constitute the maximum a posteriori (MAP) of $q^{\Theta}(\mathbf{x}_{1:T}|\mathbf{y}_{1:T})$.

## 2.3 Fixed-Interval Smoothing

In the task to annotate previously recorded data, information in all time instances in every sequence is available beforehand. Therefore, fixed-interval smoothing will be used. The two principal steps in an affine fixed-interval smoother are the forward filtering and the backward smoothing. These operations will be outlined in the subsections below. Previous work concerning the derivation and use of these techniques can be found in [3; 6; 4].

### 2.3.1 Filtering

Filtering in a Bayesian context means that state estimations are made using only past and current information in contrast to smoothing which also uses future information. Even though this thesis only considers smoothing to make state estimates, filtering is an essential building block of smoothing. According to [6, pp.132] the filtering equations that the filtering operation solves are

$$p\left(\mathbf{x}_k|\mathbf{y}_{1:k-1}\right) = \int p\left(\mathbf{x}_k|\mathbf{x}_{k-1}\right) p\left(\mathbf{x}_{k-1}|\mathbf{y}_{1:k-1}\right) \mathrm{d}\mathbf{x}_{k-1} \tag{2.11}$$

$$p\left(\mathbf{x}_k|\mathbf{y}_{1:k}\right) \propto p\left(\mathbf{y}_k|\mathbf{x}_k\right) p\left(\mathbf{x}_k|\mathbf{y}_{1:k-1}\right). \tag{2.12}$$

### 2.3.2 Kalman Filtering

The filtering equations do not have an analytical solution for general motion, $p(\mathbf{x}_k|\mathbf{x}_{k-1})$, and measurement models, $p(\mathbf{y}_k|\mathbf{x}_k)$, but for the special case of affine Gaussian mod-

els, the Kalman filter (KF) provides recursion formulas that solve the filtering equations analytically. In the derivation of the KF equations below it is assumed that $p(\mathbf{x}_k|\mathbf{x}_{k-1})$ and $p(\mathbf{y}_k|\mathbf{x}_k)$ are Gaussian affine models.

KF is divided into two steps where the first steps predicts the probability density function (pdf), $p(\mathbf{x}_k|\mathbf{y}_{1:k-1})$, of the state at the next time instance ($t = k$) given the measurements up to the current time instance ($t = k-1$). This step is usually called the prediction step (or time update step) and is equivalent to solving Equation (2.11). The second step in KF is to correct the predicted pdf given the new information from the measurement at current time ($t = k$) which gives the pdf $p(\mathbf{x}_k|\mathbf{y}_{1:k})$. This is usually called the update step and is equivalent to solving Equation (2.12). The following subsection will describe the formulas to perform KF.

### Prediction

As mentioned, the KF is a recursive solution since the solution in each time instance is based on the solution of the previous time instance. The filter solution of the previous step is given by

$$p(\mathbf{x}_{k-1}|\mathbf{y}_{1:k-1}) = \mathcal{N}(\mathbf{x}_{k-1}; \hat{\mathbf{x}}_{k-1|k-1}, P_{k-1|k-1}). \tag{2.13}$$

A new subscript notation has been introduced here: $\hat{\mathbf{x}}_{k-1|k-1}$ and $P_{k-1|k-1}$. These are the moments (mean vector and covariance matrix) of the distribution $p(\mathbf{x}_{k-1}|\mathbf{y}_{1:k-1}) = \mathcal{N}(\mathbf{x}_{k-1}; \hat{\mathbf{x}}_{k-1|k-1}, P_{k-1|k-1})$. In general, the notation can be described as $p(\mathbf{x}_n|\mathbf{y}_{1:m}) = \mathcal{N}(\mathbf{x}_n; \hat{\mathbf{x}}_{n|m}, P_{n|m})$. Given (2.13) the predicted density can be computed using the probabilistic motion model

$$\mathbf{x}_k = F_{k-1}\mathbf{x}_{k-1} + \mathbf{a}_{k-1} + \mathbf{e}_{k-1} + \mathbf{q}_{k-1}, \quad \begin{pmatrix} \mathbf{e}_{k-1} \\ \mathbf{q}_{k-1} \end{pmatrix} \sim \mathcal{N}\left( \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} \Lambda_{k-1} & 0 \\ 0 & Q_{k-1} \end{pmatrix} \right). \tag{2.14}$$

The prediction is the sum of three Gaussian random variables and such a sum can be computed as

$$\mathbf{z} = B_1\mathbf{z}_1 + B_2\mathbf{z}_2 + B_3\mathbf{z}_3 \sim \mathcal{N}\left(B_1\boldsymbol{\mu}_1 + B_2\boldsymbol{\mu}_2, B_1\Sigma_1B_1^\top + B_2\Sigma_2B_2^\top + B_3\Sigma_3B_3^\top\right) \tag{2.15}$$

where $\mathbf{z}_i$ are Gaussian random variables distributed as $\mathbf{z}_i \sim \mathcal{N}(\boldsymbol{\mu}_i, \Sigma_i)$ and the variables $B_i$ are linear mappings.

If the result in Equation (2.15) is combined with Equation (2.14) the predicted densities are obtained as

$$p(\mathbf{x}_k|\mathbf{y}_{1:k-1}) = \mathcal{N}(\mathbf{x}_k; F_{k-1}\hat{\mathbf{x}}_{k-1|k-1} + \mathbf{a}_{k-1}, F_{k-1}P_{k-1|k-1}F_{k-1}^\top + Q_{k-1} + \Lambda_{k-1}). \tag{2.16}$$

Hence, the predicted moments are

$$\hat{\mathbf{x}}_{k|k-1} = F_{k-1}\hat{\mathbf{x}}_{k-1|k-1} + \mathbf{a}_{k-1} \tag{2.17a}$$

$$P_{k|k-1} = F_{k-1}P_{k-1|k-1}F_{k-1}^\top + Q_{k-1} + \Lambda_{k-1}. \tag{2.17b}$$

#### Update

The purpose of the update step is to include the information in the new measurement and thereby improve the estimate given by the prediction step. The mathematical formulation needed to perform the update is listed in (2.18). The derivation of the update step for linear measurement models can be found in [3, pp.57]. The extension to affine measurement models is straight forward and the result is given in [4]. The corresponding equations are

$$\mathbf{v}_k = \mathbf{y}_k - H_k \hat{\mathbf{x}}_{k|k-1} - \mathbf{b}_k \tag{2.18a}$$
$$S_k = H_k P_{k|k-1} H_k^\top + R_k + \Omega_k \tag{2.18b}$$
$$K_k = P_{k|k-1} H_k^\top S_k^{-1} \tag{2.18c}$$
$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + K_k \mathbf{v}_k \tag{2.18d}$$
$$P_{k|k} = P_{k|k-1} - K_k S_k K_k^\top. \tag{2.18e}$$

### 2.3.3 Rauch-Tung-Striebel Smoothing

The state estimates in smoothing are non-causal and conditioned on all available measurements, both past, current, and future. Since more information is available for each estimate we can expect the estimates to be improved compared to filtering except for the last state in the sequence, where no future information exists and the smoothing solution coincides with the filtering solution.

The Rauch-Tung-Striebel smoother (RTS) shares the same fundamental ideas as the KF but operates backwards in time to recursively update the filtering solution with future information at each time step. The equation solved by RTS is

$$p(\mathrm{x}_k|\mathbf{y}_{1:T}) = p(\mathbf{x}_k|\mathbf{y}_{1:k}) \int \frac{p(\mathbf{x}_{k+1}|\mathbf{x}_k) p(\mathbf{x}_{k+1}|\mathbf{y}_{1:T})}{p(\mathbf{x}_{k+1}|\mathbf{y}_{1:k})} \mathrm{d}\mathbf{x}_{k+1} \tag{2.19}$$

as specified in [3, pp.135]. Apart from the smoothing solution from the previous step, $p(\mathbf{x}_{k+1}|\mathbf{y}_{1:T})$, there are no new densities present in this equation. The parts are: filtering solution, $p(\mathbf{x}_k|\mathbf{y}_{1:k})$, motion model, $p(\mathbf{x}_{k+1}|\mathbf{x}_k)$, and predicted density, $p(\mathbf{x}_{k+1}|\mathbf{y}_{1:k})$. The smoothing equation only consolidates these into one single estimate. Assuming that prediction and filtering solutions exist, the equations

$$\begin{aligned} G_k &= P_{k|k} F_k^\top P_{k+1|k}^{-1} \\ \hat{\mathbf{x}}_{k|T} &= \hat{\mathbf{x}}_{k|k} + G_k[\hat{\mathbf{x}}_{k+1|T} - \hat{\mathbf{x}}_{k+1|k}] \\ P_{k|T} &= P_{k|k} + G_k[P_{k+1|T} - P_{k+1|T}] G_k^\top \end{aligned} \tag{2.20}$$

provide the mathematical formulation to carry out RTS smoothing [3, pp.136], where $\hat{\mathbf{x}}_{k|T}$ and $P_{k|T}$ are the moments of the distribution $p(\mathbf{x}_k|\mathbf{y}_{1:T}) = \mathcal{N}(\mathbf{x}_k; \hat{\mathbf{x}}_{k|T}, P_{k|T})$. Importantly, they should not be confused with the moments $\hat{\mathbf{x}}_{n|m}$ and $P_{n|m}$ which were defined in Section 2.3.2. Since the solutions obtained are Gaussian, the solution obtained from smoothing is the MAP estimate of the state.

## 2.4 Iterated Posterior Linearization Smoother

As mentioned in Section 2.3, the RTS smoothing described above is derived for affine models. When dealing with non-affine models an approximation is needed to enable the use of previous results. There are many ways to make these approximations, such as the Gauss-Hermite Kalman, Unscented Kalman, and Extended Kalman filters [3]. Since the algorithm in this thesis has access to past, present, and future information it makes sense to utilize all information possible to find the optimal enabling approximations $\Theta$. The IPLS method, which utilizes all such information, was selected for use in this thesis. This method performs an iterative statistical linear regression (SLR) using the first two moments from the smoothing solution, $\hat{\mathbf{x}}_{k|T}$ and $P_{k|T}$, that were obtained in the previous step.

The idea of iteratively re-linearzing the measurement and motion models based on the best available approximation of the posterior density was introduced by García-Fernández et al. for the filtering problem in [5] and then extended to smoothing in [4].

### 2.4.1 Statistical Linear Regression

The SLR algorithm is a method to approximate a general function $f(\cdot)$ such that

$$f(\mathbf{x}) \approx F^{+}\mathbf{x} + \mathbf{a}^{+}. \tag{2.21}$$

The $F^{+}$ and $\mathbf{a}^{+}$ that constitute the optimal affine approximation of $f(\cdot)$ in the sense of minimizing its mean square error (MSE) with respect to a distribution $p(\mathbf{x})$ are

$$\left(F^{+}, \mathbf{a}^{+}\right) = \underset{(F,a)}{\operatorname{argmin}} \underset{p(\mathbf{x})}{\mathbb{E}} \left[ \left(f(\mathbf{x}) - F\mathbf{x} - \mathbf{a}\right)^{\top} \left(f(\mathbf{x}) - F\mathbf{x} - \mathbf{a}\right) \right] \tag{2.22}$$

$$\Lambda^{+} = \underset{p(\mathbf{x})}{\mathbb{E}} \left[ \left(f(\mathbf{x}) - F^{+}\mathbf{x} - \mathbf{a}^{+}\right) \left(f(\mathbf{x}) - F^{+}\mathbf{x} - \mathbf{a}^{+}\right)^{\top} \right] \tag{2.23}$$

where $\Lambda^{+}$ is the corresponding MSE matrix [7]. In the enabling approximations (2.6) and (2.7) the first two terms are the affine approximation parameterized according to Equation (2.22). The third term is Gaussian noise with zero mean and its covariance matrix being the MSE matrix according to Equation (2.23).

Note that the expectation is taken with respect to the pdf, $p(\mathbf{x})$. This means that motion models can be linearized based on the smoothing solution (if available) which is the most accurate estimation of the first two moments of the state distribution at each time instance. During the iterations of the IPLS the smoothing solution is updated which allows for a more accurate linearization of $f(\cdot)$, which in turn makes the resultant smoothing solution more accurate with each iteration.

Given a function $f(\cdot)$ and a pdf $p(\mathbf{x})$ with first and second moments $\hat{\mathbf{x}}$ and $P$ the SLR of $f(\cdot)$ w.r.t. $p(\mathbf{x})$ is given by [7]

$$F^+ = \Psi^\top P^{-1} \tag{2.24a}$$
$$\mathbf{a}^+ = \hat{\mathbf{z}} - F^+\hat{\mathbf{x}} \tag{2.24b}$$
$$\Lambda^+ = \Phi - F^+ P (F^+)^\top \tag{2.24c}$$

where

$$\hat{\mathbf{z}} = \int f(\mathbf{x})p(\mathbf{x})\mathrm{d}\mathbf{x} \tag{2.25a}$$
$$\Psi = \int (\mathbf{x} - \hat{\mathbf{x}})(f(\mathbf{x}) - \hat{\mathbf{z}})^\top p(\mathbf{x})\mathrm{d}\mathbf{x} \tag{2.25b}$$
$$\Phi = \int (f(\mathbf{x}) - \hat{\mathbf{z}})(f(\mathbf{x}) - \hat{\mathbf{z}})^\top p(\mathbf{x})\mathrm{d}\mathbf{x}. \tag{2.25c}$$

However, the moments in Equations (2.25) are, in general, not possible to compute in closed form. Instead, we can use sigma point methods [3] to approximate these integrals as

$$\hat{\mathbf{z}} \approx \sum_{j=1}^{m} \omega_j \mathcal{Z}_j \tag{2.26a}$$
$$\Psi \approx \sum_{j=1}^{m} \omega_j \left(\mathcal{X}_j - \hat{\mathbf{x}}\right)\left(\mathcal{Z}_j - \hat{\mathbf{z}}\right)^\top \tag{2.26b}$$
$$\Phi \approx \sum_{j=1}^{m} \omega_j \left(\mathcal{Z}_j - \hat{\mathbf{z}}\right)\left(\mathcal{Z}_j - \hat{\mathbf{z}}\right)^\top \tag{2.26c}$$

where $\mathcal{X}_j$ is the sigma point, $\omega_j$ the corresponding weight, and $\mathcal{Z}_j = f(\mathcal{X}_j)$. Given these approximations, $F^+, \mathbf{a}^+$, and $\Lambda^+$ can be computed using Equations (2.24).

### 2.4.2 Iterative Improvement of Enabling Approximations

As described in Section 2.2, the strategy is to approximate the posterior $p(\mathbf{x}_{0:T}|\mathbf{y}_{1:T}) \approx q^\Theta(\mathbf{x}_{0:T}|\mathbf{y}_{1:T})$ using the enabling approximations $\Theta$ (see Equation (2.8). Ideally we would like to find the $\Theta$ that minimizes the Kullback-Leibler divergence (KLD)

$$D(p(\mathbf{x}_{0:T}|\mathbf{y}_{1:T})||q^\Theta(\mathbf{x}_{0:T}|\mathbf{y}_{1:T})) \tag{2.27}$$

where $D(\cdot||\cdot)$ is the KLD [8] which is defined as

$$D(p(x)||q(x)) = \int_x p(x) \log \frac{p(x)}{q(x)}\mathrm{d}x. \tag{2.28}$$

The IPLS iteratively improves the enabling approximations. In iteration $i$ the approximation is given by

$$\Theta^i = (F^i_{0:T-1}, \mathbf{a}^i_{0:T-1}, \Lambda^i_{0:T-1}, H^i_{1:T}, \mathbf{b}^i_{1:T}, \Omega^i_{1:T}) \tag{2.29}$$

and the aim is to obtain $\Theta^{i+1}$. García-Fernández et al. describe in [4] that this can be done by performing SLR of the nonlinear functions with respect to the latest posterior approximation $p(\mathbf{x}_{0:T}|\mathbf{y}_{1:T}) \approx q^{\Theta^i}(\mathbf{x}_{0:T}|\mathbf{y}_{1:T})$. The steps of the IPLS are summarized in Algorithm 1.

**Algorithm 1** Forward-backward IPLS

**Data:** Prior mean:$u_0^1 = \mathbf{x}_0$, prior covariance:$W_0^1 = P_0$, motion model: $f_k(\cdot)$, $Q_k$, measurement model:$h_k(\cdot)$, $R_k$, measurement sequence: $\mathbf{y}$

**Result:** Posterior moments: $s_k^J$, $W_k^J$ for $k = \{0, \ldots, T\}$

`/* Below` $p$, $u$, $s$ `are the predicted, updated, and smoothing solutions respectively and` $P$, $U$, $S$ `are their corresponding covariance matrices. The subscript refers to which time the estimate refers to. If there is no subscript it refers to the set containing the variable at all time instances. */`

`/* Initialization: Compute` $s_k^1$, $S_k^1$ `for` $k \in \{0, \ldots, T\}$ `by a sigmapoint smoother. */`

`/* obtain smoothing solution from affine RTS */`

$(s^1, S^1) =$ `RTS` $(u^1, U^1, p^1, P^1, F^1, a^1, \Lambda^1)$

**for** *i=1 to J-1* **do**

  **for** *k = 0 to T* **do**

    $(F_k^{i+1}, \mathbf{a}_k^{i+1}, \Lambda_k^{i+1}) =$ `SLR` $(f_k(\cdot), s_k^i, S_k^i)$

    $(H_k^{i+1}, \mathbf{b}_k^{i+1}, \Omega_k^{i+1}) =$ `SLR` $(h_k(\cdot), s_k^i, S_k^i)$

  **end**

  **for** *k = 1 to T* **do**

    $(p_{k+1}^{i+1}, P_{k+1}^{i+1}) =$ `prediction` $(F_k^{i+1}, \mathbf{a}_k^{i+1}, \Lambda_k^{i+1}, u_k^i, Q_k)$

    **if** *meas exists at time k+1* **then**

      $(u_{k+1}^{i+1}, U_{k+1}^{i+1}) =$ `update` $(p_{k+1}^{i+1}, P_{k+1}^{i+1}, H_k^{i+1}, \mathbf{b}_k^{i+1}, \Omega_k^{i+1}, R_{k+1}, \mathbf{y}_{k+1})$

    **else** pass prediction as posterior

      $(u_{k+1}^{i+1}, U_{k+1}^{i+1}) = (p_{k+1}^{i+1}, P_{k+1}^{i+1})$

    **end**

  **end**

  $(s^{i+1}, S^{i+1}) =$ `RTS` $(u^{i+1}, U^{i+1}, p^{i+1}, P^{i+1}, F^{i+1}, a^{i+1}, \Lambda^{i+1}, H^{i+1}, b^{i+1}, \Omega^{i+1})$

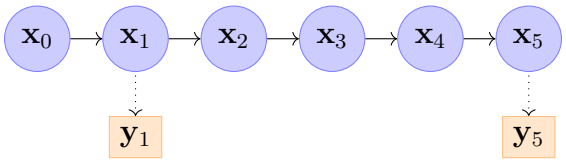
**end**

### 2.4.3 Initialization

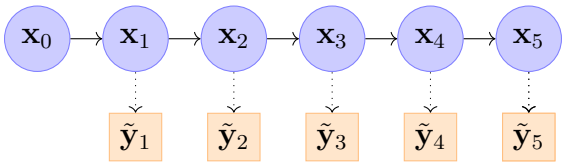
To initialize the IPLS, a normal RTS smoother using the constant velocity (CV) motion model was run to make state estimations for each time instance. The idea is to make an initial trajectory that is closer to the optimal trajectory for the non-linear motion model. If the initial trajectory is close enough to the solution this should guarantee convergence [4] to optimum and reduce the number of iterations needed.

