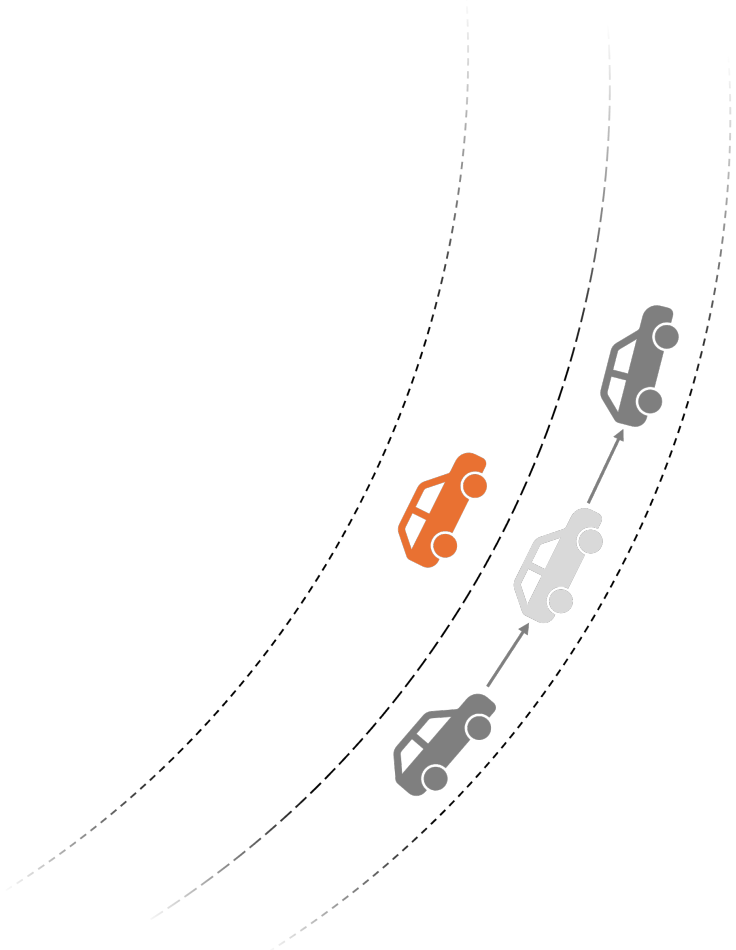
CHALMERS
UNIVERSITY OF TECHNOLOGY



World Model Online Evaluation

An Evaluation of Motion-Based Methods for Detecting Inconsistencies in Dynamic Objects

Master's thesis in Systems, Control and Mechatronics

Alice Dalevi & Tina Sjöberg

DEPARTMENT OF ELECTRICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2026
www.chalmers.se

Master's thesis 2026

World Model Online Evaluation

An Evaluation of Motion-Based Methods for Detecting Inconsistencies in Dynamic Objects

Alice Dalevi & Tina Sjöberg

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Division of Signal Processing and Biomedical Engineering
Chalmers University of Technology
Gothenburg, Sweden 2026

World Model Online Evaluation
An Evaluation of Motion-Based Methods for Detecting Inconsistencies in Dynamic Objects
Alice Dalevi & Tina Sjöberg

Supervisor: Germán Diez, Sven Eriksson, Zenseact
Examiner: Lars Hammarstrand, Department of Electrical Engineering

Master's Thesis 2026
Department of Electrical Engineering
Division of Signal Processing and Biomedical Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: A vehicle trajectory with an inconsistency at one sample. The inconsistency is colored in orange and has an offset from the true sample.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2026

World Model Online Evaluation
Alice Dalevi & Tina Sjöberg
Department of Electrical Engineering
Chalmers University of Technology

Abstract

To further advance the development of Advanced Driver Assistance Systems (ADAS), ensuring a robust learning-based perception system is essential. One approach to achieving this is the collection of diverse and informative data for training perception models. Active learning is based on the principle that exposing models to previously unseen or challenging data improves their performance.

To facilitate this, this thesis evaluates several motion-based methods for detecting inconsistencies in the behavior of perception systems. These methods aim to identify scenarios in which the system performs poorly, enabling targeted data collection. The results show that, for a learning-based perception system, more complex methods generally yield better performance. In particular, Factor Graph Optimization with a Coordinated Turn motion model demonstrates the greatest potential, which may be attributed to the complex and dynamic behavior of objects in traffic environments.

Furthermore, the findings suggest that future work exploring more advanced motion models, as well as combinations of multiple models, is a promising direction for improving perception system performance.

Keywords: Advanced Driver Assistance Systems (ADAS), Factor Graph Optimization, Inconsistency Detection, Kalman Filter

Acknowledgements

We would like to express our sincere gratitude to Zenseact for providing us with the opportunity to carry out this Master's thesis and for their continuous support throughout the project. We are especially thankful to our supervisors at Zenseact, Sven Eriksson and German Diez, for their guidance, insightful feedback, and encouragement during the course of this work. We also extend our appreciation to our manager, Tobias Karlsson, for his support. Finally, we would like to thank our examiner, Lars Hammarstrand at Chalmers University of Technology, for his guidance and for taking the time to answer our challenging questions.

Alice Dalevi, Tina Sjöberg, Gothenburg, June 2026

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

ADAS	Advanced Driver-Assistance Systems
CKF	Cubature Kalman Filter
CT	Coordinated Turn
CV	Constant Velocity
EKF	Extended Kalman Filter
FGO	Factor Graph Optimization
FN	False Negatives
FP	False Positives
GTSAM	Georgia Tech Smoothing and Mapping
IoU	Intersection over Union
KF	Kalman Filter
LiDAR	Light Detection and Ranging
NIS	Normalized Innovation Squared
OOD	Out-of-Distribution
Radar	Radio Detection and Ranging
RTS	Rauch-Tung-Striebel
TC	Temporal Coherence
TN	True Negatives
TP	True Positive
TTC	Time to collision
UKF	Unscented Kalman Filter

Nomenclature

Below is the nomenclature of indices, sets, parameters, and variables that have been used throughout this thesis.

Indices

i	Index for dynamic objects
k	Index for time-sample of dynamic object
t	Index for time step
j	Index over selected state components, $j \in \mathcal{J}_m^\tau$
m	Index for motion model, $m \in \mathcal{M}$
τ	Index over noise types, $\tau \in \mathcal{T}$
l	Index indicating if the values correspond to ego vehicle or other dynamic objects, $l \in \mathcal{L}$
f	Index of frame of view, $f \in \mathcal{F}$
β	Index of sigma-points

Sets

$\mathcal{J}_m^\tau$	Set of states where noise covariance τ is directly applied for motion model m
$\mathcal{F}$	Set of frames of view, $\mathcal{F} = \{W, E\}$
$\mathcal{M}$	Set of motion models, $\mathcal{M} = \{CV, CT\}$
$\mathcal{L}$	Set of labels, indicating if a variable corresponds to the ego vehicle, or an object it has observed, $\mathcal{L} = \{ego, obj\}$
$\mathcal{T}$	Set of noise types, $\mathcal{T} = \{meas, proc\}$

Parameters

T	Sampling time
M	Number of dynamic objects
N_i	Number of samples of dynamic object i
n	Number of states
σ_j^τ	Standard deviation directly applied on j in the noise covariance matrix indexed by τ
$\boldsymbol{\sigma}$	Vector of all σ_j^τ
$\sigma_{0j}^\tau, \mu_{0j}^\tau$	Standard deviation, respectively mean for the regularizer of σ_j^τ
$\sigma_{j,min}^\tau, \sigma_{j,max}^\tau$	Approximate minimum and maximum value of σ_j^τ
σ_{vx}, σ_{vy}	Standard deviation of velocity in x, receptively y, for objects standing still
ρ	Amount of standard deviation used for IoU
b	Bias term applied to simulate artificial errors
s_p, s_v	Position and velocity offset applied to simulate artificial errors
d_p, d_v	Position and velocity drift term for artificial error modeling
α	Scaling factor applied to d_v and d_p to model drift
$\Psi_{threshold}$	Threshold of inconsistency measure
Ψ_p	The p-th percentile value of Ψ
ν^p,ν^v	Degrees of freedom of Student's t distribution for position- respectively velocity-measurements
$\mathbf{R}^p, \mathbf{R}^v$	Position-, respectively velocity-part of measurement noise covariance
$\tilde{\mathbf{A}}$	Continuous process matrix
$\mathbf{A}_k$	Discrete process matrix at index k.
$\mathbf{Q}_k$	Process noise covariance at index k
$\mathbf{Q}$	Process noise covariance
$\mathbf{R}_k$	Measurement noise covariance at index k
$\mathbf{R}$	Measurement noise covariance
$\mathbf{H}_k$	Measurement matrix at index k
$\mathbf{f}(\cdot), \mathbf{f}'(\cdot)$	Process model and its Jacobian
$\mathbf{h}(\cdot), \mathbf{h}'(\cdot)$	Measurement model and its Jacobian
$\mathbf{NIS}_k$	Normalized Innovation Squared-value at index k

Variables

Ψ	Inconsistency measure
Φ	State-translation matrix from ego-view to world-view
$\mathbf{e}^p_{\mathbf{z}_{k,i}}, \mathbf{e}^v_{\mathbf{z}_{k,i}}$	Difference between measurements and estimates for position, respectively velocity, at time-sample k for object i
$\mathbf{e}_{\mathbf{x}_{k,i}}$	Difference between state k and its predicted state given state $k-1$ for object i
$x_k^{f,l}, y_k^{f,l}$	Position along the x-, respectively y-axis, at index k. Expressed in frame f for l.
$v_{x,k}^{f,l}, v_{y,k}^{f,l}$	Velocity along the x-, respectively y-axis, at index k. Expressed in frame f for l.
$\phi_k^{f,l}$	Yaw expressed in frame f for l, at index k
ω_k^l	Yaw-rate at index k, for l
x_k, y_k	State position along the x-axis, respectively y, at index k
$v_{x,k}, v_{y,k}$	State velocity along the x- respectively y-axis, at index k
ω_k	State yaw-rate, at index k
$\mathbf{x}_k$	State vector at index k
$\mathbf{q}_k$	Process noise at index k
$\mathbf{r}_k$	Measurement noise at index k
$\mathbf{x}(t), \dot{\mathbf{x}}(t)$	State vector and its derivative at time t
$x(t), y(t)$	State position along the x- respectively y-axis, at time t
$v_x(t), v_y(t)$	State velocity along the x- respectively y-axis, at time t
$\omega(t)$	State yaw-rate, at time t
$q_c^v(t), q_c^\omega(t)$	Process noise on velocity and yaw-rate at time t
$\hat{\mathbf{x}}_{k\|k-1}, \hat{\mathbf{x}}_{k\|k}$	State estimate at index k given observation at index $k-1$, prediction, respectively k, update
$\mathbf{P}_{k\|k-1}, \mathbf{P}_{k\|k}$	Covariance at index k, given observation at index $k-1$, prediction, respectively k, update
$\mathbf{K}_k$	Kalman gain matrix at index k
$\mathbf{S_k}$	Innovation covariance at index k
$\mathbf{z}_k$	Measurement vector at index k
$\mathbf{v}_k$	Innovation vector at index k
Δx^+, Δy^+	Displacement in the x- and y-directions, respectively, with added standard deviation.
Δx^-, Δy^-	Displacement in the x- and y-directions, respectively, with subtracted standard deviation.
O_{x^+}, O_{y^+}	Overlap in the x and y directions with added standard deviation.

O_{x^-}, O_{y^-}	Overlap in the x and y directions with subtracted standard deviation.
$\hat{I}_{++}, \hat{I}_{--}$	Predicted intersection with added and subtracted standard deviation based on velocity.
$\hat{I}_{+-}, \hat{I}_{-+}$	Predicted intersection under mixed standard deviation adjustments based on velocity.
$\hat{U}_{++}, \hat{U}_{--}$	Predicted union with added and subtracted standard deviation based on velocity.
$\hat{U}_{+-}, \hat{U}_{-+}$	Predicted union under mixed standard deviation adjustments based on velocity.
$I\hat{o}U_{++}, I\hat{o}U_{--}$	Predicted IoU with added and subtracted standard deviation based on velocity.
$I\hat{o}U_{+-}, I\hat{o}U_{-+}$	Predicted IoU under mixed standard deviation adjustments based on velocity.
I	Intersection computed from observed positions
U	Union computed from observed positions
IoU	IoU computed from observed positions

Contents

List of Figures

List of Tables

1

Introduction

The evolution of Advanced Driver-Assistance Systems (ADAS) toward fully automated driving relies heavily on robust learning-based perception systems [1]. Improvement of the performance and safety of these systems can be significantly supported by training perception models on larger and more diverse datasets [2]. Fleet data collected from customer vehicles operating in real-world environments offers the potential for a substantially richer and more diverse data source. However, indiscriminate collection and processing of this data is impractical, as uploading data streams without a principled selection strategy can lead to excessive communication and storage costs, while providing no guarantee that the collected data is informative. Furthermore, this approach risks over-representing common or already well-understood scenarios, resulting in redundant training data. While data collection can be guided by triggers that capture events of interest, a more principled selection strategy can further improve efficiency by prioritizing the most informative samples.

There are several approaches to developing such selection strategies. For instance, data may be prioritized based on risk-related factors, such as the potential severity of a collision affecting the ego vehicle or surrounding road users, as these are critical for identifying hazardous situations [3]. Alternatively, selection can focus on identifying out-of-distribution (OOD) scenarios, which pose a significant challenge to perception systems and can increase the risk of inaccurate predictions [4].

This thesis, however, focuses on developing and evaluating a method for detecting perception errors and utilizing them for selective data collection. The goal is to identify informative data points that can improve the perception system, particularly by capturing challenging or underrepresented scenarios where the system is more likely to fail, as rare and long-tail events are known to be critical for ensuring robustness in autonomous driving [5].

The proposed approach treats data collection as a selective process in which vehicles continuously buffer both input sensor data and model outputs, while an online monitoring mechanism analyzes the behavior of the perception model to detect inconsistencies, potential errors, or scenarios deemed interesting. When such events are identified, a trigger initiates the selective storage of the corresponding data, ensuring that only relevant and informative samples are collected.

1.1 Background and Methodology

A typical pipeline for automated vehicle systems is structured as illustrated in Figure 1.1, following the approach of [6].

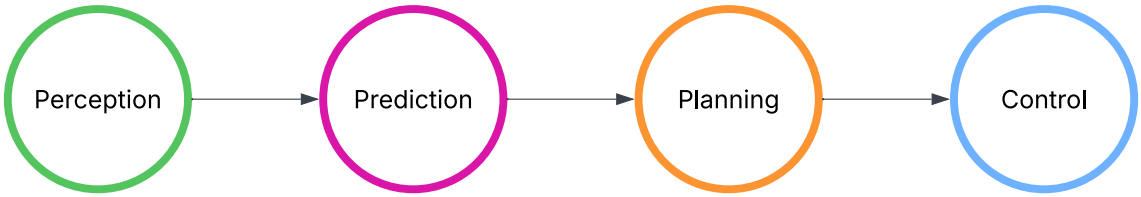


Figure 1.1: Pipeline for general Automated Vehicle System.

Perception: The perception module processes raw data from sensors such as cameras, LiDAR, and radar to estimate the state of the environment. This includes detecting and tracking objects (e.g., cars and pedestrians), estimating their positions and velocities, and understanding scene elements such as lanes and traffic signs.

Prediction: The prediction module estimates the future behavior of dynamic objects in the environment based on the perceived state. For example, it predicts trajectories of surrounding vehicles and pedestrians to anticipate potential interactions.

Planning: The planning module determines the optimal driving strategy and generates a safe and feasible trajectory for the ego vehicle. This includes both high-level decisions (e.g., lane changes, stopping) and low-level motion planning.

Control: The control module executes the planned trajectory by generating actuation commands such as steering, acceleration, and braking.

This thesis focuses specifically on the perception layer, extending existing systems by introducing an online monitoring method that actively analyzes the perception system's outputs in real time. Rather than relying on out-of-distribution (OOD) detection or predefined risk factors, the proposed approach directly examines the system's predictions to identify inconsistencies and potential errors. To this end, multiple methods for detecting anomalous or high-error events are evaluated, with particular emphasis on their ability to capture the rare and long-tail scenarios highlighted in the introduction.

This approach is motivated by two main considerations. First, different types of dynamic objects exhibit distinct behavioral patterns, suggesting that different methods may be more suitable for different object classes. Second, from a practical perspective, simpler methods are often preferable due to their lower computational cost and ease of implementation, especially when more complex methods offer only marginal performance improvements.

Motivated by these considerations, this work adopts a progressive evaluation strategy. It begins with simpler methods as baselines and incrementally introduces more

complex approaches, allowing for a systematic assessment of the trade-offs between performance, complexity, and applicability across different object categories.

1.2 Research Questions

Based on the background presented above, it is necessary to systematically evaluate and compare the effectiveness of different methods for detecting inconsistencies in learning-based perception systems.

To this end, the following research questions are formulated:

1. *What methods can be used to identify, filter, and select informative data from large vehicle fleets?*
2. *Could the proposed methods be implemented in real-time for a given vehicle?*
3. *How do the different methods compare with respect to detection performance and computational complexity, and what trade-offs can be identified between them?*

1.3 Contributions

To address the research questions, this thesis makes the following contributions:

- Implementation of anomaly detection methods for identifying informative data, including approaches based on Temporal Coherence, Kalman Filters and Factor Graph Optimization.
- Evaluation of the computational complexity of the proposed methods and assessment of their suitability for real-time implementation in a vehicle.
- Comparative analysis of the different methods with respect to detection performance, using metrics such as recall, precision, and F_1 score.
- Investigation of method performance across different dynamic object classes, including pedestrians, cars, and trucks.

1.4 Limitations

There are some limitations assumed and placed upon this project, which will affect the result somewhat. First, the datasets collected were limited to areas in and around Gothenburg, as it was most convenient for this thesis. Second, only position and velocity measurements will be evaluated as this ensures a fair comparison between all methods chosen to be evaluated. Third, when dealing with object disappearance and reappearance, all objects exhibiting this behavior will be flagged as interesting. This is done consciously of the fact that some of these instances will be due to object occlusion, and not perception system failure. Separating the two mentioned causes for this occurrence is out of scope for this thesis. Fourth, there are many ways to collect and analyze data from vehicles, this thesis deals mainly with sensor outputs and will not examine direct video frames. Fifth, this thesis focuses on identifying anomalous behavior in dynamic objects and does not consider static elements such as

traffic signs or road infrastructure. Finally, since the data collection was conducted primarily on highways, certain dynamic object types (e.g., cars) are significantly more prevalent than others, such as pedestrians. To account for this imbalance, each dynamic object category is evaluated separately. This thesis focuses on three types of dynamic objects: cars, trucks, and pedestrians.

2

Theory

This chapter provides the theoretical foundation for the thesis. It begins with a brief overview of the use of fleet data and the principles of active learning. Subsequently, it introduces the key concepts and methodologies necessary for understanding and evaluating the approaches proposed in this work.

2.1 Fleet Data and Active Learning Overview

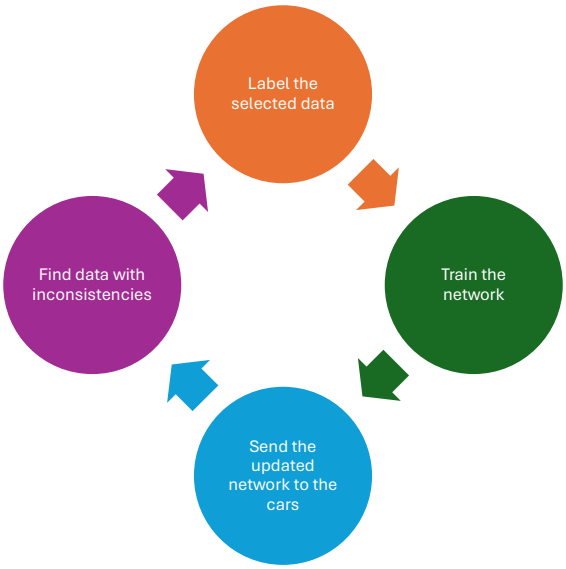


Figure 2.1: Pipeline of how the data collection could be used in a finished product. This project focuses on the purple circle, the detection of inconsistencies.

The development of autonomous driving relies on a closed-loop pipeline similar to the one depicted in Figure 2.1, that encompasses data collection, labeling, model training, simulation, real-world testing, and final deployment [1]. As this thesis focuses on optimizing data collection, it is essential to first examine the mechanisms and advantages of fleet-scale data acquisition. In addition, active learning is introduced to highlight the importance of well-formed data selection strategies.

2.1.1 Fleet data

Datasets derived from vehicle fleets, hereafter referred to as *fleet data*, can significantly contribute to the training and validation of 3D object detection models. While individual customer vehicles often lack advanced data logging capabilities and high-quality sensor configurations, for example, specialized cameras and perception systems, fleet-based data collection provides access to a substantially larger volume of data. This increased scale may help improve model robustness and generalization [2].

Fleet-scale datasets vary in structure and sensor composition. Many consist primarily of video sequences [1], while others adopt a multimodal approach by combining camera and LiDAR data [2]. Although such datasets can support a wide range of autonomous driving tasks, including prediction and planning, this work focuses exclusively on perception data.

A primary challenge in real-world data acquisition is the inherent redundancy of collected data: the majority of recordings represent routine driving scenarios, while so-called corner cases or out-of-distribution (OOD) events remain rare [1]. The importance of large and well-balanced datasets is highlighted in [7], which shows that 3D detection models trained on limited or biased data experience significant performance degradation when evaluated on more balanced and diverse test sets. Additionally, a large-scale analysis of 103 datasets reveals a systematic lack of geographical diversity [8], further emphasizing limitations in current data collection practices.

Taken together, these challenges underscore the need for large-scale and diverse datasets for training learning-based perception systems, particularly those enriched with rare and atypical events. Such diversity is more readily achieved through fleet-based data collection, where the aggregation of data across many vehicles, locations, and driving conditions increases the likelihood of capturing rare scenarios and reducing dataset bias.

2.1.2 Active Learning

Active learning is a method used in machine learning, where only a subset of the training data is required to be labeled. In general, the purpose of using this method is to obtain a good result, while not being required to label all of the data, which could save a lot of time and computational power. The active learning algorithm follows the pipeline shown in Figure 2.2. The general idea is that a subset of the training data is selected, which is then labeled and used by the model to train on. From the result of running the data through the model, additional samples are selected based on an acquisition function and then labeled. This is then performed multiple times until a desired performance is reached. The acquisition function is a metric to select data that may provide additional information to the classifier and can be defined in multiple ways [9].

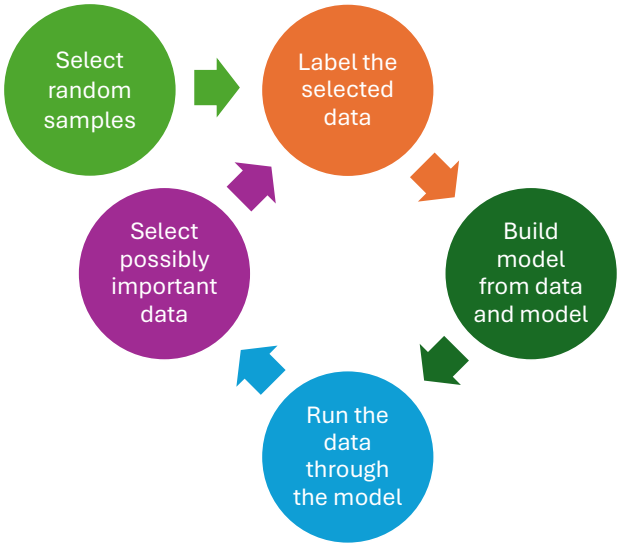


Figure 2.2: Active learning pipeline.

The methods developed and evaluated in this thesis fall within the purple circle in the pipeline depicted in Figure 2.2. These methods focus on determining when to trigger data collection, similar to how acquisition functions in active learning decide which data points are most beneficial to annotate.

In this sense, the active learning pipeline closely aligns with the objective of this thesis: identifying the most informative data for training perception models in order to improve their performance.

2.2 Temporal Coherence

Temporal Coherence in this context generally refers to the coherence between video frames, that is, for two frames next to each other in time, the expectation is that they contain essentially the same objects [10]. This is useful to determine when objects, people, or the general surroundings are acting in a way that is highly unlikely, based on the previous frame. By identifying object frames, or bounding boxes, and mapping boxes between video frames, the overlap between them can be calculated [10].

Intersection over Union (IoU) is a widely used metric in object detection for quantifying the overlap between two regions. It is defined as the ratio between the area of intersection and the area of union of two shapes A and B, as described in [11]:

$$IoU = \frac{|A \cap B|}{|A \cup B|}. \tag{2.1}$$

In the context of object detection, IoU is commonly used to compare a predicted bounding box B with the corresponding ground truth bounding box B^{gt}. As described by Hao Zhang et al. [12], this can be expressed as:

$$IoU = \frac{|B \cap B^{gt}|}{|B \cup B^{gt}|}. \tag{2.2}$$

The resulting IoU value lies in the range $[0, 1]$, where a value of 0 indicates no overlap between the bounding boxes, and a value of 1 corresponds to a perfect match. Consequently, IoU provides an intuitive and standardized measure for evaluating the accuracy of object localization.

2.3 Kalman Filters

Described by Riberio in [13], "the Kalman filter is a linear, discrete time, finite-dimensional time-varying system that evaluates the state estimate that minimizes the mean-square error". In other words, the Kalman filter takes as input previously estimated states, for instance, position and velocity, and compares them to measurement data to accurately estimate the next states. Often the method is divided into a prediction and an update step, in [13] the update step is referred to as a filtering step. Furthermore, the Kalman filter is often described as a Bayesian filter [14], since it estimates the state of the system in a probabilistic framework.

2.3.1 Simple Kalman Filter

The prediction equations use solely previous data, or priors [15], to estimate the next state at time k, without a current observation.

$$\hat{\mathbf{x}}_{k|k-1} = \mathbf{A}_{k-1}\hat{\mathbf{x}}_{k-1|k-1} \tag{2.3}$$

$$\mathbf{P}_{k|k-1} = \mathbf{A}_{k-1}\mathbf{P}_{k-1|k-1}\mathbf{A}_{k-1}^T + \mathbf{Q}_{k-1}, \tag{2.4}$$

where

- $\hat{\mathbf{x}}_{k|k-1}$ is the predicted next state and $\hat{\mathbf{x}}_{k-1|k-1}$ is the previously estimated state.
- $\mathbf{P}_{k|k-1}$ is the expected error covariance, and $\mathbf{P}_{k-1|k-1}$ is the prior expected error covariance.
- $\mathbf{A}_{k-1}$ is the state transition matrix, describing the relationship between states at time k and $k+1$.
- $\mathbf{Q}_{k-1}$ describes the uncertainty in the state transitions over time.

The update equations combine the currently predicted state and covariance from the prediction step, with the observed measurements at time k,

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k\mathbf{v}_k \tag{2.5}$$

$$\mathbf{P}_{k|k} = \mathbf{P}_{k|k-1} - \mathbf{K}_k\mathbf{S}_k\mathbf{K}_k^T, \tag{2.6}$$

where

$$\mathbf{K}_k = \mathbf{P}_{k|k-1}\mathbf{H}_k^T\mathbf{S}_k^{-1} \tag{2.7}$$

$$\mathbf{S}_k = \mathbf{H}_k\mathbf{P}_{k|k-1}\mathbf{H}_k^T + \mathbf{R}_k \tag{2.8}$$

$$\mathbf{v}_k = \mathbf{z}_k - \mathbf{H}_k\hat{\mathbf{x}}_{k|k-1} \tag{2.9}$$

Here

- $\hat{\mathbf{x}}_{k|k}$ Updated estimated next state [15].
- $\mathbf{P}_{k|k}$ updated error covariance.
- $\mathbf{K}_k$ is the Kalman Gain matrix and describes how the observed measurements affect the system state estimations [15].
- $\mathbf{v}_k$ is the innovation residual [15], which calculates the difference between the estimated state and the actual measured data.
- $\mathbf{S}_k$ is the innovation covariance [15].

- $\mathbf{z}_k$ is the measurement.

2.3.2 Extended Kalman Filter

As mentioned above, the Kalman filter is a linear system. However, there are many instances where the state dynamics are non-linear. In those cases, the Extended Kalman Filter, EKF, can be used instead [13]. Here, both the system dynamics and the observation dynamics need to be linearized around $\hat{\mathbf{x}}_{k-1|k-1}$, respectively $\hat{\mathbf{x}}_{k|k-1}$ [13], to be able to apply the linear Kalman filter.

Then the prediction step becomes

$$\hat{\mathbf{x}}_{k|k-1} = \mathbf{f}(\hat{\mathbf{x}}_{k-1|k-1}) \tag{2.10}$$
$$\mathbf{P}_{k|k-1} = \mathbf{f}'(\hat{\mathbf{x}}_{k-1|k-1})\mathbf{P}_{k-1|k-1}\mathbf{f}'(\hat{\mathbf{x}}_{k-1|k-1})^T + \mathbf{Q}_{k-1}, \tag{2.11}$$

where $\mathbf{f}'(\hat{\mathbf{x}}_{k-1|k-1})$ is the Jacobian of $\mathbf{f}(\hat{\mathbf{x}}_{k-1|k-1})$, the motion model. Furthermore, the update step becomes

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k\mathbf{v}_k \tag{2.12}$$
$$\mathbf{P}_{k|k} = \mathbf{P}_{k|k-1} - \mathbf{K}_k\mathbf{S}_k\mathbf{K}_k^T \tag{2.13}$$
$$\mathbf{K}_k = \mathbf{P}_{k|k-1}\mathbf{h}'(\hat{\mathbf{x}}_{k|k-1})^T\mathbf{S}_k^{-1} \tag{2.14}$$
$$\mathbf{S}_k = \mathbf{h}'(\hat{\mathbf{x}}_{k|k-1})\mathbf{P}_{k|k-1}\mathbf{h}'(\hat{\mathbf{x}}_{k|k-1})^T + \mathbf{R}_k \tag{2.15}$$
$$\mathbf{v}_k = \mathbf{z}_k - \mathbf{h}(\hat{\mathbf{x}}_{k|k-1}), \tag{2.16}$$

where $\mathbf{h}'(\hat{\mathbf{x}}_{k|k-1})$ is the Jacobian of $\mathbf{h}(\hat{\mathbf{x}}_{k|k-1})$, the measurement model.

