Multi-Sensor Fusion for Navigation of Ground Vehicles

by

Arsalan Ahmed

A thesis submitted to the Faculty of Graduate and Postdoctoral Affairs in partial fulfillment of the requirements for the degree of

Master of Applied Science

in

Aerospace Engineering

Carleton University
Ottawa, Ontario

Abstract

Navigation is a core requirement for autonomous vehicles and robotics. The objective of this thesis is to compute the navigation solution of a ground vehicle by fusing data from Inertial Navigation System (INS), Visual Odometry (VO), and Global Positioning System (GPS) using a Dual Extended Kalman Filter (DEKF) algorithm. The research in this thesis is conducted in three phases. The first phase deals with the development of a VO navigation system. In this phase the traditional Stereo Visual Odometry (SVO) methodology is analyzed, and an improvement is proposed at the pose estimation and pose optimization stages to present the Modified Stereo Visual Odometry (ModSVO) algorithm. The second phase deals with the development of INS/VO and INS/GPS integrated systems using EKF. It is shown that while accuracy improves compared to standalone sensors, but in case of VO or GPS failure the accuracy deteriorates. The third phase presents a solution to this problem by developing the INS/VO/GPS system using a Dual Extended Kalman Filter (DEKF) scheme. It is shown that the INS/VO/GPS system outperforms INS/VO and INS/GPS systems in cases of VO failure or GPS failure.

Acknowledgements

First of all, I would like to thank God for all the blessings bestowed upon me.

I would like to extend my gratitude to my supervisor, Professor Jurek Z. Sasiadek, whose guidance helped me achieve my research objectives.

I am deeply grateful to my Parents, who encouraged and supported me to take this step for obtaining master's degree and stood by me throughout this journey.

I would like to thank my Parents-in-Law, who spent their time and effort praying for me and inspiring me with words of encouragement.

To my wife Hira, words cannot express how invaluable your presence has been throughout this time. I could not have completed this journey without your love and support, and for that, I would like to wholeheartedly say, thank you.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

1.1 Background

The recent interest in development of autonomous ground vehicles has amplified the necessity of real-time accurate navigation solution. The task of navigation can usually be carried out by sensors such as IMU, GPS, camera etc., however, all real-world sensor readings are corrupted with noise to some degree, therefore relying on a single sensor results in erroneous estimates. In order to overcome this issue, data from multiple sensors is fused which results in improved accuracy.

An Inertial Measurement Unit (IMU) comprises of accelerometer and gyroscope sensors. It is able to provide the body specific force and the body angular velocity. This data is used to compute the position, velocity, and attitude of the vehicle through Inertial Navigation System (INS) [1]. The INS solution computation requires twice integration over time of body specific force, and once integration over time of angular velocity in order to obtain the pose information. This results in noise accumulation over time which causes a drift in estimates. Due to this reason INS on its own cannot be used for real-time navigation solution.

Global Navigation Satellite System (GNSS) is a satellite-based navigation system which can provide the position and velocity estimate of a vehicle. The most commonly used satellite system is the Global Positioning System (GPS) which has a network of 24+ satellites [2]. Unlike INS, the navigation solution obtained from GPS is not affected by the accumulation of error as the solution does not require integration over time and once the GPS receiver is able to connect to four or more satellites the position can be obtained with reasonable accuracy [3]**.** However, it still suffers from error sources such as low frequency, ionosphere delay, and receiver electronics noise [4], [5]. Furthermore, GPS signals require clear view of the sky. GPS signal outage occurs if there are tall trees, tall buildings, or if the vehicle is underground or in

a tunnel [6]. Due to these factors, GPS on its own cannot be used for real-time navigation solution.

One of the recent advancements in navigation systems is of Visual Odometry (VO). VO is an odometry technique in which the vehicle pose is estimated using sequential images from monocular or stereo cameras [7]. The VO solution can be used in GPS denied environments, however, computing the VO solution is more computationally extensive and it is affected by error accumulation over time. This is because the current camera pose is computed by concatenating the new transformation matrix with the previous camera pose. As a result, any errors present in computation of the previous pose will amalgamate with the errors present in the computation of the new pose resulting in drift of the navigation solution [8]. Furthermore, the environment in which the image is taken also affects the VO estimates. This includes lighting, snow-covered terrain, direct sunlight, or even large objects covering the camera field of view [9]. These factors may result in VO failing to obtain an accurate navigation solution. As a result, standalone VO is not suitable for real-time navigation.

Since each of the navigation systems discussed have some inherent drawbacks making them unsuitable for providing standalone navigation solution, therefore, to overcome this issue, sensor fusion algorithms are used to combine multiple sensors in a manner where they complement each other's strengths and diminish the weaknesses [10]. One of the sensor fusion techniques is combining the sensor data using the Extended Kalman Filter (EKF) [1]. For navigation of ground vehicles, EKF has been utilized for INS/VO fusion [10], and for INS/GPS fusion [11]. These integrated systems effectively overcome the drift issue in INS, the noise and low frequency issue of GPS, and error accumulation issue in VO. However, in case of GPS failure or VO failure, the system relies only on INS and as a result the navigation solution accumulates error.

This issue motivates the research problem tackled in this research. In this thesis, a Dual Extended Kalman Filter (DEKF) is used to develop an INS/VO/GPS integrated system to address the problem of navigation accuracy deterioration in case of GPS failure and VO failure.

1.2 Research contribution

The research conducted in this thesis aims to present improvements in the field of autonomous robotics and vehicles. The research presents the following contributions:

- Firstly, a Visual Odometry (VO) technique is proposed called Modified Stereo Visual Odometry (ModSVO) which improves the navigational accuracy in comparison to the traditional Stereo Visual Odometry (SVO) [12] by proposing changes in the pose estimation and pose optimization stages of the SVO pipeline. ModSVO combines the 2D-to-2D and 3D-to-2D pose estimation techniques, by estimating attitude from essential matrix decomposition [13] overcoming the non-orthonormality issues in the traditional SVO. The problem with this approach was the scale ambiguous translation vector estimate. This is overcome by minimizing the reprojection error using an updated cost function which only optimizes the translation vector, thereby, reducing the number optimizing parameters from 6 to 3. This is presented in section 4.4.
- Another important contribution is the utilization of ModSVO in the implementation of INS/VO integrated system using Extended Kalman Filter (EKF) with VO feedback. Generally, with stochastic cloning, a delayed state is appended to EKF state to improve estimation [14]. The disadvantage associated with this technique is an increase in size of the EKF state. In this thesis, the delayed position and attitude states are transformed to the camera coordinate frame directly used as an incremental step for VO pose update. This is done without appending any states to the original state vector. As a result, the size of the state does not increase. Furthermore, the sensor noise covariance matrix R_{vo} is populated using the standard deviation values of Sampson distances of inliers

obtained from estimating the essential matrix between consecutive frames, and the standard deviation values of residual reprojection errors for inliers after minimization of the reprojection errors. This is presented in section 5.3.

- The final contribution of this research is the development of INS/VO/GPS integrated system using Dual Extended Kalman Filter (DEKF). This algorithm utilizes two loosely coupled sequential EKFs to fuse the three sensors. The first EKF fuses INS and VO data, and the resulting state estimates are further corrected using the GPS data by the second EKF. In case of GPS failure, the system is able to switch to INS/VO and in case of VO failure it is able to switch to INS/GPS outperforming not only the standalone sensors but also the two sensor combinations in cases of sensor failure. This is presented in section 5.5.

1.3 Thesis Methodology

The methodology to complete the thesis objectives is presented as follows:

1. Review and implement Stereo Visual Odometry (SVO) algorithm and analyze its benefits and drawbacks.
2. Develop and implement a Modified Stereo Visual Odometry (ModSVO) algorithm in order to and present its improvements.
3. Implement standalone Inertial Navigation System (INS) and present its drawbacks as a standalone sensor.
4. Implement standalone Global Positioning System (GPS) and present its drawbacks as a standalone sensor.
5. Develop and implement INS/VO integrated system using Extended Kalman Filter (EKF) with VO feedback. Analyze its performance in normal conditions and with VO failure conditions in comparison to standalone INS and standalone ModSVO.

6. Develop and implement INS/GPS integrated system using EKF. Analyze its performance in normal conditions and with GPS failure conditions in comparison to standalone INS and standalone GPS.
7. Design and implement INS/VO/GPS integrated system using Dual Extended Kalman Filter (DEKF). Analyze its performance in normal conditions, with VO failure conditions, and with GPS failure conditions, in comparison to INS/VO and INS/GPS.

The simulations in the thesis are carried out using the MATLAB software [15].

1.4 Literature Review

The first phase of the research is concerned with development of Visual Odometry (VO) system. A review paper presented in [12] defines the basic components of feature based VO. For feature matching and detection, one of the most popular feature detectors is Harris Corner Detector [16]. However, its capacity is diminished in case of scale and rotation change between images. Lowe [17] presented a Scale Invariant Feature Transform (SIFT) detector which was scale and rotation invariant. Bay [18] presented Speeded-Up Robust Features (SURF) as an improvement to SIFT by using approximating Hessian using Box filters and using a 64-element descriptor as compared to 128-element vector in SIFT. For outlier rejection, the RANSAC algorithm in combination with the five-point algorithm for essential matrix estimation [13]. Utilizing VO for navigation and localization has been a popular choice in the literature. Cheng et al. [19] presents the algorithm used for Stereo Visual Odometry (SVO) for the rovers used in Mars. The research utilized Harris detector for feature detection, pixel intensity sum of squared difference stereo matching, and 3D-3D pose estimation algorithm for motion estimation. Kitt et al. [20] used this methodology for measuring displacement of moving stereo cameras in an urban environment without the use of additional sensors. They presented a Kalman Filter based approach using constant velocity model, removing the requirement for rectification and 3D scene reconstruction. Cao et al. [21] presents a combination of 3D-to-3D

and 3D-to-2D motion estimation algorithms. ORB feature detection method was used for keypoint selection and extraction, and the estimated pose was refined using local optimization through minimizing the reprojection error. Another approach based on selecting only ground features is presented in [22]. In this research a Kalman filter-based approach using Ackermann's model is utilized and the stability of the selected features is ensured by selecting the features present on the ground plane. Aqel et al. [9] provides a review of different VO approaches in literature and their comparisons.

The second phase of the research is concerned with INS/VO and INS/GPS sensor fusion. INS/GPS sensor fusion is a widely researched area in the field of navigation, and it has been employed in a variety of applications using different fusion techniques. Gonzalez et al. [23] presents a loosely coupled Extended Kalman Filter (EKF) solution for fusing low cost INS and GPS sensors which overcomes the drift issue of INS. Yang et al. [24] presents an improved Unscented Kalman Filter (UKF) derivation for INS/GPS integration by using an adaptive method of correlational inference. Glavine et al. [25] fuses INS with GPS using EKF. The research compares the single EKF model and an Interactive Multiple Model (IMM) EKF. The IMM approach uses two modes with varying noise parameters allowing the EKF to adapt to the optimal model with variation in the environment. The algorithm was tested using the INS and GPS data available from the KITTI (Karlsruhe Institute of Technology and Toyota Technological Institute) dataset [26]. Suwandi et al. [27] present INS/GPS integrated system using a graded Kalman filter using filter processing for INS and Pythagorean processing for GPS sensor. Chiang et al. [11] presents a tightly coupled EKF scheme which detects irregular GPS measurements and uses partial GPS signals to continue correcting INS readings in case of GPS blockage issues. The subject of the research was a ground vehicle, so non-holonomic constraints were also employed.

In situations, involving GPS signal blockage, fusion of INS and VO solutions has been a widely popular alternative. In literature, this type of integrated system is referred to as Visual-Inertial Navigation System (VINS) or Visual-Inertial Odometry (VIO) and includes several different methodologies. Randeniya et al. [28] presented an integrated INS with a single camera to provide the pose of a car driving on a road. In this approach, a decentralized Kalman Filter is used to integrate the position estimates from monocular VO (MVO) and INS. Tardif et al. [14] estimated the 6 DOF pose information of a ground vehicle in agricultural setting by using VIO. In this research, a delayed state loosely coupled EKF was used to combine the Stereo Visual Odometry (SVO) output and the INS sensor data to overcome the uncertainties of both sensors. Nutzi et al. [29] presented an algorithm for estimating scale to overcome the MVO scale ambiguity issue by combining the output with INS. A spline fitting algorithm and a multi-rate EKF algorithm were explored and compared for the scale estimation problem. Peretroukhin et al. [30] presents a neural network (NN) based approach for VINS. In this research, a defined dataset is used to train a NN model for estimating the VO estimate uncertainty for the covariance matrix. Bloesch et al. [31] uses a tightly coupled iterative extended Kalman filter (IEKF) to fuse vision and inertial data. In this methodology, the image is divided into patches, and the keypoints are tracked for all the pixel in the patches. These are used to compute the photometric error which is directly used in the innovation vector of the EKF. Martinelli [32] derives a close-form solution for fusing vision and INS data to obtain observability modes.

The third phase of the research is concerned with INS/VO/GPS fusion. The usage of INS, GPS, VO fusion in order to obtain a robust navigation system of position, velocity, and attitude is an open area of research. Very few papers are available in literature, and most of these are from the last 4 years. Shen et al. [33] uses Double Fuzzy Kalman Filter to combine INS/GPS/VO to compute heading of a tractor. The Ackermann's algorithm is employed as system model and heading measurements of INS, GPS, VO are combined using a single Kalman filter. The first

fuzzy adaptation of Kalman gain and sensor noise covariance matrix was done by referencing dead reckoning algorithm, and the second fuzzy adaptation was of system noise covariance matrix which used innovation vector information. Yue et al. [34] presented INS/VO/GPS integrated system using two local filters and a third master filter. The state covariance and noise covariance matrices for the master filter were updated using fuzzy adaptive feedback. The local filters used Multi-State Constraint Kalman Filter (MSCKF) for INS/VO and Extended Kalman Filter (EKF) for INS/GPS. The master filter utilizes Fuzzy Kalman Filter (FKF). Sun et al. [35] used sequential EKF to develop INS/VO/GPS integrated system with fusion of GPS sensor and camera for the first filter and the output was fused with INS using a second filter. They employed non-holonomic constraints to restrict velocity in body frame to a single dimension. The VO error model was computed using linear regression based on number of inliers. Song et al. [36] presented a Stereo/IMU/GPS/Barometric fusion for micro-aerial vehicles. A tightly coupled IMU and VO with stochastic cloning where a delayed VO state is appended to the state. The result is first updated using GPS, and then using barometric sensor sequentially using EKF. The literature presented in this section shows that INS/VO/GPS is not a fully discovered research area, and there is room for innovation and improvement.

1.5 Thesis Structure

The rest of the thesis is structured as follows: Chapter 2 presents details of the real-world dataset which provides camera, INS, and GPS sensor data. Chapter 3 presents the coordinate frames and their transformations. Chapter 4 presents the VO background and proposed modifications for the Modified SVO (ModSVO) algorithm. Chapter 5 presents the multi-sensor fusion algorithms for INS/VO, INS/GPS, and INS/VO/GPS systems, as well as their implementation on real world dataset and comparisons in case of sensor failure. Chapter 6 presents the possible future work pertaining to this research. Chapter 7 presents a conclusive summary of the thesis.

Chapter 2: Real World Dataset

2.1 Introduction

The objective of this research is to develop a navigation solution algorithm using sensor fusion of INS, Camera, and GPS. While the algorithm may be developed and tested using simulated data, it is not enough to validate the results of the algorithm. Therefore, it is required to test the algorithm on a real-world dataset. In real-world scenario there are unexpected noises and perturbations which lead to an environment where the robustness of the navigation system can be fully tested. Field tests conducted in the past by researchers have paved way to publicly available datasets. Since this research involves visual odometry, Inertial Navigation System (INS), and Global Positioning System (GPS), therefore the dataset required to test the algorithm presented should fulfil the following criteria:

1. The dataset should be of a ground vehicle.
2. The dataset should include stereo camera output.
3. The dataset should include raw accelerometer and gyroscope output.
4. The dataset should include raw GPS output.
5. The dataset should include ground truth data for reference.

There are several datasets that closely follow these requirements such as [37]–[39], however, for this research the KITTI dataset [26] is used to test the algorithm in real world environment. The KITTI dataset is a widely used experimental platform for researched involving VO, INS/VO, and INS/GPS integrated system [40]–[57].

2.2 KITTI Dataset

The KITTI (Karlsruhe Institute of Technology and Toyota Technological Institute) dataset [58] is a publicly available dataset for a car driving in the city of Karlsruhe. The dataset provides

data from multiple sensors in different driving scenarios like residential, highway, city etc., divided into 10 sequences. This research utilizes the sequence 10 for the development and testing of the algorithm. The sequence 10 drive was carried out on 30-09-2011. The length of sequence is approximately 2 minutes, and it contains about 1200 images.

The sensors mounted on the car (Volkswagen Passat B6) include a GPS/IMU Inertial Navigation System, a LiDAR, 2 grayscale cameras, and 2 color cameras. The dataset includes the raw output of these sensors, as well as ground truth data computed by fusing GPS/IMU and LiDAR output. Since the focus of this research is on using IMU, Camera, and GPS, therefore, only the relevant sensors are discussed further.

2.2.1 Camera

The KITTI dataset uses two grayscale cameras of 1.4 Megapixels quality. The model of the cameras is Point Grey Flea 2 (FL2-14S3M-C). The cameras are mounted on the roof of the car using a stereo camera rig, and the displacement between the cameras and the other sensors is provided in [58]. The baseline of the stereo cameras is approximately $54\ cm$. The frequency of camera image outputs is $10\ Hz$. The camera coordinate frame and transformations are presented in chapter 3. The relative position of the cameras is shown in Figure 2-1. The camera calibration information is given, as well as the provided stereo image pairs have been rectified. Therefore, in this research, rectification of stereo images is not visited rather the focus has been shifted towards development of an improved VO algorithm as well as development of a robust sensor fusion algorithm for navigation solution. The camera calibration parameters are:

$$s_u = s_v = 7.070912 \times 10^2$$
$$u_0 = 6.018873 \times 10^2$$
$$v_0 = 1.831104 \times 10^2$$
$$f_u = f_v = 1$$
$$\gamma = 0$$

Where, u is the coordinate value in horizontal direction, v is the coordinate value in vertical direction, f is the focal length, γ is the skew factor, s is the scale factor, and (u_0, v_0) are the coordinates of the principal point.

2.2.2 INS/GNSS Sensors

The KITTI dataset uses an OXTS RT 3003 GPS/IMU sensor. The OXTS device is mounted in the backseat of the car. The specification details of the sensor model are provided in the sensor manual [59]. An overview of the specifications is provided in Table 2-1.

Table 2-1 OXTS RT3003 GPS/IMU Sensor Specifications

IMU Update rate	$100\ Hz$
Gyro Bias Stability	$2\ {}^{\circ}/_{hr}$
Accelerometer Bias Stability	$2\ \mu g$
Gyro Random Walk	$0.2\ {}^{\circ}/_{\sqrt{hr}}$
Accelerometer Random Walk	$0.005\ \frac{m}{s}/_{\sqrt{hr}}$
GPS Update rate	$100\ Hz$
GPS Velocity Accuracy	$0.05\ {}^{km}/_{hr}$
GPS Position Accuracy	$0.01\ m$

The IMU accelerometer and gyroscope data is available in raw form with time stamps. The KITTI dataset does not provide GPS data on its own, rather the GPS/IMU fused data is provided. However, a low-cost GPS can be simulated by down sampling the ground truth signal and adding noise to it. This methodology has been used in [34] and in [60] to simulate the GPS signal data from KITTI dataset. For this research, a white gaussian noise with $\mu = 2\ m$ (where μ is the mean) is added to all GPS position channels, and a noise of $0.5\ {}^{m}/_{s}$ is added to all GPS velocity channels in order to simulate the GPS signal. Furthermore, the GPS update rate has been down sampled from $100\ Hz$ to $10\ Hz$.

2.2.3 Sensor Layout

The sensor layout of all the sensors mounted on the KITTI car are given presented in Figure 2-1. This figure provides the relative positions of the cameras and the GPS/IMU sensor. This information aids in transformation of coordinate frames when computing navigation solutions.

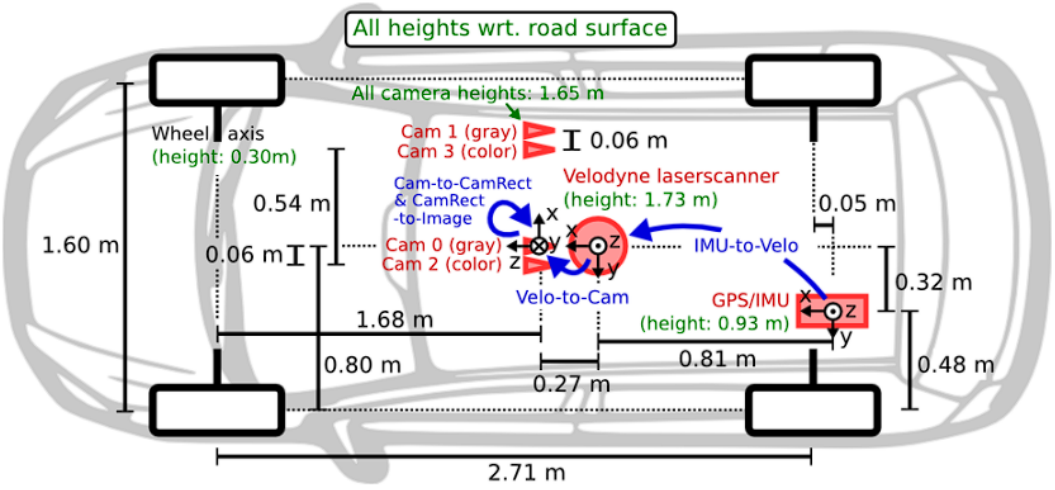

Figure 2-1 KITTI sensor layout [58]

2.3 KITTI Sequence 10 Trajectory

As mentioned in section 2.2, the KITTI sequence 10 is used in development and testing of the navigation solution algorithm presented in this research. The following figures from Figure 2-2 till Figure 2-5 show the top-view trajectory of the driving sequence, as well as the position, velocity, and orientation in x, y, z axes. Figure 2-6 shows sample image output from the cameras. The term 'gt' in legend refers to the ground truth data.

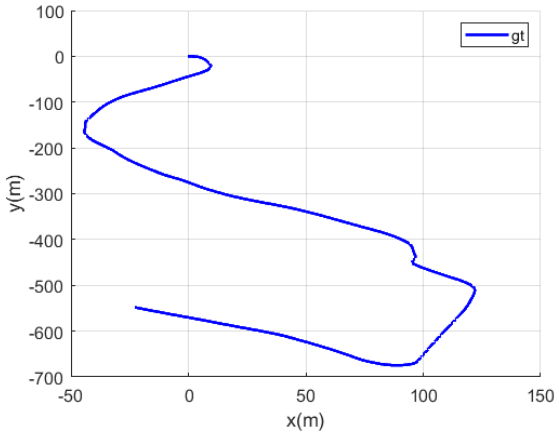

Figure 2-2 Trajectory plot of KITTI data seq 10

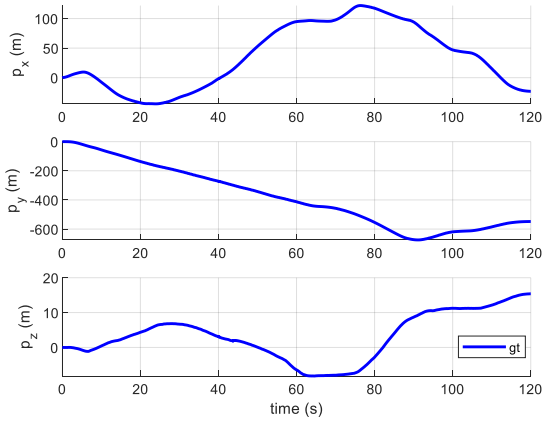

Figure 2-3 Position plot of KITTI data seq 10

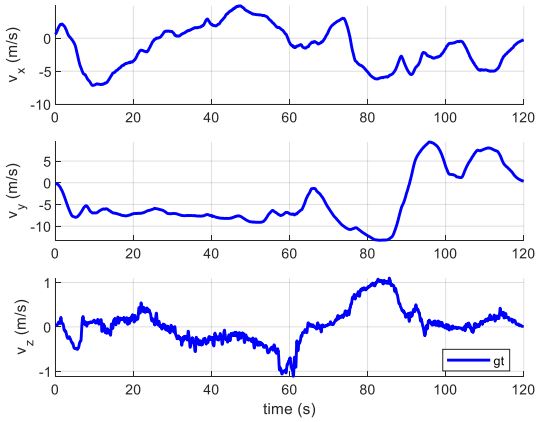

Figure 2-4 Velocity plot of KITTI data seq 10

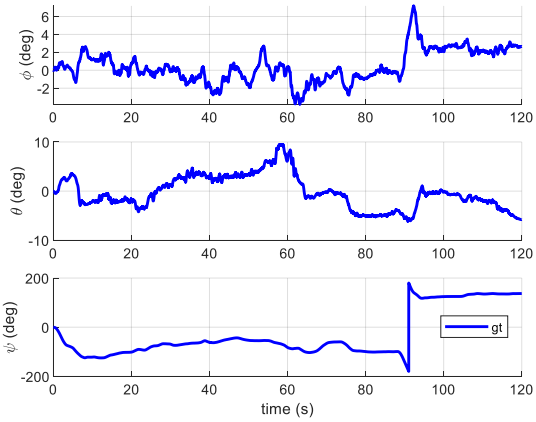

Figure 2-5 Orientation plot of KITTI data seq 10

Figure 2-6 Sample images from left grayscale camera of seq 10

Chapter 3: Coordinate Frames

3.1 Introduction

A coordinate frame of reference is a set of three orthogonal axis whose origin and orientation are defined with respect to an external reference [1]. The navigational sensors output data in their specified coordinate frame system. An important aspect of multi-sensor fusion is to transform all sensor output to a single coordinate frame, which is known as the local navigation frame. In this chapter the coordinate frames for all relevant sensors are defined.

3.2 Local Navigation Coordinate Frame ENU (East-North-Up)

The local navigation coordinate frame is used to present the navigation solution. Its origin is defined at the origin of the vehicle at time $t = 0$. The local navigation frame used in this thesis is the East-North-Up (ENU) frame which is a linear Cartesian coordinate system whose three orthogonal axes are defined as follows:

1. The x-axis points to towards east.
2. The y-axis points towards north.
3. The z-axis points towards up, perpendicular to the x-y plane.

The navigation solutions from IMU, GPS, and camera, are transformed into the local navigation coordinate frame before sensor fusion computation. It should be noted that the local navigation coordinate frame uses 'flat Earth' assumption. This is a reasonable assumption for short distances such as the driving scenarios considered in this research. However, in case of navigation solution over long distances through aircrafts this assumption does not remain valid.

3.3 Earth Centered Earth Fixed Coordinate Frame

The Earth Centered Earth Fixed Coordinate Frame (ECEF) is defined as a frame of reference whose origin is fixed to the center of the Earth [1]. The ECEF frame rotates along with the

Earth hence, the relative motion between the Earth and ECEF is zero. The ECEF is a reference frame for geographic Cartesian coordinate system and the geographic spherical coordinate system. It is important to address the ECEF coordinate frame because the output from GPS is given in terms of latitude, longitude, and altitude which are geodetic coordinates provided in the ECEF frame. This output is transformed into the ENU frame (n) in order to compute the navigation solution using the Mercator projection [61].

The three orthogonal axes of ECEF frame are defined as follows:

1. The x-axis points towards the intersection of the equator and prime meridian. This defines $0°$ latitude and $0°$ longitude.
2. The z-axis points towards the north pole of Earth.
3. The y-axis points towards $90°$ longitude in the equatorial plane.

The ECEF frame is depicted in Figure 3-1.

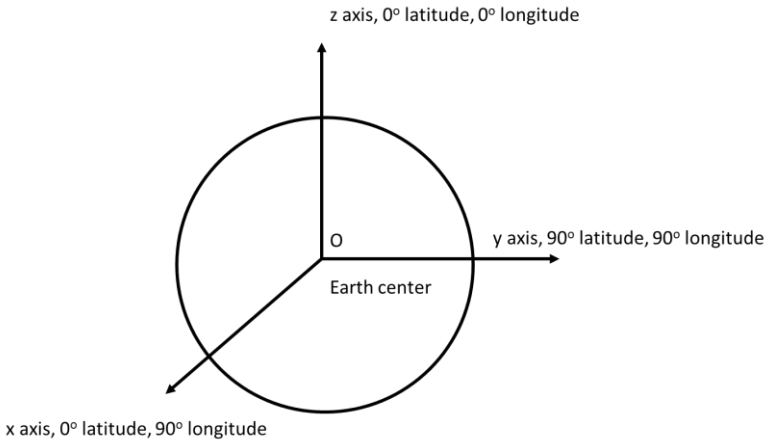

Figure 3-1 ECEF frame

3.4 Camera Frame

The visual odometry (VO) phase of the navigation solution uses images from stereo cameras mounted on the roof of the car, hence the VO output is provided in the Camera Frame

represented by c, which is converted to the navigation frame n for computation of navigation solution. The origin of camera coordinate frame is at the focal point.

The three orthogonal axes of camera frame are as follows [26]:

- The z-axis of the camera frame points forward, out of the camera lens.
- The x-axis points towards the right of the camera.
- The y-axis points down perpendicular to the x-z plane.

The clockwise rotation about the axes is considered positive. Roll (ϕ) is defined as the rotation about z-axis, pitch (θ) is defined as the rotation about x-axis, and yaw (ψ) is defined as the rotation about the y-axis. The camera coordinate frame is depicted in Figure 3-2.

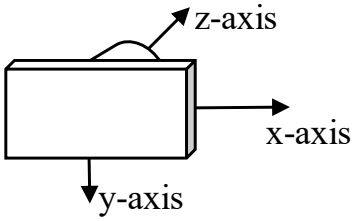

Figure 3-2 Camera coordinate frame depiction

3.5 IMU Frame

The IMU sensor outputs are given in the IMU frame i, which need to be transformed to the navigation frame (n) in order to compute the navigation solution. The IMU sensor is installed in the vehicle such that the three orthogonal axes of IMU frame are defined as follows:

1. The x-axis points towards the front of the car. It is positive for forward motion.
2. The y-axis points towards the left of the car. It is positive for leftward motion
3. The z-axis points down perpendicular to the x-y plane.

The clockwise rotation about the axes is considered positive. Roll (ϕ) is defined as the rotation about x-axis, pitch (θ) is defined as the rotation about y-axis, and yaw (ψ) is defined as the rotation about the z-axis. The IMU coordinate frame for the OXTS RT 3003 sensor presented in the sensor manual [59] and is shown in Figure 3-3. This defines the placement of the sensor in the test car.

Figure 3-3 IMU coordinate frame displayed on OXTS RT 3003 [59]

3.6 Coordinate Frame Transformations

In case of multi-sensor fusion, coordinate frame transformation is an essential step. This research uses camera, IMU, and GPS sensors each of which give output in their own coordinate frame. In order to fuse them to form a navigation solution, they must be converted to the navigation frame.