The estimates from the CV solution can be mapped into the measurement space using the the measurement equation to form artificial measurement sequences. One

advantage with this approach is that the artificial measurement sequence is dense[2] which allows for every prediction step to be updated with a measurement, see Figure 2.2.




**(a)** The original measurement sequence is possibly sparse and does not have measurment for each time instance.




**(b)** After initialization has been run the measurement sequence is dense.

**Figure 2.2:** After the initialization has been run there exist artificial measurements, here denoted $\tilde{\mathbf{y}}_k$, that will allow the first solution of the IPLS iteration to be closer to the optimum compared to if the IPLS only had had the original measurements available to make the estimation.

The mapping from state space to measurement space is done using

$$\tilde{\mathbf{y}}_k = H\hat{\mathbf{x}}_{k|T}^{\mathrm{CV}} \tag{2.30}$$

and

$$\tilde{R}_k = HP_{k|T}^{\mathrm{CV}}H^\top \tag{2.31}$$

where $\hat{\mathbf{x}}_{k|T}^{\mathrm{CV}}$ and $P_{k|T}^{\mathrm{CV}}$ are the resulting moments from the initialization.

## 2.5 Square-root Implementations

Implementations of smoothing algorithms may suffer from numerical instability. This is caused by rounding errors which in turn may cause the covariance matrices involved in the computations to be negative definite. Covariance matrices are by definition semi-positive definite, a property which is guaranteed by square-root implementations of smoothing algorithms [6; 9].

Square-root forms of filtering and smoothing algorithms propagate the square-root, $P^{1/2}$, which is a matrix such that $P = P^{1/2}P^{\top/2}$ where $P$ is the original covariance matrix and $P^{\top/2} = (P^{1/2})^\top$. The non-square-root implementations propagate the

[2] I.e. measurement exists for all time instances.

full matrix $P$ which is numerically less stable than propagating $P^{1/2}$. One numerical advantage with square-root forms is that the covariance matrices are guaranteed to be positive definite [9]. Another advantage is that the precision is increased since the condition number of $P^{1/2}$ is proportional to the square-root of the condition number of $P$. For this reason, the precision of the covariance is doubled [9]. The pseudo code that describes the implementation is analogous to the normal IPLS case but with the difference that the square root matrices described in this section are propagated through the prediction, update, and smoothing steps. The results presented in this section are inspired by Rutten [9] and the squre-root variant of the SLR algorithm is a contribution by this thesis.

### 2.5.1 QR-decomposition

Square-root implementations rely on the QR-decomposition of an $m \times n$ matrix $B$

$$B = QR \tag{2.32}$$

where $Q$ is a orthogonal matrix (with the property $Q^\top Q = I$) and $R$ is an upper-triangular matrix. Although the name of the decomposition method stems from the notation of these matrices they should not be confused with the process and measurement noise covariance matrices $Q_k$ and $R_k$. Therefore we will change to the notation

$$\Theta = Q \tag{2.33}$$

$$R_\ell = R^\top \tag{2.34}$$

$$A = B^\top = R_\ell \Theta^\top \iff A\Theta = R_\ell \tag{2.35}$$

where $R_\ell$ is lower-triangular. The central idea in a square-root implementation is to form the matrix $A$ in a smart way that makes it possible to extract a solution from $R_\ell$. The following sections will describe how this can be done to implement a square-root version of the IPLS.

### 2.5.2 Square-root Prediction

The prediction step in the affine RTS smoother is described by the equations in (2.17). The square-root version of the prediction step will be to form the instrumental matrix $A_p$ as

$$A_p = \begin{bmatrix} Q_k^{1/2} & \Lambda_k^{1/2} & F_k P_{k|k}^{1/2} \\ 0 & 0 & P_{k|k}^{1/2} \end{bmatrix} \tag{2.36}$$

which yields

$$A_p A_p^\top = \begin{bmatrix} F_k P_{k|k} F_k^\top + Q_k + \Lambda_k & F_k P_{k|k} \\ P_{k|k} F_k^\top & P_{k|k} \end{bmatrix} = \begin{bmatrix} P_{k+1|k} & F_k P_{k|k} \\ P_{k|k} F_k^\top & P_{k|k} \end{bmatrix} \tag{2.37}$$

where the top left block can be identified as the original prediction equation (2.17). The QR-decomposition of $A_p^\top$ will result in matrices $\Theta_p$ and $R_{\ell p}$ such that

$$A_p^\top = \Theta_p R_{\ell p}^\top. \tag{2.38}$$

Multiplying $A_p$ with $A_p^\top$ will then lead to

$$A_p A_p^\top = R_{\ell p}\Theta_p^\top \Theta_p R_{\ell p}^\top = R_{\ell p} R_{\ell p}^\top \tag{2.39}$$

where the matrix $R_{\ell p}$ is composed of the block matrices $X_p$, $Y_p$, and $Z_p$ as

$$R_{\ell p} = \begin{bmatrix} X_p & 0 & 0 \\ Y_p & Z_p & 0 \end{bmatrix}. \tag{2.40}$$

Multiplying $R_{\ell p}$ with its transpose yields

$$R_{\ell p} R_{\ell p}^\top = \begin{bmatrix} X_p X_p^\top & X_p Y_p^\top \\ Y_p X_p^\top & Y_p Y_p^\top + Z_p Z_p^\top \end{bmatrix}. \tag{2.41}$$

By (2.39) we have

$$\begin{bmatrix} P_{k+1|k} & F_k P_{k|k} \\ P_{k|k} F_k^\top & P_{k|k} \end{bmatrix} = \begin{bmatrix} X_p X_p^\top & X_p Y_p^\top \\ Y_p X_p^\top & Y_p Y_p^\top + Z_p Z_p^\top \end{bmatrix} \tag{2.42}$$

and from this equation we can identify $X_p = P_{k+1|k}^{1/2}$. Already at this stage it is possible to identify the smoothing gain as

$$Y_p X_p^{-1} = Y_p X_p^\top (X_p^\top)^{-1} X_p^{-1} = Y_p X_p^\top (X_p X_p^\top)^{-1} = P_{k|k} F_k^\top P_{k+1|k}^{-1} = G_k \tag{2.43}$$

which will be required later on. Equating the lower right-hand block matrices gives

$$\begin{aligned} Z_p Z_p^\top &= P_{k|k} - Y_p Y_p^\top \\ &= P_{k|k} - Y_p X_p^\top (X_p X_p^\top)^{-1} X_p Y_p^\top \\ &= P_{k|k} - P_{k|k} F_k^\top P_{k+1|k}^{-1} F_k P_{k|k} \\ &= P_{k|k} - G_k P_{k+1|k} G_k^\top \end{aligned} \tag{2.44}$$

and we can identify $Z_p = (P_{k|k} - G_k P_{k+1|k} G_k^\top)^{1/2}$ which is also needed in the square-root smoothing step. To include the time instance $k$ in the notation for this matrix, $Z_{p,k}$ will be used to denote the matrix with the property $Z_{p,k} Z_{p,k}^\top = P_{k|k} - G_k P_{k+1|k} G_k^\top$. To summarize, the important steps in the square-root prediction are:

**Algorithm 2** Square root prediction

**Data:** $\hat{\mathbf{x}}_{k|k}$, $P_{k|k}^{1/2}$, $F_k$, $\mathbf{a}_k$, $Q_k^{1/2}$ and $\Lambda_k^{1/2}$

**Result:** $\hat{\mathbf{x}}_{k+1|k}$, $P_{k+1|k}^{1/2}$, $G_k$ and $Z_{p,k}$

- Form the matrix $A_p$ as in Equation (2.36).
- Perform QR-decomposition on the matrix $A_p^T$ to obtain $R_{\ell p}$.
- From $R_{\ell p}$ as described in Equation (2.40) identify the matrices
  - $X_p = P_{k+1|k}^{1/2}$
  - $Y_p X_p^{-1} = G_k$
  - $Z_{p,k} = (P_{k|k} - G_k P_{k+1|k} G_k^T)^{1/2}$ `/* This is not needed in the prediction but will be used in the smoothing step */`
- Compute $\hat{\mathbf{x}}_{k+1|k} = F_k \hat{\mathbf{x}}_{k|k} + \mathbf{a}_k$.

### 2.5.3 Square-root Update

The update step is described by the equations in (2.18). To implement a square-root update step an instrumental matrix $A_u$ can be formed as

$$A_u = \begin{bmatrix} R_k^{1/2} & \Omega_k^{1/2} & H_k P_{k|k-1}^{1/2} \\ 0 & 0 & P_{k|k-1}^{1/2} \end{bmatrix}. \tag{2.45}$$

Multiplication with its transpose gives the matrix equation

$$A_u A_u^\top = \begin{bmatrix} H_k P_{k|k-1} H_k^\top + R_k + \Omega_k & H_k P_{k|k-1} \\ P_{k|k-1} H_k^\top & P_{k|k-1} \end{bmatrix} = \begin{bmatrix} S_k & H_k P_{k|k-1} \\ P_{k|k-1} H_k^\top & P_{k|k-1} \end{bmatrix} \tag{2.46}$$

where the top left block can be identified as Equation (2.18b) for the system uncertainty. Analogously to the prediction step the QR-decomposition $A_u^\top = \Theta_u R_{\ell u}^\top$ is made to get a lower-triangular matrix

$$R_{\ell u} = \begin{bmatrix} X_u & 0 & 0 \\ Y_u & Z_u & 0 \end{bmatrix}. \tag{2.47}$$

The equality $A_u A_u^\top = R_{\ell u} R_{\ell u}^\top$ can be explicitly expressed as

$$\begin{bmatrix} S_k & H_k P_{k|k-1} \\ P_{k|k-1} H_k^\top & P_{k|k-1} \end{bmatrix} = \begin{bmatrix} X_u X_u^\top & X_u Y_u^\top \\ Y_u X_u^\top & Y_u Y_u^\top + Z_u Z_u^\top \end{bmatrix} \tag{2.48}$$

where Kalman gain, $K_k$, can be computed from $X_u$ and $Y_u$ by

$$Y_u X_u^{-1} = Y_u X_u^\top (X_u^\top)^{-1} X_u^{-1} = Y_u X_u^\top (X_u X_u^\top)^{-1} = P_{k|k-1} H_k^\top S_k^{-1} = K_k. \tag{2.49}$$

Equating the lower right-hand block matrices gives

$$\begin{aligned} Z_u Z_u^\top &= P_{k|k-1} - Y_u Y_u^\top \\ &= P_{k|k-1} - Y_u X_u^\top (X_u X_u^\top)^{-1} X_u Y_u^\top \\ &= P_{k|k-1} - P_{k|k-1} H_k^\top S_k^{-1} H_k P_{k|k-1} \\ &= P_{k|k-1} - K_k S_k K_k^\top \end{aligned} \tag{2.50}$$

and $Z_u$ can therefore be identified as the square-root of the updated covariance $P_{k|k}$. The square-root form of the update step can now be concluded.

The update step above is derived for affine measurement models but as mentioned in Section 2.4 this thesis is only concerned with linear measurement models. To obtain the update step for linear models one can just simply set $\mathbf{b}$ and $\Omega$ to 0 where they occur in the equations.

**Algorithm 3** Square root update

**Data:** $\hat{\mathbf{x}}_{k|k-1}$, $\mathbf{y}_k$, $P^{1/2}_{k|k-1}$, $H_k$, $\mathbf{b}_k$, $R^{1/2}_k$ and $\Omega^{1/2}_k$.

**Result:** $\hat{\mathbf{x}}_{k|k}$, $P^{1/2}_{k|k}$

- Form the matrix $A_u$ as in Equation (2.45).
- Perform the QR-decomposition $A_u^T = \Theta_u R_{\ell u}^T$ to obtain $R_{\ell u}$.
- From $R_{\ell u}$ as described in Equation (2.47) identify the matrices
  - $Y_u X_u^{-1} = K_k$
  - $Z_u = P^{1/2}_{k|k}$
- Compute $\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + K_k(\mathbf{y}_k - H_k\hat{\mathbf{x}}_{k|k-1} + \mathbf{b}_k)$

### 2.5.4 Square-root Smoothing

The smoothing step is described by the equations in (2.20). To implement a square-root version of this step is now relatively easy since the square-root prediction provides some of the information that is needed (i.e. $Z_{p,k}$ and $G_k$). The instrumental matrix is now formed as

$$A_s = \begin{bmatrix} Z_{p,k} & G_k P^{1/2}_{k+1|T} \end{bmatrix}. \tag{2.51}$$

Multiplication with its transpose gives the matrix equation

$$A_s A_s^\top = Z_{p,k} Z_{p,k}^\top + G_k P_{k+1|T} G_k^\top. \tag{2.52}$$

Substituting $Z_{p,k} Z_{p,k}^\top$ gives

$$\begin{aligned} A_s A_s^\top &= P_{k|k} - G_k P_{k+1|k} G_k^\top + G_k P_{k+1|T} G_k^\top \\ &= P_{k|k} - G_k (G_k P_{k+1|T} - P_{k+1|k}) G_k^\top \end{aligned} \tag{2.53}$$

which is exactly the smoothed covariance $P_{k|T}$. The QR-decomposition $A_s^\top = \Theta_s R_{\ell s}^\top$ results in a matrix

$$R_{\ell s} = \begin{bmatrix} X_s & 0 \end{bmatrix}. \tag{2.54}$$

From the fact that $A_s A_s^\top = R_{\ell s} R_{\ell s}^\top$ it holds that $X_s X_s^\top = P_{k|k} - G_k(G_k P_{k+1|T} - P_{k+1|k}) G_k^\top$ which makes it easy to identify $X_s = P^{1/2}_{k|T}$. The smoothing step can now be summarized.

**Algorithm 4** Square root smoothing

**Data:** $\hat{\mathbf{x}}_{k+1|T}$, $\hat{\mathbf{x}}_{k|k}$, $\hat{\mathbf{x}}_{k+1|k}$, $P^{1/2}_{k+1|T}$, $Z_{p,k}$ and $G_k$.

**Result:** $\hat{\mathbf{x}}_{k|T}$ and $P^{1/2}_{k|T}$

- Form the matrix $A_s$ as in Equation (2.51).
- Perform the QR-decomposition $A_s^T = \Theta_s R_{\ell s}^T$ to obtain the lower-triangular matrix $R_{\ell s}$.
- From $R_{\ell s}$ as described in Equation (2.54) identify the matrix $X_s = P^{1/2}_{k|T}$.
- Compute $\hat{\mathbf{x}}_{k|T} = \hat{\mathbf{x}}_{k|k} + G_k(\hat{\mathbf{x}}_{k+1|T} - \hat{\mathbf{x}}_{k+1|k})$.

### 2.5.5 Square-root SLR

The IPLS iteratively performs SLR to linearize the non-linear motion and measurement models. The non-linear behavior in these functions is modeled as Gaussian noise with zero mean and covariance matrices $\Lambda_k$ and $\Omega_k$ for the motion and measurement models respectively. It would be desireable to compute the square-root of these covariance matrices since the square-root forms of the prediction and update steps require $\Lambda_k^{1/2}$ and $\Omega_k^{1/2}$ as input. Based on the SLR equations presented by [4] it is possible to create a square-root SLR method, which is one of the contributions of this thesis.

The SLR method makes use of sigma points $\mathcal{X}_i$ with weights $\omega_i$ (see Section 2.4.1) to linearize a given function $f(\cdot)$. The transformed sigma points can be computed by passing the original sigma points through the function $f(\cdot)$ as

$$\mathcal{Z}_i = f(\mathcal{X}_i). \tag{2.55}$$

The instrumental matrix $A_{slr}$ will now be formed by stacking the sigma points as

$$A_{slr} = \begin{bmatrix} \sqrt{\omega_1}(\mathcal{X}_1 - \hat{\mathbf{x}}) & \dots & \sqrt{\omega_m}(\mathcal{X}_m - \hat{\mathbf{x}}) \\ \sqrt{\omega_1}(\mathcal{Z}_1 - \hat{\mathbf{z}}) & \dots & \sqrt{\omega_m}(\mathcal{Z}_m - \hat{\mathbf{z}}) \end{bmatrix} \tag{2.56}$$

where each individual sigma point $\mathcal{X}_i$ and $\mathcal{Z}_i$ are column vectors. The vector $\hat{\mathbf{x}}$ is the mean of the distribution over $\mathbf{x}$ and $\hat{\mathbf{z}}$ is given by Equation (2.25a). Multiplying $A_{slr}$ with its transpose will then lead to

$$A_{slr}A_{slr}^\top = \begin{bmatrix} P & \Psi \\ \Psi^\top & \Phi \end{bmatrix}. \tag{2.57}$$

The QR-decomposition $A_{slr}^\top = \Theta_{slr} R_{\ell slr}^\top$ yields a lower-triangular matrix

$$R_{\ell slr} = \begin{bmatrix} X_{slr} & 0 \\ Y_{slr} & Z_{slr} \end{bmatrix} \tag{2.58}$$

that satisfies $A_{slr}A_{slr}^\top = R_{\ell slr}R_{\ell slr}^\top$ and this equality can be explicitly written as

$$\begin{bmatrix} P & \Psi \\ \Psi^\top & \Phi \end{bmatrix} = \begin{bmatrix} X_{slr}X_{slr}^\top & X_{slr}Y_{slr}^\top \\ Y_{slr}X_{slr}^\top & Y_{slr}Y_{slr}^\top + Z_{slr}Z_{slr}^\top \end{bmatrix}. \tag{2.59}$$

The constituents of the linearization are the matrix $F^+$, the bias vector $\mathbf{a}^+$, and the covariance matrix $\Lambda^+$. $F^+$ can be computed by

$$Y_{slr}X_{slr}^{-1} = Y_{slr}X_{slr}^{\top}(X_{slr}^{\top})^{-1}X_{slr}^{-1} = Y_{slr}X_{slr}^{\top}(X_{slr}X_{slr}^{\top})^{-1} = \Psi^{\top}P^{-1} = F^+ \tag{2.60}$$

where the last equality can be found in Equation (2.24a). It is now possible to identify $Z_{slr}$ as the square-root of $\Lambda^+$, which is denoted as $(\Lambda^+)^{1/2}$. Their relation is given by

$$\begin{aligned} Z_{slr}Z_{slr}^{\top} &= \Phi - Y_{slr}Y_{slr}^{\top} \\ &= \Phi - Y_{slr}X_{slr}^{\top}(X_{slr}X_{slr}^{\top})^{-1}X_{slr}Y_{slr}^{\top} \\ &= \Phi - \Psi^{\top}P^{-1}\Psi \\ &= \Phi - F^+P(F^+)^{\top} \\ &= \Lambda^+ \end{aligned} \tag{2.61}$$

where the last equality also can be found in Equation (2.24c).

