TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Tatsuki Ishikawa 214203IASM

ANALYSIS AND IMPLEMENTATION OF OPTIMAL FUSION METHOD FOR AUTONOMOUS VEHICLES IN GPS DENIED ENVIRONMENT

Master's Thesis

Supervisor: Uljana Reinsalu
PhD

Co-supervisor: Mairo Leier
PhD

Tallinn 2024

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Tatsuki Ishikawa 214203IASM

AUTONOOMSETE SÕIDUKITE OPTIMAALSE FUSIOONIMEETODI ANALÜÜS JA RAKENDAMINE GPS-SIGNAALITA KESKKONNAS

Magistritöö

Juhendaja: Uljana Reinsalu
PhD

Kaasjuhendaja: Mairo Leier
PhD

Tallinn 2024

Author’s Declaration of Originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Tatsuki Ishikawa

17.12.2024

Abstract

Accurate state estimation is essential for autonomous systems operating in complex environments, where sensor fusion plays an important role in integrating data from multiple sensors such as Inertial Measurement Units (IMU), Visual Odometry (VO), and Global Positioning System (GPS). This study evaluates five widely used Kalman filter variants, including Extended Kalman Filter (EKF), Unscented Kalman Filter (UKF), Particle Filter (PF), Ensemble Kalman Filter (EnKF), and Cubature Kalman Filter (CKF), to identify their strengths, limitations, and suitability for real-world applications. The filters were applied to the publicly available autonomous benchmarks, known as KITTI dataset, under various experimental setups, including different motion models: basic kinematics and velocity motion model; sensor configurations: IMU and VO estimates with and without GPS; and scenarios simulating sensor data loss. The results are evaluated by three main error metrics: Absolute Trajectory Error (ATE), Relative Pose Error (RPE) for translation and RPE for rotation to properly assess estimated trajectories and understand filters' characteristics.

The results reveal key trade-offs among the filters. While EKF and CKF demonstrate excellent accuracy in translation metrics such as Absolute Trajectory Error (ATE), the UKF achieves near-optimal results across all metrics, balancing sensor data effectively in multi-sensor environments. Conversely, PF and EnKF, though capable of handling non-Gaussian uncertainties, face computational challenges due to their large number of particles or ensembles. Additionally, the robustness of each filter was tested in scenarios with simulated GPS and VO data loss, reflecting conditions such as non-line-of-sight (NLOS) for GPS and terrain or lighting challenges for VO. The CKF, in particular, exhibited resilience to short-term sensor failures, maintaining reliable estimates up to 4 seconds of data loss.

Furthermore, the filters were assessed for their usability in practical applications, considering inference speed and the complexity of parameter tuning. The EKF and CKF demonstrated advantages in computational efficiency and ease of implementation, while the UKF offered dynamic noise tuning capabilities, making it ideal for scenarios with variable sensor reliability.

This paper provides a systematic evaluation of filters' estimations, highlighting their performance in realistic scenarios and their applicability to practical environments. The results achieved by this study provides insights are valuable for the development of autonomous systems, especially grounded vehicle navigation/localization, requiring robust and efficient multi-sensor fusion in challenging operational conditions.

The thesis is written in English and is 89 pages long, including 6 chapters, 23 figures and 1 table.

Annotatsioon

Autonoomsete sõidukite optimaalse fusioonimeetodi analüüs ja rakendamine GPS-signaalita keskkonnas

Täpne olekuhinnang on oluline autonoomsete süsteemide jaoks, mis tegutsevad keerukates keskkondades, kus andurite ühendamisel on oluline roll mitmete andurite, näiteks inertsiaalsensorite (IMU), visuaalse odomeetria (VO) ja GPSi andmete integreerimisel. Käesolevas lõputöös hinnatakse viit laialdaselt kasutatavat Kalmani filtri varianti, laiendatud Kalmani filtrit (EKF), mittekohandatud Kalmani filtrit (UKF), osakeste filtrit (PF), ansambli Kalmani filtrit (EnKF) ja kuubatuurset Kalmani filtrit (CKF), et teha kindlaks nende tugevused, piirangud ja sobivus reaalsete rakenduste jaoks. Filtreid rakendati avalikult kättesaadavate autonoomsete süsteemide võrdlusnäitajate nagu KITTI andmekogumi põhjal erinevate eksperimentaalsete seadistustega, sealhulgas erinevate liikumismudelite puhul: põhiline kinemaatika ja kiiruse liikumismudel; andurite konfiguratsioonid: IMU ja VO hinnangud koos GPSiga ja ilma; ning stsenaariumid, mis simuleerivad andurite andmete kadumist. Tulemusi hinnatakse kolme peamise veamõõdikuga: Absoluutne trajektooriviga (ATE), suhteline asendiviga (RPE) translatsiooni puhul ja RPE rotatsiooni puhul, et hinnata hinnangulisi trajektoore ja mõista filtrite omadusi.

Tulemustest selguvad peamised erinevused filtrite vahel. Kuigi EKF ja CKF näitavad suurepärast täpsust absoluutse trajektoorivea (ATE) mõõdikus, saavutab UKF peaaegu optimaalseid tulemusi kõigis mõõdikutes, tasakaalustades tõhusalt andurite andmeid mitme anduriga keskkondades. Seevastu PF ja EnKF, mis on küll võimelised käsitlema mitte-Gaussi ebakindluseid, seisavad suure hulga osakeste või ansamblite tõttu silmitsi arvutuslike väljakutsetega. Lisaks sellele testiti iga filtri töökindlust stsenaariumides, kus simuleeriti GPSi ja VO andmete kadumist, mis peegeldab selliseid tingimusi nagu GPSi puhul nn. no-line-of-sight (NLOS) ja VO puhul maastiku või valgustuse probleemid. CKF näitas vastupidavust lühiajalistele andurikatkestustele, säilitades usaldusväärsed hinnangud kuni 4 sekundilise andmekao korral.

Lisaks hinnati filtrite kasutatavust praktilistes rakendustes, võttes arvesse järelduste tegemise kiirust ja parameetrite häälestamise keerukust. EKF ja CKF näitasid eeliseid arvutustõhususe ja rakendamise lihtsuse osas, samas kui UKF pakkus dünaamilise müra häälestamise võimalusi, mis muudab selle ideaalseks muutuva anduri usaldusväärsusega stsenaariumide jaoks. Käesolevas töös esitatakse filtrite hinnangute süstemaatiline hindamine, rõhutades nende jõudlust realistlikes stsenaariumides ja nende rakendatavust praktilistes keskkondades. Selle uuringu tulemused annavad ülevaate, millised filtrid on väärtuslikud autonoomsete süsteemide arendamisel, eriti maapealse sõiduki navigeerimise/lokaliseerimise jaoks, mis nõuavad tugevat ja tõhusat mitme sensori fusiooni keerulistes kasutustingimustes.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 89 leheküljel, 6 peatükki, 23 joonist, ja 1 tabelit.

List of Abbreviations and Terms

AGV	Autonomous Grounded Vehicle
UAV	Unmanned Aerial Vehicle
MAV	Micro Aerial Vehicle
USV	Unmannded Surface Vehicle
IMU	Inertial Measurement Units
INS	Inertial Navigation System
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
RTK	Real Time Kinematic
WiFi	Wireless Fidelity
LTE	Long-Term Evolution
AM/FM	Amplitude Modulation/Frequency Modulation
MEMS	Micro Electronic Mechanical Systems
VO	Visual Odometry
LiDAR	Light Detection and Ranging
UWB	Ultra-wideband
EKF	Extended Kalman Filter
UKF	Unscented Kalman Filter
PF	Particle Filter
EnKF	Ensemble Kalman Filter
CKF	Cubature Kalman Filter
ES-EKF	Error State Extended Kalman Filter
LKF	Linear Kalman Filter
SR-UKF	Square Root Unscented Kalman Filter
RBPF	Rao-Blackwellized Particle Filter
KITTI	Karlsruhe Institute of Technology and Toyota Technological Institute
ATE	Absolute Trajectory Error
RPE	Relartive Pose Error
NLOS	Non-line-of-sight
ZUPT	Zero Velocity Update
SOP	Signals of Opportunity
SLAM	Simultaneous localization and mapping

ML	Machine Learning
CNN	Convolutional Neural Network
TCN	Temporal Convolutional Network
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
SSD	Sum of Squared Distance
NCC	Normalized Cross Correlation
SIFT	Scale Invariant Feature Transform
SURF	Speeded Up Robust Feature
ORB	Oriented FAST and Rotated FRIEF
KNN	K-Nearest Neighbors
RANSAC	Random Sample Consensus

Table of Contents

List of Figures

List of Tables

1. Introduction

Over the past decades, autonomous robots, such as autonomous grounded-vehicle (AGVs), unmmanned aerial vehicles (UAVs), humanoid robot, etc., grabs increasing attention for research and commercial use in various fields to improve production, and to reduce human errors. For example, AGVs have a potential to provide huge impacts on public transportation and logistics filed by enhancing transportation safety and trackability. Moreover, the applications are extended to assists military operation such as safe inspection of dangerous area and transportation of military supplies. In terms of UAVs applications, UAVs are used in a variety of fields such as military field: navigation, secure communication, and reconnaissance; industrial field: package delivery, surveillance system, environment management system, precision agriculture; and commercial field: filming, and medical service [1].

For robot automation, an accurate estimation of robot's position and motion, often represented by a state of a system in a defined coordinate, is crucial. Accurate state estimation enables reliable navigation, control, and decision-making by fusion information from various on-board sensors such as Inertial Measurement Units (IMUs), image sensors for Visual Odometry (VO), Global Positioning System (GPS), and LIDAR. In an environment where sensor data is noisy, incomplete, or subject to temporary failure, state estimation by fusion systems plays a crucial role in providing robust performance of the system. Among these sensor fusion algorithms, Kalman filters and their extended variants have increasing demand to solve system's nonlinear dynamics.

This study will focus on five prominent variants of Kalman filters: linearization-based filter, called Extended Kalman Filter (EKF), two deterministic sampling-based filters including Unscented Kalman Filter (UKF) and Cubature Kalman Filter (CKF), a stochastic sampling-based filter called Particle Filter (PF), and an ensemble-based filter, Ensemble Kalman Filter (EnKF). These filters offer distinct characteristics, different configuration for their parameters, and computational requirements, making their performance application-dependent. Additionally, the combinations of two variants of the system's motion model—defined as the basic kinematics motion model and the velocity motion model—along with various types of sensor data, such as IMUs, stereo camera sequences, and GPS will be tested.

The evaluation of these filters will be conducted using KITTI dataset, which is a widely used benchmark for localization and mapping research. Experiments will be conducted under various conditions, including the availability or absence of GPS simulating multi-sensor environment, data loss in VO to estimate the state of a system by IMU-only dead-reckoning. To assess the performance of filters, three error metrics will be considered: Absolute Trajectory Error (ATE), Relative Pose Error (RPE) for translation, and RPE for rotation. These metrics enable a comprehensive comparison of the filters' accuracy, robustness and computational efficiency.

1.1 Background

Kalman filter was invented in 1960 by Rudolf Email Kalman to obtain the specification of a linear dynamic system that achieves the prediction, separation, or detection of a random signal [2]. Due to its ability to provide filtering to extract useful information and estimate of a linear system's state achieved by using prior knowledge, it is used in a variety of fields including object tracking, weather forecast, ocean current change prediction, crop yield estimation, and localization of autonomous vehicle, as shown in the paper [3]. A well-known example of Kalman filter application is the Mars rover in a high slip environment estimating pose of the rover and the path following [4]. Kalman Filter is implemented with integration of IMU and VO to estimate the rover's motion and capture the location calculated by the kinematics equation. The wheel slip is detected by comparing the estimated motion of the vehicle with the kinematic estimate for recovery to the defined path.

For localization and navigation, the Global Navigation Satellite System (GNSS) is the widely used positioning system that offers location and time. GPS is a variant of the positioning service that is provided by the United States, which has sub-meters accuracy in open area when using accurate receiver. However, GNSS susceptible to data loss due to environmental issues, such as signal blockage by obstacles located between transmitter and receiver leading to none-line-of-sight (NLOS), reflection from objects causing multipath error, jamming and spoofing by malicious attackers.

Kalman filter and its variants play an important role in which data from different sensors are fused together to estimate and compensate when one sensor is unreliable or completely unavailable.

1.2 Literature Review

This section presents a literature review on the application of the Kalman Filter across various fields, explores different Kalman Filter designs—primarily the loosely-coupled and tightly-coupled approaches—and examines the integration of diverse filtering algorithms as well as sensor fusion techniques augmented with machine learning algorithms.

1.2.1 Application fields of sensor fusions

The application fields of sensor fusion are diverse, including unmanned ground vehicles (UGVs) for navigation and localization, unmanned aerial vehicles (UAVs) for tasks such as autonomous flight and obstacle avoidance, agricultural applications for precision farming and crop monitoring, and various other domains such as robotics, healthcare, and smart cities where integrated sensor data enhances system performance and decision-making.

Sensor Fusion in Fields of Unmanned Grounded Vehicle

In [5], wheel odometory in x and y coordinates, two IMUs, two GPS are filtered by EKF to estimate robot's position. They concluded that adding additional sensor improves the result of estimation and compensates the other sensor's faultiness and low sampling rate. The result shows that the EKF with wheel odometry, two IMUs, and two GPS configuration achieves the loop closure accuracy less than one meter. In [6], indoor grounded vehicle localization using IMU and UWB sensors based on the improved CKF involving an adaptive factor for a noise estimation is provided. To deal with the non-line-of-sight (NLOS) problem of the UWB sensor, the Levenberg-Marquardt algorithm is used to minimize the error of UWB sensor. An experiment is conducted indoor environment with three other filtering algorithms, including traditonal CKF, UKF, and EKF and the estimates are evaluated by maximum error and average error in both x- and y-axis. The resaerch concludes that the improved CKF algorithm shows a more obvious advantage both in the LOS scenario and in the NLOS scenario of UWB, where the algorithm maintains the localization error within 0.1m and 0.3m in LOS and NLOS scenarios respectively. [7] focuses on autonomous vehicle control, sensor fusion techniques and state-estimation techniques in self-driving cars. The problem is tackled by integrating Error State Extended Kalman Filter (ES-EKF) since it can handle the non-linearities that arise from the sensor measurements more effectively than EKF in general. The paper utilized CARLA simulator to validate the proposed algorithm, which incorporates IMU, LiDAR and GPS provided by CARLA simulator, and it accurately estimates the state of a self-driving car in a simulated environment with a wide range of driving scenarios. The result shows that the proposed algorithm estimates the position of the vehicle by keeping the error in range ± 0.5m. The method proposed by [8] introduces PF to integrate passive sensors, including GNSS, INS and camera data to improve the accuracy of localization estimation. It is known that GPS suffers NLOS and multipath problems in unrban scenario, such as canyon, hence weights of the particles in PF are calculated based on the position of the vehicle being suffered by multipath problem given the provided 3D map. Moreover, lane detection algorithm is also carried out by machine vision so that the state of the system, which represents the pose of the vehicle, is updated by the coordinate information resulted from the lane detection. The dynamics of particles are done using linear velocity and angular velocity provided by INS. The result reveals that the integration with three data sources: GNSS, INS and Lane detection, achieves less than one meter accuracy in the positioning error mean metric.

With regard to **visual odometry and IMU sensor fusion on KITTI dataset**, the research [9] implements a loosely-coupled EKF that integrates IMU and vision-based motion estimator referred to as Parallax Beam. The proposed algorithm predicts next state of the system using IMU and the estimation is corrected using the vision-based motion estimator. The system is tested on six sequences of KITTI dataset and evaluated by mean relative translation and mean relative rotation error metrics. The performance of the proposed approach is compared to the ablated vision-based motion estimator and the result reveals that the proposed method shows results equal to or better than the Parallax Bean alone.

Sensor Fusion in Fields of Unmanned Aerial Vehicle (UAV)

In terms of UAV/Micro Aerial Vehicle (MAV) localization, The research introduced in [10] uses Error-State EKF that accepts IMU data to propagate the system's state to track the UAV in indoor, GPS and magnetometer denied environment. Besides the IMU, it uses LiDAR, UWB and VIO provided by Intel RealSense T265 and continuously update estimated position of the UAV by extending ES-EKF model in a way to handle drift in sensor position and orientation. The experiment is conducted in an indoor environment and the result shows that the proposed filtering algorithm leverages each sensor advantage to obtain an accurate estimate. The implementation presented by [11] provides a localization of the UAV using IMU and two UWB tags for aiming safe landing on an unmanned vehicle, on which four UWB anchors installed. The experiment involves RTK localization data used as a ground truth and estimates are evaluated by Root Mean Squared Error metric to numerically compare the accuracy provided by the proposed method with two UWB tags and one UWB tag. The result shows that based on five experiments, the method with two tags provides the accuracy with the maximum RMSE of 0.2303m and the minimum RMSE of 0.092m, while the single-tag method offers the maximum RMSE is 0.6875m, and the minimum is 0.1813m. Compared with the single-tag method, the RMSE of the proposed localization system was reduced by an average of 61%.

Sensor Fusion in Other Fields

In the context of agricultural applications, [12] employs an EKF to fuse data from a low-cost 9-axis IMU and GNSS-RTK for pose estimation of a crawler-type vehicle in a path-following task. The algorithm's performance is evaluated using three trajectories, with Root Mean Square Error (RMSE) used to quantitatively assess the system's robustness. The results show that the RMSE across the three trajectories is 10 cm, with a maximum error of 30 cm. Similarly, [13] proposes a loosely-coupled EKF multi-sensor fusion algorithm that fuses IMU, GPS-RTK, VO and vehicle odometry available at different frequencies and aims to navigate agricultural grounded vehicle in a farming area. The experiments utilizes publically available Rosario dataset, and the performance of the algorithm is evaluated using Mean Squared Error (MSE). The results are compared against the IMU-ODOM fusion algorithms, which relies on IMU and wheel odometry, and the MSCKF-VIO algorithm provided by [14]. The result highlights that the proposed algorithm outperforms the other two algorithms in terms of estimation accuracy.

For pedestrian localization, where an IMU is mounted on the foot, a Zero Velocity Update (ZUPT)-aided Inertial Navigation System (INS) is commonly used for dead-reckoning [15]. The problem of foot-mounted IMU is that there is discontinuity at the end of each step. Hence, Kalman filter is introduced to smoothen the discontinuity of the pedestrian's motion. The results show that the smoothing algorithm significantly reduces deviations, producing estimated trajectories that closely align with the actual trajectory.

1.2.2 Loosely-coupled and Tightly-coupled sensor fusion

There are mainly two approaches that are commonly introduced in the fields: the loosely-coupled and the tightly-coupled integration. Generally, the tightly-coupled implementation provides better performance in estimation since the former, the loosely-coupled approach, fuses the ready information coming from a sensor directly to the system's estimation, while the latter, the tightly-coupled approach, handles raw information obtained from the sensor and provides more flexibilities to apply some prior works, such as denoising, temperature compensation, etc. Here, several examples for each implementation technique are showcased.

Loosely-coupled approach

The loosely-coupled integration provides simplicity, flexibility and ease of implementation, this approach is preferably chosen for development. For example, the paper [16] proposes a loosely-coupled integration of GNSS and IMU to compensate for errors of both systems using the EKF for the Unmanned Surface Vehicle (USV). The performance of proposed integration is compared to the integration using UKF. The proposed EKF method provides 0.41m in the accuracy of horizontal position and more smoothed linear velocity estimation than UKF. Similarly, the research proposed by [17] integrates GPS/INS sensor fusion scheme that comprise two stages cascaded EKF-LKF. The first stage that consists of two EKFs, are designed for the INS and GPS to remove noise of the raw sensor data and working as a pure state estimator, while the filter at the second stage, namely Linear Kalman Filter fuses both filtered data to estimate the position of the grounded vehicle. The performance of the proposed approach is evaluated by Mean Absolute Error (MAE) and compared to three different algorithms: the EKF, the Augmented Integral Kalman Filter (AIKF), and the FIKF proposed in [18]. The results demonstrate that the proposed filter achieves a positioning accuracy of less than 0.5 meters in the ENU coordinate system, outperforming the other filters, particularly in scenarios with small GPS outages.

Tightly-coupled approach

In the research [19], grounded-vehicle's localization problem is addressed by a fusion system that integrate low-cost IMU (MEMS) and GPS in tightly-coupled manner. In stead of directly computing the vehicle's position, velocity, and attitude, which is done in loosely-coupled approach introduced in the same paper, the system involves temperature calibration and Gyroscope recalibration for INS data and additional mathematical integrations, such as nonholonomic constraint and static condition detection, are implemented to compute vehicle's pose. As a result, the tightly-coupled implementation outperforms the other setups, including loosely-coupled commercial GPS module.

As another example, [20] integrates both loosely-coupled and tightly-coupled approach to test different INS/GPS strategies for low-cost airborne navigation. It exhibits that there is no significant difference between the loosely-coupled and the tightly-coupled integrations when a situation that all the satellites are available. However, when the number of available satellite decreases, the absolute position and velocity errors increases for loosely-coupled approach while with the tightly-coupled approach provides sufficiently accurate solution even in situations with 3 or 2 satellites available.

The research [21] stated that GPS is widely used but susceptible to jamming and spoofing by malicious attackers. Hence, the paper proposed another possibility to compensate GPS signals loss by the other ambient radio signals that are not intended for navigation purpose, such as WiFi, AM/FM radio, cellular, digital television, and they are referred to as Signals of Opportunity (SOPs). The tightly-coupled approach, radio SLAM, presented in this paper utilized a cellular LTE radio signal to compensate GNSS outage for localization. The results show that the grounded vehicle traversing a trajectory of about 5 km in 180 s in the GPS-jammed environment, during which a GPS-IMU system drifted from the vehicle's ground truth trajectory, resulting in a position RMSE of 238 m. In contrast, the radio SLAM approach with single cellular LTE SOP whose position was poorly known achieved a position RMSE of 32 m.

1.2.3 Sensor fusion integrated with various filtering algorithms

In a fusion system, various filtering algorithms are available to address system nonlinearity, with the choice of algorithm depending on specific system requirements, such as estimation accuracy, inference time, ease of implementation, and algorithm robustness. Additionally, branches of filtering algorithms are often developed to address the limitations of their parent algorithms or to enhance their robustness. For instance, [22] introduces the application of the Square Root Unscented Kalman Filter (SR-UKF) for underwater target tracking. While the Unscented Kalman Filter (UKF) deterministically propagates samples based on the mean and covariance matrix P, numerical round-off errors can sometimes lead to a non-positive definite covariance matrix, causing estimation failures. The SR-UKF addresses this issue by ensuring the positive definiteness of the covariance matrix, thereby enhancing the accuracy and stability of the filtering process. In the study, UKF, SR-UKF, and SR-UKFs, which derived from the SR-UKF with backward smoothing aiming better accuracy, are implemented and compared. The result reveals that the proposed algorithms, SR-UKFs and SR-UKF, outperforms the traditional UKF algorithm and the SR-UKFs algorithm can reduce nearly 59% of the position error and nearly 54% of the velocity error than the SR-UKF algorithm. As another example, [23] applies the Rao-Blackwellized Particle Filter (RBPF) to handle system nonlinearity. It is well understood that in a Particle Filter (PF), the variance of the particles and their weights represent the system's uncertainty, making the PF particularly suitable for systems with nonlinear dynamics or noise. However, for high-dimensional systems, a large number of particles is required, which significantly increases the computational complexity of the PF algorithm as the state dimension grows, making real-time navigation challenging. The RBPF addresses this issue by partitioning the system's state into linear and nonlinear components, delegating the linear part to a traditional KF to reduce the required size of the particles. In the study, the system's state consists of 13 variables, with only one requiring a nonlinear model for estimation, while the remaining variables are managed by the KF. This approach improves computational efficiency while delivering enhanced estimation accuracy. An experiment comparing the performance of the standard PF and the proposed RBPF highlights these advantages. The traditional PF, using 50,000 particles, demonstrated suboptimal performance, requiring 2 hours to process 100 seconds of experiment data. In contrast, the RBPF achieved optimal estimation with just 500 particles, completing the same task in approximately one minute.