2.3.3 Cubature Kalman Filter

While the Extended Kalman Filter performs better than the standard Kalman filter for nonlinear systems, it requires linearization, which may result in a poor approximation of the system for highly nonlinear systems. To better handle the nonlinear systems, sigma-point methods have been introduced, such as the Unscented Kalman Filter (UKF) [16] and the Cubature Kalman Filter (CKF) [17]. These types of filters do not use linearization to calculate estimated measurements or predict the movement, but select multiple points close to the estimate and propagate them through the motion, respectively, the measurement model, and calculates the estimate and its covariance based on the distribution of the propagated points.

The two mentioned sigma-point methods differ slightly, with the UKF using one sigma-point more and also having design parameters, which control how the sigma-points are spread [16]. The UKF can, however, get negative weights, which can result in a covariance matrix that is not positive definite, leading to instability of the filter [17]. Due to the instability of the UKF, only the CKF is evaluated in this project.

For the prediction step, the sigma-points for CKF are calculated accordingly

$$\begin{aligned} \mathcal{X}_{k-1}^{(\beta)} &= \hat{\mathbf{x}}_{k-1|k-1} + \sqrt{n}(\mathbf{P}_{k-1|k-1}^{1/2})_\beta \\ \mathcal{X}_{k-1}^{(\beta+n)} &= \hat{\mathbf{x}}_{k-1|k-1} - \sqrt{n}(\mathbf{P}_{k-1|k-1}^{1/2})_\beta, \\ W_\beta &= \frac{1}{2n} \end{aligned} \tag{2.17}$$

where $(P_{k-1|k-1}^{1/2})_\beta$ corresponds to column β of $P_{k-1|k-1}^{1/2}$ and n is the number of states.

The predicted state and covariance are estimated to be the following

$$\begin{aligned} \hat{\mathbf{x}}_{k|k-1} &\approx \sum_{\beta=1}^{2n} \mathbf{f}(\mathcal{X}_{k-1}^{(\beta)}) W_\beta \\ \mathbf{P}_{k|k-1} &\approx \mathbf{Q}_{k-1} + \sum_{\beta=1}^{2n} (\mathbf{f}(\mathcal{X}_{k-1}^{(\beta)}) - \hat{\mathbf{x}}_{k|k-1})(\cdot)^\top W_\beta. \end{aligned} \tag{2.18}$$

For the update step, the sigma-points are calculated as

$$\begin{aligned} \mathcal{X}_{k}^{(\beta)} &= \hat{\mathbf{x}}_{k|k-1} + \sqrt{n}(\mathbf{P}_{k|k-1}^{1/2})_\beta \\ \mathcal{X}_{k}^{(\beta+n)} &= \hat{\mathbf{x}}_{k|k-1} - \sqrt{n}(\mathbf{P}_{k|k-1}^{1/2})_\beta \\ W_\beta &= \frac{1}{2n}. \end{aligned} \tag{2.19}$$

The update is performed

$$\begin{aligned} \hat{\mathbf{x}}_{k|k} &= \hat{\mathbf{x}}_{k|k-1} + \mathbf{P}_{xy}\mathbf{S}_k^{-1}(\mathbf{z}_k - \hat{\mathbf{z}}_{k|k-1})\mathbf{P}_{k|k} \\ \mathbf{P}_{k|k} &= \mathbf{P}_{k|k-1} - \mathbf{P}_{xy}\mathbf{S}_k^{-1}\mathbf{P}_{xy}^\top, \end{aligned} \tag{2.20}$$

where

$$\begin{aligned} \hat{\mathbf{z}}_{k|k-1} &\approx \sum_{\beta=1}^{2n} \mathbf{h}(\mathcal{X}_{k}^{(\beta)}) W_\beta \\ \mathbf{P}_{xy} &\approx \sum_{\beta=1}^{2n} (\mathcal{X}_{k}^{(\beta)} - \hat{\mathbf{x}}_{k|k-1})(\mathbf{h}(\mathcal{X}_{k}^{(\beta)}) - \hat{\mathbf{z}}_{k|k-1})^\top W_\beta \\ \mathbf{S}_k &\approx \mathbf{R}_k + \sum_{\beta=1}^{2n} (\mathbf{h}(\mathcal{X}_{k}^{(\beta)}) - \hat{\mathbf{z}}_{k|k-1})(\cdot)^\top W_\beta. \end{aligned} \tag{2.21}$$

2.4 Smoothers

As previously described, the Kalman filters perform their calculations as a forward pass, only using the past measurements to estimate the states, and not the future ones. For real-time applications, it is only possible to use the past measurements, but for cases where it is allowed to have some delay, a smoother, which also uses the future data, could be used. One of the most common smoothers for Kalman Filters is the Rauch-Tung-Striebel (RTS) Smoother [18], which is performed as a backward pass.
The regular RTS smoother is defined as

$$\begin{aligned}\mathbf{G}_k &= \mathbf{P}_{k|k}\mathbf{A}_k^\top \mathbf{P}_{k+1|k}^{-1} \\ \hat{\mathbf{x}}_{k|K} &= \hat{\mathbf{x}}_{k|k} + \mathbf{G}_k(\hat{\mathbf{x}}_{k+1|K} - \hat{\mathbf{x}}_{k+1|k}) \\ \mathbf{P}_{k|K} &= \mathbf{P}_{k|k} - \mathbf{G}_k(\mathbf{P}_{k+1|k} - \mathbf{P}_{k+1|K})\mathbf{G}_k^\top,\end{aligned} \tag{2.22}$$

where $\hat{\mathbf{x}}_{k|K}$ and $\mathbf{P}_{k|K}$ are the smoothed state estimate, respectively, the smoothed covariance at sample k given measurements to timestep K. The backward pass is started from the last time step K, with the smoothed state, $\hat{\mathbf{x}}_{K|K}$, being equal to the last filtered state [14].

2.4.1 Extended RTS-smoother

For the EKF, the RTS smoother is defined similarly to the regular RTS [14]

$$\begin{aligned}\mathbf{G}_k &= \mathbf{P}_{k|k}\mathbf{f}'(\hat{\mathbf{x}}_{k|k})^\top \mathbf{P}_{k+1|k}^{-1} \\ \hat{\mathbf{x}}_{k|K} &= \hat{\mathbf{x}}_{k|k} + \mathbf{G}_k(\hat{\mathbf{x}}_{k+1|K} - \mathbf{f}(\hat{\mathbf{x}}_{k|k})) \\ \mathbf{P}_{k|K} &= \mathbf{P}_{k|k} - \mathbf{G}_k(\mathbf{P}_{k+1|k} - \mathbf{P}_{k+1|K})\mathbf{G}_k^\top.\end{aligned} \tag{2.23}$$

2.4.2 Cubature RTS Smoothing

The RTS smoother for CKF follows a similar structure as the above defined smoothers, but uses sigma-points similarly to how they were used in the filter [14]

$$\begin{aligned}\mathcal{X}_k^{(\beta)} &= \hat{\mathbf{x}}_{k|k} + \sqrt{n}(\mathbf{P}_{k|k}^{1/2})_\beta \\ \mathcal{X}_k^{(\beta+n)} &= \hat{\mathbf{x}}_{k|k} - \sqrt{n}(\mathbf{P}_{k|k}^{1/2})_\beta \\ W_\beta &= \frac{1}{2n}.\end{aligned} \tag{2.24}$$

The smoothing is performed accordingly

$$\begin{aligned}\mathbf{G}_k &= \mathbf{P}_{k,k+1|k}\mathbf{P}_{k+1|k}^{-1} \\ \hat{\mathbf{x}}_{k|K} &= \hat{\mathbf{x}}_{k|k} - \mathbf{G}_k(\hat{\mathbf{x}}_{k+1|K} - \hat{\mathbf{x}}_{k+1|k})) \\ \mathbf{P}_{k|K} &= \mathbf{P}_{k|k} - \mathbf{G}_k(\mathbf{P}_{k+1|k} - \mathbf{P}_{k|K})\mathbf{G}_k^\top,\end{aligned} \tag{2.25}$$

where

$$\begin{aligned}\hat{\mathbf{x}}_{k+1|k} &\approx \sum_{\beta=1}^{2n} \mathbf{f}(\mathcal{X}_k^{(\beta)}) W_\beta \\ \mathbf{P}_{k+1|k} &\approx \mathbf{Q}_k + \sum_{\beta=1}^{2n} (\mathbf{f}(\mathcal{X}_k^{(\beta)}) - \hat{\mathbf{x}}_{k+1|k})(\cdot)^\top W_\beta \\ \mathbf{P}_{k,k+1|k} &\approx \sum_{\beta=1}^{2n} (\mathcal{X}_k^{(\beta)} - \hat{\mathbf{x}}_{k|k})(\mathbf{f}(\mathcal{X}_k^{(\beta)}) - \hat{\mathbf{x}}_{k+1|k})^\top W_\beta.\end{aligned} \tag{2.26}$$

2.5 Motion Models

The Kalman filter requires, as mentioned above, a state transition matrix, an observation matrix, and the covariance matrices for the state transition error and the measurement error [15]. To acquire these, motion models describing the trajectory of the states are used. For this master's thesis, the constant velocity and the coordinated turn model were both relevant. Measurements are also limited to position and velocity to ensure a fair comparison between methods later on.

2.5.1 Constant Velocity

The constant velocity model assumes that the only states changing with time are the position, with velocity as a constant. The continuous time model is described as

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \dot{v_x}(t) \\ \dot{v_y}(t) \end{bmatrix} = \tilde{\mathbf{A}}\mathbf{x}(t) = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x(t) \\ y(t) \\ v_x(t) \\ v_y(t) \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} q_c^v(t). \tag{2.27}$$

To use the Kalman filter, the motion model needs to be discretized, here, the Euler method is used

$$\mathbf{x}_k = \mathbf{A}_{k-1}\mathbf{x}_{k-1} + \mathbf{q}_{k-1}, \tag{2.28}$$

where the transition matrix $\mathbf{A}_{k-1}$ calculated using a first-order Taylor expansion. The calculation is performed accordingly

$$\mathbf{A}_{k-1} = \mathbf{I} + T\tilde{\mathbf{A}} = \tag{2.29}$$

$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} + T \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{2.30}$$

Together

$$\mathbf{x}_k = \begin{bmatrix} x_k \\ y_k \\ v_{x,k} \\ v_{y,k} \end{bmatrix} = \begin{bmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{k-1} \\ y_{k-1} \\ v_{x,k-1} \\ v_{y,k-1} \end{bmatrix} + \mathbf{q}_{k-1}, \tag{2.31}$$

where $\mathbf{q}_{k-1} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_{k-1})$ and the process noise covariance matrix [19]

$$\mathbf{Q}_{k-1} = \sigma_v^2 \begin{bmatrix} T^3/3 \cdot \mathbf{I}_{2\times 2} & T^2/2 \cdot \mathbf{I}_{2\times 2} \\ T^2/2 \cdot \mathbf{I}_{2\times 2} & T \cdot \mathbf{I}_{2\times 2} \end{bmatrix}, \tag{2.32}$$

2.5.2 Coordinated Turn

The coordinated turn model is generally used to model a vehicle in a curve, which is very useful when estimating a vehicle's trajectory on different roads over time. The continuous time model, using Cartesian coordinates, is written as

$$\begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \dot{v}_x(t) \\ \dot{v}_y(t) \\ \dot{\omega}(t) \end{bmatrix} = \begin{bmatrix} v_x(t) \\ v_y(t) \\ -v_y(t)\omega(t) \\ v_x(t)\omega(t) \\ \omega(t) \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} q_c^v(t) \\ q_c^\omega(t) \end{bmatrix}. \tag{2.33}$$

Discretized using the Euler method:

$$\mathbf{x}_k = \mathbf{f}(\mathbf{x}_{k-1}) + \mathbf{q}_{k-1} \approx \mathbf{x}_{k-1} + T\tilde{\mathbf{a}}(\mathbf{x}_{k-1}) + \mathbf{q}_{k-1} \tag{2.34}$$

$$\Rightarrow \begin{bmatrix} x_k \\ y_k \\ v_{x,k} \\ v_{y,k} \\ \omega_k \end{bmatrix} = \begin{bmatrix} x_{k-1} + T \cdot v_{x,k-1} \\ y_{k-1} + T \cdot v_{y,k-1} \\ v_{x,k-1} - T \cdot \omega v_{y,k-1} \\ v_{y,k-1} + T \cdot \omega v_{x,k-1} \\ \omega_{k-1} \end{bmatrix} + \mathbf{q}_{k-1}, \tag{2.35}$$

where $\mathbf{q}_{k-1} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_{k-1})$. From the above definition, the Jacobian is calculated as

$$\mathbf{f}'(\mathbf{x}_{k-1}) = \begin{bmatrix} 1 & 0 & T & 0 & 0 \\ 0 & 1 & 0 & T & 0 \\ 0 & 0 & 1 & -T\omega_{k-1} & -Tv_{y,k-1} \\ 0 & 0 & T\omega_{k-1} & 1 & Tv_{x,k-1} \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \tag{2.36}$$

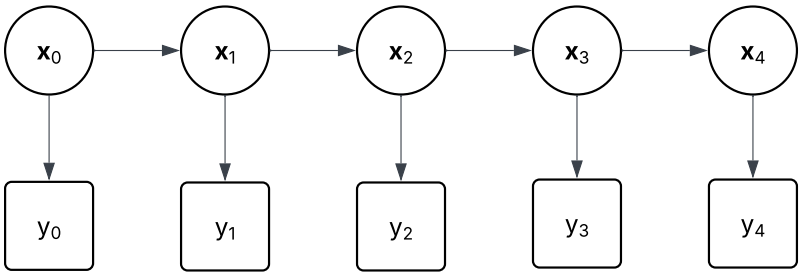
and the process noise covariance matrix [14] as

$$\mathbf{Q}_{k-1} = diag\Big(\Big[\frac{T^2}{2}\sigma_v^2, \frac{T^2}{2}\sigma_v^2, T\sigma_v^2, T\sigma_v^2, T\sigma_w^2\Big]\Big). \tag{2.37}$$

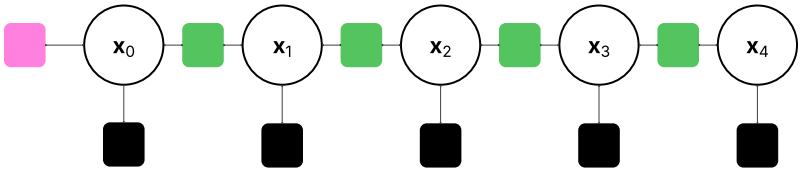
2.6 Factor Graph Optimization (FGO)

Factor Graphs are based on Bayesian networks, with some modifications. Figure 2.3 illustrates the difference between a Bayesian network and the corresponding factor graph. The Bayesian network is constructed with nodes for both states and measurements, and has directed edges symbolizing the dependencies between states, for example, with a motion model. The factor graph, however, is constructed with variable nodes for each variable, the states, and factor nodes, which are functions describing how the states are connected or constrain them in other ways [20]. The

connection between the states could be a motion model, and other constraints can be a prior on a state or measurements and their accuracy. Since the factors are distinctly defined, the only unknowns for the graph are the variables, the states.



(a) Bayesian network.

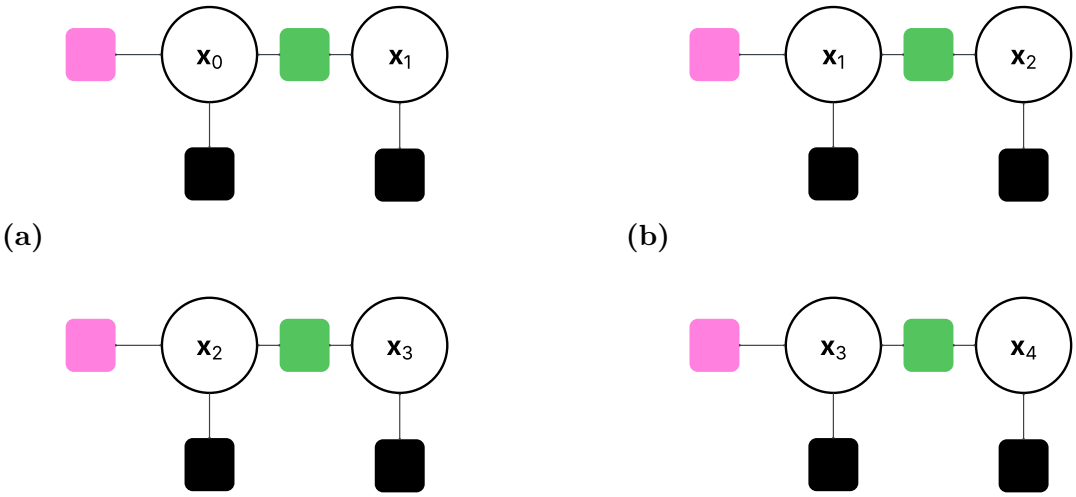


(b) Factor graph, where the black, green, and pink squares symbolize measurement-, process-, respectively prior factors.

Figure 2.3: Comparison of a Bayesian network and a Factor graph.

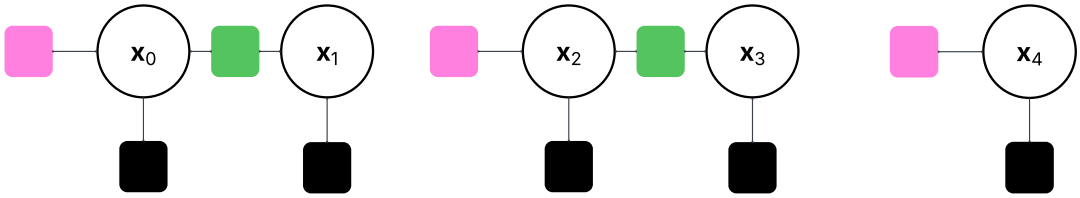
Factor Graph Optimization is a method that, similarly to Kalman filters, is based on Bayesian networks, with the major difference being that the optimization is performed on all states simultaneously, while the Kalman filter and its smoothing are performed as a forward-, respectively backward pass [21]. The FGO can be constructed similarly to the Kalman filter, using different motion models such as constant velocity and coordinated turn.

Previous studies that compared EKF, both with [22] and without smoothing [23], and FGO, have shown that the FGO in general performs better in terms of accuracy, while it requires larger computational time. The large computational time is a result of performing an optimization over all the historical data, but can be lowered if a sliding window is used [23]. In addition, [24] has shown that a larger window for FGO, in general, provides improvement of accuracy, although the gain becomes negligible above a certain size. When using the sliding window approach, it could be performed by using only the measurements inside the window. Another approach is to marginalize the old data and use it as a prior for the next window [21], as shown in Figure 2.4. The latter approach has in a previous study [25], resulted in a smaller error, especially for smaller window-sizes, in addition to slightly lower computational time, probably due to the prior bringing the optimizer closer to the optimum, which results in fewer optimization-iterations required. In addition, a tumbling window version is provided in Figure 2.5, where samples are only optimized over once, and then kept for the next window only as a prior through marginalization.



(a) (b) (c) (d)

Figure 2.4: Illustration of sliding window on a factor graph, with window size 2, where marginalization is performed between each figure and is included as a prior in the next graph. The black, green, and pink squares symbolize measurement, process, and prior factors.



(a) (b) (c)

Figure 2.5: Illustration of tumbling window on a factor graph, with window size 2, where marginalization is performed between each figure and is included as a prior in the next graph. The black, green, and pink squares symbolize measurement, process, and prior factors.

2.7 Bayesian Optimization

Bayesian Optimization is a derivative-free global optimization algorithm that is particularly useful for black-box objectives. The algorithm has been used for a long time, but has in modern times been used as a tool for hyperparameter tuning for machine learning. The algorithm is constructed such that it builds a surrogate function of the cost function, based on the parameter values it has already tested. It also constructs an acquisition function based on the surrogate function and its estimated uncertainty, such that it has larger values where the surrogate function is less certain about the behavior of the objective and where it has a larger probability of improvement. The acquisition function can be constructed in multiple different ways, with different amounts of focus on exploration and exploitation [26]

An example of the algorithm is illustrated in Figure 2.6, where each row of sub-figures shows the true and surrogate function, and the acquisition function given the surrogate function. The parameters corresponding to the largest acquisition function value are selected for the next iteration, as depicted. The figure illustrates that the acquisition function sometimes selects hyperparameters where the surrogate function has the largest uncertainty and sometimes where it expects more improvement.

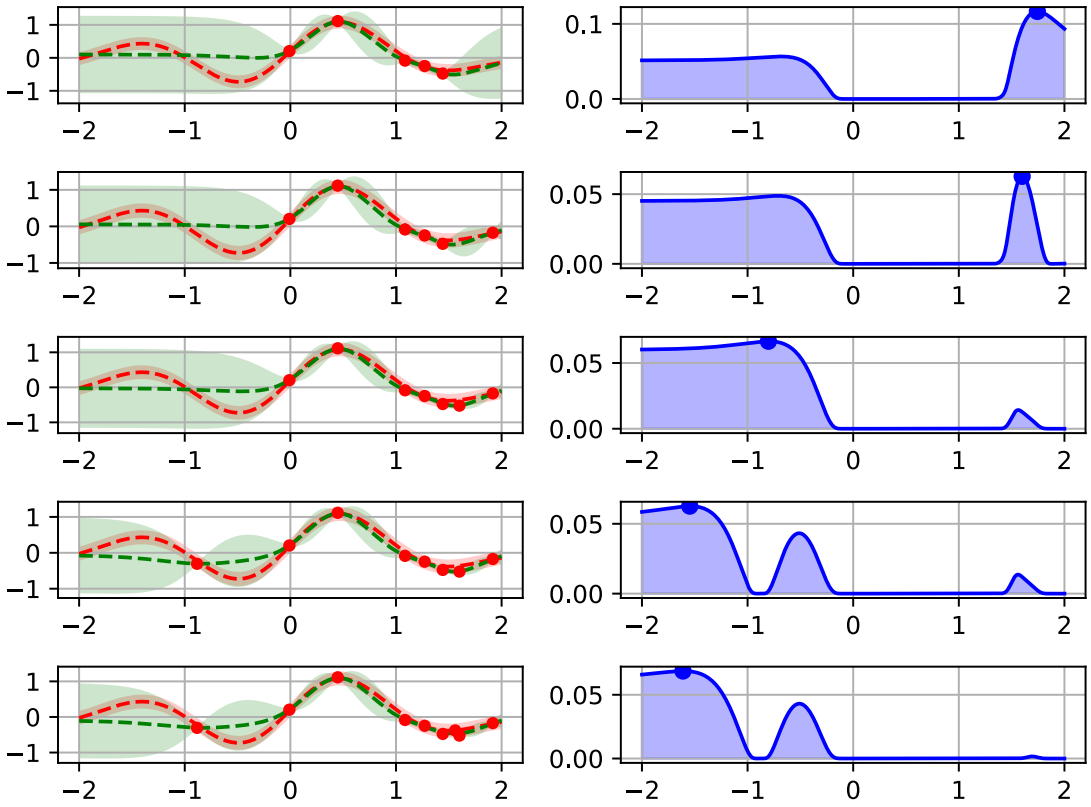


Figure 2.6: Example of Bayesian optimization, to the left, true function (red) and estimated function (green), to the right, the acquisition function. The red dots represent the selected parameters that have been tested, and the blue the parameter that should be tested next. The acquisition function GP-hedge [27] was used. The Bayesian optimization was performed with the scikit-optimize function `gp_minimize` [28].

3
Methods

The Methods chapter describes the processing of raw data and the application of multiple evaluation methods. It begins with simple approaches for detecting inconsistencies in class labels, disappearance and reappearance events, and motion. Furthermore, more advanced motion inconsistency methods based on Kalman filters and Factor Graph Optimization are introduced. Finally, the chapter outlines how the different methods are evaluated.

3.1 World Frame Definition

The International Organization for Standardization (ISO), in its standard for road vehicle dynamics (ISO 8855), defines a reference frame as a "geometric environment in which all points remain fixed with respect to each other at all times" [29]. This definition is adopted in this work, as it is specifically intended for automotive applications. According to this standard, the reference frame consists of three orthogonal axes, X, Y, and Z, defined using the right-hand rule such that

$$\vec{Z} = \vec{X} \times \vec{Y}. \tag{3.1}$$

The vehicle coordinate system is defined with the x-axis pointing forward in the direction of travel at standstill, the y-axis oriented horizontally and pointing to the left in relation to the direction of the vehicle, and the z-axis pointing upward [29].

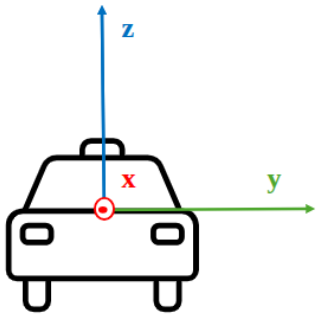


Figure 3.1: Coordinate Frame, front view.

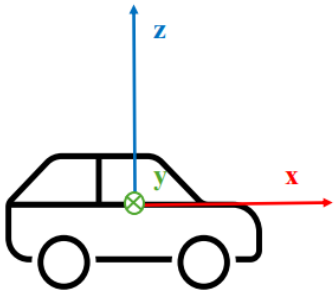


Figure 3.2: Coordinate Frame, side view.

To properly estimate not only the dynamic objects' position and rotation in relation to a world frame, but also the vehicle's position and rotation, the coordinate frame is "frozen". By dictating that the current coordinate frame is the world frame as

$$x = 0 \tag{3.2}$$
$$y = 0 \tag{3.3}$$
$$\phi = 0. \tag{3.4}$$

The homogeneous transformation matrix is computed at each timestep and can be reset, although this was not done in this thesis, to avoid accumulated drift. The transformation matrix is subsequently used within a state-space model to estimate the vehicle's trajectory from the recorded data. Dynamic objects observed during data logging are then transformed into the world reference frame in a consistent manner. The associated calculations are presented below.

3.1.1 Estimation of ego vehicle in world frame

With the initial position of the ego vehicle set to the origin in the world frame, and with the yaw set to 0, the following positions can be expressed using Euler

$$\begin{bmatrix} x_k^{W,ego} \\ y_k^{W,ego} \end{bmatrix} = \begin{bmatrix} x_{k-1}^{W,ego} \\ y_{k-1}^{W,ego} \end{bmatrix} + T \begin{bmatrix} v_{x,k-1}^{W,ego} \\ v_{y,k-1}^{W,ego} \end{bmatrix} \tag{3.5}$$
$$\phi_k^{W,ego} = \phi_{k-1}^{W,ego} + T\omega_{k-1}^{ego}. \tag{3.6}$$

The velocity of the ego vehicle is equal in magnitude in the coordinate systems, but not in rotation. To account for the rotation, the velocity in the world frame is defined as

$$\begin{bmatrix} v_{x,k-1}^{W,ego} \\ v_{y,k-1}^{W,ego} \end{bmatrix} = R(\phi_{k-1}^{W,ego}) \begin{bmatrix} v_{x,k-1}^{E,ego} \\ v_{y,k-1}^{E,ego} \end{bmatrix} \tag{3.7}$$
$$R(\phi) = \begin{bmatrix} \cos(\phi) & -\sin(\phi) \\ \sin(\phi) & \cos(\phi) \end{bmatrix}. \tag{3.8}$$

Resulting in the following state-space

$$\begin{bmatrix} x_k^{W,ego} \\ y_k^{W,ego} \\ \phi_k^{W,ego} \end{bmatrix} = \mathbf{I}_3 \begin{bmatrix} x_{k-1}^{W,ego} \\ y_{k-1}^{W,ego} \\ \phi_{k-1}^{W,ego} \end{bmatrix} + T \begin{bmatrix} R(\phi_{k-1}^{W,ego}) & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} v_{x,k-1}^{E,ego} \\ v_{y,k-1}^{E,ego} \\ \omega_{k-1}^{ego} \end{bmatrix}. \tag{3.9}$$

3.1.2 Estimation of dynamical objects in world frame

The positioning of the dynamic objects in the world frame is described with both rotation and translation, while velocity only requires rotation

$$\begin{bmatrix} x_k^{W,obj} \\ y_k^{W,obj} \end{bmatrix} = R(\phi_k^{W,ego}) \begin{bmatrix} x_k^{E,obj} \\ y_k^{E,obj} \end{bmatrix} + \begin{bmatrix} x_k^{W,ego} \\ y_k^{W,ego} \end{bmatrix} \tag{3.10}$$

$$\begin{bmatrix} v_{x,k}^{W,obj} \\ v_{y,k}^{W,obj} \end{bmatrix} = R(\phi_k^{W,ego}) \begin{bmatrix} v_{x,k}^{E,obj} \\ v_{y,k}^{E,obj} \end{bmatrix}. \tag{3.11}$$

The yaw of the object in the world frame

$$\phi_k^{W,obj} = \phi_k^{E,obj} + \phi_k^{W,ego}. \tag{3.12}$$

The state-space for the dynamical objects is thus

$$\begin{bmatrix} x_k^{W,obj} \\ y_k^{W,obj} \\ v_{x,k}^{W,obj} \\ v_{y,k}^{W,obj} \\ \phi_k^{W,obj} \end{bmatrix} = \Phi \begin{bmatrix} x_k^{E,obj} \\ y_k^{E,obj} \\ v_{x,k}^{E,obj} \\ v_{y,k}^{E,obj} \\ \phi_k^{E,obj} \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_k^{W,ego} \\ y_k^{W,ego} \\ \phi_k^{W,ego} \end{bmatrix}, \tag{3.13}$$

where $\Phi = \text{diag}(R(\phi_k^{W,ego}), R(\phi_k^{W,ego}), 1)$ and the states and inputs corresponds to time k.