**Algorithm 5** Square-root algorithm of statistical linear regression.

**Data:** Function $f(\cdot)$ and the first two moments $\hat{\mathbf{x}}$ and $P$ of a pdf $p(\cdot)$.
**Result:** SLR parameters $F^+$, $\mathbf{a}^+$ and $(\Lambda^+)^{1/2}$.
Compute $\mathcal{X}_1, \ldots, \mathcal{X}_m$ and corresponding $\omega_1, \ldots, \omega_m$ using $\hat{\mathbf{x}}$ and $P$
**forall** $\mathcal{X}_i$ **do**
  $\mathcal{Z}_i = f(\mathcal{X}_i)$; `/* Map sigma points through the non-affine function */`
**end**
- Compute $\hat{\mathbf{z}} = \sum_{i=1}^{m} \omega_i \mathcal{Z}_i$.
- Form the matrix $A_{slr}$ according to Equation (2.56).
- Perform the QR-decomposition $A_{slr}^{\top} = \Theta_{slr}R_{\ell slr}^{\top}$ to obtain $R_{\ell slr}$.
- From $R_{\ell slr}$ as described in Equation (2.58) compute
  - $F^+ = Y_{slr}X_{slr}^{-1}$
  - $\mathbf{a}^+ = \hat{\mathbf{z}} - F^+\hat{\mathbf{x}}$
  - $(\Lambda^+)^{1/2} = Z_{slr}$.

## 2.6 Motion Models

To be able to leverage on information about the state vector at other time instances it is needed to model how states are likely to evolve in time. In the scope of Bayesian inference this is done using motion models. This section outlines the motion models that were studied in this thesis.

The motion models that have been compared are

- Constant velocity (CV)
- Constant acceleration (CA)

- Coordinated turn (CT)
- Bicycle model (BM).

The motion models CV and CA are two common motion models [10; 11], which are linear and thereby relatively easy to implement in a smoothing algorithm. One issue with these models is that they do not model the yaw angle of the car, which is needed to estimate the vehicle's bounding box. To be able to use CV and CA in this thesis we propose a modification to the CV and CA models by augmenting their respective state vector with the yaw angle and model the change in yaw as a random walk. Henceforth, when referring to CV and CA we refer to these augmented versions. The state-space models of these modified CV and CA models are given by

$$\text{CV}: \quad \dot{\mathbf{x}}(t) = \tilde{F}\mathbf{x}(t) + \Gamma\tilde{\mathbf{q}}(t) = \begin{bmatrix} \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} & \mathbf{0}_{3\times1} \\ \mathbf{0}_{4\times3} & \mathbf{0}_{4\times3} & \mathbf{0}_{4\times1} \end{bmatrix} \begin{bmatrix} \mathbf{p}(t) \\ \mathbf{v}(t) \\ \phi(t) \end{bmatrix} + \begin{bmatrix} \mathbf{0}_{3\times4} \\ \mathbf{I}_{4\times4} \end{bmatrix} \tilde{\mathbf{q}}(t) \tag{2.62}$$

and

$$\text{CA}: \quad \dot{\mathbf{x}}(t) = \tilde{F}\mathbf{x}(t) + \Gamma\tilde{\mathbf{q}}(t) = \begin{bmatrix} \mathbf{0}_{6\times3} & \mathbf{I}_{6\times6} & \mathbf{0}_{6\times1} \\ \mathbf{0}_{4\times3} & \mathbf{0}_{4\times6} & \mathbf{0}_{4\times1} \end{bmatrix} \begin{bmatrix} \mathbf{p}(t) \\ \mathbf{v}(t) \\ \mathbf{a}(t) \\ \phi(t) \end{bmatrix} + \begin{bmatrix} \mathbf{0}_{6\times4} \\ \mathbf{I}_{4\times4} \end{bmatrix} \tilde{\mathbf{q}}(t) \tag{2.63}$$

where $\mathbf{p} = \begin{bmatrix} x & y & z \end{bmatrix}^\top$ and $\mathbf{a} = \dot{\mathbf{v}} = \ddot{\mathbf{p}}$. The noise vectors $\tilde{\mathbf{q}}$ are assumed to be uncorrelated Gaussians with zero mean.

The CT model is sometimes referred to as *constant turn rate* and describes motion where the acceleration vector is perpendicular to the velocity vector. As indicated by the alternative name *constant turn rate* the angular velocity is constant $\dot{\omega} = 0$. The continuous state dynamics equations can be found in [11] and they are

$$\begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \dot{v}(t) \\ \dot{\phi}(t) \\ \dot{\omega}(t) \end{bmatrix} = \begin{bmatrix} v(t)\cos\phi(t) \\ v(t)\sin\phi(t) \\ 0 \\ \omega(t) \\ 0 \end{bmatrix} + \Gamma\tilde{\mathbf{q}}(t) \tag{2.64}$$

where $\Gamma = \mathbf{I}$ since we have chosen to apply noise in all dimensions. The process noise is distributed as, $\tilde{\mathbf{q}}(t) \sim \mathcal{N}(\mathbf{0}, \tilde{Q})$, with a diagonal covariance matrix $\tilde{Q}$.

The bicycle model is inspired by the kinematics of a bicycle and therefore has a steering angle $\delta_f$ in the state vector [12; 13; 14]. Figure 2.3 illustrates the parameters

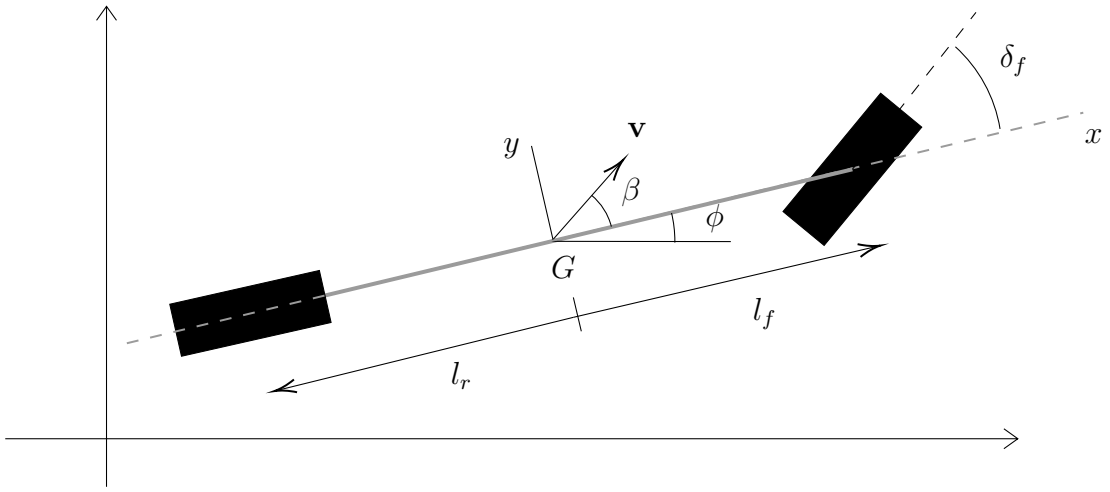



**Figure 2.3:** The figure describes geometrical interpretation of the notation used in the bicycle model (2.65).

involved and the continuous state transition equations for this motion model are

$$\begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \dot{v}(t) \\ \dot{\phi}(t) \\ \dot{\delta}_f(t) \end{bmatrix} = \begin{bmatrix} v(t)\cos(\phi(t) + \beta(\delta_f)) \\ v(t)\sin(\phi(t) + \beta(\delta_f)) \\ 0 \\ \frac{v(t)}{l_r}\sin(\beta(\delta_f(t))) \\ 0 \end{bmatrix} + \Gamma\tilde{\mathbf{q}}(t) \tag{2.65a}$$

$$\beta(\delta_f(t)) = \arctan\left(\tan(\delta_f(t))\frac{l_r}{l_f + l_r}\right) \tag{2.65b}$$

where $\dot{x}$ and $\dot{y}$ are time derivatives of the position, $\dot{v}$ is the time derivative of speed and $\dot{\phi}$ is the time derivative of yaw. $\beta(\delta_f)$ is a constraint that connect the steering angle $\delta_f$ to the yaw of the car. The distances from the origin of the velocity vector to the front and rear wheel centers are denoted by $l_f$ and $l_r$ respectively. As for CT, we have chosen to apply noise in all dimensions which leads to $\Gamma = \mathbf{I}$ and it is assumed that $\tilde{\mathbf{q}}(t) \sim \mathcal{N}(\mathbf{0}, \tilde{Q})$, with a diagonal covariance matrix $\tilde{Q}$.

For an estimated annotation to be complete, the size of the bounding box should be included. By simply augmenting the different motion models' state vectors and transition matrices with the bounding box dimensions a complete estimate of the annotation is obtained at each time instance. Under the assumption that the dimensions of the car does not change in time the motion model is given by

$$\begin{bmatrix} \mathbf{x}_{mm} \\ w \\ l \\ h \end{bmatrix}_{k+1} = \begin{bmatrix} f_{mm}(\mathbf{x}_{mm}) \\ w \\ l \\ h \end{bmatrix}_k + \begin{bmatrix} \mathbf{q} \\ 0 \\ 0 \\ 0 \end{bmatrix}_k . \tag{2.66}$$

The subscript $mm$ refers to the chosen motion model (CV, CA, CT, or BM) and $w$, $l$, and $h$ is the width, length, and height of the bounding box.

### 2.6.1 Discretization of Motion Models

The time continuous motion model,

$$\dot{\mathbf{x}}(t) = \tilde{f}(\mathbf{x}(t)) + \tilde{\mathbf{q}}(t), \tag{2.67}$$

needs to be discretized since our smoothing technique requires time discrete motion models. The discretized motion models is given on the form

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k) + \mathbf{q}_k \tag{2.68}$$

and these two terms need to be identified. The first term, $f(\mathbf{x}_k)$, represents the discretization of the expected value and the second term, $\mathbf{q}_k$, represents the linearization of the covariance. If the motion model is linear this can be done analytically, but for the non-linear motion models approximations are needed in order to obtain discrete counter parts.

#### Linear Models

In continuous time, linear models are stated as

$$\dot{\mathbf{x}}(t) = \tilde{F}\mathbf{x}(t) + \tilde{\mathbf{q}}(t), \quad \tilde{\mathbf{q}}(t) \sim \mathcal{N}(0, \tilde{Q}) \tag{2.69}$$

and the time discrete counter part is given as

$$\mathbf{x}_{k+1} = F\mathbf{x}_k + \mathbf{q}_k, \quad \mathbf{q}_k \sim \mathcal{N}(0, Q_k) \tag{2.70}$$

where $k$ is the discrete time index. To obtain the discrete motion model the differential equation in Equation (2.69) can be solved using an integrating factor and integrate from $t$ to $t + \Delta t$ where $\Delta t$ is the time step between the resulting discrete states. The discrete step is given by

$$\mathbf{x}(t + \Delta t) = \exp(\tilde{F}\Delta t)\mathbf{x}(t) + \int_t^{t+\Delta t} \exp(\tilde{F}(t + \Delta t - \tau))\tilde{\mathbf{q}}(\tau)\mathrm{d}\tau. \tag{2.71}$$

The discrete transition matrix $F_k$ and discrete noise vector $\mathbf{q}_k$ can be identified as

$$F_k = \exp(\tilde{F}\Delta t) \tag{2.72}$$

and

$$\mathbf{q}_k = \int_t^{t+\Delta t} \exp(\tilde{F}(t + \Delta t - \tau))\tilde{\mathbf{q}}(\tau)\mathrm{d}\tau. \tag{2.73}$$

This results in $\mathbf{q}_k \sim \mathcal{N}(0, Q_k)$ where the covariance matrix $Q_k$ can be computed as

$$Q_k = \mathrm{Cov}(\mathbf{q}_k) = \int_0^{\Delta t} \exp(\tilde{F}\tau)\tilde{Q}\exp\left(\tilde{F}^\top\tau\right)d\tau. \tag{2.74}$$

For details on the derivation see [6, Ch.12, pp.315].

Using the formula provided by Equation (2.74) for CA, the discretized process noise covariance matrix is given by

$$Q_{k-1}^{\text{CA}} = \begin{pmatrix} \tilde{Q}_a \frac{\Delta t_{k-1}^5}{20} & \tilde{Q}_a \frac{\Delta t_{k-1}^4}{8} & \tilde{Q}_a \frac{\Delta t_{k-1}^3}{6} & 0 \\ \tilde{Q}_a \frac{\Delta t_{k-1}^4}{8} & \tilde{Q}_a \frac{\Delta t_{k-1}^3}{3} & \tilde{Q}_a \frac{\Delta t_{k-1}^2}{2} & \vdots \\ \tilde{Q}_a \frac{\Delta t_{k-1}^3}{6} & \tilde{Q}_a \frac{\Delta t_{k-1}^2}{2} & \tilde{Q}_a \Delta t_{k-1} & 0 \\ 0 & \cdots & 0 & \tilde{Q}_{\phi,s} \Delta t_{k-1} \end{pmatrix} \tag{2.75}$$

where $\tilde{Q}_a$ is the continuous process noise for the acceleration states with only diagonal elements and $\tilde{Q}_{\phi,s}$ is the diagonal matrix with the continuous process noise parameters for the angle and size variables. Under the assumption that the car's dimensions does not change in time, the noise parameters corresponding to size should be zero. For CV the discretized process noise covariance matrix is

$$Q_{k-1}^{\text{CV}} = \begin{pmatrix} \tilde{Q}_v \frac{\Delta t_{k-1}^3}{3} & \tilde{Q}_v \frac{\Delta t_{k-1}^2}{2} & 0 \\ \tilde{Q}_v \frac{^2_{k-1}}{2} & \tilde{Q}_{vk-1} & 0 \\ 0 & 0 & \tilde{Q}_{\phi,sk-1} \end{pmatrix} \tag{2.76}$$

where $\tilde{Q}_v$ is the continuous process noise for the velocity states with only diagonal elements.

### Non-linear Models

For the non-linear motion models considered in this thesis, the time continuous differential equations can not be solved analytically and the approximate solution is obtained by employing the modified Euler method. The modified Euler method approximates the derivative $\dot{\mathbf{x}}$ as

$$\dot{\mathbf{x}}(\tau) \approx \tilde{f}(\mathbf{x}(t)) + \tilde{\mathbf{q}}(\tau), \quad \forall \tau \in [t, t + \Delta t]. \tag{2.77}$$

To discretize this relationship, the expression is integrated over the discretization interval $[t, t + \Delta t]$ which yields

$$\mathbf{x}(t + \Delta t) \approx \mathbf{x}(t) + \underbrace{\int_t^{t+\Delta t} \tilde{f}(\mathbf{x}(t)) \mathrm{d}\tau}_{= \Delta t \tilde{f}(\mathbf{x}(t))} + \int_t^{t+\Delta t} \tilde{\mathbf{q}}(\tau) \mathrm{d}\tau. \tag{2.78}$$

From the above equation the time discrete state mapping can be identified as

$$f(\mathbf{x}) = \mathbf{x} + \tilde{f}(\mathbf{x}(t)) \Delta t \tag{2.79}$$

and the discrete noise vector is the remaining term

$$\mathbf{q} = \int_t^{t+\Delta t} \tilde{\mathbf{q}}(\tau) \mathrm{d}\tau. \tag{2.80}$$

The continious noise $\tilde{\mathbf{q}}$ is assumed to be white noise with zero mean which results in that the discrete covariance $Q_k$ is

$$Q_k = \text{Cov}(\mathbf{q}) = \tilde{Q}\Delta t. \tag{2.81}$$

Deatails of the dervation can be found in [6, Ch. 12, pp. 316]. Using Equation (2.81) the discretized process noise covariance matrices are simply

$$Q_{k-1}^{\text{CT}} = \Delta t_{k-1}\tilde{Q}^{\text{CT}} \tag{2.82}$$

and

$$Q_{k-1}^{\text{BM}} = \Delta t_{k-1}\tilde{Q}^{\text{BM}} \tag{2.83}$$

for CT and BM respectively.

#### Increasing Accuracy in Simulation

To solve the differential equations posed by the motion models, in equations (2.62), (2.63), (2.64), and (2.65), the modified Euler method was used. This approximation introduces integration errors since it assumes that the derivative is constant over the discretization interval $\Delta t$. To mitigate this issue the *fast sampling* method was used [6, pp. 315]. The idea is quite straight forward and the method is just to divide each discretization interval into sub-intervals and thereby taking shorter discretization steps. By taking shorter steps the derivative is held constant over shorter time steps which reduces the integration error.

### 2.6.2 Measurement Noise

When performing the update step, information from the prediction is fused with information from the measurement. This can be thought of as a weighted average between the predicted state and the measured state where the weights are defined by the covariances. Hence, the choice of measurement noise is important and defines how much the filter should rely on the measured values. This thesis has two different methods of assigning measurement noise which will be presented below.

#### Stationary Noise

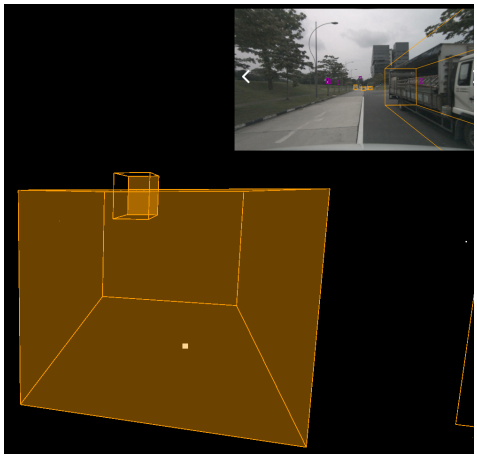
The purpose of annotations is that they should provide the ground truth from which supervised machine learning algorithms can learn. Given the assumption that annotations are ground truth data it follows that the measurement noise should be very small and stationary. This means small values in the matrix $R_k$ that do not between different time instances $k$.

### Distance Dependent Noise

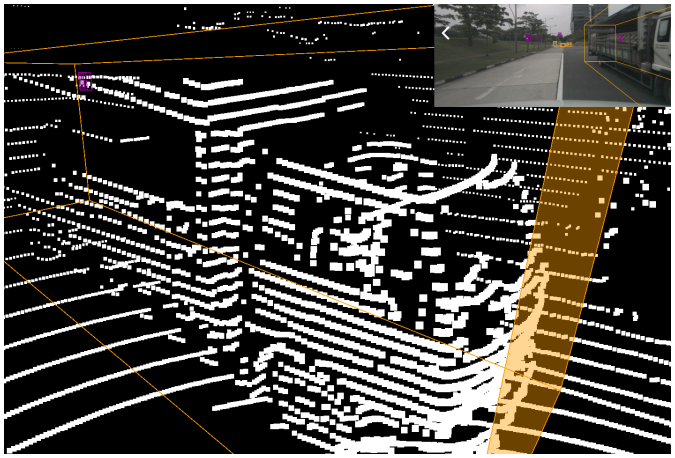
The assumption that all annotations are created under equal conditions can be questioned since there are a lot of factors that affect how well annotations can be made, e.g. distance to object, degree of occlusion, weather, and lighting conditions. Figure 2.4 shows an example of how the difficulty to make a precise annotation increases with the distance to the object. This suggests that an annotation of a far away object will have a higher uncertainty in the variables that constitutes the annotation. In a Bayesian setting, this means that an annotation at a distance should have larger elements in the measurement covariance matrix $R_k$ than an annotation at close distance.

In this thesis an ad hoc solution to incorporate a distance dependent measurement noise was adopted. To make the noise depend on distance, a mapping like the one in Figure 2.5 was used. Since different states represent different physical quantities, the numerical breakpoints can be set differently for different states.

**(a)** An overview of the annotated scene. The vehicle in Figure 2.4b is the middle box of the three orange boxes in the distance and the vehicle in Figure 2.4c is the truck to the right in this figure. Note that at the time of annotation, the annotator have access to adjacent cameras where the front of the truck is visible.



**(b)** A point cloud of a vehicle at a relatively large distance to the LiDAR sensor. Note that only one point is provided by the point cloud to infer the vehicle's size, orientation, and poistion.