A vector defined in one coordinate frame can be transformed to another coordinate frame using a transformation matrix. In this text, a 3-dimensional vector x defined in coordinate frame a is represented by x_a. Similarly, the same vector x defined in coordinate frame b will be represented by x_b. The transformation from x_b to x_a is represented by equation (3.1).

$$x_a = R_b^a x_b + t_a \tag{3.1}$$

Where R_b^a is the rotation matrix defining rotation between b and a, and t_a represents the translation vector defining translation between the origin of b and a.

3.6.1 IMU Frame to Local Navigation Frame

The conversion of IMU frame to the navigation frame is given as follows [62]:

$$R_i^n = \begin{bmatrix} cos\theta cos\psi & -cos\phi sin\psi + sin\phi sin\theta cos\psi & sin\phi sin\psi + cos\phi sin\theta cos\psi \\ cos\theta sin\psi & cos\phi cos\psi + sin\phi sin\theta sin\psi & -sin\phi cos\psi + cos\phi sin\theta sin\psi \\ -sin\theta & sin\phi cos\theta & cos\phi cos\theta \end{bmatrix} \quad (3.2)$$

Where roll (ϕ), pitch (θ), yaw (ψ) represent the orientation of IMU frame about x, y, z axis, respectively.

3.6.2 Camera Frame to Local Navigation Frame

The camera frame is first converted to the IMU frame and then converted to navigation frame. The conversion from camera to IMU is constant and is given as follows [26]:

$$R_c^i = \begin{bmatrix} 0.007813 & -0.0042792 & 0.99996 \\ -0.99986 & -0.014868 & 0.0077486 \\ 0.014834 & -0.99988 & -0.0043948 \end{bmatrix}$$

The conversion from camera to navigation frame is given as:

$$R_c^n = R_i^n R_c^i \quad (3.3)$$

The relative position and orientation of the camera frame with respect to IMU frame is shown in Figure 3-4.

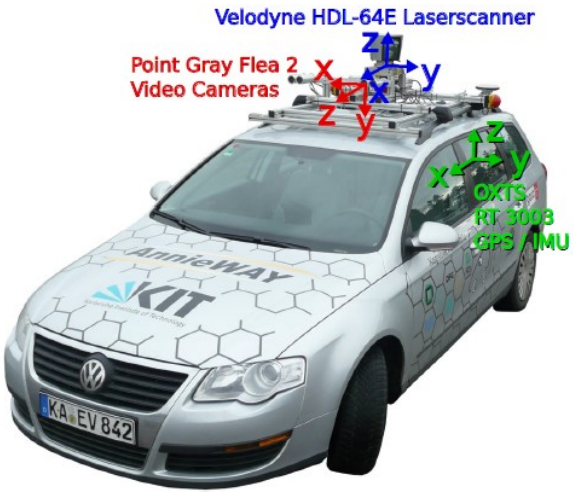

Figure 3-4 Relative position and orientation of camera frame (red) and IMU frame (green) [26]

Chapter 4: Visual Odometry

4.1 Introduction

This chapter deals with the development of Visual Odometry (VO) algorithm as a standalone navigation solution for a ground vehicle. Visual Odometry comprises of estimating pose of a moving vehicle using sequential images from a camera [12]. The VO problem can be presented as ego-motion estimation over time. Although, there are a number of VO techniques available, the most commonly used are: Stereo Visual Odometry (SVO) and Monocular Visual Odometry (MVO).

In SVO, there are two cameras mounted on the vehicle platform. The cameras are mounted such that they are separated by a fixed distance in one dimension. This distance is known as the *baseline*. In MVO only a single camera is mounted on the vehicle platform. While both SVO and MVO use feature-based approach to estimate motion, SVO utilizes the stereo images, along with the information of baseline, to compute 3D locations of the interest points and solve the Perspective-3-Point (P3P) problem in order to estimate pose, whereas MVO utilizes the epipolar constraints between consecutive images in order to compute the essential matrix which is decomposed to estimate the pose of the vehicle. Due to lack of a world coordinate scale measure (like the baseline in SVO) MVO suffers from scale estimation problem i.e., the translation of the vehicle can only be obtained up to an unknown scale factor.

As a standalone VO solution, the most commonly used technique is SVO as it provides higher accuracy in pose estimation due to absolute translation information, even though it requires higher computational processing requirement. In this research, a visual odometry approach is presented called *Modified Stereo Visual Odometry* (ModSVO). In ModSVO, the attitude of the camera is computed using the essential matrix decomposition (similar to MVO), and the translation is computed by minimizing the reprojection error (similar to SVO). This approach

has shown an improvement in accuracy as well as reduced computational requirement in comparison to the traditional SVO.

The rest of the chapter is structured as follows. Section 4.2 will present the necessary background information pertaining to feature based VO algorithms. This includes the camera intrinsic and extrinsic parameters, feature detection and matching algorithm, essential matrix estimation algorithm, 3D point location estimation, pose estimation, and pose optimization algorithm. Section 4.3 presents the algorithm of SVO, and section 4.4 presents the ModSVO algorithm. Finally, section 4.5 presents the comparison of ModSVO and SVO by implementing the algorithms on the real-world dataset.

4.2 Visual Odometry Preliminaries

4.2.1 Intrinsic Parameters

The intrinsic parameters of a camera include the focal length, scaling factor, principal point coordinates, and skew factor [63]. These parameters allow us to transform metric coordinates in the camera coordinate frame to pixel coordinates in the normalized camera frame. The normalized camera frame has a unit focal length from the optical center, and the origin is situated at the top left corner of the frame. This means that the pixel x coordinate increases towards the right and the pixel y coordinate increases towards the left. The intrinsic parameters are encapsulated in the intrinsic matrix K, such that:

$$K = \begin{bmatrix} s_u f_u & \gamma & u_0 \\ 0 & s_v f_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.1}$$

Where, with the origin at the top left corner of the image, u is the coordinate value in horizontal direction, v is the coordinate value in vertical direction, f is the focal length, γ is the skew factor, s is the scale factor, (u_0, v_0) are the coordinates of the principal point, and $K \in \mathbb{R}^{3\times3}$ is the camera intrinsic matrix.

4.2.2 Extrinsic Parameters

The extrinsic parameters of a camera define its position and orientation with respect to a reference coordinate frame. The reference frame is usually the 3D world coordinate frame. However, in case of visual odometry, origin of the left camera coordinate frame is considered the origin of the world coordinate frame. Therefore, for the first frame, the left camera coordinate frame and the world coordinate frame coincide. In matrix form, the extrinsic matrix will be represented as follows:

$$\mathcal{E} = [R \quad t] \tag{4.2}$$

Where, $\mathcal{E} \in \mathbb{R}^{3\times4}$ is the camera extrinsic matrix, $R \in \mathbb{R}^{3\times3}$ is the rotation matrix from world frame to camera frame, $t \in \mathbb{R}^{3\times1}$ is the translation vector from world frame to camera frame.

4.2.3 Camera Matrix

The camera matrix combines the intrinsic and extrinsic matrices of a camera and defines the relationship of an interest point in 3D world frame to its pixel coordinate in the camera frame. The camera matrix is defined as:

$$p = PX \tag{4.3}$$

Where, P is the camera matrix, $p \in \mathbb{R}^{3\times1}$is the homogeneous pixel coordinate of a point in the camera frame, $X \in \mathbb{R}^{4\times1}$ is the homogeneous 3D coordinate of the point in the world frame.

As stated above, the camera matrix holds the information of extrinsic and intrinsic parameters. Therefore, the camera matrix can be decomposed to obtain the intrinsic matrix, rotation matrix, and origin vector. This is shown in the following equation.

$$P = K[R \quad -Ro] \tag{4.4}$$

Where, K is the intrinsic matrix, R is the rotation matrix from world frame to camera frame o is origin of camera frame in world coordinates

4.2.4 Feature Detection and Matching

Interest points detection and matching is the crux of visual odometry. This process is used to identify keypoints in one image which can be detected and then matched with the keypoints in the other image. There are several methods for computing interest points in the literature. Each method has its merits and demerits over the other methods. For this research three methods were considered, Harris detector, Scale Invariant Feature Transform (SIFT), and Speeded Up Robust Feature (SURF). The Harris Detector [16] algorithm uses determinant of Hessian matrix of intensities in a defined region to identify corners, edges, and flat regions. The main limitation of this algorithm is that the resulting corner points are sensitive to scale and orientations changes. In order to overcome this limitation, the Scale Invariant Feature Transform (SIFT) was introduced in [17]. This algorithm detects keypoints based on the Hessian matrix at different scales by applying Difference of Gaussian (DoG). The algorithm also introduced orientation assignment to the keypoints. However, the keypoint descriptor vector consisted of 128 elements which results in a long time when computing matches. In order to improve this, Speeded Up Robust Feature (SURF) was introduced in [18]. The SURF method uses blob detection over multiple scales to identify a scale invariant interest point. However, instead of DoG the SURF algorithm uses box filters. Different scales are computed by simply up scaling the box filters and convolving them with the image. This speeds up the processing, and also allows for parallel processing of different scales. Furthermore, the descriptor vector is defined with 64 elements, thereby increasing the speed of matching the interest points. This research uses SURF to detect and match interest points for the visual odometry phase.

Speeded Up Robust Features

The interest point detection and matching in SURF is done in four steps:

1. Locate interest point.
2. Compute orientation of interest point.
3. Compute feature descriptor vector for the interest point.
4. Match the interest points between images using the feature descriptor vector.

The algorithm is presented in detail in [18]. A general overview is presented below.

Interest Point Detection

The interest point detection is carried out by computing the approximate determinant of image hessian matrix at different scales and using it to identify scale invariant blobs. The first step is to compute the integral image. This allows for calculation of sum of intensities of any region of the image with only four calculations, regardless of the size. For an image matrix I, the integral image $\mathcal{I}$ is computed as follows:

$$\mathcal{I}(x,y) = \sum_{i=0}^{i\leq x}\sum_{j=0}^{j\leq y} I(i,j)\,, \qquad \forall\ 0 \leq x \leq m, 0 \leq y \leq n \tag{4.5}$$

Where, (i,j) are the pixel coordinates of the image frame, and (x,y) are pixel coordinates of the integral image.

The next step is to compute the approximate determinant of the image Hessian matrix, H. This is done by convolving the integral image with box filters to compute second order Gaussian partial derivatives in x, y, and xy directions, denoted by G_x, G_y, G_{xy} respectively.

$$\mathrm{Det}\,H(\sigma) = G_{xx}G_{yy} - \left(wG_{xy}\right)^2 \tag{4.6}$$

where, $w = 0.9$ is the approximation weight, σ is the scale level based on the box filter size.

The value of the determinant is the intensity of the blob at this point in the image. In order to make the interest point scale invariant, the blob response must be detected at multiple scale levels in the scale space. This is done by increasing the size of the box filter. The size is increased by 6 pixels at the first octave (from 9×9 to 15×15 to 21×21), and then for each next octave the number of pixels by which the size of box filter should increase is doubled. For example, in the second octave the size of box filter will be increased by 12 pixels (from 15×15 to 27×27 and so on). The octaves are increased until the box filter is the size of the image.

The blob response for each scale level is computed using the determinant of the Hessian matrix.

After obtaining the blob response map at all scale levels, the maxima are computed by applying non-maximum suppression in $3 \times 3 \times 3$ neighborhood, where the third dimension is across the scale space. This defines the interest points.

Interest Point Orientation

In order to achieve robustness to rotation, the interest points are given an orientation. The first step to compute the orientation is to convolve the integral image with Haar-wavelet filters, in x and y direction, in the 2D neighborhood of radius 6σ. Then the sum of the response is computed in x and y direction for every $60°$ arc in the defined circular region, and the resultant vector is found. The orientation of the vector with the greatest magnitude will define the orientation of the interest point.

Interest Point Description

The region of interest needs to be represented in a form with which it can be compared with regions of interest of the other image. This form is known as the interest point descriptor, which is a 64-element vector. In order to define the interest point descriptor, a square region of size

20σ is form around the interest point with the same orientation as the obtained in the previous step. The region is split into 4×4 sub regions. Haar-wavelet filter is convolved over these sub regions in x and y directions in order to obtain the response h_x and h_y respectively. For each sub region the four-element descriptor vector can be computed as follows,

$$v = \left[\sum h_x \quad \sum h_y \quad \sum |h_x| \quad \sum |h_y|\right] \tag{4.7}$$

This vector is combined through concatenation with the descriptor vectors of other subregions of the interest point. As a result, a 64-element descriptor vector is defined for the interest point.

Interest Point Matching

In order to find the matching interest points between two images, the SURF descriptor vectors of interest points must be compared for both images. The Sum of Squared Differences (SSD) is used to compute a metric for difference between the two interest points. A threshold for the metric is defined to identify the matched interest points. Two descriptor vectors v and v' can be compared using equation (4.8).

$$\mathcal{M}_{SSD} = \sum_{i=1}^{i=64} (v_i - v_i')^2, \qquad v_i \in v, \qquad v_i' \in v' \tag{4.8}$$

Additionally, the sign of Laplacian is compared to differentiate between bright and dim blobs.

SURF Algorithm

The overall SURF algorithm for feature detection and matching is provided in the table below.

SURF Algorithm
1. Load image matrix: $I \in \mathbb{R}^{m \times n}$ 2. Compute integral image: $$\mathcal{I}(x, y) = \sum_{i=0}^{i \le x} \sum_{j=0}^{j \le y} I(i, j)$$

3. Compute second-order Gaussian partial derivatives in x, y, xy directions by convolving the image with box filters: G_x, G_y, G_{xy}
4. Compute blob response by approximate determinant of Hessian matrix.

$$\mathrm{Det}\, H(\sigma) = G_{xx}G_{yy} - \left(wG_{xy}\right)^2$$

5. Compute scale space (σ) octaves by upscaling the box filters.
6. Compute blob response for all scale levels.
7. Identify keypoint location and scale by performing 3D non-maximum suppression in the $3 \times 3 \times 3$ neighborhood of blob responses.
8. Convolve the integral image with Haar-wavelet filters, in x and y direction, in the 2D neighborhood of radius 6σ.
9. Compute sum of response in x and y direction for every $60°$ arc in the defined circular region.
10. Define interest point orientation as the orientation of vector with the highest magnitude.
11. Define a window of 20σ around the interest point with the orientation obtained and split it into 4×4 subregions.
12. Compute Haar-wavelet response in the subregions in x and y directions. h_x, h_y
13. Define descriptor vector for each subregion:

$$v = \left[\sum h_x \quad \sum h_y \quad \sum |h_x| \quad \sum |h_y|\right]$$

14. Denote the location of key features: $\hat{f}$.
15. Compare descriptor vectors using SSD (Sum of Squared Differences)

$$\mathcal{M}_{SSD} = \sum_{i=1}^{i=64} (v_i - v_i')^2$$

16. Denote the location of matching features: f.

Table 4-1 SURF feature detection and matching

Implementation Results

The interest point detection based on SURF algorithm was implemented, and the results are shown in Figure 4-1 and Figure 4-2. The images are the left and right stereo images from KITTI sequence 10. This yellow marks show the pixel location of interest point, and the descriptor vector contains information of the blob region surrounding that point.

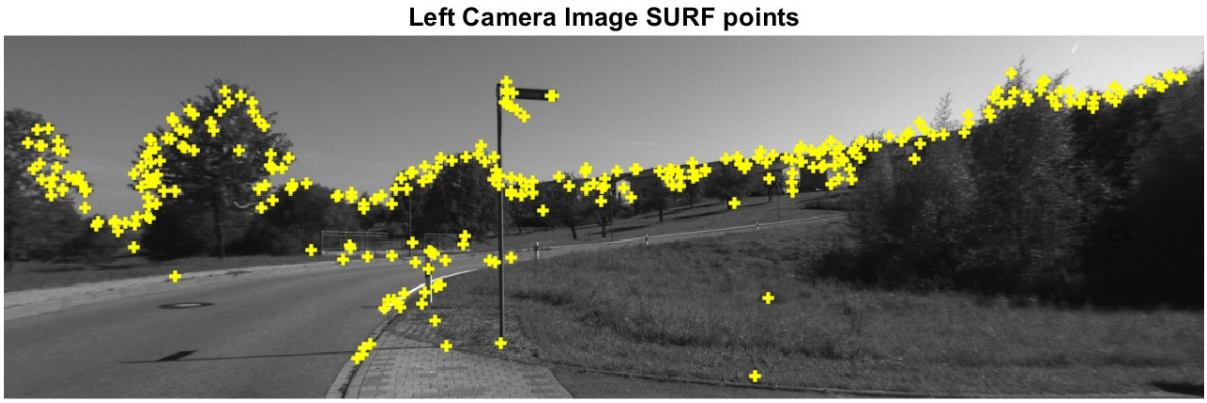

Figure 4-1 SURF interest point response using the left stereo camera image from KITTI sequence 10.

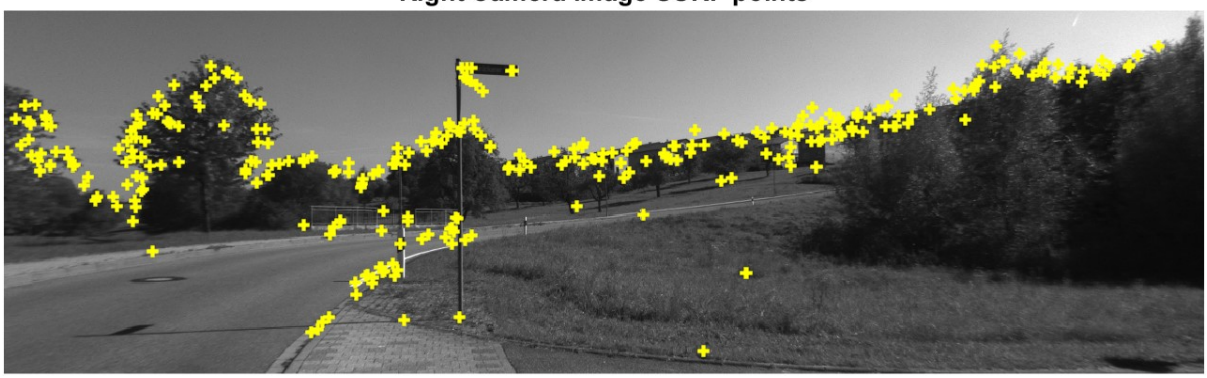

Figure 4-2 SURF interest point response using the right stereo camera image from KITTI sequence 10.

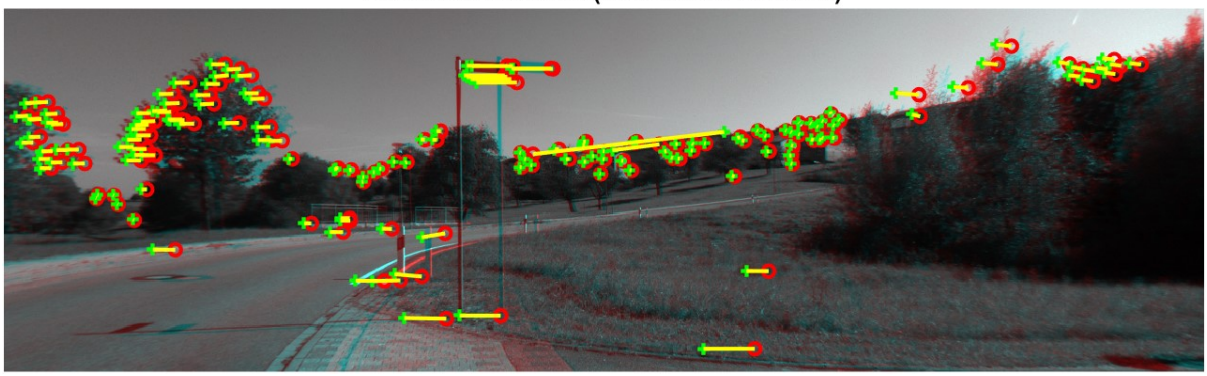

Figure 4-3 Matched features between left and right stereo images.

Interest point matching is conducted between the two images and the result is shown in Figure 4-3. It can be seen that there are outliers. This is expected because the Hessian approximation

reduces repeatability of interest point detection. In order to filter out the outliers, the epipolar constraints must be applied. This is done by estimating Essential matrix through RANSAC (Random Sample and Consensus) [64]. The details are presented in section 4.2.5.

4.2.5 Essential Matrix

Point correspondences in stereo images are constrained through epipolar lines. Observing the epipolar constraints, a homogenous point vector in camera coordinate frame of the left camera can be transformed to the camera frame of the right camera. The rotation and translation of this transformation along with the epipolar constraints are embodied in the Essential Matrix, E [63].

The relationship between the essential matrix and corresponding stereo points is given as,

$$p_r^T E p_l = 0 \tag{4.9}$$

where, E is the essential matrix, l is the left camera, r is the right camera, $p = [u \quad v \quad 1]^T$ is the homogeneous pixel coordinate vector in the camera frame, and (u, v) is the 2D pixel coordinate of the point in camera frame.

The essential matrix equation in (4.9) states that given an interest point in the image plane of left image, an epipolar line can be constructed in the image plane of right camera. The corresponding interest point of the right image will lie on the epipolar line. This is an important constraint which allows us to find and remove outliers when matching features between two images.

5-point Algorithm

The essential matrix is computed using the 5-point algorithm presented by Nister et al. [13]. The details of the 5-point algorithm are presented in the reference, and a general overview is provided below.

The 5-point algorithm estimates the essential matrix E such that the equation (4.9) is satisfied for 5 matching points between the left and the right camera images. This is done by rearranging the equation into the following form and computing the null vector of matrix A.

$$A\mathcal{E} = 0 \tag{4.10}$$

Where, $A \in \mathbb{R}^{5\times 9}$ represents a correspondence matrix containing the pixel coordinates of 5 corresponding interest points, and $\mathcal{E} \in \mathbb{R}^{9\times 1}$ represents a nine-element vector which can be rearranged to form the 3×3 essential matrix.

The 5 matching interest point pixel coordinates for left and right camera images, obtained using SURF detection and matching algorithm, are first converted to homogeneous coordinates by appending a 1 to the pixel coordinate vector. This is shown in the following equation

$$p = [u \quad v \quad 1] \tag{4.11}$$

Where, p is the homogeneous pixel coordinate vector, and (u, v) are the pixel coordinates of the interest point.

The points are then arranged in a matrix form defined by the following equation.

$$A = \begin{bmatrix} u_l^i u_r^i & u_l^i v_r^i & u_l^i & v_l^i u_r^i & v_l^i v_r^i & v_l^i & u_r^i & v_r^i & 1 \\ \vdots & & & \vdots & \vdots & & & & \vdots \\ u_l^n u_r^n & u_l^n v_r^n & u_l^n & v_l^n u_r^n & v_l^n v_r^n & v_l^n & u_r^n & v_r^n & 1 \end{bmatrix} \tag{4.12}$$

Where, A is the correspondence matrix, $p_r^i = [u_r \quad v_r \quad 1]$ is the homogenous pixel coordinate vector of the i^{th} interest point for right camera, $x_l^i = [u_l \quad v_l \quad 1]$ represents the same for the left camera, and $i = 1,2, \ldots n = 5$ is the index of the of interest point.

The vector $\mathcal{E}$ in (4.10) is the null vector of the matrix A in (4.12). Therefore, $\mathcal{E}$ can be estimated by applying the Singular Value Decomposition (SVD) algorithm on A. The

derivation and details of implementation of SVD are presented in [65] and are not revisited in this text. The following equation represents the output after applying SVD.

$$A = U\Sigma V^T \tag{4.13}$$

Where, U is an orthogonal matrix consisting of left singular vectors, Σ is a diagonal singular matrix with consisting of eigen values of $A^T A$ as diagonal elements, and V is an orthogonal matrix consisting of right singular vectors.

The last column V represents the null vector of A. Therefore, $\mathcal{E}$ is obtained as the following equation.

$$\mathcal{E} = [v_1 \quad v_2 \quad v_3 \quad v_4 \quad v_5 \quad v_6 \quad v_7 \quad v_8 \quad v_9]^T \tag{4.14}$$

Where, v_i are the elements of the last column of V.

The vector $\mathcal{E}$ is then rearranged to obtain the essential matrix. This is shown in the following equation.

$$E = \begin{bmatrix} v_1 & v_2 & v_3 \\ v_4 & v_5 & v_6 \\ v_7 & v_8 & v_9 \end{bmatrix} \tag{4.15}$$

The following algorithm gives an overview of the 5-point algorithm used to compute the Essential Matrix.

5-point Algorithm
1. Define matrix A such that: $$A = \begin{bmatrix} u_l^i p_r^i & v_l^i p_r^i & p_r^i \\ \vdots & \vdots & \vdots \\ u_l^n p_r^n & v_l^n p_r^n & p_r^n \end{bmatrix}$$ $i = 1,2, \ldots ,5$ is the index of the corresponding points

2. Use Singular Value Decomposition (SVD) to compute the null vector $\mathcal{E}$ of A

$$A = U\Sigma V^T$$

3. Define $\mathcal{E}$ as the last (right most) column vector of V

$$\mathcal{E} = [v_1 \quad v_2 \quad v_3 \quad v_4 \quad v_5 \quad v_6 \quad v_7 \quad v_8 \quad v_9]^T$$

Where v_i are the elements of the last column vector of V.

4. Rearrange the solution vector $\mathcal{E}$ to obtain the essential matrix:

$$E = \begin{bmatrix} v_1 & v_2 & v_3 \\ v_4 & v_5 & v_6 \\ v_7 & v_8 & v_9 \end{bmatrix}$$

Table 4-2 Algorithm to compute essential matrix from a set of matching points using 5-point scheme

Outlier detection using Random Sample and Consensus (RANSAC)

In section 4.2.4 a set of matching points were computed using SURF detection algorithm. However, in Figure 4-3 it was observed that there were some erroneous matches. These are referred to as outliers. For VO computation, it is necessary to remove the outliers otherwise they will become a source of error and may cause VO failure. Outlier detection can be carried out using the epipolar constraints. The matching points which do not satisfy the essential matrix equation (4.9) within a certain threshold, are deemed as outliers.

Since the essential matrix is computed using a set of 5 matching points, and we obtain around 100-300 corresponding points using SURF detection, a RANSAC (Random Sample and Consensus) algorithm [64] is used to estimate an essential matrix which satisfies the epipolar constraints for the maximum number of matching points. The remaining points are considered outliers. The RANSAC algorithm for computing essential matrix is presented in detail in [66] and an overview of the RANSAC algorithm is given in the following table.

RANSAC Algorithm
1. Define N number of random subsets containing 5 corresponding points from the set of all M corresponding interests. 2. Compute essential matrix E for the N subsets using 5-point algorithm. 3. Check error for the E matrix with all corresponding points using the cost function $p_r^T E p_l = 0$ 4. Identify the E matrix with highest number of corresponding points having error below the defined threshold.

Table 4-3 RANSAC algorithm for estimating essential matrix and detecting outliers

The resulting model from the RANSAC algorithm will be the estimated E matrix, and the corresponding points with error below threshold will be the inlier matching points.

Figure 4-4 shows inlier matched features after applying epipolar constraints through RANSAC. It can be seen that the remaining features are matched accurately with their corresponding features in the other image.

Figure 4-4 Inlier matched features after applying RANSAC

Decomposition of Essential Matrix

The essential matrix combines matching points from both images in terms of epipolar constraints, therefore, it is possible to extract information regarding the rotation and translation between the two camera frames by decomposing the essential matrix [63].

The rotation and translation between camera frames are related to the essential matrix as follows,

$$E = [t]_{\times} R \tag{4.16}$$

Where, $t = [t_x \quad t_y \quad t_z]^T$ is the translation vector, $R \in \mathbb{R}^{3\times3}$ is the rotation matrix, and $[t]_{\times}$ is the skew-symmetric matrix of the vector t defined as follows,

$$[t]_{\times} = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix} \tag{4.17}$$

The rotation matrix and translation vector can be extracted using Singular Value Decomposition algorithm**.** The decomposition algorithm results in four possible solutions. Three solutions are trivial as they do not identify the correct axes orientation of the cameras. The true solution can be identified by computing 3D location of the keypoints and identifying the solution which gives the 3D point location in front of both cameras.

While the rotation matrix estimate is accurate, the translation vector can only be estimated up to an unknown scale. The essential matrix decomposition algorithm is given as follows.

Essential Matrix Decomposition Algorithm
1. Apply SVD to Essential Matrix E: $E = U\Sigma V^T$ 2. Define $S_n = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$ 3. Compute $E_n = US_nV^T$ 4. Apply SVD to E_n: $[U_n, S_n', V_n] = \text{SVD}(E_n)$ 5. Define $W = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

6. Compute possible rotation matrices and translation vector

$$R_a = U_n W V_n^T$$
$$R_b = U_n W^T V_n^T$$
$$t = U_n \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

7. Obtain 4 possible Camera Matrices:

$$P_1 = [R_a \quad t]$$
$$P_2 = [R_b \quad t]$$
$$P_3 = [R_a \quad -t]$$
$$P_4 = [R_b \quad -t]$$

8. Compute 3D Point location of a corresponding interest point for all for solutions using Direct Linear Transformation (DLT). The left and right camera matrices will be defined as follows:

$$P_l = [I^{3\times3} \quad 0^{3\times1}]$$
$$P_r = P_i \text{ for } i = 1, \dots, 4$$

9. Select the solution which has the 3D point located in front of both cameras. This will be the pose rotation and translation estimate: R, t

Table 4-4 Decomposition of Essential matrix algorithm

4.2.6 Compute 3D Point Location

The 3D location of the point in the world coordinate frame can be computed given a set of corresponding inlier points. In this research Direct Linear Transformation (DLT) [63] is used to compute 3D point locations. It should be noted that as the depth of interest point increases, the accuracy of the 3D point location reduces. This is because if the depth is much greater than the baseline then the camera configuration degenerates to monocular case. The algorithm for computing 3D point locations is presented with detail in [63] and a general overview is provided in this text.

The camera matrix presented in section 4.2.3 provides the extrinsic relation between the 3D point coordinate and its pixel coordinate counterpart. This relation is presented in the following equations.

$$p_l = P_l X \tag{4.18}$$

$$p_r = P_r X \tag{4.19}$$

Where, $X \in \mathbb{R}^{4\times1}$ is the homogeneous 3D world coordinate vector of the interest point, P is the camera matrix, p is the homogeneous pixel coordinate vector of the interest point, l and r represent the left and right camera images.

If A matrix is defined as follows,

$$A = \begin{bmatrix} u_l P_l^{\,3} - P_l^1 \\ v_l P_l^3 - P_l^2 \\ u_r P_r^3 - P_r^1 \\ v_r P_r^3 - P_r^2 \end{bmatrix} \tag{4.20}$$

Where, (u, v) are the pixel coordinates of the interest point, P is the camera matrix, P^i is the i^{th} row of the camera matrix, and l, r represent the left and right camera images, respectively.

Then the equations (4.18) and (4.19) can be rearranged in the following form.

$$AX = 0 \tag{4.21}$$

Equation (4.21) states that X is the null vector of A. Therefore, X can be obtained using Singular Value Decomposition (SVD), in a similar fashion as that presented in section 4.2.5. The last column of V is defined as $\mathcal{X}$. In order to recover the homogeneous coordinate vector, all elements of $\mathcal{X}$ are divided by last element of $\mathcal{X}$. The resulting solution is the estimate of X, the 3D location of the interest point.

The overall algorithm for computing the 3D point location is given in the following table.