1.2.4 Sensor fusion integration with machine learning technologies

In the context of Machine Learning (ML)-based approaches, ML algorithms can be integrated with the Kalman Filter (KF) in two primary ways: dynamically tuning KF parameters or updating the state vector or measurements within the KF.

Tuning Parameters of the Kalman Filter

In this category, ML models are employed to dynamically adjust the parameters of the Kalman Filter, such as the noise covariance matrix or scale factors, throughout the estimation process, rather than relying on constant parameters. As an example of this category, the method provided by [24] introduced a CNN-based machine learning algorithm to estimate noise parameters encapsulated in the introduced Kalman Filter algorithm, Invariant Extended Kalman Filter, and tries to solve the IMU-only dead-reckoning localization problem for a vehicle. The 1D CNN takes IMU data as the input and predicts the scale factor of measurement noise for the lateral and upward velocities of the vehicle. The proposed method is applied to the KITTI dataset and evaluated using Relative Translation Error (t_{rel}) and Relative Rotational Error (r_{rel}). A batch of nine 1-minute sequences is sampled from 11 sequences in the dataset to train the ML model, with the objective of minimizing the relative translation error (t_{rel}). Experimental results demonstrate that the proposed method achieves relatively optimal performance compared to other algorithms integrated with all available sensors in the dataset.

Updating State Vector or Measurements in the Kalman Filter

Given the inherent challenges of sensor data, such as noise, errors, and occasional unavailability, ML models can be used to denoise sensor data or generate pseudo-measurements. Pseudo-measurements serve as auxiliary inputs not directly obtained from sensors but are synthesized using ML models and incorporated into the KF for state estimation.

For example, the method proposed by [25] estimates a state of the indoor drone racing system accurately using Temporal Convolutional Network (TCN) and relying only on an off-the-shelf IMU. The model takes buffered collective thrust and the angular velocity as an input and predicts displacement of the UAV position in a given time. Additionally, EKF is used to propagate the state according to the presented kinematic equation and the state is corrected by the machine learning model output. The proposed model is trained and validated on five trajectories selected from the publicly available Blackbird dataset [26]. Experimental results confirm the importance of integrating the EKF with the proposed machine learning model, referred to as IMU-Model Odometry (IMO), by comparing its performance against the ablated EKF and ML model. However, the model has a limitation that the estimation of the state works only in known, trained trajectory.

Another example is presented in [27], which introduces a UKF algorithm with two RNN models, specifically LSTM algorithm, to estimate a vehicle sideslip angle. The first model, referred to as the deep ensemble network, takes steering angle, velocity, and yawrate and lateral acceleration as an input vector and predicts estimated sideslip angle and its uncertainty. The second model, called as sensor filtering network, takes velocity, yawrate and lateral acceleration as a input vector and outputs filtered sensor signals, which are used to calculate the derivative of the sidelip angle used in the proposed kinematics-based model. Hence two models are introduced in different purposes to produce values used in the prediction step of the UKF and pseudo-measurements to correct estimation. For model training, data is collected from a simulation environment using CarSim across various velocity ranges and road surface conditions. The proposed algorithm is validated through comparisons with other algorithms in both simulation and real-car experiments. Results demonstrate that the proposed UKF algorithm, augmented with machine learning model outputs, achieves superior performance in sideslip angle estimation in both experimental setups.

1.3 Objectives

The objectives of this thesis are:

- To determine an optimal configuration of different Kalman filter algorithms, motion models for Kalman filters, and sensor inputs for the KITTI dataset. This study explores combinations of five widely used filters (EKF, UKF, PF, EnKF, and CKF), two motion model state equations (basic kinematics and velocity motion model), and sensor data inputs (IMU, VO from image sensors and GPS).
- To validate the robustness of the optimal configuration in a realistic environment. This work simulates realistic scenarios involving sensor data loss for GPS and VO along selected trajectories in the KITTI dataset, accounting for challenges such as non-line-of-sight (NLOS) and multipath effects for GPS, as well as limited terrain features and changing lighting conditions for VO.
- To assess the practicality of the filter algorithms for real-world applications. Filters are evaluated based on additional metrics such as inference speed and the number of parameters required for configuration, providing insights into their ease of implementation and usability in practical scenarios.

By addressing these objectives, this thesis will provide a comprehensive evaluation of these filters, demonstrates their applicability to practical environments, contributing valuable insights into the field of state estimation and sensor fusion.

Finally, the Python scripts and the jupyter notebooks that solve aforementioned questions are publicly available at the github repository[1].

1.4 Thesis Outline

The rest of the thesis are constructed as follows: Section 2 describes the systems overviews, which includes hardware settings and embedded sensor data, for each dataset. A detailed explanation of an algorithm for each filter technique is shown in Section 3. In Section 4, experimental setups and evaluation metrics for experiments that solves aforementioned research questions are presented. After showcasing the experiments, an analysis of the results is presented in Section 5. Finally, in Section 6 the conclusions of this work are drawn.

[1]`https://github.com/ishikawatatsuki/sensor_fusion`

2. System Overview

In the present study, we employ a publicly available dataset collected by Karlsruhe Institute of Technology and Toyota Technological Institute known as KITTI, to conduct our experimental research. KITTI dataset, is one of the most popular dataset for autonomous driving and robotics. It encompasses a diverse array of image data alongside a multitude of sensor readings across various traffic conditions. In this chapter, we highlight the hardware overview and sensors utilized in the datasets.

2.1 KITTI Vehicle

KITTI, Karlsruhe Institute of Technology and Toyota Technological Institute, is an open-source grounded vehicle driving dataset recorded in the various environments in Karlsruhe, Garmany in 2011 , including mid-size city, rural area, and highways. By driving the vehicle, specifically a Volkswagen Passat B6, with variety of sensors equipped as shown in Figure 1, a total of 3TB of data is collected from which a representative subset is selected for each task to evaluate, such as stereo and optical estimation benchmark, 3D visual odometry / SLAM, and 3D object detection and orientation estimation benchmark [28].

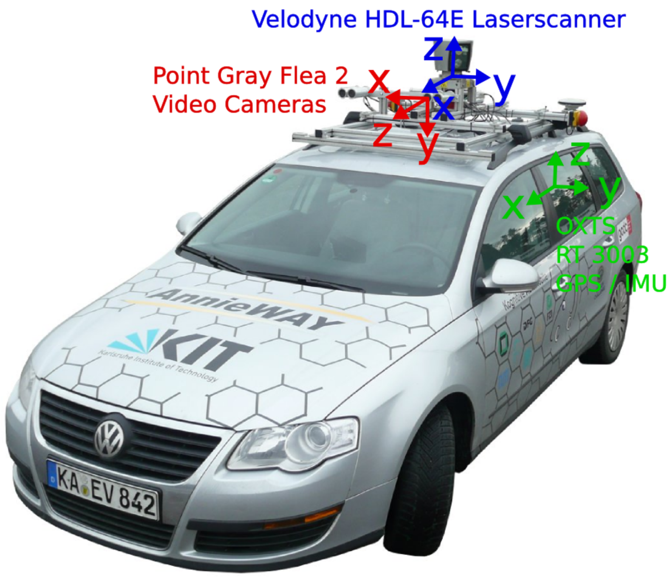
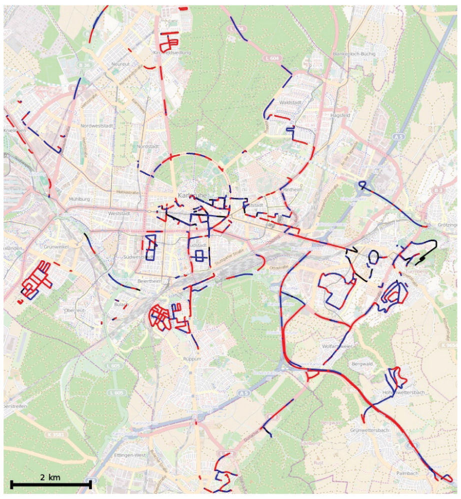
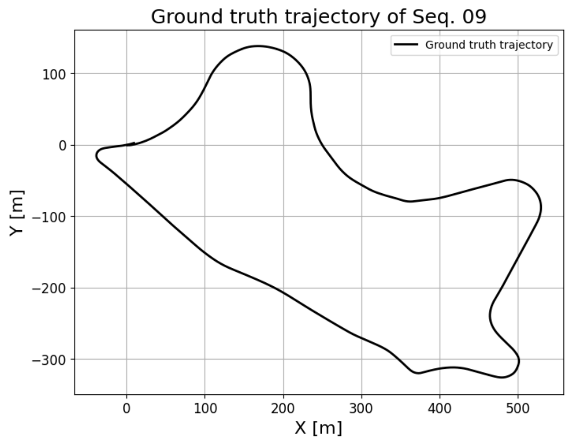


Figure 1. The KITTI vehicle [29].

KITTI raw dataset consists of approximately 25% of entire recordings separated by five majour categories: Road, City, Residential, Campus and Person [29]. Figure 2 displays the road tracks recorded during the drive in a metropolitan area of the city and the example of a GPS trajectory provided by KITTI raw dataset.



(a) Recording zone in Karlsruhe, Garmany in 2011 [29].



(b) An example GPS trajectory provided by KITTI raw dataset.

Figure 2. A recording zone with GPS traces in the metropolitan area of Karlsruhe traces and an example of GPS trajectory.

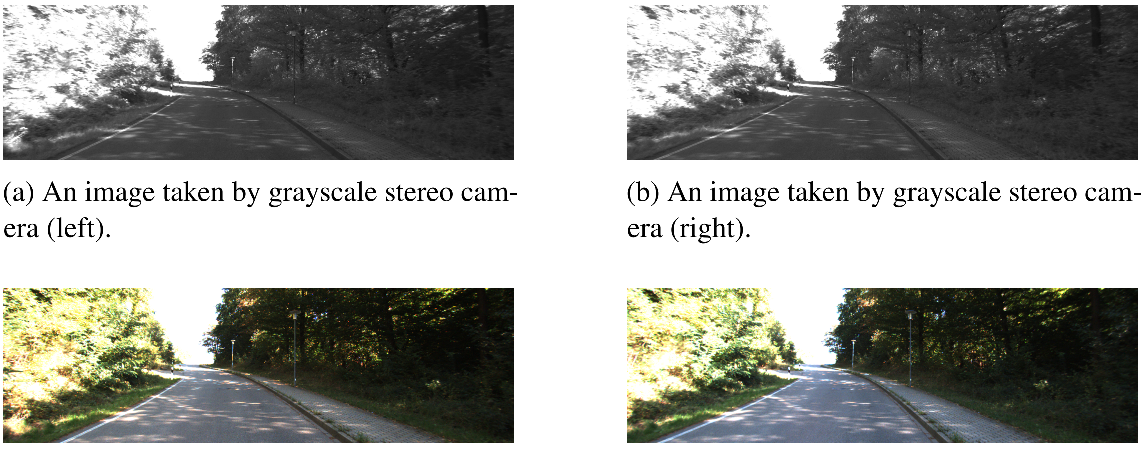
2.1.1 Sensor Setup

The detailed information for each sensor equipped with the vehicle is:

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

Stereo Cameras

As mentioned above, the vehicle is equipped with four image sensors, comprising a pair of color stereo cameras and a pair of grayscale stereo cameras. All cameras are synchronized with the Velodyne laser scanner at approximately 10Hz. The images captured by these cameras are provided as lossless compressed and rectified PNG sequences. An example image taken by both grayscale and color stereo is shown in Figure 3.


(a) An image taken by grayscale stereo camera (left).

(b) An image taken by grayscale stereo camera (right).

(c) An image taken by color stereo camera (left).

(d) An image taken by color stereo camera (right).

Figure 3. An example image taken by both grayscale and color stereo camera.

As shown in the Figure 3, stereo camera provides two images with slightly different angle, enabling depth estimation through stereo vision.

OXTS Inertial Navigation System (INS)

In the dataset, the setup uses the RT3000 series provided by OXTS, more precisely RT3000 v3 depicted in Figure 4. The system comprises high-performance GNSS-aided inertial navigation systems (GNSS/INS), which enables precise positioning, orientation, and motion data collection with maximum data output rate of 250 Hz. Due to its precision, it is widely used in automotive testing, including Advanced Driver Assistance Systems and autonomous vehicle evaluations. In KITTI raw dataset, position of the vehicle is provided by the GNSS and linear acceleration and angular velocity, with bias stability is $36deg/h$ and $1mg$ respectively, are sampled at 100Hz.

Figure 4. OXTS RT3000 v3 Inertial Navigation System (INS).

LiDAR (Velodyne)

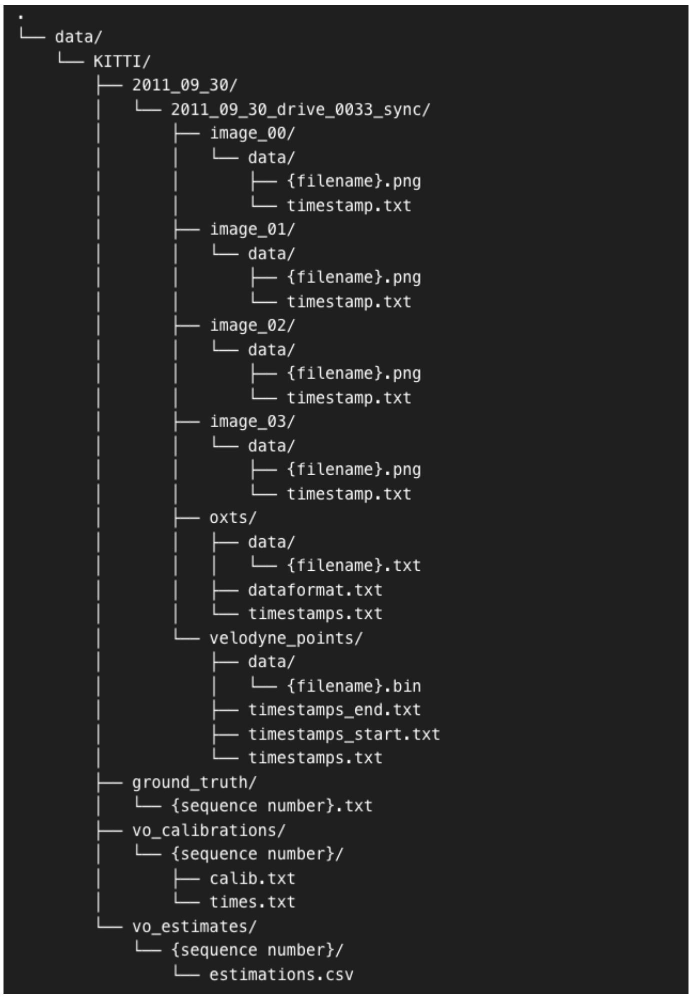
LiDAR is one of the Laser sensor, which applies the same working principle as radar and sonar such that it sends light and measures the time of flight (TOF) and/or compares the phase-shift of the light resources. In automotive application, LiDAR is integrated with GPS and IMU to improve the accuracy of the position of a vehicle. In KITTI dataset, LiDAR measures point cloud data, which consists of x, y, and z coordinate and reflectance information.

As an experimental dataset, we employed the KITTI raw data, which offers both synchronized and unsynchronized sensor data at a sampling rate of 10Hz. The dataset includes all aforementioned sensor data, along with calibration files and 3D object tracklet labels. The data is captured under diverse traffic conditions, such as in urban and residential areas. In addition, we mainly utilized synchronized sequences of image data captured by grayscale stereo camera and IMU/GPS sensor data provided by OXTS INS in our experiments.

2.2 KITTI Dataset

The KITTI raw dataset, which is available at the link[1], is downloaded by pykitti python library created by University of Tronto for loading and parsing the KITTI raw and the odometry datasets. The folder structure is shown in Figure 5.

[1]`https://www.cvlibs.net/datasets/kitti/raw_data.php`



```
.
└── data/
    └── KITTI/
        ├── 2011_09_30/
        │   └── 2011_09_30_drive_0033_sync/
        │       ├── image_00/
        │       │   └── data/
        │       │       ├── {filename}.png
        │       │       └── timestamp.txt
        │       ├── image_01/
        │       │   └── data/
        │       │       ├── {filename}.png
        │       │       └── timestamp.txt
        │       ├── image_02/
        │       │   └── data/
        │       │       ├── {filename}.png
        │       │       └── timestamp.txt
        │       ├── image_03/
        │       │   └── data/
        │       │       ├── {filename}.png
        │       │       └── timestamp.txt
        │       ├── oxts/
        │       │   ├── data/
        │       │   │   └── {filename}.txt
        │       │   ├── dataformat.txt
        │       │   └── timestamps.txt
        │       └── velodyne_points/
        │           ├── data/
        │           │   └── {filename}.bin
        │           ├── timestamps_end.txt
        │           ├── timestamps_start.txt
        │           └── timestamps.txt
        ├── ground_truth/
        │   └── {sequence number}.txt
        ├── vo_calibrations/
        │   └── {sequence number}/
        │       ├── calib.txt
        │       └── times.txt
        └── vo_estimates/
            └── {sequence number}/
                └── estimations.csv
```

Figure 5. The folder structure of KITTI dataset used in the experiments.

As shown in Figure 5, the synchronized KITTI raw data is downloaded and unzipped in the KITTI directory. The rectified images captured by the grayscale stereo camera are stored in *image_00* and *image_01* for the left and right views, respectively, while the color image sequences are stored in *image_02* and *image_03*. IMU/GPS data are stored in text file format and each folder comes with timestamp text file containing the timestamp of each frame of the sequence in nanosecond precision. In addition to KITTI data, Visual Odometry estimation for each sequence is given by [**vo-zazrul**] and its corresponding files including vehicle calibration files and ground truth are stored in the *vo_estimates*, *vo_calibrations*, and *ground_truth* folders, respectively.

IMU/GPS Data

In terms of GPS/IMU data, the values are stored in a single text file and each reading consists of 30 values collected in the NED frame, where x-, y-, and z-axis point to North, East and Down respectively. A description of each value is shown in the Table 1.

	Sensor value	Description	Unit
1	lat	latitude of the oxts-unit	deg
2	lon	longitude of the oxts-unit	deg
3	alt	altitude of the oxts-unit	deg
4	roll	roll angle, 0 is level and positive when left side up	rad
5	pitch	pitch angle, 0 is level and positive when front down	rad
6	yaw	heading, 0 is east and positive when counter clockwise	rad
7	vn	velocity towards north	m/s
8	ve	velocity towards east	m/s
9	vf	forward velocity, i.e. parallel to earth-surface	m/s
10	vl	leftward velocity, i.e. parallel to earth-surface	m/s
11	vu	upward velocity, i.e. perpendicular to earth-surface	m/s
12	ax	acceleration in x	m/s^2
13	ay	acceleration in y	m/s^2
14	az	acceleration in z	m/s^2
15	af	forward acceleration	m/s^2
16	al	leftward acceleration	m/s^2
17	au	upward acceleration	m/s^2
18	wx	angular rate around x	rad/s
19	wy	angular rate around y	rad/s
20	wz	angular rate around z	rad/s
21	wf	angular rate around forward axis	rad/s
22	wl	angular rate around leftward axis	rad/s
23	wu	angular rate around upward axis	rad/s
24	posacc	velocity accuracy north/east	m/s
25	velacc	velocity accuracy north/east	m/s
26	navstat	navigation status	
27	numsats	number of satellites tracked by primary GPS receiver	
28	posmode	position mode of primary GPS receiver	
29	velmode	velocity mode of primary GPS receiver	
30	orimode	orientation mode of primary GPS receiver	

Table 1. 30 sensor values collected by OXTS RT3000 v3 inertial and GPS navigation system.

Among the values shown in the Table 1, we are interested in lat, lon, alt for the GPS data and linear acceleration denoted by ax, ay, and az, and angular velocity described by wx, wy, wz collected by IMU.

Sensor Calibration

As shown in Figure 6, those sensors equipped on the vehicle are unaligned, which can cause large deviation during filtering process. Hence, rotational and translational operations are needed to be performed on measurement data such that the all the measurements are aligned with the body-frame.

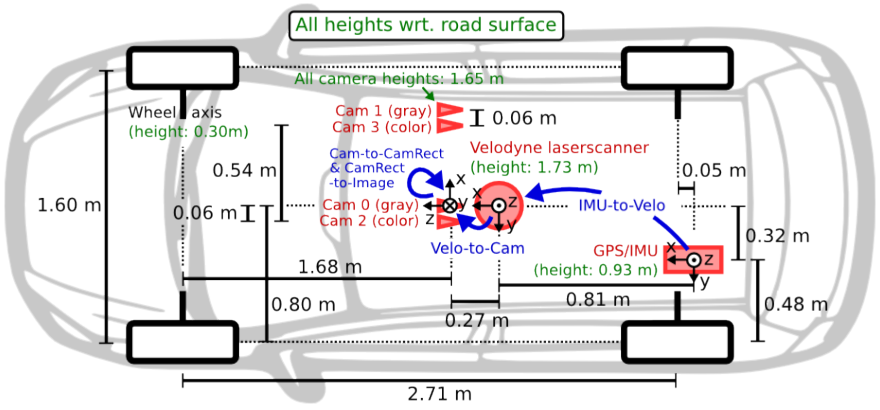


Figure 6. The top view of vehicle sensor setup.

KITTI dataset includes calibration files, which helps for transforming data in IMU frame to Velodyne frame, Velodyne frame to Camera frame, and one camera frame to another. Each calibration file contains nine values for rotation matrix and three values for translation to create a single *4x4* homogeneous transformation matrix, defined as:

$$T = \begin{pmatrix} R & t \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{12} & r_{22} & r_{23} & t_2 \\ r_{13} & r_{32} & r_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.1}$$

where R is a *3x3* rotation matrix and and t is a *3x1* translation vector. In addition, the transformation matrix, which translates data in IMU frame to Camera frame, is obtained by combining two transformation matrices from IMU to Velodyne and Velodyne to Camera, $T_{IMU \to Velodyne} \cdot T_{Velodyne \to Camera}$.

2.3 Visual Odometry

Visual Odometry (VO) is a type of localization technique that involves sequence of images taken by a single or multiple cameras to estimate the position and motion of an agent, such as grounded/aerial vehicle and robot. VO has some advantage over some other odometry techniques such as Wheel Odometry, which transform wheel's angular rotation to linear translation to estimate vehicle's position, and GPS due to tolerance for wheel slippage and capability of pose estimation in GPS-denied environment such as indoor and under-water conditions. However, VO has some constraints such as number of feature points derived from terrain texture and sufficient light source required in the environment [30]. VO can be classified by two categories: appearance-based approach, it extracts information from pixel intensity; feature-based approach, it firstly detects feature points and then computes a displacement of each matching feature points per frame.