3.2 Datasets

The datasets used for tuning and testing the proposed methods are described in Tables 3.1-3.11. The number of cars and trucks is limited to objects observed for more than two seconds, while pedestrians are limited to those observed for more than one second. This distinction was made to enable a more meaningful analysis across all object types, as the number of pedestrian instances is significantly lower. For the tuning of cars and trucks, only the logs from April were considered, and for pedestrians, all logs in which pedestrians had been observed were considered. As for the routes, they are depicted in Figure 3.3.

Test log Cars and Trucks	
Metric	**Value**
Amount of Cars	579
Amount of Trucks	116
Total amount of time	26:18 minutes
Time of Collection	April
Route	Lindholmen - Varberg - Lindholmen

Table 3.1: Description of test dataset for Cars and Trucks

Test log Pedestrians	
Metric	**Value**
Amount of Pedestrians	52
Total amount of time	08:35 minutes
Time of Collection	February
Route	Around Lindholmen

Table 3.2: Description of test dataset for Pedestrians

Tuning log 1	
Metric	**Value**
Amount of Cars	1310
Amount of Trucks	57
Amount of Pedestrians	19
Total amount of time	50:56 minutes
Time of Collection	April
Route	Lindholmen - Mölndal - Älvsborgsbron - Lindholmen

Table 3.3: Description of Tuning dataset 1.

Tuning log 2	
Metric	**Value**
Amount of Cars	882
Amount of Trucks	187
Amount of Pedestrians	2
Total amount of time	50:48 minutes
Time of Collection	April
Route	Lindholmen - Varberg - Lindholmen

Table 3.4: Description of Tuning dataset 2.

Tuning log 3	
Metric	**Value**
Amount of Cars	369
Amount of Trucks	64
Amount of Pedestrians	17
Total amount of time	15:21 minutes
Time of Collection	April
Route	Lindholmen - Kungälv

Table 3.5: Description of Tuning dataset 3.

Tuning log 4	
Metric	**Value**
Amount of Cars	341
Amount of Trucks	55
Amount of Pedestrians	0
Total amount of time	10:52 minutes
Time of Collection	April
Route	Part of Lindholmen - Kungälv

Table 3.6: Description of Tuning dataset 4.

Tuning log 5	
Metric	**Value**
Amount of Cars	571
Amount of Trucks	77
Amount of Pedestrians	4
Total amount of time	29:28 minutes
Time of Collection	April
Route	Torslanda - Kungälv

Table 3.7: Description of Tuning dataset 5.

Tuning log 6	
Metric	**Value**
Amount of Cars	24
Amount of Trucks	7
Amount of Pedestrians	1
Total amount of time	02:23 minutes
Time of Collection	April
Route	Test Track

Table 3.8: Description of Tuning dataset 6.

Tuning log 7	
Metric	**Value**
Amount of Cars	3
Amount of Trucks	0
Amount of Pedestrians	1
Total amount of time	01:42 minute
Time of Collection	April
Route	Test Track

Table 3.9: Description of Tuning dataset 7.

Tuning log 8	
Metric	**Value**
Amount of Cars	52
Amount of Trucks	6
Amount of Pedestrians	1
Total amount of time	05:50 minutes
Time of Collection	January
Route	Part of Lindholmen - Mölndal - Älvsborgsbron - Lindholmen

Table 3.10: Description of Tuning dataset 8.

Tuning log 9	
Metric	**Value**
Amount of Cars	704
Amount of Trucks	210
Amount of Pedestrians	1
Total amount of time	41:32 minutes
Time of Collection	April
Route	Torslanda - Varberg - Torslanda

Table 3.11: Description of Tuning dataset 9.

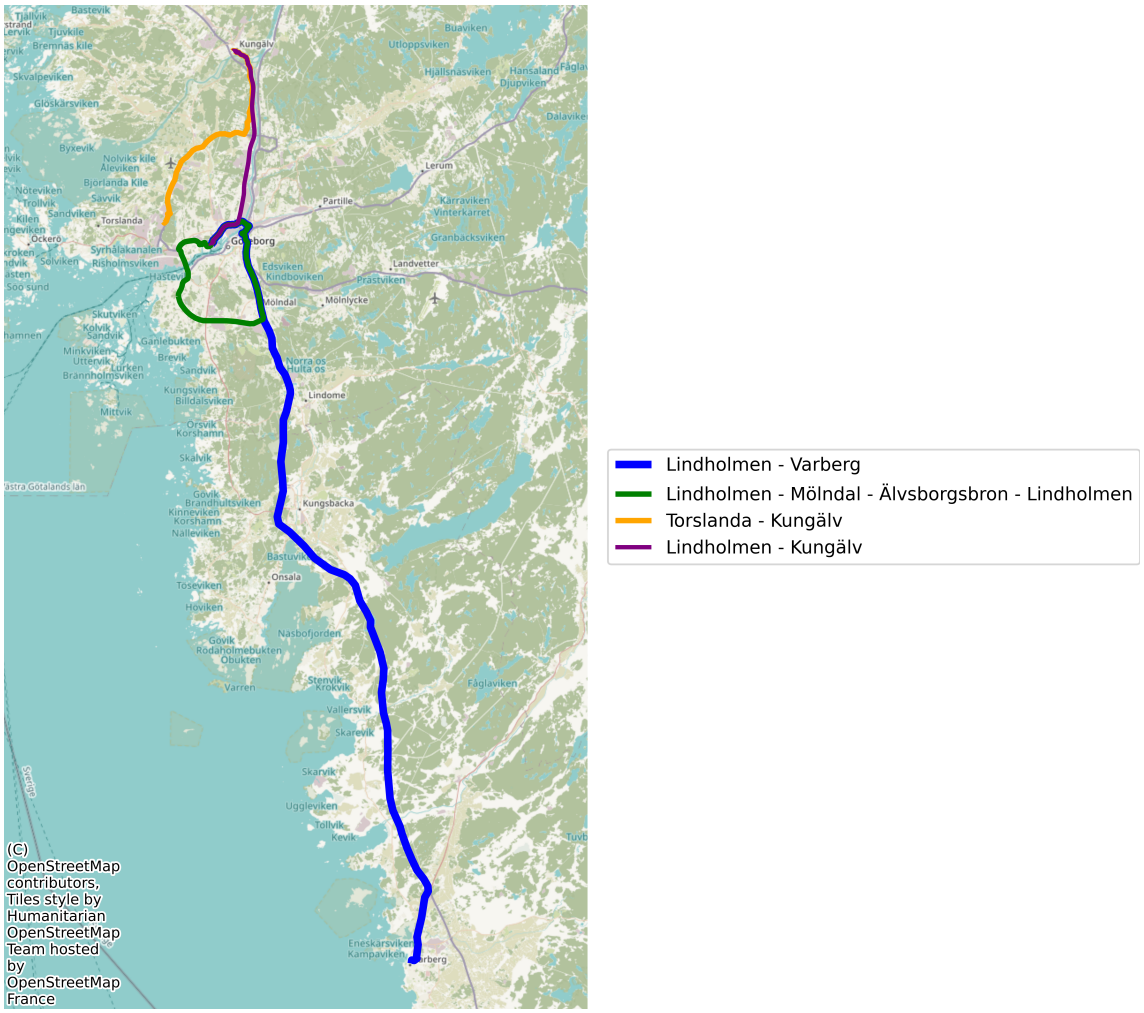


Figure 3.3: Map of the different routes

3.3 Simple Filter

First, two methods that are relatively straightforward to implement were investigated. The first is a basic anomaly detection approach, which identifies instances where object classifications change between consecutive time samples, as well as cases where objects disappear and later reappear. The second method is based on temporal coherence, specifically by evaluating the Intersection-over-Union (IoU) between bounding boxes. This approach derives bounding boxes at time steps $k-1$ and k using both position and velocity, and subsequently evaluates whether the predicted (velocity) and observed (position) bounding boxes are consistent with each other.

3.3.1 Basic Anomaly Detection

Since changes in object classification provide a clear indication of potential errors, a method for detecting such changes is required.

By evaluating the probability that a detected object belongs to each considered class, such as cars, pedestrians, or trucks, and applying a predefined threshold, the class with the highest probability is selected as the final classification, as shown in Algorithm 1.

Algorithm 1 Class Assignment

Initialize: $threshold, p_c, p_t, p_p$
for each object o_i in dynamic objects **do**
  $p_{\max} = \max(p_c, p_t, p_p)$
  if $p_{\max} > threshold$ **then**
    if $p_{\max} = p_c$ **then**
      $cls_{o_i} \leftarrow$ car
    else if $p_{\max} = p_t$ **then**
      $cls_{o_i} \leftarrow$ truck
    else if $p_{\max} = p_p$ **then**
      $cls_{o_i} \leftarrow$ pedestrian
    end if
  else
    $cls_{o_i} \leftarrow$ no object
  end if
end for

Then, if for a time sample an object that was previously detected as a car is now detected as a truck, an error is raised as shown in Algorithm 2.

Algorithm 2 Class Change Detection

Initialize: previous class cls_o^{k-1} for each tracked object
for each object o_i at time step k **do**
  if $cls_{o_i}^{k-1}$ exists **then**
    if $cls_{o_i}^{k} \neq cls_o^{k-1}$ **then**
      flag error for object o_i at time k
    end if
  end if
  Update $cls_{o_i}^{k-1} \leftarrow cls_{o_i}^{k}$
end for

Another common perception error involves objects disappearing and subsequently reappearing. Determining whether an object has truly disappeared or has simply moved outside the sensing range of the ego vehicle requires a temporal delay. It is only when the same object is detected again that it can be concluded that it temporarily disappeared from the perception system. Algorithm 3 gives pseudo-code for this method.

Algorithm 3 Disappearance and Reappearance Detection

```
Initialize: tracked objects with IDs
for each time step k do
  for each tracked object o_i do
    if o_i was present at time k − 1 and not detected at time k then
      mark o_i as temporarily missing
    end if
    if o_i was previously marked as missing and is detected again at time k then
      flag error for object o_i at time k
      remove missing status for o_i
    end if
  end for
end for
```

If an object is classified as a disappearing object, this instance is flagged as an error.

3.3.2 IoU between bounding boxes

Another method to detect unreasonable movement is temporal coherence, described in Section 2.2, which outlines how video frames can be compared from time sample to time sample to find unrealistic behavior between them. This thesis did something similar, but instead looks for matching behavior between bounding boxes calculated from position and from velocity. Essentially determining if they match.

In order to place the bounding boxes in proper relation to the world frame, which is derived as described Section 3.1.2, the length and width of each object's bounding box are used to recreate a 2D box as shown in Figure 3.4.

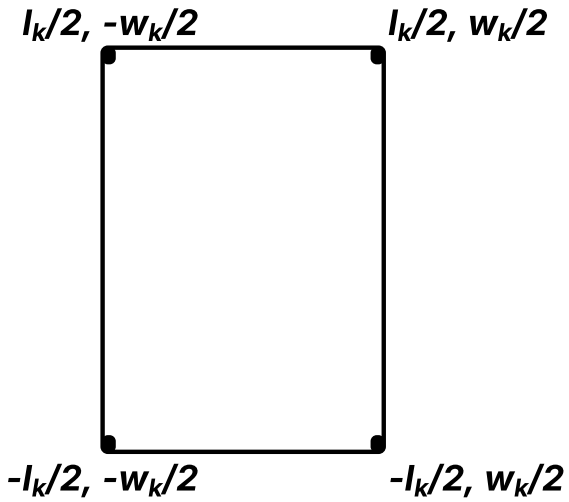


Figure 3.4: Bounding box for an object i, where l_k and w_k represents length and width respectively at time step k.

As shown in Figure 3.4, the bounding box corners are first defined relative to the object dimensions and then rotated and translated using the object pose

$$corners = \begin{bmatrix} \frac{-l_k}{2} & \frac{-w_k}{2} \\ \frac{l_k}{2} & \frac{-w_k}{2} \\ \frac{l_k}{2} & \frac{w_k}{2} \\ \frac{-l_k}{2} & \frac{w_k}{2} \end{bmatrix}, \quad R(\phi_k) = \begin{bmatrix} \cos(\phi_k) & -\sin(\phi_k) \\ \sin(\phi_k) & \cos(\phi_k) \end{bmatrix} \tag{3.14}$$

$$\mathbf{b}_{rot} = corners \cdot R(\phi_k)^T + [c_x \;\; c_y] \tag{3.15}$$

By multiplying with $R(\phi_k)$, where ϕ_k is calculated from the world frame at time step k and adding the center coordinates for x and y (c_x, c_y), the bounding box $\mathbf{b}_{rot}$ is placed correctly and rotated to accurately represent the objects rotation according to the world frame, which can be seen in Figure 3.5.

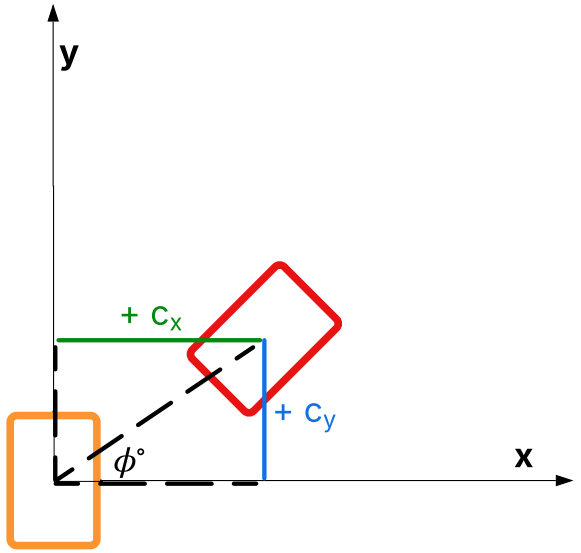


Figure 3.5: Where c_x and c_y shift the center of the box in space and $\phi°$ represents the angle of the rotation.

Since each sampled frame contains detected objects along with their corresponding bounding boxes, the overlap between consecutive boxes, $\mathbf{b}_{rot,k-1}$ and $\mathbf{b}_{rot,k}$, is expected to be relatively high when the sampling rate is sufficiently frequent. This overlap should be consistent with the measured velocity of the object, as a smaller displacement between frames results in a greater overlap. Consequently, the degree of overlap between consecutive bounding boxes can be used as an indicator of the object's movement, providing a way to estimate displacement based on how much the bounding boxes shift over time. Below are the calculations of displacement in the x- and y-directions based on the velocity. The velocities are rotated into the global frame using the rotation matrix at time step $k-1$, $R(\phi_{k-1})$. The object length and width are averaged between two consecutive time instances, and the corresponding area A is calculated as the mean of the areas at those time steps.

$$\Delta t = t(k) - t(k-1) \tag{3.16}$$

$$\begin{bmatrix} v_x \\ v_y \end{bmatrix} = R(\phi_{k-1})^T \cdot \begin{bmatrix} v_x(k-1) & v_y(k-1) \end{bmatrix} \tag{3.17}$$

$$l = \frac{l_k + l_{k-1}}{2}, \quad w = \frac{w_k + w_{k-1}}{2}, \quad A = \frac{(l_k \cdot w_k) + (l_{k-1} \cdot w_{k-1})}{2} \tag{3.18}$$

$$\Delta\sigma_v = \sigma_v \cdot \Delta t \tag{3.19}$$

$$\Delta x = v_x \cdot \Delta t, \quad \Delta y = v_y \cdot \Delta t, \tag{3.20}$$

where Δx and Δy is the expected displacement based on the velocity at time k as shown in Figure 3.6.

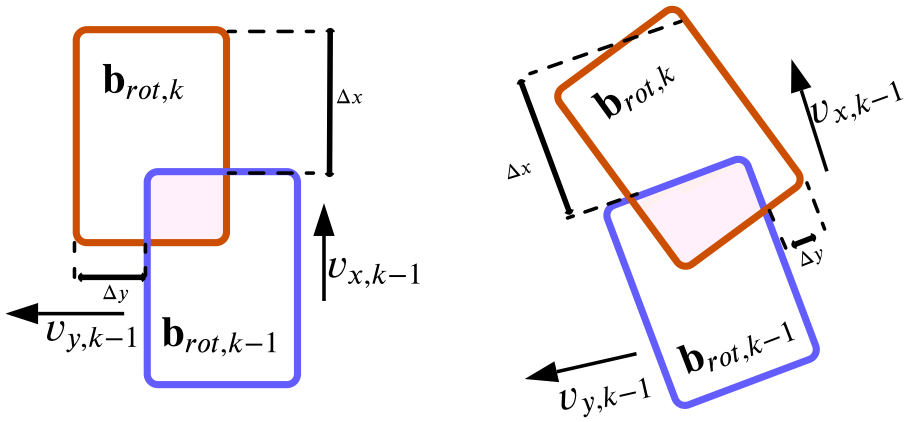


Figure 3.6: Depiction of displacement in x and y for two different scenarios.

Then, to find the overlap, IoU is calculated as described in Section 2.2.

$$\Delta x^+ = |\Delta x| + \rho \cdot \Delta\sigma_v + \rho \cdot \sigma_p, \quad \Delta y^+ = |\Delta y| + \rho \cdot \Delta\sigma_v + \rho \cdot \sigma_p \tag{3.21}$$

$$\Delta x^- = |\Delta x| - \rho \cdot \Delta\sigma_v - \rho \cdot \sigma_p, \quad \Delta y^- = |\Delta y| - \rho \cdot \Delta\sigma_v - \rho \cdot \sigma_p \tag{3.22}$$

$$O_{x^+} = \max(0, l - |\Delta x^+|), \quad O_{y^+} = \max(0, w - |\Delta y^+|) \tag{3.23}$$

$$O_{x^-} = \max(0, l - |\Delta x^-|), \quad O_{y^-} = \max(0, w - |\Delta y^-|) \tag{3.24}$$

$$\hat{I}_{++} = O_{x^+} \cdot O_{y^+}, \quad \hat{I}_{--} = O_{x^-} \cdot O_{y^-} \tag{3.25}$$

$$\hat{I}_{+-} = O_{x^+} \cdot O_{y^-}, \quad \hat{I}_{-+} = O_{x^-} \cdot O_{y^+} \tag{3.26}$$

$$\hat{U}_{++} = 2A - \hat{I}_{++}, \quad \hat{U}_{--} = 2A - \hat{I}_{--} \tag{3.27}$$

$$\hat{U}_{+-} = 2A - \hat{I}_{+-}, \quad \hat{U}_{-+} = 2A - \hat{I}_{-+} \tag{3.28}$$

$$\hat{IoU}_{++} = \frac{\hat{I}_{++}}{\hat{U}_{++}}, \quad \hat{IoU}_{--} = \frac{\hat{I}_{--}}{\hat{U}_{--}}, \quad \hat{IoU}_{+-} = \frac{\hat{I}_{+-}}{\hat{U}_{+-}}, \quad \hat{IoU}_{-+} = \frac{\hat{I}_{-+}}{\hat{U}_{-+}}, \tag{3.29}$$

where x^+ represents the upper bound obtained by adding a scaled standard deviation σ_{v_x} from the maximum value, and x^- represents the lower bound obtained by subtracting the same scaled standard deviation from the minimum value. The scaling factor ρ is chosen to match the confidence interval of the other methods evaluated in this thesis, as described in Section 3.4.3.

The standard deviation σ_v is estimated by separating the desired object from other object classes and identifying instances in which objects remain stationary. By focusing on stationary samples, the influence of motion-induced noise is minimized, allowing the remaining variance to primarily reflect measurement noise. Because bounding boxes for dynamic objects are allowed to rotate independently, their local coordinate axes (x^{o_i}, y^{o_i}) are not necessarily aligned with the world reference frame. As a result, the individual components x^{o_i}, y^{o_i}, $v_x^{o_i}$, and $v_y^{o_i}$ do not consistently represent motion or position along fixed global directions.

To obtain orientation-invariant measures, the position and velocity were therefore expressed in terms of their magnitudes, computed from the stationary object sam-

ples, as

$$v = \sqrt{v_x^2 + v_y^2} \tag{3.30}$$

$$p = \sqrt{x^2 + y^2}. \tag{3.31}$$

This ensures that the computed quantities are independent of the object's rotation and reflect the true speed and distance regardless of the coordinate alignment. Consequently, the standard deviations of position and velocity were calculated using these scalar magnitudes rather than their individual components. The resulting class-dependent standard deviations are then applied according to the classification of the current track.

To compare against the bounding box overlap derived from velocity, the overlap based on position must also be computed. Since the overlap between two bounding boxes is generally not rectangular, polygon-based geometric operations are required. Therefore, the Shapely library is used to compute the intersection and union of the polygons [30].

Here P_k denotes the polygon corresponding to the current time step and P_{k-1} the polygon from the previous time step. The intersection and union areas are defined as

$$I = \text{Area}(P_{k-1} \cap P_k) \tag{3.32}$$

$$U = \text{Area}(P_{k-1} \cup P_k). \tag{3.33}$$

The Intersection over Union (IoU) is then given by

$$\text{IoU} = \frac{I}{U}. \tag{3.34}$$

Time samples are deemed to contain errors when the measured IoU falls outside the predicted bounds from $\hat{IoU}$, indicating that the observed overlap is either greater or smaller than expected based on the object's velocity.

3.4 Kalman Filtering & Factor Graph Optimization

As indicated earlier, the Kalman filter only utilizes the past and not the future data, while the RTS smoother is performed as a backward pass and uses all the data. As a result of the smoother also using the future data, it has better accuracy and provides a more realistic movement of the states [14]. Due to that improvement and since some lag is allowed in this application, smoothing will always be performed and will be included in the Kalman filters in the report. The states referred to will therefore not be the filtered ones, but the smoothed ones.

In this thesis, the Kalman filters were constructed manually, while the FGO was implemented in Python with the GTSAM (Georgia Tech Smoothing and Mapping) toolbox [31] (version 4.2), and solved with the Levenberg-Marquardt optimizer. The tuning was performed with the scikit-optimize (version 0.10.2) function `gp_minimize` [28].

Since the Kalman filter, and its smoothing, and Factor Graph Optimization both build on the same principle, a Bayesian network based on measurements and motion models, it is reasonable to perform the tuning and outlier detection in the same manner. While the evaluation is performed on fixed windows, the tuning and the selection of outlier-threshold were performed such that all data was available for the Kalman filters and FGO. In addition, it was performed on a set of data logs to get a variety of data, but the data log on which the testing was performed was not included in the aforementioned set.

3.4.1 Measurement models

As previously stated, the measurements have been limited, for this project, to only include position and velocity. The measurement model, for both the CV and CT models, is therefore

$$\mathbf{z}_k = \mathbf{h}(\mathbf{x_k}) + \mathbf{r}_k = \begin{bmatrix} x_k \\ y_k \\ v_{x,k} \\ v_{y,k} \end{bmatrix} + \mathbf{r}_k, \tag{3.35}$$

where $\mathbf{r}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_k)$ and

$$\mathbf{R}_k = \text{diag}((\sigma_p^{meas})^2, (\sigma_p^{meas})^2, (\sigma_v^{meas})^2, (\sigma_v^{meas})^2). \tag{3.36}$$

The noise is assumed to be identical for the Cartesian components, with one covariance assigned for position and another for velocity. While it is expected to have a slightly smaller measurement noise for objects observed in front of the ego vehicle, the rotations performed, as detailed in Section 3.1, justify the selection of equal noise covariance in the Cartesian components.
The corresponding Jacobian for the CV model is

$$\mathbf{h}'(\mathbf{x_k}) = \mathbf{I}_{4\times 4}, \tag{3.37}$$

and for the CT model

$$\mathbf{h}'(\mathbf{x_k}) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}. \tag{3.38}$$

3.4.2 Tuning

The system being evaluated changes continuously in terms of accuracy since it is a system in development and will continue to change and improve as more data is

collected. As a result, it is not reasonable to tune the filters by hand, therefore auto-tuning is needed. One alternative for this is the Adaptive Kalman filter [32], respectively Adaptive Factor Graph Optimization [22]. With the adaptive methods, the filter/smoother will adapt the noise covariance to each object, which may result in a very noisy measurement not being flagged, as it would be considered normal. Therefore, the Adaptive Kalman filter, respectively, the Adaptive Factor Graph Optimization, was not considered for this project. Consequently, under the assumption of constant sampling time and stationary noise characteristics, the process and measurement noise covariance matrices were assumed to be constant

$$\begin{aligned} \mathbf{Q}_k &= \mathbf{Q} \\ \mathbf{R}_k &= \mathbf{R} \end{aligned} \qquad \forall k. \tag{3.39}$$

It should also be noted that the noise covariance matrices are not necessarily identical in terms of magnitude, since they exhibit slightly different motion characteristics. Each object within a category, for example, all pedestrians, will, however, share a noise covariance matrix.

Another method is to construct an optimization problem by using either Normalized Innovation Squared [33] or Maximum Likelihood [34] as the objective function. The problem with Normalized Innovation Squared is that it expects the data to be of normal distribution, which is often not the case for real-life data, thus it would probably result in selecting a too large measurement covariance. For maximum likelihood, any probability distribution can be used, such as normal, Student's t, or exponential, resulting in it being the selected choice for the auto-tuning.

The objective function defined for the optimization problem, with the full derivation provided in Appendix A, is

$$\begin{aligned} \arg\max_{\boldsymbol{\sigma}} \quad & \frac{1}{M}\sum_{i}^{M}\Bigg(\frac{1}{N_i}\sum_{k=1}^{N_i}\Bigg(-\frac{\nu^p+2}{2}\log\Bigg(1+\frac{1}{\nu^p}(\mathbf{e}^p_{\mathbf{z}_{k,i}})^\top(\mathbf{R}^p)^{-1}\mathbf{e}^p_{\mathbf{z}_{k,i}}\Bigg) \\ & \qquad -\frac{\nu^v+2}{2}\log\Bigg(1+\frac{1}{\nu^v}(\mathbf{e}^v_{\mathbf{z}_{k,i}})^\top(\mathbf{R}^v)^{-1}\mathbf{e}^v_{\mathbf{z}_{k,i}}\Bigg) \\ & \qquad -\frac{1}{2}\mathbf{e}^\top_{\mathbf{x}_{k,i}}\mathbf{Q}^{-1}\mathbf{e}_{\mathbf{x}_{k,i}}\Bigg)\Bigg) \\ & -\frac{1}{2}\log|\mathbf{Q}|-\frac{1}{2}\log|\mathbf{R}| \\ & +\sum_{\tau\in\mathcal{T}}\sum_{j\in\mathcal{J}_m^\tau}\Bigg(-\log(\sigma_j^\tau)-\frac{(\log(\sigma_j^\tau)-\mu_{0j}^\tau)^2}{2(\sigma_{0j}^\tau)^2}\Bigg), \end{aligned} \tag{3.40}$$

where $\tau \in \mathcal{T} \coloneqq \{meas, proc\}$ defines if it is process or measurement noise and $m \in \mathcal{M} = \{CV, CT\}$, defines which motion model it is for. The sets $\mathcal{J}_m^\tau$, are defined as

$$\begin{aligned} \mathcal{J}_{CV}^{meas} = \mathcal{J}_{CT}^{meas} &\coloneqq \{p, v\} \\ \mathcal{J}_{CV}^{proc} \coloneqq \{v\}, \quad \mathcal{J}_{CT}^{proc} &\coloneqq \{v, \omega\}, \end{aligned} \tag{3.41}$$

and represents what types of variables the corresponding noise is directly applied to.

The problem is formulated using the maximum average log likelihood, computed as the mean of per-object averages. This formulation ensures consistency across varying numbers of dynamic objects and prevents objects observed over long durations from having a disproportionate influence. The variables $\mathbf{e}^p_{\mathbf{z}_{k,i}}$ and $\mathbf{e}^v_{\mathbf{z}_{k,i}}$, are defined as the difference between a state estimate and its corresponding measurement, for position, respectively, velocity. $\mathbf{e}_{\mathbf{x}_{k,i}}$ refers to the difference observed between a state estimate and its predicted value given the previous state estimate. The following separation has been used

$$\mathbf{R} = \begin{bmatrix} \mathbf{R}^p & \mathbf{0} \\ \mathbf{0} & \mathbf{R}^v \end{bmatrix}, \tag{3.42}$$

which is possible since the measurement noise covariance matrix has been constructed as a diagonal matrix, as provided in Equation (3.36).
It has been observed that the measurement noise has a slightly heavier tail than the normal distribution, therefore it has been modeled as a Student's t distribution. The heaviness of the tail of the distribution is dependent on its degrees of freedom (ν^p, ν^v), where more degrees of freedom are closer to Gaussian. The degree of freedom can be selected to be different for position and velocity, but for this work, both are set to 4, giving $\nu^p = \nu^v = 4$. A comparison of the student's t distribution and the normal distribution is provided in Figure 3.7.

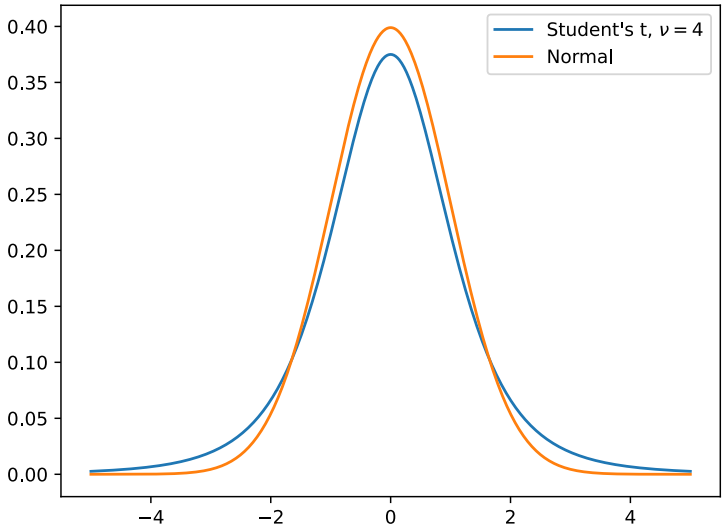


Figure 3.7: Student's t distribution with 4 degrees of freedom and Normal distribution, both with mean 0 and standard deviation 1.