3D Point Location Algorithm
1. Define A matrix such that: $$A = \begin{bmatrix} u_l {P_l}^3 - P_l^1 \\ v_l P_l^3 - P_l^2 \\ u_r P_r^3 - P_r^1 \\ v_r P_r^3 - P_r^2 \end{bmatrix}$$ 2. Compute Singular Value Decomposition (SVD) of A: $$A = U\Sigma V^T$$ 3. Obtain the null vector $\mathcal{X}$ of A which is the last column of V. $$\mathcal{X} = [v_1 \quad v_2 \quad v_3 \quad v_4]^T$$ Where v_i are the elements of the last column vector of V. 4. Divide all elements of $\mathcal{X}$ with the fourth element of $\mathcal{X}$. $$\widehat{\mathcal{X}} = \frac{\mathcal{X}}{\mathcal{X}_4}$$ 5. The solution is the homogeneous 3D point location in world coordinate frame of the interest point X. $$X = \widehat{\mathcal{X}} = [x \quad y \quad z \quad 1]$$

Table 4-5 3D point location algorithm

4.2.7 Pose Estimation

There are three common pose estimation techniques described in [12], 2D-to-2D pose estimation, 3D-to-2D pose estimation, and 3D-to-3D pose estimation. The 2D-to-2D pose estimation refers to estimating pose by decomposing the essential matrix. This has been explained in section 4.2.5. The 3D-to-3D technique is shown to be less accurate than the 3D-to-2D technique [7], therefore the 3D-to-2D pose estimation technique is used for Stereo VO algorithm in this research.

The 3D-to-2D pose estimation algorithm can be solved using the Perspective-3-Point (P3P) problem. The objective is to estimate a Perspective Projection Matrix (PPM) which will define

the rotation and translation to convert vector in world frame to a vector in camera frame. This can be used to estimate the pose between camera coordinate frames at subsequent time steps. The details of P3P algorithm are given in [12] and a general overview is provided in this thesis.

In the first step, at time step k the left and right camera images I_{l_k} and I_{r_k} are processed to obtain the set of corresponding homogeneous pixel coordinates through SURF matching and detection (Table 4-1), and outlier removal using 5-point algorithm (Table 4-2) with RANSAC (Table 4-3). The resulting vectors are given as $\hat{p}_{l_k}$ and $\hat{p}_{r_k}$.

At time step $k+1$ the left camera image $I_{l_{k+1}}$ is processed to obtain set of matching interest points between the images I_{l_k} and $I_{l_{k+1}}$ given as $\tilde{p}_{l_k}$ and $\hat{p}_{l_{k+1}}$. Intersecting coordinates between $\tilde{p}_{l_k}$ and $\hat{p}_{l_k}$ are found to obtain tracked points p_{l_k}, p_{r_k}, and $p_{l_{k+1}}$. Given at least 3 tracked points, the matrix A is defined as follows.

$$A = \begin{bmatrix} a_i \\ \vdots \\ a_n \end{bmatrix} \tag{4.22}$$

Where a_i is a two-row matrix defined as follows.

$$a_i = \begin{bmatrix} X_i^T & \mathbf{0} & -u_i X_i^T \\ \mathbf{0} & X_i^T & -v_i X_i^T \end{bmatrix} \tag{4.23}$$

Where, $X_i \in \mathbb{R}^{4\times 1}$ is the 3D point location of the i^{th} interest points, $p_{l_{k+1}}^{i} = [u_i, v_i]$ is the pixel coordinates of the corresponding interest points at time step $k+1$

The equation (4.3) can be rearranged in the following form,

$$A\mathcal{T} = 0 \tag{4.24}$$

Where, $\mathcal{T} \in \mathbb{R}^{12\times 1}$ contains the components of the transformation matrix.

Using Singular Value Decomposition (SVD), the null vector of A is computed in order to obtain $\mathcal{T}$. Finally, the elements of $\mathcal{T}$ are rearranged to obtain the transformation matrix T.

$$T = \begin{bmatrix} \mathcal{T}_1 & \mathcal{T}_2 & \mathcal{T}_3 & \mathcal{T}_4 \\ \mathcal{T}_5 & \mathcal{T}_6 & \mathcal{T}_7 & \mathcal{T}_8 \\ \mathcal{T}_9 & \mathcal{T}_{10} & \mathcal{T}_{11} & \mathcal{T}_{12} \end{bmatrix} \tag{4.25}$$

The rotation and translation vector are then obtained from the transformation matrix. It should be noted that the P3P algorithm does not guarantee orthonormal rotation matrix, therefore QR decomposition [63] or further refinement through pose optimization is required to obtain the proper rotation matrix.

The overall Perspective-3-Point algorithm for pose estimation is defined as follows,

Perspective-3-Point Algorithm
1. Compute matching interest points between left and right camera images at time k: $[\hat{p}_{l_k}, \hat{p}_{r_k}]$ 2. Compute matching interest points between left camera images at time k and $k+1$: $[\tilde{p}_{l_k}, \hat{p}_{l_{k+1}}]$ 3. Obtain a set of points satisfying the intersection of indices of $\tilde{p}_{l_k}$ and $\hat{p}_{l_k}$ to provide the tracked interest points: $[p_{l_k}, p_{r_k}, p_{l_{k+1}}]$ 4. Compute 3D point locations of the interest points at time k: X_k 5. Define matrix A such that: $$A = \begin{bmatrix} a_i \\ \vdots \\ a_n \end{bmatrix}$$ where,

$$a_i = \begin{bmatrix} X_i & \mathbf{0} & -u_i X_i \\ \mathbf{0} & X_i & -v_i X_i \end{bmatrix}$$

X_i is homogeneous 3D point coordinates of point i

(u_i, v_i) is the pixel coordinates of the point i at time $k+1$

$\mathbf{0}$ is a 1×4 zero vector

$n \geq 6$ is the number of $3\text{D} - 2\text{D}$ correspondence points

6. Compute Singular Value Decomposition (SVD) of A

$$[U, S, V] = \text{SVD}(A)$$

7. Obtain the null vector $\mathcal{T} \in \mathbb{R}^{12\times 1}$ of matrix A which is the last column of V

8. Rearrange the elements of $\mathcal{T}$ to obtain the transformation matrix $\mathcal{T}'$:

$$\mathcal{T}' = \begin{bmatrix} \mathcal{T}_1 & \mathcal{T}_2 & \mathcal{T}_3 & \mathcal{T}_4 \\ \mathcal{T}_5 & \mathcal{T}_6 & \mathcal{T}_7 & \mathcal{T}_8 \\ \mathcal{T}_9 & \mathcal{T}_{10} & \mathcal{T}_{11} & \mathcal{T}_{12} \end{bmatrix}$$

9. $\mathcal{T}'$ contains rotation and translation information according to the definition:

$$\mathcal{T}' = K[R \quad -Ro]$$

where,

K is scaling matrix

R is Rotation matrix

o is origin of camera frame at $k+1$ in world coordinates

$t = -Ro$ is the translation vector

10. The rotation matrix and translation vector can be computed given that R is orthonormal, and K is upper triangle. The matrices K, R, o, t can be obtained using RQ decomposition.

11. It should be noted that the rotation matrix computed is R_{k+1}^{k}. This means the rotation is from $k+1$ to k. In order to obtain the rotation and translation from k to $k+1$, the following equations will be used,

$$t' = -Rt$$

12. The transformation matrix is now be obtained from the computed rotation matrix and translation vector,

$$T = \begin{bmatrix} R & t \\ \mathbf{0}^{1\times 3} & 1 \end{bmatrix}$$

Table 4-6 P3P algorithm

Since, the algorithm requires at least 3 corresponding points, and the inlier points available are much greater than 3 therefore, the RANSAC algorithm (defined in section 4.2.5) is used to compute the best fit transformation matrix.

4.2.8 Pose Optimization

The pose estimate can be optimized by minimizing the reprojection error equation[12]. The reprojection error equation is given as follows,

$$\{R^*, t^*\} = \arg\min_{\hat{R},\hat{t}} \sum_{i}^{n} \left\| p_{k+1}^{i} - P\left(\hat{R}X_k^i + \hat{t}\right) \right\|^2 \tag{4.26}$$

Where, k is the time step, i is the index of the interest point, n is the total number of interest points, $p \in \mathbb{R}^{3\times 1}$ is the homogeneous pixel coordinate vector, $P \in \mathbb{R}^{3\times 4}$ is the camera matrix, $X \in \mathbb{R}^{4\times 1}$ is the homogeneous 3D world coordinate vector, $t \in \mathbb{R}^{3\times 1}$ is the translation vector between camera frames at k and $k+1$, $R \in \mathbb{R}^{3\times 3}$ is the rotation matrix between camera frames at k and $k+1$, and $T = [R \quad t]$ is the transformation matrix between camera frames at k and $k+1$.

The reproject error is the cost function, which can be optimized using the Levenberg Marquardt algorithm [67]. The details are provided in the reference, The algorithm is defined as follows:

$$x^{(j+1)} = x^{(j)} - \alpha_j \left(F x^{(j)} + \mu_j I\right)^{-1} g^{(j)} \tag{4.27}$$

Where, j is the iteration number, x is the parameter vector $x = [\phi \quad \theta \quad \psi \quad t_x \quad t_y \quad t_z]$, $g = \nabla f(x)$ is the gradient of the reprojection error function, $F(x) = \nabla^2 f(x)$ is the hessian of the reprojection error function, I is the identity matrix, α is the step size, and $\mu > 0: Fx + \mu I > 0$.

The algorithm is run iteratively until the difference $x^{j+1} - x^j$ is below a defined threshold. The resulting solution is the optimized transformation matrix.

$$T^* = [R^* \quad t^*]$$

4.2.9 Pose Update

The transformation matrix computed provides the information of rotation and translation of the camera pose at the current time step with respect to the camera frame at previous time step. In order to obtain the pose with respect to the camera frame at the initial time step, the camera pose is updated using the following equation:

$$V_{k+1} = V_k T_{k|k+1} \tag{4.28}$$

where, k is the time step, $V = \begin{bmatrix} R & t \\ \mathbf{0^{1\times 3}} & 1 \end{bmatrix}$ is the pose of the current camera frame with respect to the first frame, and T is the transformation matrix from previous camera frame to current camera frame.

It should be noted that since the estimated transformation matrix is simply concatenated to the previous estimates. However, since the estimates are accompanied with an error, therefore, the error will accumulate with each pose update causing drift as time increases.

4.3 Traditional Stereo Visual Odometry

4.3.1 Introduction

Stereo Visual Odometry (SVO) is a visual odometry technique which uses sequential images from two cameras mounted on the vehicle separated in one dimension with a distance called baseline [12]. SVO utilizes 3D-to-2D pose estimation technique by solving the P3P algorithm

followed by reprojection error minimizing to obtain the rotation and translation information. The benefit of SVO when compared to Monocular Visual Odometry (MVO) is that it provides absolute scale for translation estimates. This is because the baseline between the two cameras is a known scale metric in the world coordinate frame, allowing computation of exact 3D point location of interest points.

One of the drawbacks of SVO is that it requires more computational resources due to additional computation of processing two camera images, 3D point location, P3P pose estimation, and pose optimization. The other drawback is that the P3P pose estimate does not guarantee an orthonormal rotation matrix [12] and it requires further QR decomposition and optimization.

In this section the SVO algorithm as defined in [12] is presented, and its experimental evaluation is presented in section 4.5.

4.3.2 Algorithm

An overview of the SVO algorithm is provided as follows:

Stereo Visual Odometry
1. Obtain image matrix for left and right images at time k: I_{l_k}, I_{r_k} 2. Detect features in I_{l_k} and I_{r_k} using the SURF algorithm (presented in Table 4-1) to obtain a set of key point features for the left and right image: $(\hat{f}_{l_k}, \hat{f}_{r_k})$. 3. Match features between $\hat{f}_{l_k}$ and $\hat{f}_{r_k}$using Sum of Squared Difference (SSD) over the 64-element descriptor vectors of each key features to obtain a set of matched key features: (f_{l_k}, f_{r_k}). 4. Remove outliers using RANSAC (presented in Table 4-3) with the 5-point algorithm with (presented in Table 4-2) to obtain a set of inlier key points: $(\hat{p}_{l_k}, \hat{p}_{r_k} \in \mathbb{R}^{2\times 1})$. 5. At time step $k+1$ obtain the left image matrix $I_{l_{k+1}}$ from the left camera image.

6. Match features between $\hat{f}_{l_k}$ and $\hat{f}_{l_{k+1}}$ using SSD to obtain a set of matched key features: $(\tilde{f}_{l_k}, f_{l_{k+1}})$
7. Remove outliers using RANSAC with the 5-point algorithm to obtain a set of inlier key points: $(\tilde{p}_{l_k}, \hat{p}_{l_{k+1}})$
8. Obtain a set of points satisfying the intersection of indices of $\tilde{p}_{l_k}$ and $\hat{p}_{l_k}$: $(p_{l_k}, p_{r_k}, p_{l_{k+1}})$
9. Use corresponding key points $p^i_{l_k}$ and $p^i_{r_k}$ to compute a set of 3D location vectors $X^i_k \in \mathbb{R}^{3\times 1}$ using the Direct Linear Transformation method (presented in Table 4-5).
10. Estimate transformation matrix using Perspective-3-Point problem (presented in Table 4-6): $\hat{T}_{k|k+1} = [\hat{R}\ \ \hat{t}]$
11. Optimize $\hat{R}\ \hat{t}$ estimate by minimizing the reprojection error using the Levenberg-Marquardt algorithm (presented in section 4.2.8)

$$\{R^*, t^*\} = \arg\min_{\hat{R},\hat{t}} \sum_i \left\| p^i_{k+1_L} - P_L\left(\hat{R}X^i_k + \hat{t}\right)\right\|^2$$

12. Obtain the optimized transformation matrix:

$$T^*_{k|k+1} = \begin{bmatrix} R^* & t^* \\ \mathbf{0^{1\times 3}} & 1 \end{bmatrix}$$

13. Obtain camera pose V at time step $k+1$ by concatenating the optimized transformation matrix with the camera pose at time k (presented in section 4.2.9)

$$V_{k+1} = V_k T^*_{k|k+1}$$

Table 4-7 Stereo Visual Odometry algorithm

4.4 Modified Stereo Visual Odometry

4.4.1 Proposed Improvements

The Modified Stereo Visual Odometry (ModSVO) is a proposed VO technique which improves accuracy of the traditional SVO algorithm, while reducing the computation resource requirements. In ModSVO, the pose estimate is obtained by decomposition of the essential matrix instead of using Perspective-3-Point (P3P) solution. This avoids the orthonormality issue of P3P rotation matrix estimate. Furthermore, since the rotation matrix obtained from essential matrix decomposition does not suffer from scale issues like the translation vector, therefore, optimization is not required. The translation vector scale issue is rectified during the pose optimization phase, by minimizing the reprojection error. Therefore, the ModSVO optimization problem only has 3 optimizing parameters elements of translation vector as shown in equation (4.29), as compared to 6 optimizing parameters of translation and rotation in SVO, as shown in equation (4.26).

This section presents the algorithm for ModSVO, and its evaluation is presented in section 4.5.

4.4.2 Proposed Algorithm

The ModSVO algorithm is defined as follows:

Modified Stereo Visual Odometry
1. Obtain image matrix for left and right images at time k: I_{l_k}, I_{r_k} 2. Detect features in I_{l_k} and I_{r_k} using the SURF algorithm (presented in Table 4-1) to obtain a set of key point features for the left and right image: $(\hat{f}_{l_k}, \hat{f}_{r_k})$. 3. Match features between $\hat{f}_{l_k}$ and $\hat{f}_{r_k}$ using Sum of Squared Difference (SSD) over the 64-element descriptor vectors of each key features to obtain a set of matched key features: (f_{l_k}, f_{r_k}).

4. Remove outliers using RANSAC (presented in Table 4-3) with the 5-point algorithm with (presented in Table 4-2) to obtain a set of inlier key points: $(\hat{p}_{l_k}, \hat{p}_{r_k} \in \mathbb{R}^{2\times1})$.
5. At time step $k+1$ obtain the left image matrix $I_{l_{k+1}}$ from the left camera image.
6. Match features between $\hat{f}_{l_k}$and $\hat{f}_{l_{k+1}}$ using SSD to obtain a set of matched key features: $(\tilde{f}_{l_k}, f_{l_{k+1}})$
7. Remove outliers using RANSAC with the 5-point algorithm to obtain a set of inlier key points: $(\tilde{p}_{l_k}, \hat{p}_{l_{k+1}})$ and obtain the essential matrix $E_{k|k+1} \in \mathbb{R}^{3\times3}$.
8. Obtain a set of points satisfying the intersection of indices of $\tilde{p}_{l_k}$ and $\hat{p}_{l_k}$: $(p_{l_k}, p_{r_k}, p_{l_{k+1}})$
9. Use corresponding key points $p^i_{l_k}$ and $p^i_{r_k}$ to compute a set of 3D location vectors $X^i_k \in \mathbb{R}^{3\times1}$ using the Direct Linear Transformation method (presented in Table 4-5).
10. Decompose the essential matrix $E_{k|k+1}$ to obtain rotation matrix $R^* \in SO(3)$ and translation vector $\hat{t} \in \mathbb{R}^{3\times1}$ (presented in Table 4-4).
11. Optimize $\hat{t}$ estimate by minimizing the following reprojection error using the Levenberg-Marquardt algorithm (presented in section 4.2.8) for $\hat{t}$ only

$$\{t^*\} = \arg\min_t \sum_i \left\| p^i_{k+1_L} - P_L\left(R^* X^i_k + \hat{t}\right) \right\|^2 \tag{4.29}$$

12. Obtain the optimized transformation matrix:

$$T^*_{k|k+1} = \begin{bmatrix} R^* & t^* \\ \mathbf{0^{1\times3}} & 1 \end{bmatrix} \tag{4.30}$$

13. Obtain camera pose V at time step $k+1$ by concatenating the optimized transformation matrix with the camera pose at time k (presented in section 4.2.9)

$$V_{k+1} = V_k T^*_{k|k+1}$$

Table 4-8 ModSVO algorithm

4.5 Experimental Evaluation

In this section the KITTI sequence 10 (details in section 2.2) is used to provide reference trajectory and images for experimental evaluation of Modified Stereo Visual Odometry (ModSVO). The plots (Figure 4-5 to Figure 4-9) show the comparison between the reference ground truth (shown as 'Ref' in the plot legend), traditional Stereo VO (shown as 'SVO' in the plot legend), and Modified Stereo VO (shown as 'ModSVO' in the plot legend). The VO estimates have been transformed to the local navigation frame using the coordinate frame transformation of from camera coordinates frame to ENU (East-North-Up) coordinate frame (details in section 3.6.2).

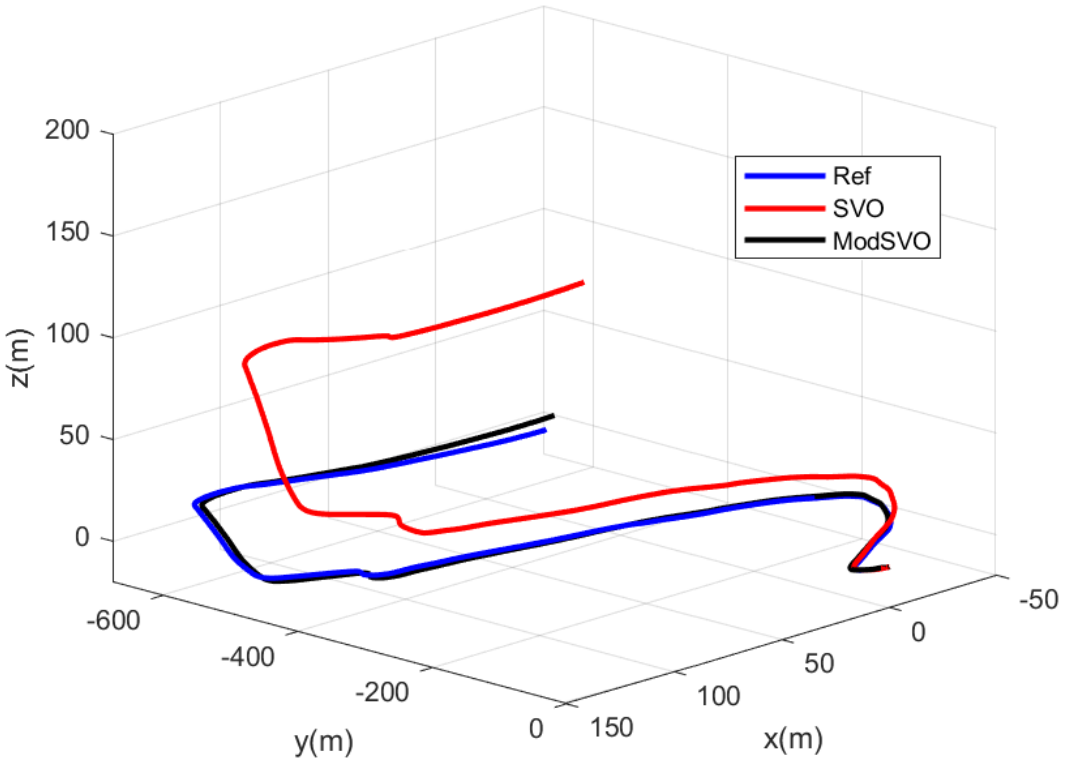

Figure 4-5 Trajectory plot comparison for Modified Stereo VO

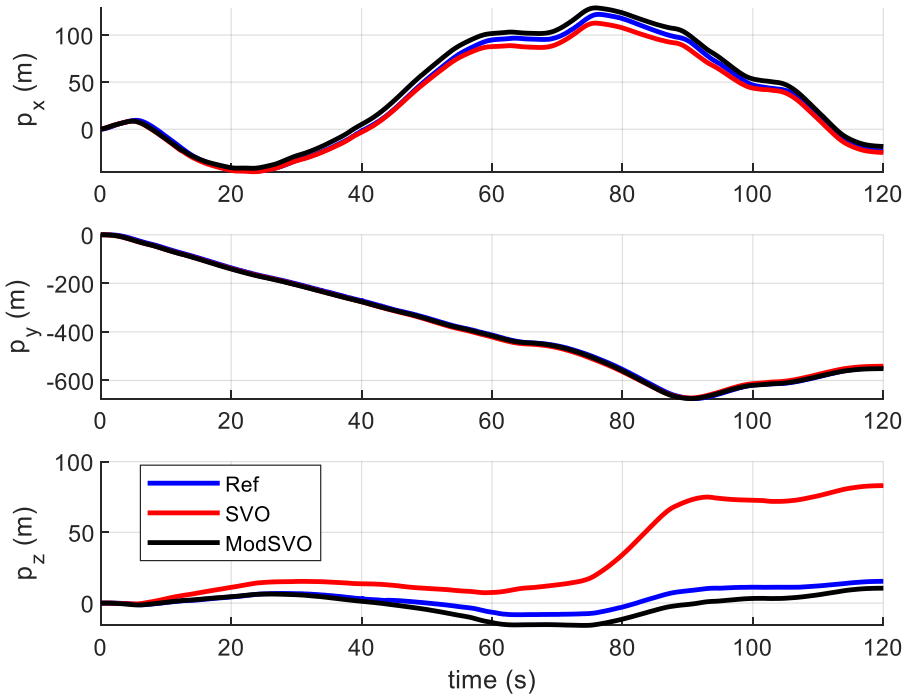

Figure 4-6 Position comparison for Modified Stereo VO

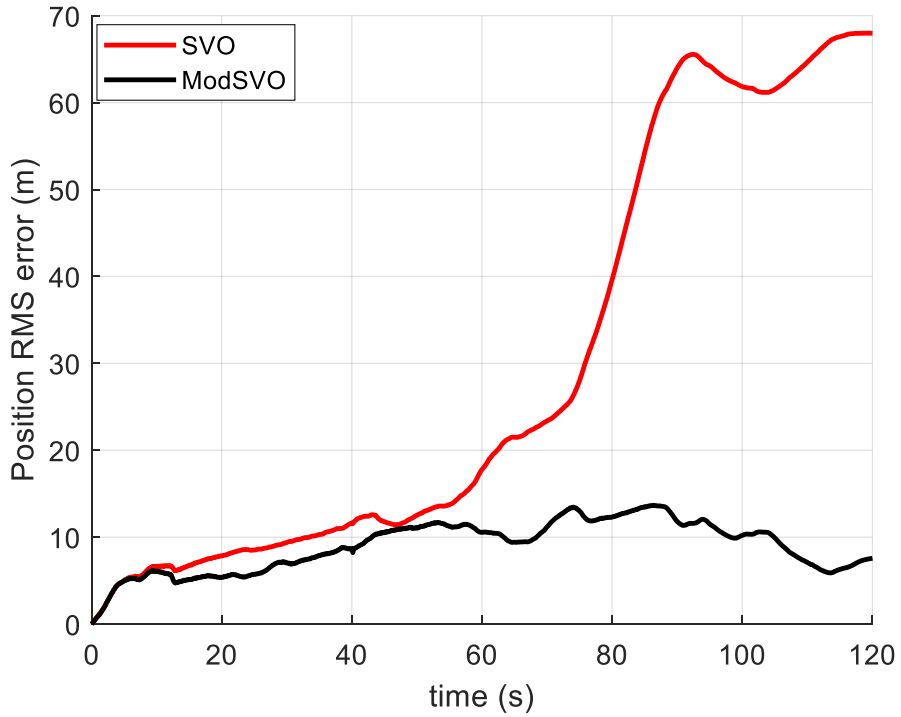

Figure 4-7 Position Root Mean Square error comparison for Modified SVO

The Figure 4-5 shows the trajectory plot comparison of ModSVO and SVO with respect to the reference , Figure 4-6 shows the position estimate comparison, where p_x, p_y, p_z is the position estimate in x, y, z axis, and Figure 4-7 shows the position Root Mean Square (RMS) error plot.

The Root Mean Square (RMS) error for position averaged over the whole trajectory is given in the following table.

	Position RMSE
ModSVO	$\mathbf{9.42\ m}$
SVO	$38.00\ m$

Table 4-9 Position Root Mean Square Error of ModSVO

The ModSVO algorithm, compared to the traditional SVO, reduces the position RMS error by 75.21 %. This shows a significant improvement in estimation of position for standalone navigation solution through visual odometry.

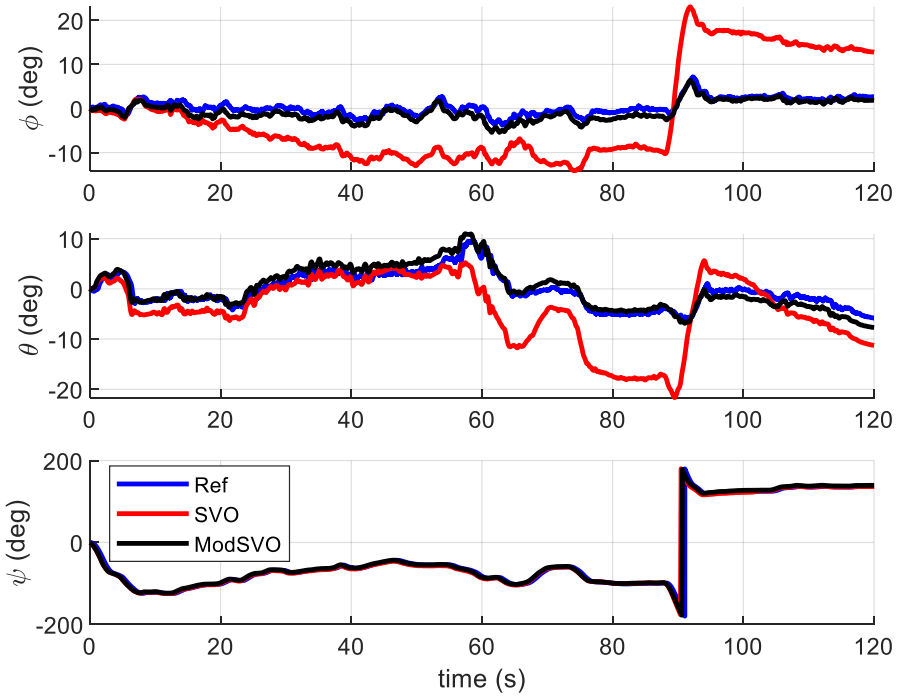

Figure 4-8 Attitude comparison for Modified Stereo VO

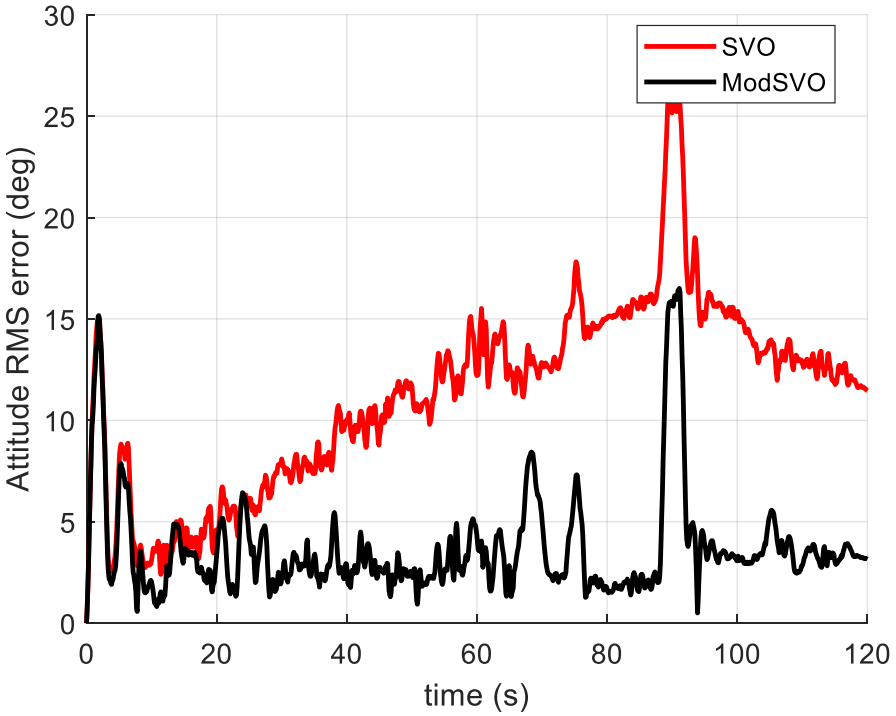

Figure 4-9 Attitude Root Mean Square error comparison for Modified SVO

Figure 4-8 shows the attitude comparison, where ϕ, θ, ψ is the roll, pitch, and yaw angle about the x, y, z axis, and Figure 4-9 shows the attitude Root Mean Square (RMS) error plot. The Root Mean Square (RMS) error for attitude averaged over the whole trajectory is given in the following table.

	Attitude RMSE
ModSVO	**4.49°**
SVO	12.16°

Table 4-10 Attitude Root Mean Square Error of ModSVO

The ModSVO algorithm, compared to the traditional SVO, reduces the attitude RMS error by 63.09 %. This shows a significant improvement in estimation of attitude for standalone navigation solution through visual odometry.

It can be observed that using the essential matrix decomposition in ModSVO, instead of P3P pose estimation, has improved the accuracy significantly. Furthermore, since P3P algorithm is not being run, and only translation parameters are being optimized therefore, ModSVO requires

lesser computational resources. It can also be observed that both VO techniques suffer from error increment with time due to error accumulation during the pose update phase. This factor motivates the next chapter, where the VO solution is fused with Inertial Navigation System (INS) and Global Positioning System (GPS) in order to further improve accuracy of the navigation solution.