**(c)** A point cloud of a vehicle at a short distance to the LiDAR sensor. When more measurements (white points) of the same vehicle exists it is easier to make a precise annotation.

**Figure 2.4:** The difficulty of making annotations increase with the distance to the object. For the case in Figure 2.4b very little information is provided to the annotator to create the annotation since the car covers few pixels and only have one LiDAR point.

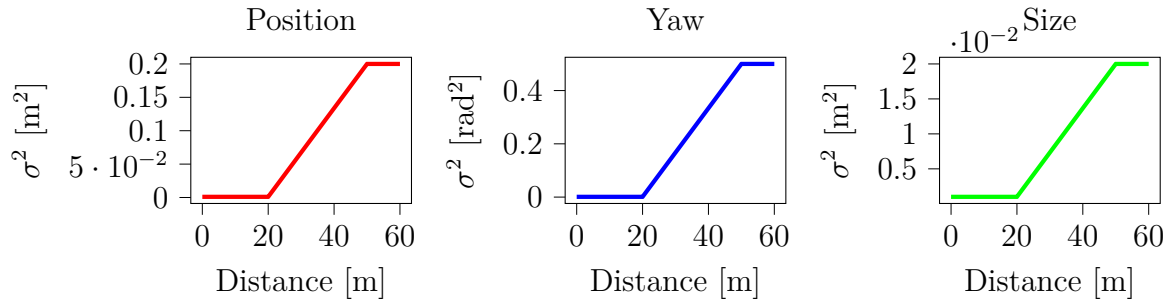



**Figure 2.5:** These three functions were used to map the distance between the sensor and object to the corresponding measurement noise parameters. Note that the different functions do not share their maximum $y$-value.

# 3

# Methods

As mentioned, this thesis aims to find a good motion model that allows for the highest level of automation. Since a strong temporal correlation exists between adjacent time instances the problem is to consolidate information about the hidden state $\mathbf{x}_k$ using both past, present, and future information. This thesis focuses on the available tools in Bayesian inference and methods used in this thesis will be described in this chapter.

## 3.1 Smoothing Implementations

This thesis solves the trajectoiry estimation as a smoothing problem and the smoother that we have choosen for this thesis is the IPLS. In the IPLS described in Section 2.4 both the motion model and the measurement models are linearized using SLR. However, in this thesis the measured states are part of the state vectors for all motion models. This means that the mapping between state and measurement is always linear. When using SLR to find an affine approximation ($h_k(\mathbf{x}) \approx H_k\mathbf{x} + \mathbf{b}_k + \mathbf{g}_k$, $\mathbf{g}_k \sim \mathcal{N}(0, \Omega_k)$) of the linear $h_k(\cdot)$, the resulting $\mathbf{b}_k$ and $\Omega_k$ will be zero since a linear function is also affine and hence the approximation is exact. In the presentation of the IPLS below the measurement model $h_k(\cdot)$ has been replaced with $H$ that represents the linear mapping between state and measurement, $\mathbf{y}_k = H\mathbf{x}_k$.

García-Fernández et al. states in [4] that initialization of the first posterior solution can be made using a sigma point smoother. As mentioned, no linearization of the measurement model is needed and therefore the initialization can be computed using SLR with the only difference that during initialization only the posterior from the filter solution is available in contrast to after initialization where the linearization can be obtained based on the smoothing solution. The pseudocode of the IPLS algorithm used in this thesis can be seen in Algorithm 6.

**Algorithm 6** The IPLS used in this thesis.

**Data:** Prior mean:$u_0^1 = \mathbf{x}_0$, prior covariance:$W_0^1 = P_0$, motion model: $f_k(\cdot)$, $Q_k$, measurement model:$H$, $R_k$, measurement sequence: $\mathbf{y}$
**Result:** Posterior moments: $s_k^J$, $W_k^J$ for $k = \{0, \ldots, T\}$
`/* Below p , u , s are the predicted, updated, and smoothing solutions respectively and P, U, S are their corresponding covariance matrices. The subscript refers to which time the estimate refers to. If there is no subscript it refers to the set containing the variable at all time instances. */`
`/* Initialize` $s_k^1$, $S_k^1$ `for` $k = \{1, \ldots, T\}$ `*/`
**for** *k = 0 to T-1* **do**
  $(F_k^1, a_k^1, \Lambda_k^1)$ = `SLR` $(f_k(\cdot), u_k^1, U_k^1)$
  $(p_{k+1}^1, P_{k+1}^1)$ = `prediction` $(F_k^1, a_k^1, \Lambda_k^1, u_k^1, U_k^1, Q_k)$
  **if** *meas exists at time k+1* **then**
    $(u_{k+1}^1, U_{k+1}^1)$ = `update` $(p_{k+1}^1, P_{k+1}^1, H, R_k, \mathbf{y}_{k+1})$
  **else** pass prediction as posterior
    $(u_{k+1}^1, U_{k+1}^1) = (p_{k+1}^1, P_{k+1}^1)$
  **end**
**end**
`/* obtain smoothing solution from affine RTS */`
$(s^1, S^1)$ = `RTS` $(u^1, U^1, p^l, P^l, F^1, a^1, \Lambda^1)$
**for** *i=1 to J-1* **do**
  **for** *k = 0 to T* **do**
    $(F_k^{i+1}, a_k^{i+1}, \Lambda_k^{i+1})$ = `SLR` $(f_k(\cdot), s_k^i, S_k^i)$
  **end**
  **for** *k = 1 to T* **do**
    $(p_{k+1}^{i+1}, P_{k+1}^{i+1})$ = `prediction` $(F_k^{i+1}, a_k^{i+1}, \Lambda_k^{i+1}, u_k^i, u_k^i, Q_k)$
    **if** *meas exists at time k+1* **then**
      $(u_{k+1}^{i+1}, U_{k+1}^{i+1})$ = `update` $(p_{k+1}^{i+1}, P_{k+1}^{i+1}, H, R_{k+1}, \mathbf{y}_{k+1})$
    **else** pass prediction as posterior
      $(u_{k+1}^{i+1}, U_{k+1}^{i+1}) = (p_{k+1}^{i+1}, P_{k+1}^{i+1})$
    **end**
  **end**
  $(s^{i+1}, S^{i+1})$ = `RTS` $(u^{i+1}, U^{i+1}, p^{i+1}, P^{i+1}, F^{i+1}, a^{i+1}, \Lambda^{i+1})$
  **if** `KLD` *($s^{i+1}$, $S^{i+1}$, $s^i$, $S^i$) < threshold)* **then** If two following solutions are close the iteration should terminate
    break
  **end**
**end**

### 3.1.1 Termination

As mentioned in Section 2.2, we would like to compute the joint posterior density $p(\mathbf{x}_{0:T}|\mathbf{y}_{1:T})$ to estimate the trajectories. In this thesis the approximation, $q^{\Theta^i}$, to

the joint posterior is iteratively obtained using IPLS as $p(\mathbf{x}_{0:T}|\mathbf{y}_{1:T}) \approx q^{\Theta^i}(\mathbf{x}_{0:T}|\mathbf{y}_{1:T})$ where

$$\Theta^i = (F^i_{0:T-1}, \mathbf{a}^i_{0:T-1}, \Lambda^i_{0:T-1}) \tag{3.1}$$

is the set of enabling approximations acquired from SLR at iteration $i$.

Ideally, the selected $\Theta^i$ would make $q^{\Theta^i}(\mathbf{x}_{0:T}|\mathbf{y}_{1:T})$ match the first two moments of $p(\mathbf{x}_{0:T}|\mathbf{y}_{1:T})$. To measure how well these moments match, the KLD can be used [8]. García-Fernández et al. states in [4] that IPLS recursion iteratively improves the set of enabling approximations $\Theta^i$. We have therefore chosen to check if

$$D(q^{\Theta^i}(\mathbf{x}_{0:T}|\mathbf{y}_{1:T})||q^{\Theta^{i+1}}(\mathbf{x}_{0:T}|\mathbf{y}_{1:T})) \tag{3.2}$$

is below a chosen threshold value as a termination condition of the iterations in the IPLS.

### 3.1.2 Shift of Observed Angles

When performing the update step an assumption of continuous state spaces is made. Since the angle state is modulo $2\pi$, different numerical values have the same physical meaning. This leads to issues when performing the update step since values that represent close angles could have numerical values that are very different.

For example, assume that the observed angles are given on the interval $[-\pi, \pi)$ and the predicted angle is $2\pi$. If an observation of 0 is obtained, the predicted angle should not be updated since $2\pi$ and 0 represent the same angle. Since 0 is lower than $2\pi$ the updated value will be something between 0 and $2\pi$ which is undesired.

To solve this issue the observed value can be shifted to an interval centered in the predicted value. Then both the angle representations will be in the same interval and all angles will be represented by the same numerical value.

## 3.2 Motion Model Evaluation

The purpose of the motion models is to drive the state variables forward in time (see Figure 2.1). To evaluate which model that performs best, state estimations were made for at least all annotated time instances (but possibly more often, see Section 3.2.1). The annotations at time instances that were not used for estimation were utilized for evaluation of the motion model. To determine how well different motion models can predict the state trajectory, the distance between human annotations provided to the algorithm was varied using two different heuristics described in Section 3.3. To quantitatively measure how well specific motion models perform, the metrics described in Section 3.2.2 was used.

### 3.2.1 Fast Sampling

As discussed in Section 2.6.1 the integration error increases with the time step taken but it is not the only error that grows with larger step sizes. When making the affine approximation using SLR, the second moment of the pdf will grow with a larger time step since the process noise grows with larger time steps. This will effectively force the SLR to adapt the approximation to a larger interval in the state space of $\mathbf{x}_k$ since the integrations in Equations (2.25) are weighted with the pdf of $\mathbf{x}_k$.

To reduce both the integration error and the "linearization" error, the method of fast sampling was implemented. This was done by increasing the frequency at which the prediction steps were made. This has the effect that the time steps $\Delta t_k$ used in prediction get smaller. For CV and CA a prediction frequency of 4.0 Hz was used and for CT and BM 10.0 Hz was used.

### 3.2.2 Metrics for Evaluation

In order to yield quantitative results on agreement between boxes from Nuscenes and estimated boxes, well defined metrics are needed. A commonly used metric to quantify similarity of two bounding boxes is IoU which measures overlap between two enclosed spaces. Since we assume the world to be flat and the box height to be constant, the IoU can be evaluated in the $xy$-plane. The definition follows as

$$\mathrm{IoU}(\hat{A}_{\mathrm{box}}, A_{\mathrm{box}}) = \frac{|\hat{A}_{\mathrm{box}} \cap A_{\mathrm{box}}|}{|\hat{A}_{\mathrm{box}} \cup A_{\mathrm{box}}|} \tag{3.3}$$

where $\hat{A}_{\mathrm{box}}$ represents the space enclosed by the estimated bounding box and $A_{\mathrm{box}}$ represents the space enclosed by the ground truth box. This metric is used by [15] to determine how well the proposed bounding boxes agrees with the ground truth. To measure an average IoU agreement the following metric can be used

$$\overline{\mathrm{IoU}} = \frac{1}{m} \sum_k \mathrm{IoU}(\hat{A}_{\mathrm{box}_k}, A_{\mathrm{box}_k}) \tag{3.4}$$

where the summation includes all estimated bounding boxes used for evaluation and the total number of those boxes is $m$. This gives a value ranging between 0 and 1 where it is desirable to achieve a result as close to 1 as possible.

IoU is invariant to a 180° flip in yaw angle another metric is needed to measure this deviation. One metric that can measure how well the yaw angle aligns with the ground truth yaw is the average heading similarity (AHS) given by

$$\mathrm{AHS} = \frac{1}{m} \sum_k \frac{1 + \cos\theta_k}{2} \tag{3.5}$$

where $\theta_k = \hat{\phi}_k - \phi_k$, which is the difference between the estimated yaw and the yaw of the ground truth box. The summation $\frac{1}{m}\sum_k$ is the same as in (3.4) and a value close to 1 is desired.

Another metric that could be used is a root-mean-square error (RMSE) on the distance between the center points of the estimated bounding boxes and the ground truth boxes. If $d$ is the Euclidean distance between the centers, then the RMSE can be defined as

$$d_{RMSE} = \sqrt{\frac{1}{m}\sum_{k} d_k^2}. \tag{3.6}$$

In this project the world is assumed to be flat and therefore $d_k^2 = (\hat{x}_k - x_k)^2 + (\hat{y}_k - y_k)^2$ which is the center deviation in the $xy$-plane.

### 3.2.3 Selected Sequences from Nuscenes

To evaluate the motion models, ground truth data is needed and for this reason we need to select some suitable annotation sequences from the Nuscenes dataset. Since we have chosen estimate car trajectories only, sequences of the category 'vehicle.car' has been used for estimation. Furthermore, there are attributes associated with each annotation and for vehicles the attributes are:

- vehicle.moving
- vehicle.stopped
- vehicle.parked

We have chosen to filter out the vehicle.car sequences that have at least one annotation with the attribute vehicle.moving. There are 422 vehicle.car sequences that satisfy this condition and the distribution of the number of annotations in those sequences is presented in Figure 3.1. This histogram shows that there is a concentration towards sequences with fewer annotation compared to those with many. We have chosen to only consider moving cars because the annotations of stationary vehicles could just be held at a constant pose in time. This is of little interest for trajectory estimation.

## 3.3 Annotation Strategies

One of the research questions is how the level of human intervention affects the quality of the automatically generated annotations. To answer that question we have chosen two annotation strategies that simulate how a human annotator interacts with the smoother and thereby add a sparse set of annotation in two different ways. We have chosen to call these strategies the *static annotation strategy* and the *adaptive annotation strategy.*

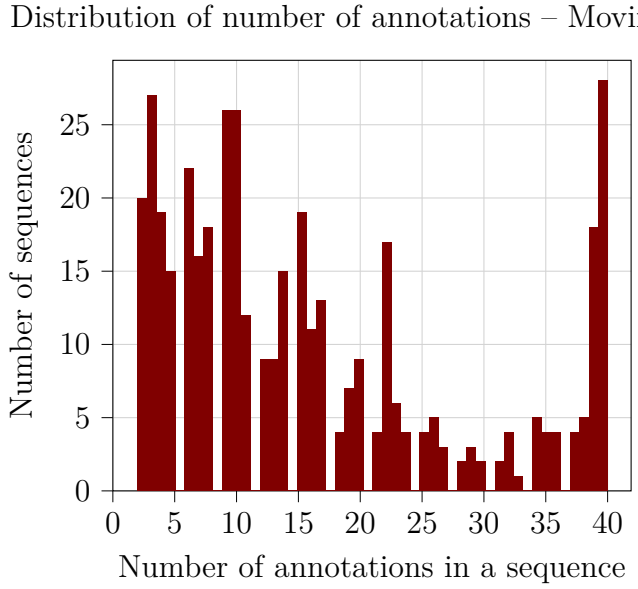



**Figure 3.1:** This histogram shows the distribution of how many annotations the 422 chosen sequences contain. The chosen sequences are of the object type vehicle.car and at least one annotation in every sequence has the attribute vehicle.moving.

### 3.3.1 Static Annotation Strategy

The static annotation strategy is a rather simple way of adding annotations as input to the smoother. The strategy is:

1. Select a step size $\ell$.
2. Add the first annotation in the sequence.
3. From the first annotation and onwards, add every $\ell$-th annotation.
4. Add the last annotation if it was not already added in the previous step.

A detailed description is provided in Algorithm 7.

**Algorithm 7** Static annotation strategy

**Data:** Sequence of annotations $\mathbf{y}_{1:T} = (\mathbf{y}_1, \ldots, \mathbf{y}_T)$ and step size $\ell$.
**Result:** Subset annotation sequence $\mathbf{Y}_s \subseteq \mathbf{y}_{1:T}$.
Initialize $\mathbf{Y}_s = \emptyset$.
Add the first, $\mathbf{y}_1$, and the last, $\mathbf{y}_T$, to $\mathbf{Y}_s$.
**forall** $i \in \{2, \ldots, T-1\}$ **do**
  **if** $i - 1$ *is a multiple of* $\ell$ **then**
    Add $\mathbf{y}_i$ to $\mathbf{Y}_s$
  **else**
    Continue
  **end**
**end**

**Table 3.1:** The number (column two) of annotations used as input to the smoother for the static annotation strategy, and proportion of total annotations as a percentage (column three) that this represents.

| Total number of annotations: 6687 | | |
|---|---|---|
| Step size, $\ell$ | Number of used annotations | Percentage |
| 2 | 3679 | 55 % |
| 4 | 2165 | 32 % |
| 6 | 1669 | 25 % |
| 8 | 1386 | 21 % |
| 10 | 1239 | 19 % |

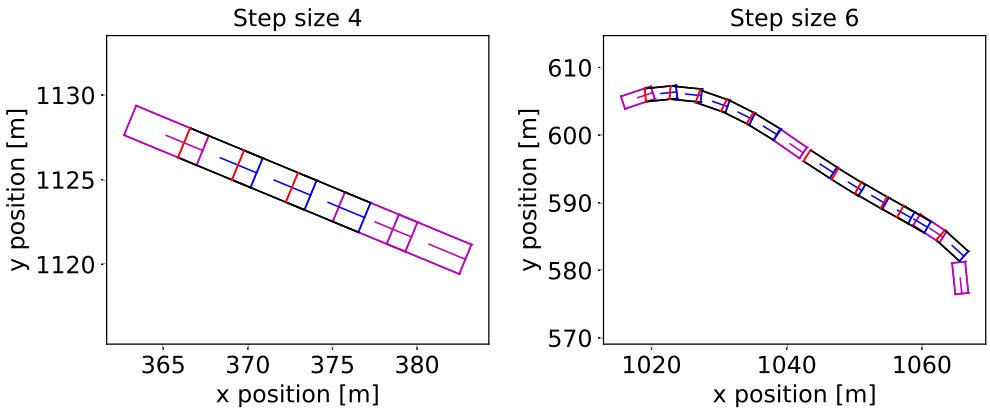



**Figure 3.2:** Examples of the static annotation strategy in two annotation sequences. The purple boxes are the selected boxes that are members of $\mathbf{Y}_s$. In the left sequence step size 4 was used and in the right step size 6 was used.

According to this strategy the number of annotations used as input to the smoother for a particular sequence will be $|\mathbf{Y}_s|$. In the 422 car sequences in Nuscenes that we have selected for analysis there are 6687 annotated boxes. If the static annotation strategy is followed for step sizes $\ell = 2, 4, 6, 8$ and 10 the number of used annotations will be as presented in Table 3.1. As the step sizes increase the difference in the percentage of used annotations gets smaller. This can be explained by the fact that many sequences have fewer than 20 annotations as Figure 3.1 shows. An example of the results of this strategy is illustrated in Figure 3.2 for step sizes 4 and 6.

There are twenty sequences that have only two annotations as shown by the leftmost bar in Figure 3.1. Since the static annotation strategy will add all annotations in these sequences there will not be any annotations left for evaluation. Therefore, the results for the static annotation strategy involves 402 sequences instead of 422.

### 3.3.2 Adaptive Annotation Strategy

The other annotation strategy is the adaptive annotation strategy that simulates a different workflow for the human annotator. The idea is that once the smoothing algorithm has generated box proposals, then the human annotator can correct the worst box proposal. The corrected box is then added to the set of boxes observed by the smoother and then smoother is run another time to generate a new trajectory. This new trajectory will most likely result in better box proposals than in the previous run. The annotator will therefore adapt the insertion of boxes to where the estimated trajectory deviates the most from the actual trajectory of the car, hence the name *adaptive annotation strategy*.