Appearance-based approach, on the other hand, focuses on the information derived from the pixel intensity of an image. The commonly used technique in the appearance-based approach is template matching method that checks the existence and position of a template, or sub-image, in the entire image, so called search area. Likewise, the feature-based approach, degree of similarity between template and search area is computed using Sum of Squared Distance (SSD) and Normalized Cross Correlation (NCC). Once degree of similarity is measured, the pixel displacement between the template and the most similar search area is computed and then the pixel displacement is converted into the physical camera coordinate to compute camera motion between previous and current time. Generally, the Appearance-based approach outperforms feature-based approach in low-textured environments, however, it is susceptible to image occlusions [31].

Feature-based approach involves several steps: feature detection, feature description, feature mapping, and motion estimation. In feature detection step, SIFT (Scale Invariant Feature Transform) [32], SURF (Speeded Up Robust Feature) [33], ORB (Oriented FAST and Rotated BRIEF) [34] are the well known algorithms to be used due to its robustness and less computational efficiency. Once features are detected, the region around the feature points is converted into a descriptor that can be compared and matched with other descriptors. The descriptor needs to be stable against changes in lighting conditions, rotation, and scaling. In the feature matching step, a descriptor in the first image and another descriptor from the second image are compared based on a similarity measure, such as SSD and NCC [35]. Finally, the rotation and translation between the current image and the previous image are computed in the motion estimation step.

In research [36], a VO algorithm based on a feature-based approach is implemented. The method utilizes the SIFT algorithm as both a feature detector and descriptor, while a K-Nearest Neighbors (KNN) algorithm, combined with a brute-force matcher, is employed to identify the two closest matching feature points between the given images. Additionally, the Random Sample Consensus (RANSAC) algorithm [37] is used to robustly estimate the motion of the system, ensuring outlier rejection and improving the reliability of the motion estimation.

The aforementioned VO algorithm was chosen for its robustness in feature-rich environments, such as those found in the KITTI dataset, as well as its accessibility (the author of the paper is part of the same laboratory). This algorithm is applied to the stereo image sequences from the KITTI raw dataset, and the resulting estimates are utilized in the experiments presented.

3. Localization

Localization is the process of determining a robot's position and orientation in its environment. Based on the information of the initial position, localization problems can be classified into three main categories, position tracking, global positioning/localization, and kidnapped robot problem [38].

In the position tracking problem, robot starts with a known position and orientation in the environment, and the task is often to keep track of the robot's pose. In the global positioning problem, on the other hand, the robot's position needs to be approximated by utilizing sensor data with the condition that the robot position is initially unknown. The task of the kidnapped robot problem is to re-localize robot position after an unexpected movement of the robot.

In this research, we focused on tracking the robot's position by utilizing motion model and a combination of sensor data.

3.1 Kalman Filter

The Kalman Filter is a specific type of Bayesian filter that optimally utilizes prior information to estimate the state of a system. The term "state," denoted by x, where k represents a discrete time step, typically refers to variables such as a robot's pose and actuator configuration in robotics or the position and velocity of a vehicle in autonomous navigation. The Kalman Filter was designed to filter and estimate the state of linear systems, assuming that the system dynamics are linear and the noise follows Gaussian distribution.

Kalman filter generally consists of two main steps:

- Time update step (prediction)
- Measurement update step (correction).

In the time update step, the system's state $\hat{x} \in \Re^n$ and state error covariance matrix $P \in \Re^{n \times n}$ are projected from time step $k-1$ to k by the following equations:

$$\hat{x}_k^- = F_k \hat{x}_{k-1} + B_k \vec{u}_k + w_k \tag{3.1}$$

$$P_k^- = F_k P_{k-1} F_k^T + Q_k \tag{3.2}$$

where Equation 3.1, commonly referred to as the state equation, updates the state $\hat{x}_k^-$, representing the priori state estimate, using the control input vector $\vec{u}_k \in \Re^l$ at time step k. F is the $n \times n$ state transition matrix, B is the $n \times l$ control transition matrix, and w_k represents additive system or process noise. The noise w_k is assumed to be independent Gaussian white noise with zero mean, expressed as $w_k \sim N(0, Q_k)$, where Q_k is the process noise covariance matrix.

In the measurement update step, the posteriori state estimate $\hat{x}_k \in \Re^n$ a time k is computed with a measurement vector $z_k \in \Re^m$ and the priori state estimate $\hat{x}_k$ in the time update step. The equations are denoted by:

$$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + R_k)^{-1} \tag{3.3}$$

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H_k \hat{x}_k^-) \tag{3.4}$$

$$P_k = (I - K_k H_k) P_k^- \tag{3.5}$$

where the $n \times n$ matrix I denotes the identity matrix, $R_k \in \Re^{m \times m}$ is the measurement noise covariance matrix at time step k, and the $m \times n$ matrix H represents the output transition matrix, which maps values from the state space to the measurement space. The Kalman gain, K_k, determines the extent to which the estimates are adjusted based on the new measurement, aiming to minimize the posteriori state error covariance matrix P_k.

3.2 State equation

The core of localizing a vehicle's position is highly dependent on an embedded kinematic state equation, whereby the vehicle position at time *k* is estimated by propagating the previous vehicle' position at time *k-1* according to the motion model.

While the traditional Kalman Filter offers simplicity and computational efficiency, the complexity of vehicle motion often makes it unsuitable for accurately modeling system dynamics with linear functions. Therefore, in this section, we introduce two nonlinear motion models utilized in the experiments: the kinematic motion model, which is based on fundamental kinematic equations, and the velocity motion model, which describes a vehicle's motion in terms of two velocities, a translational and a rotational velocity.

3.2.1 Kinematic State Equation

In this section, the state equation that conform to the basics of kinematic equation is explained.

The Kalman Filter estimates the state of a system by propagating the state according to the motion model in discrete-time interval Δt.

Suppose we have a current state x_k, which is expressed in the following vector notation:

$$\hat{x}_k = [p_k, v_k, q_k]^T \tag{3.6}$$

where, the properties stored in the state vector are described below:

- $p_k \in \Re^3$ - A vector that contains position of the vehicle in inertial frame at time k.
- $v_k \in \Re^3$ - A vector that consists of the velocity of the vehicle at time k.
- $q_k \in \Re^4$ - A vector that holds rotational information of the vehicle represented in quaternion at time k. The rotational information is utilized to compute rotation matrix $R(q_k)$ that rotates data measured in the body frame to the inertial frame.

In the time update step of the Kalman Filter, control input vector $\vec{u}_k$ is fed to the state transition equation as shown in the Eq.3.1. Given that the control input vector $\vec{u}_k$ consists of two properties, linear acceleration and angular velocity, which are obtained from an IMU sensor, and the vector is denoted as $\vec{u}_k = [a_k, w_k]^T$, where a_k and w_k represent linear acceleration and angular velocity respectively.

According to the kinematics equations, displacement and velocity of the vehicle at time k are obtained by:

$$p_k^- = p_{k-1} + v_{k-1}\Delta t + \frac{1}{2}a_k\Delta t^2 \tag{3.7}$$

$$v_k^- = v_{k-1} + a_k\Delta t \tag{3.8}$$

According to [39], the following equation is derived for updating quaternion:

$$q_k^- = (cos(\frac{\|w_k\|\Delta t}{2})I_4 + \frac{1}{\|w_k\|}sin(\frac{\|w_k\|\Delta t}{2})\Omega_k)q_{k-1} \tag{3.9}$$

where, the norm of angular velocities about each axis, x, y, z is obtained as $\|w_k\| =$

$\sqrt{w_{x,k}^2 + w_{y,k}^2 + w_{z,k}^2}$, and the quaternion update matrix Ω_k at time k is obtained as:

$$\Omega_k = \begin{bmatrix} 0 & -w_k^T \\ w_k & -\lfloor w \rfloor_k \end{bmatrix} = \begin{bmatrix} 0 & -w_{x,k} & -w_{y,k} & -w_{z,k} \\ w_{x,k} & 0 & w_{z,k} & -w_{y,k} \\ w_{y,k} & -w_{z,k} & 0 & w_{x,k} \\ w_{z,k} & w_{y,k} & -w_{x,k} & 0 \end{bmatrix}$$

The representation of

$$\lfloor w \rfloor_k = \begin{bmatrix} 0 & -w_{z,k} & w_{y,k} \\ w_{z,k} & 0 & -w_{x,k} \\ -w_{y,k} & w_{x,k} & 0 \end{bmatrix}$$

expresses skew-symmetric matrix, which expands a vector $w_k \in \Re^3$ into a 3x3 matrix at time k.

Finally, the inertial navigation equations are written by:

$$p_k^- = p_{k-1} + v_{k-1}\Delta t + \frac{1}{2}\Delta t^2 (R(q_{k-1})a_k - g) \tag{3.10}$$

$$v_k^- = v_{k-1} + \Delta t (R(q_{k-1})a_k - g) \tag{3.11}$$

$$q_k^- = (cos(\frac{\|w_k\|\Delta t}{2})I_4 + \frac{1}{\|w_k\|} sin(\frac{\|w_k\|\Delta t}{2})\Omega_k) q_{k-1} \tag{3.12}$$

where the linear acceleration a_k measured in the body frame is rotated to the inertial frame by the rotation matrix $R(q_{k-1})$ derived from the quaternion q_{k-1} at time step $k-1$ and the gravitational acceleration denoted as g is compensated.

In the presented work, the effects of Earth rotation and the Coriolis acceleration are ignored. Moreover, the Earth is considered locally flat and the gravitational force vector $g \in \Re^3$ is set constant.

3.2.2 Velocity Model State Equation

According to [40], the velocity motion model describes the motion of a rigid body in terms of two velocities: rotational velocity and translational velocity.

Given a state vector x at time step k:

$$x_k = \begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix} \tag{3.13}$$

where x and y are coordinate in 2D space of the Cartesian coordinate and θ represents the angle along z-axis.

Now, the control input vector $\vec{u}_k$ of the system at time step k is composed of a translational and a rotational velocity, denoted by v_k and w_k respectively. Hence the vector is expressed as follows:

$$\vec{u}_k = \begin{bmatrix} \mathtt{v_k} \\ \mathtt{w_k} \end{bmatrix} \tag{3.14}$$

where positive rotational velocity w_k induces a counter clockwise rotation, left turn, whereas positive translational velocity v_k corresponds to the forward velocity of the vehicle.

We assume that the vehicle has circular motion when both velocities are kept constant in the time interval Δt, yielding the state vector after Δt of motion derived by:

$$\hat{x}_k^- = \begin{bmatrix} \mathtt{x_{c,k}} + \frac{\mathtt{v_k}}{\mathtt{w_k}}\mathtt{sin}(\theta_{\mathtt{k-1}} + \mathtt{w_k}\Delta\mathtt{t}) \\ \mathtt{y_{c,k}} + \frac{\mathtt{v_k}}{\mathtt{w_k}}\mathtt{cos}(\theta_{\mathtt{k-1}} + \mathtt{w_k}\Delta\mathtt{t}) \\ \theta_{\mathtt{k-1}} + \mathtt{w_k}\Delta\mathtt{t} \end{bmatrix} \tag{3.15}$$

where, $x_{c,k}$ and $y_{c,k}$ represent the center of circular motion at time step k calculated by:

$$x_{c,k} = x_{k-1} - \frac{v_k}{w_k} sin(\theta_{k-1}) \tag{3.16}$$

$$y_{c,k} = y_{k-1} + \frac{v_k}{w_k} cos(\theta_{k-1}). \tag{3.17}$$

By replacing x_c and y_c in the state vector equation 3.15, we obtain:

$$\hat{x}_k^- = \begin{bmatrix} \mathtt{x_{k-1}} - \frac{\mathtt{v_k}}{\mathtt{w_k}} * \mathtt{sin}(\theta_{\mathtt{k-1}}) + \frac{\mathtt{v_k}}{\mathtt{w_k}}\mathtt{sin}(\theta_{\mathtt{k-1}} + \mathtt{w_k}\Delta\mathtt{t}) \\ \mathtt{y_{k-1}} + \frac{\mathtt{v_k}}{\mathtt{w_k}} * \mathtt{cos}(\theta_{\mathtt{k-1}}) + \frac{\mathtt{v_k}}{\mathtt{w_k}}\mathtt{cos}(\theta_{\mathtt{k-1}} + \mathtt{w_k}\Delta\mathtt{t}) \\ \theta_{\mathtt{k-1}} + \mathtt{w}\Delta\mathtt{t} \end{bmatrix}. \tag{3.18}$$

KITTI dataset provides forward velocity vf available from the OXTS INS. However, the value of forward velocity is likely derived from a filtered norm of position displacement at each timestamp as shown in the Figure 7.

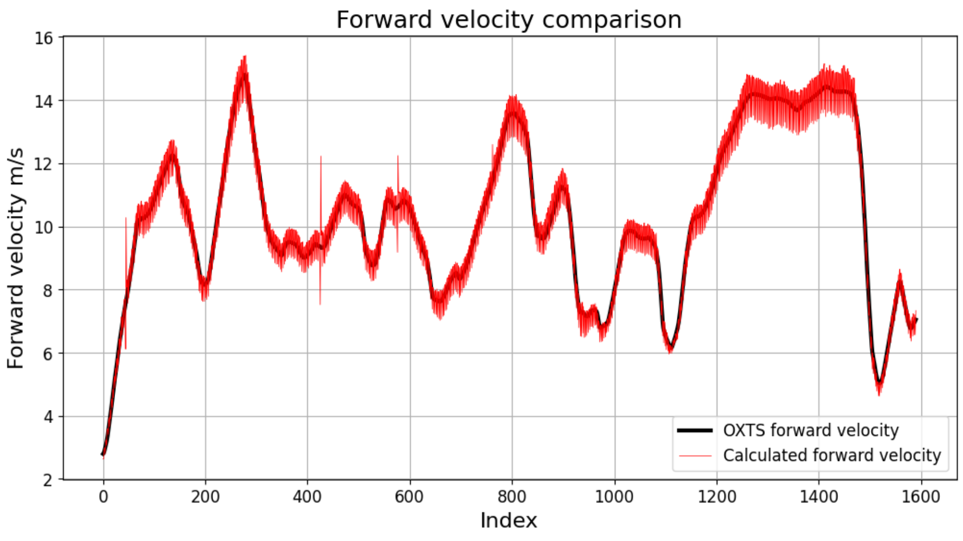


Figure 7. Comparison of forward velocity.

During a GPS outage, forward velocity data becomes unavailable, requiring the inclusion of velocities along all three axes in the state vector x. Additionally, to facilitate VO recovery when VO loses its tracking capability, a homogeneous transformation matrix T, representing the current translation and rotation relative to the original pose in the inertial frame, is required. Deriving this transformation matrix requires both the position vector and the rotation vector. Consequently, the state vector x is expanded to store the position, velocity, and quaternion components along all three axes, resulting in the following state vector:

$$\hat{x}_k = [p_k, v_k, q_k]^T \tag{3.19}$$

which coincides with the state described in the previous section.

To update the state vector x, the velocity and the quaternion vector are updated by Equation 3.12 with the control input vector $\vec{u}$, which is constructed by the IMU sensor data. In contrast, the position vector is updated by followings:

$$\hat{p}_k^- = \begin{bmatrix} \mathtt{p_{x,k-1}} - \frac{\mathtt{v_k}}{\mathtt{w_{z,k}}} * \mathtt{sin}(\psi_{\mathtt{k-1}}) + \frac{\mathtt{v_k}}{\mathtt{w_{z,k}}}\mathtt{sin}(\psi_{\mathtt{k-1}} + \mathtt{w_{z,k}}\Delta \mathtt{t}) \\ \mathtt{p_{y,k-1}} + \frac{\mathtt{v_k}}{\mathtt{w_{z,k}}} * \mathtt{cos}(\psi_{\mathtt{k-1}}) + \frac{\mathtt{v_k}}{\mathtt{w_{z,k}}}\mathtt{cos}(\psi_{\mathtt{k-1}} + \mathtt{w_{z,k}}\Delta \mathtt{t}) \\ \mathtt{p_{z,k-1}} + \frac{\mathtt{v_k}}{\mathtt{w_{x,k}}} * \mathtt{sin}(\phi_{\mathtt{k-1}}) + \frac{\mathtt{v_k}}{\mathtt{w_{x,k}}}\mathtt{sin}(\phi_{\mathtt{k-1}} + \mathtt{w_{x,k}}\Delta \mathtt{t}) \end{bmatrix} \tag{3.20}$$

where, ψ_k and ϕ_k denote the Euler angles at time k along the z-axis and x-axis, respectively. These Euler angles are derived directly from the quaternion vector for use in the velocity model. The forward velocity, v_k, represents the magnitude of velocity at time k and is calculated as the norm of the velocity components along the three axes.

3.3 Types of Kalman Filter

This section provides a comprehensive overview of five key variants of the Kalman Filter, including the Extended Kalman Filter (EKF), Unscented Kalman Filter (UKF), Particle Filter (PF), Ensemble Kalman Filter (EnKF), and Cubature Kalman Filter (CKF). Each filter is described in detail, highlighting its unique features, underlying principles, and the specific types of problems it is best suited to address, particularly in handling system nonlinearity, uncertainty, and high-dimensional state spaces.

3.3.1 Extended Kalman filter

Kalman filter is often used to estimate state vector, such as position of vehicle, in linear system. However, Kalman filter can only be applied to the linear system where current state vector has a linear depenency to the previous state vector. Extended Kalman filter, on the other hand, can estimate state vector in non-linear system, where output after transforming previous state vector is not gaussian, by approximating non-linear function with linear function using first-order Taylor expansion.

Taylor expansion consists of infinite sum of polynomials as follows:

$$f(x) \approx f(a) + (x-a) * f'(a)/1! + (x-a)^2 * f''(a)/2! + (x-a)^n * f^{(n)}(a)/n! \tag{3.21}$$

$$\approx \sum_{n=0}^{\infty} (x-a)^n * f^{(n)}(a)/n! \tag{3.22}$$

where $f^{(n)}(a)$ is n^{th} order derivative function of $f(x)$ evaluated at point a.

In Extended kalman filter, we are interested in the first derivative of the function $f(x)$ evaluated at μ. Hence, the approximated function is expressed by:

$$f(x) \approx f(\mu) + (x - \mu) * f'(\mu) \tag{3.23}$$

where, $f'(\mu) = \partial f(\mu)/\partial x$.

By Taylor expansion, we can approximate our function and obtain linearized function $f(x)$ evaluated at point μ so that Gaussian distribution is obtained after linear transformation of an input.

Now, we have linearlized state equation and measurement model as shown below:

$$f(x) \approx f(\mu) + (x - \mu) * f'(\mu) \tag{3.24}$$
$$h(x) \approx h(\mu) + (x - \mu) * h'(\mu) \tag{3.25}$$

Since we have nonlinear model for time update and measurement update step, those model cannot be directly applied to the covariance matrix. Thus, Jacobians are used instead of the function f and h for propagating the covariance matrix.

The Jacobian matrices are obtained as follows:

$$\mathbf{F}_j = \begin{bmatrix} \partial \mathtt{f}_1/\partial \mathtt{x}_1 \partial \mathtt{f}_1/\partial \mathtt{x}_2 ... \partial \mathtt{f}_1/\partial \mathtt{x}_\mathtt{n} \\ \partial \mathtt{f}_2/\partial \mathtt{x}_1 \partial \mathtt{f}_2/\partial \mathtt{x}_2 ... \partial \mathtt{f}_2/\partial \mathtt{x}_\mathtt{n} \\ ... \\ \partial \mathtt{f}_\mathtt{n}/\partial \mathtt{x}_1 \partial \mathtt{f}_\mathtt{n}/\partial \mathtt{x}_2 ... \partial \mathtt{f}_\mathtt{n}/\partial \mathtt{x}_\mathtt{n} \end{bmatrix} \tag{3.26}$$

$$\mathbf{H}_j = \begin{bmatrix} \partial \mathtt{h}_1/\partial \mathtt{x}_1 \partial \mathtt{h}_1/\partial \mathtt{x}_2 ... \partial \mathtt{h}_1/\partial \mathtt{x}_\mathtt{n} \\ \partial \mathtt{h}_2/\partial \mathtt{x}_1 \partial \mathtt{h}_2/\partial \mathtt{x}_2 ... \partial \mathtt{h}_2/\partial \mathtt{x}_\mathtt{n} \\ ... \\ \partial \mathtt{h}_\mathtt{m}/\partial \mathtt{x}_1 \partial \mathtt{h}_\mathtt{m}/\partial \mathtt{x}_2 ... \partial \mathtt{h}_\mathtt{m}/\partial \mathtt{x}_\mathtt{n} \end{bmatrix} \tag{3.27}$$

where $F_j \in \Re^{n \times n}$ and $H_j \in \Re^{m \times n}$ are Jacobian metrices for process model and measurement model.

Finally, the equations of Extended Kalman filter are explained as follows:

In the time update step:

$$\hat{x}_k^- = f(\hat{x}_{k-1}, \vec{u}_k) \tag{3.28}$$

$$P_k^- = F_{j,k} P_{k-1} F_{j,k}^T + Q_k \tag{3.29}$$

where, in contrast to Equation 3.1, the function f represents nonlinear function to propagate state vector x and the computed Jacobian matrix $F_{j,k}$ is used to update the error state covariance matrix P in the Extended Kalman Filter.

In the measurement update step:

$$K_k = P_k^- H_{j,k}^T (H_{j,k} P_k^- H_{j,k}^T + R_k) \tag{3.30}$$

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - h(\hat{x}_k^-)) \tag{3.31}$$

$$P_k = (I - K_k H_{j,k}) P_k^- \tag{3.32}$$

where Jacobian matrix $H_{j,k}$ is used to calculate the Kalman gain K_k and the posterior state error covariance matrix P_k. Additionally the nonlinear measurement model h, which performs a nonlinear mapping from the state space to the measurement space, is employed to compute the posterior state estimate.

3.3.2 Unscented Kalman filter

Unscented Kalman filter approximate nonlinear motion model by using unscented transform instead of analytical transform that EKF performs. The unscented transformation is a method for calculating the mean and covariance of a random variable by passing it through a nonlinear function.

Like other filtering technique, Unscented Kalman filter follows the time update and measurement update steps.

In the time update step, given a state vector $x \in \Re^n$, we compute a subset of points, known as sigma point and denited as $\chi_{(i)}$, where $i = 1...2n + 1$. Then a sigma vector χ_k at time step k is formed as follows:

$$\chi_{k-1}^{(i)} = \hat{x}_{k-1} \tag{3.33}$$
$$\chi_{k-1}^{(i)} = \hat{x}_{k-1} + (\sqrt{(n+\lambda) * P_x})_i, i = 1, ..., n \tag{3.34}$$
$$\chi_{k-1}^{(i)} = \hat{x}_{k-1} - (\sqrt{(n+\lambda) * P_x})_i, i = n+1, ..., 2n \tag{3.35}$$

where, λ is a caling parameter $\lambda = \alpha^2(n+\kappa) - n$. α determines the spread of the sigma points around the sample mean $\hat{x}$, which is normally small positive value such as 0.001. Moreover, κ is secondary scaling parameter, which is usually set to 0 [41].

The number of sigma points required to approximate the nonlinearly transformed distribution is $2n+1$. So, sigma points are chosen one point for the mean, and pair of points for the symmetrically distributed about the mean. Once sigma points are selected, the propagation of the sigma points through the nonlinear function f

$$\chi_{k-1}^{(i)} = f(\chi_{k-1}^{(i)}, \vec{u}_{k-1}, w_{k-1}) \tag{3.36}$$

where $\vec{u}_{k-1}$ and w_{k-1} are control input vector and process noise respectively.