Chapter 5: Multi-Sensor Fusion

5.1 Introduction

Sensor fusion algorithms are used to compute integrated navigation solutions by combining measurements for multiple sensors. All real-world sensors have some level of noise affecting their output. Inertial Measurement Unit (IMU) sensors are used to measure body acceleration and body angular velocity which is used to compute position, velocity, and attitude using the Inertial Navigation System (INS). The IMU sensor readings are affected by noise. These errors propagate over time as the navigation solution of IMU is integrated over time. Global Positioning System (GPS) sensors are used to measure position and velocity of the vehicle. While GPS sensors are not affected by error propagation over time, they are still affected with other sources of errors, for example, GPS signal blockage where the GPS receiver cannot get a clear line-of-sight to the GPS satellite due tall trees, tall buildings, and tunnels. Furthermore, the quality of GPS receiver affects the levels of noise in the estimated position and velocity. Cameras are used as sensors for Visual Odometry (VO). The accuracy of VO is affected by lighting conditions, number of inliers, and outliers. The VO error also suffers from compounded errors, as the subsequent estimated camera poses are concatenated to the previous estimated camera poses.

Therefore, it is clear that standalone sensors will not yield navigation solution results of reasonable accuracy in order to be used for ground vehicle guidance. In this case, sensor fusion algorithms are employed to combine sensor readings from multiple sources thereby predicting and mitigating errors and increasing accuracy.

There are a multitude of sensor fusion algorithms available in literature as presented in section 1.4. For this research, the Extended Kalman Filter is used. The Kalman Filter (KF) was first proposed by Kalman in 1960 [68]. Since then, it has undergone many improvements, and

currently it is one of the most popular methods of implementing integrated navigation solutions. KF employs information of the system model, along with the information of sensor uncertainties to compute optimal estimates of the states. The optimality and stability of the basic KF algorithm is limited to linear system models with white gaussian noise. This is a serious constraint, because the real-world systems are generally non-linear, and while white noise can be considered a reasonable assumption, it is not guaranteed. A modification of Kalman Filter called the Extended Kalman Filter (EKF) is therefore presented. This allows the usage of non-linear functions for system and sensor model [1]. This research will focus on the use of EKF to develop the proposed Dual Extended Kalman Filter (DEKF) algorithm for INS/VO/GPS integration.

This chapter is structured as follows. Section 5.1 will provide background information pertaining necessary to develop the sensor fusion algorithms. This includes introduction to the Inertial Navigation System (INS) along with its error sources and model and Global Positioning System (GPS). The basic Extended Kalman Filter (EKF) algorithm is also presented. Section 5.2 presents the structure of INS/VO integration along with experimental evaluation in case of VO failure. Section 5.3 presents the structure of INS/GPS integration along with experimental evaluation in case of GPS failure. Finally, Section 5.4 presents the proposed Dual EKF structure for INS/VO/GPS fusion. Experimental evaluation will be carried out in case of GPS failure and VO failure. The algorithms will be tested and compared on KITTI sequence 10, the details of which are provided in chapter 2.

5.2 Preliminaries

5.2.1 Inertial Navigation System

5.2.1.1 Introduction

The Inertial Navigation Solution (INS) refers to computation of position, velocity, and attitude of a vehicle using sensors measuring motion by utilizing the properties of inertia. INS does not

require an external reference, rather it only utilizes known initial conditions and sensor output to compute the navigation solution. The sensors used in INS are known as inertial sensors. The two main types of inertial sensors used for ground vehicle INS are accelerometers and gyroscopes. The accelerometers measure the body acceleration in terms of specific force, and the gyroscopes measure the body angular rates. Both these types of sensors provide inertial measurement in one dimension, and hence can only contribute to navigation solution in one dimension. In order to obtain 3-dimensional navigation solution for position, velocity, and orientation, an arrangement of 3 accelerometers and 3 gyroscopes is required such that each sensor is aligned with the three orthogonal body frame axes. An assembly unit which contains 3 accelerometers, and 3 gyroscopes is known as an Inertial Measurement Unit (IMU).

INS comprises of two components, IMU and a navigation computer. INS has an advantage of providing high frequency navigation output allowing real time localization and control, however, as a standalone sensor the INS has a major drawback that over time the error in navigation solution accumulates rapidly [9]. This is due to the fact that INS requires integration of specific force, and angular rates in order to obtain the position, velocity, and orientation. Integration over time has a property of aggregating the previous states. However, since each state estimation is erroneous to some extent, the errors of each solution iteration accumulate resulting in large drift in the navigation solution over time. This characteristic renders the INS unsuitable in scenarios involving large distances. In order to use INS for navigation of a ground vehicle, its solution must be periodically corrected using a different sensor. The resulting setup is known as the Integrated Navigation System [62].

In this chapter, the Inertial Measurement Unit and its associated errors are characterized, and the INS algorithm used to compute the position, velocity, and orientation of the vehicle is presented.

5.2.1.2 Inertial Measurement Unit (IMU)

The IMU sensor is an assembly of 3 accelerometers, and 3 gyroscopes. The IMU sensor provides output in the IMU frame i. As stated in section 3.5 this coordinate frame consists of 3 orthogonal axes (x, y, z axes). Therefore, the accelerometer and gyroscopes are aligned with each of these axes to provide 3-dimensional specific force and angular rates.

The accelerometer measures specific force of the vehicle, denoted by f, with the units $\frac{m}{s^2}$. The gyroscope measures the angular rate denoted by ω with the units $\frac{rad}{s}$. The type of IMU is categorized by the kind of accelerometers and gyroscopes used in it [1]. There are different types of accelerometers, such as pendulous accelerometers, vibrating-beam accelerometers, and MEMS (Micro-Electro-Mechanical Systems) accelerometers. Each kind has different properties, merits, and demerits. Similarly, there are different types of gyroscopes such as spinning mass gyroscope, optical gyroscope, vibratory gyroscopes, and MEMS gyroscopes. The details of the principles behind the workings of each type of accelerometer and gyroscope, as well as supporting diagrams are provided in [1]. The details are not revisited in this thesis, rather the focus is aimed towards IMU sensor errors.

In real world applications, the output from any type of sensor is affected by some form of error. In case of IMU there are several sources of error. As the IMU readings are twice integrated in case of accelerometer and once integrated in terms of gyroscope, these sources of errors result in a large drift in the solution, therefore it is important to calibrate and correct the errors in IMU readings in order to avoid this issue. Generally, the source of error depends on the type of accelerometer and gyroscope sensor being used, however there are a few error categories which are common to all types of IMU sensors, these are: bias errors, errors due to scale factor, errors due to cross-coupling, mechanical vibrations, electrical signal errors, errors due to temperature variations. From these errors, the bias errors have the most dominant effect, followed by

random noise errors such as electrical signal errors. Therefore, in this research, both these types of errors are corrected for the INS navigation solution.

The bias errors b refer to an offset in the sensor output. This means that the IMU sensor output will have an added value which needs to be subtracted in order to obtain the true value. Bias can be divided in two categories, static bias b_s and dynamic bias b_d. This is represented in the following equation.

$$b_a = b_{a_s} + b_{a_d} \tag{5.1}$$

Where, b_a is the accelerometer bias, b_{a_s} is the accelerometer static bias, and b_{a_d} is the accelerometer dynamic bias. The gyroscope bias can be represented similarly.

The static bias is characterized by having a constant value throughout the IMU operation, however, the static bias constant value changes whenever the IMU is turned on. The dynamic bias, on the other hand, varies throughout the IMU operation and needs to be estimated in real time in order to correct it. The static bias can be estimated by keeping the vehicle stationary after turning on the IMU. A constant offset in the output from the zero value will be the static bias. The dynamic bias can be estimated using the Kalman Filter technique which is presented in section 5.2.3.

The angular random walk and velocity random walk refer to random noise affecting the gyroscope and accelerometer readings, respectively. At low frequencies, this noise can be characterized by white gaussian noise. The distinctiveness of white noise is that the next reading is not dependent on the previous reading, therefore this form of noise cannot be corrected by calibration, rather in can be mitigated by applying Kalman Filter correction. The manufacture values of the IMU noise are provided in Table 2-1.

5.2.1.3 INS computation

INS algorithm uses the accelerometer and gyroscope readings in order to obtain the position, velocity, and orientation of the vehicle. The three main components of this algorithms are bias correction, coordinate transformation, and integration [62]. Since the stream of IMU data arrives at $100\ Hz$ frequency, it is not continuous and hence the computation is carried out in discrete form.

A schematic diagram showing INS computation is presented in the figure below.

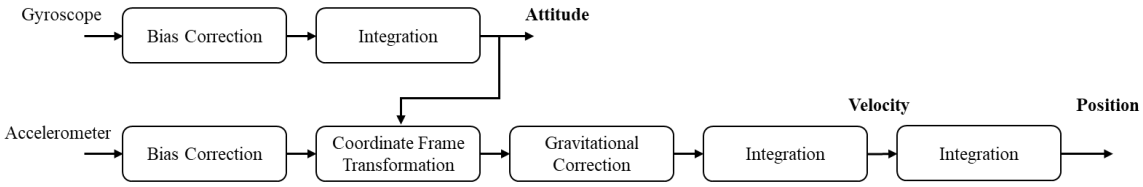

Figure 5-1 INS computation schematic

The discretized equations for INS computation are provided in the following subsections.

Attitude Update:

In the first iteration, the initial roll (ϕ), pitch (θ), yaw (ψ) is known. The first step is to convert these Euler angles Θ to Direction Cosine Matrix (DCM) C_b^n. This is shown in equation (5.2).

$$\mathcal{C}(\Theta) = C_b^n = \begin{bmatrix} c\theta c\psi & -c\phi s\psi + s\phi s\theta c\psi & s\phi s\psi + c\phi s\theta c\psi \\ c\theta s\psi & c\phi c\psi + s\phi s\theta s\psi & -s\phi c\psi + c\phi s\theta s\psi \\ -s\theta & s\phi c\theta & c\phi c\theta \end{bmatrix} \quad (5.2)$$

Where, $\mathcal{C}(\Theta)$ is a function which converts Euler angles to DCM, c and s represent cosine and sine functions.

Bias correction is then applied to the angular rate output from gyroscope, and the result is used to update the DCM as shown in equation (5.3).

$$C_{b\,k+1}^n = C_{b\,k}^n + C_{b\,k}^n \left[\omega_k - b_{g_k}\right]_\times \delta t \quad (5.3)$$

In (5.3) k is the time step at which the solution is being computed. C_b^n is the DCM computed from Euler angles (rad) , ω_k $\left(\frac{rad}{s}\right)$ is the angular rate measured by gyroscope, b_g $\left(\frac{rad}{s}\right)$ is the gyroscope bias, and δt (s) is the duration of time step from k to $k+1$. The operator $[\ ]_\times$ is represents skew symmetric matrix transformation of a vector. Consider a vector defined as $r = [r_1 \quad r_2 \quad r_3]$ then $[r]_\times$ will be as shown in equation (5.4),

$$[r]_\times = \begin{bmatrix} 0 & -r_3 & r_2 \\ r_3 & 0 & -r_1 \\ -r_2 & r_1 & 0 \end{bmatrix} \tag{5.4}$$

Finally, the updated DCM is converted back to Euler angles using equation (5.5).

$$\mathcal{A}(C) = \Theta = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} \tan^{-1}\left(\frac{C_{3,2}}{C_{3,3}}\right) \\ \sin^{-1}(-C_{3,1}) \\ \tan^{-1}\left(\frac{C_{2,1}}{C_{1,1}}\right) \end{bmatrix} \tag{5.5}$$

Where, $\mathcal{A}(C)$ is a function which converts DCM to Euler angles, and $C_{i,j}$ is the element in i^{th} row and j^{th} column of the matrix C.

It should be noted that Euler angles and DCM have an issue of "Gimbal Lock" when the pitch (θ) angle is $90°$ [62]. However, this is not a problem for the scope of this research, as it is focused on ground vehicles, therefore the navigation solution being computed is not expected to have high pitch or roll angles. Furthermore, for sensor fusion the error state modification for Extended Kalman Filter is used which utilizes small changes in angles rather than actual angles. This ensures that their values remain close to 0, hence avoiding any potential singularities.

Velocity Update:

Similar to the attitude update, in the first iteration the initial velocity is known, and then in every subsequent iteration the previous velocity value is used to update the state. The first step

for velocity update is to perform bias correction of the body acceleration output from accelerometer $a_k \left(\frac{m}{s^2}\right)$. This is followed by transformation from body frame to the ENU frame and performing gravitational acceleration correction. The result is integrated over time in a discretized form. The overall equation representing velocity update is given in the following equation.

$$v_{k+1} = v_k + \left[C_b^n{}_k\left(a_k - b_{a_k}\right) - g\right] \delta t \tag{5.6}$$

In (5.6) k is the time step at which the solution is being computed, C_b^n is the DCM computed using equation (5.2), $a_k \left(\frac{m}{s^2}\right)$ is the body acceleration measured by accelerometer, $b_a \left(\frac{m}{s^2}\right)$ is the accelerometer bias, $g = [0 \quad 0 \quad -9.807]^T \frac{m}{s^2}$ is the gravitational acceleration vector, and $\delta t\ (s)$ is the duration of time step from k to $k+1$.

It should be noted that the gravitational acceleration vector is taken as constant rather than varying with respect to position on the surface of the Earth because in the scenario of a ground vehicle navigation solution, the change in g was observed and found to be negligible. This allows neglecting variation of g over time resulting in reduced computational processing without loss of accuracy.

Position Update:

Similarly, as in attitude and velocity update, in the first iteration the initial position is known, and then in every subsequent iteration the previous position value is used to update the state. The position update is relatively simple to compute. The velocity state is integrated over time to obtain the new position. This is presented in equation (5.7).

$$p_{k+1} = p_k + v_k\, \delta t \tag{5.7}$$

A higher order position update equation, given in (5.8), was also tested but the difference was negligible therefore, equation (5.7) was used to reduce computational processing requirement.

$$p_{k+1} = p_k + v_k\,\delta t + \frac{1}{2}\left[C_{b\,k}^{n}\left(a_k - b_{a_k}\right) - g\right]\delta t^2 \tag{5.8}$$

Following the provided INS computation algorithm, the navigation solution will be given in the ENU frame (presented in section 3.6). Moving ahead, the output from GPS and VO sensors will also be converted to ENU frame in order to implement sensor fusion.

5.2.1.4 Results

This section shows the results for implementation and testing of INS computation. Testing the INS computation on the real-world dataset section 2.2 is not insightful. This is because the real-world IMU sensor readings are affected with noise which result in rapid accumulation of error for INS solution. This can be seen in Figure 5-19 – Figure 5-25. Therefore, a simulated trajectory is designed. The acceleration and angular rates are computed using a simulated IMU sensor function. However, no bias or noise is added to the IMU sensor readings. The results are shown in Figure 5-2 to Figure 5-10.

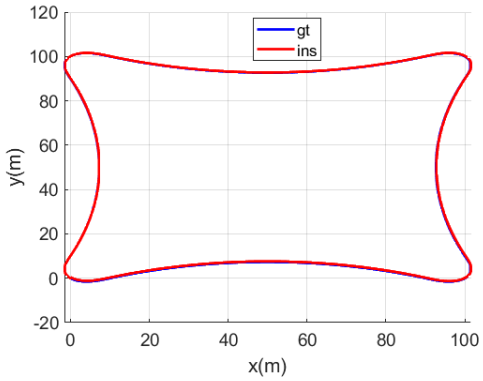

Figure 5-2 Ground Truth vs Inertial Navigation System trajectory without IMU noise

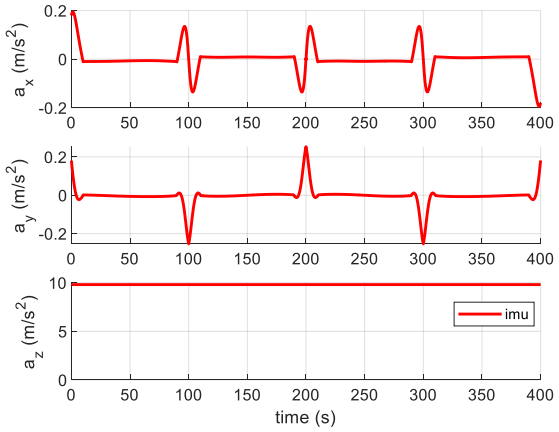

Figure 5-3 Simulated accelerometer data without noise

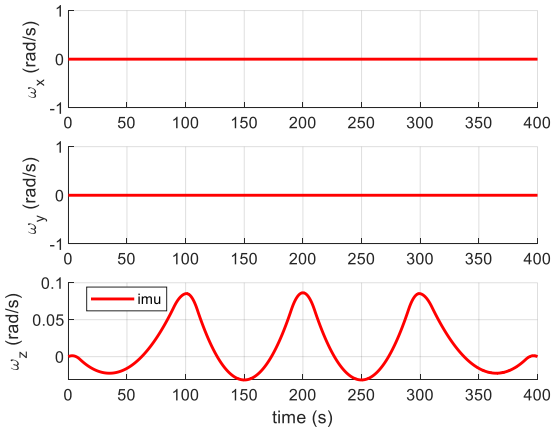

Figure 5-4 Simulated gyroscope data without noise

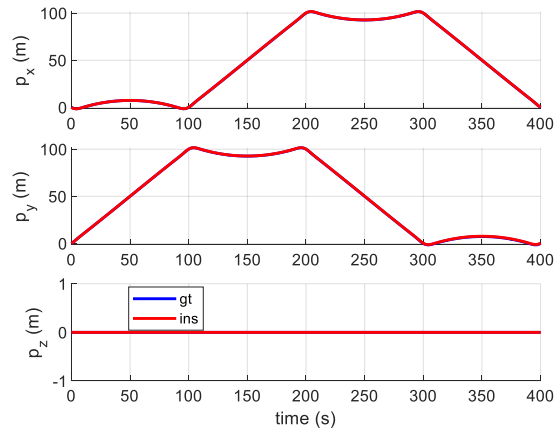

Figure 5-5 Ground Truth vs INS position for clean IMU data

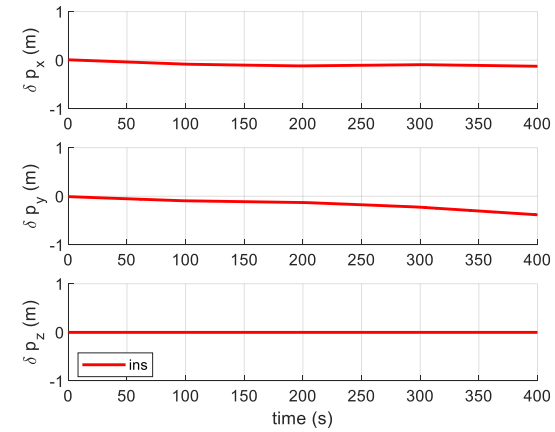

Figure 5-6 INS position error for clean IMU data

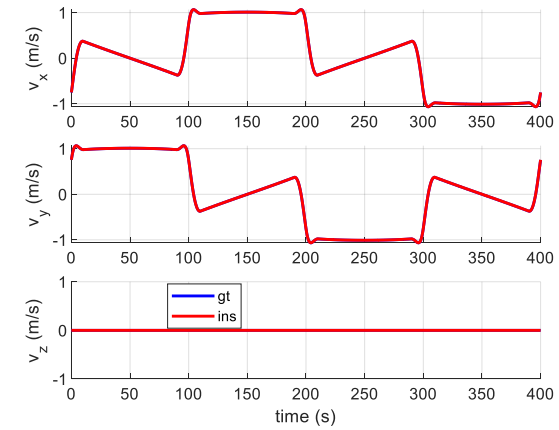

Figure 5-7 Ground Truth vs INS velocity for clean IMU data

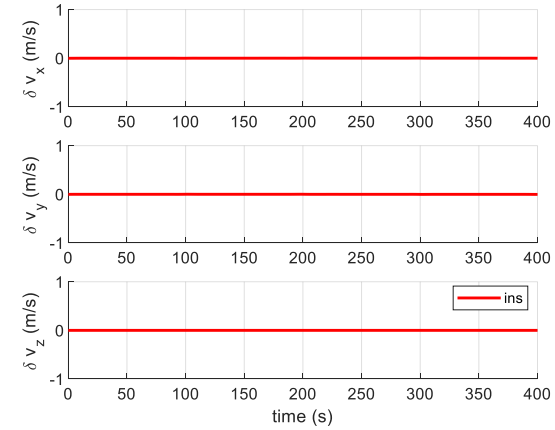

Figure 5-8 INS velocity error for clean IMU data

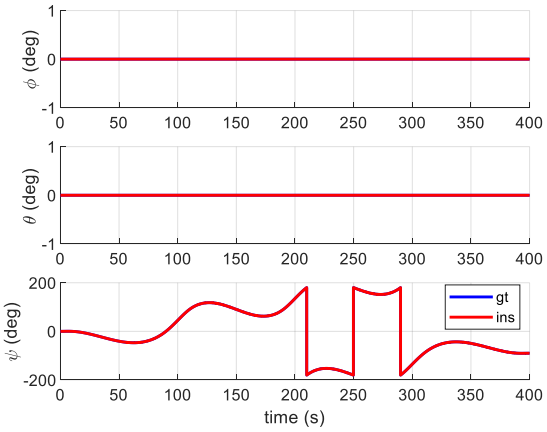

Figure 5-9 Ground Truth vs INS attitude

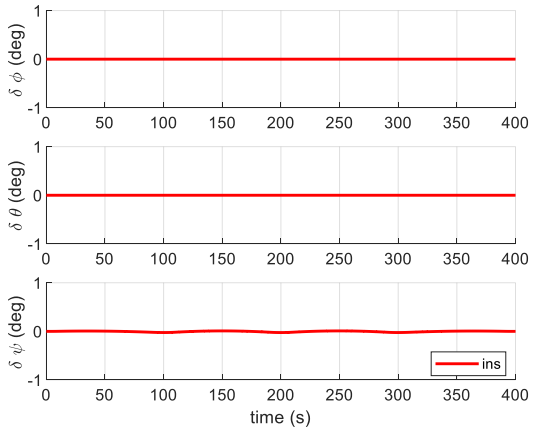

Figure 5-10 INS attitude error for clean IMU data

Figure 5-3 and Figure 5-4 show the body accelerometer and body angular rate sensor readings from the simulated IMU which show that the IMU data is not affected by noise. It can be seen that in the absence of noise the INS algorithm follows the trajectory with minimal deviation (Figure 5-6 to Figure 5-10). The small deviations are a result of lower order integration terms used in the algorithm. However, since the errors are low and the current implementation reduces the complexity and processing of the computation significantly, therefore, these discrete integration assumptions can be considered valid.

Using the same simulated data, white Gaussian noise is now added to all channels of IMU sensor output. The resulting IMU signal are shown in Figure 5-11 and Figure 5-12. The corresponding INS solutions are shown in Figure 5-13 to Figure 5-18.

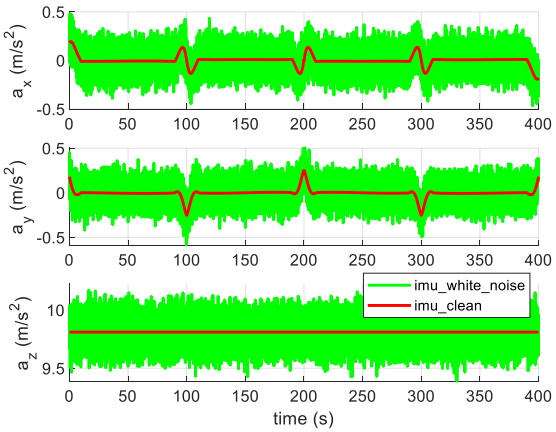

Figure 5-11 Simulated accelerometer (white noise)

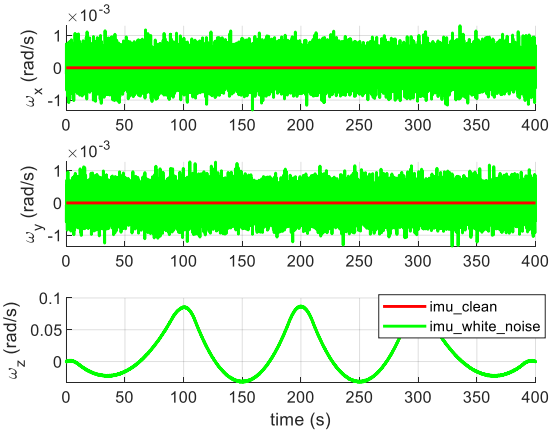

Figure 5-12 Simulated gyroscope (white noise)

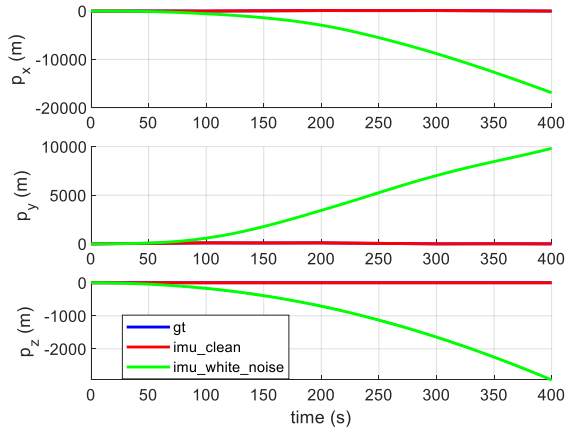

Figure 5-13 Position plot of IMU without noise vs IMU with white noise

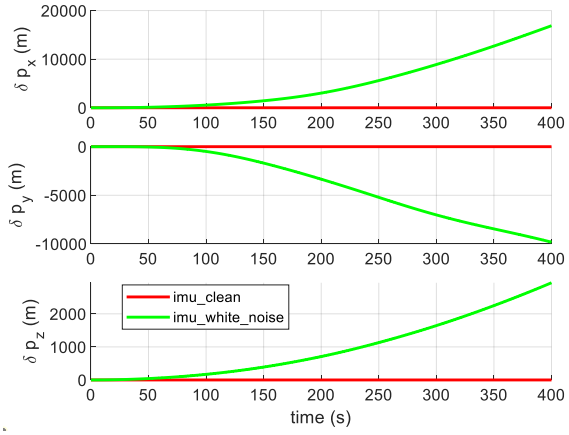

Figure 5-14 Position error plot of IMU without noise vs IMU with white noise

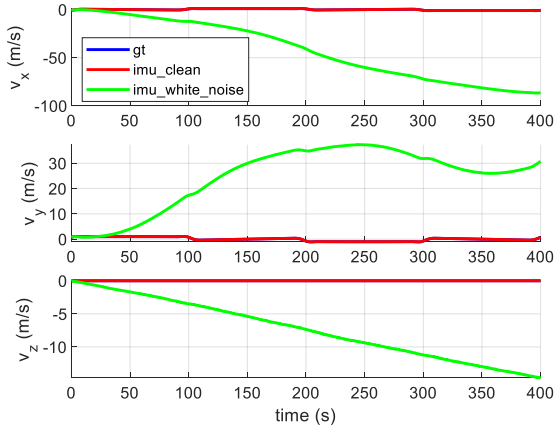

Figure 5-15 Velocity plot of IMU without noise vs IMU with white noise

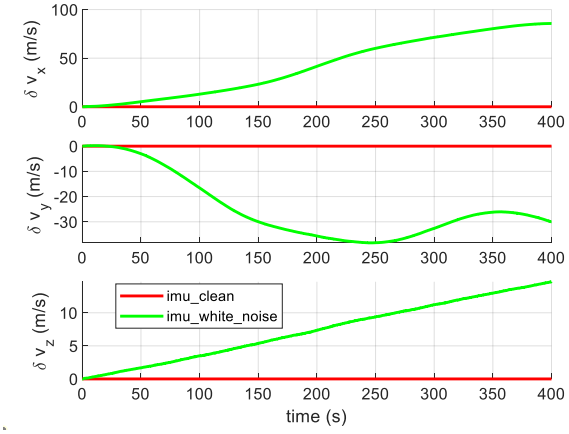

Figure 5-16 Velocity error plot of IMU without noise vs IMU with white noise

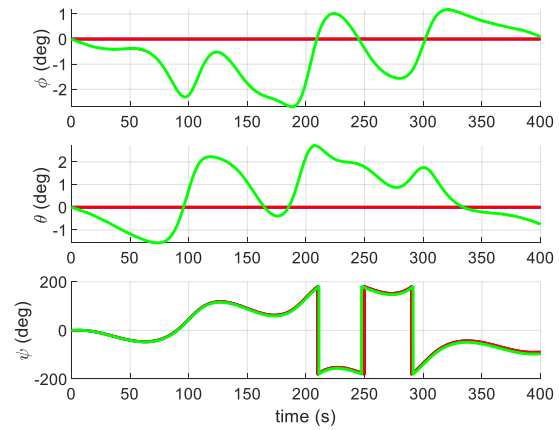

Figure 5-17 Attitude plot of IMU without noise vs IMU with white noise

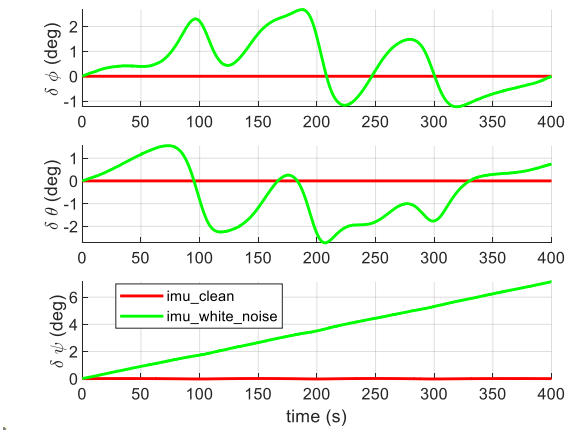

Figure 5-18 Attitude error plot of IMU without noise vs IMU with white noise

The result shows rapid accumulation of error in position and velocity resulting in large drift in solution. The attitude solution does not deviate as rapidly because the attitude computation only requires a single integration resulting in much less error accumulation.

The INS algorithm is also tested on the KITTI sequence 10 dataset whose results is shown in Figure 5-19 to Figure 5-25.