The annotator will keep adding annotations until the annotator thinks that the estimated trajectory represents ground truth. When evaluating the adaptive annotation strategy we have access to the annotations from the dataset that are not used for estimation (boxes marked in a combination of black, red, and blue in Figure 3.3). As a proxy for the human decision to stop adding boxes we have chosen to use a threshold on the minimum IoU value. When all boxes' IoU values are above the chosen threshold the algorithm terminates. The pseudocode for this strategy is outlined in Appendix A.1.

### 3.3.3 Design of Experiments

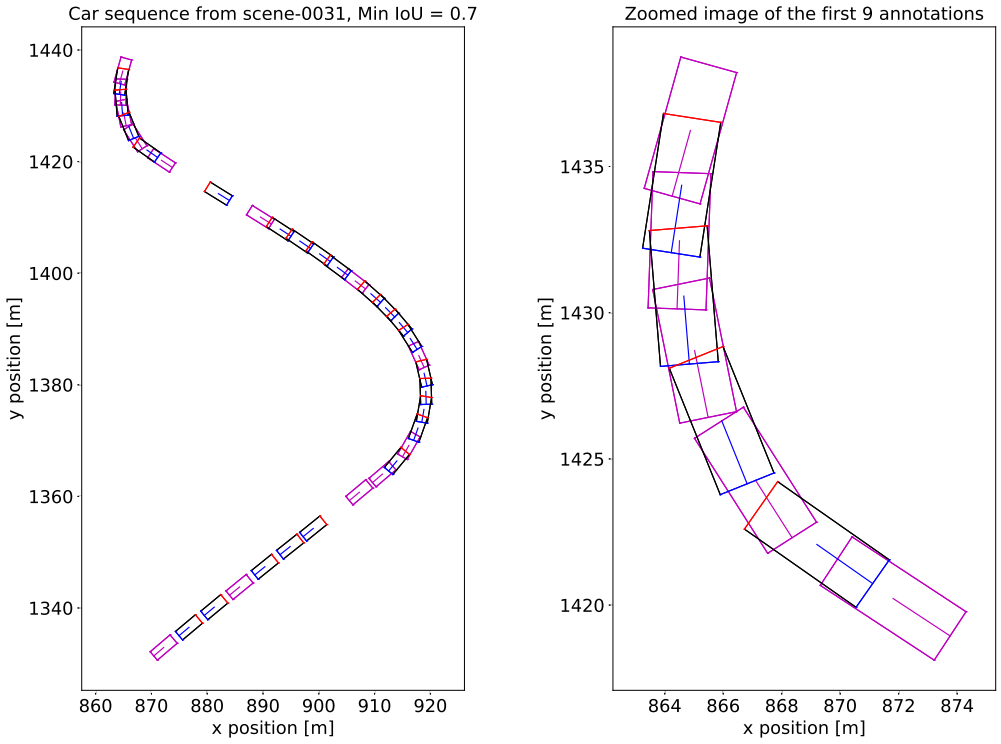
To be able to draw conclusions on which motion model performs best and what the implications of the different annotation strategies are, the experiments were divided into two parts, one for each annotation strategy.

#### Static Annotation Strategy

For each motion model the algorithm was run for step sizes 2, 4, 6, 8 and 10 between the annotated time instances. Larger step sizes than 10 was not included since we deemed that it did not provide more insight to answer our research questions. The results for different step sizes were evaluated using the metrics described in Section 3.2.2.

#### Adaptive Annotation Strategy

For each motion model the algorithm was run for minimum IoU values of 0.6, 0.7, 0.8 and 0.9. The results for different minimum IoU values were then evaluated by computing the fraction between the number of added human annotations and the total number of annotations.




**(a)** The full sequence.

**(b)** The first 9 annotations from the sequence to the left.

**Figure 3.3:** Examples of the adaptive annotation strategy in one annotation sequence from scene-0031. The purple boxes are the selected annotations from the dataset that are observed by the smoother when making trajectory estimates. The other boxes are the annotations from the dataset used to decide when no more boxes is needed for the smoother to estimate a good trajectory. Note that the purple annotations that are added to the observed annotations are not equidistant, suggesting that some annotations hold more information than others.

### Assigning Parameter Values

We also have to assign values to the motion model parameters. For example, BM has parameters $l_f$ and $l_r$ which represent the distance from the center of mass to the front and rear axles respectively [14]. However, we assume that no slip occurs at the wheels and the only point of interest is the geometrical center (rather the the mass center). Hence, the BM parameters should be $l_f = l_r = L/2$ where $L$ is the total length of the vehicle. The $x$ and $y$ position of the annotations is the geometrical center and this choice aligns the $x$ and $y$ position of the motion model with the $x$ and $y$ position of the corresponding annotation.

In an effort to tune the process noise covariance parameters, a coordinate search to find parameters that maximizes the prediction likelihood was used, as suggested by Abbeel et al. in [16].

# 4

# Results

This chapter will present the results from the experiments that show how our different metrics depend on the sparsity of the available annotations. We have chosen not to present results of the metric AHS because this metric was found to provide little information on the quality of the estimated trajectory. The reason why it held little information was that it was always close to its maximum value for all trajectories where estimation was possible, independent of the annotation sparsity.

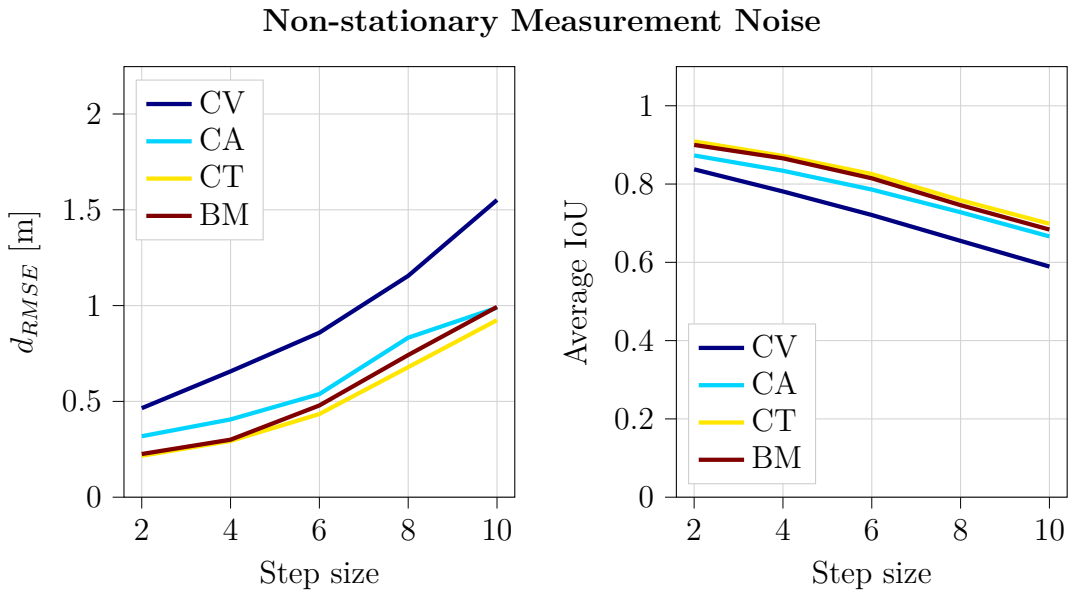
## 4.1 Static Annotation Strategy

The static annotation strategy as described in Section 3.3.1 adds every $\ell$-th annotation to the observed annotations. That fixed number $\ell$ is called the step size and has been assigned the values 2, 4, 6, 8 and 10. For each of these five assignments simulations have been run and the quality of the estimated trajectories are evaluated below.

### 4.1.1 Non-stationary Measurement Noise

When using non-stationary measurement noise the measurement uncertainty increases if the boxes are further away from the ego-vehicle. The results for this noise model is shown in Figure 4.1 that includes RMSE for the center point deviation and the average IoU for the sequences where the smoother successfully produced trajectory estimations. The numbers of successful sequences for each step size and motion model are specified in Table 4.1. The sequences for which the smoother failed are not used to compute the metrics and are therefore counted as "Number sequences not used" in the table.

Since there is a difference between the motion models in terms of how they model the kinematics of the cars, it is interesting to filter out the sequences that turn more than others. To separate the turning sequences from straight sequences we have chosen to classify a sequence as turning if the largest difference between the yaw angles of two ground truth boxes is larger than 0.5 radians ($\approx 29°$). Sequences that




**(a)** $d_{RMSE}$ in the $xy$-plane. **(b)** Average IoU in the $xy$-plane.

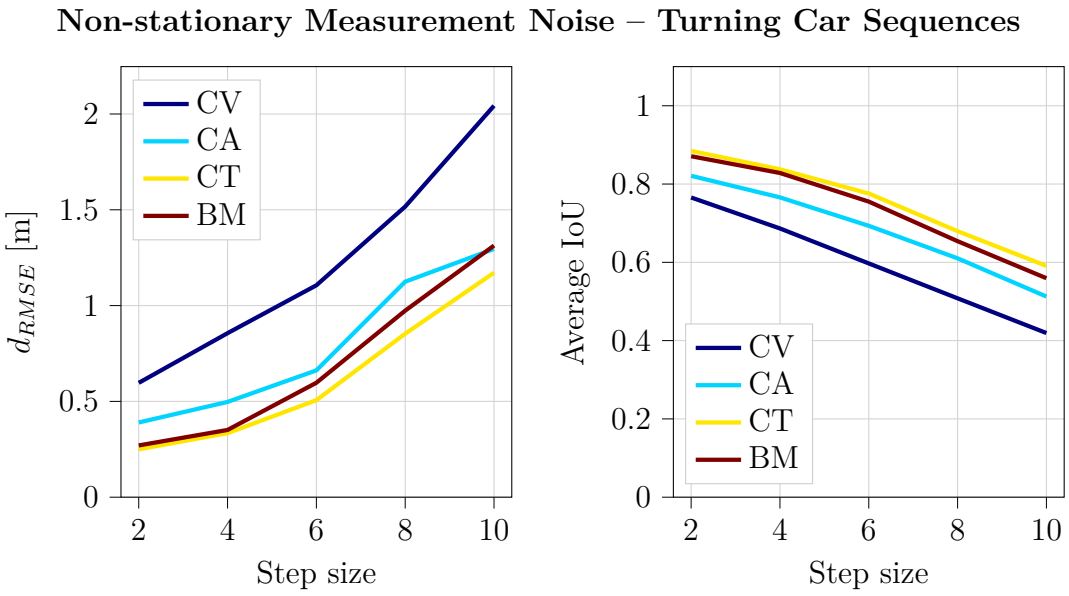
**Figure 4.1:** Results from the static annotation strategy when using non-stationary measurement noise. The sequences that have been used to compute these results are the ones that have successfully estimated trajectories.

**Table 4.1:** The numbers of sequences that were evaluated in the static annotation strategy when using non-stationary measurement noise are specified as "Number of sequences used". "Number of sequences not used" is the number of sequences for which the smoother failed.

| Number of sequences not used / Number of sequences used | | | | | |
|---|---|---|---|---|---|
| Step size | 2 | 4 | 6 | 8 | 10 |
| CV | 0 / 402 | 0 / 402 | 0 / 402 | 0 / 402 | 0 / 402 |
| CA | 0 / 402 | 0 / 402 | 0 / 402 | 0 / 402 | 0 / 402 |
| CT | 1 / 401 | 1 / 401 | 2 / 400 | 2 / 400 | 2 / 400 |
| BM | 1 / 401 | 1 / 401 | 1 / 401 | 1 / 401 | 2 / 400 |

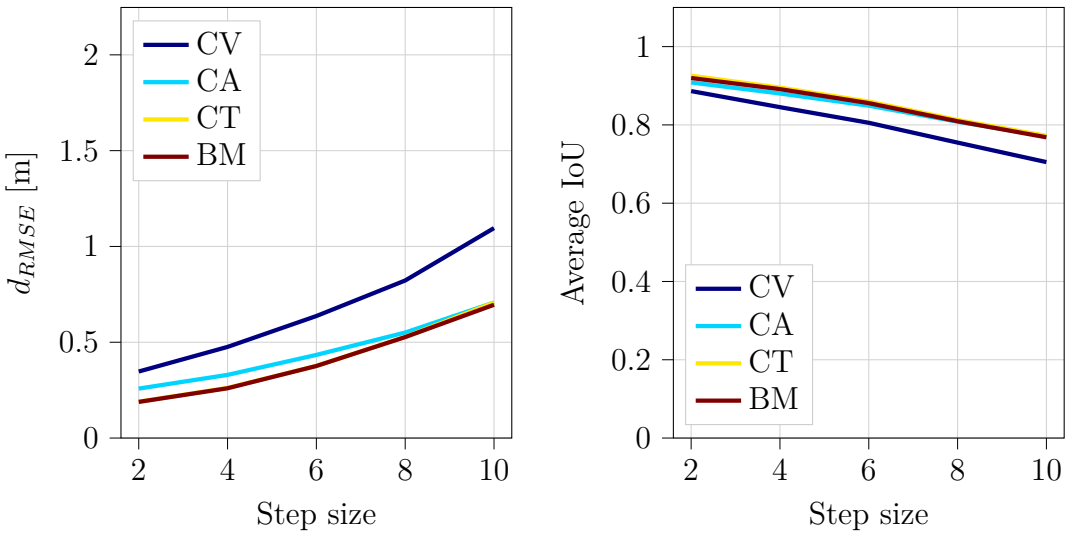
do not satisfy this condition are considered as straight. Out of the 422[1] moving car sequences there are 116 turning sequences and 306 straight sequences. The results for the turning and the straight sequences are presented in Figure 4.2 and 4.3 respectively.

[1]Of these 422 sequences, keep in mind that 20 of them have only 2 annotations and therefore have no annotations left for evaluation. This makes the number of annotations used for evaluation equal to 402 for the static annotation strategy.




**(a)** $d_{RMSE}$ in the $xy$-plane. **(b)** Average IoU in the $xy$-plane.

**Figure 4.2:** Results from the static annotation strategy when using non-stationary measurement noise. The sequences that have been used to compute these results are the turning sequences that have successfully estimated trajectories.



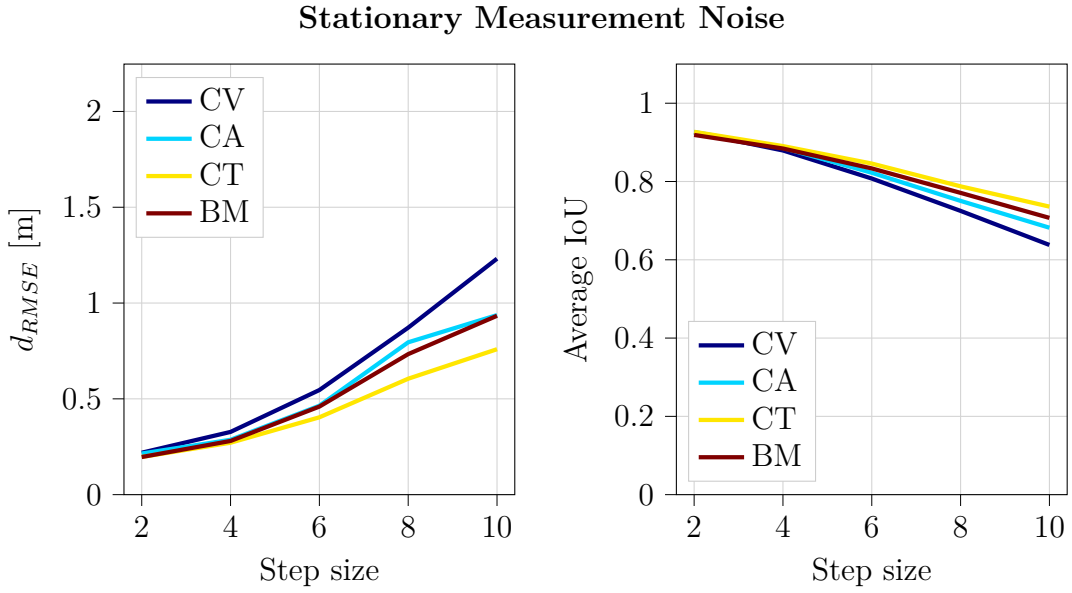

**(a)** $d_{RMSE}$ in the $xy$-plane. **(b)** Average IoU in the $xy$-plane.

**Figure 4.3:** Results from the static annotation strategy when using non-stationary measurement noise. The sequences that have been used to compute these results are the straight sequences that have successfully estimated trajectories.

### 4.1.2 Stationary Measurement Noise

The stationary measurement noise used in this section is a model that puts a lot of trust in the annotations used to estimate the trajectory. Additionally, the model

assumes the same measurement accuracy for annotations close to the ego-vehicle as for annotations far away. The results when using this measurement noise model is shown in Figure 4.4. As previously explained, only sequences where the smoother successfully generated trajectories are used to compute these results. The numbers of sequences used are specified in Table 4.2.



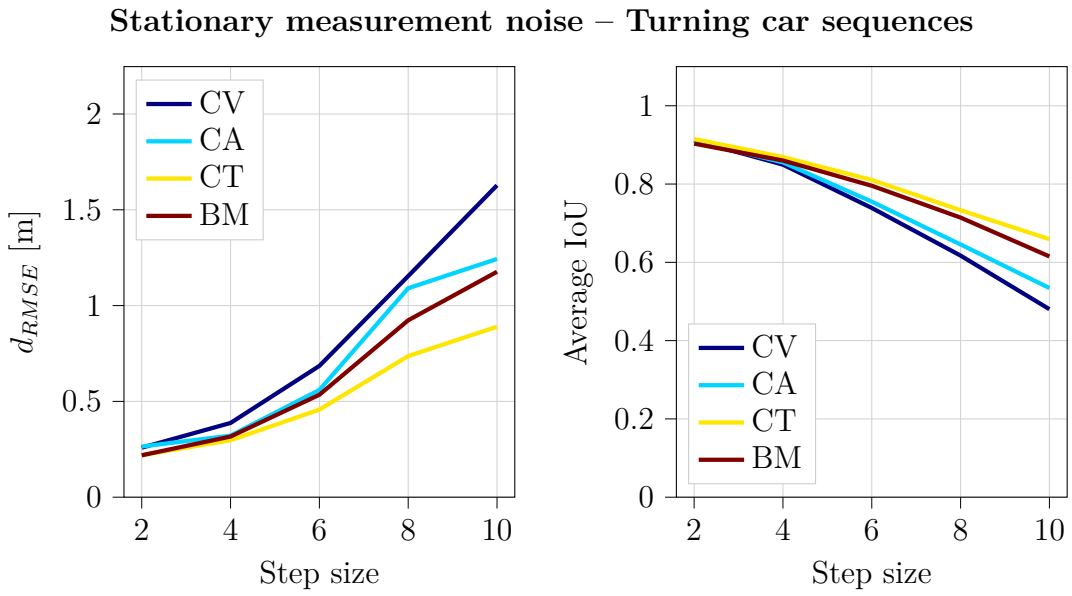

**(a)** $d_{RMSE}$ in the $xy$-plane. **(b)** Average IoU in the $xy$-plane.

**Figure 4.4:** Results from the static annotation strategy when using stationary measurement noise. The sequences that have been used to compute these results are the ones that have successfully estimated trajectories.

**Table 4.2:** The number of sequences that were evaluated in the static annotation strategy when using stationary measurement noise are specified as "Number of sequences used". "Number of sequences not used" is the number of sequences for which the smoother failed.