After propagation of all sigma points, sample mean and sample covariance of the propagated points are computed according to the equation below:

$$\hat{x}_k^- = \sum_{i=0}^{2n} W_i^{(m)} \chi_{k-1}^{(i)} \tag{3.37}$$
$$P_k^- = \sum_{i=0}^{2n} W_i^{(c)} (\chi_{k-1}^{(i)} - \hat{x}_k^-)(\chi_{k-1}^{(i)} - \hat{x}_k^-)^T + Q_{k-1} \tag{3.38}$$

where, Q is additive process noise and the weights for the sigma points and their covariance are given by:

$$W_0^{(m)} = \lambda/(n+\lambda) \tag{3.39}$$
$$W_0^{(c)} = \lambda/(n+\lambda) + (1 - \alpha^2 + \beta) \tag{3.40}$$
$$W_i^{(m)} = W_i^{(c)} = 1/\{2(n+\lambda)\}; i = 1, ..., 2n. \tag{3.41}$$

The $W^{(m)}$ and $W^{(c)}$ indicates weight for the sigma points and covariance respectively. As explained, α and κ are set to scale the sigma points around the mean, in addition, β is used to incorporate prior knowledge of the distribution of x. For instance, $\beta = 2$ when the distribution is Gaussian.

In the measurement update step, after computing sample mean and covariance of propagated points, the measurements are predicted by applying the measurement function h to the updated sigma points

$$\mathcal{Y}_k^{(i)} = h(\chi_{k-1}^{(i)}). \tag{3.42}$$

After sigma points are transformed to the measurement space, the weighted mean denoted by $\hat{y_k} \in \Re^m$ and covariance matrix of predicted measurements $P_{y,k} \in \Re^{m \times m}$, and cross-covariance between propagated distribution and distribution of predicted measurement designated by $P_{xy,k} \in \Re^{n \times m}$ are obtained as follows:

$$\hat{y_k} = \sum_{i=0}^{2n} W_i^{(m)} \mathcal{Y}_k^{(i)} \tag{3.43}$$

$$P_{y,k} = \sum_{i=0}^{2n} W_i^{(c)} (\mathcal{Y}_k^{(i)} - \hat{y}_k)(\mathcal{Y}_k^{(i)} - \hat{y}_k)^T + R_k \tag{3.44}$$

$$P_{xy,k} = \sum_{i=0}^{2n} W_i^{(c)} (\chi_{k-1}^{(i)} - \hat{x}_k^-)(\mathcal{Y}_k^{(i)} - \hat{y}_k)^T \tag{3.45}$$

where R_k is an additive measurement noise at time step k.

Finally, the Kalman gain and correction of the system's state and state covariance matrix are performed by

$$K_k = P_{xy,k} P_{y,k}^{-1} \tag{3.46}$$

$$\hat{x}_k = \hat{x}_k^- + K_k (y_k - \hat{y}_k) \tag{3.47}$$

$$P_k = P_k^- - K_k P_{y,k} K_k^T \tag{3.48}$$

where y_k is sensor data in the measurement update step.

3.3.3 Particle filter

Particle filter is widely used for dynamic system's state estimation. While Extended Kalman filter provides analytical approximation to estimate the state of a dynamic system by linearizing non-linear motion models around the current estimate, Particle filter estimates the state of an intractable system probabilistically, integrating the Monte Carlo sampling and Baysian inference.

In Bayesian filter, the state estimate at time k given all measurements and control inputs up to time k is denoted by the following conditional probability density function (pdf) and is denoted as $p(x_k|u_{1:k}, z_{1:k})$, where $u_{1:k} = u_i; i = 1, ..., k$ represents the sequence of control input and $z_{1:k} = z_i; i = 1, ..., k$ indicates the sequence of measurement input.

However, storing the history of all the parameters requires exponentially growing computational power over time. Hence, to make computation trackable, it is assumed that the Bayes filter follows Markov assumption, meaning that the current state estimate only relies on the previous state and it is denoted as

$$p(x_k|x_{1:k-1}, z_{1:k-1}) = p(x_k|x_{k-1}) \tag{3.49}$$

$$p(z_k|x_{1:k}) = p(z_k|x_k). \tag{3.50}$$

In Bayesian theory, the posterior distribution is expressed by

$$p(A|B) = \frac{p(B|A)p(A)}{p(B)}. \tag{3.51}$$

When Bayes theory is applied to the Particle filter, the posterior probability density function of the state is obtained from

$$p(x_k|z_{1:k}) = \frac{p(z_k|x_k)p(x_k|z_{1:k-1})}{p(z_k|z_{1:k-1})} \tag{3.52}$$

where, the prior distribution is resulted from the combination of the posterior distribution at the previous time step, $p(x_{k-1}|z_{1:k-1})$, and the process model that describes how the state evolves over time in the prediction step. The prior distribution is denoted as

$$p(x_k|z_{1:k-1}) = \int p(x_k|x_{k-1})p(x_{k-1}|z_{1:k-1})dx_{k-1}. \tag{3.53}$$

Moreover, the likelihood $p(z_k|x_k)$ represents conditional probability of a measurement given the predicted state and the marginal likelihood is denoted as

$$p(z_k|z_{1:k-1}) = \int p(z_k|x_k)p(x_k|z_{1:k-1})dx_k. \tag{3.54}$$

The integrals represent that the probability is continuous distribution. However, the integrals can be solved under strong assumption such as the motion model is linear and the pdf is Gaussian [42].

Instead of approximating the posterior distribution by assuming that the motion model is linear, Particle filter approximates the pdf representing the posterior by a discrete pdf such that there are minimal restriction on the models involved.

The continuous pdf approximated by a sum of weighted samples are expressed as

$$p(x_{0:k}|z_{1:k}) \approx \sum_{i=1}^{N} w_k^i \delta(x_{0:k} - x_{0:k}^i) \tag{3.55}$$

where $\{w_k^i, x_{0:k}^i\}_{i=1}^N$ is a set containing the state sequence. δ is Dirac delta function, which is used to extend pdf to discrete random variable such that, for a discrete random variable χ with range $R_\chi = \{x_1, x_2, ..., x_n\}$ and the probability mass function (pmf) $p_\chi(x_k)$, the pdf is expressed as

$$f_\chi(x) = \sum_{x_k \in R_\chi} P_\chi(x_k)\delta(x - x_k). \tag{3.56}$$

According to the equation 3.55, the weights for each sample directly indicate the likelihood of the pdf $p(x_{0:k}|z_{1:k})$.

The challenge derived from the sample-based approximation is that the posterior distribution at time k is unknown, meaning that the samples need to be drawn from the other distribution. The distribution is called importance density, denoted by q. The weights compensate for the fact that the samples are drawn from the importance density q instead of the posterior pdf. The weights are computed by

$$w_k^i \propto \frac{p(x_k^i|z_{1:k})}{q(x_k^i|z_{1:k})} \tag{3.57}$$

where, the weights are equal to the ratio between the posterior and the importance density for an individual particle with index i.

In the recursive process, the state at time k, $p(x_k|z_{1:k})$, is of interest, rather than the full state sequence $p(x_{0:k}|z_{1:k})$ up to time k. The weights are rewritten as

$$w_k^i \propto w_{k-1}^i \frac{p(z_k|x_k^i)p(x_k^i|x_{k-1}^i)}{q(x_k^i|x_{k-1}^i, z_{1:k})}. \tag{3.58}$$

Thus, the posterior pdf at time k resulted from the Particle filter is expressed as

$$p(x_k|z_{1:k}) \approx \sum_{i=1}^{N} w_k^i \delta(x_k - x_k^i). \tag{3.59}$$

Resampling

One of the important step involved in the Particle filter is resampling particles. The aim of resampling is to propagate particles such that the particles from the state space at time k with large weights are more likely to dominate in the state space at time k+1 than particles with small weights. This results in the improvement in exploration of the state space after sampling [43].

After several iterations of the particle propagation process, the Particle Filter (PF) may encounter a problem known as Sample Degeneracy, where the weights become concentrated in a small number of particles, while the remaining particles carry negligible weights. When resampling is performed under these conditions, it can lead to Sample Impoverishment. In this scenario, only particles with large weights are retained and resampled, while particles with smaller weights are discarded [44]. This process reduces the diversity of the particles and results in a limited variance, making the particles less capable of accurately representing the posterior distribution.

To mitigate this issue, the Effective Sample Size (ESS), denoted as $\hat{N}_{eff}$, is introduced. According to [45], a logical computable approximation of Neff is provided by:

$$\hat{N}_{eff} = \frac{1}{\sum_i (w_k^i)^2}. \tag{3.60}$$

The $\hat{N}_{eff}$ has a range $1 \leq \hat{N}_{eff} \leq N$, where N is the number of particles in the system.

The resampling condition in the PF can be determined by defining a threshold, for example, $\hat{N}_{eff} < \frac{N}{2}$.

Resample algorithms

The main processes of general resampling algorithms are to generate random samples $\chi_{k+1} = \{x_{k+1}^i, w_{k+1}^i\}$ from the original random samples $\chi_k = \{x_k^i, w_k^i\}$ in the state space and to replace the new set of the samples with the previous samples.

The common resampling algorithms are listed below.

Multinomial resampling

Multinomial resampling is one of the most straightforward resampling algorithm that it generates N independent random numbers, denoted as u_k^n from the uniform distribution on (0, 1] and use them to select particles from χ_k.

In the nth selection, the particle x_k^m is chosen when the following condition is satisfied:

$$Q_k^{m-1} < u_k^n \leq Q_k^m, \tag{3.61}$$

where

$$Q_k^m = \sum_{i=1}^{m} w_k^i. \tag{3.62}$$

Thus, the probability of selecting x_k^m is the same as that of u_k^n being in the interval bounded by the cumulative sum of the normalized weights.

Residual resampling

Residual resampling algorithm has two stages. In the first stage, the particles whose weight is bigger than 1/N, are replicated deterministically. In the second stage, multinomial sampling using the remaining of the weights, which are referred to as residuals, is conducted.

In detail, number of replication of the highly weighted particles are deterimined by $N_k^m = Nw_k^m$, where N is total number of particles. Consequently, in the following stage, weights for each residual are computed by

$$\hat{w}_k^m = w_k^m - \frac{N_k^m}{N}. \tag{3.63}$$

Finally, the multinomial sampling is applied to the residuals using the computed weights.

Stratified resampling

Stratified resampling algorithm divides the entire population of particles into chunk of subpopulation, called strata. It partitions the $[0, 1]$ interval into N disjoint subintervals: $[0, 1/N) \cup [1/N, 2/N) \cup ... \cup [1 - 1/N, 1]$. The N random numbers, $\{u_k^n\}_{n=1}^N$, are drawn independently in each of these subintervals at time k as follows:

$$u_k^n \sim U\left[\frac{n-1}{N}, \frac{n}{N}\right), n = 1, 2, ..., N. \tag{3.64}$$

Then the bouding method based on the cumulative sum of normalized weights is performed.

Stratified resampling

Systematic resampling algorithm also performs the population division to obtain disjoint subintervals, but generation of the independent random number u_k^n is different from that of Stratified resampling. Firstly, u_k^1 is drawn from the uniform distribution within the interval of (0, 1/N], and the rest of the random numbers, $u_k^i; i \in 2, ..., N$, are obtained deterministically by the following procedure:

$$u_k^1 \sim U\left[0, \frac{1}{N}\right), \tag{3.65}$$

$$u_k^n = u_k^1 + \frac{n-1}{N}, n = 2, 3, ..., N. \tag{3.66}$$

The variance of the number of a resampled particle by the systematic method is smaller than that of the stratified algorithm.

3.3.4 Ensemble Kalman filter

Ensemble Kalman filter (EnKF), introduced by Evensen in 1994[46], is a Monte Carlo approximation of the Kalman filter (KF), in which system's state is represented by samples, a.k.a ensemble member.

In the case of a system that requires a large number of state variables such as numerical weather prediction where the number of variables is $n \approx O(10^7)$, the classic KF can become expensive to describe the system. The EnKF, on the other hand, estimate the state's uncertainty by the spread in the ensemble at a given time instead of using error covariance matrices [47], which reduces computational power required to estimate the system.

The main difference between EnKF and the other sequential Monte Carlo algorithm such as Particle filter is the use of linear updating rule. When new observation is available, each ensemble member is updated by a linear shift based on the assumption of a linear Gaussian motion model, while the Particle filter uses a reweighting or a resampling approach [48], which may experience high-dimensionality problem, called as "curse of dimensionality".

There are many different types of EnKF and they can be classified by two main categories, Stochastic algorithms, such as Perturbed Observation Kalman Filter, and Deterministic algorithms, such as Ensemble Transform Kalman Filter.

Commonly, in the time update step, forecast ensemble member is obtained by applying the motion model to each sample and denoted as

$$\hat{x}_k^{(i),-} = F_k \hat{x}_{k-1}^{(i)} + w_k^{(i)}, w_k^{(i)} \sim N(0, Q_k), i = 1, ..., N \tag{3.67}$$

where the $\hat{x}_k^{(i),-}$ represents the forecast ensemble member i at time step k and N is the ensemble size. The F_k is the state transition matrix that updates each ensemble member based on a given state equation and $w_k^{(i)}$ represents a process noise for ensemble member i, sampled from the zero mean multivariate Gaussian distribution with process noise covariance matrix Q_k.

After forecasting each ensemble member, an ensemble mean $\hat{x}_k^-$ and its covariance P_k at time step k are computed by:

$$\hat{x}_k^- = \frac{1}{N}\sum_{i=1}^{N} \hat{x}_k^{(i),-} \tag{3.68}$$

$$P_k = \frac{1}{N-1}\sum_{i=1}^{N} (\hat{x}_k^- - \hat{x}_k^{(i),f})(\hat{x}_k^- - \hat{x}_k^{(i),f})^T. \tag{3.69}$$

In the Perturbed Observation Kalman Filter (POKF), which is implemented in this study, observations are perturbed in order for the variance of the ensemble after the update step to correctly represent the uncertainty in the analysis [49]. The update step of the POKF is written as

$$\hat{K}_k = P_k H_k (H_k P_k H^T + R)^{-1} \tag{3.70}$$

$$\hat{x}_k^{(i)} = \hat{x}_k^{(i),f} + \hat{K}_k (y_k^{(i)} - H \hat{x}_k^{(i),f}) \tag{3.71}$$

where the Kalman gain is replaced by an estimate $\hat{K}_k$ based on the variance of ensemble [48]. $y_k^{(i)} = y_k + v_k^{(i)}$ is a perturbed observations and the measurement noise $v_k^{(i)}$ is Gaussian with zero mean and covariance, R_k, such that $v_k^{(i)} \sim N(0, R_k)$.

In the Deterministic algorithm, on the other hand, ensemble members are obtained by deterministically shifting the prior ensemble toward the posterior and scaling the ensemble so that the resulting ensemble has a smaller variance than the prior [48].

3.3.5 Cubature Kalman filter

Cubature Kalman filter (CKF) is introduced in 2009 by [50] that follows Spherical-radial Cubature rule allowing to select a set of cubature points to efficiently compute Gaussian weighted integrals represented in *nonlinear function* * *Gausssian probability density function*.

In the context of filtering and estimation, we often compute integrals of the form:

$$I = \int f(x) N(x|\mu, \Sigma) dx \tag{3.72}$$

where $f(x)$ is a nonlinear function of the state vector x, $N(x|\mu, \Sigma)dx$ represents a Gaussian probability distribution with mean μ and covariance Σ.

Direct computation of such integral is usually not feasible since $f(x)$ is nonlinear, hence, CKF draws a finite number of evaluation points, a.k.a. cubature points, and weights to approximate these integrals. This approach falls to the local approach including EKF and UKF, where we approximate the nonlinear system dynamics by using linearization and point evaluation; and global approach such as Particle filter, where instead of approximating the nonlinear system dynamics, draw samples in the state space, which decreases computational efficiency but achieves higher-order accuracy.

Consider the integral, especially in the form of:

$$I(f) = \int_{R^n} f(x) exp(-x^T x) dx \tag{3.73}$$

the following steps are applied to numerically compute the integral:

- to transform the integral function into the spherical-radial integration form,
- to propose a third-degree spherical-radial rule.

Firstly, the integral Eq.3.73 represented in Cartesian space is transformed into a spherical-radial coordinate system and the integral is represented by *radial integral* and *spherical integral*: The radial integral

$$I = \int_0^\infty S(r) r^{n-1} exp(-r^2) dr \tag{3.74}$$

the spherical integral

$$S(r) = \int_{U_n} f(ry) d\sigma(y) \tag{3.75}$$

where the Cartesian vector $x \in R^n$ is converted into a radius vector r and a direction vector y and it follows $x = ry$, $y^T y = 1$, $x^T x = r^2, r \in [0, \infty)$. Moreover, U_n is the surface of the sphere defined by $U_n = \{y \in R^n | y^T y = 1\}$ and $\sigma(\cdot)$ is the area element on U_n. Then those two integrals, spherical and radial integrals, are numerically computed by the spherical cubature rule and the Gaussian quadrature rule respectively [50].

Finally, the spherical and radial rules are combined to obtain accurate approximation of the integrals and ensures the cubature points are symmetrically distributed in the area of integration.

Filter derivation

In the time update step, firstly the state error covariance matrix is factorized by Cholesky decomposition such that:

$$P_{k-1}^{-} = L_{k-1} L_{k-1}^T. \tag{3.76}$$

Then the lower triangular matrix L is used to compute cubature points for $i = 1, 2, .., n_x$, where n_x represents the dimensions of the state:

$$\chi_{k-1}^{(i)} = \hat{x}_{k-1} + \sqrt{n_x} L_{k-1} \tag{3.77}$$

$$\chi_{k-1}^{(i+n)} = \hat{x}_{k-1} - \sqrt{n_x} L_{k-1}. \tag{3.78}$$

Subsequently, the cubature points χ are propagated through the nonlinear function and the predicted state is estimated by computing the mean value of the cubature points as follows:

$$\chi_k^{(i),-} = f(\chi_{k-1}^{(i)}, \vec{u}_k) \tag{3.79}$$

$$\hat{x}_k^{-} = \sum_{i=1}^{2n_x} W^{(i)} \chi_k^{(i),-} \tag{3.80}$$

where the weight of sigma point is: $W^{(i)} = \frac{1}{2n_x}$.

Finally, the state error covariance matrix P_k at time step k is computed as follows:

$$P_k^- = \sum_{i=1}^{2n_x} W^{(i)}(\chi_k^{(i),-} - \hat{x}_k^-)(\chi_k^{(i),-} - \hat{x}_k^-)^T + Q_k \tag{3.81}$$

where Q_k is an additive noise.

In the measurement update step, as similar to the time update step, factorize the state error covariance matrix and compute cubature points using the Eq.3.76 and Eq.3.88. Then the cubature points are propagated to observation space and the predicted state is computed by:

$$Z_k^{(i)} = h(\chi_k^{(i)-}, \vec{u}_k) \tag{3.82}$$

$$\hat{z}_k = \sum_{i=1}^{2n_x} W^{(i)} Z_k^{(i)}. \tag{3.83}$$

Subsequently, innovations covariance matrix and the cross-covariance matrix are computed according to the following equations:

$$P_{zz,k} = \sum_{i=1}^{2n_x} W^{(i)}(Z_k^{(i)} - \hat{z}_k)(Z_k^{(i)} - \hat{z}_k)^T + R_k \tag{3.84}$$

$$P_{xz,k} = \sum_{i=1}^{2n_x} W^{(i)}(\chi_k^{(i),-} - \hat{x}_k^-)(Z_k^{(i)} - \hat{z}_k)^T \tag{3.85}$$

where, R is additive noise. Finally, Kalman gain K at time step k is estimated and the corrections of the state of the system and state error covariance matrix are performed by:

$$K_k = P_{xz,k} P_{zz,k}^{-1} \tag{3.86}$$

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - \hat{z}_k) \tag{3.87}$$

$$P_k = P_k^- - K_k P_{zz,k} K_k^T \tag{3.88}$$

where z_k represents measurement input passed to the filter at time k.

4. Experiments

In this paper, KITTI dataset is used to estimate the trajectory of the vehicle in the spatial Cartesian coordinate system and results are compared to determine the optimal filter algorithms for the given dataset on the basis of three evaluation metrics, which is described at the end of the chapter. Moreover, we demonstrated multi-sensor environment that involves data drop during the trajectory simulating real-life scenario.

4.1 Experimental Setups

To address the research questions, following experimental setups are proposed.

- Evaluation of Filters using KITTI Dataset: The robustness of the filters is evaluated by testing them on selected sequences from the KITTI dataset, chosen based on their trajectory length and the extent of translation and rotation. The sequences analyzed include 03, 07, and 09. Furthermore, two distinct motion models, primarily utilizing IMU and VO data, are implemented to compare the resulting state estimations on the basis of the evaluation metrics. Moreover, we evaluate the inference speed of the filter algorithms, the ease of implementation, and usability in practical scenarios.
- Comparison of Estimated Trajectories Using Multi-Sensor Data: To evaluate the applicability of the filters in a multi-sensor environment, GPS data is incorporated alongside VO data during the measurement update step of each filter. This experiment provides insights into the estimation behavior of the filters when multiple sensor inputs are utilized.
- Simulating VO Data Drop: To further assess filter robustness, a real-world scenario is simulated using the KITTI dataset, wherein consecutive segments of VO estimations are intentionally omitted. This scenario mimics challenges such as frame loss, reduced feature detection caused by occlusions and lighting variations, and matching algorithm failures.
- Filter application in a real life scenario: A selected filter is now tested on a multi-sensor real life scenario where, consecutive GPS and VO data loss is simulated during the estimation and measure the applicability of the filter in real life environment.

4.1.1 Evaluation of Filters Using KITTI Datasets

Motion models

Two motion models: basics of kinematics and velocity motion model as described in Sec. 3.2, are implemented in time update step of each filter to propagate previous state estimate with given current control inputs, obtained from the IMU.

Sequence 03

Sequence 03, depicted in Figure 8a, comprises 801 data points (approximately an 80-second journey) and features complex motion patterns, including substantial rotations, sharp turns, and varying speeds.

Sequence 07

Sequence 07 includes 1101 data points (a 110-second journey) with several nearly-right angle rotations and large translation in x- and y-axis making closed loop as shown in Figure 8b. Moreover, the trajectory includes some stops and changes in speed, which makes estimation challenging.

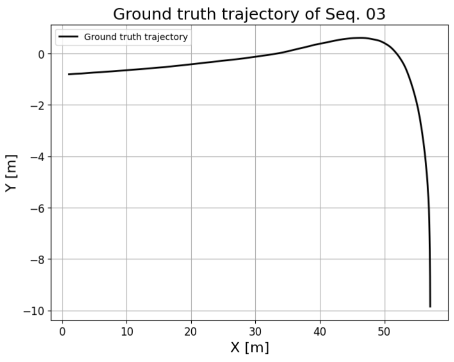
Sequence 09

Figure 8c shows sequence 09, which is composed of 1591 data points resulting in approximately a 160-second of trip with longer stretches of straight motion with occasional slight curves and minimal rotational movements formulating closed loop.