In addition to utilizing the dynamic objects' behavior given different measurement noise and process noise, the parameters were regularized using the log-normal distribution and an approximate span of what the parameters probably are, given the behavior observed. The regularization is needed to avoid degenerate solutions, where

the optimizer may push noise covariances to unrealistic extreme values.

The regularization parameters were calculated accordingly, from the definition of random variables of a log-normal distribution

$$\begin{aligned} \sigma_{0j}^{\tau} &= \frac{\log(\sigma_{j,max}^{\tau}) - \log(\sigma_{j,min}^{\tau})}{2} \\ \mu_{0j}^{\tau} &= \frac{\log(\sigma_{j,max}^{\tau}) + \log(\sigma_{j,min}^{\tau})}{2}, \end{aligned} \tag{3.43}$$

where $\sigma_{j,max}^{\tau}$ and $\sigma_{j,min}^{\tau}$ represent the approximate span of what it should be, although the optimizer is allowed to select parameters outside of the span if the dynamic objects' behavior supports it. The parameters σ_{0j}^{τ} and μ_{0j}^{τ} correspond to the approximated standard deviation and mean value of the distribution from which the parameter σ_{j}^{τ} can be selected.
The following spans were selected for the measurement covariance for all objects

$$\begin{aligned} \sigma_{p,min}^{meas} &= 0.1, \qquad \sigma_{p,max}^{meas} = 0.4 \\ \sigma_{v,min}^{meas} &= 0.2, \qquad \sigma_{v,max}^{meas} = 0.5. \end{aligned} \tag{3.44}$$

For the process noise covariance, the following spans were selected for all objects

$$\begin{aligned} \sigma_{v,min}^{proc} &= 0.5, \qquad \sigma_{v,max}^{proc} = 1.2 \\ \sigma_{\omega,min}^{proc} &= 0.05, \qquad \sigma_{\omega,max}^{proc} = 0.1. \end{aligned} \tag{3.45}$$

The process noise on ω only applies to the CT model. For this project, the data was limited to mostly highways, meaning that for city driving, the process noise span may need to be increased slightly for vehicles.

Multiple optimization methods can be used for solving the optimization problem. For this scenario, only the cost will be available with no exact gradients, while it will also consist of multiple local optimums. As a result, the local optimizers and the gradient-based methods can not be used. Therefore, a derivative-free global optimizer has to be used, and for this project, a Bayesian Optimizer has been selected, with the acquisition function set to GP-hedge [27]. GP-hedge is constructed such that it, based on a probability, selects an acquisition method at every iteration. As a result, it is able to perform well during both exploration and exploitation, but as a consequence, it also requires more computational cost.

3.4.3 Outlier detection

For inconsistency detection, multiple methods have been proposed, such as Innovation Saturation [35] and Normalized Innovation Squared [36]. Innovation Saturation is a method where innovations above a threshold are saturated to not influence the filter too much, where the innovation can be both constant and adaptive [35]. The adaptive one does, however, require a lot of tuning parameters, for which the tuning will be too tedious and unreasonable for this problem, where the system tested might change in accuracy. The Normalized Innovation Squared is a method used

most commonly for evaluating if the covariances of the Kalman filter are consistent [36], where each value is calculated

$$\mathbf{NIS}_{k,i} = \mathbf{v}_{k,i}^{\top} \mathbf{S}_{k,i}^{-1} \mathbf{v}_{k,i}, \tag{3.46}$$

where $\mathbf{v}_{k,i}$ is the innovation and $\mathbf{S}_{k,i}$ is the innovation covariance for timesample k for dynamic object i. Given Gaussian noise, it should follow a Chi-squared distribution, $\mathcal{X}^2$, of the same degrees of freedom as the number of measurements. The method can also be used as a measure of inconsistencies of singular measurements, where the larger NIS values are flagged. The threshold is generally selected such that values above a certain percentage are flagged [36]. The illustration of an expected distribution, with 4 measurements and multiple percentage-based thresholds, is provided in Figure 3.8.

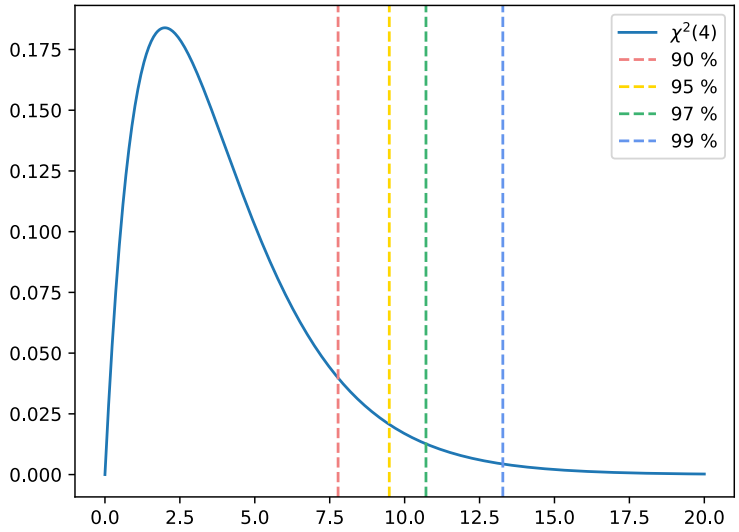


Figure 3.8: Chi-squared distribution with 4 degrees of freedom, $\mathcal{X}^2(4)$, and threshold for Normalized Innovation Squared given different percentages.

By using the innovation covariance matrix, $\mathbf{S}_{k,i}$, it is ensured that something is only flagged as an inconsistency if the filter is certain. As a result of that, if the measurements are extremely inconsistent, the filter will be highly uncertain, resulting in some major errors not being flagged or getting flagged too late. In addition, the innovation covariance matrix is only present in the regular Kalman filter and not in the smoothing or FGO. Thus, using the standard Normalized Innovation Squared might not be ideal.
The metric used for the outlier detection for the Kalman filters and FGO was selected based on NIS, but modified to account for the previously described problems

$$\Psi_{k,i} = \mathbf{e}_{\mathbf{z}_{k,i}}^{\top} \mathbf{R}^{-1} \mathbf{e}_{\mathbf{z}_{k,i}}, \tag{3.47}$$

where $\mathbf{R}$ is the measurement covariance matrix and $\mathbf{e}_{\mathbf{z}_{k,i}}$ is the difference between the state and corresponding measurement of timesample k for object i. Given a Gaussian

distribution of the measurements and correct states, the distribution of the above measure, $\Psi_{k,i}$ should be a Chi-squared distribution, $\mathcal{X}^2$, and should therefore be possible to use in the same manner as NIS for outlier detection.
As previously stated, the measurement noise has a too heavy tail to be considered Gaussian, which, as a consequence, gives an outlier metric with a heavy tail. To select based on the Chi-squared distribution, $\mathcal{X}^2$, would therefore not be representative, as a much larger percentage would be selected compared to what is expected from the $\mathcal{X}^2$-distribution. Therefore, the threshold is calculated after the tuning is performed, based on the desired percentage and the percentile value of the tuning data. Then

$$\Psi_{k,i} > \Psi_{threshold} \tag{3.48}$$

indicates that a data point should be flagged, where $\Psi_{threshold} = \Psi_p$ is selected such that the most inconsistent $(1 - p)$ % of observations are flagged.

3.5 Evaluation

In order to determine how well each method performs, three different evaluation metrics should be calculated. Recall, Precision, and an F_1 score [37], where True Positives, False Positives, and False Negatives are referred to as TP, FP, and FN, respectively.

$$Recall = \frac{TP}{TP + FN} \qquad Precision = \frac{TP}{TP + FP} \tag{3.49}$$

$$F_1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{3.50}$$

To compute these metrics, the evaluation is conducted in two steps: first, on simulated data where objects have been modified to emulate errors, and second, on the original, unaltered dataset.

3.5.1 Simulation Setup

The simulation uses tumbling windows, which were briefly mentioned in Section 2.6, to partition the data stream into fixed-size, non-overlapping batches. Unlike sliding windows, which allow overlapping intervals, each data point in a tumbling window belongs to exactly one window.

These experiments are conducted using different window sizes (1 s, 3 s, and 5 s). For each window size, a method, for example, the Kalman filter, is evaluated across all error types described in Section 3.5.2. This enables a comparison of how window size affects performance. In particular, smaller window sizes may reduce computational cost and latency, while larger window sizes can improve estimation accuracy. Thus, the evaluation reflects the well-known trade-off between accuracy and computational time [38]. It should also be noted, as discussed in Section 3.1, that no reset is performed to account for drift. However, this should not affect the comparison

between the methods, as all were evaluated using the same logs.

A total of five primary error types were generated, as described in Section 3.5.2, with one simulation run performed for each type. As a result, each method was evaluated for each object class using three different window sizes across all five error scenarios. For these simulations, an infinite buffer was assumed. While this does not reflect the constraints of an actual vehicle fleet, it is useful for evaluating the methods.

Each object class is evaluated using different values of Ψ and the corresponding ρ, defined as

$$\text{Car: } \Psi = \Psi_{99.25} \quad \rho = 2.680 \tag{3.51}$$

$$\text{Trucks: } \Psi = \Psi_{99} \quad \rho = 2.576 \tag{3.52}$$

$$\text{Pedestrian: } \Psi = \Psi_{90} \quad \rho = 1.645. \tag{3.53}$$

These values are chosen to account for variations in the behavior of the perception system across different object types, as reflected in the data logs used in this thesis. The specified thresholds were determined empirically through extensive testing.

To calculate precision, the evaluation is performed on unaltered data. For each object identified as inconsistent by a given method, a manual annotation is carried out to classify it as either a True Positive (TP) or a False Positive (FP). Violin plots are generated for each method and window size to illustrate how effectively the methods separate True Positives (TP) from False Positives (FP). These plots are based on the extent to which each object lies beyond the corresponding decision threshold. For the temporal coherence method, the threshold definition differs slightly. As described in Section 3.3.2, the Intersection-over-Union (IoU) for bounding boxes is defined by both an upper and a lower bound. Consequently, the deviation is computed as the distance from the nearest violated bound, regardless of whether the object exceeds the upper limit or falls below the lower limit. For methods based on Ψ, objects are instead characterized by how far their values exceed the predefined threshold. In all cases, the maximum deviation per object is used for visualization. It should also be noted that temporal coherence is evaluated using a single window size, as the method operates in real time and does not require any delay. As the threshold for temporal coherence is dynamic, it's not displayed in the figures.

Errors were also introduced at varying levels of severity, categorized as *moderate* and *severe*. Moderate errors reflect more commonly occurring discrepancies, whereas severe errors represent more extreme deviations that, while less frequent, still occur naturally in the data. Accordingly, recall and F_1-score are evaluated under both conditions.

The artificial errors were created and included in the data as mentioned below in 3.5.2

3.5.2 Artificial Errors

By studying behavior that might show up naturally in the reference system, five typical errors were determined to be a good representation of errors that could show up during a drive.

Bias: By adding a bias term b to the velocity in both x and y as

$$v'_x = v_x + b \tag{3.54}$$
$$v'_y = v_y + b. \tag{3.55}$$

This type of error is induced during the entire duration that an object was seen, so as not to trigger identification of an error when the velocity suddenly either jumps up or down.

Spike in Velocity: As with bias, here a spike term s_v is added to the velocity in both x and y as

$$v'_x = v_x + s_v \tag{3.56}$$
$$v'_y = v_y + s_v. \tag{3.57}$$

This, however, is only applied to a single timestamp.

Spike in Position: Exactly as above but s_p is applied to x and y as

$$x' = x + s_p \tag{3.58}$$
$$y' = y + s_p, \tag{3.59}$$

again to a single timestamp.

Drift in Velocity: Similar to bias in that it's an error applied over a period of time, the drift error aims to emulate the slow change of velocity or position over time. It was done as

$$v'_x = v_x + \alpha \cdot d_v \tag{3.60}$$
$$v'_y = v_y + \alpha \cdot d_v, \tag{3.61}$$

where d_v is multiplied by α, which will increase for each sample the error is being induced. After half the time, the drift will instead start to go down again by allowing α to decrease.

Drift in Position: As above

$$x' = x + \alpha \cdot d_p \tag{3.62}$$
$$y' = y + \alpha \cdot d_p, \tag{3.63}$$

where d_p is multiplied by α.

Two additional error types were introduced: one related to class assignment and one related to object disappearance and reappearance.

For the class-related error, let c_1 denote the originally most probable class and c_2 another class. The probability mass of c_1 is reassigned to c_2 and increased by an offset o_c, such that c_2 becomes the new most probable class

$$p(c_1) \leftarrow 0 \tag{3.64}$$
$$p(c_2) \leftarrow p(c_1) + o_c. \tag{3.65}$$

Some objects in the dataset naturally contained instances where they disappeared and reappeared. As such, this error type was not artificially introduced but instead directly included from the data. These objects were absent for no more than a single time step.

4

Results

This chapter presents the results obtained from the different evaluated methods. The results of the basic anomaly detection are presented first, followed by the motion anomaly detection algorithms. The results are further organized by object class, namely pedestrians, cars, and trucks.

4.1 Basic Anomaly Detection

Pedestrians		
Error	**Class**	**Rediscovery**
Recall	91.43%	100.0%

(a) Pedestrians.

Cars		
Error	**Class**	**Rediscovery**
Recall	99.0%	100.0%

(b) Cars.

Trucks		
Error	**Class**	**Rediscovery**
Recall	97.0%	100.0%

(c) Trucks.

Table 4.1: Recall for basic anomaly detection across object types.

4.2 Object type: Pedestrians

For pedestrians, tests were done on a dataset of objects where 35 of them contained artificial errors. Tests were done as described in Section 3.5, with $\Psi_{threshold} = \Psi_{90}$ and $\rho = 1.645$. Below follow figures depicting recall for moderate errors Figure 4.1 and recall for severe errors Figure 4.2. As well as precision Figure 4.3 and F_1-score Figure 4.4, Figure 4.5 for moderate and severe errors respectively. Tables with data from the simulation can be found in Appendix B.1.

4.2.1 Recall, Precision and F_1 Score for Pedestrians

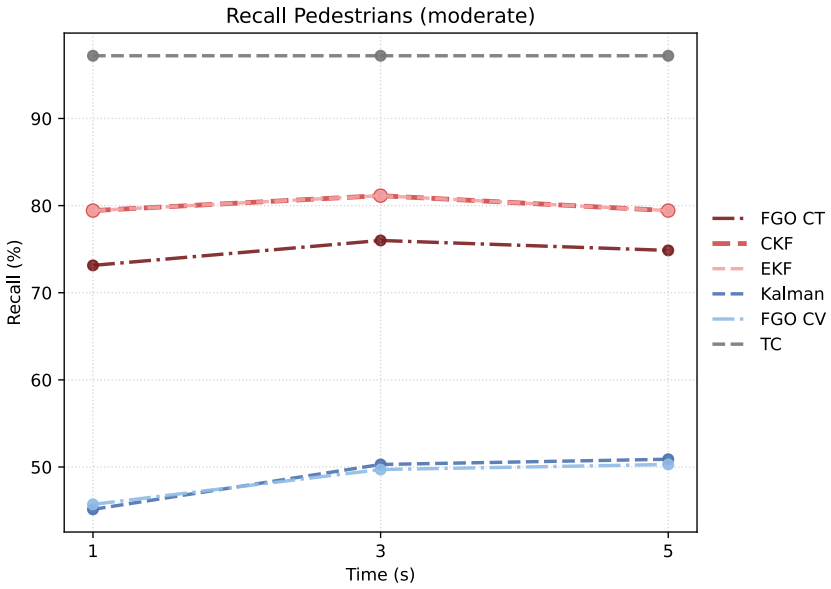


Figure 4.1: Recall for moderate induced errors for pedestrians.

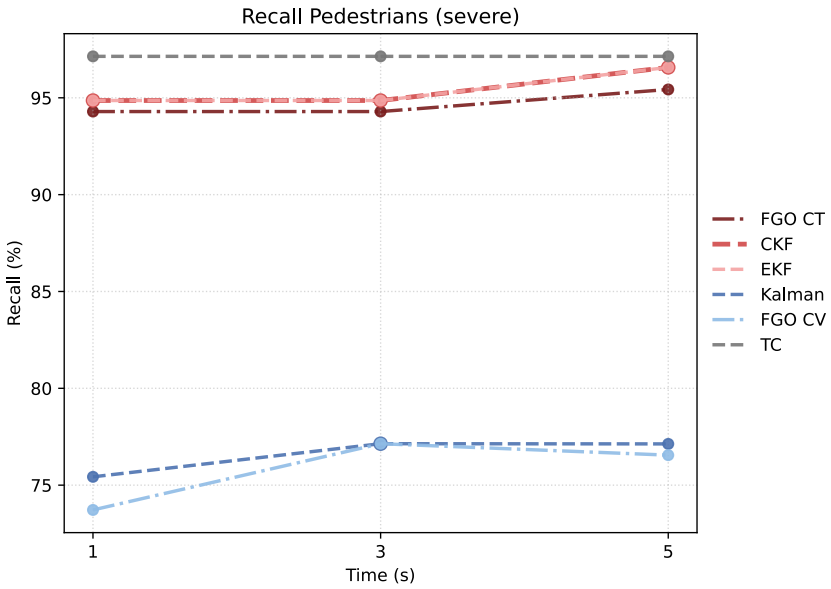


Figure 4.2: Recall for severe induced errors for pedestrians.

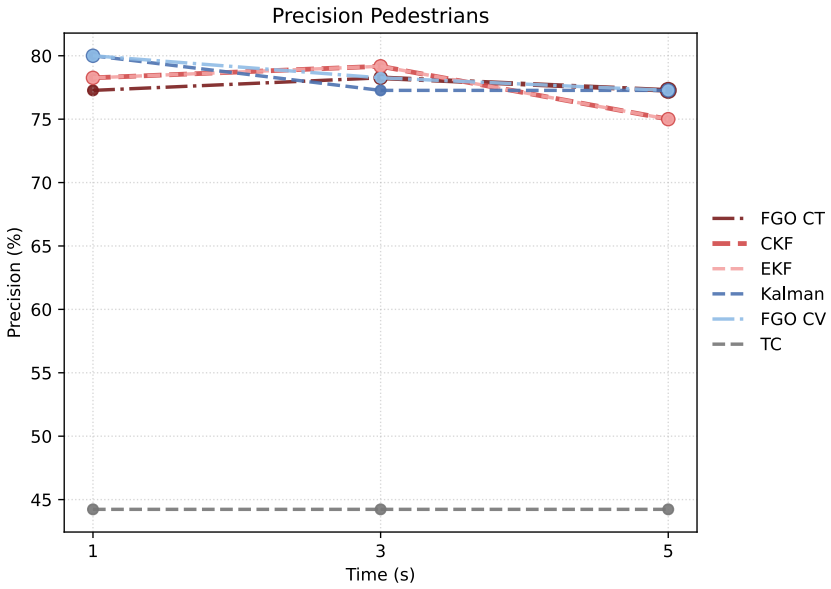


Figure 4.3: Precision for pedestrians.

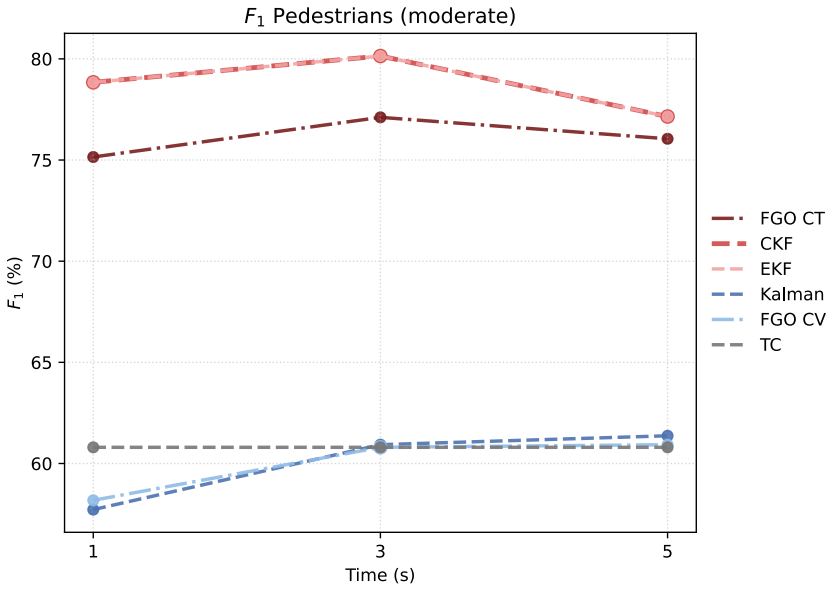


Figure 4.4: F_1 for moderate induced errors for pedestrians.

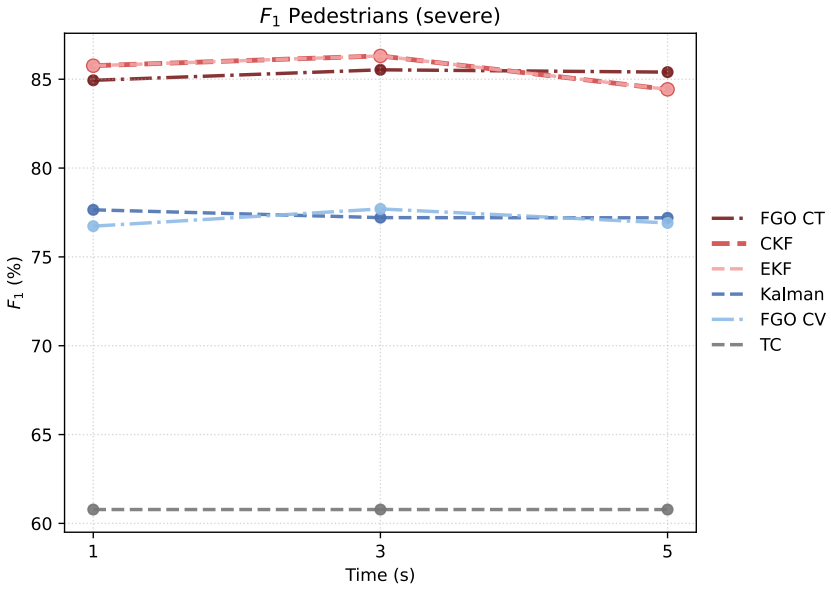


Figure 4.5: F_1 for severe induced errors for pedestrians.

4.2.2 Separation of Consistent and Inconsistent Data for Pedestrians

As described in Section 3.5, the following figures illustrate each method's ability to distinguish between consistent and inconsistent data.

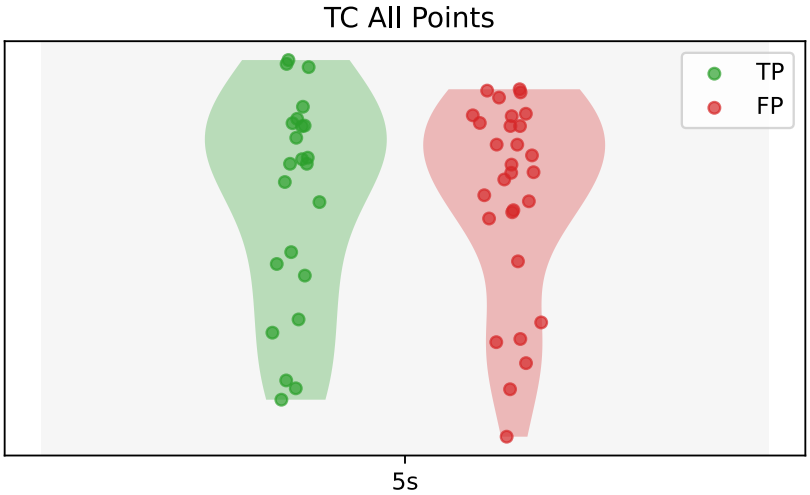
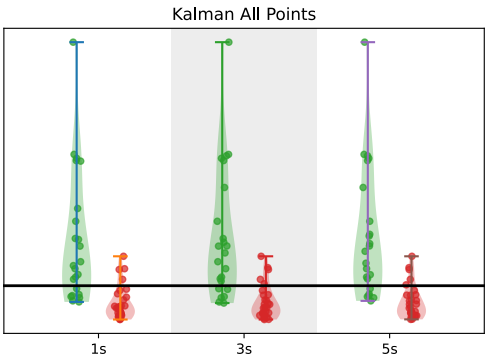
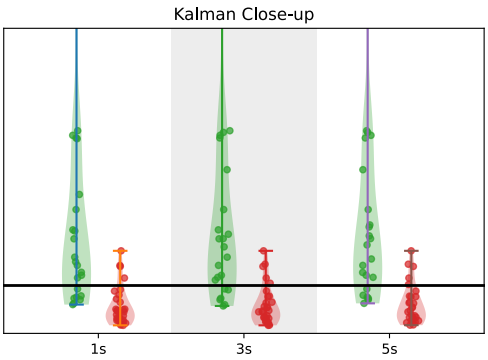


Figure 4.6: Violin plot for Temporal Coherence showing overlap between consistent and inconsistent data.



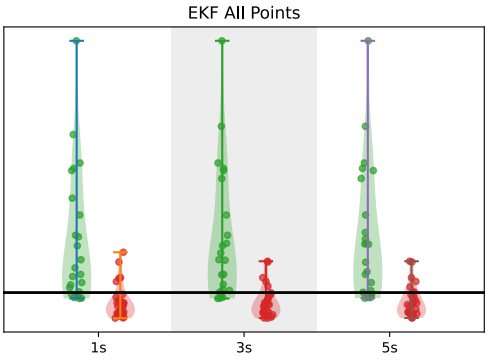
(a) Complete Distribution of Data Points.



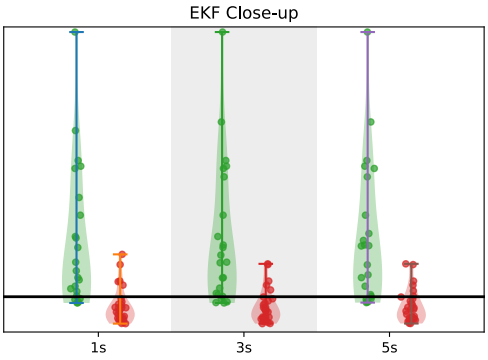
(b) Detailed View of Data Point Distribution.

Figure 4.7: Violin plots for the Kalman Filter showing overlap between consistent and inconsistent data, comparing batch sizes of 1 second (left), 3 seconds (middle), and 5 seconds (right).

.

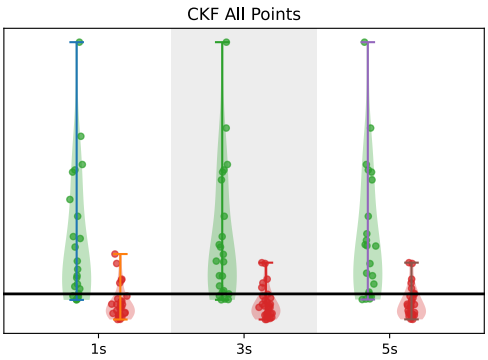


(a) Complete Distribution of Data Points.

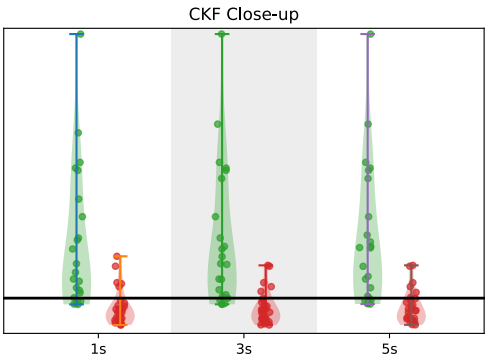


(b) Detailed View of Data Point Distribution.

Figure 4.8: Violin plots for the Extended Kalman Filter showing overlap between consistent and inconsistent data, comparing batch sizes of 1 second (left), 3 seconds (middle), and 5 seconds (right).

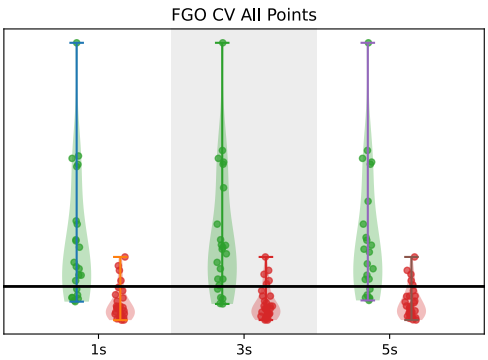


(a) Complete Distribution of Data Points.

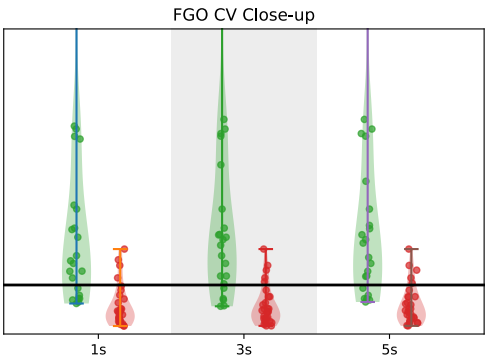


(b) Detailed View of Data Point Distribution.

Figure 4.9: Violin plots for the Cubature Kalman Filter showing overlap between consistent and inconsistent data, comparing batch sizes of 1 second (left), 3 seconds (middle), and 5 seconds (right).