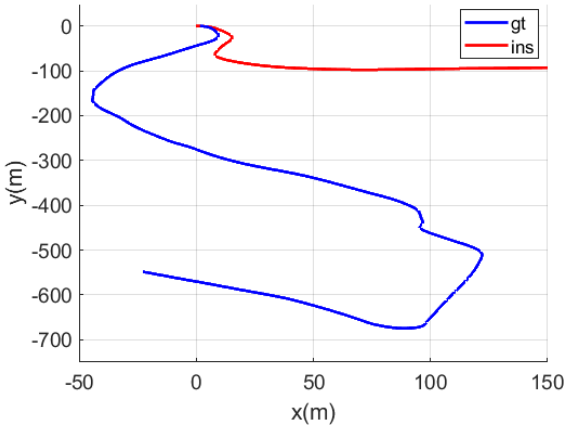

Figure 5-19 Trajectory plot of Ground Truth vs INS solution

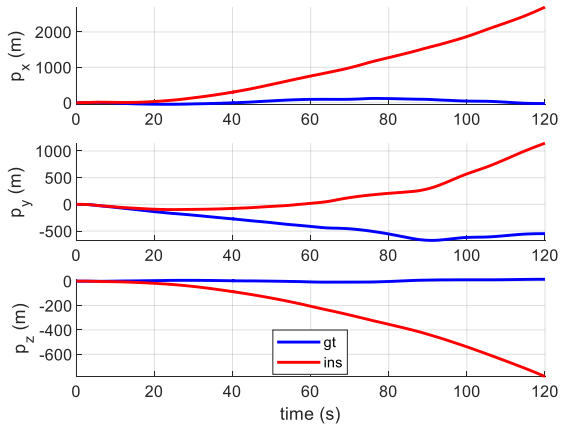

Figure 5-20 Position plot of Ground Truth vs INS solution

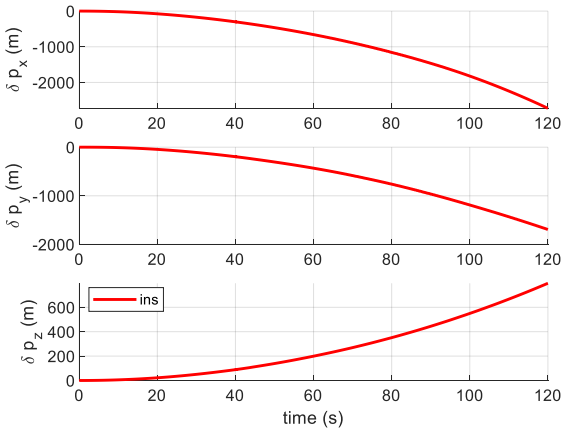

Figure 5-21 Position error plot of Ground Truth vs INS solution

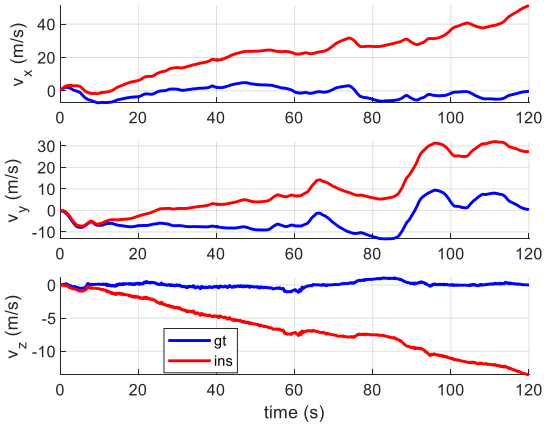

Figure 5-22 Velocity plot of Ground Truth vs INS solution

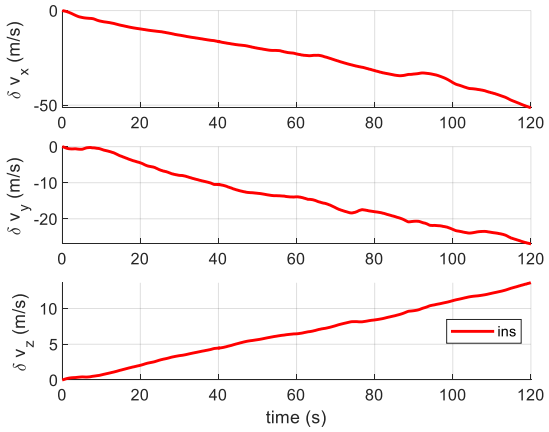

Figure 5-23 Velocity error plot of Ground Truth vs INS solution

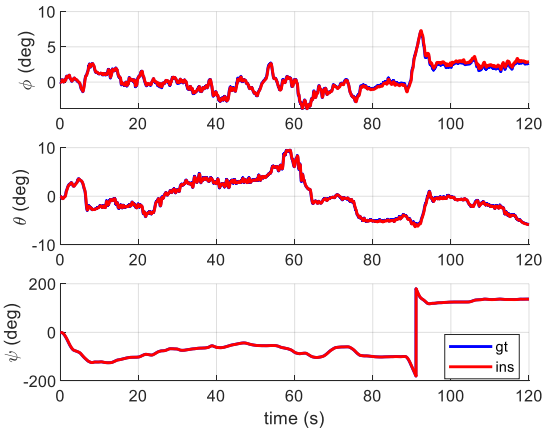

Figure 5-24 Attitude plot of Ground Truth vs INS solution

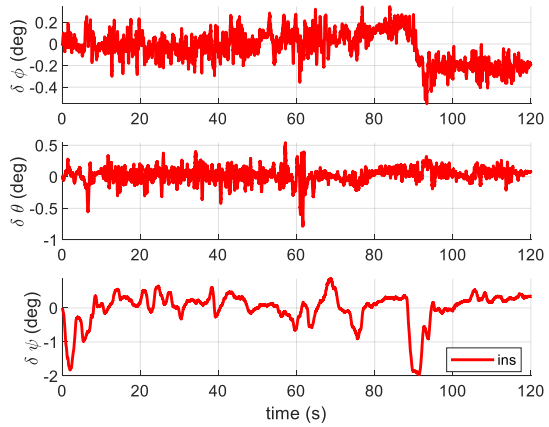

Figure 5-25 Attitude error plot of Ground Truth vs INS solution

These results depict the same drift phenomenon that was observed in simulation-based results with added white gaussian noise in Figure 5-16 and Figure 5-14. The velocity and attitude requiring only a single integration results in relatively smaller errors, whereas position requiring two integrations shows a large drift in solution. The error in Figure 5-21 and Figure 5-23 is due to the noisy IMU sensor readings. The INS computation requires integration of the body acceleration over each time step, and the output position and velocity values are added to the previous estimates. With noise in the IMU sensor readings, the estimates of each time step will have their own errors and they also accumulate the errors due to noisy estimates for all previous time steps. As a result, an increasing drift in estimates is seen when compared to the ground truth.

5.2.2 Global Positioning System

5.2.2.1 Introduction

Global Navigation Satellite System (GNSS) is navigation setup which involves obtaining position and velocity measurements using satellites. A GNSS setup consists of 3 components: satellite, receiver, and processor [1]. The satellites transmit their relative position information to the receiver. The processor uses triangulation methods the obtain position and velocity in the ECEF geodetic coordinate frame. Obtaining accurate position requires range information from at least 4 satellites. Due to this, the GNSS setups require a constellation of satellites in different orbital planes around the Earth. This ensures that at any given time, any point on the surface of the Earth can be observed by at least 4 satellites. GNSS setups have been implemented by USA, China, Russia, and Europe. The currently available GNSS setups, their country of origin, and the number of satellites is shown in the table below [2].

Name	Operated By	Number of Satellites
Global Positioning System (GPS)	United States	31
BeiDou Navigation Satellite System (BDS)	People's Republic of China	35
Galileo	European Union	24+
Globalnaya Navigazionnaya Sputnikovaya Sistema (GLONASS)	Russian Federation	24+

Table 5-1 Different GNSS constellations in use

The Global Positioning System (GPS) is developed and maintained by USA, and it is publicly available at no cost to anyone with a GPS receiver. The real-world dataset used in this research obtains the GNSS information from GPS, therefore this research will focus on the GPS sensor.

5.2.2.2 GPS Computation

The GPS sensor used in this research is capable of providing absolute position of the sensor directly. Since the tightly-coupled approach is not utilized, this absolute position and velocity can be used directly for the multi-sensor fusion task. Therefore, the in-depth computation for obtaining the GPS data from its signal is beyond the scope of this thesis and hence it is not provided here. However, a brief overview of the computation is given. The mechanism of GPS involves transmission of radio frequency signals from the satellites, which are received and interpreted by the GPS receiver. The signals contain the information of satellite orbit, and time at which the signal was transmitted. The receiver comprises of its own internal clock. This is used to obtain the time at which the signal is received. Knowing that the radio frequency signals travel at the speed of light, the difference in transmitted time and receiving time can be used to obtain the range between the receiver and the satellite [1].

$$\rho = (t_{transmitted} - t_{received})\, c \tag{5.9}$$

In equation (5.9) ρ is the pseudo-range of the satellite, $t_{transmitted}$ is the time at which signal was transmitted by the satellite, $t_{received}$ is the time at which the signal was received by the GPS receiver, and c is the speed of light constant. It should be noted that ρ is called 'pseudo-range' because of the clock ambiguity. This is due to the fact that the actual atomic clock tick of the satellite may be different than the one in the receiver, hence it is known as pseudo-range.

The pseudo-range must be obtained from at least 3 satellites to compute the latitude, longitude, and altitude of the receiver. The signal from the 4^{th} satellite is used to correct the clock ambiguity. As a result, an accurate position is obtained. Velocity is obtained by differentiating the position values over time.

The actual computation of position using satellite orbit computation and pseudo range is beyond the scope of this research. This is because the GPS receiver has a built-in processor

which decrypts the elements of the radio frequency signal and computes the position in ECEF frame. Secondly in this research loosely coupled integration methods are used for sensor fusion, as a result only the position and velocity output from the GPS receiver are used, rather than the intermediate values of pseudo range and clock times.

5.2.2.3 Error Sources

The GPS provides accurate position and velocity data which is not affected by error accumulation over time and drift. However, it cannot be used as a sole sensor for navigation applications. This is because firstly the GPS update rate is usually at a low rate of $1 - 10\ Hz$ compared to the frequency of IMU $(100\ Hz)$. This does not allow real time navigation and control of the vehicle.

Secondly, the GPS signal is susceptible to outages. This is a result of GPS signal blockage of one or more of the 4 satellites by tall surrounding objects, like buildings or trees. This also is caused when the vehicle is underground, or in a tunnel. Another aspect maybe GPS denial using jamming devices or deliberate degradation of GPS signals using selective ability [1].

Finally, the GPS signals are also affected by error. These include clock ambiguity, signal noise, impact of outer atmosphere layers, and receiver electronics noise. Most of these error sources are modelled and corrected in the GPS processor, however the receiver electric noise and GPS outage issue require using integrated navigation systems.

5.2.2.4 GPS dataset

The GPS sensor resolves the position and velocity measurements to ECEF geodetic coordinate frame to provide longitude λ (°), latitude φ (°), and altitude $h\ (m)$. In order to transform these readings to ENU frame, first the Mercator projection method is used to convert them to body frame, and then C_b^n transformation matrix is used to convert them into ENU frame. The details

of Mercator projection can be found in [61]. The set of equations governing the transformation are as follows,

$$x_b = \lambda\left(\frac{\pi}{180}\right) R_E \cos\left(\varphi_0\left(\frac{\pi}{180}\right)\right) \tag{5.10}$$

$$y_b = \log\left(\tan\left((90+\varphi)\left(\frac{\pi}{360}\right)\right)\right) R_E \cos\left(\varphi_0\left(\frac{\pi}{180}\right)\right) \tag{5.11}$$

$$z_b = h \tag{5.12}$$

$$\begin{bmatrix} x_n \\ y_n \\ z_n \end{bmatrix} = C_b^n \begin{bmatrix} x_b \\ y_b \\ z_b \end{bmatrix} \tag{5.13}$$

In equations (5.10) – (5.13), λ is the longitude in degrees, φ is the latitude in degrees, R_E is the average radius of Earth, φ_0 is the latitude at initial conditions, h is the altitude in m, (x_b, y_b, z_b) are the GPS position values in body coordinate frame, (x_n, y_n, z_n) are the GPS position values in ENU frame, and C_b^n is the transformation matrix from body frame to ENU frame given by equation (5.2).

It should be noted that $R_E = 6378137m$ is taken as the average radius of Earth because, even though the radius of Earth is not uniform, for the limited distances expected from ground vehicles, the radius of Earth can be assumed to be constant. The result of conversion from geographic coordinate (Figure 5-26) to ENU coordinate (Figure 5-27) is shown as follows.

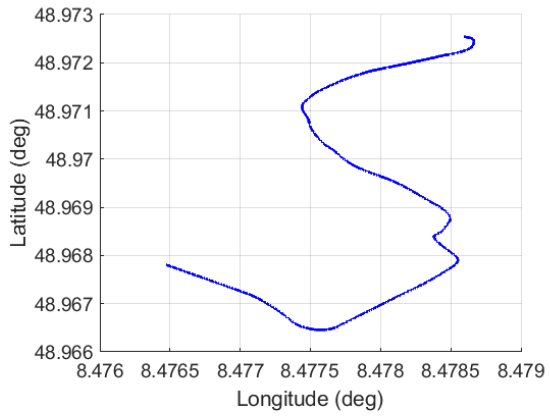

Figure 5-26 GPS position in ECEF frame

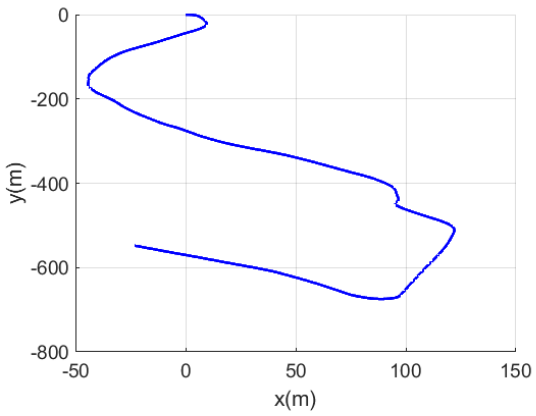

Figure 5-27 GPS position in ENU frame

The KITTI dataset uses the OXTS 3003 model IMU/GPS sensor. The specifications are given in section 2.2.2. While the KITTI dataset provides the IMU and ground truth data, the standalone GPS data is not provided. However, this can be resolved by simulating a low-cost GPS through adding white noise, and down sampling the ground truth data. This method is supported by literature in [34] and in [60]. A white gaussian noise with a mean of $[2 \quad 2 \quad 2]\ m$ is added to all position channels, and a white gaussian noise with a mean of $[0.5 \quad 0.5 \quad 0.5]\ m$ is added to velocity channels in order to simulate the GPS signal. Furthermore, the GPS update rate has been down sampled from $100Hz$ to $10\ Hz$. The resulting GPS signal is shown in the following figures.

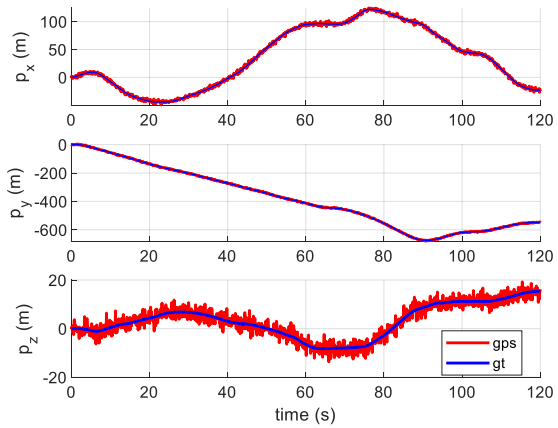

Figure 5-28 Noise-added and down sampled GPS position signal compared to the Ground Truth

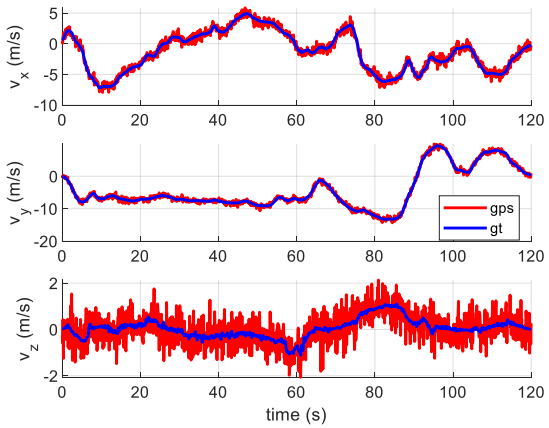

Figure 5-29 Noise-added and down sampled GPS velocity signal compared to the Ground Truth

5.2.3 Extended Kalman Filter

The Extended Kalman Filter (EKF) is a type of recursive Bayesian estimator [1]. It utilizes the knowledge of standard deviations of sensor and state errors along with the system and sensor model to predict and correct the state estimate. The EKF may be implemented as Total-State EKF or Error-State EKF (ESKF) [69]. In total-state EKF, the predicted state estimate contains complete knowledge of the state, and the complete state is corrected in the correction phase. Whereas, in ESKF only the state errors are predicted and corrected, and the corrected state

errors are used to update the complete state. The ESKF has a benefit of operating close to the point of linearized state Jacobian matrix, hence the non-linearity has a lesser effect on the estimates. Furthermore, the ESKF attitude states are close to zero thereby avoiding any singularity cases. Therefore, in this research the error-state implementation of EKF is used.

The main components of EKF are state vector, state covariance matrix, system model, measurement vector, measurement covariance matrix, and sensor model. The state vector x is set a variable which define the position, velocity, and attitude of the ground vehicle at any given time. The state covariance matrix P defines the expected error in the state estimates. The system model predicts the update in state using a non-linear function f given the current state x, input vector u, and time step δt. The system noise covariance matrix Q defines the noise present in the system model. The measurement vector z is a set of variables which define the sensor output at current time step. The sensor model H defines the predicted sensor output given the current state. The measurement noise covariance matrix R defines the noise present in the sensor model.

The EKF consists of 3 phases, prediction phase, correction phase, and update phase. In the prediction phase the previous state is updated using the non-linear system model. The linearized Jacobian matrix at the current state is computed from the system model and it is used along with system noise matrix to update the state covariance matrix. Equations (5.14) to (5.16) show the equations governing the prediction phase of EKF.

$$x_{k|k+1} = f(x_k, u_k, \delta t) \tag{5.14}$$

$$F_k = \frac{\partial f}{\partial x}\Big|_{x = x_k} \tag{5.15}$$

$$P_{k|k+1} = F_k P_k F_k^T + Q \tag{5.16}$$

Where k is the time step, x is the state vector, f is the function of the system model, F is the system Jacobian matrix, P is the state covariance matrix, Q is the covariance matrix of system noise w, x is the state vector, u is the input vector, δt is the duration of time step.

In the correction phase, the measurement vector and the sensor model Jacobian is used to obtain the innovation in the state estimate. The sensor noise covariance matrix and the residual covariance matrix is used to compute the Kalman gain. This gain is applied to the innovation vector in order to compute the corrected state estimate errors. Equation (5.17) – (5.22) show the equations governing the correction phase of EKF.

$$H_{k+1} = \frac{\partial h}{\partial x} \Big|_{x_{k|k+1}} \tag{5.17}$$

$$y = z_{k+1} - h\left(x_{k|k+1}\right) \tag{5.18}$$

$$S = H_{k+1}\bar{P}_{k|k+1}H_{k+1}^{T} + R \tag{5.19}$$

$$K = P_{k|k+1}H_{k+1}^{T}S^{-1} \tag{5.20}$$

$$\delta x_{k+1} = Ky \tag{5.21}$$

$$P_{k+1} = (I - KH_{k+1})P_{k|k+1} \tag{5.22}$$

Where, $h(x)$ is the sensor model, H is the Jacobian of the sensor model computed at $x_{k|k+1}$, z is the measurement vector, y is the innovation vector, R is the covariance matrix of sensor noise v, S is the residual covariance matrix, K is the Kalman gain, δx is the state error estimate.

In the update phase, the predicted state is updated using the corrected state estimate errors. Equations (5.23) presents the update equation for position, velocity, and bias states.

$$x_{k+1} = x_{k|k+1} + \delta x_{k+1} \tag{5.23}$$

Equation (5.24) presents the update equation for attitude states.

$$x_{k+1} = \mathcal{A}\left(\mathcal{C}(x_{k|k+1})\, \mathcal{C}(\delta x_{k+1}) \right) \tag{5.24}$$

Where, $\mathcal{C}$ represents a function transforming Euler angles to DCM shown in equation (5.2), $\mathcal{A}$ represents the function transforming DCM to Euler angles shown in equation (5.5). The updated states are then fed back to the system model to predict the next state estimate.

The overall EKF algorithm is compiled in the following table.

EKF Algorithm
Prediction Stage: $x_{k\|k+1} = f(x_k, u_k, \delta t)$ $F_k = \frac{\partial f(x_k, u_k, \delta t)}{\partial x}$ $P_{k\|k+1} = F_k P_k F_k^T + Q$
Correction Stage: $H_{k+1} = \frac{\partial h(x_{k+1})}{\partial x}$ $y = z_{k+1} - h(x_{k\|k+1})$ $S = H_{k+1} P_{k\|k+1} H_{k+1}^T + R$ $K = P_{k\|k+1} H_{k+1}^T S^{-1}$ $\delta x_{k+1} = Ky$ $P_{k+1} = (I - KH_{k+1}) P_{k\|k+1}$
Update phase: For position, velocity, and biases states: $x_{k+1} = x_{k\|k+1} + \delta x_{k+1}$

For attitude states: $$x_{k+1} = \mathcal{A}\left(\mathcal{C}(x_{k\|k+1})\, \mathcal{C}(\delta x_{k+1})\right)$$

Table 5-2 EKF algorithm

This research uses EKF as a basis for developing sensor fusion algorithms for the following integrated systems:

1. INS/VO: Loosely coupled EKF with delayed VO feedback is used.
2. INS/GPS: Loosely coupled EKF is used.
3. INS/VO/GPS: Loosely coupled Dual EKF is used.

Each of these integrated systems are presented in the subsequent sections of this chapter. It should be noted that INS is used as the system prediction model in all of the integrated system, hence there are some common elements in all of the aforementioned navigation systems. The structures of these elements are presented.

The state vector is defined as follow,

$$\boldsymbol{x} = \begin{bmatrix} p_x & p_y & p_z & v_x & v_y & v_z & \phi & \theta & \psi & b_{a_x} & b_{a_y} & b_{a_z} & b_{g_x} & b_{g_y} & b_{g_z} \end{bmatrix}^T \quad (5.25)$$

Where, $\boldsymbol{x} \in \mathbb{R}^{15\times 1}$ is the state vector, (x, y, z) are the resolving axes of ENU frame, p is the position, v is the velocity, (ϕ, θ, ψ) are the roll, pitch, yaw angle respectively, b_a is the accelerometer bias, and b_g is the gyroscope bias. The initial value of $\boldsymbol{x}$ is defined by the GPS or VO solution at the initial conditions.

The input vector is defined as:

$$\boldsymbol{u} = \begin{bmatrix} a_x & a_y & a_z & \omega_x & \omega_y & \omega_z \end{bmatrix}^T \quad (5.26)$$

Where, $\boldsymbol{u} \in \mathbb{R}^{6\times1}$ is the input vector, (x, y, z) are the resolving axes of the body frame, a is the body acceleration, ω is the body angular rate. The input vector elements are taken from the IMU sensor.

The system model function f is defined as follows:

State	Variables	model
Orientation	$\Theta = [\phi, \theta, \psi]^T$	$\Theta^k = [\phi^k, \theta^k, \psi^k]^T$ $C^k = \begin{bmatrix} c\theta c\psi & -c\phi s\psi + s\phi s\theta c\psi & s\phi s\psi + c\phi s\theta c\psi \\ c\theta s\psi & c\phi c\psi + s\phi s\theta s\psi & -s\phi c\psi + c\phi s\theta s\psi \\ -s\theta & s\phi c\theta & c\phi c\theta \end{bmatrix}$ $C^{k+1} = C^k + C^k\left[\omega_m - b_g\right]_\times \delta t$ $\Theta^{k+1} = \begin{bmatrix} \tan^{-1}\left(\frac{C_{3,2}^{k+1}}{C_{3,3}^{k+1}}\right) \\ \sin^{-1}\left(-C_{3,1}^{k+1}\right) \\ \tan^{-1}\left(\frac{C_{2,1}^{k+1}}{C_{1,1}^{k+1}}\right) \end{bmatrix}$
Position	$p = \left[p_x, p_y, p_z\right]^T$	$p^{k+1} = p^k + v^k\,\delta t$
Velocity	$v = \left[v_x, v_y, v_z\right]^T$	$v^{k+1} = v^k + [C(a_m - b_a) - g]\,\delta t$

Table 5-3 Non-linear system model for EKF

The system model Jacobian F is given in the following equations

$$F = \begin{bmatrix} 1 & 0 & 0 & \delta t & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & \delta t & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & \delta t & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & \frac{\partial v_x}{\partial \phi} & \frac{\partial v_x}{\partial \theta} & \frac{\partial v_x}{\partial \psi} & \frac{\partial v_x}{\partial b_{a_x}} & \frac{\partial v_x}{\partial b_{a_y}} & \frac{\partial v_x}{\partial b_{a_z}} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & \frac{\partial v_y}{\partial \phi} & \frac{\partial v_y}{\partial \theta} & \frac{\partial v_y}{\partial \psi} & \frac{\partial v_y}{\partial b_{a_x}} & \frac{\partial v_y}{\partial b_{a_y}} & \frac{\partial v_y}{\partial b_{a_z}} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & \frac{\partial v_z}{\partial \phi} & \frac{\partial v_z}{\partial \theta} & 0 & \frac{\partial v_z}{\partial b_{a_x}} & \frac{\partial v_z}{\partial b_{a_y}} & \frac{\partial v_z}{\partial b_{a_z}} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{\partial \phi}{\partial \phi} & \frac{\partial \phi}{\partial \theta} & 0 & 0 & 0 & 0 & -\delta t & \frac{\partial \phi}{\partial b_{g_y}} & \frac{\partial \phi}{\partial b_{g_z}} \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{\partial \theta}{\partial \phi} & 1 & 0 & 0 & 0 & 0 & 0 & \frac{\partial \theta}{\partial b_{g_y}} & \frac{\partial \theta}{\partial b_{g_z}} \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{\partial \psi}{\partial \phi} & \frac{\partial \psi}{\partial \theta} & 1 & 0 & 0 & 0 & 0 & \frac{\partial \psi}{\partial b_{g_y}} & \frac{\partial \psi}{\partial b_{g_z}} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (5.27)$$

$$\frac{\partial v_x}{\partial \phi} = \delta t \left((s\phi s\psi + c\phi c\psi s\theta) \left(a_y - b_{a_y} \right) + (c\phi s\psi - c\psi s\phi s\theta)\left(a_z - b_{a_z} \right) \right) \quad (5.28)$$

$$\frac{\partial v_x}{\partial \theta} = \delta t \left((c\phi c\psi c\theta)\left(a_z - b_{a_z} \right) - (c\psi s\theta)\left(a_x - b_{a_x} \right) + (c\psi c\theta s\phi)(a_y - b_{a_y}) \right) \quad (5.29)$$

$$\frac{\partial v_x}{\partial \psi} = -\delta t \left(\begin{matrix} (c\phi c\psi + s\phi s\psi s\theta) \left(a_y - b_{a_y} \right) - (c\psi s\phi - c\phi s\psi s\theta)\left(a_z - b_{a_z} \right) \\ +(c\theta s\psi)\left(a_x - b_{a_x} \right) \end{matrix} \right) \quad (5.30)$$

`

$$\frac{\partial v_x}{\partial b_{a_x}} = -\delta t (c\psi c\theta) \quad (5.31)$$

$$\frac{\partial v_x}{\partial b_{a_y}} = -\delta t (c\phi s\psi - c\psi s\phi s\theta) \quad (5.32)$$

$$\frac{\partial v_x}{\partial b_{a_z}} = -\delta t (s\phi s\psi + c\phi c\psi s\theta) \quad (5.33)$$

$$\frac{\partial v_y}{\partial \phi} = -\delta t \left((c\psi s\phi - c\phi s\psi s\theta) \left(a_y - b_{a_y} \right) + (c\phi c\psi + s\phi s\psi s\theta)\left(a_z - b_{a_z} \right) \right) \quad (5.34)$$

$$\frac{\partial v_y}{\partial \theta} = \delta t \left(c\phi c\theta s\psi \left(a_z - b_{a_z} \right) - s\psi s\theta \left(a_x - b_{a_x} \right) + c\theta s\backslash phi\ s\psi \left(a_y - b_{a_y} \right) \right) \quad (5.35)$$

$$\frac{\partial v_y}{\partial \psi} = \delta t \left(\begin{matrix} (s\phi s\psi + c\phi c\psi s\theta)\left(a_z - b_{a_z} \right) - (c\phi s\psi - c\psi s\phi s\theta) \left(a_y - b_{a_y} \right) \\ +c\psi c\theta \left(a_x - b_{a_x} \right) \end{matrix} \right) \quad (5.36)$$

$$\frac{\partial v_y}{\partial b_{a_x}} = -\delta t (c\theta s\psi) \quad (5.37)$$

$$\frac{\partial v_y}{\partial b_{a_y}} = -\delta t (c\phi c\psi + s\phi s\psi s\theta) \quad (5.38)$$

$$\frac{\partial v_y}{\partial b_{a_z}} = -\delta t (c\psi s\phi - c\phi s\psi s\theta) \quad (5.39)$$

$$\frac{\partial v_z}{\partial \phi} = \delta t \left(c\phi c\theta \left(a_y - b_{a_y} \right) - c\theta s\phi \left(a_z - b_{a_z} \right) \right) \quad (5.40)$$

$$\frac{\partial v_z}{\partial \theta} = -\delta t \left(c\theta \left(a_x - b_{a_x} \right) + c\phi s\theta \left(a_z - b_{a_z} \right) + s\phi s\theta \left(a_y - b_{a_y} \right) \right) \tag{5.41}$$

$$\frac{\partial v_z}{\partial b_{a_x}} = \delta t s\theta \tag{5.42}$$

$$\frac{\partial v_z}{\partial b_{a_y}} = -\delta t (c\theta s\phi) \tag{5.43}$$

$$\frac{\partial v_z}{\partial b_{a_z}} = -\delta t (c\phi c\theta) \tag{5.44}$$

$$\frac{\partial \phi}{\partial \phi} = 1 - \delta t \left(c\phi \left(b_{g_y} - \omega_y \right) - s\phi \left(b_{g_z} - \omega_z \right) \right) \tan\theta \tag{5.45}$$

$$\frac{\partial \phi}{\partial \theta} = -\delta t \left(c\phi \left(b_{g_z} - \omega_z \right) + s\phi \left(b_{g_y} - \omega_y \right) \right) (\tan^2 \theta + 1) \tag{5.46}$$

$$\frac{\partial \phi}{\partial b_{g_y}} = -\delta t \; s\phi \tan\theta \tag{5.47}$$

$$\frac{\partial \phi}{\partial b_{g_z}} = -\delta t \; c\phi \tan\theta \tag{5.48}$$

$$\frac{\partial \theta}{\partial \phi} = \delta t \left(c\phi \left(b_{g_z} - \omega_z \right) + s\phi \left(b_{g_y} - \omega_y \right) \right) \tag{5.49}$$

$$\frac{\partial \theta}{\partial b_{g_y}} = -\delta t \; c\phi \tag{5.50}$$

$$\frac{\partial \theta}{\partial b_{g_z}} = \delta t \; s\phi \tag{5.51}$$

$$\frac{\partial \psi}{\partial \phi} = -\delta t \frac{\left(c\phi \left(b_{g_y} - \omega_y \right) - s\phi \left(b_{g_z} - \omega_z \right) \right)}{c\theta} \tag{5.52}$$

$$\frac{\partial \psi}{\partial \theta} = -\delta t \frac{s\theta \left(c\phi \left(b_{g_z} - \omega_z \right) + s\phi \left(b_{g_y} - \omega_y \right) \right)}{c\theta^2} \tag{5.53}$$

$$\frac{\partial \psi}{\partial b_{g_y}} = -\delta t \frac{s\phi}{c\theta} \tag{5.54}$$

$$\frac{\partial \psi}{\partial b_{g_z}} = -\delta t \frac{c\phi}{c\theta} \tag{5.55}$$

The initial values of state covariance matrix P, system noise covariance matrix Q are based on IMU sensor specifications provided in section 2.2.2 and tuned according to trial-and-error method.