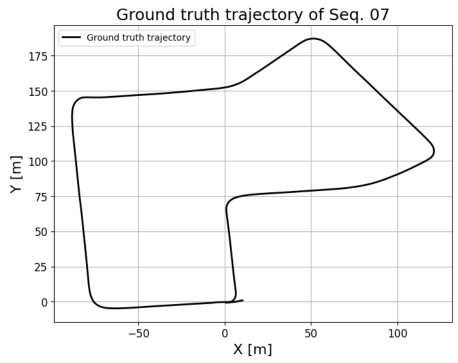
Figure 8 shows the trajectories for selected sequences from the KITTI dataset. The trajectories are derived from ground-truth GPS data, rotated to align with the inertial frame, which coincides with the initial IMU frame, and visualized in 2D coordinates (x and y). The state of each filter is initialized by the initial pose of the vehicle.

4.1.2 Comparison of Estimated Trajectories Using Multi-Sensor Data

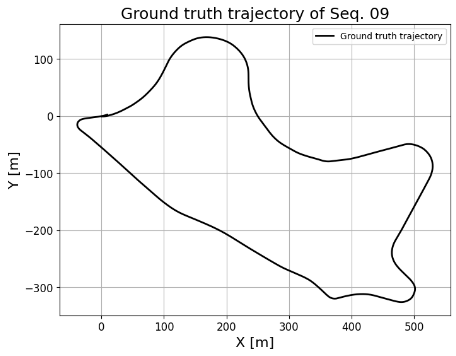
To assess the applicability of filters in a multi-sensor scenario, GPS data is incorporated into the measurement update step of the filters, and the resulting estimations are compared. In this experiment, the sensor noise configurations are set such that the measurement noise of the GPS is twice as low as that of the VO, reflecting the high accuracy of GPS data in the KITTI dataset, which is also used as ground truth. In addition to evaluating the performance of the filters, their behavior and responsiveness when handling data from multiple sensors are analyzed.



(a) The trajectory of seq. 03.



(b) The trajectory of seq. 07.



(c) The trajectory of seq. 09.

Figure 8. Trajectories of selected sequence in KITTI dataset.

4.1.3 Simulating VO Data Drop

In typical VO estimation, the transformation at time t, which includes both the translation and rotation matrices, is computed as the product of the transformation obtained at the previous time step $(t-1)$and the translational and rotational displacements derived from image sequences at time t. Consequently, the ego-motion's linear velocity at time t is calculated as the ratio of the translational displacement to the time interval (Δt) between the two image sequences.

In this study, the measurement update step of the Kalman filter incorporates both a position vector and a velocity vector to correct the system's state. To evaluate the robustness of the filters, simulations are conducted where VO data dropouts are introduced during the estimation process.

The performance of the filters is systematically analyzed by varying number of frame drops from 10 to 160 frames (from 1 to 16 seconds in duration) in two trajectories where rotational changes and translational changes are significant as shown in the Figure 9, and the results of the trajectory estimations are compared. In this experiment, sequence 09 is utilized due to its length and characteristics of the trajectory.

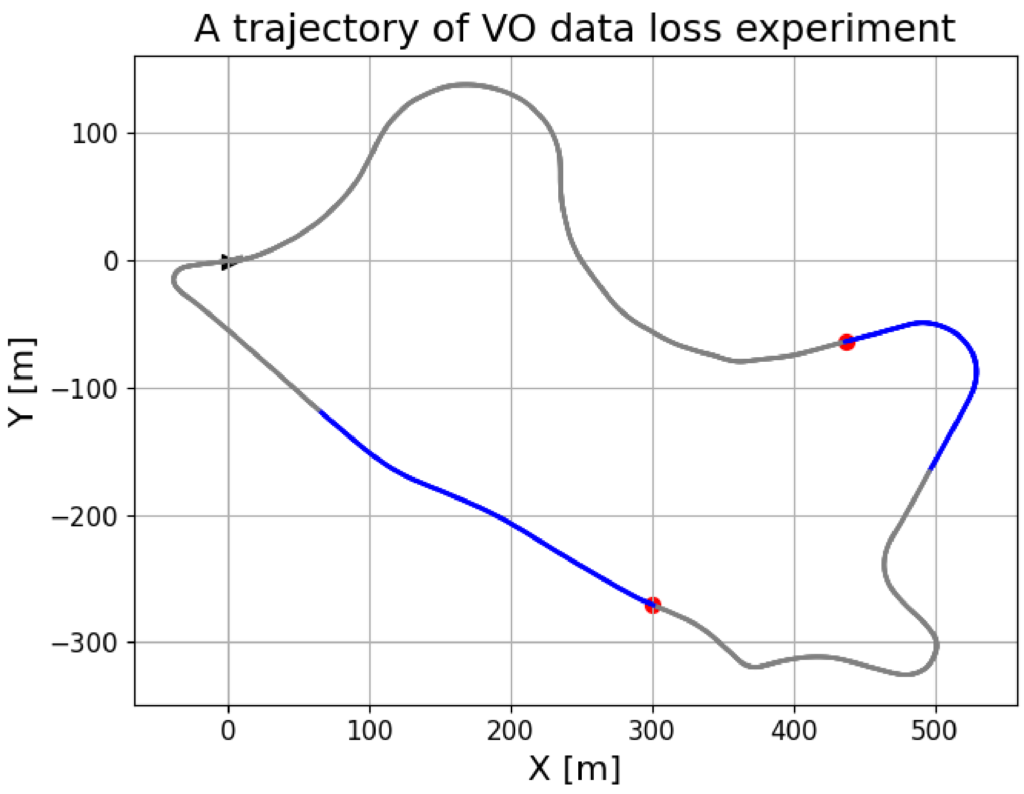


Figure 9. The trajectory highlighting the location of VO data loss.

Figure 9 illustrates the trajectory of KITTI sequence 09, highlighting the locations where VO data loss occurs. The red dots indicate the starting points of the data drop for the curved trajectory (frame 600) and the straight trajectory (frame 1200). Additionally, the number of dropped frames is progressively increased to 10, 20, 40, 80, and 160, allowing for the analysis of the resulting estimation error growth and the differences in impact based on the data drop locations.

4.1.4 Filter Application in a Real Life Scenario

Once aforementioned experiments are conducted, we selected EKF to estimate the vehicle's position in multi-sensor real life scenario. As an example trajectory, we selected sequence 09 in KITTI dataset due to its length and a variety of motion. In the trajectory, we consecutively dropped GPS and camera frame for VO data loss or both in the middle of the journey, which simulates a tunnel, less feature environments, a bridge, or combination of these environments to provide some difficulties during the estimation. The trajectory is drawn in Figure 10.

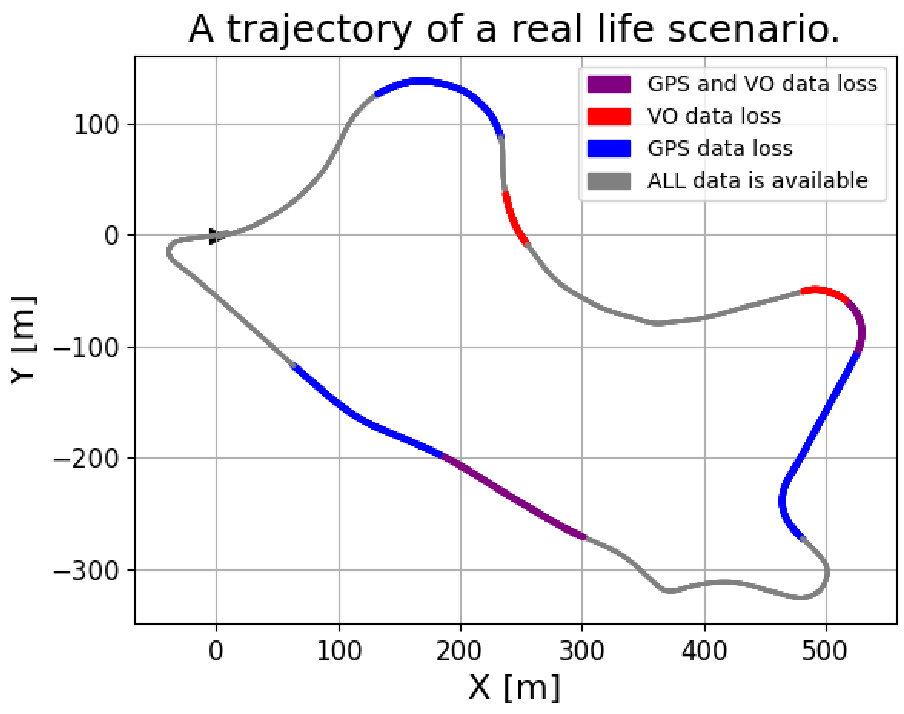


Figure 10. A trajectory of a real life scenario.

In Figure 10, the gray line represents the trajectory, where both GPS and VO data are available. The red and blue line indicate GPS data loss and VO data loss respectively. Finally, the purple colored line provide neither GPS nor VO data hence the filter performs IMU-only dead-reckoning to estimate the vehicle's position.

Finally, all the filters shares the same process noise covariance matrix Q and measurement noise covariance matrix R for corresponding sensors across all experiments.

4.2 Evaluation Metrics

The trajectory estimations are assessed using three quantitative evaluation metrics: Absolute Trajectory Error (ATE), Relative Pose Error in meter, and Relative Pose Error in degree, with reference trajectories derived from the KITTI dataset ground truth. The evaluation process is completed using the tools provided by [51]. Additionally, the inference time for a single iteration of the Kalman filter, encompassing both the time update and measurement update steps, is measured for each filter to highlight computational speed constraints.

4.2.1 Absolute Trajectory Error (ATE [m])

The Absolute Trajectory Error (ATE) is calculated by the difference between the estimated trajectory and the ground-truth trajectory after transforming the estimated position so that the estimation is aligned to the ground-truth, and the root mean square error is used

$$ATE_{pos} = \sqrt{\frac{1}{N}\sum_{i=0}^{N-1} ||\Delta p_i||^2}, \tag{4.1}$$

where N is the number of estimated points and Δp_i represents the estimated position at *i*th point.

ATE provides an intuitive error computation and it is easy to compare, however, the metric is time sensitive. The earlier an error is created, the larger the ATE becomes since the early error produces accumulated errors during the entire path of estimation.

4.2.2 Relative Pose Error (RPE [m/deg])

The Relative Pose Error (RPE) is computed by calculating the error of the end state of each sub-trajectory. The sub-trajectories are selected by defined length d and the error is computed for each sub-trajectory from the estimation and ground-truth after aligning the first state of each sub-trajectory of the estimation with that of the ground-truth. The PRE provides a collection of errors for all the sub-trajectories rather than a single value as the ATE, hence the statistics such as the median, average and percentiles can be derived [52]. To compute the error, firstly K set of length of sub-trajectories, $D = [d_1, d_2, .., d_K]$, are selected and then a set of sub-trajectories from both estimation and ground-truth for each length, $d_k = \{\hat{x}_{estimation}, \hat{x}_{ground-truth}\}$ where $\hat{x}$ represents a sub-trajectory, is derived. After that we compute the relative error for each sub-trajectories as follows:

$$RPE_{pos} = \frac{1}{D} \sum_{i=0}^{K-1} \delta_{p_k} \tag{4.2}$$

$$RPE_{rot} = \frac{1}{D} \sum_{i=0}^{K-1} \delta_{R_k} \tag{4.3}$$

where δp_k and δR_k represent a difference of relative position and rotation for sub-trajectory with the length k. The advantage of RPE over the ATE is that it is less sensitive to the large deviation at the beginning of the trajectory unlike the ATE, since its focus is the local consistency of relative motion.

5. Results and Analysis

The proposed variants of Kalman filter are implemented with Python 3.10. All filters employ the same state equation in the time update step of the filters and are initialized with the same state error covariance matrix P, process noise covariance Q and measurement noise covariance R for GPS and VO data in order to conduct aforementioned experiments under the same condition.

The filters that require parameters, such as UKF, PF, and EnKF, are initialized as follows:

- UKF: $\alpha = 1., \beta = 2., \kappa = 0.,$
- PF: $N_{\textit{particle size}} = 2048$, *stratified resampling algorithm*, $N_{th} = \frac{N}{2}$,
- EnKF: $N_{\textit{ensemble size}} = 256$,

where N_{th} represents the threshold for PF resampling and the particle size of PF is set to a large number to avoid filter divergence.

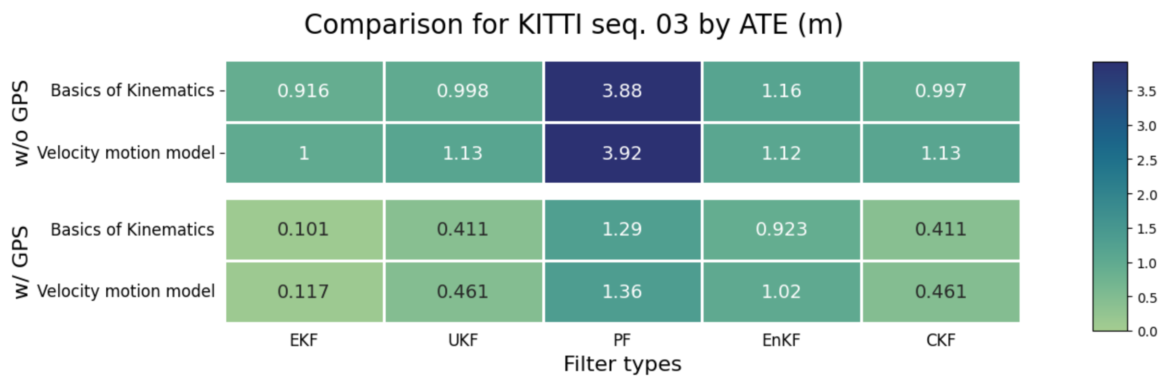
The first two experiments utilize pre-computed VO estimations, saved as numpy files in advance, to evaluate the inference speed of the filtering process for each filter.

5.1 Result of different trajectory experiment

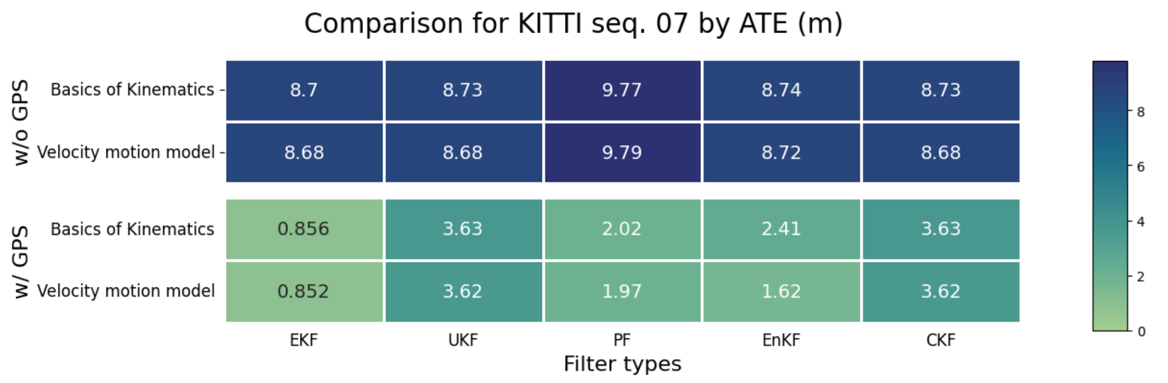
In this section, a performance comparison of the experimental results computed by the all filters with different motion models (with and without GPS as an input data) applying to the selected KITTI driving sequences, specifically sequence 04, 07 and 09, is provided. A quantitative and qualitative analysis of the estimated results in multi-sensor environment is discussed.

Analysis of global trajectory alignment and relative pose estimation

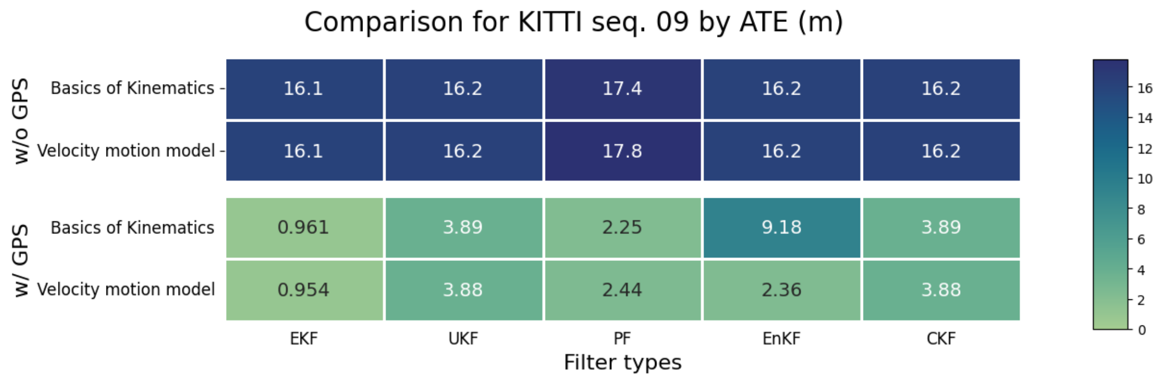
Figures 11 presents the Absolute Trajectory Error (ATE) in meters, primarily reflecting the global alignment between the estimated and ground-truth trajectories. Across all sequences, there are generally no significant differences in ATEs between the two motion models, as the provided VO data effectively corrects the filter's state estimates during the measurement update step.



(a) Heatmap representing ATE in meters for sequence 03.



(b) Heatmap representing ATE in meters for sequence 07.



(c) Heatmap representing ATE in meters for sequence 09.

Figure 11. Performance comparison of various filters with different motion models (with and without GPS data) for the selected KITTI sequences 03(11a), 07(11b), and 09(11c), evaluated by Absolute Trajectory Error (ATE).

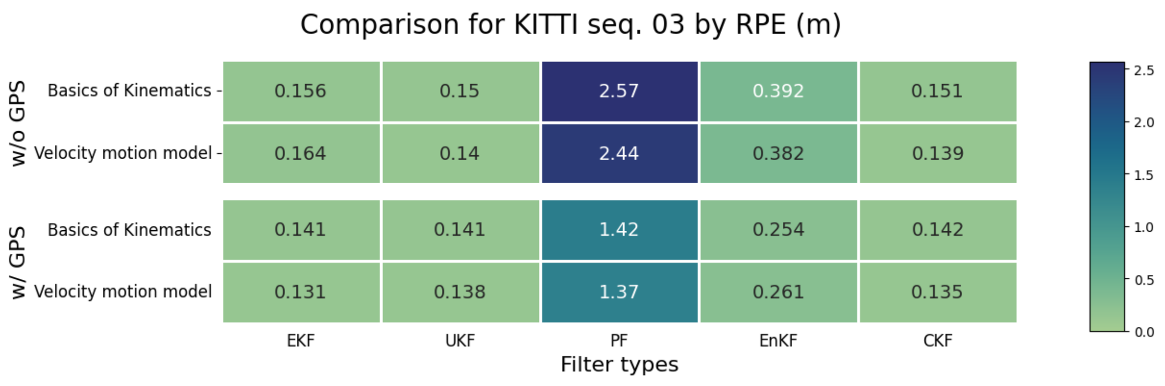
Furthermore, Gaussian-based filters, such as the EKF, UKF, and CKF, demonstrate higher accuracy compared to probabilistic filters like the PF and EnKF. For the EnKF, the initialization of the ensembles significantly impacts the filter's estimation, particularly with respect to rotational components. In this experiment, the ensembles are propagated using a multivariate distribution with a mean value corresponding to the initial state vector x

and a covariance equal to the initial state error covariance matrix P. This initialization introduces angular errors that propagate through the trajectory, affecting the ATE metric. If the angular errors in the state are not adequately corrected during estimation, they can impact the overall trajectory alignment.

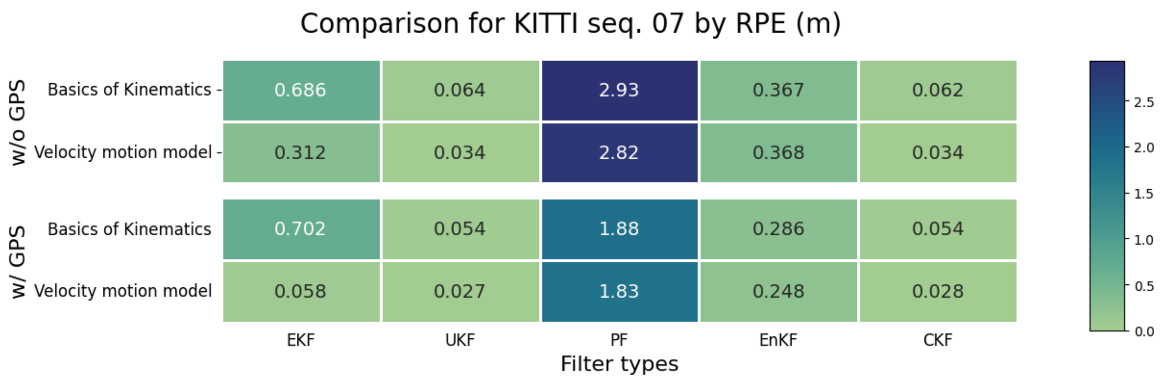
In terms of the Relative Pose Error (RPE) for translation as shown in Figure 12, EnKF's rotational initialization error is not the problem if error is computed segment by segment indicating that EnKF has acceptable performance in all kind of sequences. As it can clearly be seen, EKF, UKF, EnKF, and CKF estimate the position of the vehicle accurately, especially UKF and CKF, which utilize deterministic sampling technique capturing the system's nonlinearity better than the other, such as EKF, outperform the other filters.

As a characteristic of the PF, particles are propagated randomly around the initial state x, the weighted average is taken as a state estimate the state estimate is estimated by calculating weighted average of particles and the particles are resampled during the estimate.

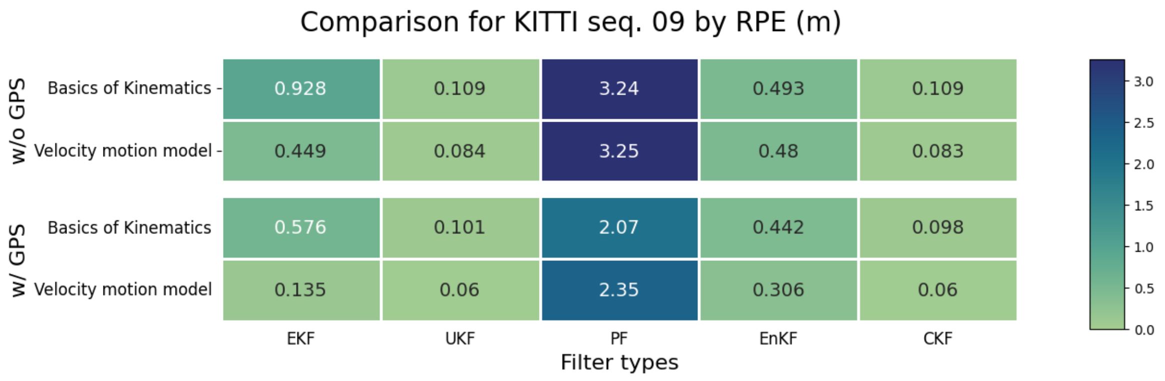
PF provides suboptimal performance in terms of RPE due to its reliance on the weighted averaging of particles to estimate the current state, combined with the resampling process. The state computed by weighted average provides moderate accuracy in terms of global trajectory consistency being expressed by ATE, however, this process causes tiny jumps of the state vector in the space, which leads to the accuracy drops in RPE. Hence, the characteristics of the PF make it less suited for capturing the fine-grained local dynamics required for precise relative pose estimation.