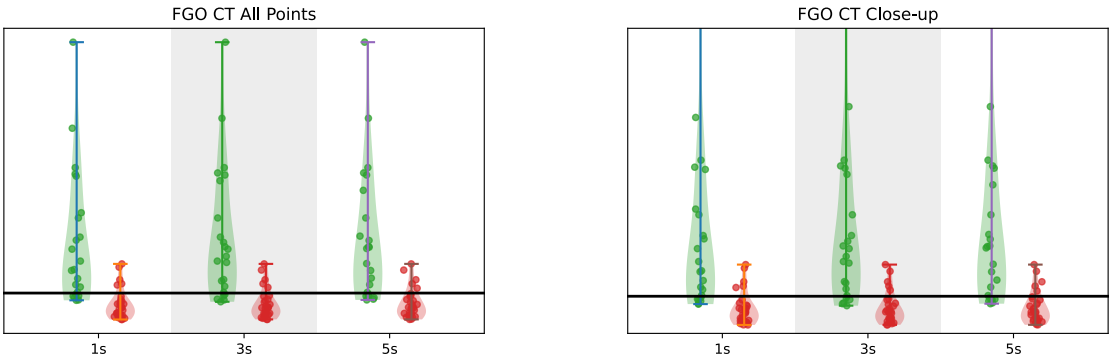


(a) Complete Distribution of Data Points.



(b) Detailed View of Data Point Distribution.

Figure 4.10: Violin plots for Factor Graph Optimization with CV showing overlap between consistent and inconsistent data, comparing batch sizes of 1 second (left), 3 seconds (middle), and 5 seconds (right).



(a) Complete Distribution of Data Points.

(b) Detailed View of Data Point Distribution.

Figure 4.11: Violin plots for Factor Graph Optimization with CT showing overlap between consistent and inconsistent data, comparing batch sizes of 1 second (left), 3 seconds (middle), and 5 seconds (right).

4.3 Object type: Cars

For cars, tests were done on a dataset of objects where 100 of them contained artificial errors. Tests were done as described in Section 3.5, with $\Psi_{threshold} = \Psi_{99.25}$ and $\rho = 2.68$. Below follow figures depicting recall for moderate errors Figure 4.12 and recall for severe errors Figure 4.13. As well as precision Figure 4.14 and F_1-score Figure 4.15, Figure 4.16 for moderate and severe errors respectively. Tables with data from the simulation can be found in Appendix B.2.

4.3.1 Recall, Precision and F_1 Score for Cars

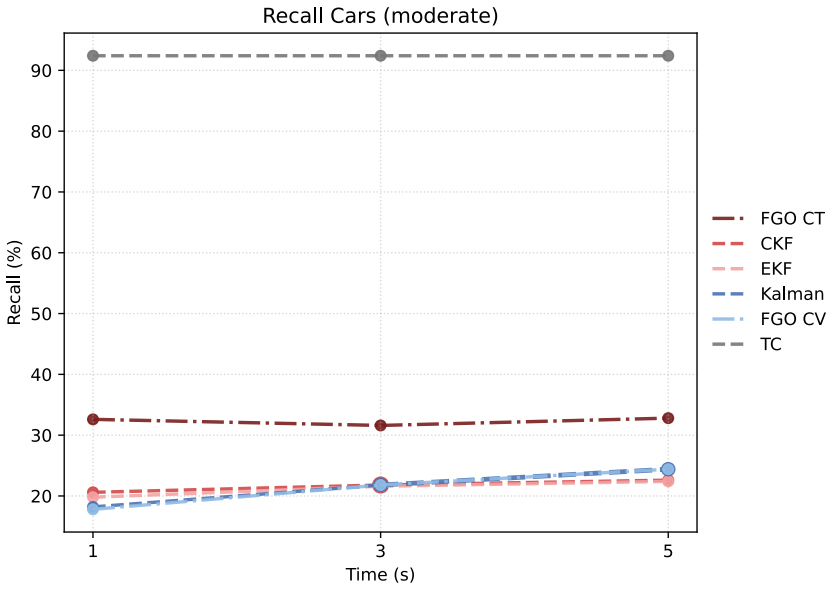


Figure 4.12: Recall for moderate induced errors for cars.

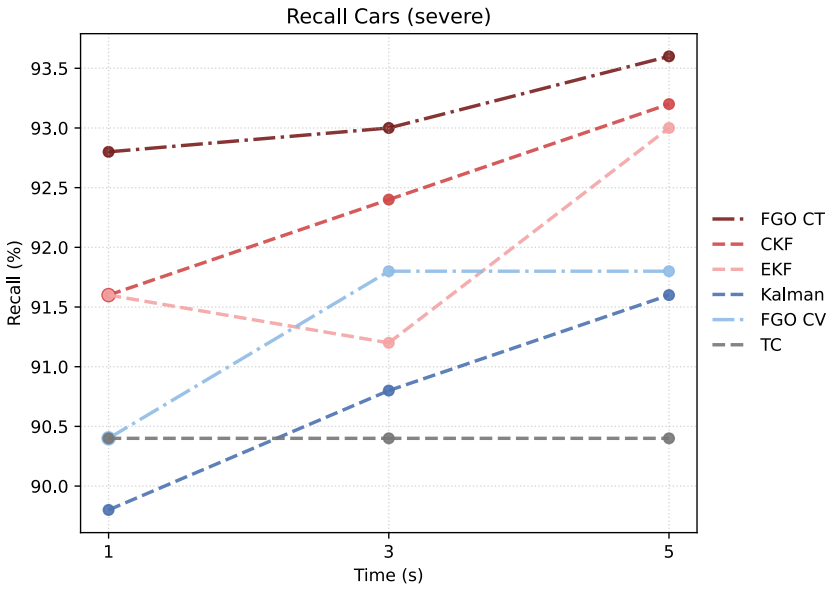


Figure 4.13: Recall for severe induced errors for cars.

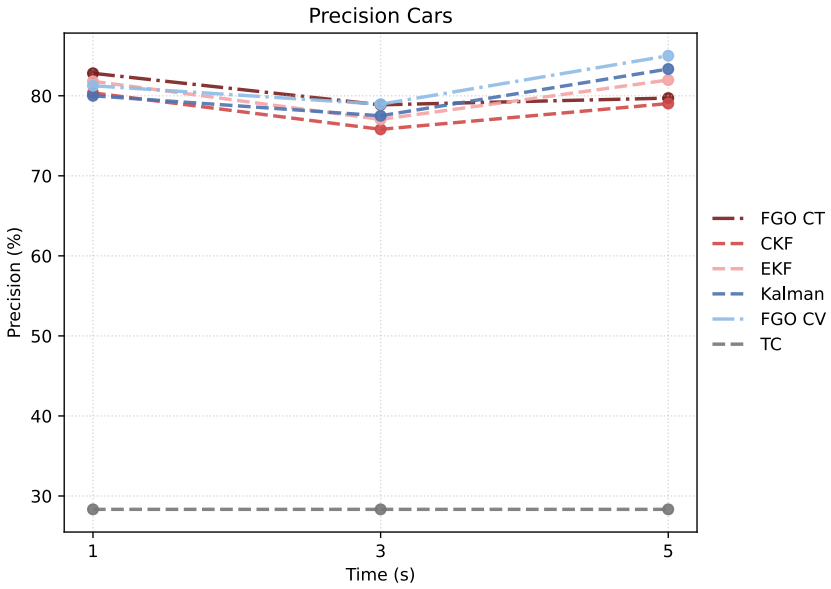


Figure 4.14: Precision for cars.

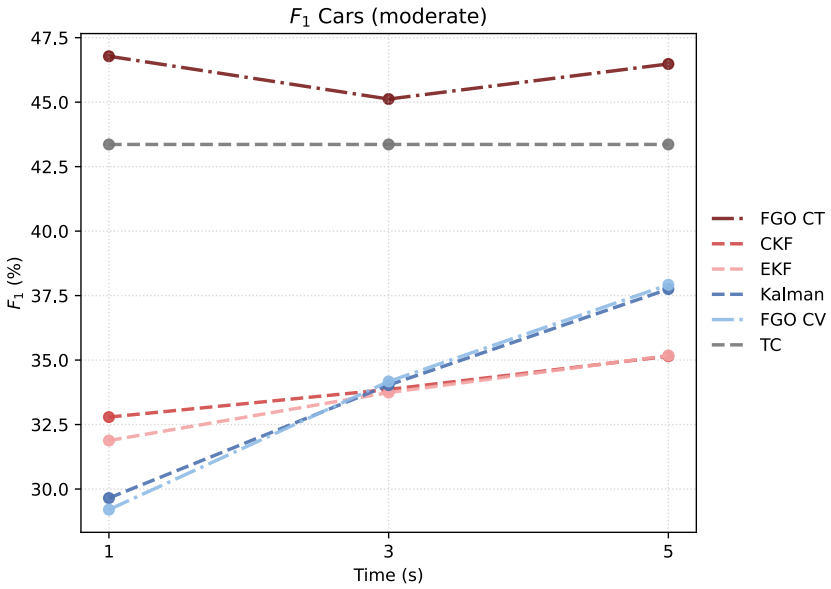


Figure 4.15: F_1 for moderate induced errors for cars.

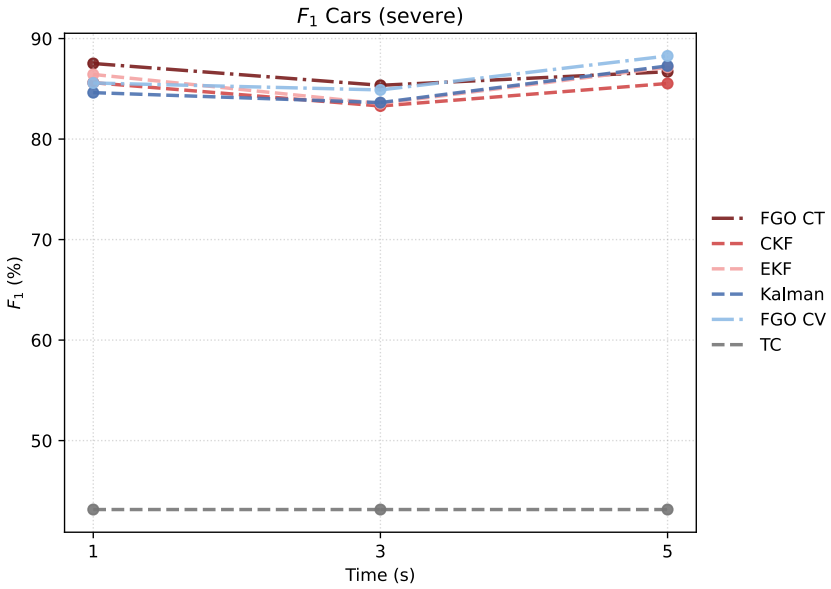


Figure 4.16: F_1 for severe induced errors for cars.

4.3.2 Separation of Consistent and Inconsistent Data for Cars

The following depicts each method's ability to differentiate between consistent and inconsistent data, as described in Section 3.5.

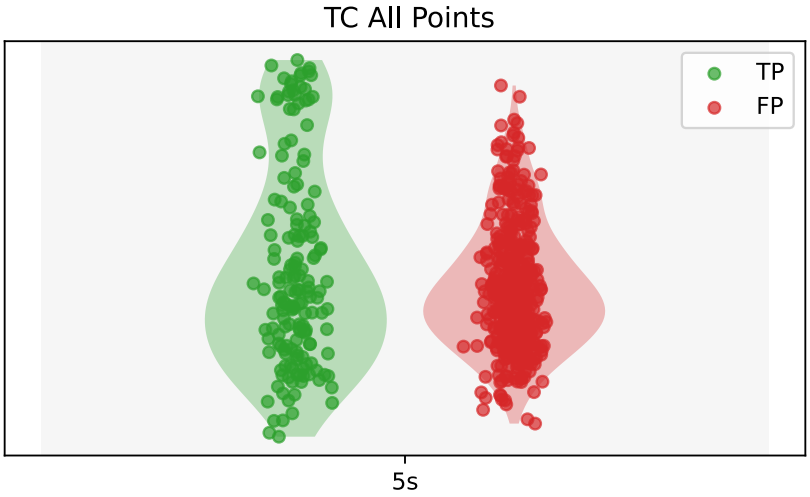
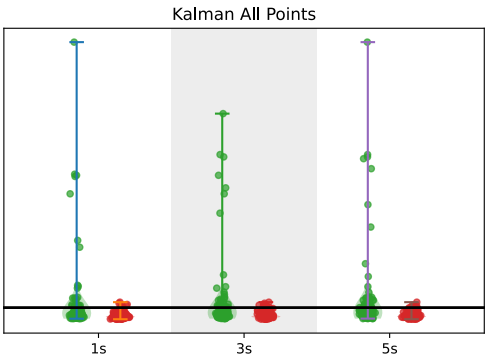
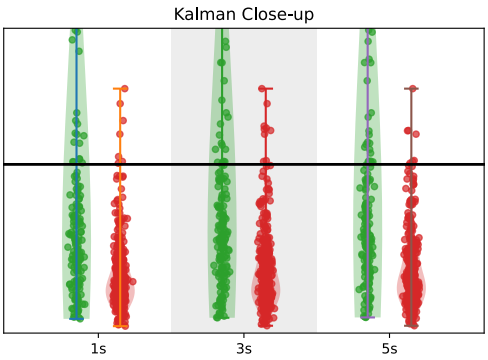


Figure 4.17: Violin plot for Temporal Coherence showing overlap between consistent and inconsistent data.

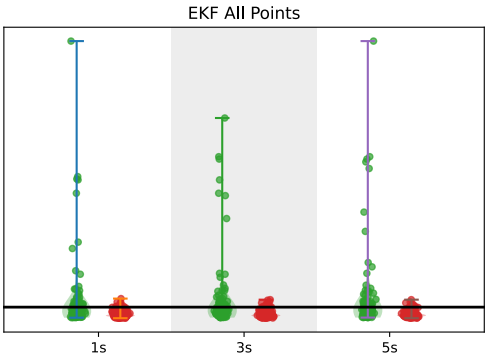


(a) Complete Distribution of Data Points.

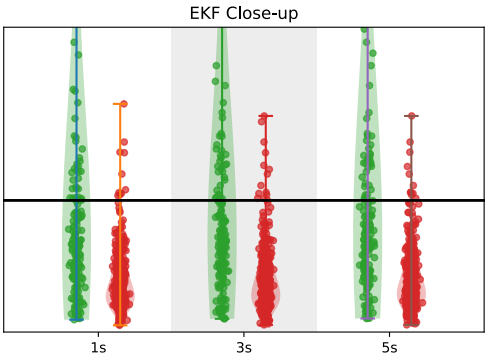


(b) Detailed View of Data Point Distribution.

Figure 4.18: Violin plots showing overlap between consistent and inconsistent data for Kalman Filter, comparing batch sizes of 1 second (left), 3 seconds (middle), and 5 seconds (right).

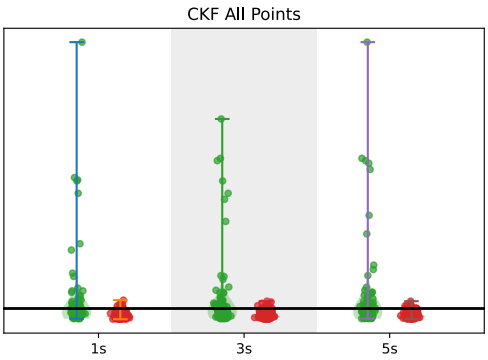


(a) Complete Distribution of Data Points.

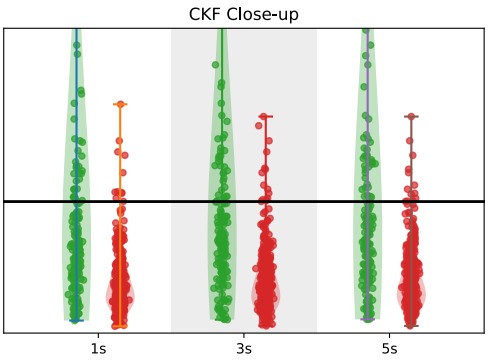


(b) Detailed View of Data Point Distribution.

Figure 4.19: Violin plots showing overlap between consistent and inconsistent data for EKF, comparing batch sizes of 1 second (left), 3 seconds (middle), and 5 seconds (right).

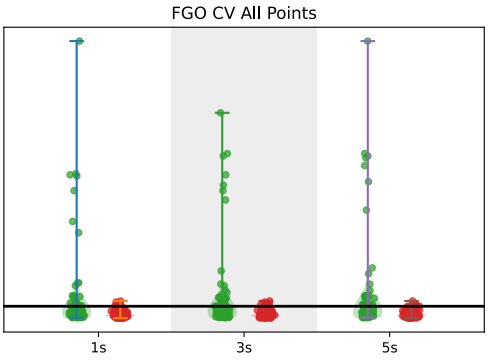


(a) Complete Distribution of Data Points.

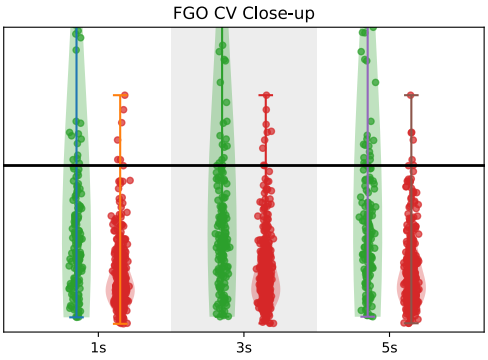


(b) Detailed View of Data Point Distribution.

Figure 4.20: Violin plots showing overlap between consistent and inconsistent data for CKF, comparing batch sizes of 1 second (left), 3 seconds (middle), and 5 seconds (right).

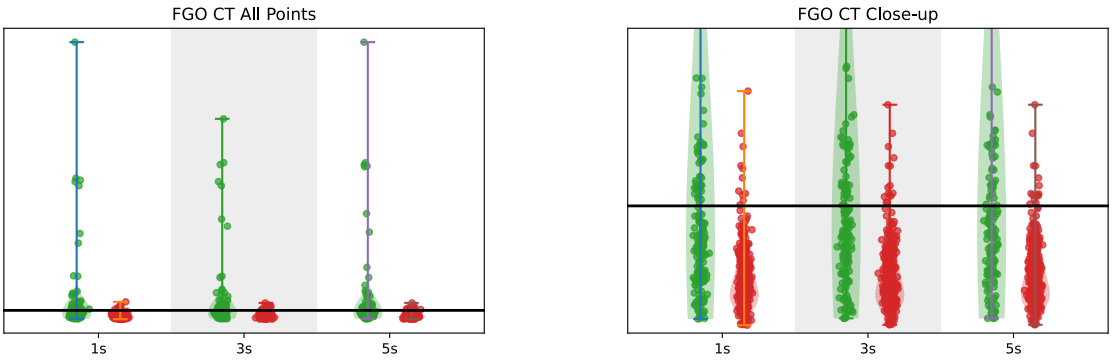


(a) Complete Distribution of Data Points.



(b) Detailed View of Data Point Distribution.

Figure 4.21: Violin plots showing overlap between consistent and inconsistent data for FGO CV, comparing batch sizes of 1 second (left), 3 seconds (middle), and 5 seconds (right).



(a) Complete Distribution of Data Points.

(b) Detailed View of Data Point Distribution.

Figure 4.22: Violin plots showing overlap between consistent and inconsistent data for FGO CT, comparing batch sizes of 1 second (left), 3 seconds (middle), and 5 seconds (right).

4.4 Object type: Trucks

For trucks, tests were done on a dataset of objects where 100 of them contained artificial errors. Tests were done as described in Section 3.5, with $\Psi_{threshold} = \Psi_{99}$ and $\rho = 2.68$. Below follow figures depicting recall for moderate errors Figure 4.23 and recall for severe errors Figure 4.24. As well as precision Figure 4.25 and F_1-score Figure 4.26, Figure 4.27 for moderate and severe errors respectively. Tables with data from the simulation can be found in Appendix B.3.

4.4.1 Recall, Precision and F_1 Score for Trucks

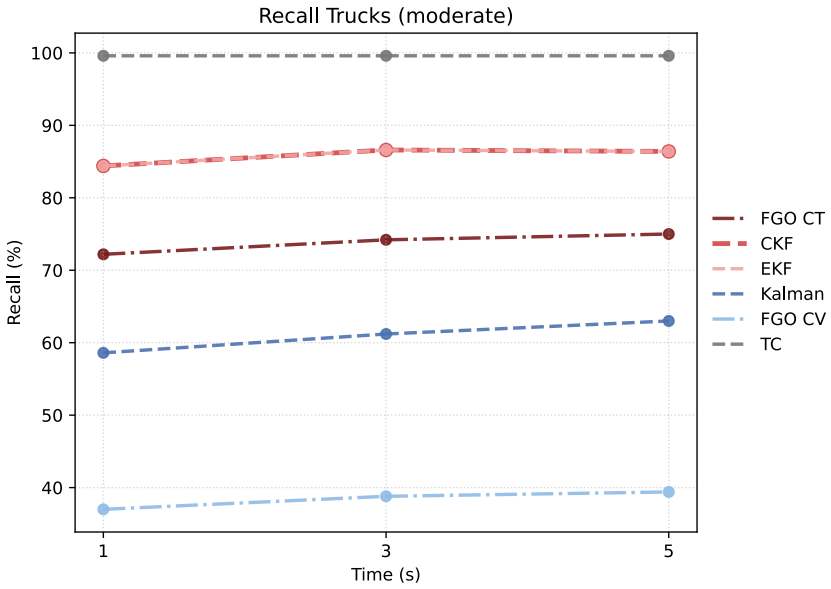


Figure 4.23: Recall for moderate induced errors for trucks.

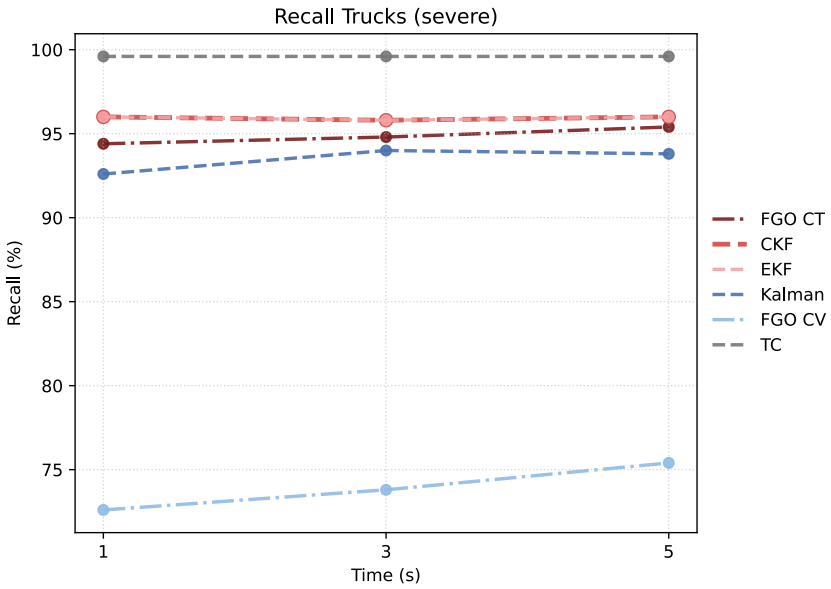


Figure 4.24: Recall for severe induced errors for trucks.

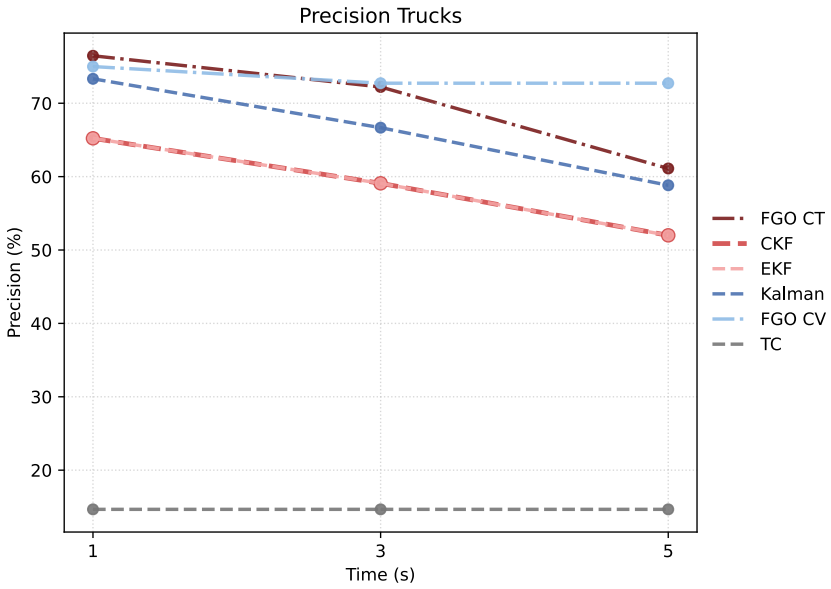


Figure 4.25: Precision for trucks.

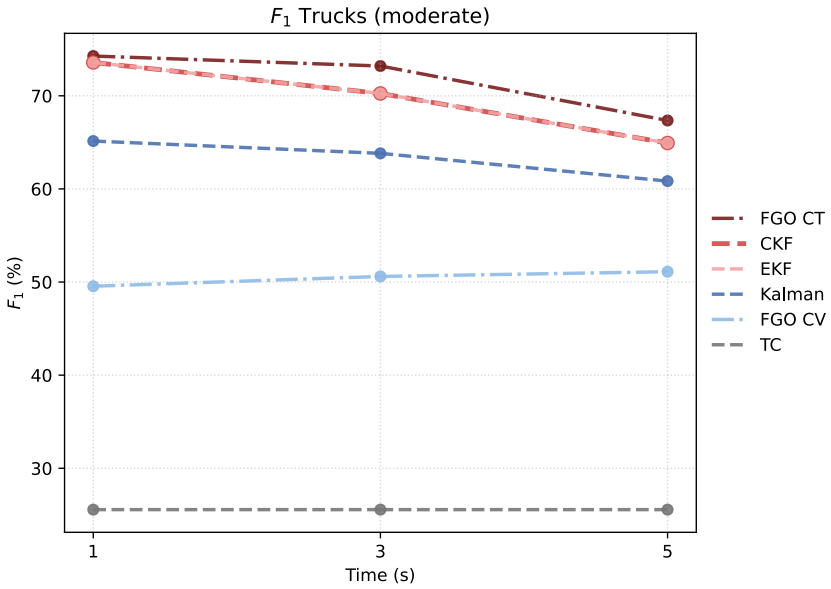


Figure 4.26: F_1 for moderate induced errors for trucks.

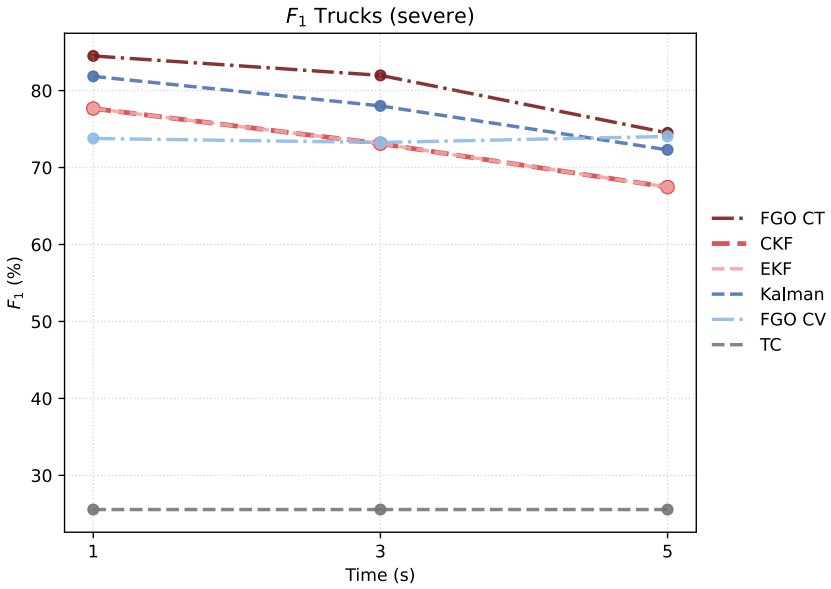


Figure 4.27: F_1 for severe induced errors for trucks.

4.4.2 Separation of Consistent and Inconsistent Data for Trucks

The following depicts each method's ability to differentiate between consistent and inconsistent data, as described in Section 3.5.

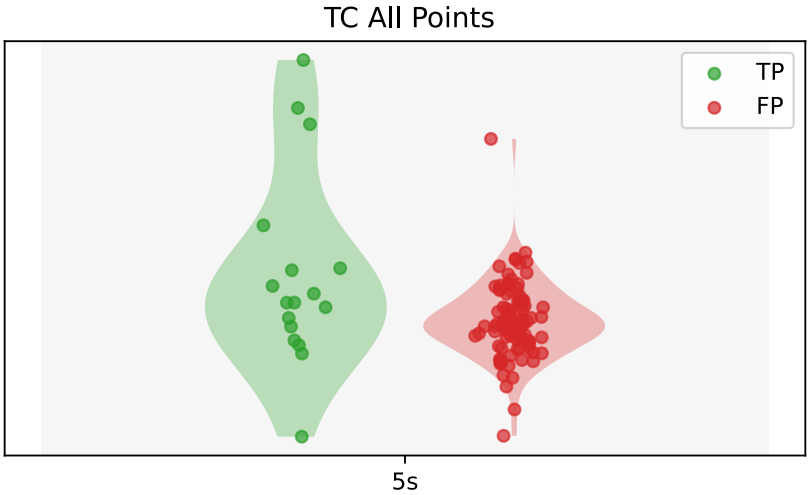
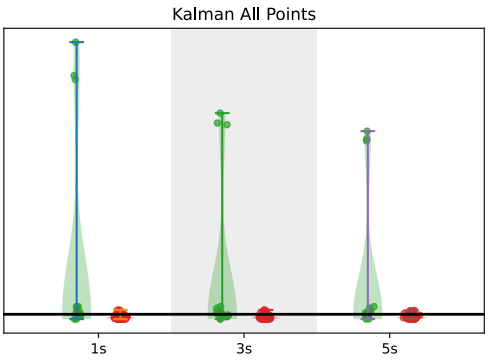
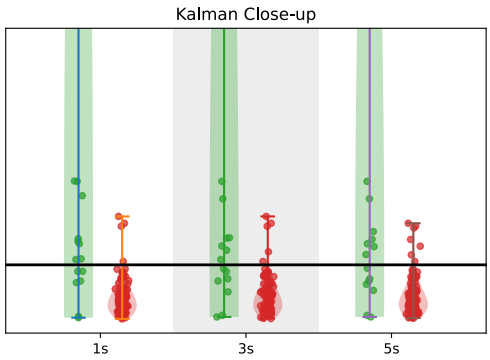


Figure 4.28: Violin plot for Temporal Coherence showing overlap between consistent and inconsistent data.