5.3 INS/VO Integrated System

5.3.1 Algorithm

This section presents the algorithm and implementation for INS/VO sensor fusion using EKF. The INS/VO integration structure is shown in Figure 5-30.

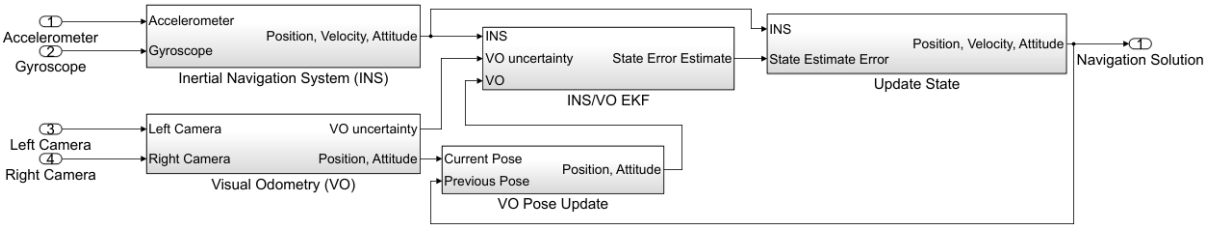

Figure 5-30 INS/VO Integration Structure

The INS/VO algorithm uses the navigation solution computed from IMU input as the state vector and state covariance prediction. The measurement vector is generated by VO. As presented in section 4.4, the Modified Stereo Visual Odometry (ModSVO) algorithm provides the best navigation solution estimate compared to Stereo Visual Odometry (SVO), therefore, for INS/VO integration, the visual odometry solution is computed using ModSVO algorithm. The position, velocity, and attitude values from INS and position and attitude values from VO are input to the EKF block along with the VO uncertainty.

The EKF algorithm is described in Table 5-3. The state vector $\boldsymbol{x}$, system model f, system Jacobian F, state covariance matrix P, and system noise covariance matrix Q will be the same

as defined in section 5.2.3. The remaining elements specific to INS/VO integration are described in this section.

The sensor model function h_{vo} directly outputs the position and attitude states. The sensor model Jacobian H is given in (5.56).

$$H_{vo} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{5.56}$$

The measurement vector z is defined by (5.57)

$$z_{vo} = [p_{x_{vo}} \quad p_{y_{vo}} \quad p_{z_{vo}} \quad \phi_{vo} \quad \theta_{vo} \quad \psi_{vo}]^T \tag{5.57}$$

Defining the sensor noise covariance matrix R_{vo} for VO is not as straightforward as defining Q because unlike IMU, the noise specifications for VO are not directly provided. [14] proposed a VO uncertainty estimation method using number of inliers correlation. In this research, the VO uncertainty for position, $\sigma_{p_{vo}}$ is obtained by computing standard deviation of residuals obtained from non-linear optimization of reprojection error for all inlier points, and the uncertainty for attitude, $\sigma_{\Theta_{vo}}$ is obtained by computing the standard deviation of Sampson distances [63] of inliers obtained from estimating the essential matrix between consecutive frame. As a result, R_{vo} is updated with new VO measurements. This method has shown to drastically improve state estimates when compared to a constant R_{vo}.

$$R_{vo} = \begin{bmatrix} \sigma^2_{p_{vo}} & 0 \\ 0 & \sigma^2_{\Theta_{vo}} \end{bmatrix} \tag{5.58}$$

Applying this method, the $\sigma_{p_{vo}}$ in case of KITTI sequence 10, is shown in Figure 5-31.

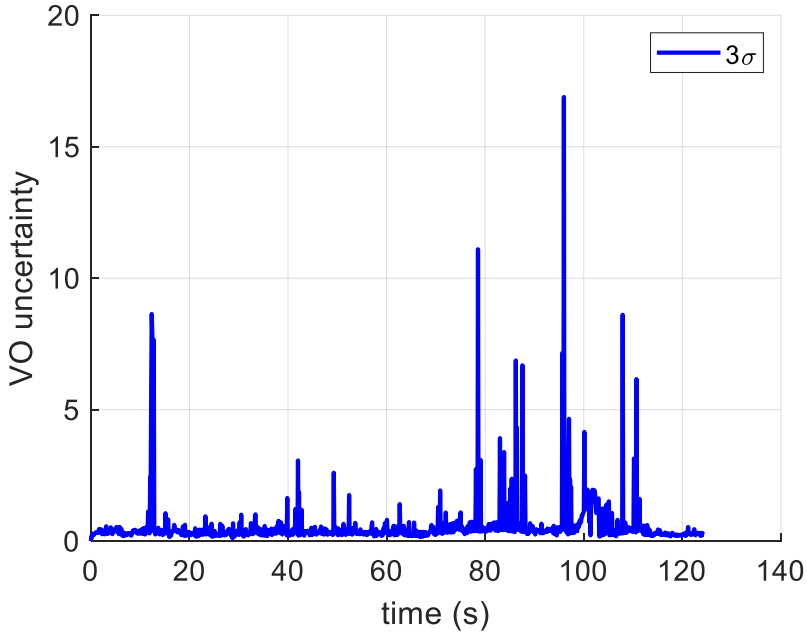

Figure 5-31 VO uncertainty computed using reprojection error

The INS/VO EKF provides corrected error of state estimate. These are used to update the INS estimated states. The output provides the final estimate of position, velocity, and attitude. As stated in section 4.2.9 the camera transformation T_k computed from images at time k is pre-multiplied with the previous VO pose V_{k-1} in order to update the pose V_k. In case of INS/VO integration the final estimate is converted to camera coordinate frame obtaining $\tilde{V}_k$. This is provided as a feedback to update VO pose T_{k+1} at timestep $k+1$. This is shown in the equation below.

$$V_{k+1} = \tilde{V}_k T_{k+1} \tag{5.59}$$

This technique gives the benefit of reducing the accumulated error in VO thereby improving the estimates fed into the INS/VO EKF.

5.3.2 Experimental Evaluation

In this section the INS/VO algorithm is evaluated using the KITTI sequence 10 dataset presented in section 2.2. The evaluation is done for two cases. The first case is the normal case in which there is no sensor failure. This test allows observation of improvement in navigational

accuracy when compared to INS only, and VO only solution. The second case is the failure case. In this case, a VO failure is simulated by blocking VO input to the sensor fusion algorithm. This test allows observation of navigational accuracy deterioration in case of sensor failure.

5.3.2.1 INS/VO (without VO failure)

In this section, the INS/VO integrated system explained in section 5.3 is implemented and is compared with standalone INS and standalone VO with respect to the ground truth reference. In the following figures (Figure 5-32 to Figure 5-38) '**Ref**' refers to ground truth reference, '**INS**' refers to INS only solution, '**VO**' refers to VO only solution computed by ModSVO, and '**INS_VO**' refers to INS/VO integrated solution.

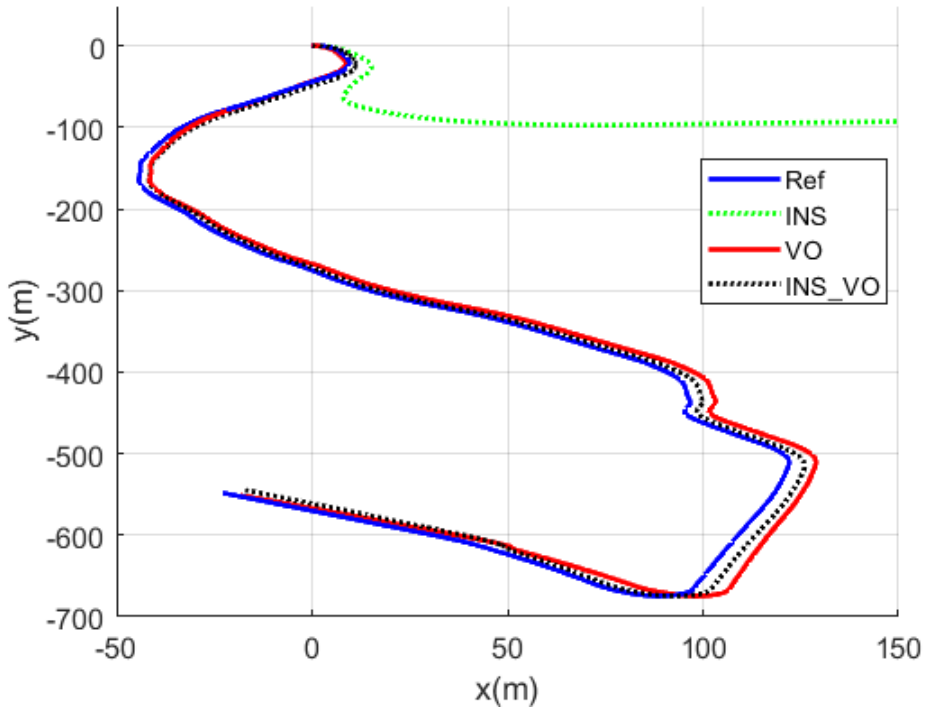

Figure 5-32 Trajectory plot comparison for INS/VO

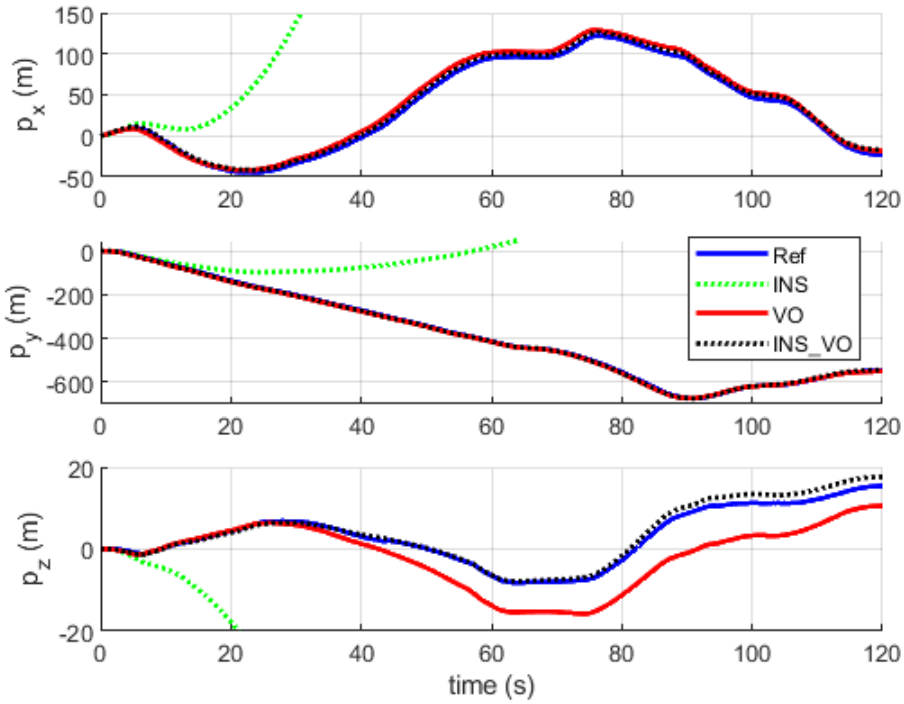

Figure 5-33 Position plot comparison for INS/VO

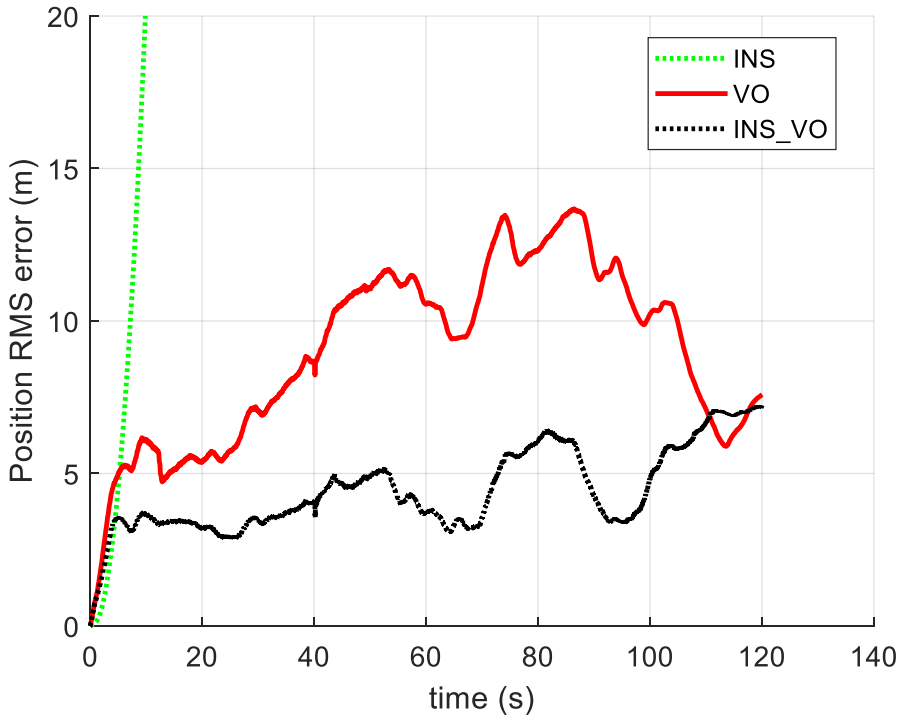

Figure 5-34 Position error comparison for INS/VO

The Figure 5-32 presents the trajectory comparison of INS, VO, and INS/VO with respect to the reference, Figure 5-33 shows the position estimate comparison, where p_x, p_y, p_z is the position estimate in x, y, z axis, and Figure 5-34 shows the position Root Mean Square (RMS) error plot. It can be seen that INS/VO follows the trajectory with higher accuracy as compared

to VO. As expected, the INS only solution rapidly accumulates error resulting in drift from the trajectory. The Root Mean Square (RMS) error for position averaged over the whole trajectory is given in the following table.

	Position RMSE
INS/VO	$\mathbf{4.63\ m}$
VO only	$9.42\ m$
INS only	$1454.5\ m$

Table 5-4 RMSE position comparison for INS/VO

The INS/VO integrated solution, compared to the VO only solution, reduces the position RMS error by 50.89%. This shows that the INS/VO fusion provides significant improvement in position estimates.

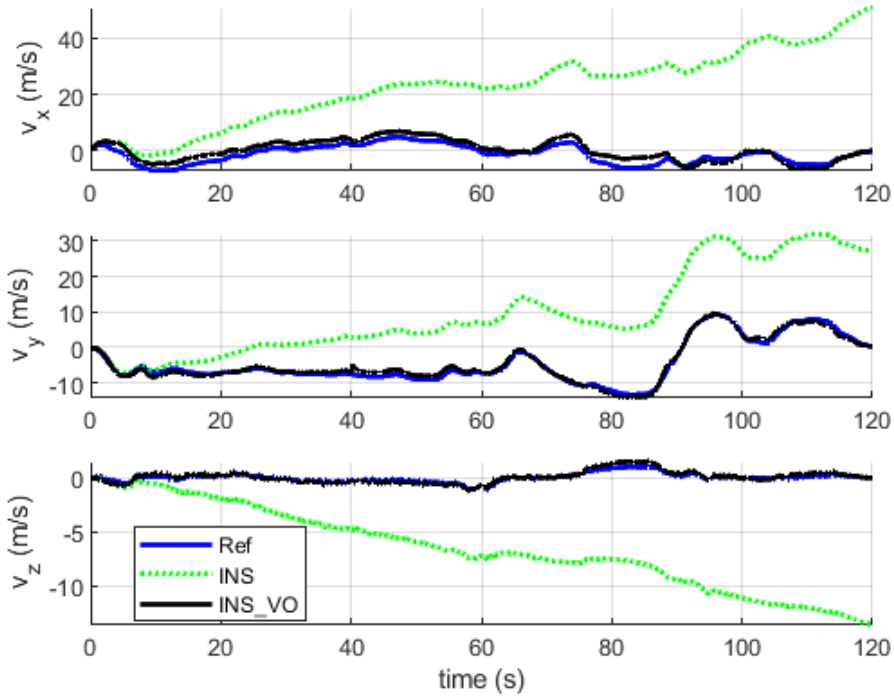

Figure 5-35 Velocity plot comparison for INS/VO

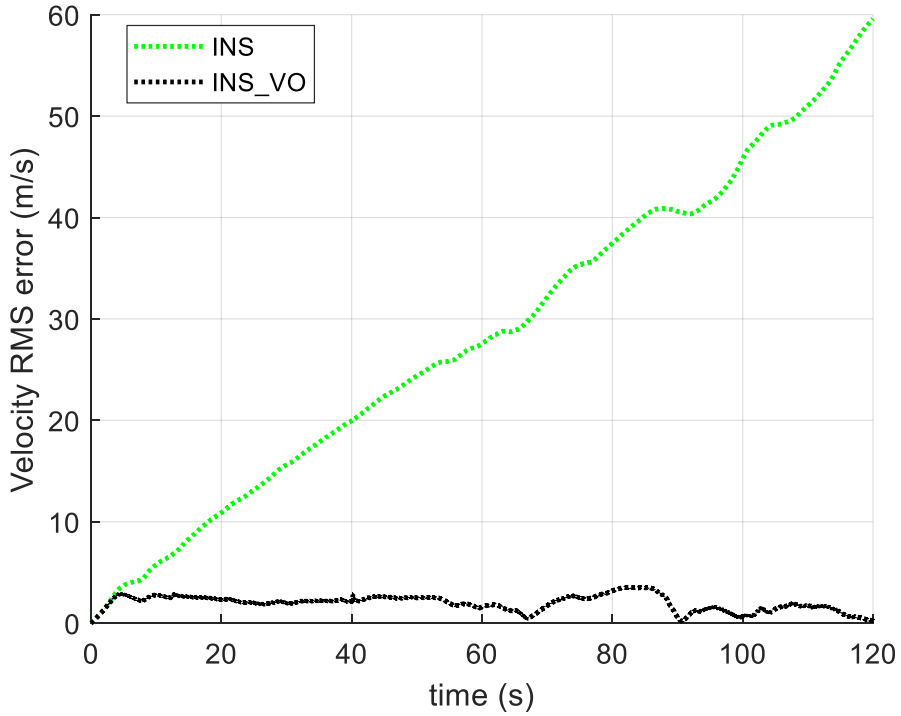

Figure 5-36 Velocity error comparison for INS/VO

Figure 5-35 shows the velocity comparison of INS standalone and INS/VO with respect to the reference where v_x, v_y, v_z is that velocity estimate in x, y, z axis, and Figure 5-36 shows the velocity Root Mean Square (RMS) error plot. The reason VO standalone is not compared is because VO provides only position estimates and attitude estimates. The position estimates can be differentiated over time to obtain velocity measurements, but they are subject to jitters due to uneven time steps and require smoothing. Smoothing, however, leads to data lag or even data loss in some cases, making it unsuitable as a reference for comparison. Therefore, the velocity comparison is carried out between INS and INS/VO. The Root Mean Square (RMS) error for velocity averaged over the whole trajectory is given in the following table.

	Velocity RMSE
INS/VO	$\mathbf{2.12\ m/s}$
VO only	-
INS only	$32.44\ m/s$

Table 5-5 RMSE velocity comparison for INS/VO

The INS/VO integrated solution, compared to the INS only solution, reduces the velocity RMS error by 93.45%. It can be seen that INS/VO integrated solution provides a significant improvement in the velocity state. The Figure 5-37 shows the attitude comparison, where ϕ, θ, ψ is the roll, pitch, and yaw angle about the x, y, z axis, Figure 5-38 shows the attitude Root Mean Square (RMS) error plot. The Root Mean Square (RMS) error for attitude averaged over the whole trajectory is given in Table 5-6.

	Attitude RMSE
INS/VO EKF	**0.50°**
VO only	4.49°
INS only	0.52°

Table 5-6 RMSE attitude comparison for INS/VO

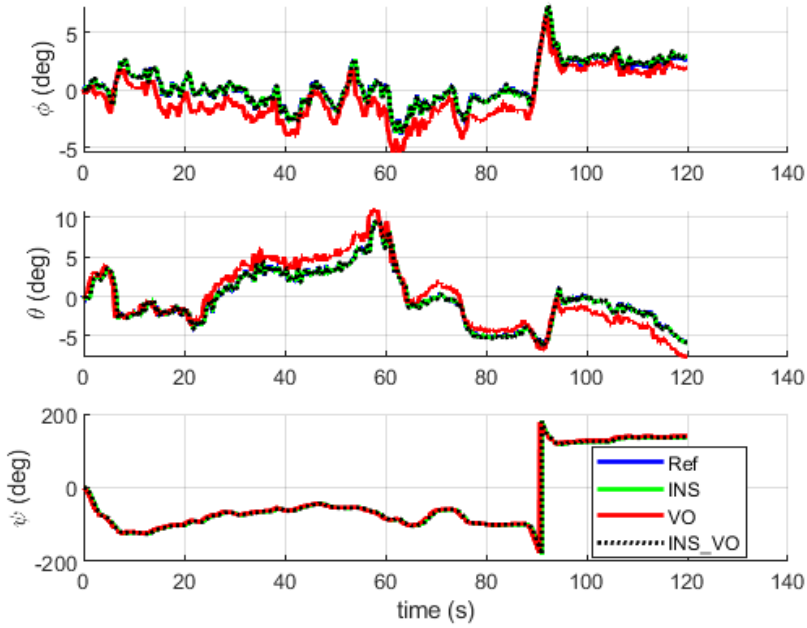

Figure 5-37 Attitude plot comparison for INS/VO

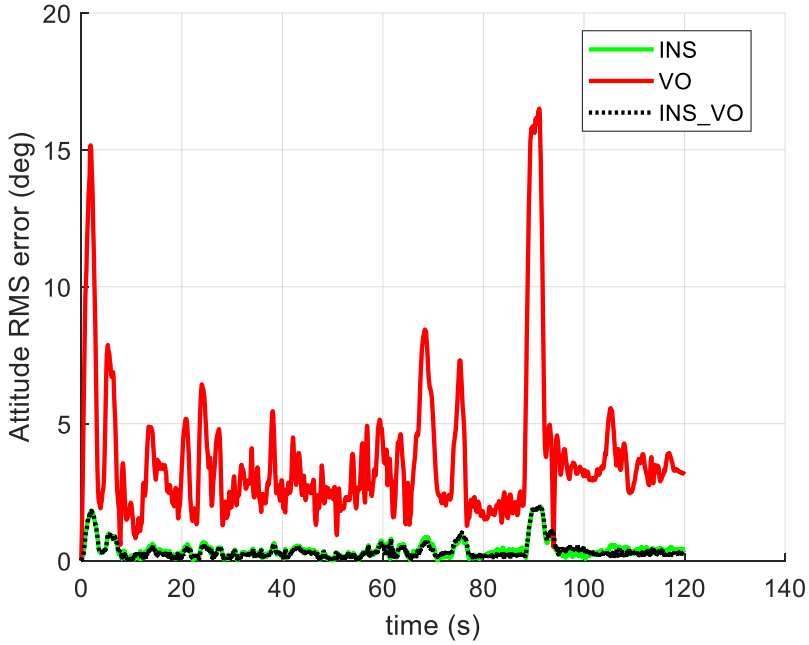

Figure 5-38 Attitude error comparison for INS/VO

The INS/VO algorithm, compared to VO only solution, reduces the attitude error by 88.86%. With respect to INS, the INS/VO algorithm shows 2.988 % improvement. It can be seen that INS/VO integrated solution provides a significant improvement in in attitude, when compared to VO estimate, and a minor improvement when compared to INS estimate.

5.3.2.2 INS/VO (with VO failure)

In this section a VO failure is simulated for 30 seconds starting from $t = 20s$ till $t = 50s$. The sources of VO failure can be direct sunlight to the camera lens, blocking of camera field of view by some large vehicle, and unfavorable lighting conditions. Utilizing the same design as in section 5.3 the solutions of INS/VO with VO failure and INS/VO without VO failure are computed. In the following figures (Figure 5-39 to Figure 5-43) '**Ref**' refers to ground truth reference, '**INS_VO_without_vo_failure**' refers to the INS/VO solution in normal case without any sensor failure, and '**INS_VO_with_vo_failure**' refers to the INS/VO solution with 30 second VO failure.

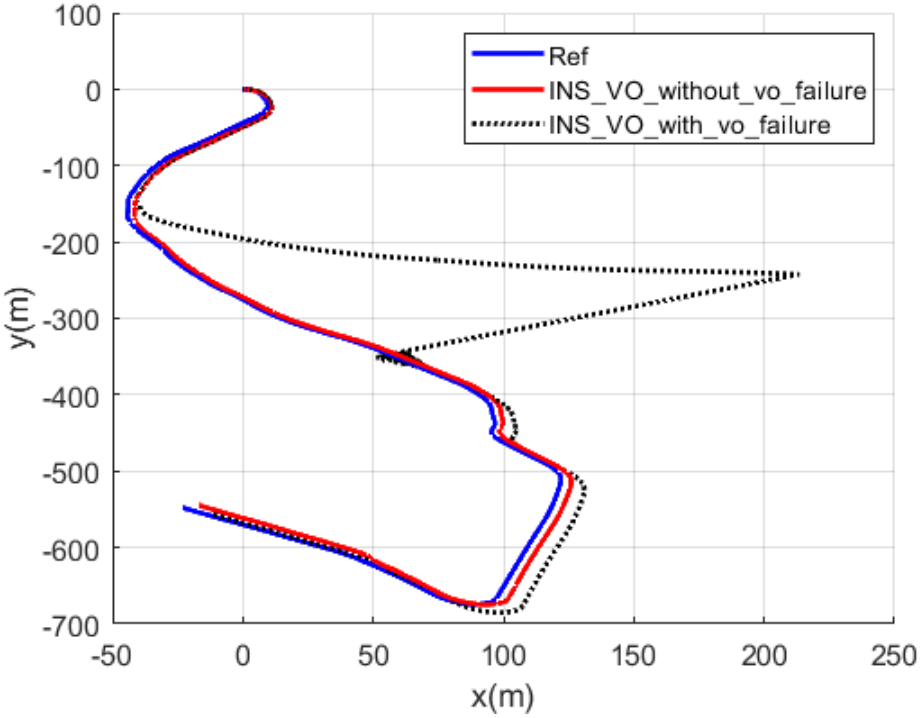

Figure 5-39 Trajectory plot comparison for INS/VO with VO failure

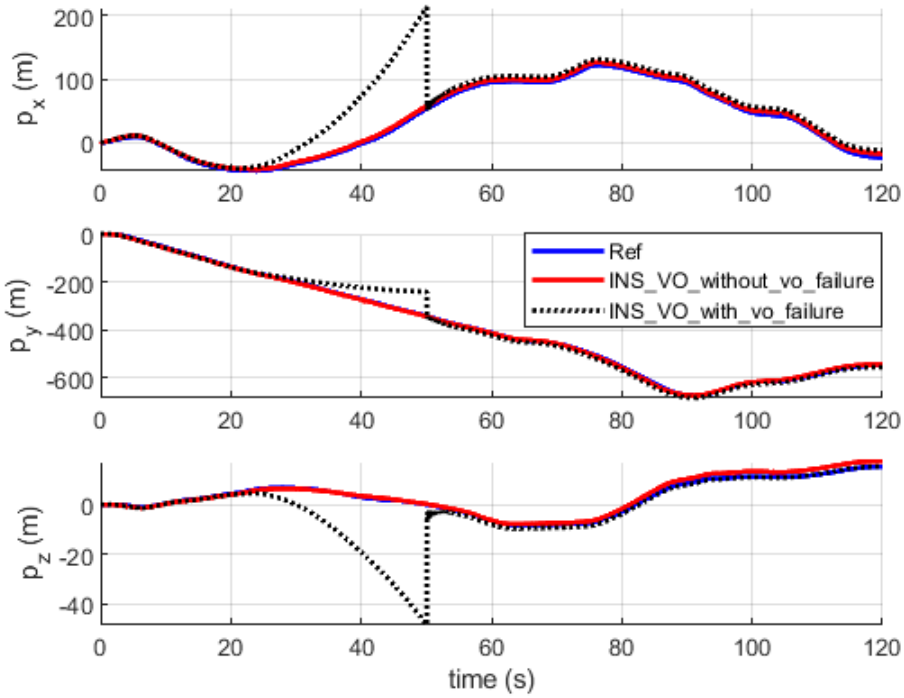

Figure 5-40 Position plot comparison for INS/VO with VO failure

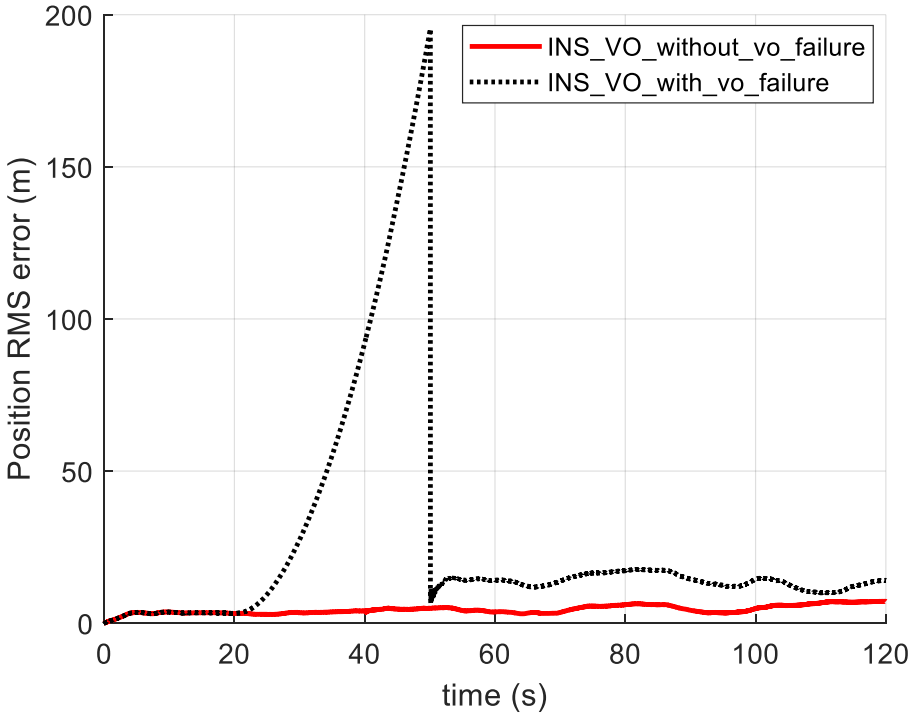

Figure 5-41 Position error plot comparison for INS/VO with VO failure

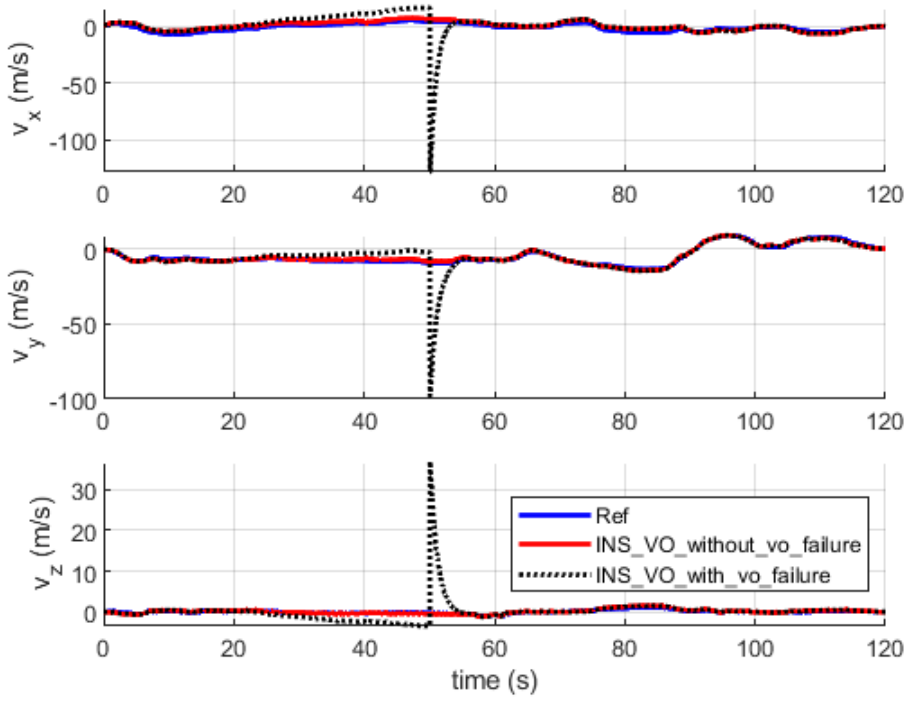

Figure 5-42 Velocity plot comparison for INS/VO with VO failure

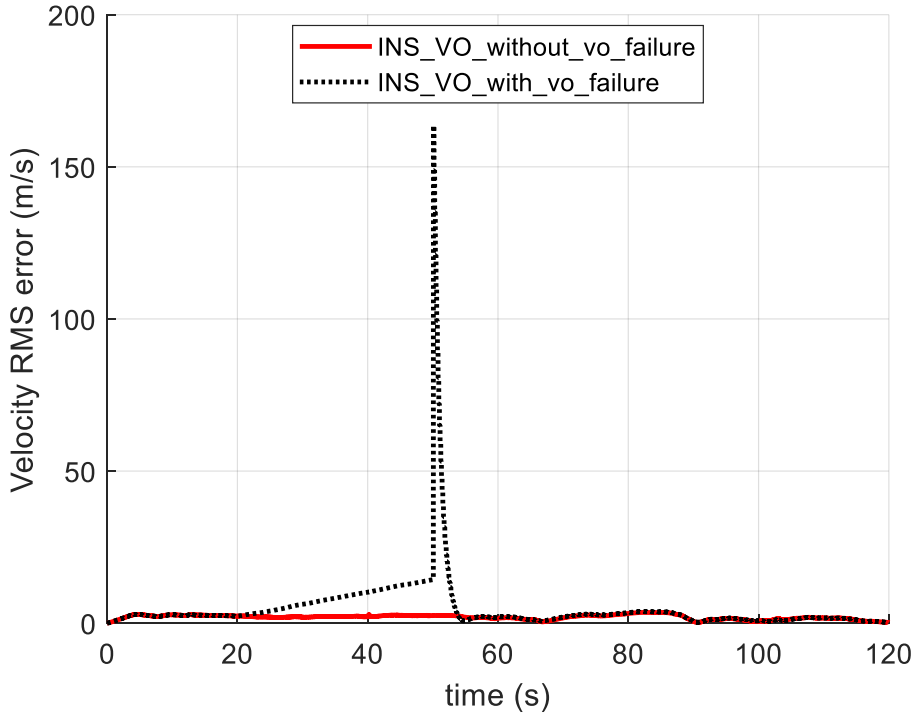

Figure 5-43 Velocity error plot comparison for INS/VO with VO failure

It can be seen that VO failure forces the navigation solution to utilize only the INS solution. As a result, the position and velocity state estimates quickly accumulate errors and start to drift from the reference trajectory and are not corrected until the VO data is available. This is shown in the position and velocity RMS error averaged over the whole trajectory in the following table. This result show the necessity to develop a navigation solution that is able to maintain accuracy in case of VO failure occurrence.