(a) Heatmap representing RPE for translation in meters for sequence 03.



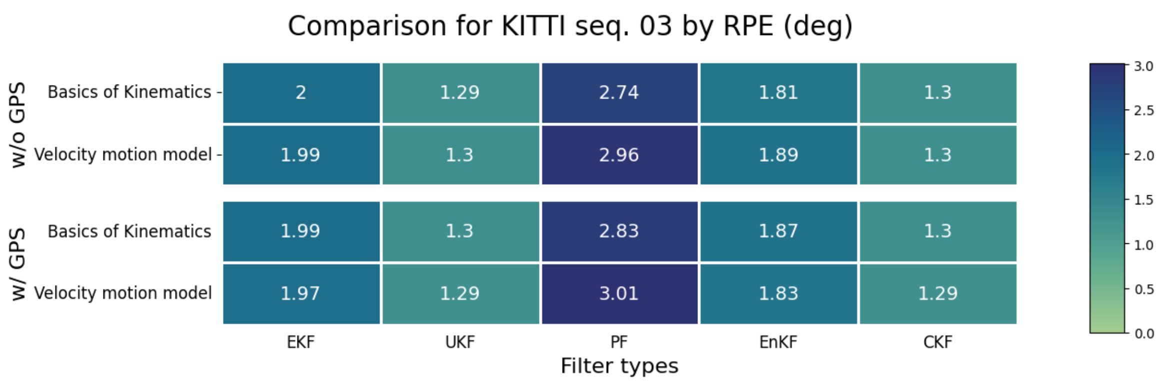
(b) Heatmap representing RPE for translation in meters for sequence 07.



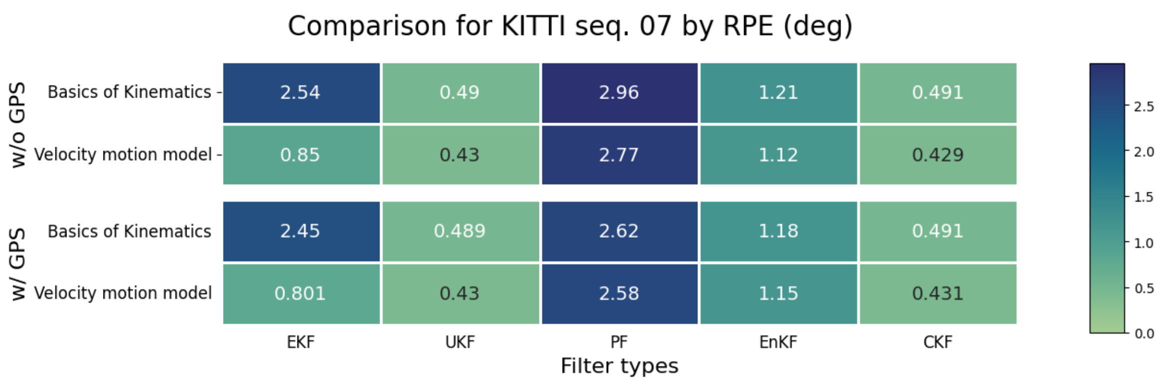
(c) Heatmap representing RPE for translation in meters for sequence 09.

Figure 12. Performance comparison of various filters with different motion models (with and without GPS data) for the selected KITTI sequences 03(12a), 07(12b), and 09(12c), evaluated by Relative Pose Error (RPE) for translation.

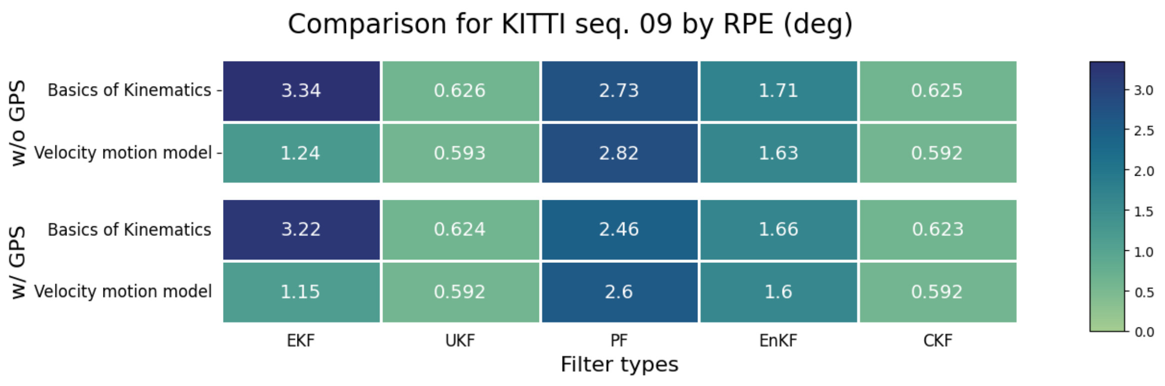
For RPE in angle estimation as depicted in Figure 13, the UKF and CKF consistently demonstrate superior performance across all sequences, while the EKF exhibits suboptimal results in this metric. Empirical observations reveal that the state of the EKF is highly influenced by the measurement update step, particularly where the rotational component, represented by the quaternion in the state vector, is affected by corrections to other elements, such as the position vector. This side-effect leads to degradation in the rotation estimation, as highlighted by the RPE metric. However, when a correction mask is applied to the innovation vector during the measurement update step to exclude irrelevant values, the rotational error of the EKF aligns closely with the performance of the UKF and CKF.



(a) Heatmap representing RPE for rotation in degree for sequence 03.



(b) Heatmap representing RPE for rotation in degree.



(c) Heatmap representing RPE for rotation in degree.

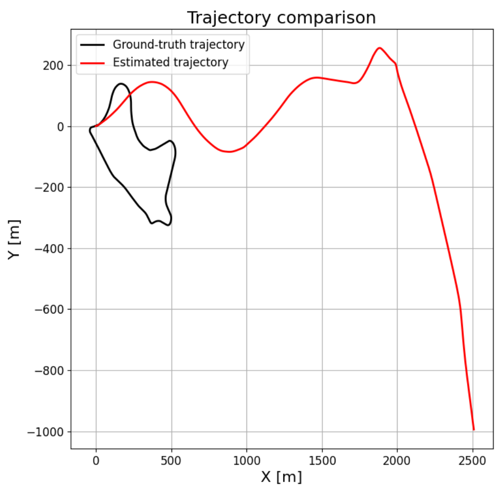
Figure 13. Performance comparison of various filters with different motion models (with and without GPS data) for the selected KITTI sequences 03(12a), 07(12b), and 09(12c), evaluated by Relative Pose Error (RPE) for rotation.

Analysis of different motion models

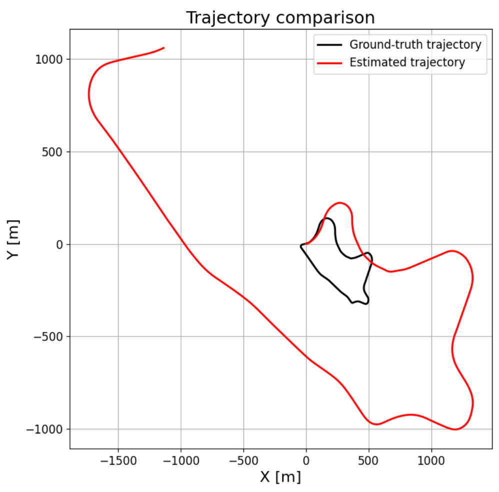
The ATE results indicate that there are no significant performance differences between the two motion models. However, in terms of relative errors represented by the RPE, the velocity motion model demonstrates superior performance across most filters and various sequences. This improvement is attributed to the fewer integrations required in the velocity motion model's state equations.

In the motion model based on basic kinematic equations, two integrations are involved to estimate the vehicle's position: first, from acceleration (provided by the IMU) to linear velocities along the three axes, and second, from the linear velocities to the vehicle's position. These multiple integration steps introduce cumulative errors that degrade the accuracy of the system's estimations [31], [53]. In contrast, the velocity motion model requires only a single integration. Specifically, angular velocities provided by the IMU are integrated to calculate delta angles, which are then used to estimate the vehicle's position. Additionally, linear accelerations are integrated with respect to the time step Δt, but the computed velocities are not directly used for pose estimation. Instead, they are stored in the state vector and corrected during the measurement update step using sensor data.

Furthermore, as illustrated in Figure 14, trajectories estimated by the EKF using IMU-only dead-reckoning demonstrate that, despite scaling issues, the velocity motion model's state equations outperform the pose estimation achieved by the basic kinematic motion model. Thus, it can be concluded that filters utilizing the velocity motion model generally provide more accurate estimations compared to those based on basic kinematic equations.



(a) Estimated trajectory using Basics of Kinematics Motion Model.



(b) Estimated trajectory by Velocity Motion Model.

Figure 14. Estimated trajectory by EKF IMU-only dead-reckoning using Basics of Kinematics Motion Model (14a) and Velocity Motion Model (14b).

Analysis of estimation result w/wo GPS input data and behavior of each filter

In addition to the performance comparisons across all filters and various error metrics, performance comparison of estimated trajectories obtained from applying filters with and without GPS input data are also provided by Figure 11, 12, and 13.

The result highlights that the inclusion of GPS input data significantly improves the estimation accuracy for all sequences. Beyond the accuracy of the estimation, the filters' responses to the introduction of new sensor data provide valuable insights into their applicability in practical scenarios.

The EKF, PF, and EnKF demonstrate substantial improvements in accuracy when GPS data is integrated, resulting in excellent performance in terms of Absolute Trajectory Error (ATE). Conversely, the UKF and CKF generally exhibit slightly lower accuracy compared to the other three filters.

Figure 23 illustrates the trajectory comparisons produced by these filters, with and without GPS data, for sequence 07 of the KITTI dataset. As shown in Figures 23b, 23f, and 23h, the red-colored estimated trajectories from the EKF, PF, and EnKF closely align with the GPS data. This behavior can be attributed to the dominant influence of the correction step with GPS data, which has significantly lower measurement noise than the VO data. Since the KITTI dataset uses GPS as ground truth, this results in highly accurate trajectory estimations. However, this reliance introduces a potential drawback: the filter may over-rely on a single sensor's data (in this case, GPS), even when it is not ground truth but has a smaller error compared to other sensors. This could lead to an imbalanced filter estimation.

In contrast, the estimations provided by the UKF and CKF, as shown in Figures 23d and 23j, display a different characteristic. The red-colored trajectories estimated by these filters are positioned between the GPS and VO data, reflecting their ability to balance sensor inputs based on the measurement noise, with the GPS data weighted twice as heavily as the VO data. This demonstrates that the UKF and CKF consider both sources of information proportionally, offering a more balanced fusion process. This behavior highlights their adaptability and allows for fine-tuning of estimation by dynamically adjusting the noise parameters of each sensor.

Therefore, while the UKF and CKF may show suboptimal accuracy in terms of ATE compared to the EKF, PF, and EnKF, their balanced handling of sensor data and intuitive behavior make them more suitable for practical multi-sensor environments, where dynamic noise adjustment and flexibility are critical.

Analysis of inference time

Figure 15 presents a comparison of the inference times for different filters applied to various sequences of the KITTI dataset. The results demonstrate that Gaussian-based filters, such as EKF, UKF, and CKF, offer computational efficiency, requiring less than 10 milliseconds to complete one processing cycle, including the time update and measurement update steps. In contrast, the EnKF, while specifically designed for high-dimensional state estimation problems (e.g., geoscientific applications), is less suitable for simpler state estimation tasks like ground vehicle localization due to its relatively higher computational overhead. Similarly, the PF involves computationally intensive operations, such as calculating likelihoods for each particle based on the probabilistic distribution of sensor data and performing resampling based on particle weights. Additionally, the inference time of the PF increases with the number of particles. Given these characteristics, the PF and EnKF are generally not an optimal choice for systems requiring real-time solutions.

Conclusion of the section

In summary, considering factors such as overall accuracy in terms of ATE and RPE, the inference time required to complete one filtering cycle, and the intuitively interpretable behavior of the filter in a multi-sensor environment, the CKF emerges as a suitable choice for solving problems like ground vehicle localization.

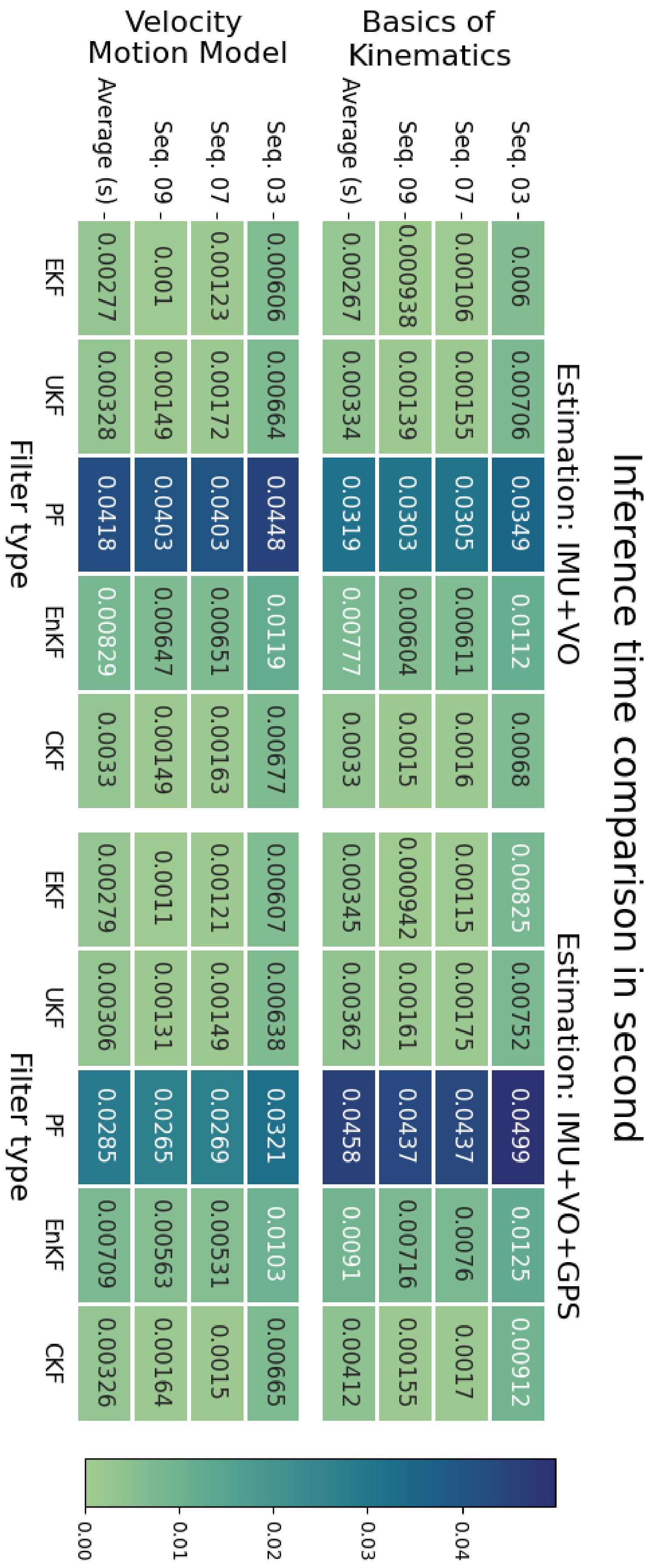


Figure 15. Heatmap representing a comparison of inference times for various filters across different sequences measured by 2021, M1 Macbook Pro.

5.2 Result of VO data drop experiment

So far, we evaluated various filters using different sequences fo trajectory in KITTI dataset with mono- or multi-senor scenario and revealed that EKF, UKF and CKF provides the highest accuracy in terms of selected metrics.

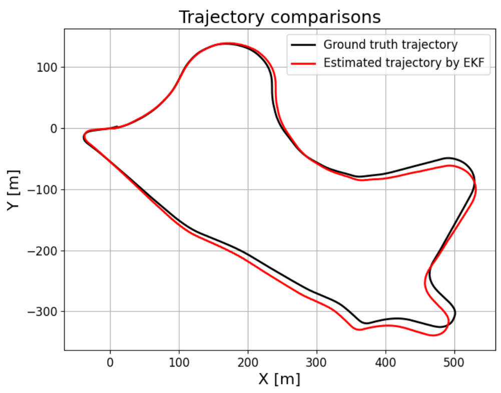
In this section, we selected the most optimal algorithm, CKF, and additionally EKF is used as a baseline of the experiment to work with. Here, the selected filters are applied to two different driving scenario that contains various durations of VO data drop in two locations of the trajectory, including curved and straight line. The estimated trajectories provided by each filter for each scenario are systematically analyzed for the experiment.

The figures depicted in Fig. 16 presents the estimated trajectories with varying durations of VO data loss in curved trajectory starting from the frame 600 in sequence 09 of KITTI dataset. The figures on the left, including 16a, 16c, and 16e, show the performance of EKF estimated trajectories, while the figures on the right, including 16b, 16d, and 16f, are provided by CKF.

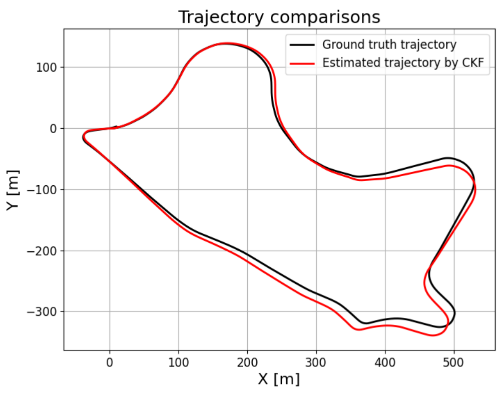
A noticeable deviation on estimations by both EKF and CKF can be observed as duration of data loss increased in curved trajectory. Moreover, CKF faces sever scaling issue when VO data loss is more than 8 seconds, which requires some semantic constraints for a specific application such as setting constant velocity during the curved trajectory in the application like grounded-vehicle localization.

Fig. 17 shows the estimated trajectories using EKF and CKF with data drops along a straight trajectory starting from the frame 1200 in sequence 09 of KITTI dataset. The figures on the left, comprising 17a, 17c, and 17e, illustrate the EKF's estimated trajectories. The figures on the right, including 17b, 17d, and 17f, depict the estimated trajectories obtained by CKF.

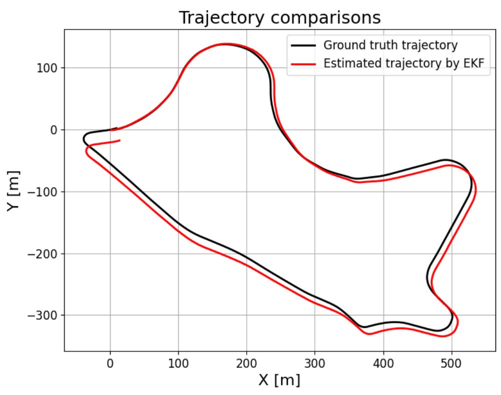
Similar to the data loss observed in the curved trajectory, the translation error accumulates as the duration of the VO data drop increases. Specifically, during data loss at the straight trajectory, the EKF underestimates the translation, resulting in a trajectory that is shorter than the actual one. This occurs due to incorrect velocity estimation from the IMU acceleration in the absence of correction. Conversely, the CKF exhibits a large positive error during the data loss period, where the estimated velocity surpasses the actual velocity,



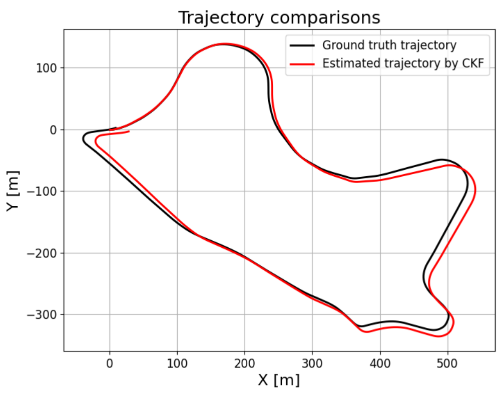
(a) No VO data drop at curved line for EKF.



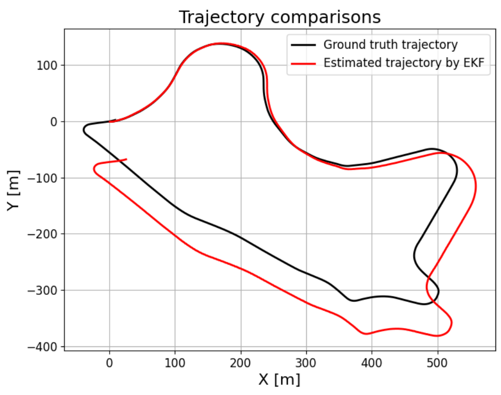
(b) No VO data drop at curved line for CKF.



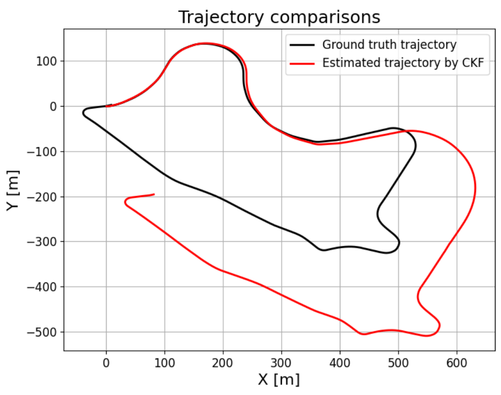
(c) 40 frames (4 seconds) of VO data drop at curved line for EKF.



(d) 40 frames (4 seconds) of VO data drop at straight line for CKF.



(e) 160 frames (16 seconds) of VO data drop at curved line for EKF.

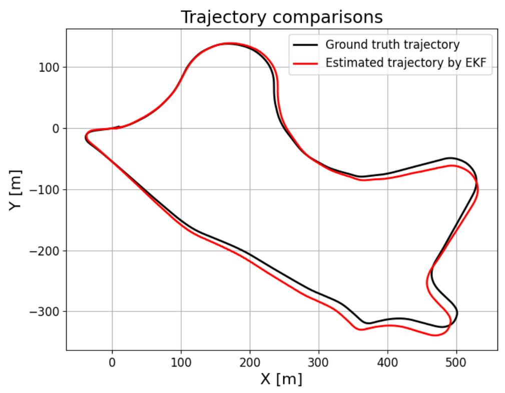


(f) 160 frames (16 seconds) of VO data drop at curved line for CKF.

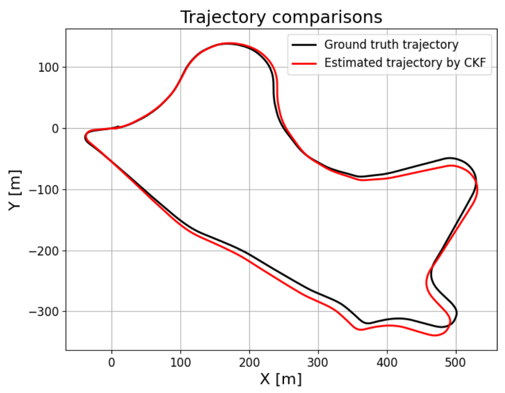
Figure 16. Estiamted trajectories of KITTI sequence 09 by EKF and CKF with VO data drop at different durations during curved trajectory.

leading to a deviation of endpoint.