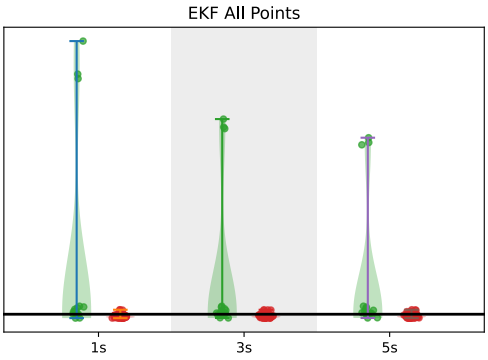


(a) Complete Distribution of Data Points.

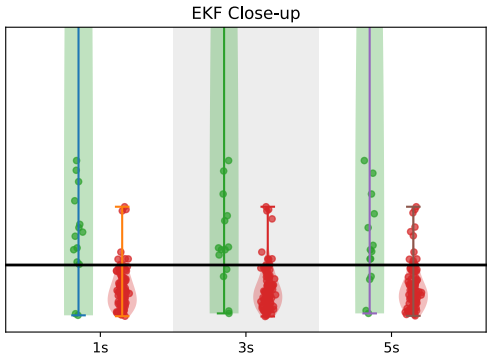


(b) Detailed View of Data Point Distribution.

Figure 4.29: Violin plots showing overlap between consistent and inconsistent data for Kalman Filter, comparing batch sizes of 1 second (left), 3 seconds (middle), and 5 seconds (right).

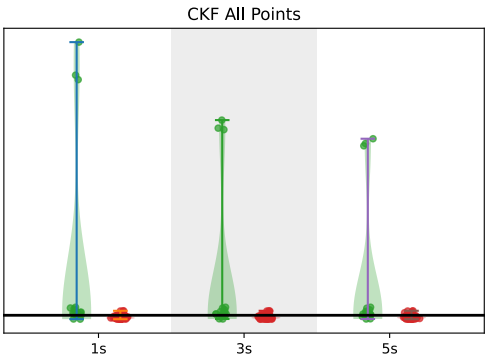


(a) Complete Distribution of Data Points.

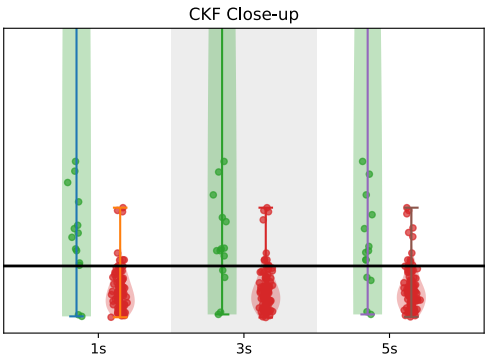


(b) Detailed View of Data Point Distribution.

Figure 4.30: Violin plots showing overlap between consistent and inconsistent data for EKF, comparing batch sizes of 1 second (left), 3 seconds (middle), and 5 seconds (right).

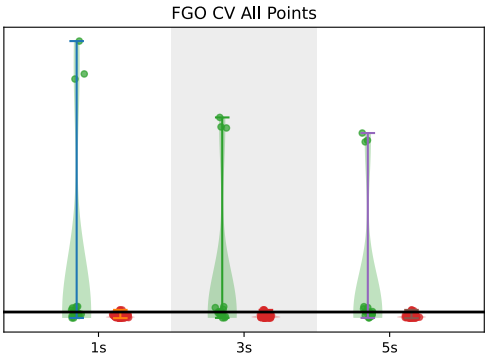


(a) Complete Distribution of Data Points.

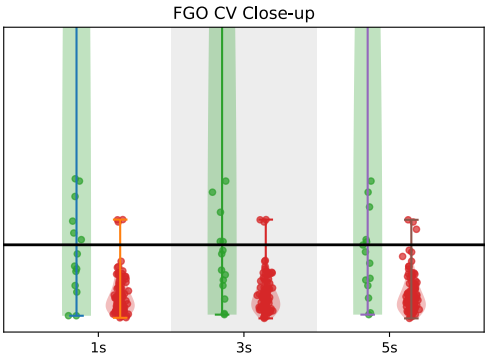


(b) Detailed View of Data Point Distribution.

Figure 4.31: Violin plots showing overlap between consistent and inconsistent data for CKF, comparing batch sizes of 1 second (left), 3 seconds (middle), and 5 seconds (right).

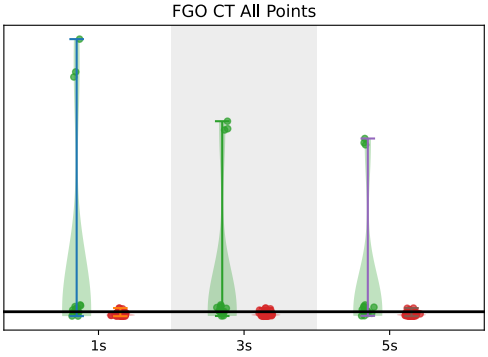


(a) Complete Distribution of Data Points.

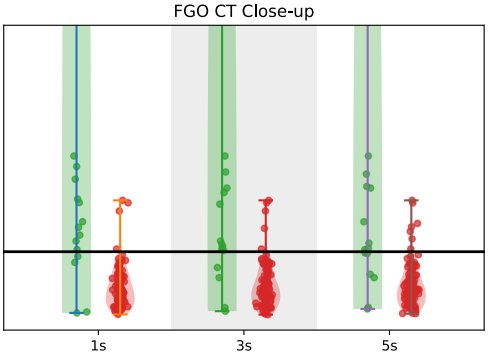


(b) Detailed View of Data Point Distribution.

Figure 4.32: Violin plots showing overlap between consistent and inconsistent data for FGO CV, comparing batch sizes of 1 second (left), 3 seconds (middle), and 5 seconds (right).



(a) Complete Distribution of Data Points.



(b) Detailed View of Data Point Distribution.

Figure 4.33: Violin plots showing overlap between consistent and inconsistent data for FGO CT, comparing batch sizes of 1 second (left), 3 seconds (middle), and 5 seconds (right).

4.5 Computational Time

To evaluate the computational cost of each method, calculations related to transforming data into the world frame and extracting dynamic objects were excluded. These steps are not representative of the processing time within a vehicle system and would therefore provide limited insight into real-world performance. Additionally, the reported computation times were aggregated across all object classes to provide a more comprehensive view of overall system performance.

The total processing time for complete logs is shown in Figure 4.34. The average computation time per batch is presented in Figure 4.35. It should be noted that some batches may contain no detectable objects, resulting in a computation time of 0, while others may include a larger number of objects and therefore require more processing time. As these results are averaged, such variations are not directly visible in the figure. To evaluate the maximum processing time per batch for each method, Figure 4.36 presents the maximum time required by each method.

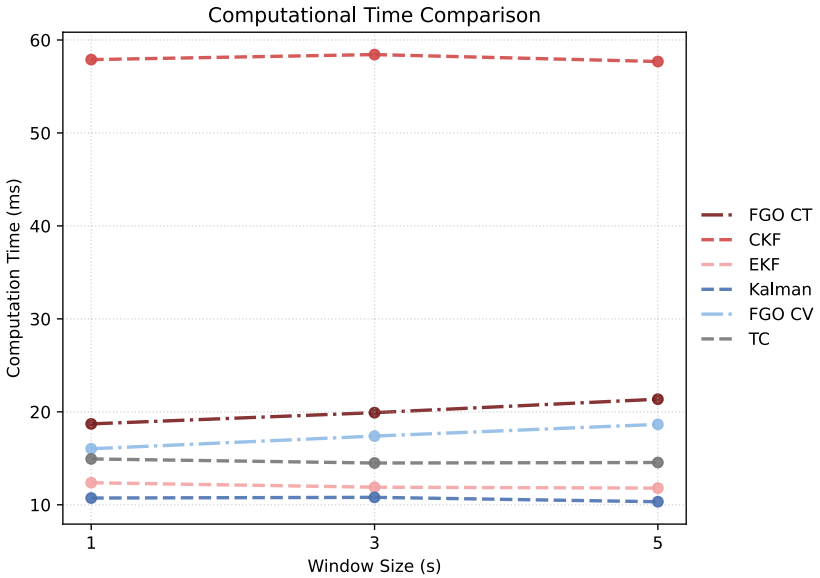


Figure 4.34: Computational Time for each method for all classes.

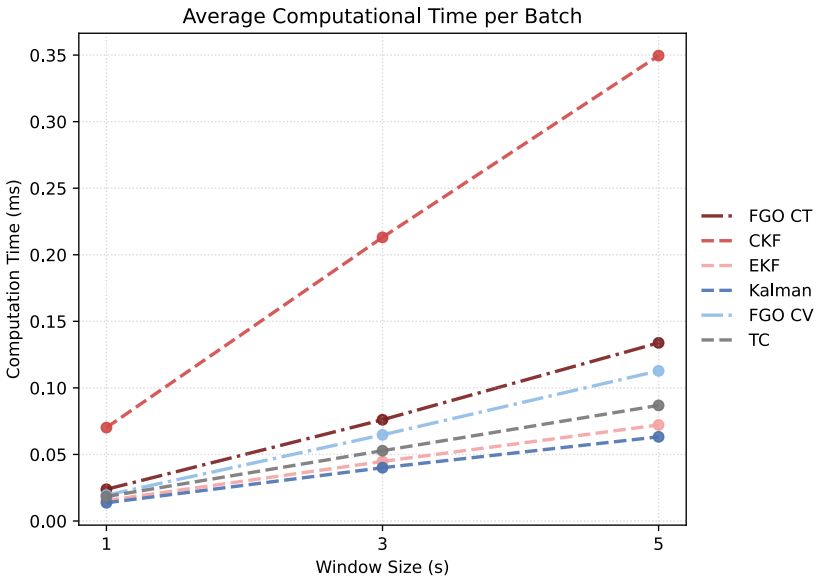


Figure 4.35: Average Computational time per batch.

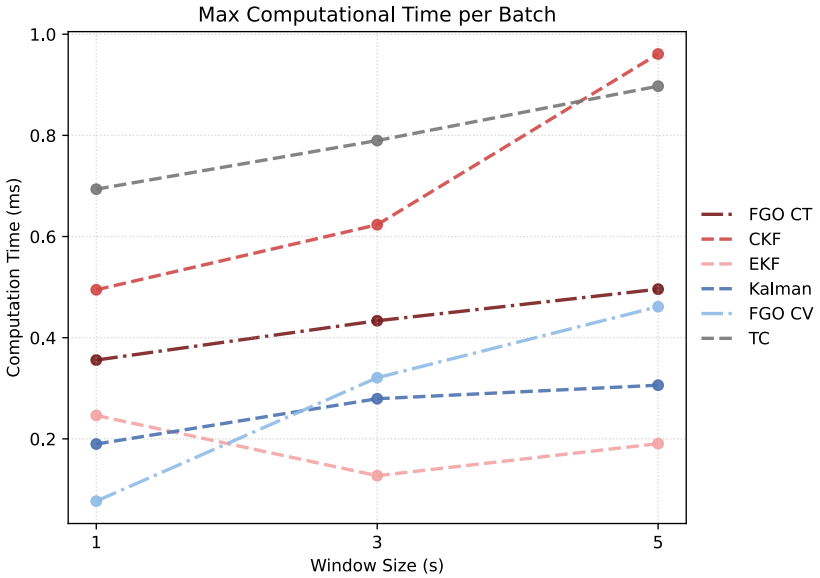


Figure 4.36: Max Computational time per batch.

5
Discussion

This chapter presents a discussion of the results of this thesis, alternative evaluation methods, and the societal, ethical, and environmental aspects.

5.1 Simple Filter

The recall of the simple filter is presented in Table 4.1. The rediscovery performs well with 100 % recall for all classes tested here. The class inconsistencies show that the simple filter performs well for all three classes, but better for cars, and slightly worse for pedestrians. As expected, the simple filter fails to flag the induced motion errors, such as bias, spikes, and drift. Since the recall values of both rediscovery and class inconsistency are quite large, the simple filter appears to be successful.

It should be noted that no class inconsistency was flagged on the data when no inconsistency was induced. Therefore, neither precision nor the F_1-score has been calculated for those inconsistencies. In the simulation and induction of errors, it has not been accounted for that class inconsistencies likely occur simultaneously with motion inconsistencies due to, for example, occlusion. Therefore, it is likely that class-inconsistencies in real data would also capture motion inconsistencies.

In addition, the rediscovery inconsistencies may be justified in cases where an object temporarily obstructs the view. Here, all rediscoveries have been defined as true inconsistencies, which may not be completely accurate. It may be possible to include some evaluation to conclude if an object is blocked. It should, however, be noted that, while that would remove some justified inconsistencies, it could also remove some actual inconsistencies, such as hallucinations of objects, which often occur in front of, or behind other objects.

5.2 Pedestrians

For pedestrians, it is observed in Figures 4.1 and 4.2 that the temporal coherence method has a good recall for both moderate and severe inconsistencies. In addition, it can be seen that the CT model outperforms the CV model in terms of recall for both moderate and severe inconsistencies. Furthermore, while the Kalman Filters and their corresponding FGOs perform fairly equally, it is observed that the Kalman Filters have a slightly better recall.

Figure 4.3 reveals that most of the Kalman Filters and FGOs have a fairly equal precision, while the temporal coherence is divergent and inferior to the rest of the methods. The poor precision of the temporal coherence suggests that the method seems to flag too much as inconsistent.

The poor performance of the temporal coherence method is also observed for the F_1-score of severe inconsistencies, as presented in Figure 4.5. Furthermore, for moderate inconsistencies, as illustrated in Figure 4.4, the F_1-score reveals that the temporal coherence may be somewhat equal to the CV model, and that the CT model is superior.

Figure 4.6 reveals the temporal coherence method's inability to separate data as inconsistent and consistent, where the overlap is almost complete. Thus, the method appears to be inadequate for this purpose. The corresponding violin plots for the Kalman Filters and FGOs, as presented in Figures 4.7 to 4.11, reveal a much better ability to separate into inconsistent and consistent data, although not perfect. Furthermore, the CT model shows a slightly better ability to separate the data than the CV model.

5.3 Cars

As presented in Figure 4.12, for moderate errors, the temporal coherence exhibits a better recall than all other methods. In addition, the FGO with CT outperforms the FGO with CV and all of the Kalman Filters. For more severe errors, as visualized in Figure 4.13, the methods are fairly equal in terms of recall, although it should be noted that while the Kalman Filters and FGOs have an enormous increase in recall for more severe inconsistencies, the temporal coherence has a decrease.

From Figure 4.14, it is revealed that while the Kalman Filters and FGOs have a precision of approximately 80 %, the temporal coherence exhibits a significantly lower precision, indicating that it may flag too much.

For moderate errors, it is apparent from the F_1-score presented in Figure 4.15 that the FGO with CT is preferable as it balances precision and recall the best. In terms of severe errors, it is again visible that the temporal coherence seems to be imbalanced, where it flags to an excessive extent, as presented in Figure 4.16.

The violin plots in Figure 4.17 show what has already been indicated by precision, that the temporal coherence method is unable to separate what is consistent and not. Therefore, the temporal coherence method is inadequate for cars as well. The corresponding plots for the different Kalman Filters and FGOs are illustrated in Figures 4.18 to 4.22, which shows that all of those methods are able to separate the data of inconsistencies and consistencies in an adequate manner, although not perfectly, as an overlap is apparent.

5.4 Trucks

The recall plots for trucks, in Figures 4.23 and 4.24, show superior recall of the temporal coherence methods, with the CT models providing a better recall than the CV, and the Kalman Filters a better one than their corresponding FGOs. In addition, the EKF and CKF methods have the same recall for all window sizes.

While the temporal coherence method was superior in recall, it is inferior in terms of precision, with approximately 15 % as presented in Figure 4.25. In addition, it is observed that for all methods, the precision is best for a smaller window size.

Due to the inferior precision of temporal coherence, the balances F_1-score as illustrated in Figures 4.26 and 4.27 also imply that the temporal coherence is unbalanced and flags too much data as inconsistent.

As for the other two classes, the violin plot of trucks, Figure 4.28, shows that the temporal coherence method is unable to separate the data. The corresponding plots for the Kalman Filters and FGOs, as presented in Figures 4.29 to 4.33, show that they are able to somewhat separate the data but that the CT models are slightly better.

5.5 Computational Time

As observed in Section 4.5, the CKF (Cubature Kalman Filter) generally exhibits the highest computational cost among all methods. However, as shown in Figure 4.36, temporal coherence yields a higher maximum processing time per batch for most window sizes, with the exception of the five-second window. When processing complete data logs, the maximum runtime was approximately 60 ms (see Figure 4.34), with CKF once again demonstrating the highest computational cost. This is primarily due to the additional computations required to generate the sigma points, resulting in increased algorithmic complexity.

Interestingly, the EKF (Extended Kalman Filter) ranks second in terms of computational efficiency, while the standard Kalman Filter achieves the lowest average computational cost (see Figure 4.35, Figure 4.34). For window sizes of 3 and 5, the EKF demonstrates the most favorable worst-case processing time Figure 4.36.

FGO-CV and FGO-CT (Factor Graph Optimization using Constant Velocity and Coordinated Turn models, respectively) occupy an upper-middle position among the evaluated methods in terms of computational cost. Although they are the most computationally complex methods implemented in this thesis, the use of the Georgia Tech Smoothing and Mapping toolbox (GTSAM) [31], which relies on optimized C++ implementations, results in comparatively efficient performance. Consequently, these methods exhibit faster execution than several of the other approaches that were implemented manually. This likely explains why these methods exhibit

the observed computational times, despite their higher theoretical complexity.

The Kalman filter is, in most cases, the least computationally demanding method (see Figure 4.36), with the exception of the maximum batch time. This is expected, as it assumes a linear system model and therefore avoids the additional computations required to handle nonlinearities. Furthermore, the Kalman filter demonstrates greater efficiency than simpler methods such as temporal coherence, which, despite its conceptual simplicity, exhibits a higher computational cost. Although all methods perform computations at each time step, temporal coherence does not retain historical state information or reuse previous computations. Instead, it repeatedly computes bounding boxes for both position and velocity at each step. For objects observed over long durations, this leads to increased computational demand. As a result, its worst-case complexity grows significantly, as illustrated in Figure 4.36.

Nevertheless, all methods are capable of operating within the time constraints of a vehicle system, as even in the worst-case scenario (CKF), the processing of 5 seconds of data does not exceed approximately 1 millisecond.

5.6 Overall

For all three classes, it has been observed that the CT-model performs slightly better than the CV-model, especially in terms of separating the true and false errors, although it is still not performed perfectly. While the difference between the two models was observable, it was marginal, especially for cars and trucks. This is likely as a result of the data mostly being from highways, which means that less abrupt maneuvers will occur, and that the CT model, therefore, will give fairly similar estimates as the CV model. It is therefore expected to see a larger difference between them in data from city-driving, where sharp turns are observed frequently. Additionally, the Temporal Coherence method was deemed inadequate as it was unable to separate the true and false errors.

Furthermore, it was apparent that different threshold percentiles would be reasonable for the different object types, where pedestrians require a smaller one, and cars and trucks a larger one. This is a result of pedestrians being more challenging for perception systems than compact vehicles, such as cars and trucks. Therefore, it is expected that more inconsistencies occur for pedestrians than for cars and trucks.

In terms of window size, the effect is, for most errors, minimal, with some minor loss of performance for the 1-second window for some of the methods. Therefore, the choice of window-size, between these sizes, is almost insignificant, although a choice of 1 second would probably result in a slightly worse performance.

It is also worth highlighting that for most of the sampled false positives, a common denominator has been observed to be a filter/smoother which appear to be overly stiff. For those scenarios, the measurement in position and velocity appear to agree, and also appear reasonable, but the motion model seems to be a bit too restricted

to follow the motion accurately.

Furthermore, visual inspection suggested that the estimates of the Kalman Filters and their corresponding FGOs differ slightly. A major difference is observed in how they handle bias in the velocity, where the Kalman Filters are forced to place the last smoothed estimate at the same position as the last filtered estimate, whereas the FGOs are not limited in the same sense. Additionally, from previous studies [23, 22], it appears that the FGOs are mostly superior for nonlinear motions, and it is therefore possible that the FGO will be better when more abrupt turns are observed than the ones observed on highways.

Overall, the computational complexity of these methods is relatively low. The Kalman filter consistently exhibits the lowest computational cost, which is expected given its linear model assumptions. In contrast, temporal coherence performs poorly in separating true positives from false positives, and its high maximum batch processing time further limits its suitability.

The EKF, CKF, and FGO-based methods demonstrate a strong balance between computational efficiency and detection performance. Consequently, further development of these approaches, such as incorporating more advanced motion models, for example, including acceleration, could prove highly beneficial.

5.7 Other potential methods

While the methods tested have an adequate performance, other potential methods should also be considered.

5.7.1 Only Filter

For the Kalman filters, one method could potentially be to only use the filtering, as opposed to what was performed in this thesis, where smoothing was also used. Then, it would be possible to run it in somewhat real-time, instead of with a lag as a result of the smoothing requiring it. It would, however, likely result in estimates closer to the measurements, including the outliers, which could result in some undetected outliers. Therefore, solely using the filtered estimates would only be relevant if the smoothers are deemed to be too computationally heavy.

5.7.2 Innovation Saturation & Normalized Innovation Squared

Within outlier detection for Kalman, other methods could be used, for example, the ones introduced in Section 3.4.3, Innovation Saturation, and Normalized Innovation Squared. The Innovation Saturation was discarded here as it required a lot of tuning, but could potentially be of interest. It is, however, relevant to point out that an improvement, if obtained, would probably be quite small, and as it is relatively time-consuming to perform the tuning, its practical benefit may be limited. This

limitation is especially important to consider as the perception system may improve and require additional tuning continuously, costing even more resources. It may, however, be possible to perform the tuning by constructing it as an optimization problem similarly to the one used for the tuning of the Kalman filters and FGOs as described in Section 3.4.2.

The Normalized Innovation Squared was also discarded earlier, due to its dependency on the innovation covariance matrix, $\mathbf{S}_k$, which is only part of the filtering stage and not the smoothing. It would therefore not be applicable for FGOs or RTS smoothers. It is therefore not relevant for the methods tested in this report, but if only filtering was used, it could be.

5.7.3 Other Estimation Methods

In addition to the Kalman Filters and FGO, other methods could be used to perform estimations of the states. As aforementioned, the data is not fully Gaussian, and has for the objective function of the tuning been approximated as a Student's t distribution. The filters used have, however, assumed a Gaussian distribution, which may not be fully optimal. A filter which assumes Student's t distribution may therefore be worth exploring in the future, for example, the Robust Student's t-Based Kalman Filter presented in [39]. Another alternative to handle the non-Gaussian behavior observed may be the Particle filter [40]. The Particle filter handles non-Gaussian behavior through Monte Carlo sampling, instead of directly modeling it as a distribution. Again, here the data was limited to highways, and an increase in variation of data may result in an even less Gaussian distribution. As a result of that, Particle filters could be highly relevant, more so than a filter based on Student's t-distribution.

Furthermore, the non-Gaussianity could also be managed for FGO, for example, with robust loss functions [41]. The robust loss functions penalize deviation from probable outliers less, resulting in the outliers not corrupting the estimations as much. This could potentially be useful to get even more accurate estimations to compare measurements with, possibly resulting in better performance. It is, however, to be considered that the use of robust loss functions also introduces a higher computational cost, resulting in an increased time for tuning and testing, which may be too costly to perform in real-time with a lag.

Beyond different types of motion-based methods, multiple other filters and smoothers could be used for the estimation of states based on motion models, for example, Weighted Moving Average and Exponential Smoothing [42]. The advantages of these methods are that they are easy and fast to perform and provide a smoothed estimate, which could be used to detect spikes in the data. They do, however, only perform smoothing and would therefore not be able to detect inconsistencies between position and velocity measurements. Those methods could therefore be relevant if the other methods are deemed to be too computationally heavy and if it would be considered enough to only flag spikes or other fast changes in trajectory.

5.7.4 Learning-based Anomaly Detection

Alternatives to the filters and smoothers aforementioned are learning-based methods. A comprehensive survey on anomaly detection [43] in Deep Learning has summarized multiple methods for different data types, for example, for Multivariate time series data and Vehicle trajectory data. While most of the methods perform well, they also have some disadvantages and limitations, with common ones being that it is expensive to tune hyperparameters, train the networks, and run in real-time, as well as that they require a lot of data. In terms of advantages, compared to filters and smoothers, learning-based methods could generally handle complex behaviors better, whereas the filters and smoothers require a correctly defined motion model. In contrast, the learning-based methods are often not as based on realistic behaviors, which may cause them to overlook some inconsistencies.

5.8 Applicability Across System Types

Here, the system, on which the methods are evaluated, is a learning-based perception system, which could result in slightly different behaviors and errors compared to systems based on, for example, Kalman filters. Learning-based systems are not constrained by motion models, which may allow for adaptiveness to different situations, but may also result in less predictable behavior, for example, with sudden spikes. It may also result in errors where inconsistencies occur between velocity and position. Kalman filters are generally more predictable, but when errors occur, it is as a result of suboptimal construction, with too large or too small measurement or process noise, incorrect motion model, or incorrect noise assumption, where the noise is non-Gaussian. In addition, if the EKF is used, linearization may also introduce errors. While those errors could include some spikes, they will likely not be as extreme as the ones produced by learning-based perception systems. The most apparent errors of the Kalman filter will likely be drift, due to incorrect model and noise assumptions.

While the Kalman Filter is not a learning-based filter and would not use the flagged data as training, it could still be useful to identify failures to evaluate how to improve it in the future. The use of an additional Kalman Filter, which may or may not perform better than one already in use, would be redundant. It could, however, be reasonable to include a filter that performs smoothing afterwards, assuming that is not already done in the one in use, and flag it if the filtered and smoothed estimates have a large disagreement. While this would result in some spikes getting flagged, it would likely fail to detect most of the drift. It could potentially be better to run an FGO on the Kalman estimates instead, as it is not restricted by the last estimate of the Kalman filter, as the RTS smoother is.

A learning-based perception system would presumably have the most benefit of this type of outlier detection, compared to Kalman filters. This is a result of Kalman

filters already being restricted by motion models, which are used for the detection, while learning-based systems are not, and are solely dependent on previous experience.

5.9 Societal, ethical and ecological aspects

The United Nations Sustainable Development Goals most relevant to this thesis are Goal 11 and Goal 12. Goal 11, Sustainable Cities and Communities, aims to "make cities and human settlements inclusive, safe, resilient and sustainable" [44]. This objective aligns closely with the focus of this thesis, as enhancing the robustness of automated driving systems through the identification of informative training data directly contributes to improved road safety. By enabling perception models to better detect and handle uncertain or unfamiliar situations, such systems can make more reliable decisions, ultimately reducing the risk of accidents.

Goal 12, Responsible Consumption and Production, aims to "ensure sustainable consumption and production patterns" [44]. This goal is relevant in the context of autonomous driving, as such systems depend on large volumes of sensor data and computational resources. Improving the efficiency of data selection by prioritizing informative or high-value data for training can help reduce unnecessary data processing and storage. In turn, this contributes to more resource-efficient model development. At the same time, it highlights the importance of designing energy-efficient data handling and training pipelines to support sustainable system development.

Beyond the UN sustainability goals, several ethical aspects must be considered. The use of customer data introduces significant GDPR and privacy concerns, especially when sensor modalities such as cameras inadvertently capture pedestrians or other road users who may be unaware that data collection is taking place. Furthermore, autonomous systems inherently raise questions regarding safety trade-offs, such as how decisions balance the protection of the vehicle occupants versus pedestrians or other drivers. Although this project does not directly address decision-making policies, the methods it developed may indirectly influence the reliability of perception systems used during such critical moments.

Overall, the ethical and sustainability considerations emphasize both the societal benefits, such as safer mobility and reduced environmental impact, and the potential risks-privacy violations, systemic biases, and dependence on AI-driven decisions. A responsible approach requires careful handling of data, transparency, and consideration of long-term societal consequences.

6
Conclusion

This thesis aimed to investigate different methods to detect inconsistencies in a learning-based perception system. The methods evaluated were different Kalman Filters, Factor Graph Optimization (FGO), and Temporal Coherence. The Kalman Filters used were a regular Kalman Filter with a CV model, an EKF with a CT model, and a CKF with a CT model. All the Kalman Filters used RTS smoothing to give a more feasible estimation compared to the estimation of solely filtering the data. The Factor Graph Optimization was performed both with a CV and a CT model, and Temporal Coherence was defined to compare the Intersection over Union (IoU), calculated from position to the IoU calculated from velocity. Furthermore, the Kalman Filters and the FGOs were not tuned manually, but with an optimization algorithm to accommodate potential improvements of the perception system. The thesis was limited to data collected in the vicinity of Gothenburg, with most of it collected on highways, and the testing was only performed on pedestrians, cars, and trucks.

It was concluded that the CT-model performs slightly better than the CV-model, with the most apparent difference being observed for moderate inconsistencies. Although the difference was observable, it is possible that data with more city-driving would result in a larger difference between the models, as that would result in more turns, which is where the CT-model is expected to outperform the CV-model. In the comparison between Kalman Filters and FGOs, they appeared to be somewhat equal in the capability of separating the inconsistencies, although the FGOs seemed better at estimating the true states. The Temporal Coherence method showed poor capability at identifying inconsistencies and was therefore deemed insufficient for this purpose.