	Position RMSE	Velocity RMSE
INS/VO (With 30 sec VO failure)	$\mathbf{46.56\ m}$	$\mathbf{12.19\ m/s}$
INS/VO (Without VO failure)	$4.63\ m$	$2.12\ m/s$

Table 5-7 RMSE position and velocity comparison for VO failure case

It should be noted that in case of attitude, since the INS solution was reasonably accurate the Kalman filter gain gives more weightage to the INS solution resulting no significant changes in attitude.

5.4 INS/GPS Integrated System

5.4.1 Algorithm

This section presents the implementation for INS/GPS sensor fusion using Extended Kalman Filter (EKF). The INS/GPS integration structure is shown in Figure 5-44.

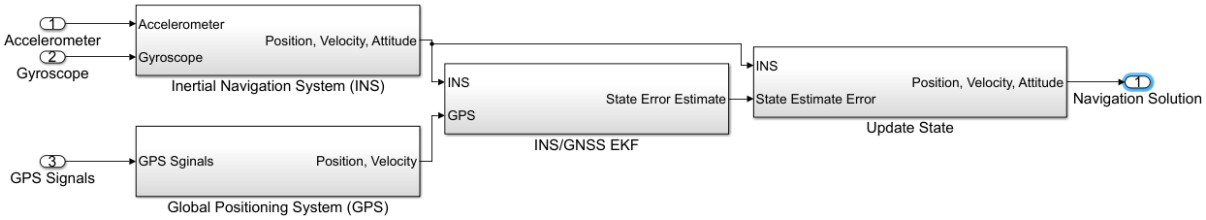

Figure 5-44 INS/GPS Integration Structure

The INS/GPS algorithm uses the navigation solution computed from INS as the state vector and state covariance prediction. The measurement vector is generated by GPS. The position, velocity, and attitude values from INS, and position and velocity values from GPS are input to the EKF block.

The EKF algorithm is described in Table 5-3. The state vector $\boldsymbol{x}$, system model f, system Jacobian F, state covariance matrix P, and system noise covariance matrix Q will be the same as defined in section 5.2.3. The remaining elements specific to INS/GPS integration are described in this section.

The sensor model function h_{gps} directly outputs the position and velocity states. The sensor model Jacobian H_{gps} is given in (5.60)

$$H_{gps} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{5.60}$$

The measurement vector z_{gps} is defined by (5.61).

$$z_{gps} = [p_{x_{gps}} \quad p_{y_{gps}} \quad p_{z_{gps}} \quad v_{x_{gps}} \quad v_{y_{gps}} \quad v_{z_{gps}}]^T \tag{5.61}$$

The sensor noise covariance matrix R_{gps} is defined using the GPS noise specifications provided in section 5.2.2.4. The INS/GPS EKF provides corrected error of state estimate. These are used to update the INS estimated states. The output provides the final estimate of position, velocity, and attitude.

5.4.2 Experimental Evaluation

In this section the INS/GPS algorithm is evaluated using the KITTI sequence 10 dataset presented in section 2.2. The evaluation is done for two cases. The first case is the normal case in which there is no sensor failure. This test allows observation of improvement in navigational accuracy when compared to INS only, and GPS only solution. The second case is the failure case. In this case, a GPS failure is simulated by blocking GPS input to the sensor fusion algorithm. This test allows observation of navigational accuracy deterioration in case of sensor failure.

5.4.2.1 INS/GPS (without GPS failure)

The INS/GPS design explained in section 5.4 is implemented and comparison is drawn between ground truth, standalone INS, standalone GPS, and INS/GPS integrated solution. In the following figures (Figure 5-45 to Figure 5-51), '**Ref**' refers to ground truth reference, '**INS'** refers to INS only solution, '**GPS'** refers to GPS only solution, and '**INS_GPS**' refers to the INS/GPS integrated solution.

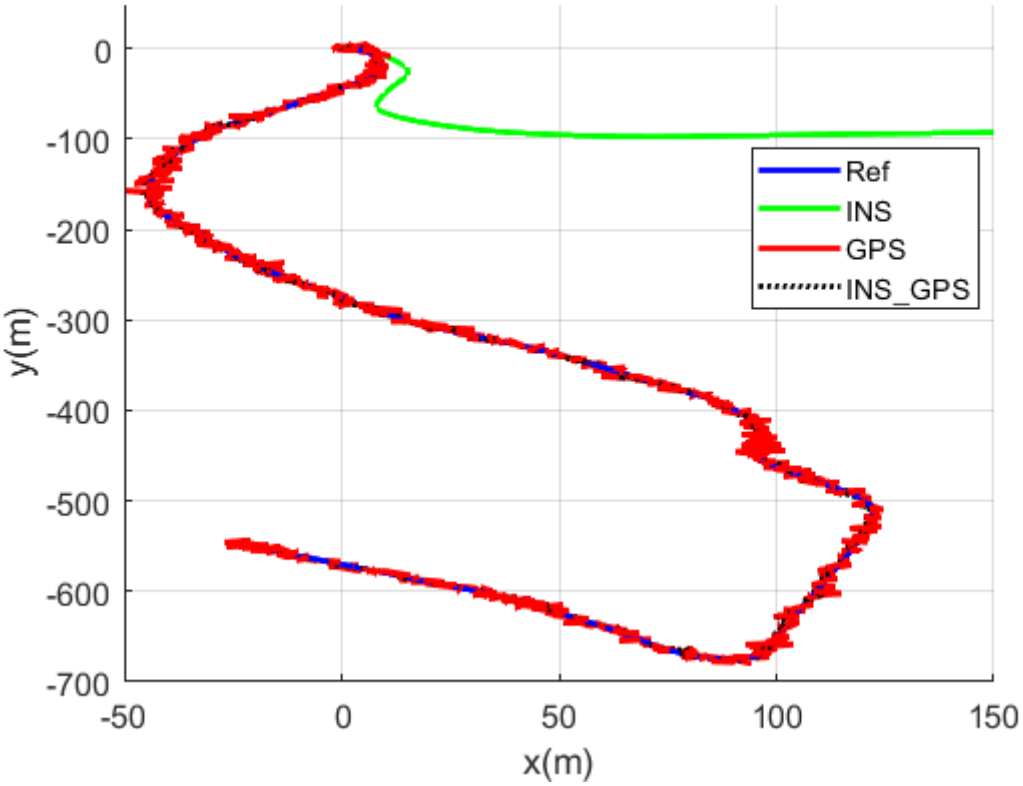

Figure 5-45 Trajectory plot comparison for INS/GPS

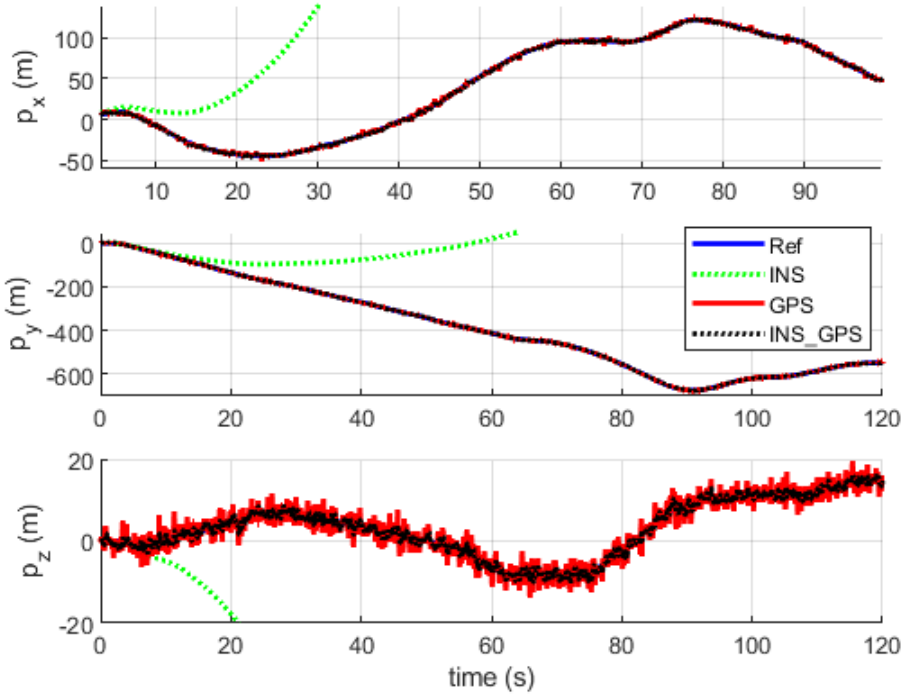

Figure 5-46 Position plot comparison for INS/GPS

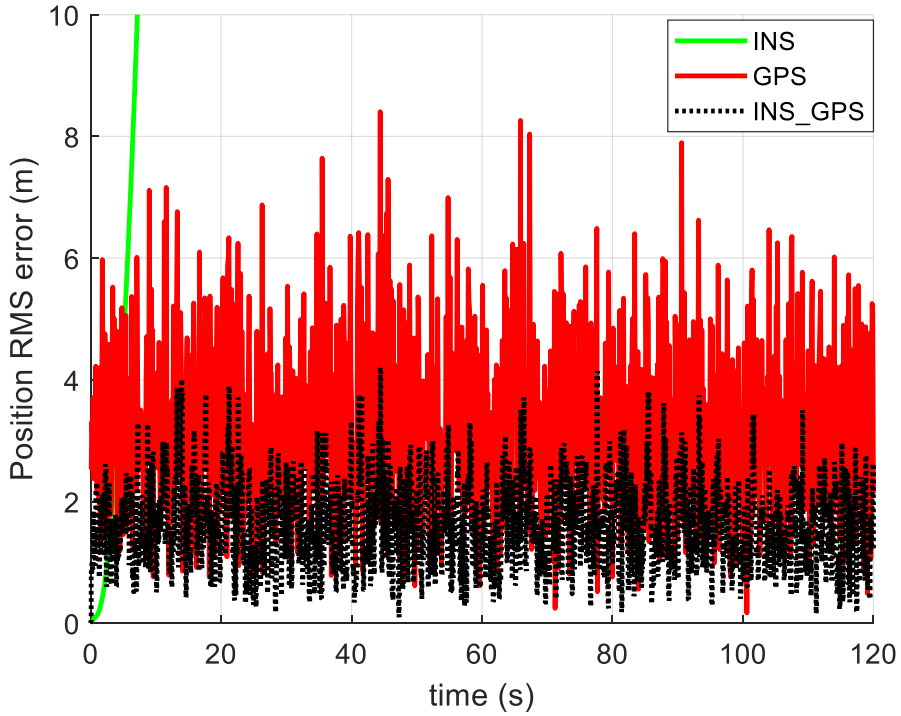

Figure 5-47 Position error comparison for INS/GPS

The Figure 5-45 presents the trajectory comparison of INS, GPS, and INS/GPS with respect to the reference, Figure 5-46 shows the position estimate comparison, where p_x, p_y, p_z is the position estimate in x, y, z axis, and Figure 5-47 shows the position Root Mean Square (RMS) error plot. It can be seen that INS/GPS follows the trajectory with higher accuracy as compared to GPS. The INS only solution is seen to rapidly accumulate error as expected. The Root Mean Square (RMS) error for position averaged over the whole trajectory is given in the following table.

	Position RMSE
INS/GPS	$\mathbf{1.77\ m}$
GPS only	$3.52\ m$
INS only	$1454.5\ m$

Table 5-8 RMSE position comparison for INS/GPS

The INS/GPS integrated solution, compared to the GPS only solution, reduces the position RMS error by 49.76%. This shows that the INS/GPS fusion provides significant improvement in position estimates.

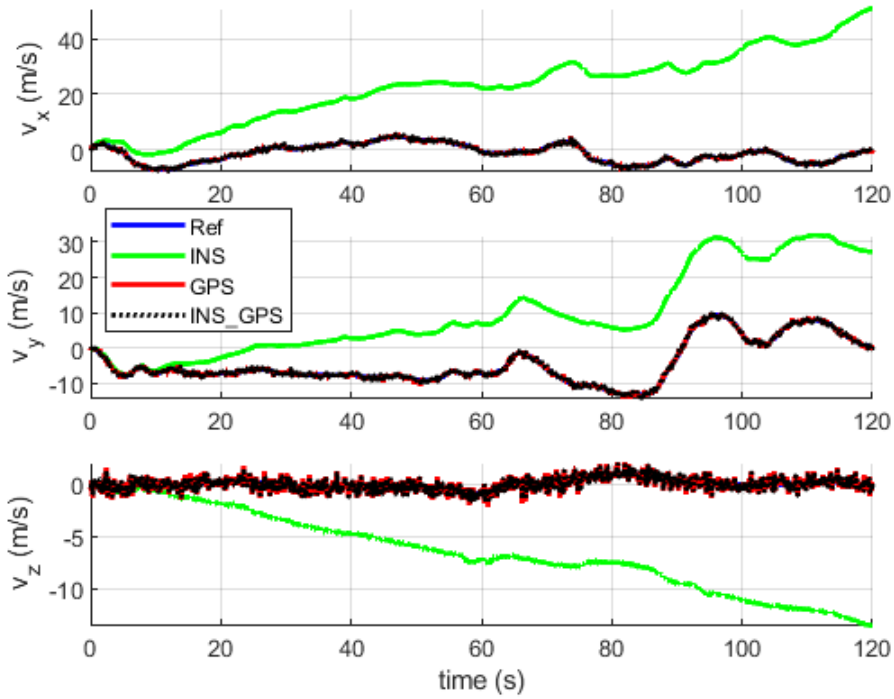

Figure 5-48 Velocity plot comparison for INS/GPS

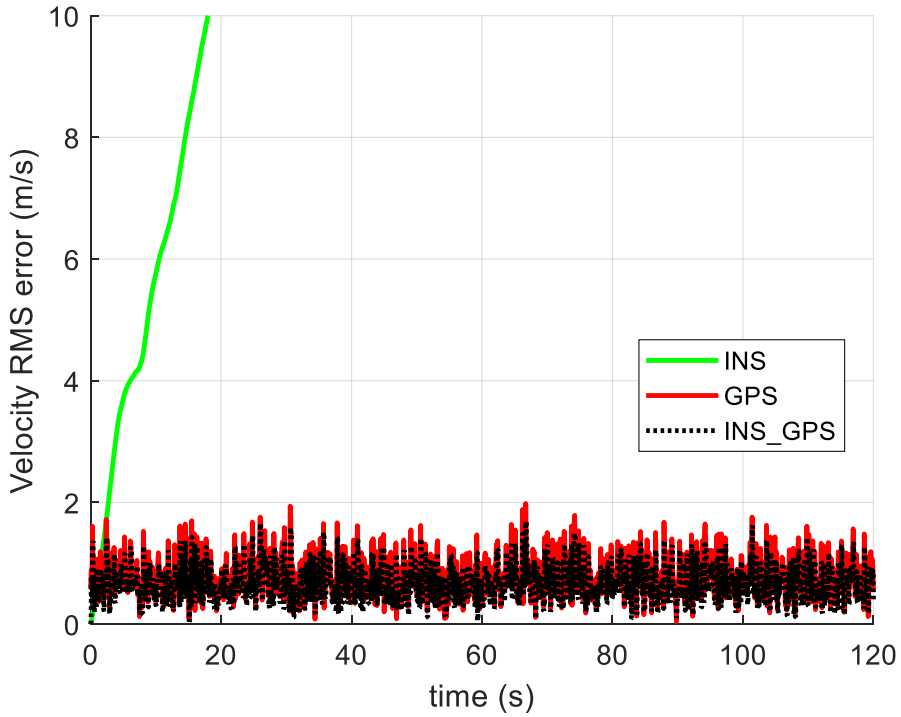

Figure 5-49 Velocity error comparison for INS/GPS

Figure 5-48 shows the velocity comparison of INS standalone, GPS standalone, and INS/GPS integrated solution with respect to the reference where v_x, v_y, v_z is that velocity estimate in x, y, z axis, and Figure 5-49 shows the velocity Root Mean Square (RMS) error plot. The INS results in Figure 5-49 are the same as were shown in Figure 5-23. The y-axis of the plot is

limited here to better observe the difference between GPS and INS/GPS estimates. The standalone INS solution shows a higher degree of error because it relies on the previous state estimates to obtain the new state. Due to the noisy sensor data, the state estimates computed by integration during INS computation are erroneous. These erroneous estimates are used to obtain the new estimates resulting in an accumulation of error. If there is no correction, then the INS solution diverges from the ground truth. This phenomenon is observed in Figure 5-49. The Root Mean Square (RMS) error for velocity averaged over the whole trajectory is given in the following table.

	Velocity RMSE
INS/GPS	$\mathbf{0.73\ m/s}$
GPS only	$0.86\ m/s$
INS only	$32.44\ m/s$

Table 5-9 RMSE velocity comparison for INS/GPS

For velocity, it can be seen that INS/GPS provides a minor improvement of 15.24% when comparing to GPS only solution, and significant improvement of 97.74% when compared to INS only solution.

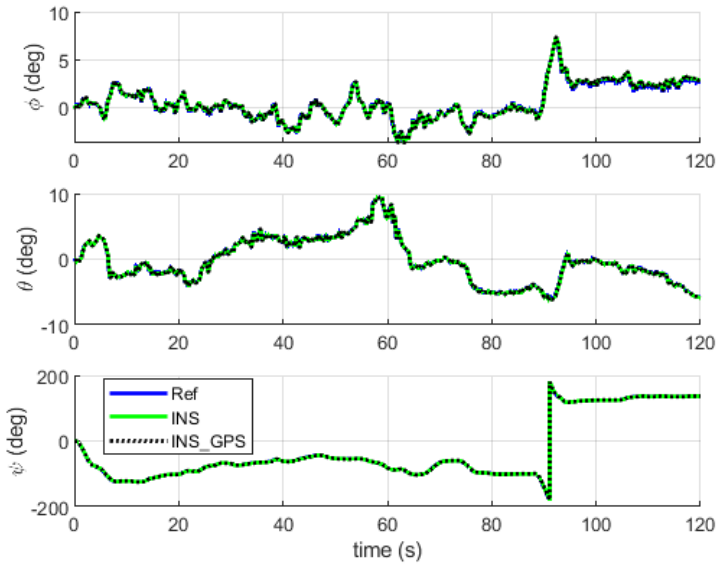

Figure 5-50 Attitude plot comparison for INS/GPS

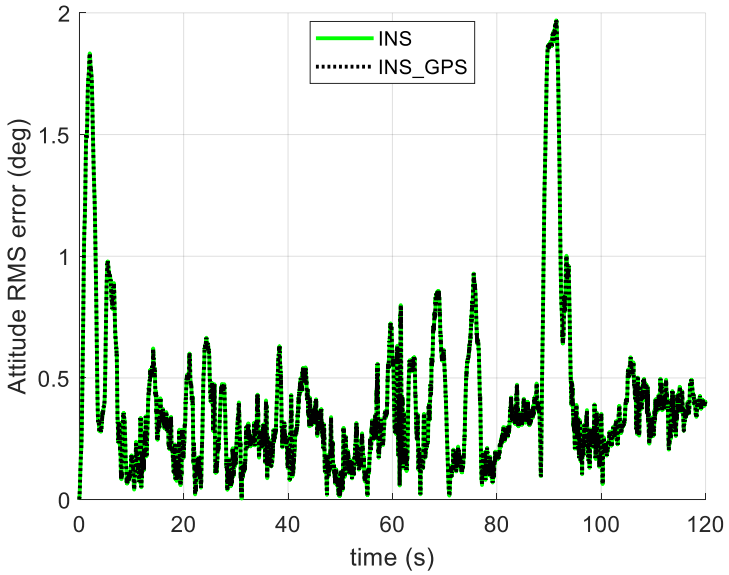

Figure 5-51 Attitude error comparison for INS/GPS

The Figure 5-50 shows the attitude comparison, where ϕ, θ, ψ is the roll, pitch, and yaw angle about the x, y, z axis Figure 5-51 shows the attitude Root Mean Square (RMS) error plot. Since GPS does not provide attitude information, therefore the comparison is done between the INS only solution and INS/GPS solution. The Root Mean Square (RMS) error for attitude averaged over the whole trajectory is given in the following table.

	Attitude RMSE
INS/GPS	$\mathbf{0.51497°}$
GPS only	–
INS only	0.51501°

Table 5-10 RMSE attitude comparison for INS/GPS

It can be seen that since the INS only solution for attitude already has a high accuracy, the INS/GPS solution does not provide any significant of improvement in the attitude states. The INS/GPS algorithm, compared to INS only solution, provides less than 0.01% reduction in attitude errors.

5.4.2.2 INS/GPS (with GPS failure)

In this section a GPS failure is simulated for 30 seconds starting from $t = 20s$ till $t = 50s$. GPS failure can be caused by blockage of line of sight of receiver and satellite due to tall trees, buildings, underground tunnels etc. Utilizing the same design as in section 5.4 the solutions of INS/GPS with GPS failure and INS/GPS without GPS failure are computed. In the following figures (Figure 5-52 to Figure 5-56) '**Ref**' refers to ground truth reference, '**INS_GPS_without_gps_failure**' refers to the INS/GPS solution in normal case without any sensor failure, and '**INS_GPS_with_gps_failure**' refers to the INS/GPS solution with 30 second GPS failure.

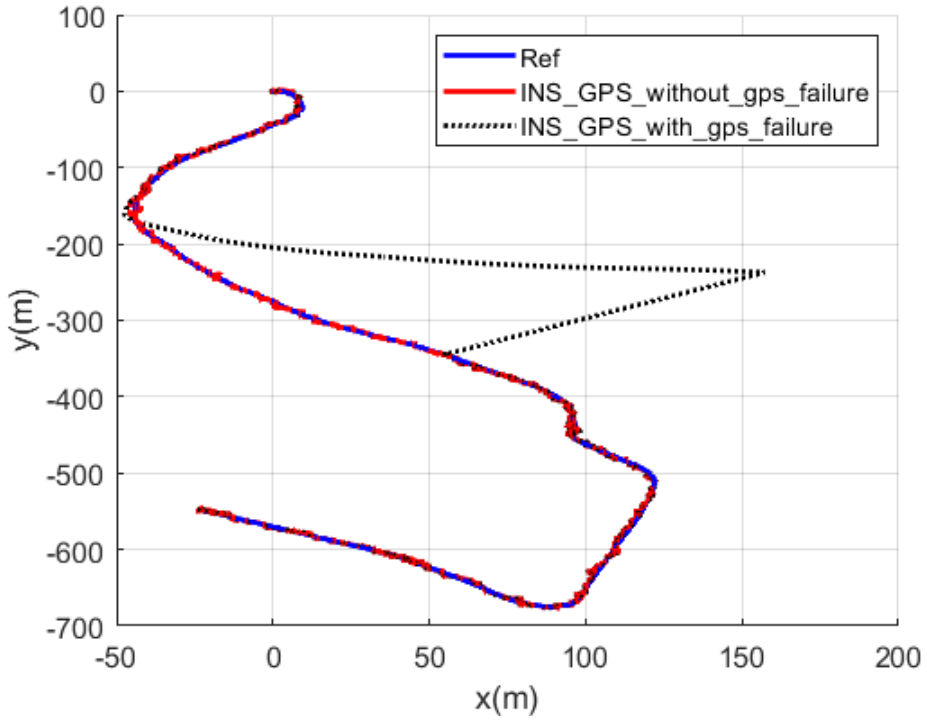

Figure 5-52 Trajectory plot comparison for INS/GPS with GPS failure

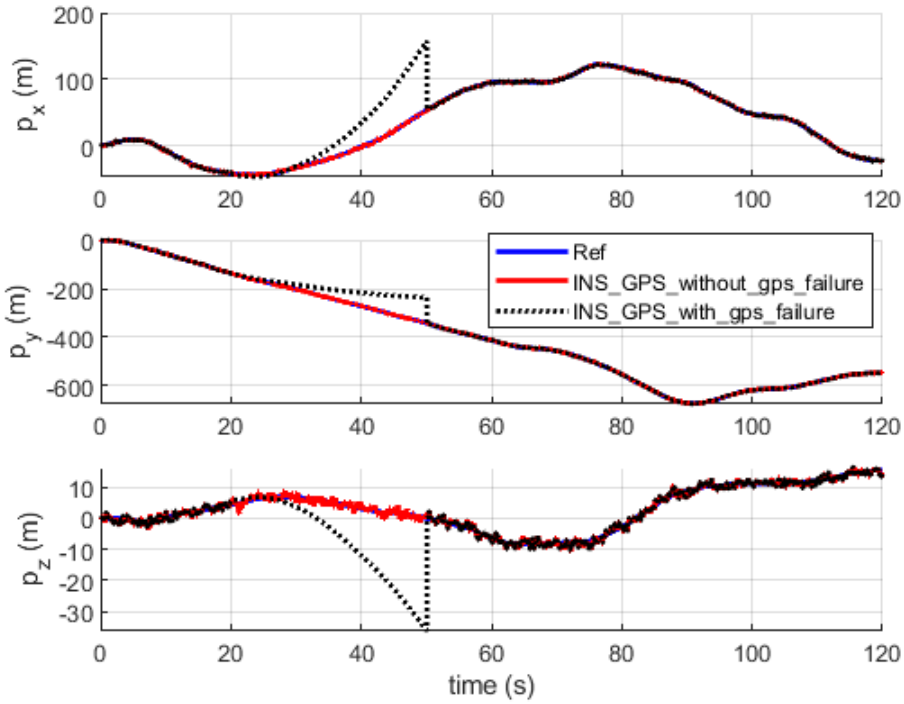

Figure 5-53 Position plot comparison for INS/GPS with GPS failure

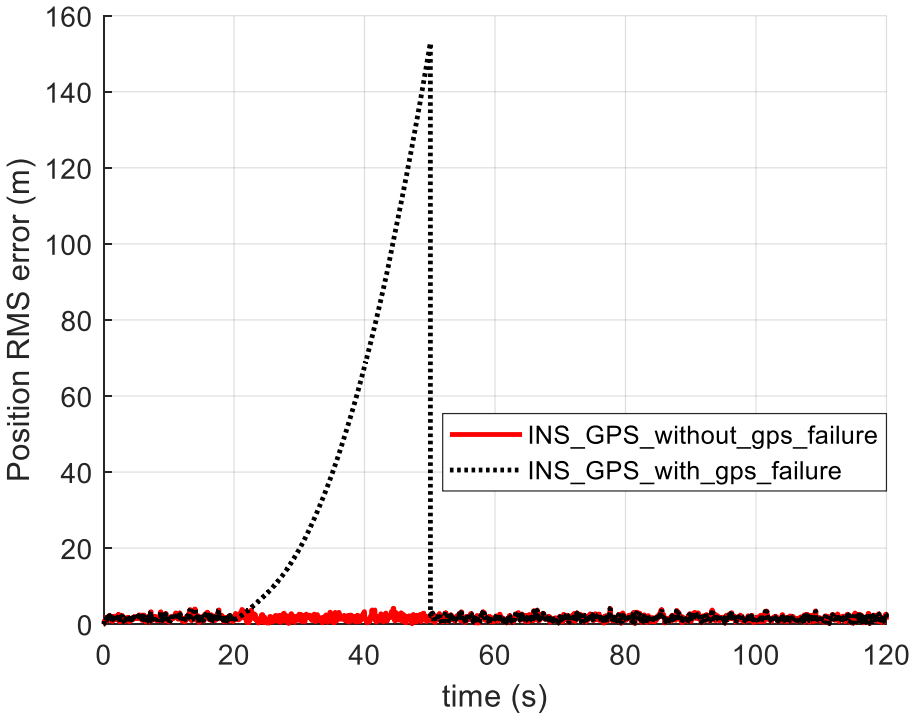

Figure 5-54 Position error plot comparison for INS/GPS with GPS failure

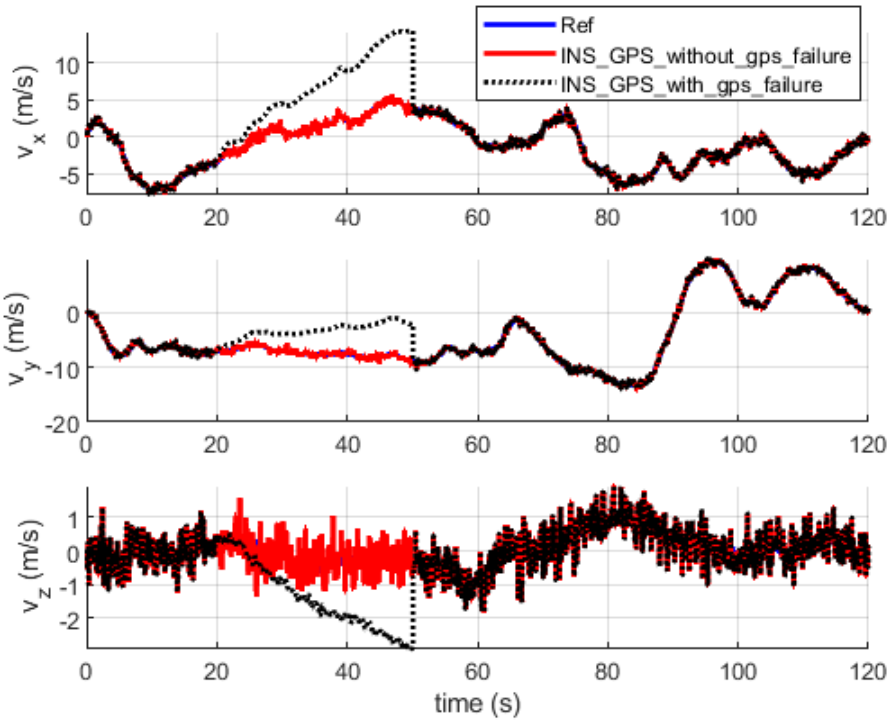

Figure 5-55 Velocity plot comparison for INS/GPS with GPS failure

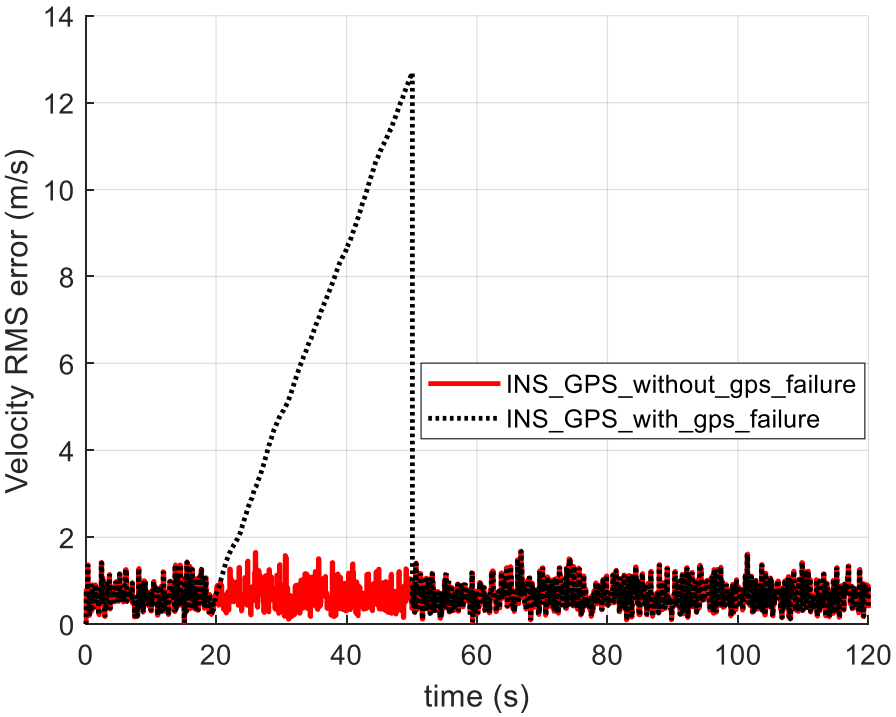

Figure 5-56 Velocity error plot comparison for INS/GPS with GPS failure

It can be seen that GPS failure forces the navigation solution to utilize only the INS solution. As a result, the position and velocity state estimates quickly accumulate errors and start to drift from the reference trajectory and are not corrected until the GPS data is available. This is shown in the position and velocity RMS error averaged over the whole trajectory in the following

table. This result shows the necessity to develop a navigation solution that is able to maintain accuracy in case of GPS failure occurrence.