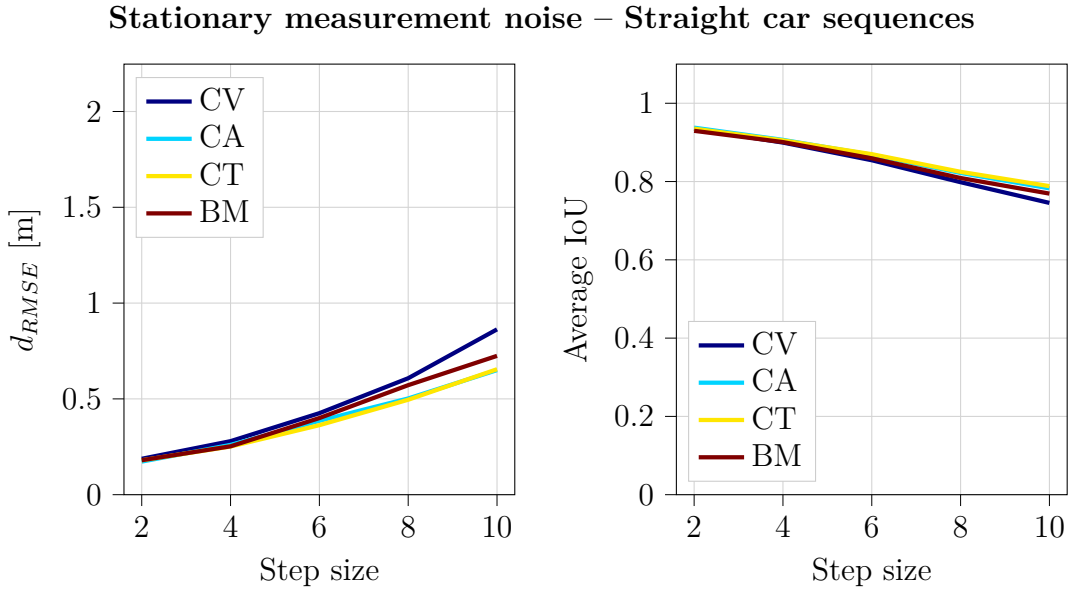
| Number of sequences not used / Number of sequences used | | | | | |
|---|---|---|---|---|---|
| Step size | 2 | 4 | 6 | 8 | 10 |
| CV | 0 / 402 | 0 / 402 | 0 / 402 | 0 / 402 | 0 / 402 |
| CA | 0 / 402 | 0 / 402 | 0 / 402 | 0 / 402 | 0 / 402 |
| CT | 1 / 401 | 1 / 401 | 2 / 400 | 3 / 399 | 3 / 399 |
| BM | 2 / 400 | 1 / 401 | 1 / 401 | 3 / 399 | 3 / 399 |

As for the non-stationary measurement noise model, a separation of the turning and the straight sequences has been made. Figure 4.5 shows the results for the turning sequences and Figure 4.6 shows the results for the straight sequences.




**(a)** $d_{RMSE}$ in the $xy$-plane. **(b)** Average IoU in the $xy$-plane.

**Figure 4.5:** Results from the static annotation strategy when using stationary measurement noise. The sequences that have been used to compute these results are the turning sequences that have successfully estimated trajectories.




**(a)** $d_{RMSE}$ in the $xy$-plane. **(b)** Average IoU in the $xy$-plane.

**Figure 4.6:** Results from the static annotation strategy when using stationary measurement noise. The sequences that have been used to compute these results are the straight sequences that have successfully estimated trajectories.

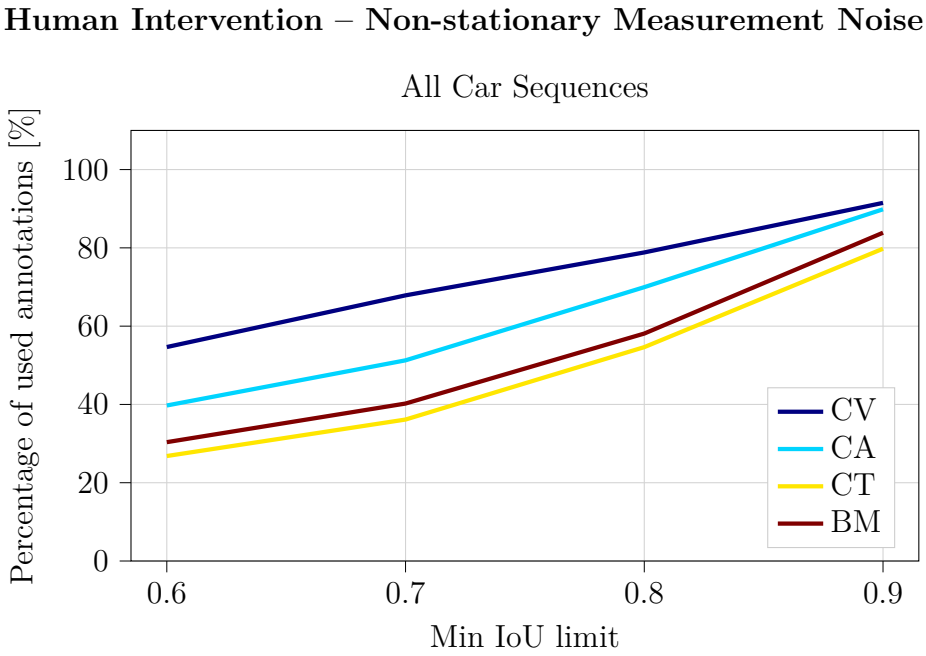
## 4.2 Adaptive Annotation Strategy

The level of automation that our smoothing algorithm can achieve is best evaluated by how much human intervention that is needed in the annotation process. In Section 3.3.2 we described adaptive annotation, which is an algorithm that simulates the workflow of a human annotator when using the smoother as an aid in the annotation process. A sensible assessment of the level of automation is the following: Count the number of annotations that the annotator must add to enable the smoother to generate a trajectory from which all box proposals satisfy the chosen quality metric. The chosen metric is IoU as described in Section 3.3.2.

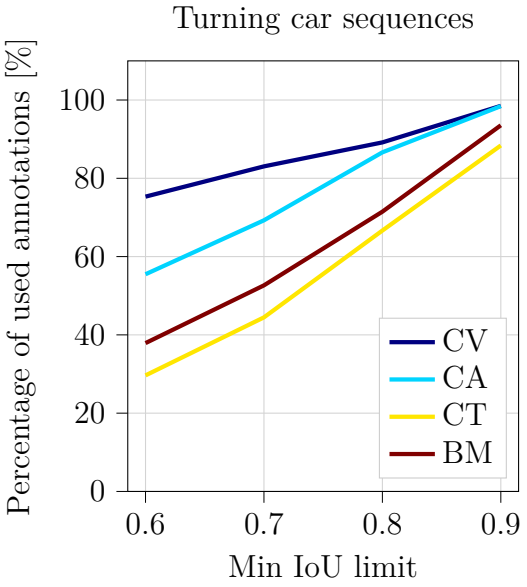
When using the non-stationary measurement noise, Figure 4.7a shows the number of annotations that must be added in order to reach a given minimum IoU limit. The vertical axis represents the number of used annotations as a percentage of the total number of annotations. This percentage is the measure of human intervention for the adaptive annotation strategy. As for the static annotation strategy, a separation of the turning and the straight sequences has been made in Figures 4.7b and 4.7c respectively.

As for the static annotation strategy, the difference in human intervention for each motion models gets smaller when the stationary measurement noise is used. This can be observed in Figure 4.8a that shows the human intervention for each motion model when the stationary noise model is used. The separation of turning and straight sequences is presented in Figures 4.8b and 4.8c. For both noise models, it appears that the CT motion model requires the least number of boxes in order to reach the specified minimum IoU level.

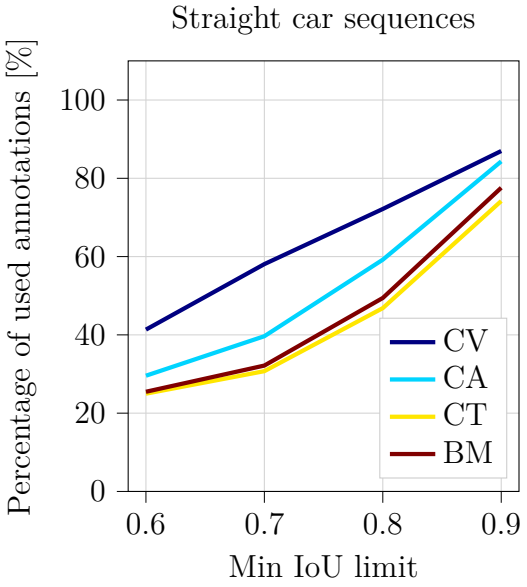
In the adaptive annotation strategy it might be the case that some sequences never reach a particular minimum IoU level. The choice of IoU level has an impact on how many sequences that reach it; A higher IoU level makes it more difficult to reach that level, which is shown in Figure 4.9. But more importantly, the different motion models also affect the number of sequences that reach the minimum IoU levels. CT is the motion model that enable the most sequences to reach the different IoU levels.




**(a)** Human intervention for all car sequences measured by the percentage of used annotations. The total number of annotations available is 6713 which corresponds to 100 %.

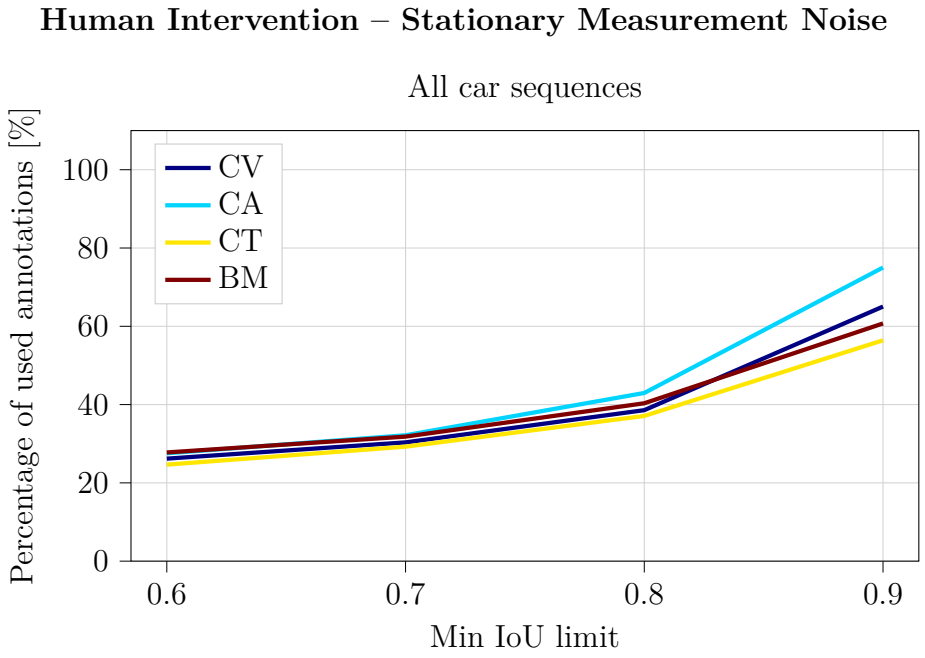



**(b)** Human intervention for turning car sequences measured by the percentage of used annotations. The total number of annotations available in the turning sequences is 2627 which corresponds to 100 %.

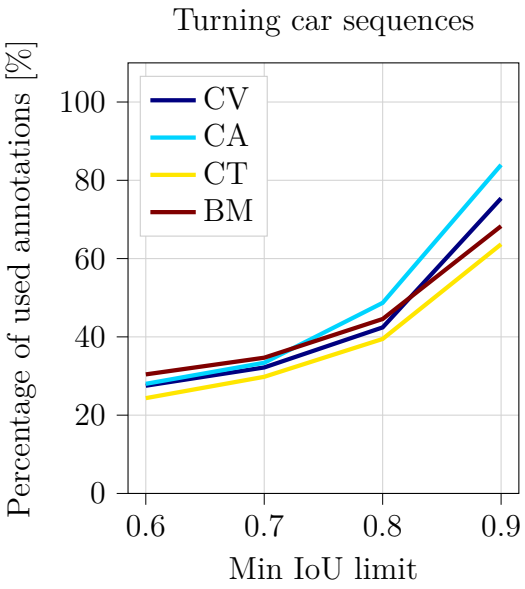



**(c)** Human intervention for straight car sequences. The total number of annotations available in the straight sequences is 4086 which corresponds to 100 %.

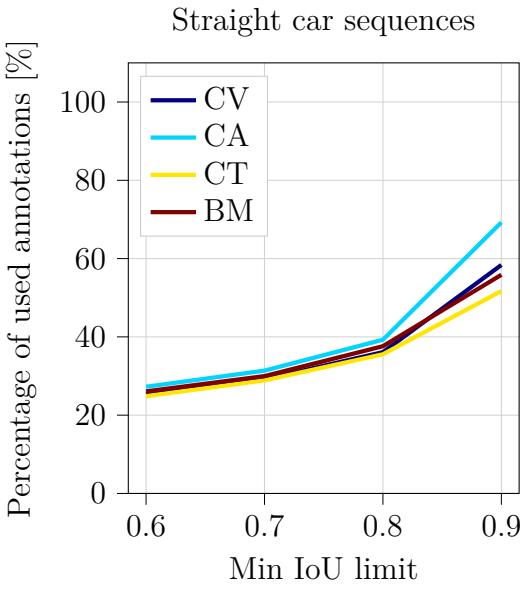
**Figure 4.7:** A measure of the level of automation is how many annotations the human has to provide in order to achieve a minimum IoU-value. The figure present how many annotations that were required for the different motion models in different scenarios measured by the percentage of used annotations.




**(a)** Human intervention for all car sequences. The total number of annotations available is 6713 which corresponds to 100 %.

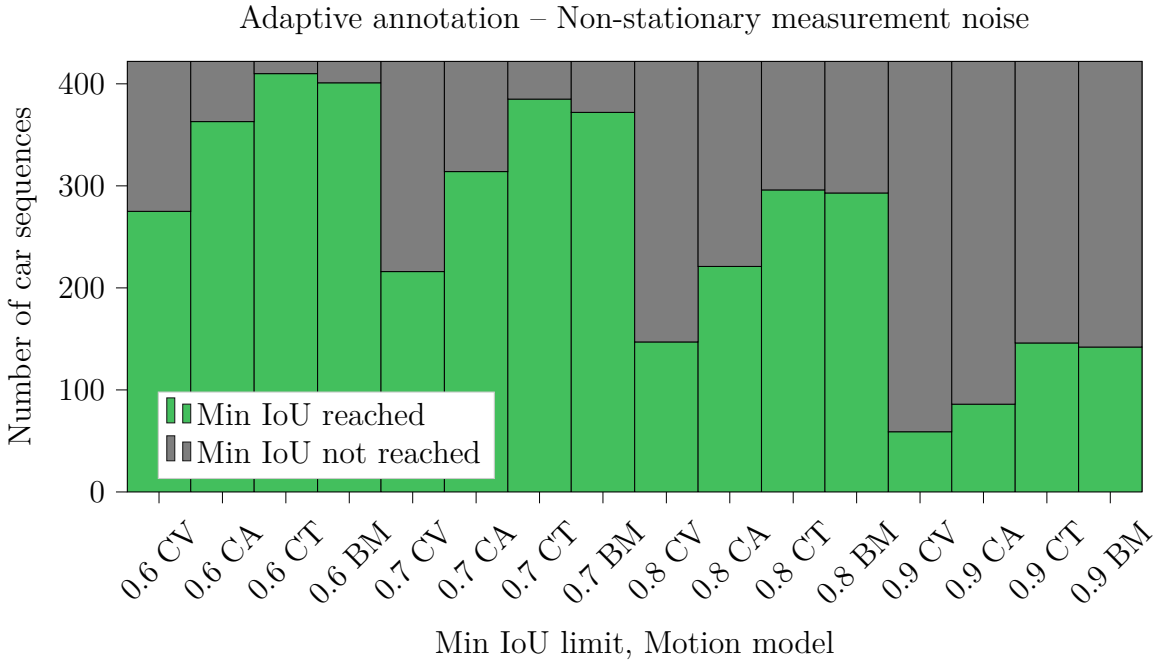



**(b)** Human intervention for turning car sequences. The total number of annotations available in the turning sequences is 2627 which corresponds to 100 %.

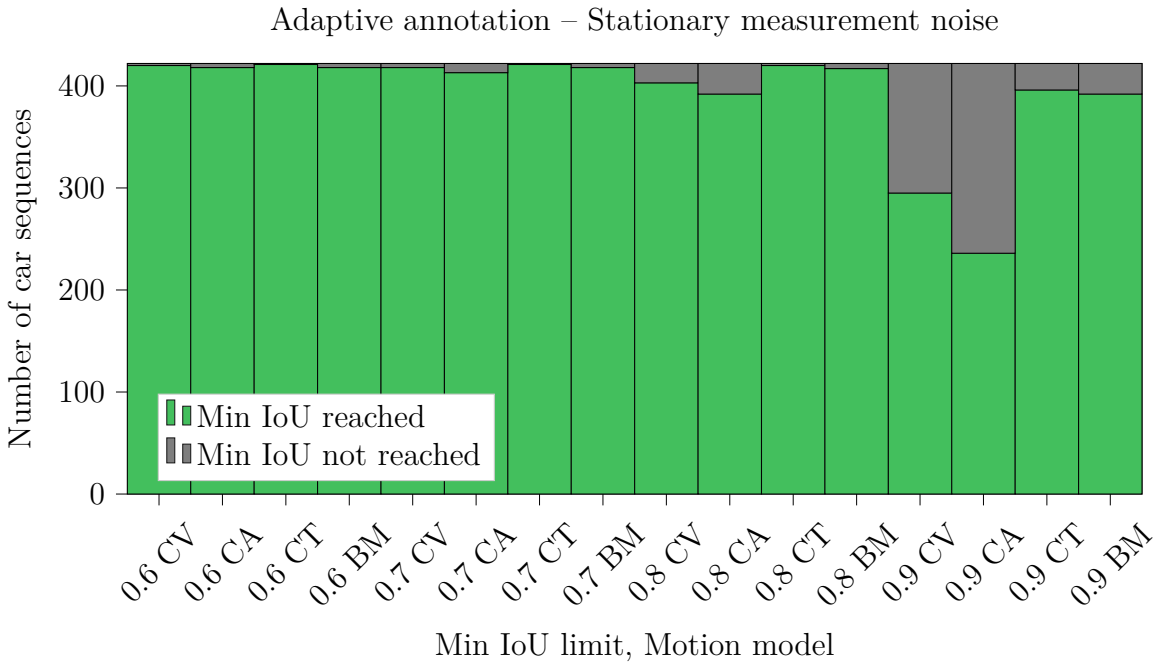



**(c)** Human intervention for straight car sequences. The total number of annotations available in the straight sequences is 4086 which corresponds to 100 %.

**Figure 4.8:** In this figure the percentage of human annotations used to achieve a minimum IoU above a threshold specified by the $x$-value in the graphs. In the top figure all sequences have been consolidated and in the bottom two figures the sequences have been split between turning and straight sequences. CT requires the least number of human annotations to achieve a minimum IoU above the different thresholds.




**(a)**




**(b)**

**Figure 4.9:** "Min IoU reached" is the number of sequences that reached the specified minimum IoU level in the adaptive annotation algorithm. "Min IoU not reached" is the number of sequences for which a trajectory was estimated but never reached the minimum IoU level.