The research questions posed in this thesis can be answered as follows:

1. *What methods can be used to identify, filter, and select informative data from large vehicle fleets?*

   The Kalman Filters and Factor Graph Optimization methods were shown to be efficient, with both of them acquiring a precision of approximately 80 %. The temporal coherence method was deemed insufficient since it was unable to accurately separate inconsistent and consistent data.

2. *Could the proposed methods be implemented in real-time for a given vehicle?*

All of the methods show a significantly shorter computational time than the corresponding windows. Therefore, all of the methods show potential for being implemented in real-time.

3. *How do the different methods compare with respect to detection performance and computational complexity, and what trade-offs can be identified between them?*

   The Kalman Filters and Factor Graph Optimization methods constructed with the same motion model have a similar detection performance, although the Factor Graph Optimization, in general, performs slightly better. The largest difference was observed for different motion models, where the Coordinated Turn model had a better performance than the Constant Velocity model, likely due to its ability to better model the behaviors.

   With respect to the computational complexity, most methods show similar performance. The primary exception is the Cubature Kalman Filter, which exhibits the highest overall computational cost.

   Considering both detection performance and computational cost, the Factor Graph Optimization method with the Coordinated Turn model is the most favorable method, as it provides the best detection performance without a significant increase in computational cost.

The Kalman Filters and FGOs have been proven to work adequately in separating consistent and inconsistent data, especially the CT-model, although not perfectly. Therefore, it has the potential to be used as an online model evaluator. It should also be considered that, with current construction, it will sometimes flag data that is consistent and fail to flag some data that is inconsistent. As a result, some consistent data would be provided as training data for the perception system, which might be acceptable, but if not, additional filtering would be required after the collection of data. While this is a problem to consider, it could potentially be mitigated through improved modeling of the different objects.

Finally, this thesis contributes a framework for identifying informative data in learning-based perception systems through the application of anomaly detection techniques. By implementing and systematically evaluating methods based on Temporal Coherence, Kalman Filters, and Factor Graph Optimization, the work provides insight into their practical applicability for large-scale vehicle data collection and online model evaluation. The results demonstrate that motion-model-based approaches are effective for detecting inconsistencies, with Factor Graph Optimization using a coordinated turn model emerging as the most promising approach among the methods studied.

6.1 Future Works

6.1.1 World Frame Reset

In this thesis, the world frame was fixed, given the initial position of the ego-vehicle. As described in Section 3.1, this could result in an accumulated drift, especially for longer drives and for objects observed over a long duration. This could potentially be resolved by performing a reset on the world frame regularly. This reset will need to be performed at a high enough frequency to minimize drift. It is, however, to be noted that the Kalman Filters and FGOs also will need a reset, potentially causing less accurate estimates. It should therefore be explored how much a reset affects the performance and what reset frequency is preferable.

6.1.2 Extension of data

For this project, the data was limited to mostly highways, meaning that the filters are not fully applicable to other datasets, where the process noise may need to be larger, or requires more complex motion models. Therefore, an extension of the data with a larger environmental and dynamical diversity would be of interest.

In addition to being limited to highways, the number of objects of some classes was inadequate to be able to perform both tuning and testing. For learning-based perception systems, it is expected to have a better performance on frequently seen objects, which also behave predictably, such as cars, and a worse performance on objects observed less frequently or with less predictable motion, such as pedestrians and animals. Therefore, an extension of different object classes would be of interest.

Furthermore, additional data increases the likelihood of class-inconsistencies, which could be of interest to explore the simple filter and whether it is able to catch motion inconsistencies consequently.

6.1.3 Evaluate Other Motion and Measurement Models

As aforementioned, it was observed that the estimated motions were a bit too conservative, where some reasonable acceleration was observed in the measurements, but was not included in the estimates. Those scenarios could potentially result in false positives, which could potentially be avoided with another motion model that better models that behavior. It is therefore reasonable to further evaluate different motion models, for example, models with higher derivative order, such as the Constant Acceleration model [45]. In addition, since the movement of the dynamic objects can behave differently, a smoother that can change between different models instead of only one, as presented in [46] for Kalman Filters and in [47] for Factor Graph Optimization, would be desirable. These models could include both different types of motion models, such as CV and CT, but also different noise levels to be able to handle both smooth motions, such as the one observed on highways, and more dynamic motions, as seen in city-driving, with frequent stops and turns.

In addition, it is possible that different velocities and driving patterns result in different measurement noise. Therefore, it may be relevant to evaluate if interactive models [46, 47] could increase performance. For this report, only measurements of position and velocity have been used and evaluated. It could also be of interest to include other measurements, which would require other measurement models.

6.1.4 Inspect Other Parts of World Model

As mentioned above, additional measurements of the dynamic objects could be included in the evaluation. Furthermore, for this project, the evaluation has been focused on detecting inconsistencies in dynamic objects, such as vehicles and pedestrians, but the world model does not solely consist of dynamic objects. Therefore, inconsistencies of other parts of the world model, for example, static objects, could be of interest to detect. Inconsistencies that could be of interest in static objects could be both in terms of change of position, but also in terms of interpretation, for example, if a sign has been interpreted differently at different time instances.

6.1.5 Additional Outlier Detection Methods

As described in Section 5.7, other motion-based methods could be interesting to evaluate, especially those that perform well on non-Gaussian noise, such as the Particle Filter and FGO with robust loss. This is particularly of interest with an extension of data, since that likely will introduce even more non-Gaussian noise.

Furthermore, it may be reasonable to compare the proposed methods with some learning-based methods to fully compare the different methods. It could also be of interest to explore if a combination of these methods could increase the performance.

6.1.6 Buffer Prioritization

While in reality not possible, here an infinite buffer has been assumed. If a finite buffer were used, some selection would be required as to which flagged error to save. This could be conducted with a simple method, for example, by selecting at random or the first error observed. If the method were used in fleet vehicles, a high volume of data would be collected, meaning that even with a simple method, as proposed above, the data would be diverse and include some of the more critical errors. This type of collection method may therefore be sufficient.

It could, however, be of interest to select data with a more sophisticated approach, such that data of higher importance is selected, which reduces the risk of missing important information. This could be performed in multiple different ways, for example, the measurement used here, Ψ, as defined earlier in Equation (3.47), or time to collision (TTC) or its extensions [48]. As previously discussed, some classes may be of more interest, such as the ones with less predictable motion or those classes that are observed less frequently. In addition, some classes may have a higher risk of getting hurt by the ego vehicle, for example, pedestrians, and some have a higher

risk of hurting the ego vehicle, such as trucks. Therefore, the classes may be used as a prioritization criterion as well.

While all aforementioned criteria could be useful separately, a combination of them would probably be most successful, for example, by using:

$$\textit{Priority-score} = \sum_{e \in \mathcal{E}} w_e e \tag{6.1}$$

where w_e is a weight to scale and let some measures in $\mathcal{E}$ have higher importance. $\mathcal{E}$ could include TTC or its extensions, our measure Ψ, predictability of the motion of a class, frequency of class observation, and risk of accident. The latter one could possibly be both in terms of what the impact would be with the specific class, where pedestrians are less protected and more at risk from the ego, while trucks are a greater danger for the ego, and in how large the impact would be based on velocity.

It should be noted that the Ψ-values will be slightly different for different classes, with different thresholds, denoted as $\Psi_{threshold}$ and defined in Equation (3.48), therefore it might be relevant to normalize it such that, $\frac{\Psi}{\Psi_{threshold}}$, is used instead of Ψ.

6.1.7 Duplicate Object Detection

Here, only dynamic objects with one constant identifier have been evaluated. It is, however, possible that an object has duplicate identifiers, which should be considered an inconsistency, but will be undetected for the current methods. This could be observed as an object first having one identifier and then switching to another, possibly due to it being occluded for some time, or if two identifiers are observed almost entirely at the same position. The first one could potentially be identified by performing predictions from the last detection of an identifier and finding potential matches of other identifiers. The latter could probably be evaluated by investigating the overlap between objects, with a threshold probably decided by the class types.

Bibliography

[1] Lincan Li et al. *Data-Centric Evolution in Autonomous Driving: A Comprehensive Survey of Big Data System, Data Mining, and Closed-Loop Technologies*. 2024. arXiv: 2401.12888 [cs.RO]. URL: https://arxiv.org/abs/2401.12888.

[2] John Houston et al. *One Thousand and One Hours: Self-driving Motion Prediction Dataset*. 2020. arXiv: 2006.14480 [cs.CV]. URL: https://arxiv.org/abs/2006.14480.

[3] Zhenhai Gao et al. "Collision Risk Assessment for Intelligent Vehicles Considering Multi-Dimensional Uncertainties". In: *IEEE Access* 12 (2024), pp. 57780–57795. DOI: 10.1109/ACCESS.2024.3354383.

[4] Ke Wang et al. "Application of Uncertainty to Out-of-Distribution Detection for Autonomous Driving Perception Safety". In: *IEEE Transactions on Intelligent Transportation Systems* 26.8 (2025), pp. 11276–11293. DOI: 10.1109/TITS.2025.3557996.

[5] Henry X. Liu and Shuo Feng. "Curse of rarity for autonomous vehicles". In: *Nature Communications* 15.1 (2024), p. 4808. DOI: 10.1038/s41467-024-49194-0.

[6] Scott Drew Pendleton et al. "Perception, Planning, Control, and Coordination for Autonomous Vehicles". In: *Machines* 5.1 (2017), p. 6. DOI: 10.3390/machines5010006.

[7] Pei Sun et al. *Scalability in Perception for Autonomous Driving: Waymo Open Dataset*. 2020. arXiv: 1912.04838 [cs.CV]. URL: https://arxiv.org/abs/1912.04838.

[8] Leandro Masello et al. "From Traditional to Autonomous Vehicles: A Systematic Review of Data Availability". In: *Transportation Research Record* 2676.4 (2022), pp. 161–193. DOI: 10.1177/03611981211057532. eprint: https://doi.org/10.1177/03611981211057532. URL: https://doi.org/10.1177/03611981211057532.

[9] Dan Pelleg and Andrew Moore. "Active Learning for Anomaly and Rare-Category Detection". In: *Advances in Neural Information Processing Systems*. Ed. by L. Saul, Y. Weiss, and L. Bottou. Vol. 17. MIT Press, 2004. URL: https://proceedings.neurips.cc/paper_files/paper/2004/file/8c59fd6fbe0e9793ec2b27971221cace-Paper.pdf.

[10] Javad Zolfaghari Bengar et al. "Temporal Coherence for Active Learning in Videos". In: *CoRR* abs/1908.11757 (2019). arXiv: 1908.11757. URL: http://arxiv.org/abs/1908.11757.

[11] Hamid Rezatofighi et al. “Generalized Intersection Over Union: A Metric and a Loss for Bounding Box Regression”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019.

[12] Hao Zhang, Cong Xu, and Shuaijie Zhang. “Inner-IoU: more effective intersection over union loss with auxiliary bounding box”. In: *arXiv preprint arXiv:2311.02877* (2023).

[13] Maria Isabel Ribeiro. “Kalman and extended kalman filters: Concept, derivation and properties”. In: *Institute for Systems and Robotics* 43.46 (2004), pp. 3736–3741.

[14] Simo Särkkä. *Bayesian Filtering and Smoothing*. Vol. 3. Institute of Mathematical Statistics Textbooks. Cambridge: Cambridge University Press, 2013.

[15] Fajar Astuti Hermawati et al. “Trajectory Prediction Using Kalman Filter Method As Collision Risk Assessment On Autonomous Tram”. In: *2023 14th International Conference on Information Communication Technology and System (ICTS)*. 2023, pp. 249–254. DOI: 10.1109/ICTS58770.2023.10330883.

[16] E.A. Wan and R. Van Der Merwe. “The unscented Kalman filter for nonlinear estimation”. In: *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No.00EX373)*. 2000, pp. 153–158. DOI: 10.1109/ASSPCC.2000.882463.

[17] Ienkaran Arasaratnam and Simon Haykin. “Cubature Kalman Filters”. In: *IEEE Transactions on Automatic Control* 54.6 (2009), pp. 1254–1269. DOI: 10.1109/TAC.2009.2019800.

[18] Herbert E Rauch, Fong Tung, and Charlotte T Striebel. “Maximum likelihood estimates of linear dynamic systems”. In: *AIAA journal* 3.8 (1965), pp. 1445–1450.

[19] Daniel Svensson, Martin Ulmke, and Lars Hammarstrand. “Multitarget Sensor Resolution Model and Joint Probabilistic Data Association”. In: *IEEE Transactions on Aerospace and Electronic Systems* 48.4 (2012), pp. 3418–3434. DOI: 10.1109/TAES.2012.6324722.

[20] Frank R Kschischang, Brendan J Frey, and H-A Loeliger. “Factor graphs and the sum-product algorithm”. In: *IEEE Transactions on information theory* 47.2 (2001), pp. 498–519.

[21] Frank Dellaert and Michael Kaess. “Factor Graphs for Robot Perception”. In: *Foundations and Trends in Robotics* 6.1-2 (Aug. 2017), pp. 1–139. ISSN: 1935-8253. DOI: 10.1561/2300000043. eprint: https://www.emerald.com/ftrob/article-pdf/6/1-2/1/11046280/2300000043en.pdf. URL: https://doi.org/10.1561/2300000043.

[22] Zhengdao Li et al. “Intelligent environment-adaptive GNSS/INS integrated positioning with factor graph optimization”. In: *Remote Sensing* 16.1 (2023), p. 181. URL: https://www.mdpi.com/2072-4292/16/1/181.

[23] Weisong Wen et al. “Factor graph optimization for GNSS/INS integration: A comparison with the extended Kalman filter”. In: *NAVIGATION* 68.2 (2021), pp. 315–331. DOI: https://doi.org/10.1002/navi.421. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/navi.421. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/navi.421.

[24] Jun Dai et al. “UAV localization algorithm based on factor graph optimization in complex scenes”. In: *Sensors* 22.15 (2022), p. 5862. URL: https://www.mdpi.com/1424-8220/22/15/5862.

[25] Amjad Hussain Magsi and Luis Enrique Díez Blanco. “Performance Analysis of FGO Windowing Strategies for PDR+GNSS Fusion Architecture”. In: *IEEE Access* 13 (2025), pp. 113461–113481. DOI: 10.1109/ACCESS.2025.3584664.

[26] Peter I. Frazier. *A Tutorial on Bayesian Optimization*. 2018. arXiv: 1807.02811 [stat.ML]. URL: https://arxiv.org/abs/1807.02811.

[27] Eric Brochu, Matthew W. Hoffman, and Nando de Freitas. “Portfolio Allocation for Bayesian Optimization”. In: *CoRR* abs/1009.5419 (2010). arXiv: 1009.5419. URL: http://arxiv.org/abs/1009.5419.

[28] scikit-optimize contributors. *skopt.gp_minimize*. URL: https://scikit-optimize.github.io/stable/modules/generated/skopt.gp_minimize.html (visited on 05/20/2026).

[29] *Road vehicles — Functional safety — Part 1: Vocabulary*. International Organization for Standardization, 2018. URL: https://www.iso.org/standard/68383.html.

[30] Sean Gillies et al. *Shapely: manipulation and analysis of geometric objects*. https://shapely.readthedocs.io. 2023.

[31] Frank Dellaert. “Factor graphs and GTSAM: A hands-on introduction”. In: *Georgia Institute of Technology, Tech. Rep* 2.4 (2012).

[32] Yuanxi Yang and Weiguang Gao. “An Optimal Adaptive Kalman Filter”. In: *Journal of Geodesy* 80.4 (2006), pp. 177–183. ISSN: 1432-1394. DOI: 10.1007/s00190-006-0041-0. URL: https://doi.org/10.1007/s00190-006-0041-0.

[33] Zhaozhong Chen et al. “Kalman Filter Auto-Tuning With Consistent and Robust Bayesian Optimization”. In: *IEEE Transactions on Aerospace and Electronic Systems* 60.2 (2024), pp. 2236–2250. DOI: 10.1109/TAES.2024.3350587.

[34] Vinay A. Bavdekar, Naresh N. Nandola, and Sachin C. Patwardhan. “Maximum likelihood estimation of noise covariance matrices for state estimation of autonomous hybrid systems”. In: *Computers Chemical Engineering* 94 (2016), pp. 28–44. ISSN: 0098-1354. DOI: https://doi.org/10.1016/j.compchemeng.2016.07.010. URL: https://www.sciencedirect.com/science/article/pii/S0098135416302307.

[35] Huazhen Fang, Mulugeta A. Haile, and Yebin Wang. “Robust Extended Kalman Filtering for Systems With Measurement Outliers”. In: *IEEE Transactions on Control Systems Technology* 30.2 (2022), pp. 795–802. DOI: 10.1109/TCST.2021.3077535.

[36] Yaakov Bar-Shalom, X Rong Li, and Thiagalingam Kirubarajan. *Estimation with applications to tracking and navigation: theory algorithms and software*. John Wiley & Sons, 2001.

[37] David M. W. Powers. *Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation*. 2020. arXiv: 2010.16061 [cs.LG]. URL: https://arxiv.org/abs/2010.16061.

[38] Matthew Halpern et al. “One Size Does Not Fit All: Quantifying and Exposing the Accuracy-Latency Trade-Off in Machine Learning Cloud Service APIs via Tolerance Tiers”. In: *2019 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. IEEE, Mar. 2019, pp. 34–47. DOI: 10.1109/ispass.2019.00012. URL: http://dx.doi.org/10.1109/ISPASS.2019.00012.

[39] Yulong Huang et al. “A Novel Robust Student’s t-Based Kalman Filter”. In: *IEEE Transactions on Aerospace and Electronic Systems* 53.3 (2017), pp. 1545–1554. DOI: 10.1109/TAES.2017.2651684.

[40] Zhe Chen et al. “Bayesian filtering: From Kalman filters to particle filters, and beyond”. In: *Statistics* 182.1 (2003), pp. 1–69.

[41] Zhexin Xu et al. “Implementing Robust M-Estimators with Certifiable Factor Graph Optimization”. In: *arXiv preprint arXiv:2603.20932* (2026).

[42] Ajiono Ajiono and Taqwa Hariguna. “Comparison of three time series forecasting methods on linear regression, exponential smoothing and weighted moving average”. In: *International Journal of Informatics and Information Systems* 6.2 (2023), pp. 89–102.

[43] Haoqi Huang et al. “Deep Learning Advancements in Anomaly Detection: A Comprehensive Survey”. In: *IEEE Internet of Things Journal* 12.21 (2025), pp. 44318–44342. DOI: 10.1109/JIOT.2025.3585884.

[44] United Nations. *The 17 Goals*. URL: https://sdgs.un.org/goals. (accessed: 01.12-2025).

[45] X. Rong Li and V.P. Jilkov. “Survey of maneuvering target tracking. Part I. Dynamic models”. In: *IEEE Transactions on Aerospace and Electronic Systems* 39.4 (2003), pp. 1333–1364. DOI: 10.1109/TAES.2003.1261132.

[46] Anthony F Genovese. “The interacting multiple model algorithm for accurate state estimation of maneuvering targets”. In: *Johns Hopkins APL technical digest* 22.4 (2001), pp. 614–623.

[47] Varun Agrawal and Frank Dellaert. “Variable Elimination in Hybrid Factor Graphs for Discrete-Continuous Inference & Estimation”. In: *arXiv preprint arXiv:2601.00545* (2026).

[48] Michiel M. Minderhoud and Piet H.L. Bovy. “Extended time-to-collision measures for road traffic safety assessment”. In: *Accident Analysis Prevention* 33.1 (2001), pp. 89–97. ISSN: 0001-4575. DOI: https://doi.org/10.1016/S0001-4575(00)00019-1. URL: https://www.sciencedirect.com/science/article/pii/S0001457500000191.

A

Derivation of objective function

With the objective function being a maximum likelihood, factorization can be performed accordingly:

$$\begin{aligned}
\arg\max_{\boldsymbol{\sigma}} \quad L(\mathbf{Q},\mathbf{R}|\mathbf{x},\mathbf{z}) &= \arg\max_{\boldsymbol{\sigma}} \quad p(\mathbf{x},\mathbf{z}|\mathbf{Q},\mathbf{R}) \\
&= \arg\max_{\boldsymbol{\sigma}} \quad p(\mathbf{z}|\mathbf{x},\mathbf{Q},\mathbf{R}) \cdot p(\mathbf{x}|\mathbf{Q},\mathbf{R}) \\
&= \arg\max_{\boldsymbol{\sigma}} \quad p(\mathbf{z}|\mathbf{x},\mathbf{R}) \cdot p(\mathbf{x}|\mathbf{Q}) \\
&= \arg\max_{\boldsymbol{\sigma}} \quad p(\mathbf{e}_{\mathbf{z}}|\mathbf{R}) \cdot p(\mathbf{e}_{\mathbf{x}}|\mathbf{Q}) \\
&= \arg\max_{\boldsymbol{\sigma}} \quad \prod_{i=1}^{M}\Big(\prod_{k=1}^{N_i} p(\mathbf{e}_{\mathbf{z}_{k,i}}|\mathbf{R}) \cdot p(\mathbf{e}_{\mathbf{x}_{k,i}}|\mathbf{Q})\Big)
\end{aligned} \tag{A.1}$$

where M is the number of dynamic objects and N_i is the number of time samples dynamic object i was observed for. The array $\boldsymbol{\sigma}$ is the defined as:

$$\boldsymbol{\sigma} = (\sigma_j^{\tau})_{\tau\in\mathcal{T},j\in\mathcal{J}_m^{\tau}}$$

where $\tau \in \mathcal{T} \coloneqq \{meas, proc\}$ defines if it is process or measurement noise and $m \in \mathcal{M} = \{CV, CT\}$, defines which motion model it is for. The sets $\mathcal{J}_m^{\tau}$, are defined as:

$$\begin{aligned}
\mathcal{J}_{CV}^{meas} = \mathcal{J}_{CT}^{meas} &\coloneqq \{p, v\} \\
\mathcal{J}_{CV}^{proc} \coloneqq \{v\}, \quad \mathcal{J}_{CT}^{proc} &\coloneqq \{v, \omega\}
\end{aligned} \tag{A.2}$$

and represents what types of variables the corresponding noise is directly applied to.
Using log-likelihood and the average of all per-object averages:

$$\begin{aligned}
&\arg\max_{\boldsymbol{\sigma}} \quad \log\Big(\frac{1}{M}\prod_{i=1}^{M}\Big(\frac{1}{N_i}\prod_{k=1}^{N_i} p(\mathbf{e}_{\mathbf{z}_{k,i}}|\mathbf{R}) \cdot p(\mathbf{e}_{\mathbf{x}_{k,i}}|\mathbf{Q})\Big)\Big) \\
=&\arg\max_{\boldsymbol{\sigma}} \quad \frac{1}{M}\sum_{i=1}^{M}\Big(\frac{1}{N_i}\sum_{k=1}^{N_i}\Big(\log(p(\mathbf{e}_{\mathbf{z}_{k,i}} \mid \mathbf{R})) + \log(p(\mathbf{e}_{\mathbf{x}_{k,i}} \mid \mathbf{Q}))\Big)\Big)
\end{aligned} \tag{A.3}$$

The motion noise is assumed to be of normal distribution, resulting in:

$$\begin{aligned}
p(\mathbf{e}_{\mathbf{x}_{k,i}} \mid \mathbf{Q}) &= (2\pi)^{-n/2}|\mathbf{Q}|^{-1/2}\exp\Big(-\frac{1}{2}\mathbf{e}_{\mathbf{x}_{k,i}}^{\top}\mathbf{Q}^{-1}\mathbf{e}_{\mathbf{x}_{k,i}}\Big) \\
\log(p(\mathbf{e}_{\mathbf{x}_{k,i}} \mid \mathbf{Q})) &= -\frac{n}{2}\log(2\pi) - \frac{1}{2}\log(|\mathbf{Q}|) - \frac{1}{2}\mathbf{e}_{\mathbf{x}_{k,i}}^{\top}\mathbf{Q}^{-1}\mathbf{e}_{\mathbf{x}_{k,i}}
\end{aligned} \tag{A.4}$$

where n is the dimension of $\mathbf{Q}$ and the number of states.

The measurement noise has been observed to have a heavier tail than the normal distribution exhibits, and is therefore modeled as a Student-t distribution and here separated for position and velocity:

$$
\begin{aligned}
p(\mathbf{e}_{\mathbf{z}_{k,i}} \mid \mathbf{R}) &= p(\mathbf{e}^p_{\mathbf{z}_{k,i}} \mid \mathbf{R}^p) \cdot p(\mathbf{e}^v_{\mathbf{z}_{k,i}} \mid \mathbf{R}^v) \\
\mathbf{R} &= \begin{bmatrix} \mathbf{R}^p & \mathbf{0} \\ \mathbf{0} & \mathbf{R}^v \end{bmatrix} \\
p(\mathbf{e}^p_{\mathbf{z}_{k,i}} \mid \mathbf{R}^p) &= \frac{\Gamma[(\nu^p + n_{meas})/2]}{\Gamma(\nu^p/2)(\nu^p)^{n_{meas}/2}\pi^{n_{meas}/2}|\mathbf{R}^p|^{1/2}} \left[1 + \frac{1}{\nu^p}(\mathbf{e}^p_{\mathbf{z}_{k,i}})^\top (\mathbf{R}^p)^{-1}\mathbf{e}^p_{\mathbf{z}_{k,i}}\right]^{-(\nu^p+n_{meas})/2} \\
p(\mathbf{e}^v_{\mathbf{z}_{k,i}} \mid \mathbf{R}^v) &= \frac{\Gamma[(\nu^v + n_{meas})/2]}{\Gamma(\nu^v/2)(\nu^v)^{n_{meas}/2}\pi^{n/2}|\mathbf{R}^v|^{1/2}} \left[1 + \frac{1}{\nu^v}(\mathbf{e}^v_{\mathbf{z}_{k,i}})^\top (\mathbf{R}^v)^{-1}\mathbf{e}^v_{\mathbf{z}_{k,i}}\right]^{-(\nu^v+n_{meas})/2}
\end{aligned}
\tag{A.5}
$$

where n_{meas} is the dimension of $\mathbf{R}^p$, respectively $\mathbf{R}^v$. Both position and velocity have two dimensions, in x and y, such that $n_{meas} = 2$, which simplifies to:

$$
\begin{aligned}
p(\mathbf{e}^p_{\mathbf{z}_{k,i}} \mid \mathbf{R}^p) &= \frac{1}{2\pi|\mathbf{R}^p|^{1/2}} \left[1 + \frac{1}{\nu^p}(\mathbf{e}^p_{\mathbf{z}_{k,i}})^\top (\mathbf{R}^p)^{-1}\mathbf{e}^p_{\mathbf{z}_{k,i}}\right]^{-(\nu^p+2)/2} \\
p(\mathbf{e}^v_{\mathbf{z}_{k,i}} \mid \mathbf{R}^v) &= \frac{1}{2\pi|\mathbf{R}^v|^{1/2}} \left[1 + \frac{1}{\nu^v}(\mathbf{e}^v_{\mathbf{z}_{k,i}})^\top (\mathbf{R}^v)^{-1}\mathbf{e}^v_{\mathbf{z}_{k,i}}\right]^{-(\nu^v+2)/2}
\end{aligned}
\tag{A.6}
$$

The logarithmic distribution is thus:

$$
\begin{aligned}
\log(p(\mathbf{e}^p_{\mathbf{z}_{k,i}} \mid \mathbf{R}^p)) = &- \log(2\pi) - \frac{1}{2}\log(|\mathbf{R}^p|) \\
&- \frac{\nu^p + 2}{2}\log\left(1 + \frac{1}{\nu^p}(\mathbf{e}^p_{\mathbf{z}_{k,i}})^\top (\mathbf{R}^p)^{-1}\mathbf{e}^p_{\mathbf{z}_{k,i}}\right) \\
\log(p(\mathbf{e}^v_{\mathbf{z}_{k,i}} \mid \mathbf{R}^v)) = &- \log(2\pi) - \frac{1}{2}\log(|\mathbf{R}^v|) \\
&- \frac{\nu^v + 2}{2}\log\left(1 + \frac{1}{\nu^v}(\mathbf{e}^v_{\mathbf{z}_{k,i}})^\top (\mathbf{R}^v)^{-1}\mathbf{e}^v_{\mathbf{z}_{k,i}}\right)
\end{aligned}
\tag{A.7}
$$