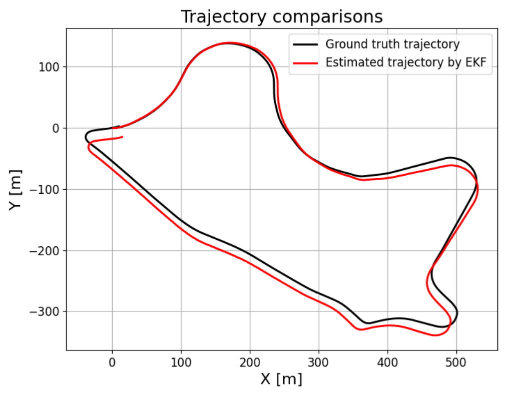
To quantitatively evaluate the estimations using the defined error metrics, aforementioned



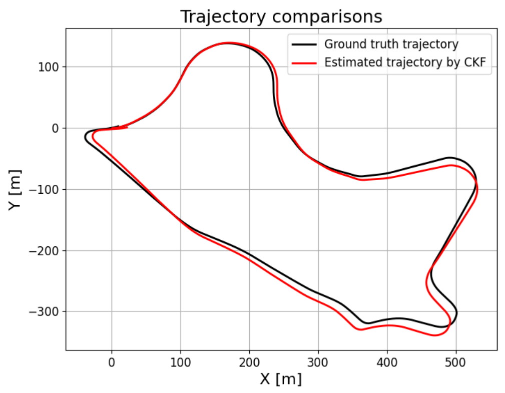
(a) No VO data drop at straight line for EKF.



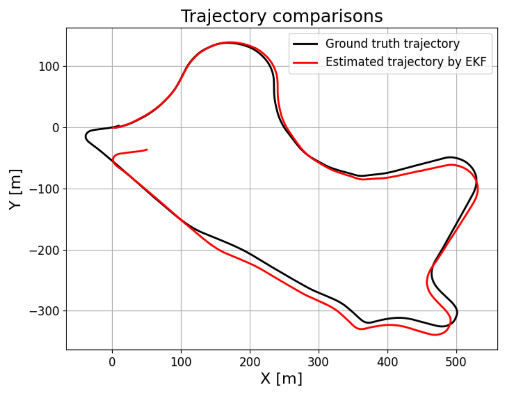
(b) No VO data drop at straight line for CKF.



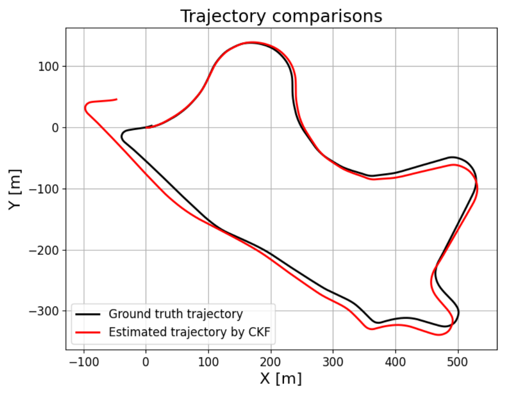
(c) 40 frames (4 seconds) of VO data drop at straight line for EKF.



(d) 40 frames (4 seconds) of VO data drop at straight line for CKF.



(e) 160 frames (16 seconds) of VO data drop at straight line for EKF.

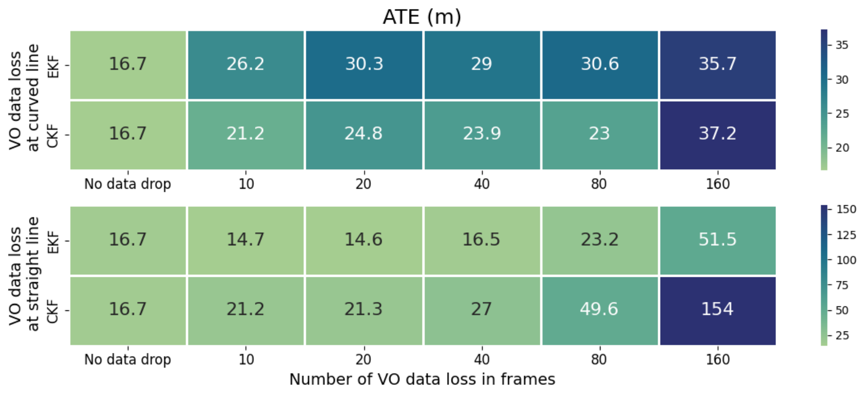


(f) 160 frames (16 seconds) of VO data drop at straight line for CKF.

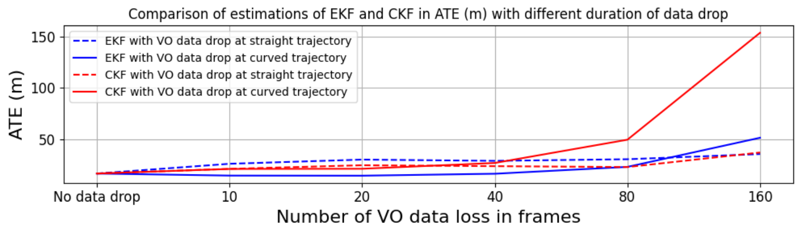
Figure 17. Estiamted trajectories of KITTI sequence 09 by EKF and CKF with VO data drop at different durations during straight trajectory.

evaluation tool is applied to exported pose data.

Figure 18 explains that the ATE increases as the VO data drop increases for both straight and curved line. The growth of the ATE is significantly different between locations of data drop, where , for both filters, drop at curved trajectory increases ATE drastically after 80 frames while ATE of the data drop at straight line slowly increases during the experiment. Moreover, as discussed, CKF is suffered by scaling issue severely, especially the data drop at the curved trajectory, the heatmap highlights the error of CKF clearly.



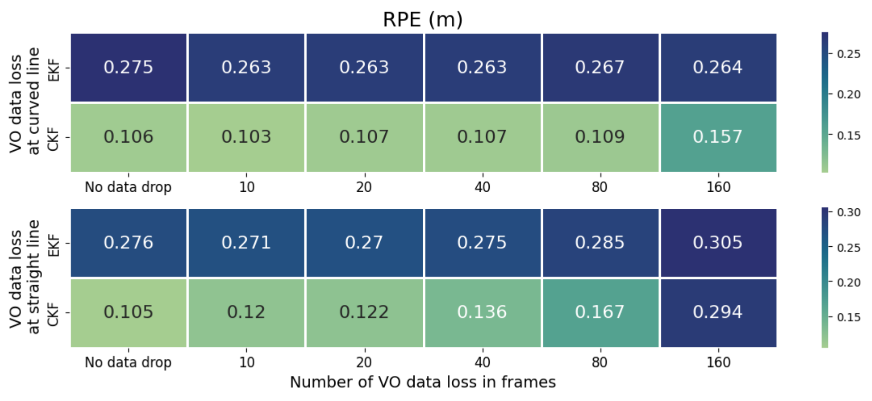
(a) Heatmap representation of ATE in meters.



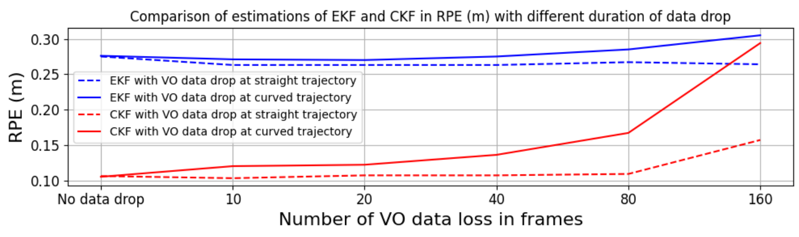
(b) Line graph representation of ATE in meters.

Figure 18. The Absolute Trajectory Error [m] for the EKF and CKF estimation results during the VO data drop experiment.

In terms of the Relative Pose Error (RPE), which focuses on the local consistency of the two given trajectories, most of the error slightly increases as the duration of data drop increases in both curved and straight trajectory. However, even with 160 frames of VO data loss in a straight trajectory, for example, the RPE for translation increased by less than 20cm, which can be considered negligible, as shown in Figure 19. Furthermore, Figure 20 indicates that the RPE for the estimated angle (in degrees) remains intact, suggesting that only translational errors occur, regardless of where the VO data drop occurs. As discussed in [52], data drop in the curved trajectory, which is located relatively earlier in the sequence than the straight trajectory, leads to misalignment between the estimated trajectory and the ground truth, resulting in a significant increase in ATE.

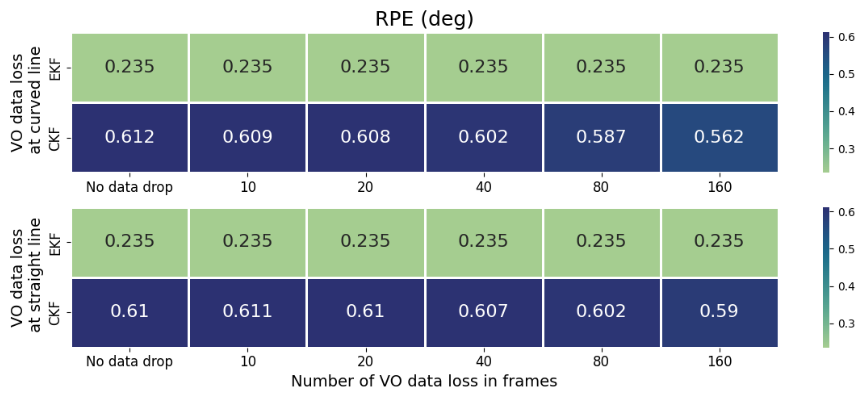


(a) Heatmap representation of RPE in meters.

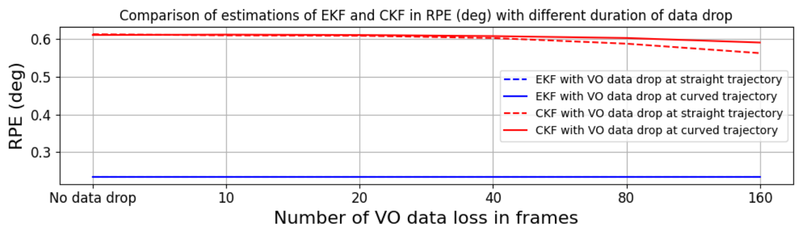


(b) Line graph representation of RPE in meters.

Figure 19. The Relative Pose Error [m] for the EKF and CKF estimation results during the VO data drop experiment.



(a) Heatmap representation of RPE in degree.



(b) Line graph representation of RPE in degree.

Figure 20. Heatmap representing the Relative Pose Error [m] for the EKF and CKF estimation results during the VO data drop experiment.

In conclusion of this section, in terms of local consistency, both EKF and CKF with velocity motion model is capable of estimating relative motion of the vehicle even if it encounters measurement data drop for 160 frames (16 seconds). However, in terms of global trajectory alignment, which is typically evaluated by ATE, an estimation of vehicle position is challenging especially a trajectory involves significant amount of data loss in a trajectory. Moreover, since CKF suffers velocity estimation issue, which leads to large translation error, semantic limitations such as configuring upper bound for velocity could be introduced in the application of grounded-vehicle localization.

5.3 Result of application of filter in real-life scenario

This section presents an analysis of the results obtained by applying EKF and CKF in a real-life scenario where GPS and VO data are intentionally dropped at various points along the trajectory. As shown in Figure 21, the GPS and VO inputs are intermittently unavailable, while IMU data remains available throughout the 1591 frames of the entire trajectory.

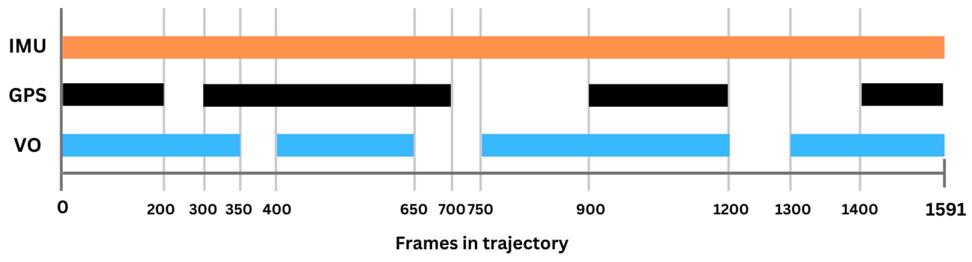
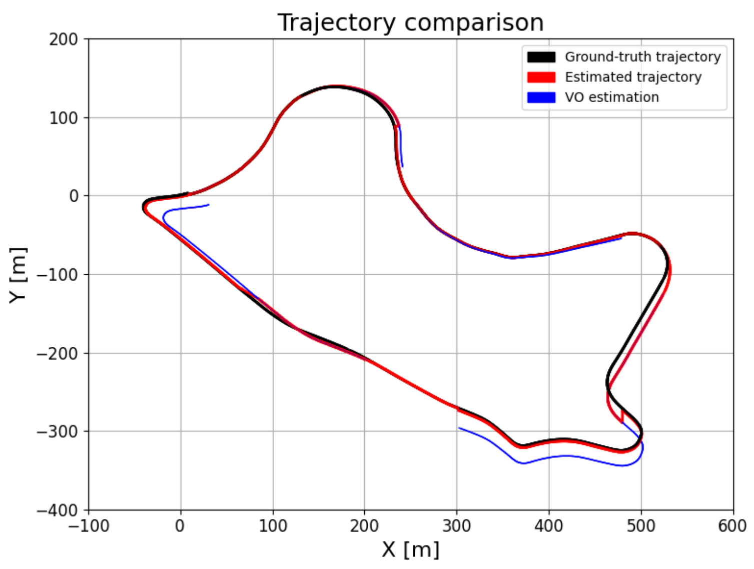


Figure 21. Visualization of data loss frames for each sensor in the trajectory.

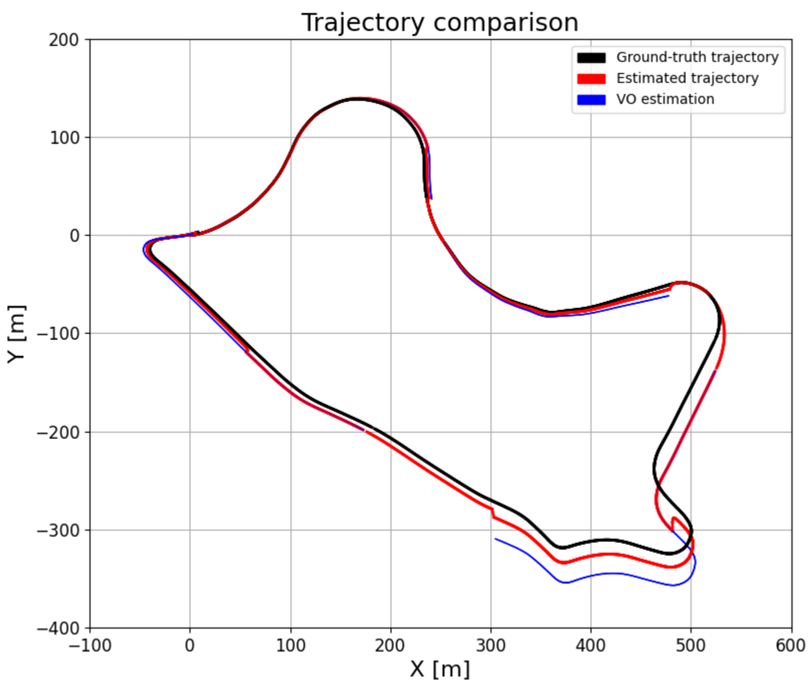
Figure 22 shows the estimated trajectory provided by EKF and CKF in real-life scenario.

In Figure 22, both GPS and VO data are provided to the filters, with the sensor data exhibiting lower measurement noise—specifically GPS—playing a dominant role in influencing the filter. As a result, the estimated trajectory in this segment aligns more closely with the GPS measurements. Between frames 200 and 300, where $x \in [120, 220]$ and $y \in [95, 150]$, GPS data becomes unavailable, prompting the filter to rely on VO estimations. Once GPS data is restored, the filter's estimation is corrected to align with the GPS measurements. During the VO data loss occurring between frames 350 and 400, within the region $x \in [230, 255]$ and $y \in [-10, 35]$, the filter continues to refine its estimations using GPS measurements.

At frame 650, VO loses tracking, and from frame 700 onward, GPS signals are also lost. Consequently, the filter operates in IMU-only dead-reckoning mode for 5 seconds, estimating the vehicle position within the coordinate range $x \in [520, 550]$ and $y \in [-105, -55]$. Five seconds later, a sufficient number of feature points become available, allowing the VO process to restart. The filter provides the current pose estimation to VO, which resumes trajectory estimation using stereo sequences, even as GPS remains unavailable. However, due to translational errors accumulated during the IMU-only phase,



(a) Trajectory estimation by EKF.



(b) Trajectory estimation by CKF.

Figure 22. Estimated trajectory by (a) EKF and (b) CKF in real-life scenario.

VO inherits the erroneous pose, resulting in an estimated trajectory offset from the ground truth. Once GPS signals are restored at frame 900, the filter corrects its estimations based on the GPS measurements.

From frame 1200, at the coordinate $x = 300$ and $y = -260$, both GPS and VO lose measurement data simultaneously for 10 seconds, likely due to an environment such as a tunnel. During this period, the filter relies exclusively on IMU data for position estimation.

At frame 1300, the VO process resumes with some translational errors, and the filter's estimation is subsequently corrected once GPS measurements become available at frame 1400. The process concludes with the system returning to its original position.

When comparing the estimations provided by EKF and CKF, the EKF tends to rely heavily on GPS data when it is available, resulting in better performance in terms of the metrics discussed in previous experiments. In contrast, the CKF balances the influence of GPS and VO data when both are available by considering their respective measurement noise levels. This balance allows the CKF to provide more robust estimations and introduces capabilities such as dynamic noise management, which surpass those of the EKF. Hence, in practical application, CKF is an optimal filter algorithm to solve localization problem of grounded-vehicle.

6. Conclusions and Future work

In this paper, we presented and analyzed five variants of the Kalman filter applied to the KITTI dataset across different experimental setups. Each filter was evaluated using three error metrics to thoroughly understand its characteristics.

The Particle Filter (PF) demonstrated suboptimal performance across all experiments, particularly due to its limitations in inference time. It requires a large number of particles to prevent filter divergence and accurately capture the system's motion, making it computationally expensive.

The Ensemble Kalman Filter (EnKF) also requires a moderate ensemble size to effectively represent the variance of the estimation, which results in a computational disadvantage compared to the EKF, UKF, and CKF. While the EnKF generally performs better than the PF, its accuracy is still inferior to these three filters. Specifically, the ATE for the EnKF highlights the significant impact of ensemble initialization, where misaligned rotations at the initial state propagate throughout the trajectory, leading to large ATE values. Proper initialization of the ensembles is therefore critical, balancing the variance needed to represent the system's dynamics and alignment with the actual system state.

The Unscented Kalman Filter (UKF) provides near-optimal estimation results across all three error metrics, achieving accuracy levels comparable to the CKF. The UKF, like the CKF, demonstrates the ability to balance sensor inputs based on the ratio of measurement noise, enabling robust trajectory estimation and dynamic noise tuning. However, when measurement data is unavailable, the UKF's state estimation deteriorates, leading to reduced accuracy.

The Extended Kalman Filter (EKF) performs well in terms of translation accuracy, as shown by the ATE and RPE for relative translation across all sequences in the KITTI dataset. Additionally, the EKF offers superior velocity estimation compared to the CKF during IMU-only dead-reckoning, keeping translation errors minimal. A key limitation of the EKF, however, is its tendency to allow a sensor with smaller measurement noise to dominate the state estimation, making it challenging to properly balance noise contributions from multiple sensors.

Finally, the Cubature Kalman Filter (CKF) achieves optimal accuracy across all metrics and is capable of maintaining trajectory estimates during short durations of sensor data loss (approximately up to 4 seconds or 40 frames in the experiment). Beyond this point, translation errors grow exponentially, making accurate localization increasingly difficult. Despite this limitation, the CKF excels in practical multi-sensor environments by effectively balancing sensor noise to estimate the system's state. The estimated trajectory reflects the noise ratios of the provided sensors, offering the ability to dynamically adjust noise parameters according to real-world conditions, such as GPS jamming or environments with limited visual features for VO. Consequently, the CKF provides high accuracy in most experimental scenarios, an intuitive understanding of estimation behavior, the ability to tune estimations dynamically, and ease of implementation, as it requires no additional parameters to set up.

In conclusion, both EKF and CKF demonstrate excellent localization performance for grounded vehicles in the KITTI dataset experiments. The EKF excels in maintaining low errors during IMU-only dead reckoning and outperforms other fusion algorithms based on the provided error metrics in this kind of scenario. On the other hand, the CKF delivers robust estimation, particularly in multi-sensor environments, and its ability to fine-tune sensor noise parameters enhances its adaptability to various real-world scenarios.

6.1 Future work

For future work, the following implementations and applications are proposed:

Motion Models: Alternative state equations to describe the system's dynamics, such as the Runge–Kutta method, which provides a fourth-order approximation of nonlinear systems (commonly referred to as *RK4*), can be implemented. This would allow a comparison of estimation results between the proposed motion models and higher-order approximation methods.

Parameter Tuning for Filters: The proposed filters, including the UKF, PF, and EnKF, require parameter configurations, such as the $\alpha, \beta, and \kappa$ values for the UKF, the sample size ($N_{sample\ size}$) and resampling algorithm for the PF, and the ensemble size ($N_{ensemble\ size}$) for the EnKF. Proper tuning of these parameters could enhance the filters' ability to capture the system's uncertainty and non-linearity, potentially leading to more accurate state estimations.

Noise Parameter Tuning: In the current experiments, noise parameters were configured uniformly across all filters. Dynamic noise parameter tuning during the filtering process could improve estimation accuracy and enable better adaptation to challenging application environments. Such tuning would enhance the sensor fusion capabilities of the filters, especially under varying noise conditions.

Additional Filters: Extended versions of the proposed filter algorithms, such as the Rao-Blackwellized Particle Filter, Multi-State Constraint Kalman Filter, and Square-Root Cubature Kalman Filter, could be explored. These advanced filters may offer improved state estimation and potentially outperform the filters presented in this study.

Application to New Environments: The strengths and limitations of each filter identified in the current experiments suggest their applicability to a range of new environments and datasets. Testing the proposed filters on diverse datasets could help evaluate their robustness and suitability for practical applications. Examples of such datasets include ground vehicle datasets like the Rosario agricultural dataset [54], UAV datasets such as the EuRoC dataset [55] and the Blackbird UAV dataset [26], and indoor datasets like the TUM LSI Dataset [56].

These proposed directions for future work have the potential to enhance filter performance and expand their applicability to more challenging and diverse scenarios.

References

[1] Syed Agha Hassnain Mohsan et al. "Unmanned aerial vehicles (UAVs): practical aspects, applications, open challenges, security issues, and future trends". In: *Intelligent Service Robotics* (Jan. 2023). ISSN: 1861-2784. DOI: 10.1007/s11370-022-00452-4. URL: http://dx.doi.org/10.1007/s11370-022-00452-4.

[2] Rudolph Emil Kalman. "A New Approach to Linear Filtering and Prediction Problems". In: *Transactions of the ASME–Journal of Basic Engineering* 82.Series D (1960), pp. 35–45.

[3] Bowen Wang et al. "Kalman Filter and Its Application in Data Assimilation". In: *Atmosphere* 14.8 (Aug. 2023), p. 1319. ISSN: 2073-4433. DOI: 10.3390/atmos14081319. URL: http://dx.doi.org/10.3390/atmos14081319.