# 5

# Discussion

When we evaluate our results we assume that the Nuscenes dataset provide the true value for the annotations and we measure how well our estimates agree with these annotations. On the other hand, when making the estimates we model the annotations in the dataset as outcomes from a probability distribution which makes them noisy, as the annotations in Figure 5.1 illustrate. The right sequence has annotations that deviate sideways from an otherwise smooth trajectory. The sequence to the left represent a car standing still for most of the time but the annotations seem to have a variance in yaw and position. The inconsistency of both considering the annotations as pure ground truth and as outcomes of a random process makes it hard to truly evaluate how well our algorithm performs. However, this issue should average out with a large number of annotated sequences and should not affect the relative ordering of the different motion models' performance.

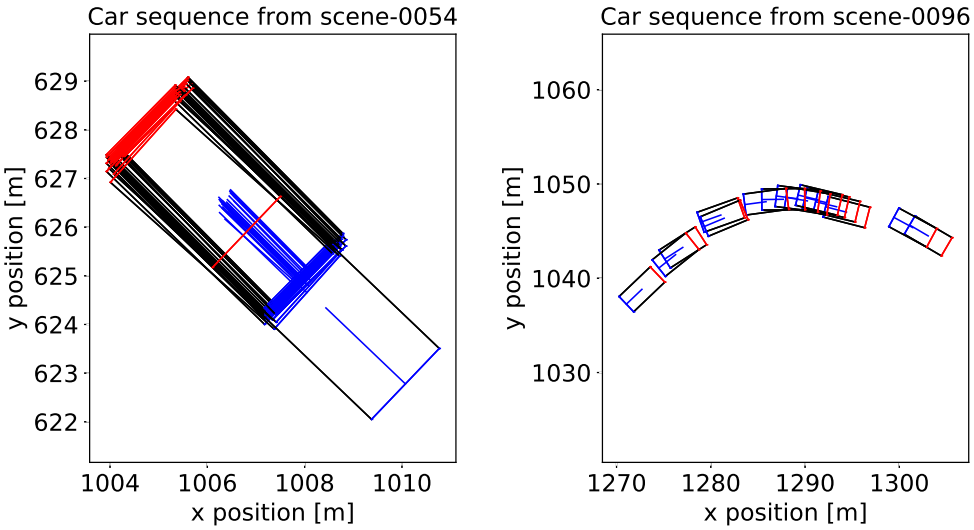



**Figure 5.1:** Two car sequences that showcase the noise in the process of creating annotations. To the left: A car standing still for most time instances, but starts to move in the end of the sequence. To the right: A moving sequence with annotations that slightly deviate from a smooth trajectory.

Given that annotations are subject to uncertainty it could be that in some cases that the algorithm's estimate is better than the annotation but the evaluated score could

be low since we compare to this potentially poor annotation. One such scenario could be a case where the car is heavily occluded and the state variables of the annotation is hard to infer from what is visible to the annotator. The algorithm can make use of both future and past positions, velocities, and yaw angles to make a good estimate even though the information from the current state is poor.

## 5.1 Motion Models

The results for the static annotation strategy show that there is a significant difference between the different motion models' ability to generate accurate trajectory estimations, in particular with the non-stationary measurement noise model. The non-linear motion models CT and BM yield trajectory estimations with lower $d_{RMSE}$ and higher average IoU than the linear models CV and CA. They also enable a lower human intervention as demonstrated by the adaptive annotation results.

There is a clear difference in the results for CV and CA where CA performs better both in terms of $d_{RMSE}$, average IoU and human intervention. We believe that this is explained by the inability of CV to capture the accelerations of the cars in the dataset. This raises the question whether the performance of CT and BM could be improved since they currently assume a constant $xy$-velocity ($\dot{v} = 0$ as shown by Equation (2.64) and (2.65)). Therefore, a hypothesis for future studies is to model the $xy$-acceleration as constant for CT and BM, i.e. $\dot{v} = a$ and $\dot{a} = 0$.

The difference between the performance of the linear and the non-linear motion models can be explained by the fact that CT and BM model the kinematics of a car in a more realistic way compared to CV and CA. CT models the change in the heading direction by assuming that the the turn rate is constant (i.e. the angular velocity of the yaw angle is constant) and couples the yaw angle to the $x$- and $y$-velocity by projecting speed, $v$, onto the $x$, and $y$ axis (see Equation (2.64)). BM couples the turn rate to the $xy$-velocity and to a front steering angle, $\delta_f$ (see Equation (2.65)). These assumptions are reasonable descriptions of how cars turn compared to CV and CA that do not couple the turn rate to the heading of the car (see Equations (2.62) and (2.63)).

A weakness of our evaluation of results is that the motion models can make estimates that are physically unlikely in normal city traffic without having a big impact on the evaluated metric scores. In Figure 5.2 the estimates for one of the more difficult sequences is made using different motion models. All estimates are made using the adaptive annotation strategy. Note that CV and CA make estimates that would require that the car slides slightly sideways, a motion that is not present in the true trajectory.

The sequence in Figure 5.2 is particularly difficult for the motion models involved in this thesis. Due the the sharp turn in the beginning it is only CT that has enabled the adaptive annotation process to estimate a trajectory with a minimum IoU larger

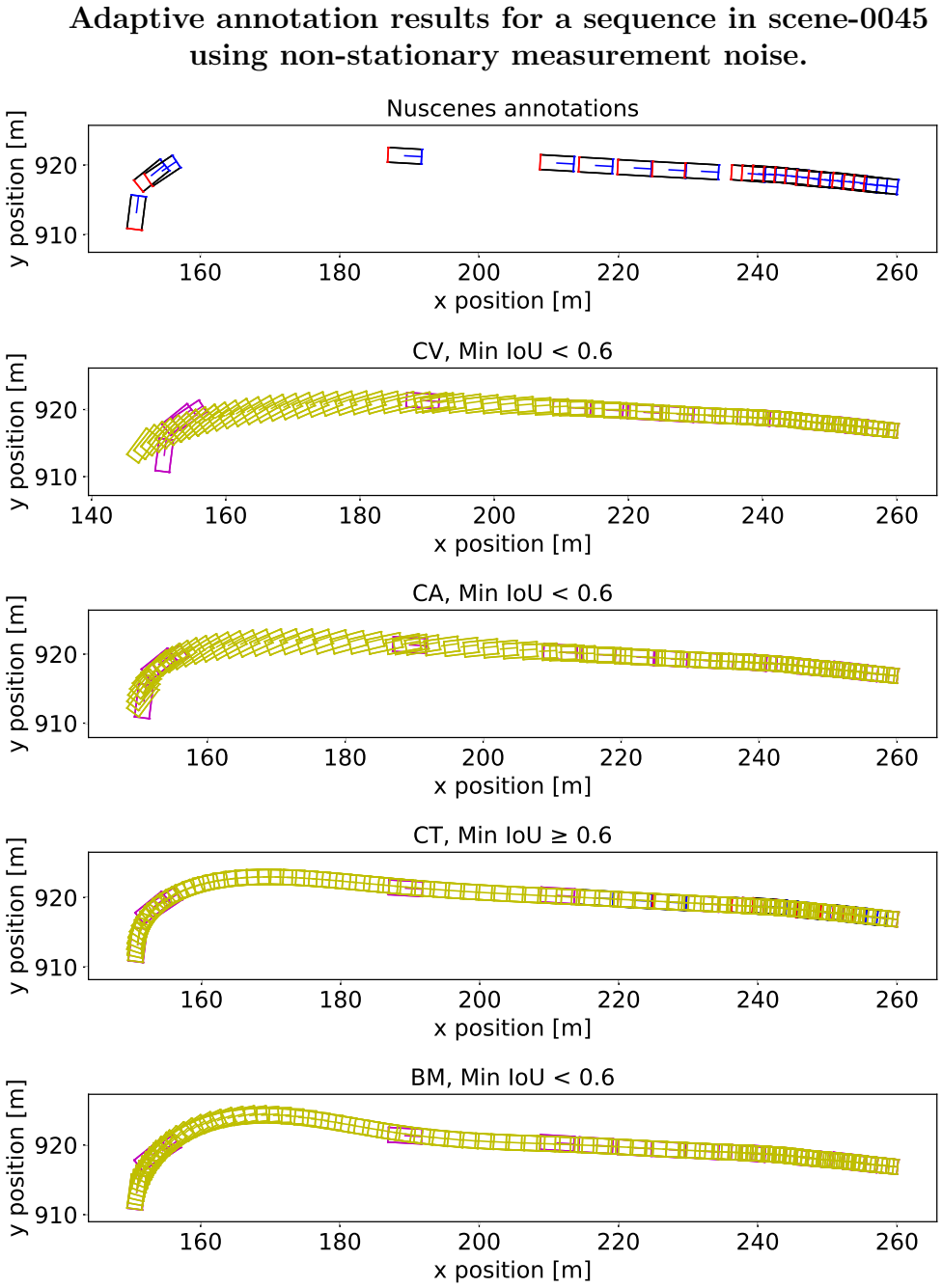



**Figure 5.2:** This figure aims to visualize that the non-linear motion models yields estimated trajectories that are more likely to happen in city traffic. The estimates for the linear models show behaviours where the heading and yaw of the car match in such a way that would require cars to slide. This does not happen in the dataset but our current metrics can not penalize that the estimated trajectories suggest this.

than 0.6 using the non-stationary measurement noise. BM is almost there but has an IoU lower than 0.6 for one of the boxes in the beginning of the sequence. However, it is clear that both CT and BM generate more realistic trajectories than CV and CA.

## 5.2 Failing Sequences

The smoother fails to produce trajectory estimations for a few sequences and the number of failed sequences ranges from three to six according to the Tables 4.1 and 4.2. In this section the different causes for these failures are discussed and analyzed.

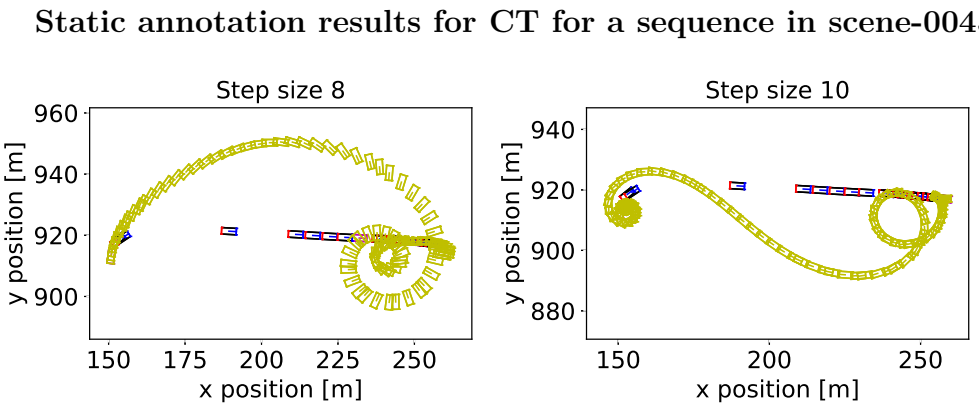
### 5.2.1 Insensible Trajectory Estimations

In the case of static annotation and large step sizes there is a higher risk for bad trajectory estimations. This is easily explained by the fact very few annotations are observed by the smoother. For step size 8 and 10 only 21 % and 19 % of the total number of annotations are observed respectively (see Table 3.1). For CT it leads to the trajectories shown in Figure 5.3a that obviously are false. However, it is possible to see the characteristics of the CT model in these results: a constant turn rate. Additionally, the kinematics for this motion model does not require the velocity vector to point in the same direction as the front of the car. With this in mind, it is not strange that these trajectories have been created since the algorithm uses the first two boxes in to guess a prior velocity vector and orientation. This is clearly demonstrated by the result for step size 8 in Figure 5.3a.

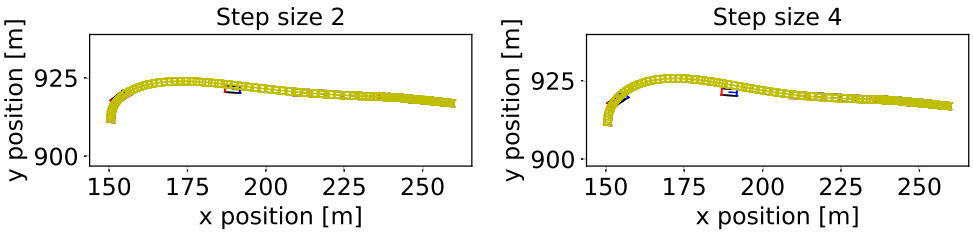
The fact that CT does not couple the direction of the box to the direction of the velocity vector can be thought of as one of its disadvantages. On the other hand, this is a rare case and once a couple of more annotations are added a sensible trajectory is estimated as shown in Figure 5.3b. This is a better result but a bit too far from the optimal solution. It is clear that the static annotation strategy suffers from that it does not add the box that contains most information about the trajectory. However, it is safe to consider this as a special case since the sequence difficult to process and has caused the algorithm to fail several times.

### 5.2.2 Motion Model Linearization Failures

Another reason for failure is when the SLR performs a bad linearization of the motion model $f_k(\mathbf{x}_k)$. This may happen if there is an attempt to linearize $f_k$ in a region where it is highly non-linear. This causes numerical problems and leads to the matrix $F_k$ being ill-conditioned. In these situations, the elements of $F_k$ explode and become very large which in turn leads to very large numbers in the estimated




**(a)** Trajectory estimations using large step sizes.




**(b)** Trajectory estimations using step size 2 and 4.

**Figure 5.3:** The trajectory estimations for this sequence using step size 6, 8 and 10 were classified as failed (results for step size 6 are omitted here). The used prediction frequency is 10 Hz but the estimated trajectory (yellow boxes) is plotted at 5 Hz. The step sizes 2 and 4 generate more sensible results. Note that the fourth annotation contains a lot of information but is never observed.

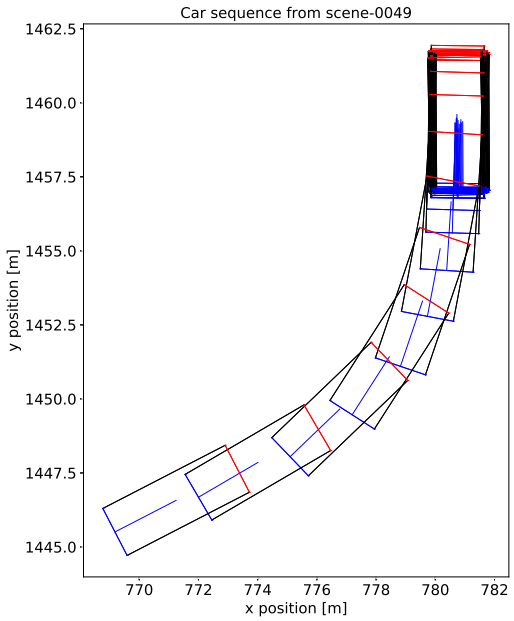



**Figure 5.4:** For this sequence a linearization failure occurred for static annotation, step size 2 while using BM as motion model. In the beginning of the sequence the car has stopped but the annotations are noisy for those time instances. Relying too much on those boxes would imply sideways movement of the car.

state vectors $\mathbf{x}_k$. These solutions have been marked as failed since the box center coordinates exceed reasonable levels.

The IPLS may also raise errors in these situations and it is the matrix inversion operation $X_{slr}^{-1}$ in Algorithm 5 that fails and causes the IPLS to exit without returning any trajectory.

In the results for static annotation this failure only happens for BM and not for the other motion models. What makes BM different from the other motion models is that it involves the unbounded function $\tan(\delta_f)$ which approaches $\pm\infty$ as the front steering angle, $\delta_f$, approaches $\pm\pi/2$ radians. Even though a front wheel steering angle close to 90° does not make sense for a real car it might be the case that $\delta_f$ ends up close to 90° if subsequent observed annotations move sideways.

A situation where a linearization error has happened is for static annotation of the sequence in Figure 5.4 using step size 2. In the beginning of the sequence the annotations are concentrated around a stopped car but they are noisy and deviate both in the heading direction and sideways. It is plausible that the smoother estimates $\delta_f$ to be almost 90° if the relative displacement of two subsequent observed boxes is completely sideways.

Possible solutions to this problem is to model the measurement noise in a different way. An increased measurement uncertainty would cause the model to rely more on

the motion model. Another solution could be to start providing observed annotations to the smoother at the time when the car starts to move. There is no point in providing many annotations in a time interval where the car does not move.

Even though the square-root implementation of the IPLS is a remedy to the problem of negative definite covariance matrices, it is still sensitive to linearization of highly non-linear motion models. However, the BM has succeeded to produce accurate trajectory estimations for the majority of the sequences and performs reasonably well.

## 5.3 Measurement Noise Models

As seen in Chapter 4 the results depend on the choice of measurement model and this section will discuss the advantages and drawbacks of the different choices.

### 5.3.1 Implications of an Overconfident Measurement Model

The difference between the motion models in terms of $d_{RMSE}$, average IoU and percentage of used annotations gets smaller when changing the measurement noise from the stationary to the non-stationary model. The reason is that the diagonal values in the covariance matrix $R_k$ are small in the stationary model and thereby it assumes that the annotations are close to the actual ground truth. This forces the estimated boxes to end up closer to the observed annotations which is demonstrated in Figure 5.5. The right plot represents the stationary measurement model that forces the estimate of the last box closer to the annotation compared to the non-stationary measurement model in the left plot.

In Figure 5.5 static annotation with step size 6 has been used. If the step size is smaller then the estimated trajectory will be even closer to the annotations in the dataset. This means that the model relies more on the observed annotations and less on the motion model predictions. At the same time the annotations are considered to be outcomes of a noisy measurement process which puts a lower bound on the $d_{RMSE}$ and an upper bound on the average IoU.

This is the reason why the values of $d_{RMSE}$ and the average IoU converge to the same value for smaller step sizes in the Figures 4.4, 4.5, and 4.6. Also, the difference between the motion models in terms of human intervention is smaller with stationary measurement noise as Figure 4.8c shows. The case where the errors are the smallest and there is almost no difference between the motion models is in Figure 4.6, that corresponds to straight car sequences turning less than 0.5 radians.

With the stationary measurement noise it is easy for the IPLS to generate a trajectory that fits tight to the annotations. In static annotation, it appears that the difference in error between the motion models is almost zero for all step sizes and

**Illustration of the effects of the measurement models on the estimated trajectories**

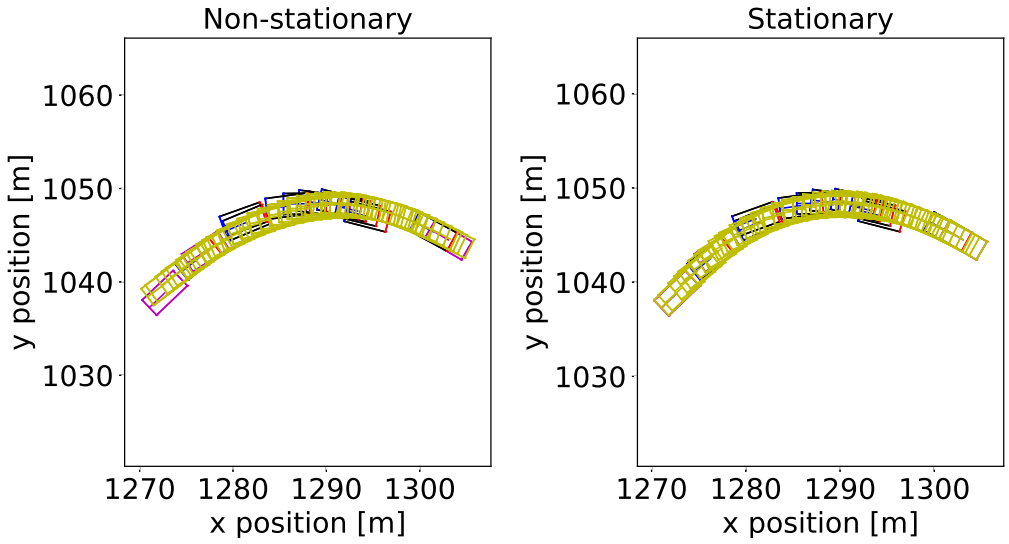



**Figure 5.5:** This figure shows estimated trajectories (yellow boxes) drawn on top of the Nuscenes annotation sequence shown in Figure 5.1. The purple boxes are the observed annotations and the difference between the estimations is the model used for $R_k$. These estimation are the result of the static annotation strategy with step size 6.

therefore we cannot expect the estimated trajectories to have smaller $d_{RMSE}$ than in Figure 4.6a and higher average IoU than in Figure 4.6b.