This results in the following objective function:

$$
\begin{aligned}
\arg\max_{\boldsymbol{\sigma}} \quad & \frac{1}{M}\sum_{i=1}^{M}\Big(\frac{1}{N_i}\sum_{k=1}^{N_i}\Big(\log(p(\mathbf{e}_{\mathbf{z}_{k,i}} \mid \mathbf{R})) + \log(p(\mathbf{e}_{\mathbf{x}_{k,i}} \mid \mathbf{Q}))\Big)\Big) \\
= \arg\max_{\boldsymbol{\sigma}} \quad & \frac{1}{M}\sum_{i=1}^{M}\Big(\frac{1}{N_i}\sum_{k=1}^{N_i}\Big(-\log(2\pi) - \frac{1}{2}\log|\mathbf{R}^p| \\
& \qquad - \frac{\nu^p+2}{2}\log\Big(1+\frac{1}{\nu^p}(\mathbf{e}^p_{\mathbf{z}_{k,i}})^\top(\mathbf{R}^p)^{-1}\mathbf{e}^p_{\mathbf{z}_{k,i}}\Big) \\
& \qquad - \log(2\pi) - \frac{1}{2}\log|\mathbf{R}^v| \\
& \qquad - \frac{\nu^v+2}{2}\log\Big(1+\frac{1}{\nu^v}(\mathbf{e}^v_{\mathbf{z}_{k,i}})^\top(\mathbf{R}^v)^{-1}\mathbf{e}^v_{\mathbf{z}_{k,i}}\Big) \\
& \qquad - \frac{n}{2}\log(2\pi) - \frac{1}{2}\log|\mathbf{Q}| - \frac{1}{2}\mathbf{e}^\top_{\mathbf{x}_{k,i}}\mathbf{Q}^{-1}\mathbf{e}_{\mathbf{x}_k}\Big)\Big) \\
= \arg\max_{\boldsymbol{\sigma}} \quad & \frac{1}{M}\sum_{i=1}^{M}\Big(\frac{1}{N_i}\sum_{k=1}^{N_i}\Big(-\frac{1}{2}\log|\mathbf{R}| \\
& \qquad - \frac{\nu^p+2}{2}\log\Big(1+\frac{1}{\nu^p}(\mathbf{e}^p_{\mathbf{z}_{k,i}})^\top(\mathbf{R}^p)^{-1}\mathbf{e}^p_{\mathbf{z}_{k,i}}\Big) \\
& \qquad - \frac{\nu^v+2}{2}\log\Big(1+\frac{1}{\nu^v}(\mathbf{e}^v_{\mathbf{z}_{k,i}})^\top(\mathbf{R}^v)^{-1}\mathbf{e}^v_{\mathbf{z}_{k,i}}\Big) \\
& \qquad - \frac{1}{2}\log|\mathbf{Q}| - \frac{1}{2}\mathbf{e}^\top_{\mathbf{x}_{k,i}}\mathbf{Q}^{-1}\mathbf{e}_{\mathbf{x}_{k,i}}\Big)\Big) \\
= \arg\max_{\boldsymbol{\sigma}} \quad & \frac{1}{M}\sum_{i=1}^{M}\Big(\frac{1}{N_i}\sum_{k=1}^{N_i}\Big(-\frac{\nu^p+2}{2}\log\Big(1+\frac{1}{\nu^p}(\mathbf{e}^p_{\mathbf{z}_{k,i}})^\top(\mathbf{R}^p)^{-1}\mathbf{e}^p_{\mathbf{z}_{k,i}}\Big) \\
& \qquad - \frac{\nu^v+2}{2}\log\Big(1+\frac{1}{\nu^v}(\mathbf{e}^v_{\mathbf{z}_{k,i}})^\top(\mathbf{R}^v)^{-1}\mathbf{e}^v_{\mathbf{z}_{k,i}}\Big) \\
& \qquad - \frac{1}{2}\mathbf{e}^\top_{\mathbf{x}_{k,i}}\mathbf{Q}^{-1}\mathbf{e}_{\mathbf{x}_{k,i}}\Big)\Big) - \frac{1}{2}\log|\mathbf{Q}| - \frac{1}{2}\log|\mathbf{R}|
\end{aligned}
\tag{A.8}
$$

To avoid degenerate solutions, a regularizer can be used, such that undesired behaviors are penalized. For the Kalman Filters and FGOs, these degenerate solutions push either the process- or measurement-noise close to 0, resulting in either too much trust in the model or too much trust in the measurements. To avoid this, a penalizer is added, which penalizes covariances too far from what has been observed by the user. The tuning of Kalman Filters and FGOs is generally primarily performed on a logarithmic scale, as the largest difference is observed on a multiplicative scale and not on an additive one. Therefore, the penalization utilizes a log-normal distribution and observed approximate value of the noise to avoid degenerate solutions:

$$
\begin{aligned}
p(\boldsymbol{\sigma}) &= \prod_{\tau\in\mathcal{T}}\prod_{j\in\mathcal{J}^\tau_m} p(\sigma^\tau_j) \\
&= \prod_{\tau\in\mathcal{T}}\prod_{j\in\mathcal{J}^\tau_m} \frac{1}{\sigma^\tau_j\sigma^\tau_{0j}\sqrt{2\pi}}\exp\Big(-\frac{(\log(\sigma^\tau_j)-\mu^\tau_{0j})^2}{2(\sigma^\tau_{0j})^2}\Big) \\
\log(p(\boldsymbol{\sigma})) &= \sum_{\tau\in\mathcal{T}}\sum_{j\in\mathcal{J}^\tau_m}\Big(-\log(\sigma^\tau_j) - \log(\sigma^\tau_{0j}\sqrt{2\pi}) - \frac{(\log(\sigma^\tau_j)-\mu^\tau_{0j})^2}{2(\sigma^\tau_{0j})^2}\Big)
\end{aligned}
\tag{A.9}
$$

Thus, the updated optimization problem is:

$$
\begin{aligned}
\arg\max_{\boldsymbol{\sigma}^{\mathbf{Q}},\boldsymbol{\sigma}^{\mathbf{R}}} \quad & \frac{1}{M}\sum_{i=1}^{M}\Bigg(\frac{1}{N_i}\sum_{k=1}^{N_i}\Big(\log(p(\mathbf{e}_{\mathbf{z}_{k,i}} \mid \mathbf{R})) + \log(p(\mathbf{e}_{\mathbf{x}_{k,i}} \mid \mathbf{Q}))\Big)\Bigg) \\
& + \log(p(\boldsymbol{\sigma})) \\
= \arg\max_{\boldsymbol{\sigma}^{\mathbf{Q}},\boldsymbol{\sigma}^{\mathbf{R}}} \quad & \frac{1}{M}\sum_{i}^{M}\Bigg(\frac{1}{N_i}\sum_{k=1}^{N_i}\Bigg(-\frac{\nu^{pos}+2}{2}\log\Bigg(1+\frac{1}{\nu^{pos}}(\mathbf{e}^{pos}_{\mathbf{z}_{k,i}})^{\top}(\mathbf{R}^{pos})^{-1}\mathbf{e}^{pos}_{\mathbf{z}_{k,i}}\Bigg) \\
& \qquad - \frac{\nu^{vel}+2}{2}\log\Bigg(1+\frac{1}{\nu^{vel}}(\mathbf{e}^{vel}_{\mathbf{z}_{k,i}})^{\top}(\mathbf{R}^{vel})^{-1}\mathbf{e}^{vel}_{\mathbf{z}_{k,i}}\Bigg) \\
& \qquad - \frac{1}{2}\mathbf{e}^{\top}_{\mathbf{x}_{k,i}}\mathbf{Q}^{-1}\mathbf{e}_{\mathbf{x}_{k,i}}\Bigg)\Bigg) \\
& - \frac{1}{2}\log|\mathbf{Q}| - \frac{1}{2}\log|\mathbf{R}| \\
& + \sum_{\Sigma\in\{\mathbf{R},\mathbf{Q}\}}\sum_{j\in C^{\Sigma}}\Bigg(-\log(\sigma_j^{\Sigma}) - \frac{(\log(\sigma_j^{\Sigma})-\mu_{0j}^{\Sigma})^2}{2(\sigma_{0j}^{\Sigma})^2}\Bigg)
\end{aligned}
\tag{A.10}
$$

B

Extended Results

The following tables provide the complete numerical results that form the basis for all figures presented in Chapter 4.

B.1 Pedestrian

The following is the numerical results for Figure 4.1- Figure 4.5 for pedestrians.

B.1.1 Temporal Coherence

Temporal Coherence Pedestrians Moderate						
	Bias	**Spikes v**	**Spikes p**	**Drift v**	**Drift p**	**Summary**
Recall	100.0%	94.29%	94.29%	100.0%	100.0%	97.72%

Table B.1: Recall of Temporal Coherence under moderate induced errors for pedestrians.

Temporal Coherence Pedestrians Severe						
	Bias	**Spikes v**	**Spikes p**	**Drift v**	**Drift p**	**Summary**
Recall	100.0%	94.29%	94.29%	97.14%	100.0%	97.14%

Table B.2: Recall of Temporal Coherence under severe induced errors for pedestrians.

B.1.2 Simple Kalman Filters

Kalman Filter (CV-Pedestrian) Moderate						
	Bias	**Spikes v**	**Spikes p**	**Drift v**	**Drift p**	**Summary**
1 s	65.71%	25.71%	40.0%	37.14%	57.14%	45.14%
3 s	60.0%	28.57%	54.29%	45.71%	62.86%	50.29%
5 s	62.86%	28.57%	54.29%	45.71%	62.86%	50.86%

Table B.3: Recall of Kalman Filter, with CV model, under moderate induced errors for pedestrians.

Kalman Filter (CV-Pedestrian) Severe						
	Bias	**Spikes v**	**Spikes p**	**Drift v**	**Drift p**	**Summary**
1 s	88.57%	28.57%	80.0%	94.29%	85.71%	75.43%
3 s	88.57%	34.29%	80.0%	94.29%	88.57%	77.14%
5 s	88.57%	31.34%	82.86%	94.29%	88.57%	77.13%

Table B.4: Recall of Kalman Filter, with CV model, under severe induced errors for pedestrians.

B.1.3 Non-linear Kalman Filters

EKF (CT-Pedestrians) Moderate						
	Bias	**Spikes v**	**Spikes p**	**Drift v**	**Drift p**	**Summary**
1 s	80.0%	82.86%	85.71%	77.14%	71.43%	79.43%
3 s	74.29%	82.86%	82.86%	77.14%	88.57%	81.14%
5 s	77.14%	82.86%	80.0%	77.14%	80.0%	79.43%

Table B.5: Recall of Extended Kalman Filter, with CT model, under moderate induced errors for pedestrians.

EKF (CT-Pedestrians) Severe						
	Bias	**Spikes v**	**Spikes p**	**Drift v**	**Drift p**	**Summary**
1 s	91.43%	94.29%	97.14%	97.14%	94.29%	94.86%
3 s	94.29%	91.43%	97.14%	97.14%	94.29%	94.86%
5 s	97.14%	97.14%	97.14%	97.14%	94.29%	96.57%

Table B.6: Recall of Extended Kalman Filter, with CT model, under severe induced errors for pedestrians.

CKF (CT-Pedestrians) Moderate						
	Bias	**Spikes v**	**Spikes p**	**Drift v**	**Drift p**	**Summary**
1 s	80.0%	82.86%	85.71%	77.14%	71.43%	79.43%
3 s	74.29%	82.86%	82.86%	77.14%	88.57%	81.14%
5 s	77.14%	82.86%	80.0%	77.14%	80.0%	79.43%

Table B.7: Recall of Cubature Kalman Filter, with CT model, under moderate induced errors for pedestrians.

CKF (CT-Pedestrians) Severe						
	Bias	**Spikes v**	**Spikes p**	**Drift v**	**Drift p**	**Summary**
1 s	91.43%	94.29%	97.14%	97.14%	94.29%	94.86%
3 s	94.29%	91.43%	97.14%	97.14%	94.29%	94.86%
5 s	97.14%	97.14%	97.14%	97.14%	94.29%	96.57%

Table B.8: Recall of Cubature Kalman Filter, with CT model, under severe induced errors for pedestrians.

B.1.4 Factor Graph Optimization

FGO (CV-Pedestrians) Moderate						
	Bias	**Spikes v**	**Spikes p**	**Drift v**	**Drift p**	**Summary**
1 s	68.57%	25.71%	40.0%	40.0%	54.29%	45.71%
3 s	62.86%	28.57%	54.29%	45.71%	57.14%	49.71%
5 s	65.71%	28.57%	54.29%	45.71%	57.14%	50.29%

Table B.9: Recall of FGO, with CV model, under moderate induced errors for pedestrians.

FGO (CV-Pedestrians) Severe						
	Bias	**Spikes v**	**Spikes p**	**Drift v**	**Drift p**	**Summary**
1 s	88.57%	28.57%	74.29%	94.29%	82.86%	73.72%
3 s	88.57%	34.29%	82.86%	94.29%	85.71%	77.14%
5 s	91.43%	31.34%	80.0%	94.29%	85.71%	76.55%

Table B.10: Recall of FGO, with CV model, under severe induced errors for pedestrians.

FGO (CT-Pedestrians) Moderate						
	Bias	**Spikes v**	**Spikes p**	**Drift v**	**Drift p**	**Summary**
1 s	80.0%	74.29%	74.29%	71.43%	65.71%	73.14%
3 s	74.29%	77.14%	77.14%	68.57%	82.86%	76.0%
5 s	77.14%	74.29%	77.14%	68.57%	77.14%	74.86%

Table B.11: Recall of FGO, with CT model, under moderate induced errors for pedestrians.

FGO (CT-Pedestrians) Severe						
	Bias	**Spikes v**	**Spikes p**	**Drift v**	**Drift p**	**Summary**
1 s	91.43%	94.29%	94.29%	97.14%	94.29%	94.29%
3 s	94.29%	91.43%	94.29%	97.14%	94.29%	94.29%
5 s	97.14%	91.43%	97.14%	97.14%	94.29%	95.43%

Table B.12: Recall of FGO, with CT model, under severe induced errors for pedestrians.

B.1.5 Evaluation Metrics

Recall Pedestrians Moderate						
	TC	**Kalman**	**EKF**	**CKF**	**FGO CV**	**FGO CT**
1 s	97.2%	45.14%	79.43%	79.43%	45.71%	73.14%
3 s		50.29%	81.14%	81.14%	49.71%	76.0%
5 s		50.89%	79.43%	79.43%	50.29%	74.86%

Table B.13: Comparison between recall for different methods, under moderate induced errors for pedestrians.

Recall Pedestrians Severe						
	TC	**Kalman**	**EKF**	**CKF**	**FGO CV**	**FGO CT**
1 s	97.14%	75.43%	94.86%	94.86%	73.72%	94.29%
3 s		77.14%	94.86%	94.86%	77.14%	94.29%
5 s		77.13%	96.57%	96.57%	76.55%	95.43%

Table B.14: Comparison between recall for different methods, under severe induced errors for pedestrians.

Precision Pedestrians						
	TC	**Kalman**	**EKF**	**CKF**	**FGO CV**	**FGO CT**
1 s	44.23%	80.0%	78.26%	78.26%	80.0%	77.27%
3 s		77.27%	79.17%	79.17%	78.26%	78.26%
5 s		77.27%	75.0%	75.0%	77.27%	77.27%

Table B.15: Comparison between precision for different methods for pedestrians.

F_1 Pedestrians Moderate						
	TC	**Kalman**	**EKF**	**CKF**	**FGO CV**	**FGO CT**
1 s	60.8%	57.72%	78.84%	78.84%	58.18%	75.15%
3 s		60.93%	80.14%	80.14%	60.8%	77.11%
5 s		61.37%	77.15%	77.15%	60.93%	76.05%

Table B.16: Comparison between F_1 for different methods, under moderate induced errors for pedestrians.

F_1 Pedestrians Severe						
	TC	**Kalman**	**EKF**	**CKF**	**FGO CV**	**FGO CT**
1 s	60.78%	77.65%	85.76%	85.76%	76.73%	84.94%
3 s		77.21%	86.31%	86.31%	77.70%	85.53%
5 s		77.20%	84.43%	84.43%	76.91%	85.40%

Table B.17: Comparison between F_1 for different methods, under severe induced errors for pedestrians.

B.2 Cars

The following is the numerical results for Figure 4.12- Figure 4.16 for cars.

B.2.1 Temporal Coherence

Temporal Coherence Cars Moderate						
	Bias	**Spikes v**	**Spikes p**	**Drift v**	**Drift p**	**Summary**
Recall	100.0%	79.0%	85.0%	99.0%	99.0%	92.4%

Table B.18: Recall of Temporal Coherence under moderate induced errors for cars.

Temporal Coherence Cars Severe						
	Bias	**Spikes v**	**Spikes p**	**Drift v**	**Drift p**	**Summary**
Recall	91.0%	78.0%	85.0%	99.0%	99.0%	90.4%

Table B.19: Recall of Temporal Coherence under severe induced errors for cars.

B.2.2 Simple Kalman Filter

Kalman Filter (CV-Cars) Moderate						
	Bias	**Spikes v**	**Spikes p**	**Drift v**	**Drift p**	**Summary**
1 s	28.0%	2.0%	8.0%	29.0%	24.0%	18.2%
3 s	35.0%	4.0%	9.0%	36.0%	25.0%	21.8%
5 s	40.0%	5.0%	10.0%	38.0%	29.0%	24.4%

Table B.20: Recall of Kalman Filter, with CV model, under moderate induced errors for cars.

Kalman Filter (CV-Cars) Severe						
	Bias	**Spikes v**	**Spikes p**	**Drift v**	**Drift p**	**Summary**
1 s	100.0%	81.0%	95.0%	77.0%	96.0%	89.8%
3 s	100.0%	84.0%	97.0%	77.0%	96.0%	90.8%
5 s	100.0%	87.0%	98.0%	77.0%	96.0%	91.6%

Table B.21: Recall of Kalman Filter, with CV model, under severe induced errors for cars.

B.2.3 Non-linear Kalman Filters

EKF (CT-Cars) Moderate						
	Bias	**Spikes v**	**Spikes p**	**Drift v**	**Drift p**	**Summary**
1 s	25.0%	9.0%	12.0%	28.0%	25.0%	19.8%
3 s	32.0%	14.0%	12.0%	27.0%	23.0%	21.6%
5 s	34.0%	12.0%	12.0%	28.0%	26.0%	22.4%

Table B.22: Recall of Extended Kalman Filter, with CT model, under moderate induced errors for cars.

EKF (CT-Cars) Severe						
	Bias	**Spikes v**	**Spikes p**	**Drift v**	**Drift p**	**Summary**
1 s	100.0%	92.0%	96.0%	74.0%	96.0%	91.6%
3 s	99.0%	95.0%	97.0%	74.0%	96.0%	91.2%
5 s	98.0%	98.0%	99.0%	73.0%	97.0%	93.0%

Table B.23: Recall of Extended Kalman Filter, with CT model, under severe induced errors for cars.

CKF (CT-Cars) Moderate						
	Bias	**Spikes v**	**Spikes p**	**Drift v**	**Drift p**	**Summary**
1 s	26.0%	9.0%	12.0%	29.0%	27.0%	20.6%
3 s	30.0%	14.0%	13.0%	27.0%	25.0%	21.8%
5 s	33.0%	12.0%	13.0%	28.0%	27.0%	22.6%

Table B.24: Recall of Cubature Kalman Filter, with CT model, under moderate induced errors for cars.

CKF (CT-Cars) Severe						
	Bias	**Spikes v**	**Spikes p**	**Drift v**	**Drift p**	**Summary**
1 s	100.0%	92.0%	96.0%	74.0%	96.0%	91.6%
3 s	99.0%	95.0%	98.0%	74.0%	96.0%	92.4%
5 s	98.0%	98.0%	100.0%	73.0%	97.0%	93.2%

Table B.25: Recall of Cubature Kalman Filter, with CT model, under severe induced errors for cars.

B.2.4 Factor Graph Optimization

FGO (CV-Cars) Moderate						
	Bias	**Spikes v**	**Spikes p**	**Drift v**	**Drift p**	**Summary**
1 s	29.0%	3.0%	8.0%	27.0%	22.0%	17.8%
3 s	37.0%	5.0%	9.0%	34.0%	24.0%	21.8%
5 s	41.0%	5.0%	10.0%	37.0%	29.0%	24.4%

Table B.26: Recall of FGO, with CV model, under moderate induced errors for cars.

FGO (CV-Cars) Severe						
	Bias	**Spikes v**	**Spikes p**	**Drift v**	**Drift p**	**Summary**
1 s	100.0%	86.0%	94.0%	76.0%	96.0%	90.4%
3 s	100.0%	90.0%	97.0%	76.0%	96.0%	91.8%
5 s	100.0%	90.0%	97.0%	76.0%	96.0%	91.8%

Table B.27: Recall of FGO, with CV model, under severe induced errors for cars.

FGO (CT-Cars) Moderate						
	Bias	**Spikes v**	**Spikes p**	**Drift v**	**Drift p**	**Summary**
1 s	30.0%	38.0%	29.0%	34.0%	32.0%	32.6%
3 s	31.0%	41.0%	34.0%	24.0%	28.0%	31.6%
5 s	33.0%	42.0%	35.0%	22.0%	32.0%	32.8%

Table B.28: Recall of FGO, with CT model, under moderate induced errors for cars.

FGO (CT-Cars) Severe						
	Bias	**Spikes v**	**Spikes p**	**Drift v**	**Drift p**	**Summary**
1 s	99.0%	97.0%	98.0%	74.0%	96.0%	92.8%
3 s	98.0%	99.0%	98.0%	74.0%	96.0%	93.0%
5 s	99.0%	100.0%	100.0%	73.0%	96.0%	93.6%

Table B.29: Recall of FGO, with CT model, under severe induced errors for cars.

B.2.5 Evaluation Metrics

Recall Cars Moderate						
	TC	**Kalman**	**EKF**	**CKF**	**FGO CV**	**FGO CT**
1 s	92.4%	18.2%	19.8%	20.6%	17.8%	32.6%
3 s		21.8%	21.6%	21.8%	21.8%	31.6%
5 s		24.4%	22.4%	22.6%	24.4%	32.8%

Table B.30: Comparison between recall for different methods, under moderate induced errors for cars.

Recall Cars Severe						
	TC	**Kalman**	**EKF**	**CKF**	**FGO CV**	**FGO CT**
1 s	90.4%	89.8%	91.6%	91.6%	90.4%	92.8%
3 s		90.8%	91.2%	92.4%	91.8%	93.0%
5 s		91.6%	93.0%	93.2%	91.8%	93.6%

Table B.31: Comparison between recall for different methods, under severe induced errors for cars.

Precision Cars						
	TC	**Kalman**	**EKF**	**CKF**	**FGO CV**	**FGO CT**
1 s	28.33%	80.0%	81.81%	80.36%	81.25%	82.81%
3 s		77.5%	77.05%	75.81%	78.95%	78.87%
5 s		83.34%	81.97%	79.03%	85.0%	79.71%

Table B.32: Comparison between precision for different methods for cars

F_1 Cars Moderate						
	TC	**Kalman**	**EKF**	**CKF**	**FGO CV**	**FGO CT**
1 s	43.36%	29.65%	31.88%	32.79%	29.2%	46.78%
3 s		34.03%	33.74%	33.86%	34.17%	45.12%
5 s		37.75%	35.18%	35.15%	37.92%	46.48%

Table B.33: Comparison between F_1-score for different methods, under moderate induced errors.

F_1 Cars Severe						
	TC	**Kalman**	**EKF**	**CKF**	**FGO CV**	**FGO CT**
1 s	43.14%	84.62%	86.43%	85.61%	85.58%	87.52%
3 s		83.63%	83.53%	83.29%	84.89%	85.35%
5 s		87.28%	87.14%	85.53%	88.27%	86.70%

Table B.34: Comparison between F_1-score for different methods, under severe induced errors.

B.3 Trucks

The following is the numerical results for Figure 4.23- Figure 4.27 for trucks.

B.3.1 Temporal Coherence

Temporal Coherence Trucks Moderate						
	Bias	**Spikes v**	**Spikes p**	**Drift v**	**Drift p**	**Summary**
Recall	100.0%	99.0%	99.0%	100.0%	100.0%	99.6%

Table B.35: Recall of Temporal Coherence under moderate induced errors for trucks.

Temporal Coherence Trucks Severe						
	Bias	**Spikes v**	**Spikes p**	**Drift v**	**Drift p**	**Summary**
Recall	100.0%	99.0%	99.0%	100.0%	100.0%	99.6%

Table B.36: Recall of Temporal Coherence under severe induced errors for trucks.

B.3.2 Simple Kalman Filter

Kalman Filter (CV-Trucks) Moderate						
	Bias	**Spikes v**	**Spikes p**	**Drift v**	**Drift p**	**Summary**
1 s	83.0%	7.0%	51.0%	76.0%	76.0%	58.6%
3 s	82.0%	12.0%	52.0%	78.0%	82.0%	61.2%
5 s	88.0%	14.0%	54.0%	78.0%	81.0%	63.0%

Table B.37: Recall of Kalman Filter, with CV model, under moderate induced errors for trucks.

Kalman Filter (CV-Trucks) Severe						
	Bias	**Spikes v**	**Spikes p**	**Drift v**	**Drift p**	**Summary**
1 s	100.0%	83.0%	96.0%	90.0%	94.0%	92.6%
3 s	99.0%	88.0%	98.0%	90.0%	95.0%	94.0%
5 s	99.0%	86.0%	99.0%	90.0%	95.0%	93.8%

Table B.38: Recall of Kalman Filter, with CV model, under severe induced errors for trucks.

B.3.3 Non-linear Kalman Filters

EKF (CT-Trucks) Moderate						
	Bias	**Spikes v**	**Spikes p**	**Drift v**	**Drift p**	**Summary**
1 s	90.0%	88.0%	84.0%	79.0%	81.0%	84.4%
3 s	84.0%	95.0%	84.0%	79.0%	91.0%	86.6%
5 s	88.0%	93.0%	85.0%	79.0%	87.0%	86.4%

Table B.39: Recall of Extended Kalman Filter, with CT model, under moderate induced errors for trucks.

EKF (CT-Trucks) Severe						
	Bias	**Spikes v**	**Spikes p**	**Drift v**	**Drift p**	**Summary**
1 s	100.0%	100.0%	99.0%	86.0%	95.0%	96.0%
3 s	98.0%	100.0%	100.0%	86.0%	95.0%	95.8%
5 s	99.0%	100.0%	100.0%	86.0%	95.0%	96.0%

Table B.40: Recall of Extended Kalman Filter, with CT model, under severe induced errors for trucks.

CKF (CT-Trucks) Moderate						
	Bias	**Spikes v**	**Spikes p**	**Drift v**	**Drift p**	**Summary**
1 s	90.0%	88.0%	84.0%	79.0%	81.0%	84.4%
3 s	84.0%	95.0%	84.0%	79.0%	91.0%	86.6%
5 s	88.0%	93.0%	85.0%	79.0%	87.0%	86.4%

Table B.41: Recall of Cubature Kalman Filter, with CT model, under moderate induced errors for trucks.

CKF (CT-Trucks) Severe						
	Bias	**Spikes v**	**Spikes p**	**Drift v**	**Drift p**	**Summary**
1 s	100.0%	100.0%	99.0%	86.0%	95.0%	96.0%
3 s	98.0%	100.0%	100.0%	86.0%	95.0%	95.8%
5 s	99.0%	100.0%	100.0%	86.0%	95.0%	96.0%

Table B.42: Recall of Cubature Kalman Filter, with CT model, under severe induced errors for trucks.

B.3.4 Factor Graph Optimization

FGO (CV-Trucks) Moderate						
	Bias	**Spikes v**	**Spikes p**	**Drift v**	**Drift p**	**Summary**
1 s	46.0%	1.0%	27.0%	68.0%	43.0%	37.0%
3 s	47.0%	1.0%	29.0%	73.0%	44.0%	38.8%
5 s	44.0%	1.0%	29.0%	74.0%	49.0%	39.4%

Table B.43: Recall of FGO, with CV model, under moderate induced errors for trucks.

FGO (CV-Trucks) Severe						
	Bias	**Spikes v**	**Spikes p**	**Drift v**	**Drift p**	**Summary**
1 s	87.0%	5.0%	96.0%	84.0%	91.0%	72.6%
3 s	84.0%	11.0%	97.0%	84.0%	93.0%	73.8%
5 s	90.0%	12.0%	98.0%	84.0%	93.0%	75.4%

Table B.44: Recall of FGO, with CV model, under severe induced errors for trucks.

FGO (CT-Trucks) Moderate						
	Bias	**Spikes v**	**Spikes p**	**Drift v**	**Drift p**	**Summary**
1 s	71.0%	72.0%	74.0%	74.0%	70.0%	72.2%
3 s	66.0%	76.0%	74.0%	73.0%	82.0%	74.2%
5 s	64.0%	80.0%	78.0%	76.0%	77.0%	75.0%

Table B.45: Recall of FGO, with CT model, under moderate induced errors for trucks.

FGO (CT-Trucks) Severe						
	Bias	**Spikes v**	**Spikes p**	**Drift v**	**Drift p**	**Summary**
1 s	98.0%	98.0%	98.0%	83.0%	95.0%	94.4%
3 s	96.0%	100.0%	99.0%	84.0%	95.0%	94.8%
5 s	98.0%	100.0%	100.0%	84.0%	95.0%	95.4%

Table B.46: Recall of FGO, with CT model, under severe induced errors for trucks.

B.3.5 Evaluation Metrics

Recall Trucks Moderate						
	TC	**Kalman**	**EKF**	**CKF**	**FGO CV**	**FGO CT**
1 s	99.6%	58.6%	84.4%	84.4%	37.0%	72.2%
3 s		61.2%	86.6%	86.6%	38.8%	74.2%
5 s		63.0%	86.4%	86.4%	39.4%	75.0%

Table B.47: Comparison for recall between different methods, under moderate induced errors for trucks.

Recall Trucks Severe						
	TC	**Kalman**	**EKF**	**CKF**	**FGO CV**	**FGO CT**
1 s	99.6%	92.6%	96.0%	96.0%	72.6%	94.4%
3 s		94.0%	95.8%	95.8%	73.8%	94.8%
5 s		93.8%	96.0%	96.0%	75.4%	95.4%

Table B.48: Comparison for recall between different methods, under severe induced errors for trucks.

Precision Trucks						
	TC	**Kalman**	**EKF**	**CKF**	**FGO CV**	**FGO CT**
1 s	14.66%	73.33%	65.22%	65.22%	75.0%	76.47%
3 s		66.67%	59.09%	59.09%	72.73%	72.22%
5 s		58.82%	52.0%	52.0%	72.73%	61.11%

Table B.49: Comparison for precision between different methods for trucks.

F_1 Trucks Moderate						
	TC	**Kalman**	**EKF**	**CKF**	**FGO CV**	**FGO CT**
1 s	25.56%	65.14%	73.58%	73.58%	49.55%	74.27%
3 s		63.82%	70.25%	70.25%	50.6%	73.20%
5 s		60.84%	64.93%	64.93%	51.11%	67.35%

Table B.50: Comparison for F_1-score between different methods, under moderate induced errors for trucks.

F_1 **Trucks Severe**						
	TC	**Kalman**	**EKF**	**CKF**	**FGO CV**	**FGO CT**
1 s	25.56%	81.85%	77.67%	77.67%	73.78%	84.50%
3 s		78.01%	73.10%	73.10%	73.26%	81.98%
5 s		72.30%	67.46%	67.46%	74.04%	74.50%

Table B.51: Comparison for F_1-score between different methods, under severe induced errors for trucks.

DEPARTMENT OF ELECTRICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY