# PRECISE VEHICLE LOCALIZATION USING FUSION OF MULTIPLE SENSORS FOR SELF-DRIVING

Undergraduate graduation project report submitted in partial fulfillment of
the requirements for the
Degree of Bachelor of Science of Engineering
in

The Department of Electronic & Telecommunication Engineering
University of Moratuwa.

Supervisor:
Dr. Peshala Jayasekara

Group Members:
S. P. Dissanayake (160134U)
G. M. C. P. Jayathissa (160188L)
S. H. P. H. P. Jayawardana (160246N)
A. S. R. Kumara (160318M)

March, 2021

Approval of the Department of Electronic & Telecommunication Engineering

. . . . . .
Head, Department of Electronic &
Telecommunication Engineering

This is to certify that I/we have read this project and that in my/our opinion it is fully adequate, in scope and quality, as an Undergraduate Graduation Project.

Supervisor: Dr. Peshala Jayasekara

Signature: . . . . . .

Date: . . . . . .

# DECLARATION

This declaration is made on March 21, 2021.

**Declaration by Project Group**

We declare that the dissertation entitled "Precise Vehicle Localization Using Fusion of Multiple Sensors for Self-Driving" and the work presented in it are our own. We confirm that:

- this work was done wholly or mainly in candidature for a B.Sc. Engineering degree at this university,
- where any part of this dissertation has previously been submitted for a degree or any other qualification at this university or any other institute, has been clearly stated,
- where we have consulted the published work of others, is always clearly attributed,
- where we have quoted from the work of others, the source is always given,
- with the exception of such quotations, this dissertation is entirely our own work,
- we have acknowledged all main sources of help,

. . . . . .
Date

. . . . . .
S. P. Dissanayake (160134U)

. . . . . .
G. M. C. P. Jayathissa (160188L)

. . . . . .
S. H. P. H. P. Jayawardana (160246N)

. . . . . .
A. S. R. Kumara (160318M)

# DECLARATION BY SUPERVISOR

I have supervised and accepted this dissertation for the submission of the degree.

..... .....

Dr. Peshala Jayasekara Date

# ABSTRACT


## PRECISE VEHICLE LOCALIZATION USING FUSION OF MULTIPLE SENSORS FOR SELF-DRIVING

Group Members: S. P. Dissanayake, G. M. C. P. Jayathissa, S. H. P. H. P. Jayawardana, A. S. R. Kumara

Supervisor: Dr. Peshala Jayasekara

Keywords: Self-Driving, State Estimation, Localization, Sensor Fusion, Bayesian Filters.

This project focuses on creating a mechanism for estimating the state of a self-driving vehicle, including its location, speed and orientation, relative to a coordinate frame fixed to earth. We expect to achieve this using data from sensors such as Inertial Measurement Unit (IMU), Global Navigation Satellite System (GNSS) receivers, stereo camera pairs and Light Detection and Ranging (LiDAR) sensors. The main objective is to deliver a well-documented software stack which includes the state estimator running on Robot Operating System (ROS). The estimator should be capable of providing uninterrupted state estimations with enough accuracy and frequency to facilitate self-driving.

The main drawback observed in current state-of-the-art work is, the dependency of the solution on pre-generated highly-detailed maps of different forms, which in-turn reduces the scalability of the solution. This dependency reduces the feasibility of those solutions in the long run due to the fact that it is hard to maintain such highly-detailed maps in midst of constantly and unexpectedly changing environments, prevailing in countries such as Sri Lanka. It is the intention of this project to mitigate this dependency through means of improving the state estimation algorithm. We also intend to implement the solution in a modularized architecture to facilitate easy modifications, which in-turn will allow the solution to be used in different applications.

While self-driving is itself a novel concept in Sri Lankan context, this project aims to facilitate the state estimation under constrained resource availability (such as excluding highly-detailed maps, enhanced GNSS technologies such as Differential Global Positioning System (DGPS) or Real Time Kinematics (RTK) Global Positioning System (GPS), reliable road features such as consistent lane markings and curbs etc.), which is the condition experienced in countries like Sri Lanka.

Other than the self-driving research communities, we expect the outcome of this project will benefit different parties such as robot developers and navigational solution providers, who have similar requirements.

# DEDICATION

TODO.

# ACKNOWLEDGEMENTS

TODO.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# INTRODUCTION

## 1.1 Problem definition and scope

The amount of autonomy in vehicles is divided into six levels and vehicles beyond level three are considered as self-driving. Such vehicles are expected to travel from a starting point to a given destination, with minimal human-driver involvement. Therefore, they need to know their location with a very high accuracy, relative to their immediate environment as well as in a global level. Other than the location itself, it is important to provide details of other state variables of the vehicle such as the speed and the orientation, which will be used by top-level controlling algorithms. This information should be updated uninterruptedly and frequently to preserve accuracy under higher speeds, which is the job of a localization module. The most widely used mechanism for fulfilling this requirement is, fusing data obtained from different sensors such as Global Navigation Satellite System (GNSS), Light Detection and Ranging (LiDAR), Radio Detection and Ranging (RADAR) and cameras to obtain the most probable state using a Bayesian filter. This is known as sensor fusion.

As we have noticed, a main limitation of the existing state-of-the-art work in this regard is the dependency of these solutions on different kinds of existing, accurate feature maps. These maps are used as inputs to the localization module, which reduces the scalability of the solution due to the fact that creating, updating and storing such highly-detailed maps of an entire region or a country is not so feasible. We also note the absence of a detailed work-flow describing complete implementation of a localization module, which in-turn wastes the time and effort of the research community, by having to start from the beginning, all the time.

Therefore, it is the aim of this project to implement a localization module which addresses problems mentioned above, while providing enough accuracy and update frequency to allow self-driving. Mentioning specifically, a sub-meter level positional accuracy is targeted along with a frequency of 70 Hz or more, which will be sufficient for speeds below 50 km $h^{-1}$[2]. As this work is a part of the top-level project aiming the construction of a fully autonomous vehicle, the system is implemented on Robot Operating System (ROS) to facilitate easy integration with other modules. The solution will initially be tuned and tested using freely available datasets and, eventually, it will be tested using actual sensors. The accuracy will only be evaluated using datasets, by comparing with the provided ground

truth data.

While self-driving is itself a novel concept in the Sri Lankan context, this project aims in resolving the dependency of state-of-the-art work on feature maps, thereby allowing accurate localization in unstructured, constantly changing environments. We intend to achieve this goal mainly through improving the data fusion algorithm. Other than the field of self-driving itself, we expect that this mechanism will be useful in different applications such as robotics navigation and navigational equipment development. The project is carried out with the collaboration of Creative Software Private Limited.

### 1.2 Related work

Localization using multi-sensor fusion is not a brand-new idea. A lot of researches have been done in this area for the past decade. The self-driving car concept was firstly addressed with Military European Land Robot Trial (M-ELROB) and Defence Advance Research Project Agency (DARPA) Urban Challenge competitions in 2006 and 2007[3]. The SmartTer [4] and Stanford Junior [5] are two self-driving car projects who won these competitions. Google's car [6], VisLab's Car [7], Apollo [8] and Autoware Auto [9] are some examples of successful research projects. However, self-driving cars are still in the introductory phase of the product life cycle.

Normally, GNSS, Inertial Measurement Unit (IMU), LiDAR, RADAR and wheel encoders are used to localize a robot. The combination of sensors depends on the design. Different sensors have different shortcomings. GNSS signal may not be available on a covered area, underground, or in a tunnel. Normally GNSS accuracy is about 10m due to satellite orbit and clock errors [10]. In addition, when the vehicle drives next to large structures, GNSS measurement can go wrong due to reflections [4]. GISA, which is a Brazilian platform for autonomous car trials uses Differential Global Positioning System (DGPS) as one of the sensors [3]. It gives better accuracy than normal GNSS. DGPS has also been used by the SmartTer. But standard GNSS is available more often when compared to DGPS because it does not rely on the visibility of geostationary satellites which provide the DGPS corrections [4]. The enhanced GNSS technique known as Real Time Kinematics (RTK) has been used by Wan et al. in their self-driving car for localization [10]. However, this method is also prone to significant errors caused by multipath effects and signal blockages due to its dependency on precision carrier-phase positioning techniques. It is noted that most of the projects use normal GNSS[5] [11][12][13]. LiDAR works well when the environment is full of 3D or texture features, but it fails in open spaces [10]. 3D LiDAR sensors were used by GIZA[3], Wan et al. [10] and Levinson et al. [14], while 2D LiDAR sensors were used by Soloviev [11], [15] and Baldwin and Newman [16]. Erik Ward and John Folkesson used

RADAR as the measurement model input as mentioned in [17]. Both LiDAR and RADAR sensors have the same kind of behavior when they act as measurement model inputs. IMU sensors have been used in almost every project. However, it suffers from the accumulation of integration errors [10]. Wheel encoders have been used along with IMU in [4], but it provides incorrect measurements when wheels slip. As mentioned above, each sensor has its own drawbacks and advantages. Other than these sensors Wan et al. [10], Levinson and Thrun [12], Stanford Junior [5], and Apollo [8] projects have used a pre-built map. Currently, some measurement companies have already begun to prepare map databases for self-driving vehicles [18].

In the works of Wan et al., they have used LiDAR intensity and altitude cues with 3D geometry for their LiDAR based localization module. They have obtained a grid-cell representation of the environment using a single Gaussian distribution to model the environment which involved both the intensity and the altitude. Finally, an Error-State Kalman filter was applied to fuse the data from the sensors. They have achieved 0.05-0.1 m Root Mean Square (RMS) accuracy in both longitudinal and lateral directions [10]. The Stanford Junior which won the second place of DARPA Urban Challenge competition was given a digital map of the road network in the form of a Route Network Definition File (RNDF). The RNDF contained geometric information about lane markings, stop signs, parking lots and special checkpoints. They were specified in GNSS coordinates. Local alignment between the RNDF and the vehicle's current position was estimated using GNSS along with two laser sensor measurements [5]. After the competition, they have upgraded the ground map using GNSS, IMU and Velodyne LiDAR data. Here, every cell was represented as its own Gaussian distribution. By this method, they have achieved a lateral RMS accuracy better than 0.1 m. Levinson and Thrun have localized the vehicle using a probabilistic map. They have modeled the environment as a probabilistic grid whereby every cell is represented as its own Gaussian distribution over remittance values. Furthermore, offline Simultaneous Localization and Mapping (SLAM) was used to align multiple passes of the same environment. Once a map had been built, they have used it to localize the vehicle in real time by representing the likelihood distribution of possible x and y offsets with a 2-dimensional histogram filter. The resulting error after localization has been extremely low, with an RMS value of 0.09 m [12]. A hybrid model with both Kalman filter and particle filter has been proposed by Won et al. in [19] as the sensor fusion algorithm. They have used the particle filter to estimate the orientation and the Kalman filter for estimating the position and velocity. As per the above discussion, it is clear that the most widely used and successful method for localization for self-driving is fusing data obtained from different sensors. These sensors should be selected carefully, so that they have complementary

properties and functioning capabilities under different environmental conditions. As an example, a GNSS receiver may function well in an open environment, in which a LiDAR is of very little use. Conversely, a LiDAR can give a very detailed output while in an urban environment, in which a GNSS receiver may fail due to multipath effects and signal blockage. We also note that in almost all the works, a Bayesian filter (Extended Kalman, Particle filter etc.) or a combination of many, has been used as the data fusion algorithm.

Another important deduction that can be made from the above comparison is that most of the state-of-the-art work depends on the assumption of an existing accurate map, which is given as an input to the localization mechanism. Even though this dependency is satisfiable under constrained environments such as competitions, it reduces the scalability of the solution drastically when it comes to unstructured environments, which a self-driving vehicle will be experiencing most frequently under normal operation. This is also the case for countries like Sri Lanka where highly detailed mapping of the entire country is a tedious task due to resource constrains and frequent and unplanned changings of the features. Hence, we find it is extremely important to focus on reaping the maximum localization accuracy possible, from a given combination of sensors.

### 1.3 Method of investigation and results

After a comprehensive study of literature pertaining to current developments, we decided to use a Bayesian framework of sensor fusion, along with the sensor data from (not limited to) GNSS receiver, IMU, LiDAR, stereo camera pair, magnetometer and wheel odometer. In-order to decide upon the type of the Bayesian filter to be used, several such filters with a minimal complexity were implemented, and accuracies were compared using data from existing datasets. As developing visual odometry (from stereo pair of cameras) and LiDAR odometry algorithms were out of the scope of the project, existing state-of-the-art algorithms were used with minor modifications. A filter framework with stochastic cloning and backward smoothing functionality was implemented in Python. Special attention was given to implement this framework in an easily modifiable manner (mainly, the motion model and state variables), so that, it facilitates experimenting with different filter structures. Output of the framework is compared with the ground truth provided in the datasets to obtain accuracy measures. As per the current state of the project, two main problems have been identified; mechanism to estimate an error covariance matrix for visual/LiDAR odometry algorithms and treating the correlated noise of the GNSS measurements. A detailed discussion of the method of investigation and the results obtained will be carried out in the subsequent chapters.

# Chapter 2

# METHODOLOGY

## 2.1 System architecture

As depicted in figure 2.1, we use the pose estimates calculated using stereo images and LiDAR point clouds as relative measurements. Oriented FAST and Rotated BRIEF Simultaneous Localization and Mapping 2 (ORB SLAM2) and LegoLOAM algorithms, respectively, are used for this purpose. Furthermore, GNSS measurements and orientation estimated from magnetometer measurements act as positional and rotational absolute measurements to the fusion mechanism. The fusion mechanism is an Error State Extended Kalman Filter (ES-EKF) with 16 variables in the state space and 15 variables in the error state space. The output of the system consists of a state vector including position, orientation and velocity of the vehicle relative to a global frame of reference. Estimated covariance of the error is also provided, from which, the 99% confidence interval ($3\sigma$ bound) can be derived. Output is compared with the ground truth provided with the dataset being used, to calculate the resultant error margins.

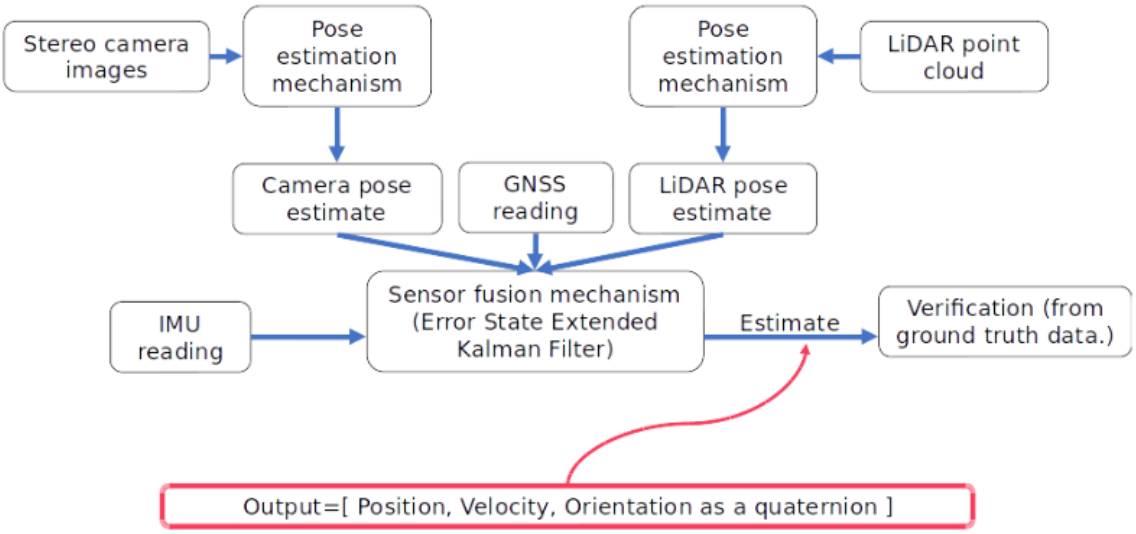



Figure 2.1: System block diagram

The system is implemented on Robot Operating System (ROS), along with evaluation and visualization mechanisms for demonstrating functionality. Main programming lan-

guage used is Python. Each of the components mentioned in the above discussion will be explained in detail, in the subsequent sections.

### 2.2 Coordinate frames

Apart from each sensor's own coordinate frame, in which they provide measurements, we define the following coordinate frames which will be used in the rest of this report.

| | |
|---|---|
| Inertial frame | An earth fixed right-handed rectilinear coordinate frame with x, y and z axes pointing towards East, North and Up directions respectively(ENU frame). The origin of the frame is determined by the information given in the dataset being used. |
| Body frame | A right-handed rectilinear coordinate frame fixed to the vehicle with x, y and z directions pointing lateral, front and upward directions of the vehicle respectively. |

Figure 2.2 illustrates the above-mentioned coordinate frames.

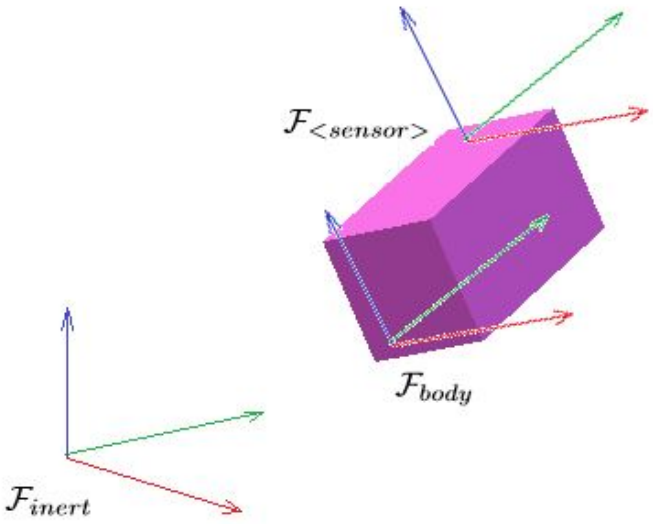



Figure 2.2: Illustration of the coordinate frames. x, y and z axes of each coordinate frame is depicted in red, green and blue colours, respectively. The cuboid represents the vehicle. The sensor is assumed to be mounted on the front side of the roof of the vehicle.

### 2.3 Datasets

We have being using three public datasets, namely, University of Michigan, North Campus Long-Term (NCLT) dataset[20], Karlsruhe Institute of Technology and Toyota Technolog-

ical Institute (KITTI) dataset for tracking[21] and Korean Advanced Institute of Science and Technology (KAIST) Urban dataset[22]. In addition, we expect to use the EU-Long Term dataset[23]. All these datasets include data from sensors that are sufficient for the sensor fusion mechanism, along with ground truth data. Details of the datasets are given in table 2.1.

| Dataset | Sensors included | Ground truth |
|---|---|---|
| 1 | 6 | 87837 |
| 2 | 7 | 78 |
| 3 | Banana | 778 |

Table 2.1: Dataset information

### 2.4 Comparison of Bayesian filters

Kalman Filters are the most widely used sensor data fusion algorithms. There are many extensions to the basic Kalman Filter (which is known as the Linear Kalman Filter), allowing them to be used in variety of applications. Table 2.2 provides a comparison between different types of Kalman filters. Note that the computational complexity of the filters increases from left to right.

| **Linear Kalman Filter** | **Extended Kalman Filter** | **Error State Extended Kalman Filter** | **Unscented Kalman Filter** |
|---|---|---|---|
| Linear Gaussian model | Non-linear Gaussian model | Non-linear Gaussian model | Non-linear Gaussian model |
| Best linear unbiased estimator (Blue) | Linearize the non-linear dynamics using Jacobeans (Taylor series) | Linearize the non-linear dynamics of error using Jacobeans (Taylor series) | Calculate sigma points and use Unscented Transform to approximate the Probability Density Function (PDF)s directly |
| Estimates the state | Estimates the state | Estimates the error of the state | Estimates the state |

Table 2.2: Comparison of different types of Kalman Filters

Kalman Filters can only be used for Gaussian models. However, we cannot use the Linear Kalman Filter at all because the localization task is not linear. Assuming that the localization problem is Gaussian, other Kalman Filters can be used to tackle the problem. We selected Unscented Kalman Filter (UKF) and Error State Extended Kalman Filter (ES-EKF) to implement in python since those two algorithms have shown the best results among Kalman filters according to [24],[25] and [26].

Particle Filter (PF) was recently suggested after Kalman Filter (KF)s to process arbitrary inertial sensor characteristics, motion dynamics, and noise distributions. When dealing with non-linear models in motion equations and measurement relations with a non-Gaussian noise assumption, the Kalman Filter methods may lead to non-optimal solutions. However, particle filters provide general solutions to many problems where linearization and Gaussian approximations are intractable [27]. A brief comparison of PFs and KFs is given in Table 2.3.

| **Kalman Filters** | **Particle Filters** |
|---|---|
| Only for Gaussian models | For both Gaussian and non-Gaussian models |
| Only for linear models | For both linear and non-linear models |
| Low computational complexity | High computational complexity |

Table 2.3: A brief comparison of Kalman Filters and Particle Filters

Three basic level filters (namely, PF, ES-EKF, and UKF) were implemented and compared using the NCLT data set to select the best filter for our application. Afterwards, the performance of the best filter was improved by increasing the number of elements in the state vector, adding Zero Velocity Update (ZUPT) measurements, and fusing relative measurements (visual and LiDAR odometry) etc. The state vector used for the basic filter implementations was

$$\mathbf{x} = \begin{bmatrix} \mathbf{p} \\ \mathbf{v} \\ \mathbf{q} \end{bmatrix} \tag{2.1}$$

where,

$$\mathbf{p} = (p_x, p_y, p_z) \text{ position relative to the inertial frame} \tag{2.2}$$

$$\mathbf{v} = (v_x, v_y, v_z) \text{ velocity relative to the inertial frame} \tag{2.3}$$

$$\mathbf{q} = (q_\omega, q_x, q_y, q_z) \text{ quaternion relative to the inertial frame.} \tag{2.4}$$

The ES-EKF estimates the error state directly and uses it as a correction to the nominal state. Number of particles used in the PF was 1000. Gaussian random noise was added to the particles before applying the motion model, to increase the diversity of the particles. Systematic Resampling was used to resample particles when needed and squared error function was used as the weight function. Following values were selected for the

parameters (as defined in [26]) of the UKF;

$$\alpha = 1 \text{ (controls the spread of sigma)} \tag{2.5}$$

$$\beta = 2 \text{ (related to the distribution)} \tag{2.6}$$

$$k = 1 \text{ (controls the spread of sigma).} \tag{2.7}$$

Based on the results obtained (see section 3.1), ES-EKF was selected as the best filter to tackle our problem.

## 2.5 Sensor fusion mechanism

### *2.5.1 The Error State Extended Kalman Filter*

The ES-EKF acts as the component responsible for fusing sensor data. As mentioned in section 2.1, our ES-EKF currently has 16 nominal state space variables and 15 error state space variables, grouped into 5 sub-vectors, as listed below;

$$\text{Nominal state vector: } \mathbf{x} = \begin{bmatrix} \mathbf{p} \\ \mathbf{v} \\ \mathbf{q} \\ \boldsymbol{a_b} \\ \boldsymbol{\omega_b} \end{bmatrix} \tag{2.8}$$

where

$$\begin{aligned} \mathbf{p} &= (p_x, p_y, p_z) \text{ position relative to the inertial frame} \\ \mathbf{v} &= (v_x, v_y, v_z) \text{ velocity relative to the inertial frame} \\ \mathbf{q} &= (q_w, q_x, q_y, q_z) \text{ quaternion relative to the inertial frame} \\ \boldsymbol{a_b} &= (a_{bx}, a_{by}, a_{bz}) \text{ acceleration biases of the IMU relative to the body frame} \\ \boldsymbol{\omega_b} &= (\omega_{bx}, \omega_{by}, \omega_{bz}) \text{ angular velocity biases of the IMU relative to the body frame} \end{aligned} \tag{2.9}$$

and

$$\text{Error state vector: } \boldsymbol{\delta}\mathbf{x} = \begin{bmatrix} \boldsymbol{\delta}\mathbf{p} \\ \boldsymbol{\delta}\mathbf{v} \\ \boldsymbol{\delta\theta} \\ \boldsymbol{\delta a_b} \\ \boldsymbol{\delta\omega_b} \end{bmatrix}. \tag{2.10}$$

Here, $\boldsymbol{\delta\theta}$ is the error in orientation, expressed as an axis-angle vector. Furthermore, $\boldsymbol{\delta a_b}$ and $\boldsymbol{\delta\omega_b}$ are the considered as global errors. The motion model for the prediction step of the filter is given below.
Nominal state update:

$$\check{\mathbf{p}}_k = \hat{\mathbf{p}}_{k-1} + \hat{\mathbf{v}}_{k-1}\Delta t + \frac{1}{2}\left(\mathbf{R}_{inert,body}\left(\mathbf{a}_{m_{k-1}} - \hat{\mathbf{a}}_{b_{k-1}}\right) + \mathbf{g}\right)\Delta t^2 \tag{2.11}$$

$$\check{\mathbf{v}}_k = \hat{\mathbf{v}}_{k-1} + \left(\mathbf{R}_{inert,body}\left(\mathbf{a}_{m_{k-1}} - \hat{\mathbf{a}}_{b_{k-1}}\right) + \mathbf{g}\right)\Delta t \tag{2.12}$$

$$\check{\mathbf{q}}_k = \hat{\mathbf{q}}_{k-1} \otimes \mathbf{q}\left\{\left(\boldsymbol{\omega}_{m_{k-1}} - \hat{\boldsymbol{\omega}}_{b_{k-1}}\right)\Delta t\right\} \tag{2.13}$$

$$\check{\mathbf{a}}_{b_k} = \hat{\mathbf{a}}_{b_{k-1}} \tag{2.14}$$

$$\check{\boldsymbol{\omega}}_{b_k} = \hat{\boldsymbol{\omega}}_{b_{k-1}} \tag{2.15}$$

with

$$\mathbf{a}_{m_k} = \text{Acceleration measured by accelerometer at k}^{\text{th}}\text{ instance} \tag{2.16}$$

$$\boldsymbol{\omega}_{m_k} = \text{Angular velocity measured by gyroscope at k}^{\text{th}}\text{ instance} \tag{2.17}$$

$$\mathbf{R}_{inert,body} = \text{Rotation matrix corresponding to } \hat{\mathbf{q}}_{k-1} \tag{2.18}$$

$$\mathbf{g} = \text{Gravity vector w.r.t inertial frame} \tag{2.19}$$

$$\otimes = \text{Quaternion composition operator.} \tag{2.20}$$

Error state covariance matrix update:

$$\check{\mathbf{P}}_k = \mathbf{F}_x\hat{\mathbf{P}}_{k-1}\mathbf{F}_x^T + \mathbf{F}_i\mathbf{Q}_i\mathbf{F}_i^T \tag{2.21}$$

where

$$\mathbf{P}_k = \text{Error state covariance matrix at k}^{\text{th}}\text{ instance} \tag{2.22}$$

$$\mathbf{F}_x = \left.\frac{\partial f}{\partial \boldsymbol{\delta}\mathbf{x}}\right|_{\mathbf{x}=\hat{\mathbf{x}}_{k-1},\boldsymbol{\delta}\mathbf{x}=\mathbf{0},\mathbf{w}=\mathbf{0}} \tag{2.23}$$

$$\mathbf{F}_i = \left.\frac{\partial f}{\partial \mathbf{w}}\right|_{\mathbf{x}=\hat{\mathbf{x}}_{k-1},\boldsymbol{\delta}\mathbf{x}=\mathbf{0},\mathbf{w}=\mathbf{0}} \tag{2.24}$$

$$f(.) = \text{Function representing the motion model} \tag{2.25}$$

$$\mathbf{w} = \text{Process noise vector} \tag{2.26}$$

$$\mathbf{Q}_i = \text{Process noise covariance matrix.} \tag{2.27}$$

The equations pertaining to the correction process, upon receiving a measurement update is given below.

$$\mathbf{K}_k = \check{\mathbf{P}}_k \mathbf{H}^T \left( \mathbf{H} \check{\mathbf{P}}_k \mathbf{H}^T + \mathbf{V} \right)^{-1} \tag{2.28}$$

$$\hat{\mathbf{P}}_k = \left( \mathbf{I} - \mathbf{K}_k \mathbf{H} \right) \check{\mathbf{P}}_k \tag{2.29}$$

$$\hat{\boldsymbol{\delta}\mathbf{x}}_k = \mathbf{K}_k \left( \mathbf{y} - h\left( \check{\mathbf{x}}_k \right) \right) \tag{2.30}$$

$$\hat{\mathbf{x}}_k = \check{\mathbf{x}}_k \oplus \hat{\boldsymbol{\delta}\mathbf{x}}_k \tag{2.31}$$

$$\hat{\boldsymbol{\delta}\mathbf{x}}_k \leftarrow \mathbf{0}. \tag{2.32}$$

Here,

$$h(.) = \text{Measurement function} \tag{2.33}$$

$$\mathbf{H} = \text{Jacobian of the measurement function, relative to the error state, evaluated at zero} \tag{2.34}$$

$$\mathbf{V} = \text{Measurement noise matrix} \tag{2.35}$$

$$\mathbf{y} = \text{Measurement vector} \tag{2.36}$$

$$\mathbf{I} = \text{Identity matrix of suitable dimension} \tag{2.37}$$

$$\oplus = \text{Operator representing the combination of nominal state and error state vectors.} \tag{2.38}$$

For a detailed presentation of the matrices involved in above equations, refer Appendix 4[28].

### *2.5.2 Stochastic cloning and backward smoothing*

Following the works of Emter et. al[29], we have implemented a stochastic cloning framework along with the ES-EKF, as a mean of integrating relative measurements obtained from sensors such as wheel odometer and visual/LiDAR odometry algorithms. Since a relative measurement relates the current state of the vehicle to a previous state, it is required to propagate the corrections resultant from absolute measurements to all the previous states as well. In-order to achieve this, a Rauch-Tung-Striebel (RTS) backward smoother was integrated.

### *2.5.3 State buffer*

It is essential to store a window of previous estimates obtained as the output of the ES-EKF, in-order to be used when fusing a relative measurement, under stochastic cloning

framework. A buffer with predefined length was implemented to achieve this functionality. Once a prediction is done by the ES-EKF, the estimate is added to the buffer. If the buffer is full, the oldest estimate will be removed, keeping the buffer length a constant.

Other than the estimate itself, the motion model inputs that resulted the prediction of the estimate and the prediction covariance matrix are also stored in this buffer. In addition to the stochastic cloning mechanism, they are used in propagating the effect of an absolute measurement correction across all the states of the buffer.

### *2.5.4 Time synchronization*

Measurements from different sensors arrive at different times, in different rates. These asynchronous arrivals are handled using the ROS's in-built multi-threading behaviour of subscriber callback functions. Each sensor publishes data to its own ROS topic, for which the localization module is subscribed to. Hence, each publication will invoke a callback function, which includes the logic for carrying out either a prediction or a correction step, depending on the measurement received.

When an absolute correction is to be carried out, the filter first searches the state buffer, for the state with the closest timestamp to that of the measurement. Then it performs correction steps on that state. After it has been completed, the effect of the correction has to be propagated to all the remaining states in the buffer. States having later timestamps will be adjusted by re-predicting them based on the corrected state. States with earlier timestamps will be adjusted through performing RTS smoothing. A similar procedure is carried out upon receiving a relative measurement, except that, backward smoothing with RTS will not be performed.

In this manner, a delayed measurement can still contribute to a correction, unless it is too delayed, so that the state corresponding to its timestamp no longer exists in the state buffer.

If a prediction is to be carried out, the filter will only search for the latest timestamp of the states. If it is greater than that of the prediction input, the input will be neglected. Otherwise, a prediction will be carried out and the newly predicted state will be added to the state buffer.

### *2.5.5 Zero Velocity Update measurements*

Some of the physical constraints that govern the motion of a vehicle can be incorporated into the sensor fusion mechanism, in-order to make the estimate more accurate. ZUPT measurements, generated based on the fact that the motion of a vehicle is constrained only towards its longitudinal direction, are one such constraint that can be employed to couple

the positional measurements with the heading estimation [30].

However, it should be noted that, the advantages of ZUPT measurements come at a cost. Since they are fed into the ES-EKF as ordinary absolute measurements, they reduce the estimated covariance of the error, causing the filter to be overconfident on its estimate. Furthermore, once diverged due to erroneous measurements, the estimate takes a longer time to converge back to the ground truth, after starting to receive correct measurements. These effects should be balanced out by carefully tuning the noise variance of the ZUPT measurement.

## 2.6 Visual Odometry

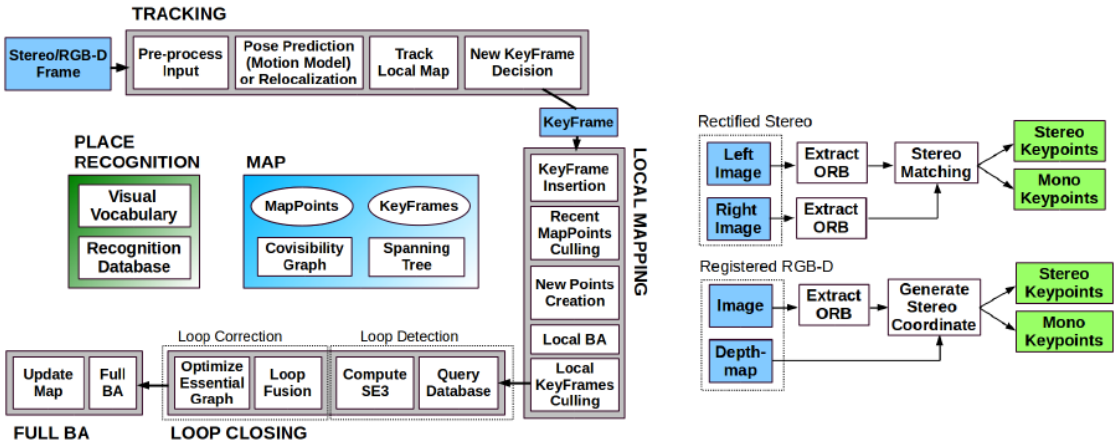
Out of the sensors that we use for localization process, cameras are the cheapest. However, it is capable of providing rich information of the environment, which allows us to recognize the environment robustly and accurately. Therefore, camera based visual odometry SLAM solutions are becoming very popular, and can be used in our system as a solution for GNSS outages.

### *2.6.1 System Selection*

Monocular camera based systems are the most commonly used for visual odometry because of the low cost and the smaller sensor setup. However, estimations are not reliable and inaccurate because depth cannot be calculated using one camera [1]. Therefore, stereo camera based systems were selected as the best solution for our system. As implementation of this algorithm was not within the project scope, we had to find a solution with an existing implementation. Following criteria were used when selecting a suitable method:

- Should not depend on previously generated feature maps
- Should provide real time output
- Accuracy
- Availability of proper implementation

ORB SLAM2 [1] and Stereo Parallel Tracking and Mapping (S-PTAM) [31] were the best solutions available. Furthermore, they had proper implementations, available for free. Comparing the results they have obtained by testing these algorithms using the KITTI dataset [32], ORB SLAM2 showed superior perfomance over S-PTAM. Furthermore, the implementation of the ORB SLAM2 algorithm was fairly detailed. Hence, we selected ORB SLAM2 as the best solution.




(a) System Threads and Modules. (b) Input pre-processing

Figure 2.3: ORB SLAM2 architecture [1]

### *2.6.2 ORB SLAM2 algorithm*

This is an extension of the previously implemented ORB SLAM2 algorithm [33], which was only for the monocular systems. In the ORB SLAM2, they have extended the previous implementation for the stereo cameras with accuracy improvements[1].

Pre-processing architecture is shown in figure 2.3(b). Features of the rectified stereo images will be extracted using the ORB feature extractor [34] and extracted features will be matched with the previous frame to identify the stereo key points.

General view of the system architecture is shown in figure 2.3(a). System is consisted of three main parallel threads, namely;

1. the Tracking thread to localize the camera with every frame by finding feature matches to the local map and minimizing the re-projection error applying motion-only Bundle Adjustment (BA)

2. the Local Mapping thread to manage the local map and optimize it, performing local BABA

3. the Loop Closing thread to detect large loops and correct the accumulated drift by performing a pose-graph optimization [1].

In the existing code, the vehicle is localized on a local map created using the feature points. However, for our system, we needed to get the relative transform of the vehicle, between two stereo frames. Hence, we modified the existing algorithm to receive relative transform of the vehicle, directly, as the output. The result is published as a ROSROS message.

ORB SLAM2 has been designed and setup only for the KITTI and Technical University of Munich (TUM) datasets. We added the image rectification functionality for the KAIST, as we will be using that dataset for our testing purposes.

One drawback of the ORB SLAM2 algorithm is the absence of a reliability metric (eg: estimated variance of the error) for the estimate. In order to circumvent this problem, we have been testing different methods to obtain an error covarince matrix. One of the methods tested was to estimate covariance values based on the number of features matched between each pair of frames. We hypothesized that the error should be reduced if a higher number of features have been detected. However, this hypothesis has been observed to be invalid (see section 3.3). Hence, alternatives are being sought to overcome this issue.

## 2.7 Implementation on Robot Operating System

The system was implemented on ROS, targetting the easy integration with other modules (path planning, perception etc.) of the autonomous vehicle. Furthermore, the ROS environment facilitates the following;

- Visualizing results (through Rviz, Multiplot etc.)
- In-built multi-threading for handling asynchronous data feeds
- Controlling simulation time
- Inter-process communication through ROS topics
- Debugging (using tools such as Node Graphs, TF Tree etc.)
- Handling transformations (through TF)

Apart from the nodes for visual and LiDAR odometry algorithms, the system consists of 3 main nodes, namely, the Data Feeder, Locator and the Evaluator. Each of these nodes will be described separately, in the following discussion. Figure 2.4 shows the Node Graph obtained for the system.

### *2.7.1 Data Feeder*

This ROS node is responsible for publishing sensor data obtained from the dataset, into pre-defined ROS topics. Furthermore, it is capable of controlling the simulation clock, so that the simulation timescale can be adjusted. It carries out all the data conversions required (Eg: GNSS sensor data need to be converted from World Geodetic System (WGS) 84 to the local inertial frame) and publishes the data according to the rates specified by the specifications of the corresponding sensor. Each dataset in use requires a separate Data Feeder, tailored to suit its data format specifications.

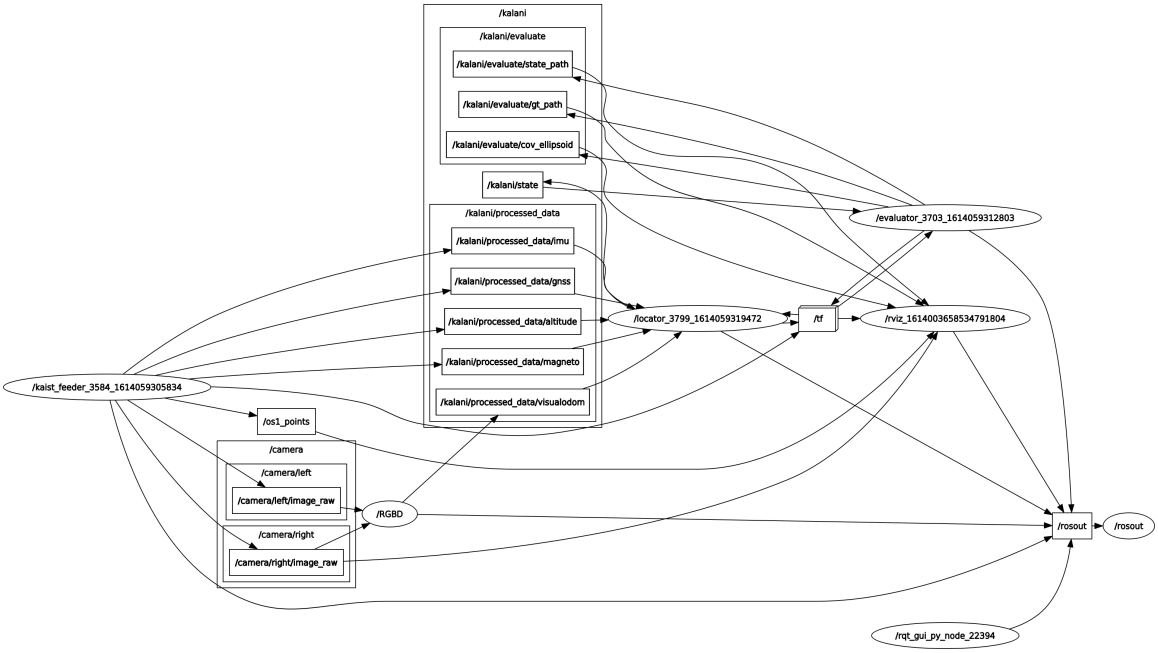



Figure 2.4: ROS Node Graph

### *2.7.2 Locator*

This is the node that is responsible for invoking prediction and correction functions of the ES-EKF, upon receiving data from the Data Feeder. Once a prediction is done, it publishes the latest state estimate both as a TF frame update as well as a ROS message under a pre-defined topic. The structure of the ES-EKF (including state variables, motion model and its Jacobians) is defined within this node, along with the measurement matrices for both absolute and relative measurement updates.

### *2.7.3 Evaluator*

Purpose of this node is to calculate and publish the errors and error confidence bounds ($3\sigma$ bounds) for the state estimate. In addition, it publishes the errors caused in the GNSS and magnetometer measurements, with respect to the ground truth. These errors can be visualized on either Rviz or Multiplot tools. Furthermore, it keeps a history of the paths of the estimate and the ground truth, which is also can be visualized on Rviz.

# Chapter 3

# RESULTS

NCLT dataset was used for the comparison of different filters. Functionality of each component has been tested with KITTI, KAIST and NCLT datasets.

## 3.1 Comparison of Bayesian filters

Data from the NCLT dataset has been used for this comparison and the results shown in the discussion have been obtained using sequence 2013.01.10.

### *3.1.1 Performance of translation estimation*

Overall performances (RMS error) of all the filters are better than the raw GNSS measurements. UKF has given the worst estimation while PF and ES-EKF have given better results. Table 3.1 shows a summary of the results.

| **Method** | **x direction(m)** | **y direction(m)** | **z direction(m)** | **Overall position(m)** |
|---|---|---|---|---|
| GPS | 6.3278 | 9.8746 | 5.6387 | 13.0133 |
| ESKF | 4.6824 | 2.8655 | 7.0109 | 8.9044 |
| UKF | 4.7371 | 2.7663 | 11.2554 | 12.5211 |
| PF | 4.4488 | 2.8587 | 6.8126 | 8.6242 |

Table 3.1: Translational RMS errors for different Bayesian filters

Even though PF has given the most accurate results, as shown in figures 3.1 to 3.4, output of the filter is not smooth. Furthermore, the estimate rapidly diverges upon receiving erroneous GNSS measurements (see figure 3.4). Adding random samples or increasing the size of the particle set are alternatives for this problem. However, both of these solutions will increase the computational complexity. If both RMS error and smoothness are considered, ES-EKF gives the best solution. Z-direction has shown the worst error in all the three filters resulting a large overall positional error. This behaviour is a result of the large error in the altitude measurement, obtained by the GNSS receiver. It can be mitigated by using an Altimeter (which is only present in the KAIST dataset, out of the three datasets being used).

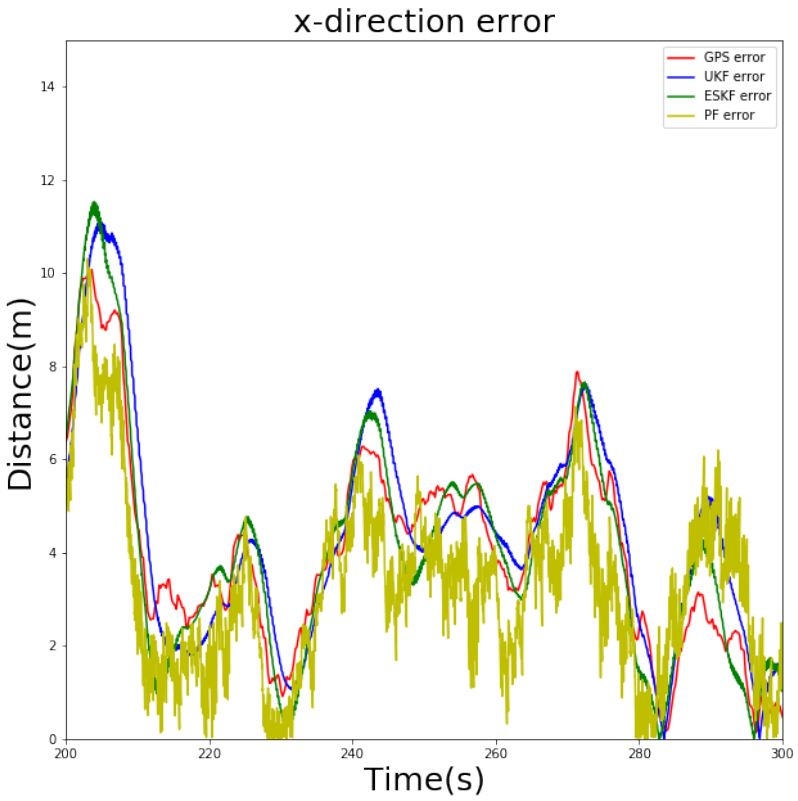



Figure 3.1: Filter comparison - error in x direction

### *3.1.2 Performance of orientation estimation*

Raw orientation measurement obtained using the IMU + Magnetometer is better than the estimates of all the three filters (see table 3.2). This is due to the noise in the angular velocity measurement, which is confirmed from the reduction of the error when the angular measurement noise variance parameter is increased. PF has given the worst estimation. Results of the other two filters are somewhat similar. However, UKF has given the best estimation. Error plots are shown in figures 3.5, 3.6 and 3.7.

| **Method** | **Roll (deg)** | **Pitch (deg)** | **Yaw (deg)** |
|---|---|---|---|
| IMU+Magnetometer | 1.1555 | 1.7171 | 7.7622 |
| ESKF | 2.8258 | 3.3362 | 8.8584 |
| UKF | 1.7216 | 1.8304 | 8.6043 |
| PF | 14.76759 | 12.32548 | 37.5694 |

Table 3.2: Rotational RMS errors of different Bayesian filters

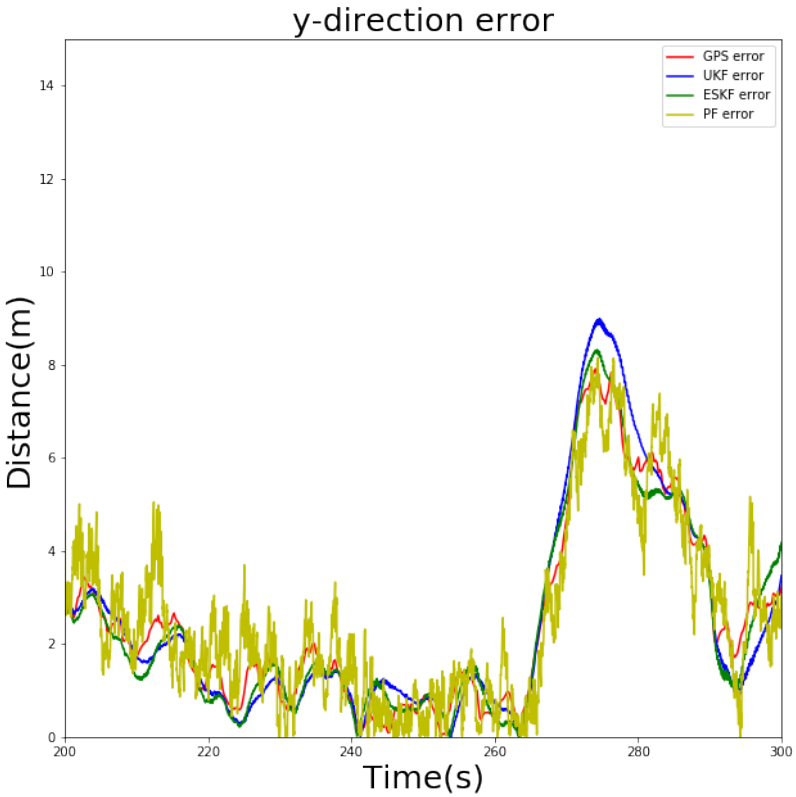



Figure 3.2: Filter comparison - error in y direction

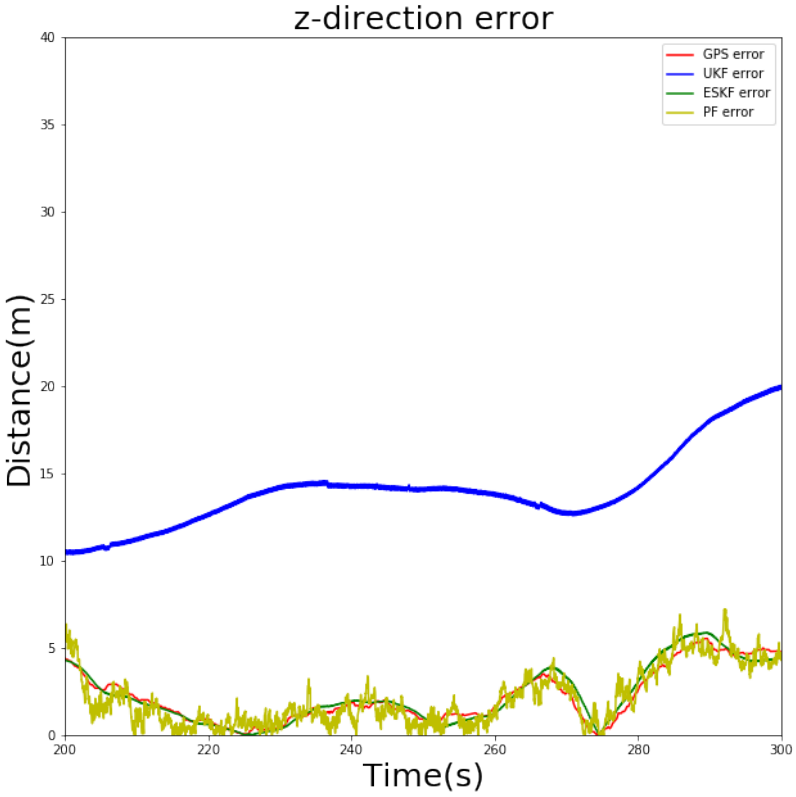
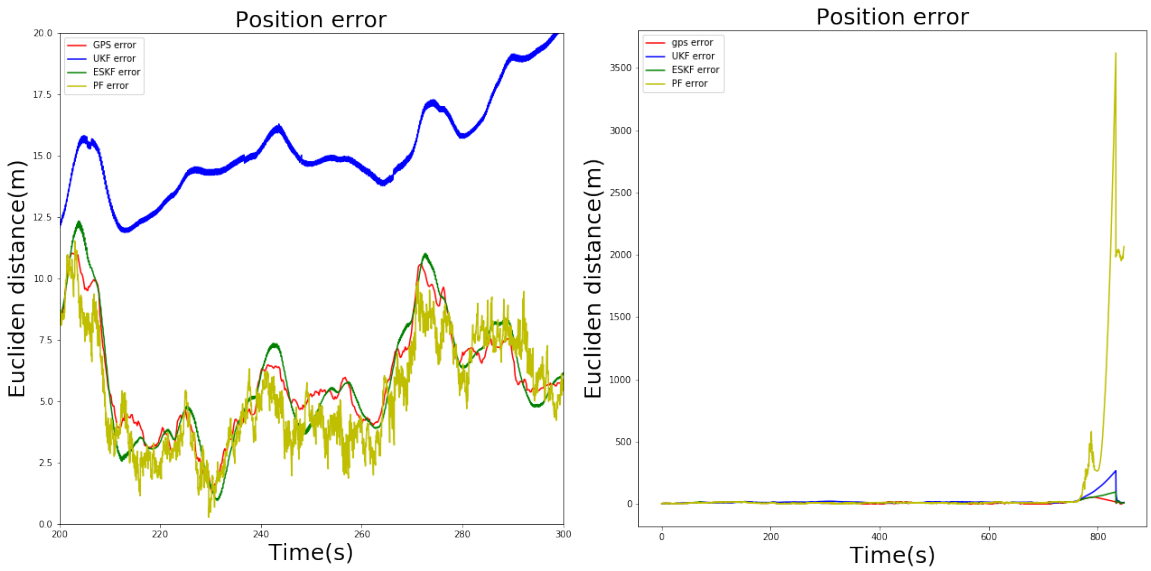



Figure 3.3: Filter comparison - error in z direction




(a) Excluding the diverged PF result

(b) Including the diverged PF result

Figure 3.4: Filter comparison - overall translational error

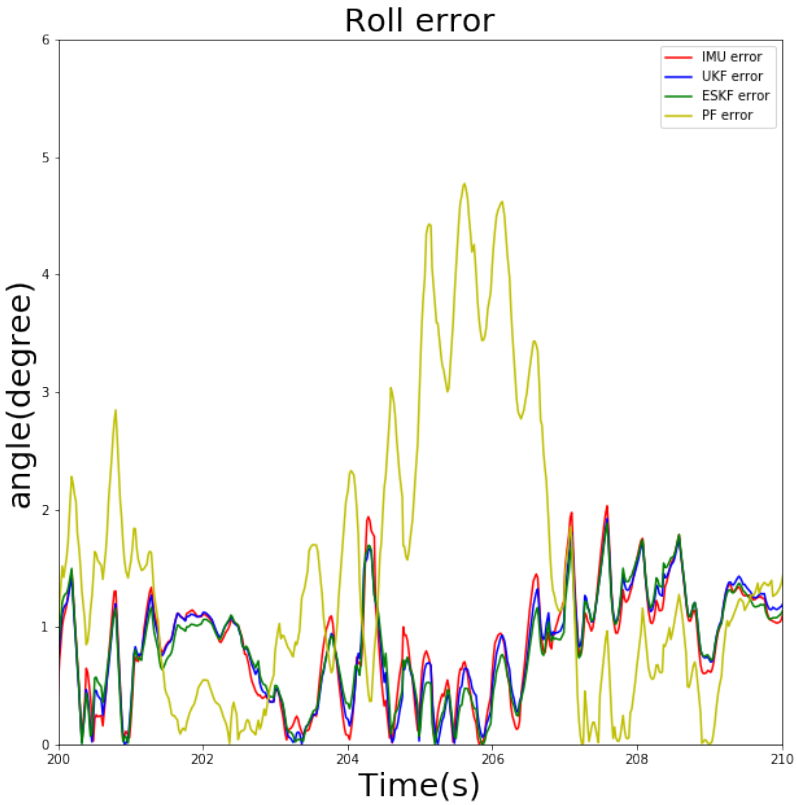



Figure 3.5: Filter comparison - error in roll

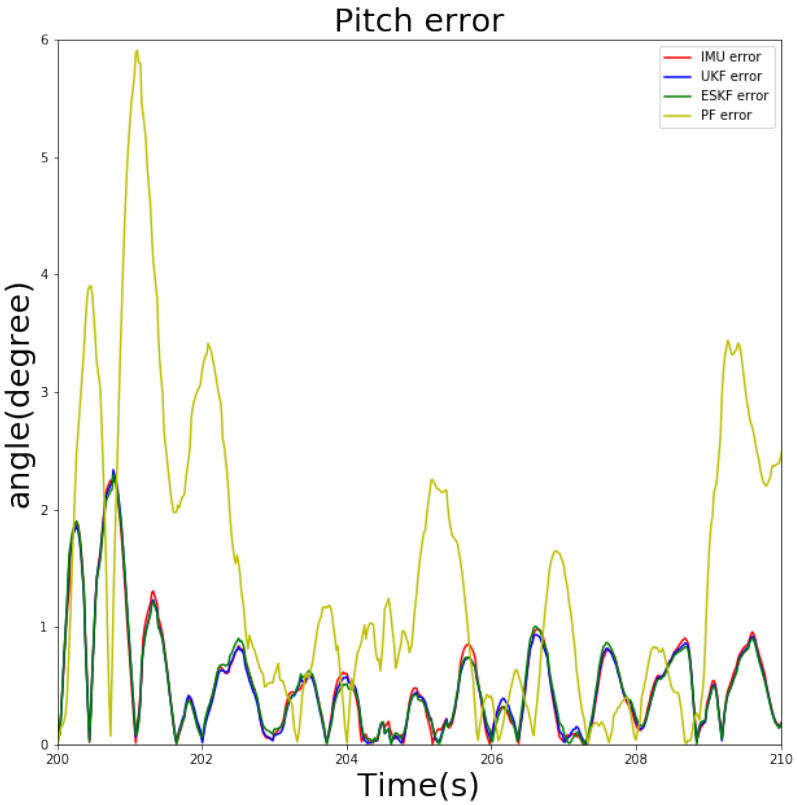



Figure 3.6: Filter comparison - error in pitch

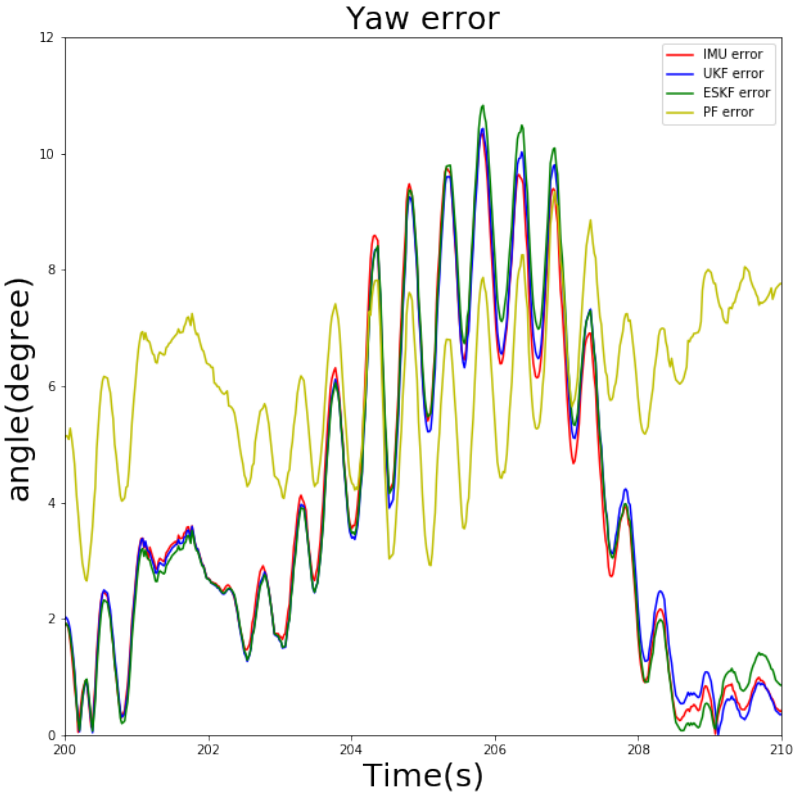
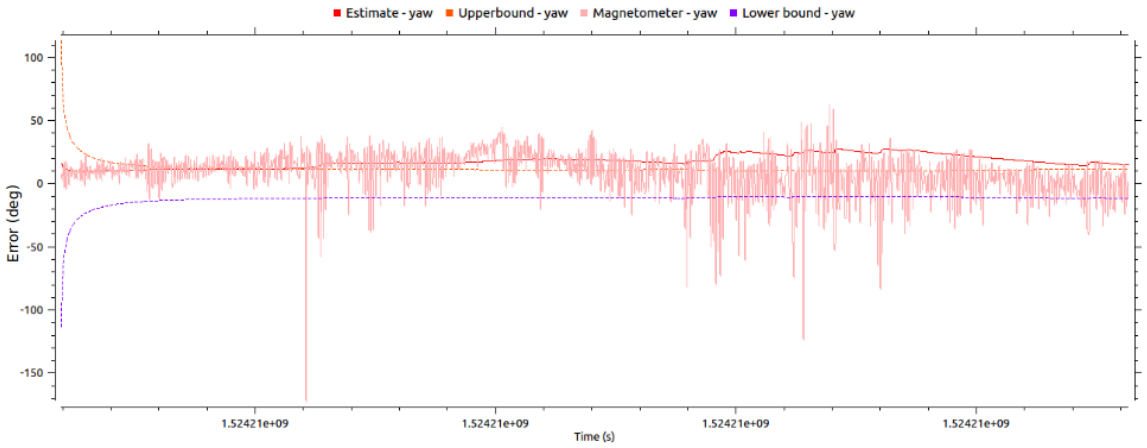


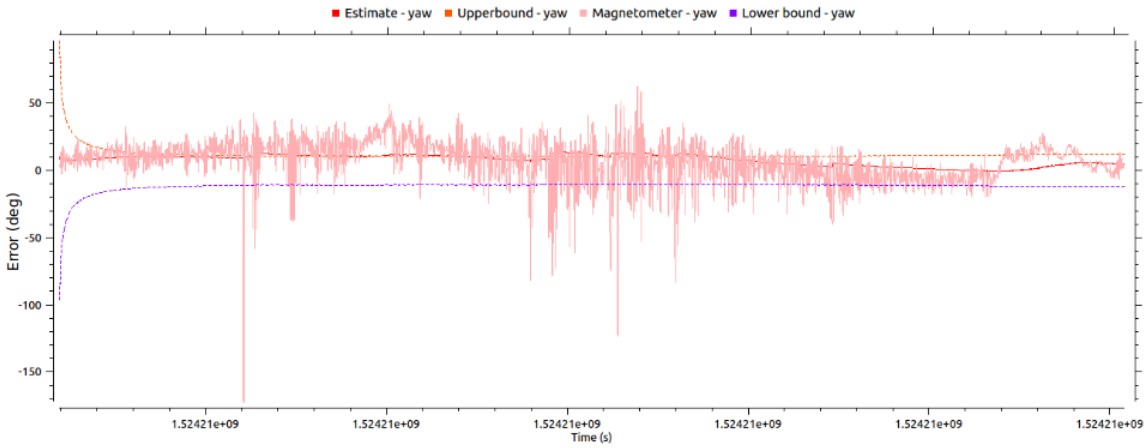

Figure 3.7: Filter comparison - error in yaw

## 3.2 Sensor fusion mechanism

Adding ZUPT measurements has a significant effect on the accuracy of the yaw estimate, as seen from figure 3.8. When ZUPT measurements are applied, the yaw estimate error remains within the estimated error bounds ($3\sigma$ bounds), despite the fact that the magnetometer-estimated yaw is biased.




(a) Yaw estimate without ZUPT measurements




(b) Yaw estimate with ZUPT measurements

Figure 3.8: Effect of ZUPT measurements on the yaw estimate. Pink: Magnetometer estimated yaw, Red: Filter output, Orange (dashed): Estimated upper bound of the error, Violet (dashed): Estimated lower bound of the error.

## 3.3 Visual Odometry

Experiments were conducted using the KITTI and KAIST datasets on the ORB SLAM2. Images in the KITTI odometry dataset are rectified images and the ones in the KAIST dataset are non-rectified. Therefore, images should be rectified in system for the KAIST dataset. Results in the subsequent discussion have been obtained using the KITTI dataset.

Coordinate axes, x, y and z refers to the coordinate axes of the initial camera frame (Coordinate frame of the camera, when the vehicle started moving).

While running ORB SLAM2 algorithm, it creates a feature points map using the detected feature points (see figure 3.9) and localize the vehicle in the feature map. Figure 3.10 shows the estimated path of the vehicle in the map. Blue coloured triangles show the localized vehicle (pose if the camera) for every image frame it receives.


Figure 3.9: Feature points detected in a frame of KITTI sequence 09 when running ORB SLAM2. Green colour squares indicate the feature points.

Figure 3.11 shows the position error in the directions of x, y and z with the frame index, and figure 3.12 shows the error in the orientation (roll, pitch and yaw) with frame index. It can be observed that the total position error in a given direction increase with time due to the accumulation of the relative position errors in the estimate.

In-order to test the effect of error accumulation, we subtracted the error of the previous frame, from the total error of the current frame, thereby removing the effect of accumulated error for each frame. This revealed that the error in the relative position/orientation estimate of the algorithm is extremely low, which is desirable when using the output as a relative measurement in the fusion mechanism (see figures 3.13 and 3.14).

With the intention of obtaining insights for implementing an error covariance estimation mechanism for ORB SLAM2, a histogram was obtained by counting the number of feature points detected in a given frame. It appeared to be concentrated around 300 for the KITTI dataset (figure 3.15). Furthermore, we calculated the average RMS error obtained for a given number of matched points in a frame (see figure 3.16), which showed that up to some optimal number, the RMS error decreases with the number of matched points. However, beyond this point, it was observed that the RMS error increases. A possible reason for this is the confusion caused by the large number of feature points (with similar features) detected very close to each other.