### 5.3.2 Distance Dependent Measurement Accuracy

The stationary measurement model assumes the same accuracy for all annotations regardless of the distance to the ego-vehicle, whereas the non-stationary model assigns a larger uncertainty to annotations far away. As discussed in Section 2.6.2 this is probably a more realistic assumption.

The way that the trajectories are evaluated is a bit contradictory to this way of thinking of the annotations since we evaluate as if the annotations are always true. Under the assumption that annotations are outcomes from a probability distribution it would make more sense to use a statistical evaluation method.

One possible way to evaluate could be to examine a Gaussian pdf centered at the estimated annotation, $H\hat{\mathbf{x}}_k$, and with the covariance matrix, $R_k$, of the annotation, $\mathbf{y}_k$. Given this distribution $\mathcal{N}(H\hat{\mathbf{x}}_k, R_k)$ how likely it is that annotation is an outcome of this distribution? Hence the new metric would yield a probability value for all estimates with a corresponding annotation. Using this new metric the heuristic of the adaptive annotation strategy could be modified accordingly and add the annotation which has the lowest probability instead of the one with largest center deviation.

## 5.4 Annotation Strategies

The second research question presented in Section 1.3 is "How does the level of human intervention affect the quality of the automatically generated annotations?". The results for the static annotation strategy provides answer to this question under the assumption that quality means a low RMSE for center deviation and high average IoU. The adaptive annotation strategy is developed with the interpretation of quality being "no annotation is allowed to be worse than a lower bound". The static annotation view on quality might be more suitable if the goal is to generate large quantities of data but the consistency requirements are low. Adaptive annotation is instead more focused on providing a high consistency in the annotations, potentially at a cost of more time spent per annotation.

With this in mind, we find it likely that the adaptive annotation strategy achieves a higher level of automation because some annotations are more informative compared to others, a fact that the adaptive annotation strategy tries to leverage. For example, an annotation that has the same position, yaw, and size as the estimate from the prediction step would not change the mean of the filtering estimate after the update step, which causes the MAP estimate to be the same as if no measurement was present at the current time instance. This is likely the reason that the non-linear motion models requires fewer annotations compared to the linear models since the kinematics of the non-linear models better predict the next state.

The adaptive annotation strategy seems to be the favorable way to achieve a higher level of automation. However, the search for where to add the next annotation is more complex in the adaptive case compared to the static. Our algorithm can not provide where to add the next annotation and would rely on the annotators to identify this themselves.

The saved time from reducing the number of human annotations needed could potentially be consumed by the time the annotators would spend identifying where to add the next annotation. Another drawback is that the annotator's choice is not necessarily the choice that provides the most information to the algorithm. The trade off between reducing the number of human annotations in relation to the cost of introducing the search for the next time instance needs to be further investigated to have a conclusive result on how much time can be saved using the adaptive strategy.

### 5.4.1 Combining Static and Adaptive Annotation

The two strategies are not necessarily competing alternatives, one could imagine a combination of both strategies. The static annotation strategy could provide a starting point for the adaptive strategy. Given the sparse set of annotations from the static strategy, an initial trajectory could be estimated which is likely to be fairly close to the true trajectory. Then the annotator could review the initial trajectory,

identify where the error is the largest and add an annotation there as suggested by the adaptive annotation strategy.

### 5.4.2 Change from Relative to Inertial Reference Frame

At the moment of data gathering, all sensors are mounted on the ego vehicle and information from e.g. LiDAR and camera are captured with respect to the body frame of the vehicle. For this reason it is likely that the annotations are made in the coordinate system of the body frame. The algorithm in this thesis require the annotations to be expressed in the inertial frame of reference where the ego vehicle move. This means that the pose of the ego vehicle have to be estimated at each time instance to enable to mapping of annotations from the body frame to the inertial frame. As always when estimating a quantity the estimate is not guaranteed to completely agree with the true value. This means that the uncertainties from ego pose estimation will propagate into the annotation poses in the inertial frame. In the left plot of Figure 5.1 a sequence of annotations can be seen where the car is labeled as 'stopped' by the the annotator for almost all time instances. The jitter present in the annotations could possibly stem from uncertainties in the yaw angle estimate of the ego vehicle. The error propagation of ego yaw angle error scales with the distance to the annotations which is one more argument for that the measurement noise should increase with distance.

# 6

# Conclusion

This chapter will first make a few concluding remarks and then suggest future work that likely would improve our algorithm, thereby further raising the level of automation without increased error.

## 6.1 The Best Performing Motion Model

The CT motion model consistently performs quantitatively best for all combinations of annotation strategy and type of measurement noise. The qualitative results presented in Figure 5.2 also suggests that it make estimations that are physically probable.

The next best model is the BM which follow closely after the CT and it is possible that an even better tuning of the models could shift the relative ordering of their performance. The CT model have the advantage of being simpler using fewer state variables but still outperforms the more complex BM. The advantage of a lower dimensional state vector also gives a lower computational cost when making the estimations. The CT model never crashed as a result of linearization issues in the SLR which the BM did for one sequence.

## 6.2 The Achieved Level of Automation

Level of automation is in this thesis measured by the level of human intervention needed to achieve a certain value of a quality indicator: minimum IoU for the boxes in a sequence. It can be concluded that CT has the best performance and requires the least amount of human intervention as Figure 4.7 show and the intervention values for non-stationary measurement noise are listed in Table 6.1. The CT model also performs slightly better than BM in terms of how many sequences that reach the minimum IoU limits and how many that do not as shown by Figure 4.9.

The Nuscenes dataset have annotations at 2 Hz which means that sets an lower bound on how often evaluations can be made. If the aim is to annotate a dataset

**Table 6.1:** The incidence of human intervention for the CT model using non-stationary measurement noise. These values correspond to the plotted line for CT in Figure 4.7a.

| Min IoU limit | 0.6 | 0.7 | 0.8 | 0.9 |
|---|---|---|---|---|
| Human intervention [%] | 27 | 36 | 55 | 80 |

at higher frequency the level of automation will almost certainly increase. For example if a dataset consisting of 500 s of sampled data should be annotated at 2 Hz (1000 frames) our results suggests that 80 % or 800 of the frames would have to be annotated to achieve a minimum IoU of 0.9. If the same 500 s should be annotated at 36 Hz, i.e. 18000 frames, we find it very likely that the number of human annotations does not increase since the trajectory is estimated over the same time span. This would mean that for a dataset annotated at 36 Hz the fraction between human annotated frames and total annotated frames is 0.044, i.e. a human intervention of 4.4 %.

## 6.3 Future Work

There are still components that can be improved and we will suggest some different areas where further work can be done.

### 6.3.1 Better Approximations of Process Noise

When discretizing the covariance matrix of the process noise an integral was solved analytically for the linear models but for the non-linear models the continuous covariance matrices were simply multiplied with the length of the time step taken. This approximation is not the best possible. Gustafsson suggests that the integral which analytically discretizes the linear process noise (Equation (2.74)) by using fast sampling (see Section 2.6.1) can be approximated by a Riemann sum to obtain a noise more accurate noise covariance matrix [6].

### 6.3.2 Modeling of Measurement Noise

The annotations are results of human judgment and it can be questioned if humans are capable of providing ground truth in the sense of absolute certainty. We believe that the annotations are outcomes of a stochastic process where the certainty depend on a number of circumstances. One improvement to the algorithm suggested in this thesis is a better modeling of the measurement noise based on these circumstances that influence the ability of the annotator to accurately create the annotations. Currently, both the stationary and non-stationary noise models are fairly arbitrary and not the result of a stringent optimization process. By considering distance from the ego-car, occlusion rate, number of LiDAR-points on the object, and if the object

is in focus in the camera the noise modeling could be improved. When the model is established the optimal parameters could be found using some form of optimization method, for example EM-algorithm.

### 6.3.3 Object Detection and Tracking

Our hypothesis is that 3D object detections from a suitable object detection algorithm, such as [15; 17; 18; 19], can improve the estimated trajectory compared to those were only sparse annotations and motion models are used. This would imply that the need for human annotation is reduced without losing accuracy.

The object detections, represented by a measurement vector $\mathbf{z}_k$, will likely be less reliable compared to the annotations, $\mathbf{y}_k$, and therefore a different measurement model is needed to account for this uncertainty:

$$\mathbf{z}_k = h_k^{OD}(\mathbf{x}_k) + \mathbf{r}_k^{OD}. \tag{6.1}$$

Here the measurement model for object detections is $h_k^{OD}(\cdot)$ and the associated measurement noise is $\mathbf{r}_k^{OD}$. Since the annotations are likely to be more accurate than object detections it will probably not be very useful to consider the information from object detections in the annotated time instances. In the unannotated time instances it could be of greater utility to use that information. Figure 6.1 shows such an SSM where, hypothetically, the information from $\mathbf{z}_2$ could improve the estimated state vector $\mathbf{x}_2$ which will lead to an updated smoothing solution for the trajectory.

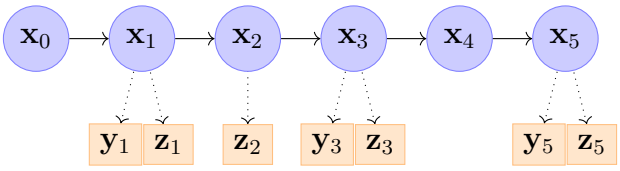



**Figure 6.1:** The figure shows the state space model that could be used to include object detections in the state estimates. The blue nodes ($\mathbf{x}_i$) are the hidden states (pose of cuboid) and the orange nodes are observation of the states, $\mathbf{y}_i$ being human annotations and $\mathbf{z}_i$ output from object detections.

### 6.3.4 Optimal Choice of Next Time Step to Annotate

The adaptive annotation strategy needs to identify the time instance to annotate next. As mentioned this identification is currently expected to be done by the annotator which adds to the average time needed to create each annotation. If this task could be performed automatically the annotator would only have to provide the actual annotation and not the search for where to annotate. One possible idea is to use the state covariance matrix, $P_{k|T}$, provided by our algorithm for each estimate,

which could be used as an uncertainty measurement that hopefully has a positive correlation with the estimation error. If this correlation exists the state covariance could be used as a proxy when searching for the largest estimation error by simply searching for the largest state covariance instead.

A second idea is to use the result from object tracking and compare the results from object tracking to the state estimations from smoothing and add annotations where the disagreement is largest.

# Bibliography


[1] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," *arXiv preprint arXiv:1903.11027*, 2019.

[2] F. Yu, W. Xian, Y. Chen, F. Liu, M. Liao, V. Madhavan, and T. Darrell, "BDD100K: A diverse driving video database with scalable annotation tooling," *CoRR*, vol. abs/1805.04687, 2018.

[3] S. Särkkä, *Bayesian Filtering and Smoothing*. Cambridge University Press, 2013.

[4] A. F. García-Fernández, L. Svensson, and S. Särkkä, "Iterated posterior linearization smoother," *IEEE Transactions on Automatic Control*, vol. 62, pp. 2056–2063, April 2017.

[5] . F. García-Fernández, L. Svensson, M. R. Morelande, and S. Särkkä, "Posterior linearization filter: Principles and implementation using sigma points," *IEEE Transactions on Signal Processing*, vol. 63, pp. 5561–5573, Oct 2015.

[6] F. Gustafsson, *Statistical sensor fusion*. Studentlitteratur, 2010.

[7] I. Arasaratnam, S. Haykin, and R. J. Elliott, "Discrete-time nonlinear filtering algorithms using gauss–hermite quadrature," *Proceedings of the IEEE*, vol. 95, no. 5, pp. 953–977, 2007.

[8] C. M. Bishop, *Pattern recognition and machine learning.* Information science and statistics, New York, NY : Springer, cop. 2006., 2006.

[9] M. G. Rutten, "Square-root unscented filtering and smoothing," in *2013 IEEE Eighth International Conference on Intelligent Sensors, Sensor Networks and Information Processing*, pp. 294–299, IEEE, 2013.

[10] S. S. Blackman and R. Popoli, *Design and analysis of modern tracking systems.* The Artech radar library, Norwood, MA : Artech House, cop. 1999., 1999.

[11] X. R. Li and V. P. Jilkov, "Survey of maneuvering target tracking. part i. dynamic models," *IEEE Transactions on aerospace and electronic systems*, vol. 39, no. 4, pp. 1333–1364, 2003.

[12] J. WONG, "Theory of ground vehicles," 1978.

[13] P. Polack, F. Altché, B. d'Andréa Novel, and A. de La Fortelle, "The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles?," in *Intelligent Vehicles Symposium (IV), 2017 IEEE*, pp. 812–818, IEEE, 2017.

[14] J. Kong, M. Pfeiffer, G. Schildbach, and F. Borrelli, "Kinematic and dynamic vehicle models for autonomous driving control design.," in *Intelligent Vehicles Symposium*, pp. 1094–1099, 2015.

[15] W. Luo, B. Yang, and R. Urtasun, "Fast and furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3569–3577, 2018.

[16] P. Abbeel, A. Coates, M. Montemerlo, A. Y. Ng, and S. Thrun, "Discriminative training of kalman filters.," in *Robotics: Science and systems*, vol. 2, p. 1, 2005.

[17] J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. Waslander, "Joint 3d proposal generation and object detection from view aggregation," *IROS*, 2018.

[18] Y. Zhou and O. Tuzel, "Voxelnet: End-to-end learning for point cloud based 3d object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4490–4499, 2018.

[19] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "Pointpillars: Fast encoders for object detection from point clouds," *arXiv preprint arXiv:1812.05784*, 2018.

# A

# Appendix 1

The appendix provides implementation details regarding the adaptive annotation strategy.

## A.1 Adaptive Annotation Implementation Details

A detailed description of the adaptive annotation process is given in Algorithm 8 that starts with an annotation sequence, $\mathbf{y}_{1:T} = (\mathbf{y}_1, \ldots, \mathbf{y}_T)$, from Nuscenes, along with a specified IoU limit as input. The result is a selected subset of annotations, $\mathbf{Y}_s$, that when passed to the IPLS results in an estimated trajectory, $\mathbf{x}_{1:T}$ stored in latestSuccessfulTrajectory. Additionally, the output variables sequenceFailed and IoULimitReached each contain a boolean value that tell how the adaptive annotation process terminated. sequenceFailed is *true* if the IPLS failed in all iterations in the while loop, otherwise it is $false$. IoULimitReached is *true* if the adaptive annotation process reached a trajectory $\mathbf{x}_{1:T}$ that has a larger minimum IoU value equal to or larger than the IoU limit, otherwise it is $false$.

As previously explained, there is a risk that the IPLS fails to estimate a trajectory and that is why the function `IPLS` outputs not only the trajectory, $\mathbf{x}_{1:T}$, but also a boolean value stored in success that indicates if the smoothing was successful or not along with failureReason that specifies the cause for failure. If the smoothing was successful then the minimum IoU of the estimated trajectory is computed by `computeMinIoU`[1] and compared to the IoU limit. Instead if a failure happens, then the adaptive algorithm tries to add an annotation to $\mathbf{Y}_s$ by the method `indexToAddHeuristic` which is described in Algorithm 9.

The function `distance` calculates the two-dimensional Euclidean distance between the $xy$-coordinates of a given annotation, $\mathbf{y}_i$, and the $xy$-coordinates of the estimated state vector, $\mathbf{x}_i$, at the same time instance. This is used to find the index $j$ of the time instance where the $xy$ center point deviates the most from the annotation.

[1] IoU is computed in the $xy$-plane according to Equation (3.3). Details about the implementation of `computeMinIoU` are not included in this document.

**Algorithm 8** Adaptive annotation strategy

**Data:** Sequence of annotations: $\mathbf{y}_{1:T} = (\mathbf{y}_1, \ldots, \mathbf{y}_T)$, IoU limit: IoUlim
**Result:** $\mathbf{Y}_s \subseteq \mathbf{y}_{1:T}$, latestSuccessfulTrajectory, sequenceFailed and IoULimitReached

```
Initialize Y_s = ∅
Initialize successList as an empty ordered list.
Add the first, y_1, and the last, y_T, to Y_s.
continueLoop ← true, adapIter ← 0, sequenceFailed ← false
while continueLoop do
    adapIter ← adapIter + 1
    x_{1:T}, success, failureReason ← IPLS(Y_s)
    Append success to successList
    if success then
        latestSuccessfulTrajectory ← x_{1:T}
        minIoU ← computeMinIoU(x_{1:T})
        if minIoU >= IoUlim then
            The minimum IoU level has been reached: DONE!
            continueLoop ← false, IoULimitReached ← true
        else if Y_s = y_{1:T} then
            continueLoop ← false, IoULimitReached ← false
        end
        if continueLoop then
            d_max ← 0.0
            foreach i such that y_i is not in Y_s do
                d ← distance(y_i, x_i)
                if d > d_max then
                    d_max ← d
                    j ← i
                end
            end
            Add y_j to Y_s.
        end
    else
        if Y_s = y_{1:T} then
            continueLoop ← false, IoULimitReached ← false
            if all elements in successList are false then
                sequenceFailed ← true
            end
        else
            j ← indexToAddHeuristic(failureReason, adapIter, successList, latestSuccessfulTrajectory, y_{1:T}, Y_s)
            Add y_j to Y_s
        end
    end
end
```

The function `indexToAddHeuristic` described in Algorithm 9 determines which annotation to add in case that the most recent call to `IPLS` failed. Depending on the failureReason, which can be either *exception* or *large distance*, different choices to assign the index $j$ are made.

**Algorithm 9** `indexToAddHeuristic`

**Data:** failureReason, adaptIter, successList, latestSuccessfulTrajectory, $\mathbf{y}_{1:T}$, $\mathbf{Y}_s$
**Result:** Index of the annotation to add: $j$

```
if failureReason = exception then
    if adaptIter = 1 and first element in successList = false and |y_{1:T}| >= 3 then
        j ← roundDownToNearestInteger(T/2)
    else if all elements in successList are false then
        dtLimit ← 0.6 sec
        j ← heuristicDt(Y_s, dtLimit)
        if j = null then
            j ← heuristicTimeGap(Y_s)
        end
    else
        If this happens then there is a previous success available
        x_{1:T} ← latestSuccessfulTrajectory
        x_i are the elements of x_{1:T}.
        d_max ← 0.0
        foreach i such that y_i is not in Y_s do
            d ← distance(y_i, x_i)
            if d > d_max then
                d_max ← d
                j ← i
            end
        end
    end
end
if failureReason = large distance then
    j ← heuristicTimeGap(Y_s)
end
```