[4] D.M. Helmick et al. "Path following using visual odometry for a Mars rover in high-slip environments". In: *2004 IEEE Aerospace Conference Proceedings (IEEE Cat. No.04TH8720)*. Vol. 2. 2004, 772–789 Vol.2. DOI: 10.1109/AERO.2004.1367679.

[5] Thomas Moore and Daniel W. Stouch. "A Generalized Extended Kalman Filter Implementation for the Robot Operating System". In: *Annual Meeting of the IEEE Industry Applications Society*. 2014. URL: https://api.semanticscholar.org/CorpusID:26564553.

[6] Pengfei Ji, Zhongxing Duan, and Weisheng Xu. "A Combined UWB/IMU Localization Method with Improved CKF". In: *Sensors* 24.10 (May 2024), p. 3165. ISSN: 1424-8220. DOI: 10.3390/s24103165. URL: http://dx.doi.org/10.3390/s24103165.

[7] Madan H T et al. "An Error-State Extended Kalman Filter Based State Estimation and Localization Algorithm for Autonomous Systems". In: *2023 International Conference on Smart Systems for applications in Electrical Sciences (ICSSES)*. 2023, pp. 1–6. DOI: 10.1109/ICSSES58299.2023.10199535.

[8] Shunsuke KAMIJO, Yanlei GU, and Li-Ta HSU. "Autonomous Vehicle Technologies:Localization and Mapping". In: *IEICE ESS Fundamentals Review* 9.2 (2015), pp. 131–141. ISSN: 1882-0875. DOI: 10.1587/essfr.9.2_131. URL: http://dx.doi.org/10.1587/essfr.9.2_131.

[9] Gaëtan Chevrin et al. “Visual-Inertial Fusion on KITTI using MSF-EKF”. In: *2022 17th International Conference on Control, Automation, Robotics and Vision (ICARCV)*. 2022, pp. 35–40. DOI: 10.1109/ICARCV57592.2022.10004344.

[10] Lovro Marković et al. “Error State Extended Kalman Filter Multi-Sensor Fusion for Unmanned Aerial Vehicle Localization in GPS and Magnetometer Denied Indoor Environments”. In: *2022 International Conference on Unmanned Aircraft Systems (ICUAS)*. 2022, pp. 184–190. DOI: 10.1109/ICUAS54217.2022.9836124.

[11] Qingxi Zeng et al. “A UAV Localization System Based on Double UWB Tags and IMU for Landing Platform”. In: *IEEE Sensors Journal* 23.9 (2023), pp. 10100–10108. DOI: 10.1109/JSEN.2023.3260311.

[12] Joong-hee Han et al. “Performance Evaluation of Autonomous Driving Control Algorithm for a Crawler-Type Agricultural Vehicle Based on Low-Cost Multi-Sensor Fusion Positioning”. In: *Applied Sciences* 10.13 (July 2020), p. 4667. ISSN: 2076-3417. DOI: 10.3390/app10134667. URL: http://dx.doi.org/10.3390/app10134667.

[13] Meibo Lv et al. “A Loosely Coupled Extended Kalman Filter Algorithm for Agricultural Scene-Based Multi-Sensor Fusion”. In: *Frontiers in Plant Science* 13 (Apr. 2022). ISSN: 1664-462X. DOI: 10.3389/fpls.2022.849260. URL: http://dx.doi.org/10.3389/fpls.2022.849260.

[14] Anastasios I. Mourikis and Stergios I. Roumeliotis. “A Multi-State Constraint Kalman Filter for Vision-aided Inertial Navigation”. In: *Proceedings 2007 IEEE International Conference on Robotics and Automation*. 2007, pp. 3565–3572. DOI: 10.1109/ROBOT.2007.364024.

[15] David Simón Colomar, John-Olof Nilsson, and Peter Händel. “Smoothing for ZUPT-aided INSs”. In: *2012 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. 2012, pp. 1–5. DOI: 10.1109/IPIN.2012.6418869.

[16] “Loosely Coupled GNSS and IMU Integration for Accurate i-Boat Horizontal Navigation”. In: *International Journal of Geoinformatics* (2022). URL: https://api.semanticscholar.org/CorpusID:249972339.

[17] Nader Nagui et al. “Improved GPS/IMU Loosely Coupled Integration Scheme Using Two Kalman Filter-based Cascaded Stages”. In: *Arabian Journal for Science and Engineering* 46.2 (Dec. 2020), pp. 1345–1367. ISSN: 2191-4281. DOI: 10.1007/s13369-020-05144-8. URL: http://dx.doi.org/10.1007/s13369-020-05144-8.

[18] Alireza Shaghaghian and Paknoosh Karimaghaee. “Improving GPS/INS Integration Using FIKF-Filtered Innovation Kalman Filter”. In: *Asian Journal of Control* 21.4 (Oct. 2018), pp. 1671–1680. ISSN: 1934-6093. DOI: 10.1002/asjc.1931. URL: http://dx.doi.org/10.1002/asjc.1931.

[19] Gianluca Falco, Marco Pini, and Gianluca Marucco. “Loose and Tight GNSS/INS Integrations: Comparison of Performance Assessed in Real Urban Scenarios”. In: *Sensors* 17.2 (Jan. 2017), p. 255. ISSN: 1424-8220. DOI: 10.3390/s17020255. URL: http://dx.doi.org/10.3390/s17020255.

[20] Adriano Solimeno. “Low-Cost INS/GPS Data Fusion with Extended Kalman Filter for Airborne Applications”. In: 2007. URL: https://api.semanticscholar.org/CorpusID:220644469.

[21] Zaher M. Kassas et al. “I Am Not Afraid of the GPS Jammer: Resilient Navigation Via Signals of Opportunity in GPS-Denied Environments”. In: *IEEE Aerospace and Electronic Systems Magazine* 37.7 (2022), pp. 4–19. DOI: 10.1109/MAES.2022.3154110.

[22] Qi-guo Yao, Yuxiang Su, and Lili Li. “Application of Square-Root Unscented Kalman Filter Smoothing Algorithm in Tracking Underwater Target”. In: 2018. URL: https://api.semanticscholar.org/CorpusID:96426255.

[23] Haoyu Zhou et al. “Rao-Blackwellised particle filtering for low-cost encoder/INS/GNSS integrated vehicle navigation with wheel slipping”. In: *IET Radar, Sonar Navigation* 13.11 (Nov. 2019), pp. 1890–1898. ISSN: 1751-8792. DOI: 10.1049/iet-rsn.2019.0108. URL: http://dx.doi.org/10.1049/iet-rsn.2019.0108.

[24] Martin Brossard, Axel Barrau, and Silvère Bonnabel. *AI-IMU Dead-Reckoning*. 2019. DOI: 10.48550/ARXIV.1904.06064. URL: https://arxiv.org/abs/1904.06064.

[25] Giovanni Cioffi et al. “Learned Inertial Odometry for Autonomous Drone Racing”. In: (2022). DOI: 10.48550/ARXIV.2210.15287. URL: https://arxiv.org/abs/2210.15287.

[26] Amado Antonini et al. “The Blackbird Dataset: A large-scale dataset for UAV perception in aggressive flight”. In: *2018 International Symposium on Experimental Robotics (ISER)*. 2018. DOI: 10.1007/978-3-030-33950-0_12. URL: https://doi.org/10.1007/978-3-030-33950-0_12.

[27] Dongchan Kim et al. “An Integrated Deep Ensemble-Unscented Kalman Filter for Sideslip Angle Estimation With Sensor Filtering Network”. In: *IEEE Access* 9 (2021), pp. 149681–149689. DOI: 10.1109/ACCESS.2021.3125351.

[28] Andreas Geiger, Philip Lenz, and Raquel Urtasun. "Are we ready for autonomous driving? The KITTI vision benchmark suite". In: *2012 IEEE Conference on Computer Vision and Pattern Recognition*. 2012, pp. 3354–3361. DOI: `10.1109/CVPR.2012.6248074`.

[29] Andreas Geiger et al. "Vision meets robotics: the KITTI dataset". In: *The International Journal of Robotics Research* 32 (Sept. 2013), pp. 1231–1237. DOI: `10.1177/0278364913491297`.

[30] Davide Scaramuzza and Friedrich Fraundorfer. "Visual Odometry Part I: The First 30 Years and Fundamentals". In: URL: `https://api.semanticscholar.org/CorpusID:207872004`.

[31] Mohammad O. A. Aqel et al. "Review of visual odometry: types, approaches, challenges, and applications". In: *SpringerPlus* 5 (2016). URL: `https://api.semanticscholar.org/CorpusID:17005811`.

[32] David G. Lowe. "Distinctive Image Features from Scale-Invariant Keypoints". In: *International Journal of Computer Vision* 60.2 (Nov. 2004), pp. 91–110. ISSN: 0920-5691. DOI: `10.1023/b:visi.0000029664.99615.94`. URL: `http://dx.doi.org/10.1023/B:VISI.0000029664.99615.94`.

[33] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. "SURF: Speeded Up Robust Features". In: *Computer Vision – ECCV 2006*. Springer Berlin Heidelberg, 2006, pp. 404–417. ISBN: 9783540338338. DOI: `10.1007/11744023_32`. URL: `http://dx.doi.org/10.1007/11744023_32`.

[34] Ethan Rublee et al. "ORB: An efficient alternative to SIFT or SURF". In: *2011 International Conference on Computer Vision*. 2011, pp. 2564–2571. DOI: `10.1109/ICCV.2011.6126544`.

[35] Friedrich Fraundorfer and Davide Scaramuzza. "Visual Odometry : Part II: Matching, Robustness, Optimization, and Applications". In: *IEEE Robotics Automation Magazine* 19.2 (2012), pp. 78–90. DOI: `10.1109/MRA.2012.2182810`.

[36] Mohamed Nazrul Mohamed Nazeer. *Development of a visual interstitial-based real-time location detection system for use in a GNSS-free environment Development of Visual-Inertial Odometry Based Real Time Localization System for GNSS Denied Environment*. [Protected: 03.06.2024].

[37] Martin A. Fischler and Robert C. Bolles. "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography". In: *Commun. ACM* 24 (1981), pp. 381–395. URL: `https://api.semanticscholar.org/CorpusID:972888`.

[38] Prabin Kumar Panigrahi and Sukant Kishoro Bisoy. “Localization strategies for autonomous mobile robots: A review”. In: *Journal of King Saud University - Computer and Information Sciences* 34.8, Part B (2022), pp. 6019–6039. ISSN: 1319-1578. DOI: https://doi.org/10.1016/j.jksuci.2021.02.015. URL: https://www.sciencedirect.com/science/article/pii/S1319157821000550.

[39] M Homelius. “Tracking of ground vehicles, Evaluation of Tracking Performance Using Different Sensors and Filtering Techniques”. MA thesis. Linköping University, 2018.

[40] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. Cambridge, Mass.: MIT Press, 2005. ISBN: 0262201623 9780262201629. URL: http://www.amazon.de/gp/product/0262201623/102-8479661-9831324?v=glance&n=283155&n=507846&s=books&v=glance.

[41] 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.

[42] Jos Elfring, Elena Torta, and René van de Molengraft. “Particle filters: A hands-on tutorial”. en. In: *Sensors (Basel)* 21.2 (Jan. 2021), p. 438.

[43] Tiancheng Li, Miodrag Bolic, and Petar M. Djuric. “Resampling Methods for Particle Filtering: Classification, implementation, and strategies”. In: *IEEE Signal Processing Magazine* 32.3 (2015), pp. 70–86. DOI: 10.1109/MSP.2014.2330626.

[44] Tiancheng Li et al. “Fighting Sample Degeneracy and Impoverishment in Particle Filters: A Review of Intelligent Approaches”. In: (2013). DOI: 10.48550/ARXIV.1308.2443. URL: https://arxiv.org/abs/1308.2443.

[45] Fredrik Gustafsson. “Particle filter theory and practice with positioning applications”. In: *IEEE Aerospace and Electronic Systems Magazine* 25.7 (2010), pp. 53–82. DOI: 10.1109/MAES.2010.5546308.

[46] Geir Evensen. “Sequential data assimilation with a nonlinear quasi-geostrophic model using Monte Carlo methods to forecast error statistics”. en. In: *J. Geophys. Res.* 99.C5 (May 1994), pp. 10143–10162.

[47] R.N. Bannister. *A Square-Root Ensemble Kalman Filter Demonstration with the Lorenz model*. Accessed: 03-04-2024.

[48] Matthias Katzfuss, Jonathan R Stroud, and Christopher K Wikle. “Understanding the ensemble Kalman filter”. en. In: *Am. Stat.* 70.4 (Oct. 2016), pp. 350–357.

[49] Alison Fowler. *The Ensemble Kalman filter*. https://www.met.reading.ac.uk/~darc/training/ecmwf_collaborative_training/2020/EnKF_AFowler.pdf. Mar. 2019.

[50] 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.

[51] Huangying Zhan et al. “Visual Odometry Revisited: What Should Be Learnt?” In: *arXiv preprint arXiv:1909.09803* (2019).

[52] Zichao Zhang and Davide Scaramuzza. “A Tutorial on Quantitative Trajectory Evaluation for Visual(-Inertial) Odometry”. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2018, pp. 7244–7251. DOI: 10.1109/IROS.2018.8593941.

[53] Y.K. Thong et al. “Numerical double integration of acceleration measurements in noise”. In: *Measurement* 36.1 (July 2004), pp. 73–92. ISSN: 0263-2241. DOI: 10.1016/j.measurement.2004.04.005. URL: http://dx.doi.org/10.1016/j.measurement.2004.04.005.

[54] Taihú Pire et al. “The Rosario dataset: Multisensor data for localization and mapping in agricultural environments”. In: *The International Journal of Robotics Research* 38.6 (2019), pp. 633–641. DOI: 10.1177/0278364919841437.

[55] Michael Burri et al. “The EuRoC micro aerial vehicle datasets”. In: *The International Journal of Robotics Research* 35.10 (Jan. 2016), pp. 1157–1163. ISSN: 1741-3176. DOI: 10.1177/0278364915620033. URL: http://dx.doi.org/10.1177/0278364915620033.

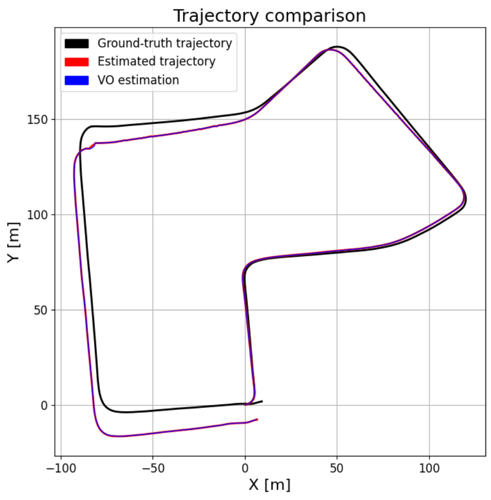
[56] Florian Walch et al. “Image-based localization using LSTMs for structured feature correlation”. In: *IEEE International Conference on Computer Vision (ICCV)*. Oct. 2017. URL: http://arxiv.org/abs/1611.07890.

Appendices

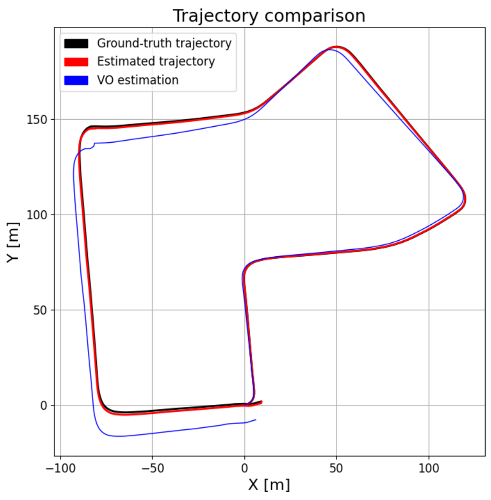
Appendix 1 – Non-Exclusive License for Reproduction and Publication of a Graduation Thesis[1]

[1]The non-exclusive licence is not valid during the validity of access restriction indicated in the student’s application for restriction on access to the graduation thesis that has been signed by the school’s dean, except in case of the university’s right to reproduce the thesis for preservation purposes only. If a graduation thesis is based on the joint creative activity of two or more persons and the co-author(s) has/have not granted, by the set deadline, the student defending his/her graduation thesis consent to reproduce and publish the graduation thesis in compliance with clauses 1.1 and 1.2 of the non-exclusive licence, the non-exclusive license shall not be valid for the period.

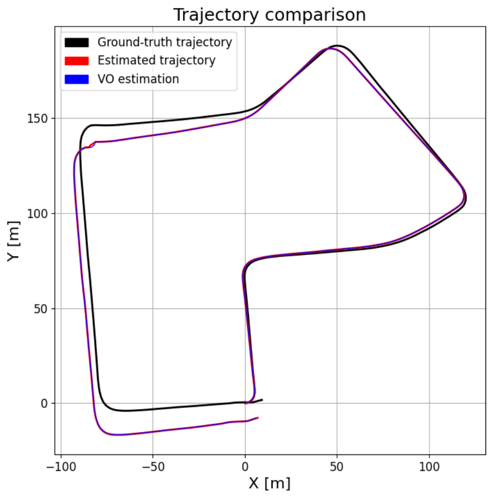
Appendix 2 – Performance comparisons of application of filters in mono- and multi-sensor environment.



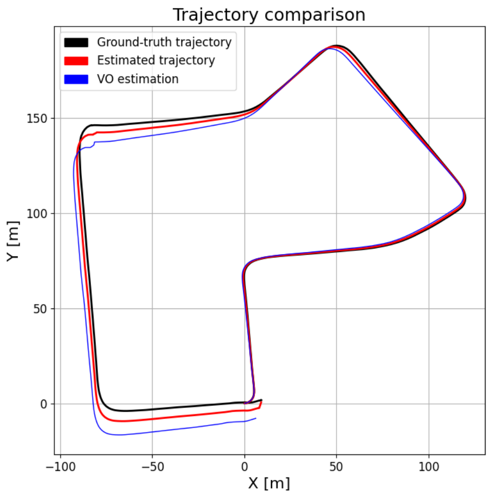
(a) EKF without GPS input data.



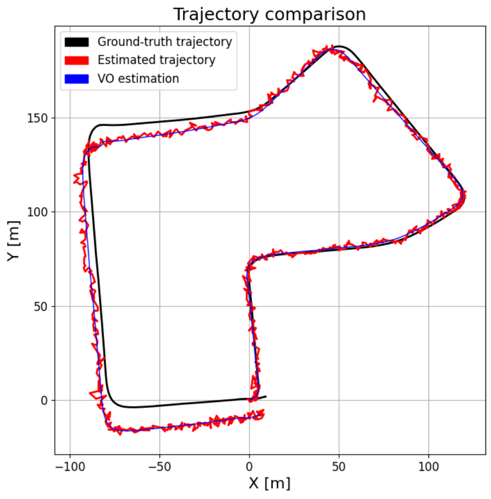
(b) EKF with GPS input data.



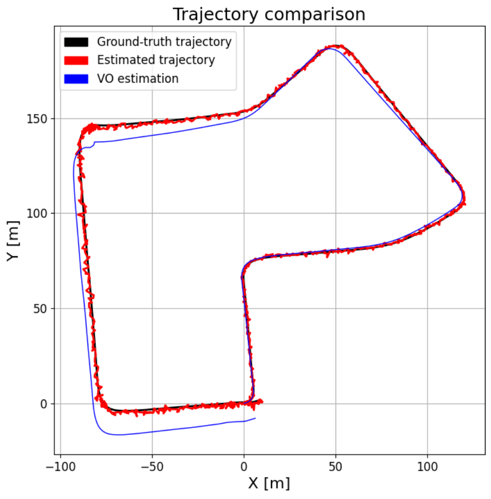
(c) UKF without GPS input data.



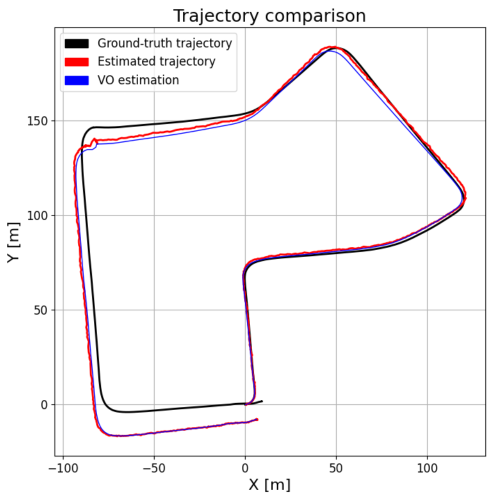
(d) UKF with GPS input data.



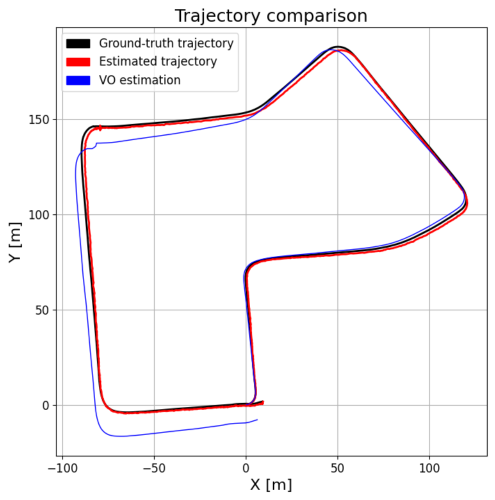
(e) PF without GPS input data.



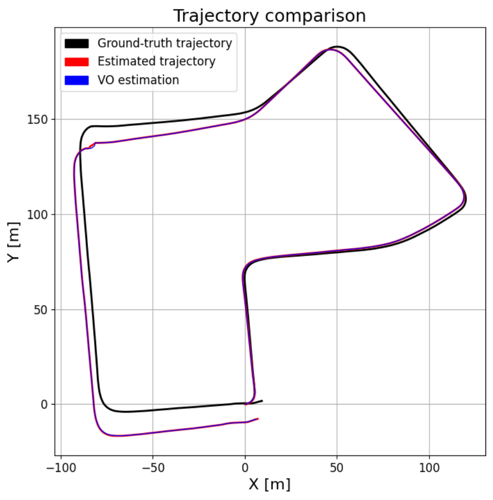
(f) PF with GPS input data.



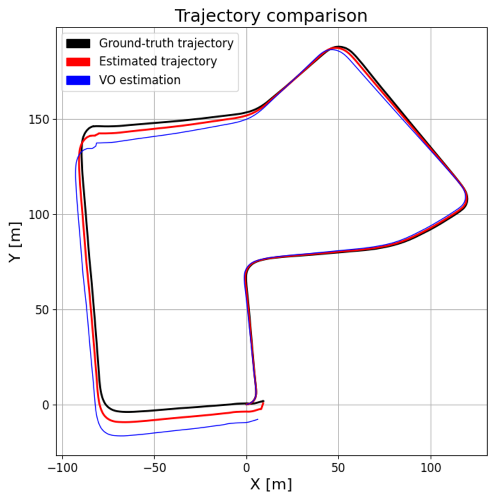
(g) EnKF without GPS input data.



(h) EnKF with GPS input data.



(i) CKF without GPS input data.



(j) CKF with GPS input data.

Figure 23. Comparisons of estimated trajectories for five filters (EKF, UKF, PF, EnKF, CKF) using velocity motion model, with and without GPS data for sequence 07 of the KITTI dataset. The first column shows the estimations without GPS, while the second columns illustrates the estimations with GPS.