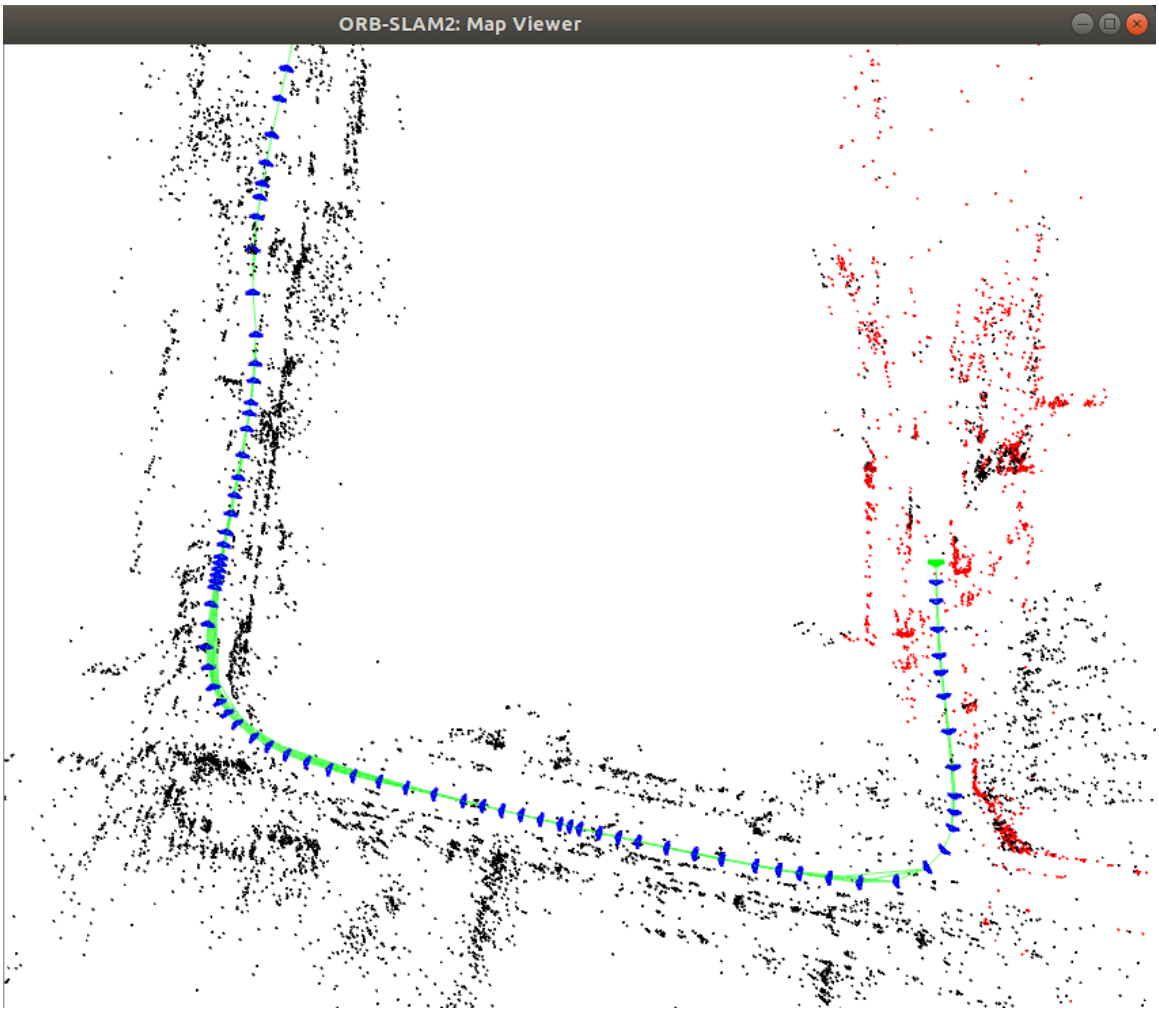



Figure 3.10: Part of the feature point map generated from ORB SLAM2, using KITTI sequence 00. Estimated path of the vehicle is shown in green colour. Blue triangles show the pose of the camera, in the respective frames.

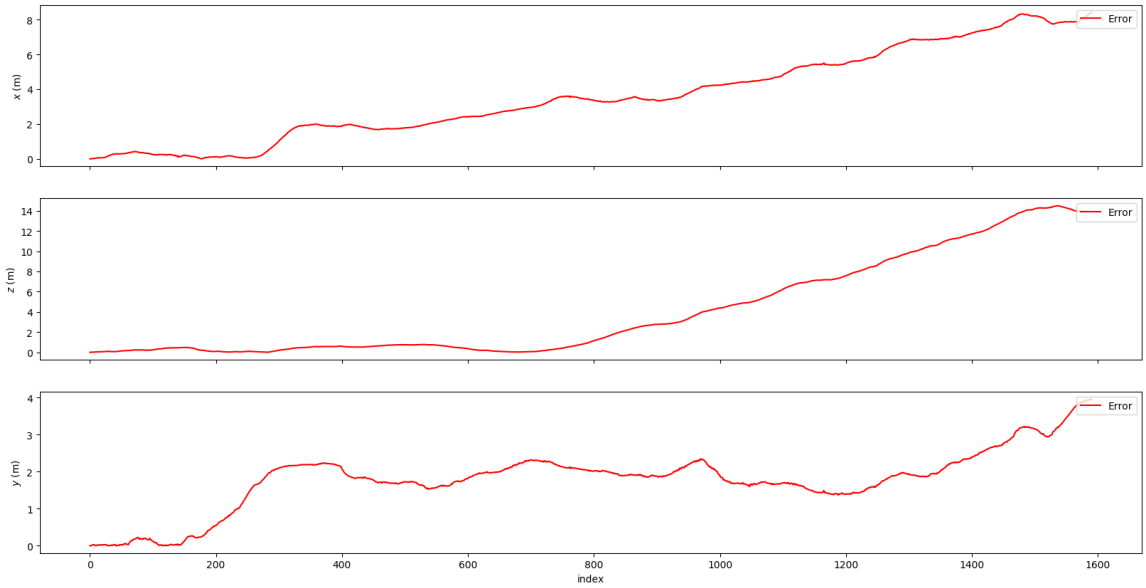



Figure 3.11: Positional error of ORB SLAM2 (KITTI sequence 09)

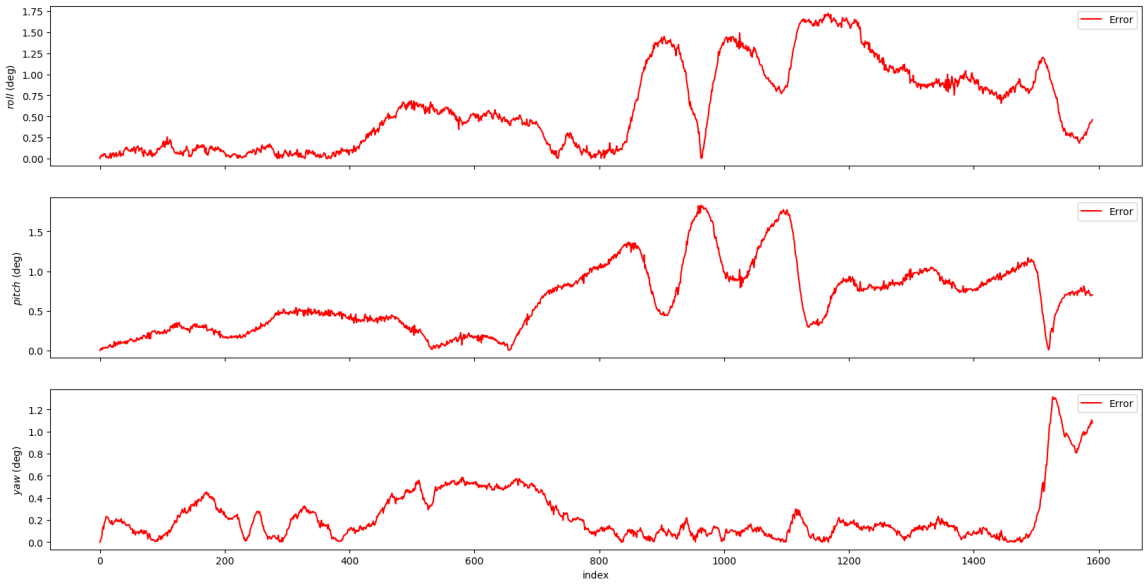



Figure 3.12: Rotational error of ORB SLAM2 (KITTI sequence 09)

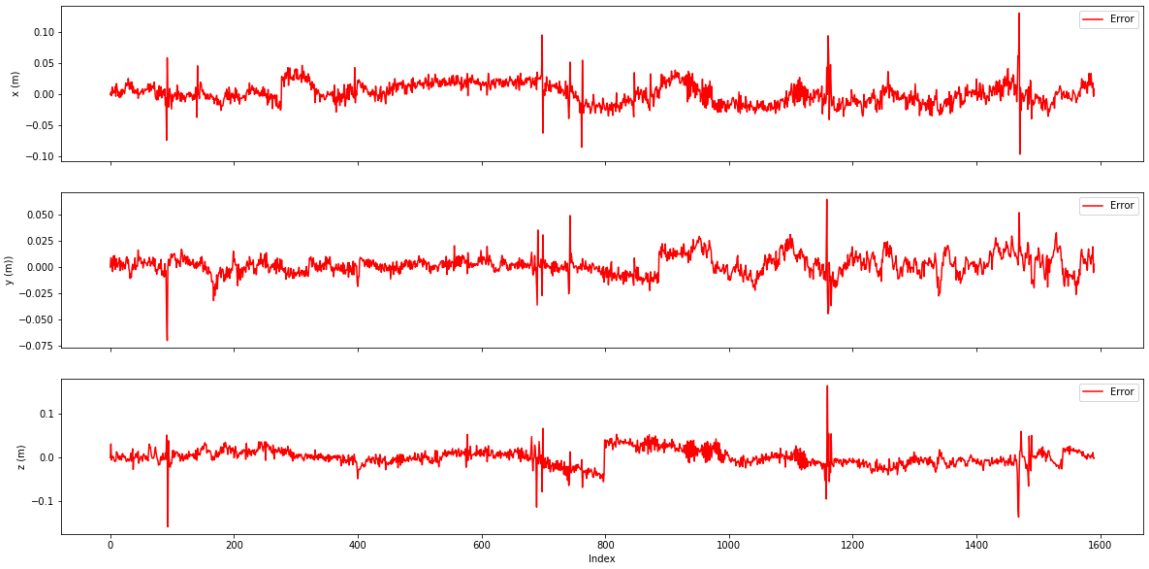



Figure 3.13: Positional error of ORB SLAM2, after removing the effect of error accumulation

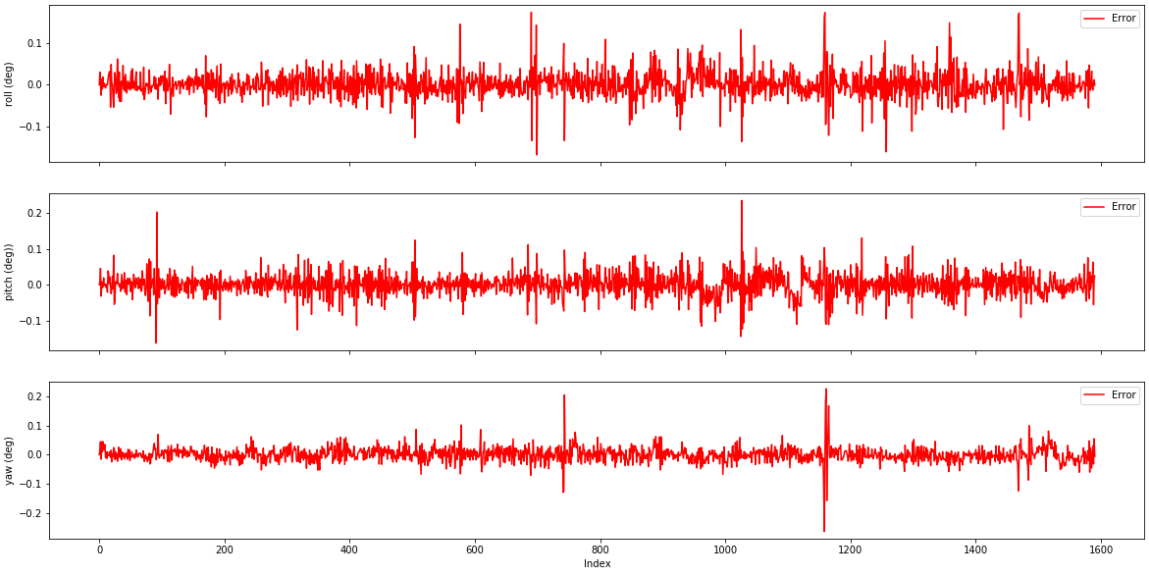



Figure 3.14: Rotational error of ORB SLAM2, after removing the effect of error accumulation

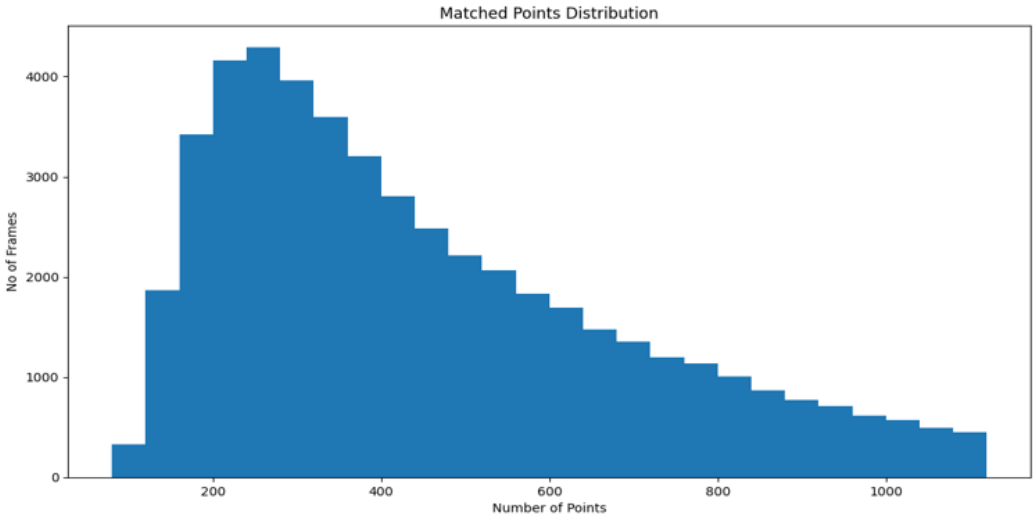



Figure 3.15: Histogram of the number of matched points. Vertical axis denotes the number of frames, with a given number of feature points detected.

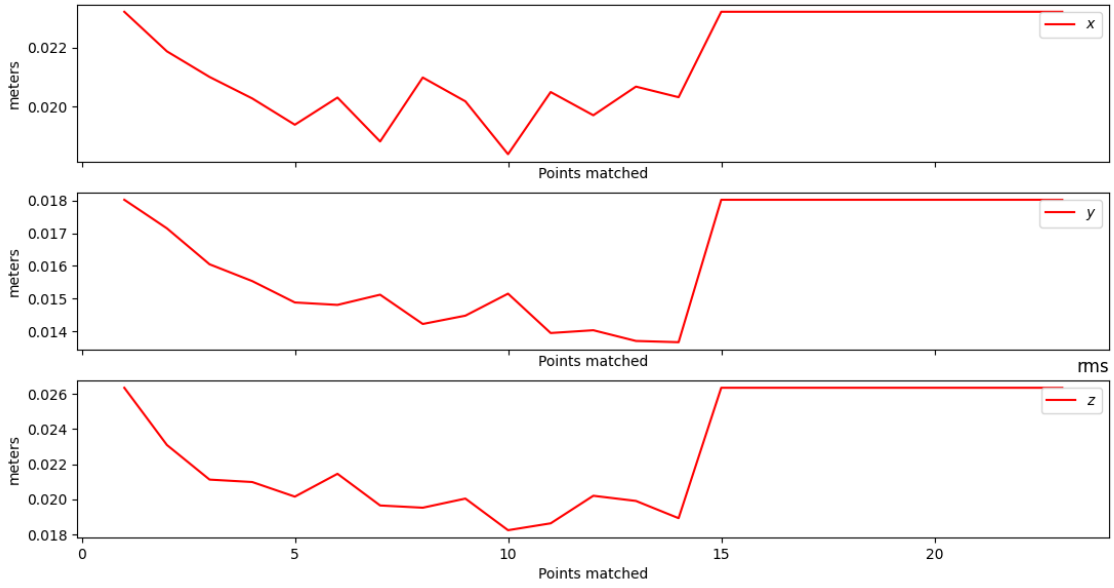



Figure 3.16: Variation of average RMS error, with the number of matched points

# Chapter 4

# DISCUSSION AND CONCLUSION

# REFERENCES


[1] R. Mur-Artal and J. D. Tardós, "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[2] S. Liu, J. Tang, Z. Zhang, and J. Gaudiot, "Computer architectures for autonomous driving," *Computer*, vol. 50, no. 8, pp. 18–25, 2017.

[3] A. C. Hernandes, A. S. Brito, H. Roncancio, D. V. Magalhães, M. Becker, R. C. B. Sampaio, and B. T. Jensen, "Gisa: A brazilian platform for autonomous cars trials," in *2013 IEEE International Conference on Industrial Technology (ICIT)*, 2013, pp. 82–87.

[4] P. Lamon, S. Kolski, R. Triebel, R. Siegwart, and W. Burgard, "The smartter for elrob 2006 - a vehicle for fully autonomous navigation and mapping in outdoor environments," *Navigation*, 2006.

[5] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke, D. Johnston, S. Klumpp, D. Langer, A. Levandowski, J. Levinson, J. Marcil, D. Orenstein, J. Paefgen, I. Penny, A. Petrovskaya, M. Pflueger, G. Stanek, D. Stavens, A. Vogt, and S. Thrun, "Junior: The stanford entry in the urban challenge," *Journal of Field Robotics*, vol. 25, no. 9, pp. 569–597, 2008. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.20258

[6] "Ieee spectrum: Technology, engineering, and science news, 2020." [Online]. Available: http://spectrum.ieee.org/automaton/robotics/artificialintelligence/googles-autonomous-car-takes-to-the-streets

[7] M. Bertozzi, A. Broggi, E. Cardarelli, R. I. Fedriga, L. Mazzei, and P. P. Porta, "Viac expedition toward autonomous mobility [from the field]," *IEEE Robotics & Automation Magazine*, vol. 18, no. 3, pp. 120–124, 2011.

[8] "Apollo open platform." [Online]. Available: https://apollo.auto/developer.html

[9] "Autoware.auto." [Online]. Available: https://www.autoware.auto/

[10] G. Wan, X. Yang, R. Cai, H. Li, Y. Zhou, H. Wang, and S. Song, "Robust and precise vehicle localization based on multi-sensor fusion in diverse city scenes," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 4670–4677.

[11] A. Soloviev, "Tight coupling of gps, laser scanner, and inertial measurements for navigation in urban environments," in *2008 IEEE/ION Position, Location and Navigation Symposium*, 2008, pp. 511–525.

[12] J. Levinson and S. Thrun, "Robust vehicle localization in urban environments using probabilistic maps," in *2010 IEEE International Conference on Robotics and Automation*, 2010, pp. 4372–4378.

[13] S. I. Roumeliotis, G. S. Sukhatme, and G. A. Bekey, "Circumventing dynamic modeling: evaluation of the error-state kalman filter applied to mobile robot localization," in *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, vol. 2, 1999, pp. 1656–1663 vol.2.

[14] J. Levinson, J. Askeland, J. Becker, J. Dolson, D. Held, S. Kammel, J. Z. Kolter, D. Langer, O. Pink, V. Pratt, M. Sokolsky, G. Stanek, D. Stavens, A. Teichman, M. Werling, and S. Thrun, "Towards fully autonomous driving: Systems and algorithms," in *2011 IEEE Intelligent Vehicles Symposium (IV)*, 2011, pp. 163–168.

[15] A. SOLOVIEV, D. BATES, and F. VAN GRAAS, "Tight coupling of laser scanner and inertial measurements for a fully autonomous relative navigation solution," *NAVIGATION*, vol. 54, no. 3, pp. 189–205, 2007. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/j.2161-4296.2007.tb00404.x

[16] I. Baldwin and P. Newman, "Road vehicle localization with 2d push-broom lidar and 3d priors," in *2012 IEEE International Conference on Robotics and Automation*, 2012, pp. 2611–2617.

[17] E. Ward and J. Folkesson, "Vehicle localization with low cost radar sensors," in *2016 IEEE Intelligent Vehicles Symposium (IV)*, 2016, pp. 864–870.

[18] K. Yoneda, H. Tehrani, T. Ogawa, N. Hukuyama, and S. Mita, "Lidar scan feature for localization with highly precise 3-d map," in *2014 IEEE Intelligent Vehicles Symposium Proceedings*, 2014, pp. 1345–1350.

[19] S. P. Won, W. W. Melek, and F. Golnaraghi, "A kalman/particle filter-based position and orientation estimation method using a position sensor/inertial measurement unit hybrid system," *IEEE Transactions on Industrial Electronics*, vol. 57, no. 5, pp. 1787–1798, 2010.

[20] N. Carlevaris-Bianco, A. K. Ushani, and R. M. Eustice, "University of michigan north campus long-term vision and lidar dataset," *The International Journal of Robotics Research*, vol. 35, no. 9, pp. 1023–1035, 2016. [Online]. Available: https://doi.org/10.1177/0278364915614638

[21] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013. [Online]. Available: https://doi.org/10.1177/0278364913491297

[22] J. Jeong, Y. Cho, Y.-S. Shin, H. Roh, and A. Kim, "Complex urban dataset with multi-level sensors from highly diverse urban environments," *The International Journal of Robotics Research*, p. 0278364919843996, 2019.

[23] Z. Yan, L. Sun, T. Krajnik, and Y. Ruichek, "EU long-term dataset with multiple sensors for autonomous driving," in *Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[24] M. St-Pierre and D. Gingras, "Comparison between the unscented kalman filter and the extended kalman filter for the position estimation module of an integrated navigation information system," in *IEEE Intelligent Vehicles Symposium, 2004*. IEEE, 2004, pp. 831–835.

[25] V. Madyastha, V. Ravindra, S. Mallikarjunan, and A. Goyal, "Extended kalman filter vs. error state kalman filter for aircraft attitude estimation," in *AIAA Guidance, Navigation, and Control Conference*, 2011, p. 6615.

[26] 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)*. Ieee, 2000, pp. 153–158.

[27] F. Ababsa, M. Mallem, and D. Roussel, "Comparison between particle filter approach and kalman filter-based technique for head tracking in augmented reality systems," in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, vol. 1. IEEE, 2004, pp. 1021–1026.

[28] J. Solà, "Quaternion kinematics for the error-state kalman filter," 2017.

[29] T. Emter and J. Petereit, "Stochastic cloning and smoothing for fusion of multiple relative and absolute measurements for localization and mapping," in *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, 2018, pp. 1508–1513.

[30] G. Dissanayake, S. Sukkarieh, E. Nebot, and H. Durrant-Whyte, "The aiding of a low-cost strapdown inertial measurement unit using vehicle model constraints for land vehicle applications," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 5, pp. 731–747, 2001.

[31] T. Pire, T. Fischer, G. Castro, P. De Cristóforis, J. Civera, and J. Jacobo Berlles, "S-ptam: Stereo parallel tracking and mapping," *Robotics and Autonomous Systems*, vol. 93, pp. 27–42, 2017.

[32] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[33] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "Orb-slam: A versatile and accurate monocular slam system," *IEEE Transactions on Robotics*, vol. 31(5), p. 1147–1163, 2015.

[34] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *2011 International Conference on Computer Vision*, 2011, pp. 2564–2571.

# APPENDIX I