	Position RMSE	Velocity RMSE
INS/GPS (With 30 sec GPS failure)	$\mathbf{34.45\ m}$	$\mathbf{3.85\ m/s}$
INS/GPS (Without GPS failure)	$1.77\ m$	$0.73\ m/s$

Table 5-11 RMSE position and velocity comparison for GPS failure case

It should be noted that in case of attitude, since the INS solution was reasonably accurate the Kalman filter gain gives more weightage to the INS solution resulting no significant changes in attitude.

5.5 INS/VO/GPS – Dual Extended Kalman Filter

This section presents the structure of an integrated system termed Dual Extended Kalman Filter (DEKF) for the implementation INS/VO/GPS sensor fusion.

The idea behind DEKF is to utilize both VO and GPS measurements as means to correct the INS state estimates. While the INS/GPS solution (presented in section 5.4.2.1) provides a higher accuracy when compared to the INS/VO solution (presented in section 5.3.2.1) due to higher accuracy of GPS, however the solution quickly degrades to high error levels when GPS failure occurs (presented in section 5.4.2.2). GPS failure is a common occurrence which can be triggered by surrounding tall buildings, trees, underground/ tunnel environment. As a result, the INS/GPS navigation solution is not completely reliable. Furthermore, it was observed that while INS/VO implementation significantly reduces the navigation errors compared to INS only and VO only solutions, however, in case of VO failure, the solution rapidly accumulates error, resulting in a large drift (presented in section 5.3.2.2). Similar to the case of GPS failure, VO failure is not an uncommon occurrence, and it can be caused due to low light conditions, low number of matched inliers, direct sunlight, and camera field of view blocked by large

vehicle. Hence, using solely INS/VO is also not recommended for cases outside of test environments. In order to overcome these problems, both VO and GPS solutions are utilized to develop INS/VO/GPS integrated system through DEKF.

5.5.1 Algorithm

The input from accelerometer and gyroscope is used to compute the INS state estimates (section 5.2.1). The inputs images from left and right cameras are used to compute the VO state estimates (section 4.4), VO uncertainty (section 5.3), and an indicator for occurrence of VO failure. This flag is true if the VO algorithm was unable to converge to a solution. Otherwise, in presence of valid VO solution, the flag is set to false. The VO estimates and INS estimates are forwarded to the INS/VO EKF algorithm described in section 5.3. The output is the state estimate error, which is used to correct the INS solution and provide an updated estimate of position, velocity, and attitude. These are provided as input to the INS/GPS EKF. At this point, the GPS solution is obtained and also fed into INS/GPS EKF. In case the VO failure indicator is true, the original INS estimates are directly provided to INS/GPS EKF rather than the updated states. An indicator is updated which defines if the GPS is connected to less than 4 satellites is defined. A true value of this indicator means that due to signal blockage the GPS is unable to provide a reliable position estimate. The INS/GPS EKF, as explained in section 5.4, provides the state estimate errors. These are used to update the INS/VO state estimates. The result is the final estimate of position, velocity, and attitude. In case of GPS failure however, the INS/VO state estimates are not updated, and directly presented as the final state estimates. The final state estimates are fed back to the VO estimates with a time step delay. This feedback is used to update the VO pose. This is described in section 5.3.

The architecture of INS/VO/GPS using DEKF is presented in Figure 5-57.

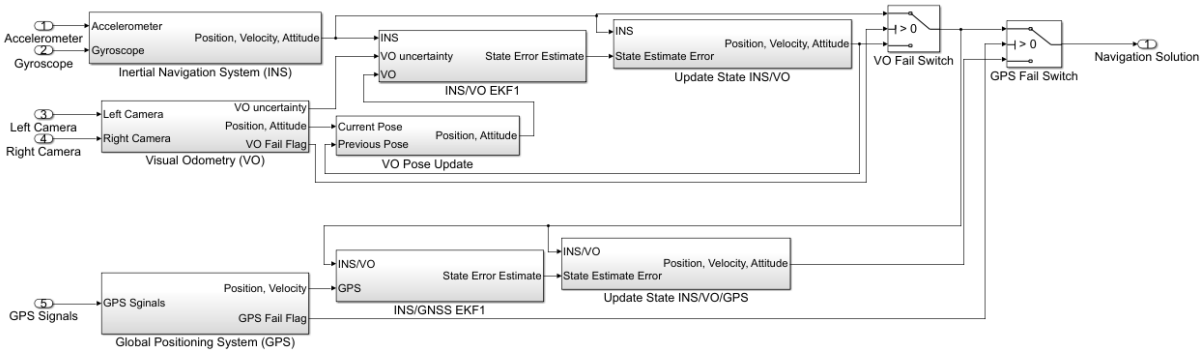

Figure 5-57 General structure of INS/VO/GPS Dual EKF integrated system

A comprehensive block diagram for INS/VO/GPS Dual Extended Kalman Filter is presented in Figure 5-58.

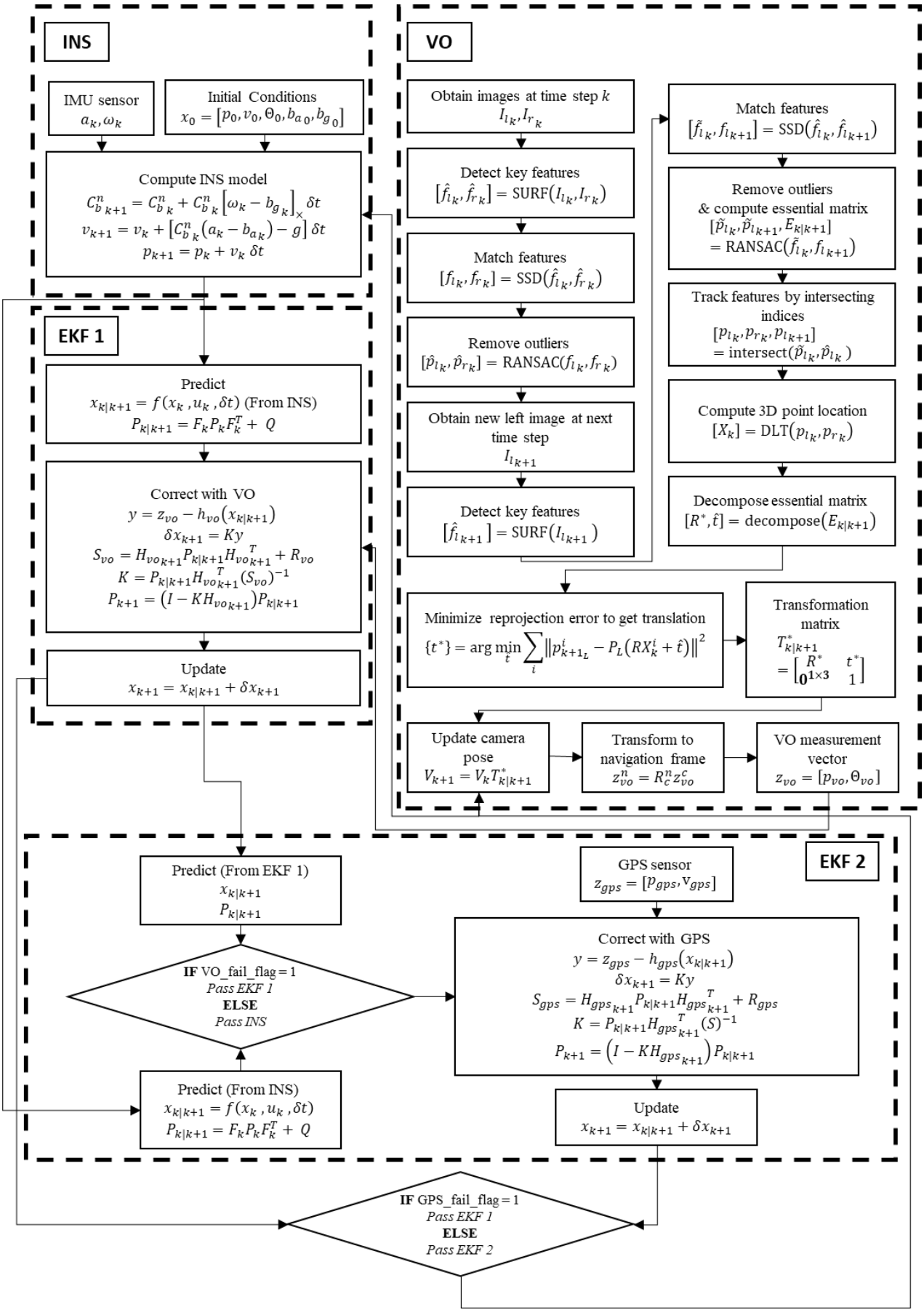

Figure 5-58 Comprehensive block diagram for INS/VO/GPS Dual EKF system

5.5.2 Experimental Evaluation

In this section the INS/VO/GPS DEKF algorithm is evaluated using the KITTI sequence 10 dataset presented in section 2.2. In section 5.3.2.2 and 5.4.2.2 it is shown that the INS/VO and INS/GPS integrated solutions lose accuracy in case of sensor failure. The INS/VO/GPS integrated system provides a robust solution where both VO and GPS are utilized. The evaluation is done for three cases. The first case is VO failure case. In this case, VO failure is simulated by blocking its input to the sensor fusion algorithm, and the INS/VO/GPS is compared with INS/VO. The second case is the GPS failure case. In this case, a GPS failure is simulated by blocking GPS input to the sensor fusion algorithm, and the INS/VO/GPS is compared with INS/GPS. The third case is the normal case in which there is no sensor failure. In this case INS/VO/GPS is compared with standalone INS, standalone VO, and standalone GPS solution.

5.5.2.1 INS/VO/GPS (with VO failure)

In this section a VO failure is simulated for 30 seconds starting from $t = 20s$ till $t = 50s$. INS/VO integrated system is implemented as explained in section 5.3, and INS/VO/GPS is implemented as explained in section 5.5. In the following figures (Figure 5-59 to Figure 5-63) '**Ref**' refers to ground truth reference, '**INS_VO_with_vo_failure**' refers to the INS/VO solution with 30 second VO failure, and '**INS_VO_GPS_with_vo_failure**' refers to the INS/VO/GPS solution with the same 30 second VO failure.

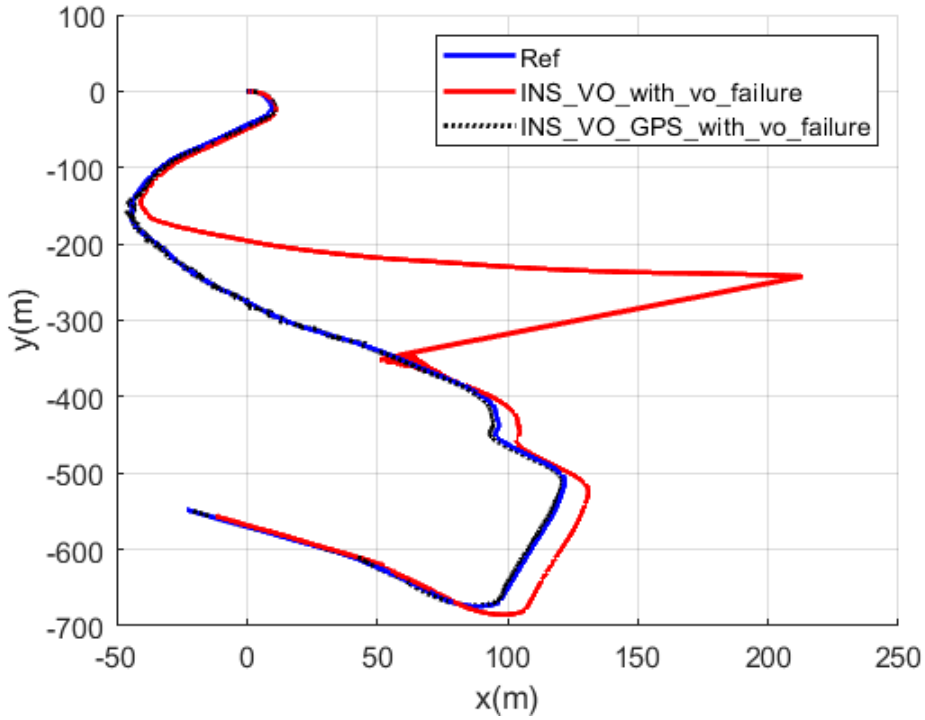

Figure 5-59 Trajectory plot comparison for INS/VO/GPS with VO failure

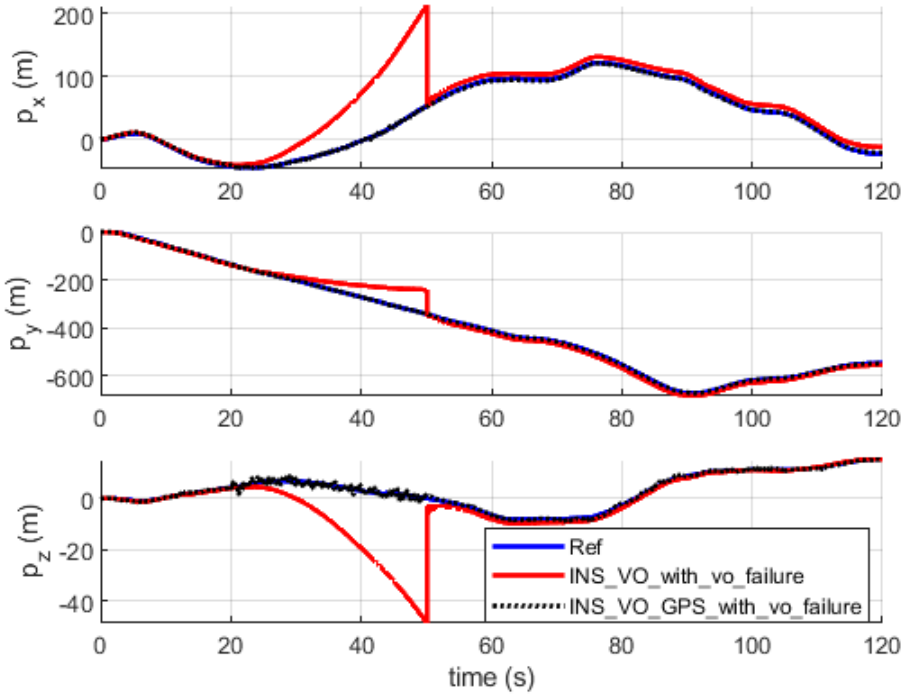

Figure 5-60 Position plot comparison for INS/VO/GPS with VO failure

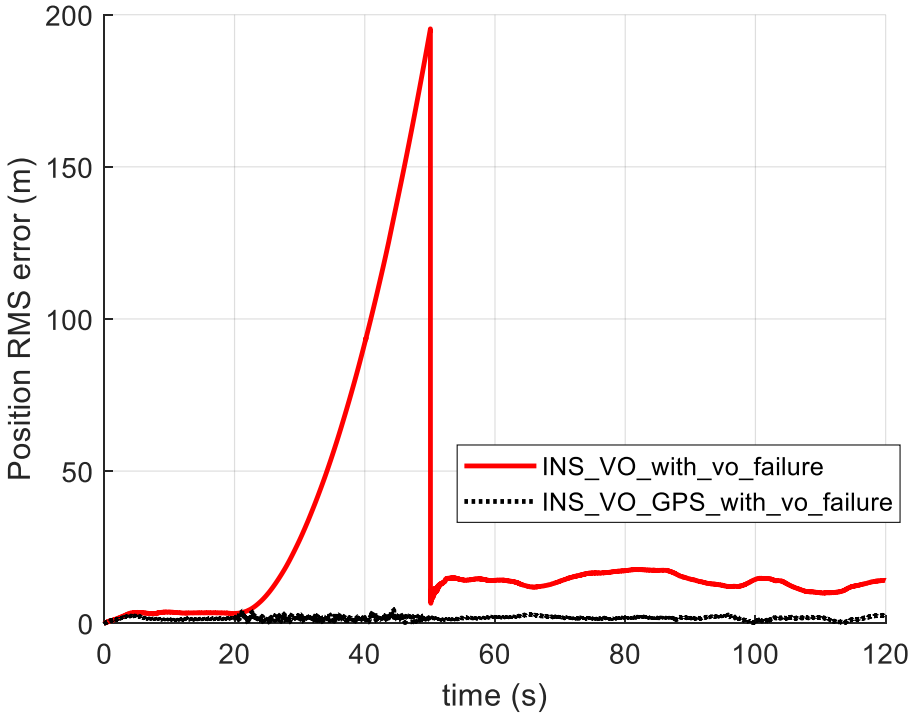

Figure 5-61 Position error plot comparison for INS/VO/GPS with VO failure

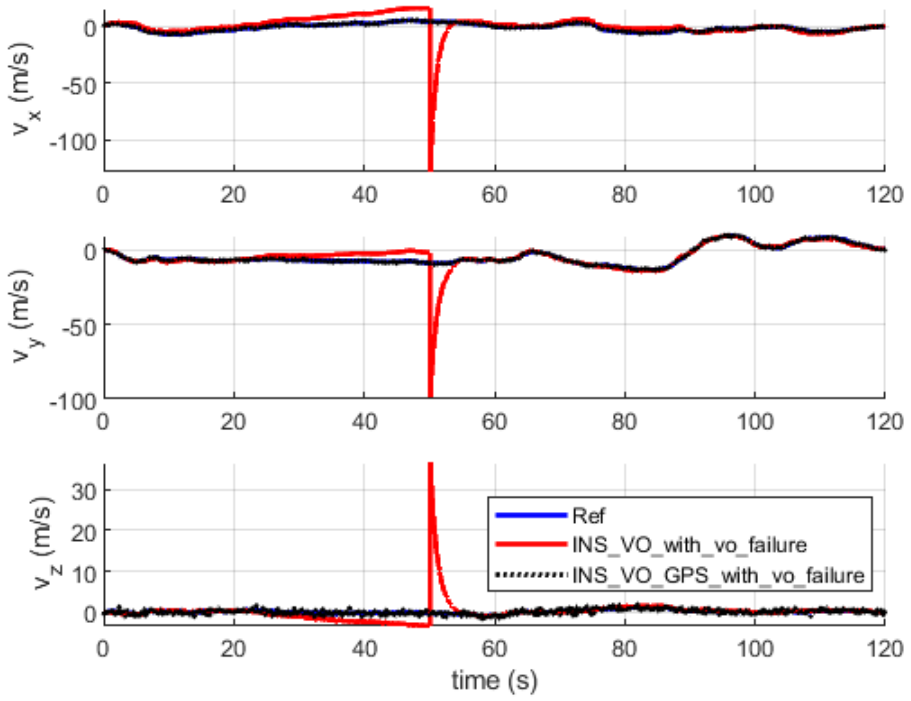

Figure 5-62 Velocity plot comparison for INS/VO/GPS with VO failure

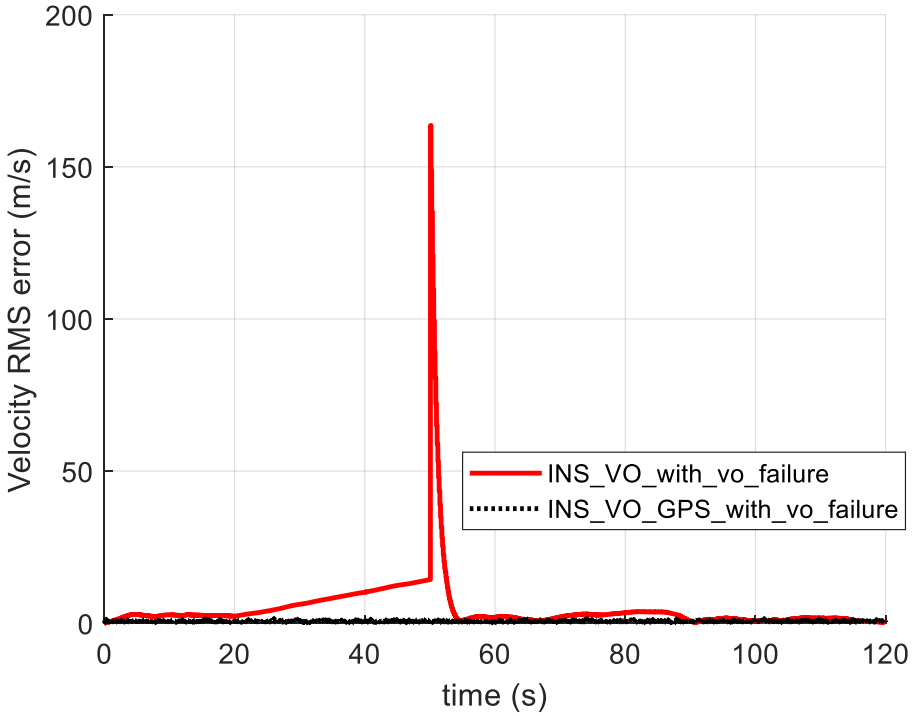

Figure 5-63 Velocity error plot comparison for INS/VO/GPS with VO failure

It can be seen that VO failure forces the INS/VO navigation solution to utilize only the INS solution and as a result, the position and velocity state estimates quickly accumulate errors and start to drift from the reference trajectory and are not corrected until the VO data is available. However, in case of INS/VO/GPS the effect of VO failure is negated due to presence of GPS data, hence the INS solution continues to be corrected using GPS measurements. This improvement can also be seen in the position and velocity RMS error averaged over the whole trajectory in the following table.

	Position RMSE	Velocity RMSE
INS/VO/GPS (With 30 sec VO failure)	$\mathbf{1.76\ m}$	$\mathbf{0.74\ m/s}$
INS/VO (With 30 sec VO failure)	$46.56\ m$	$12.19\ m/s$

Table 5-12 RMSE position and velocity comparison for VO failure case

These results validate the DEKF algorithm for INS/VO/GPS fusion by maintaining accuracy in case of VO failure. The next section will investigate its performance in case of GPS failure.

5.5.2.2 INS/VO/GPS (with GPS failure)

In this section a GPS failure is simulated for 30 seconds starting from $t = 20s$ till $t = 50s$. INS/GPS integrated system is implemented as explained in section 5.4, and INS/VO/GPS is implemented as explained in section 5.5. In the following figures (Figure 5-64 to Figure 5-68) '**Ref**' refers to ground truth reference, '**INS_GPS_with_gps_failure**' refers to the INS/GPS solution with 30 second GPS failure, and '**INS_VO_GPS_with_gps_failure**' refers to the INS/VO/GPS solution with the same 30 second GPS failure.

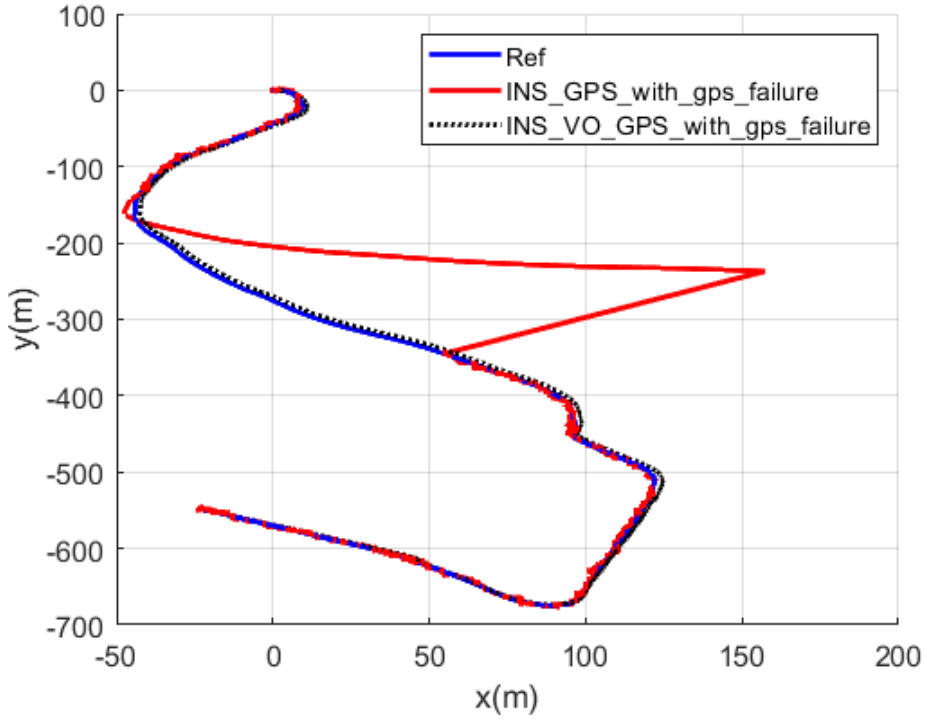

Figure 5-64 Trajectory plot comparison for INS/VO/GPS with GPS failure

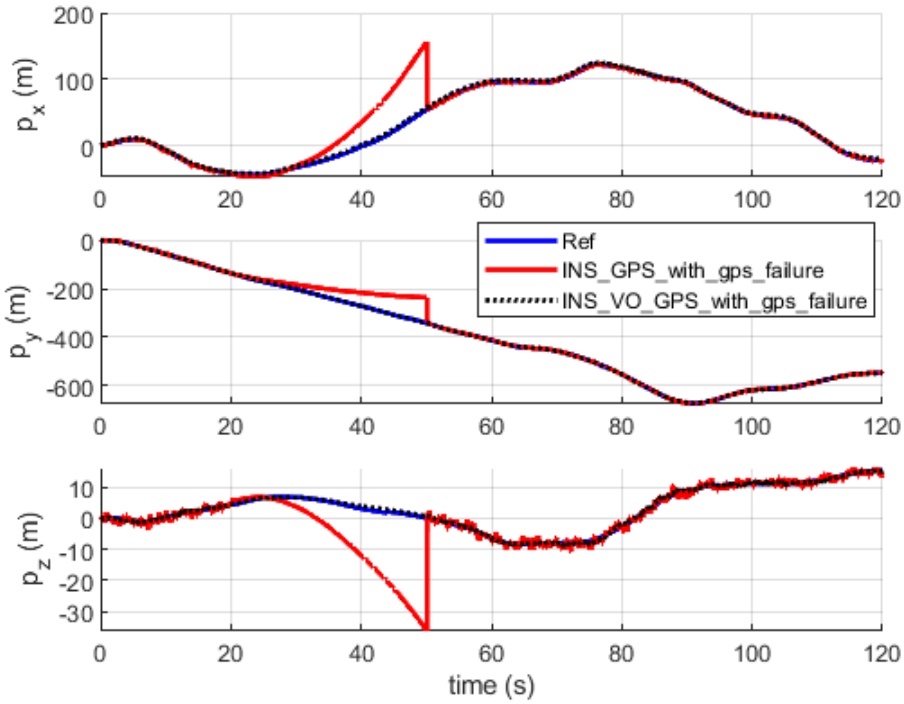

Figure 5-65 Position plot comparison for INS/VO/GPS with GPS failure

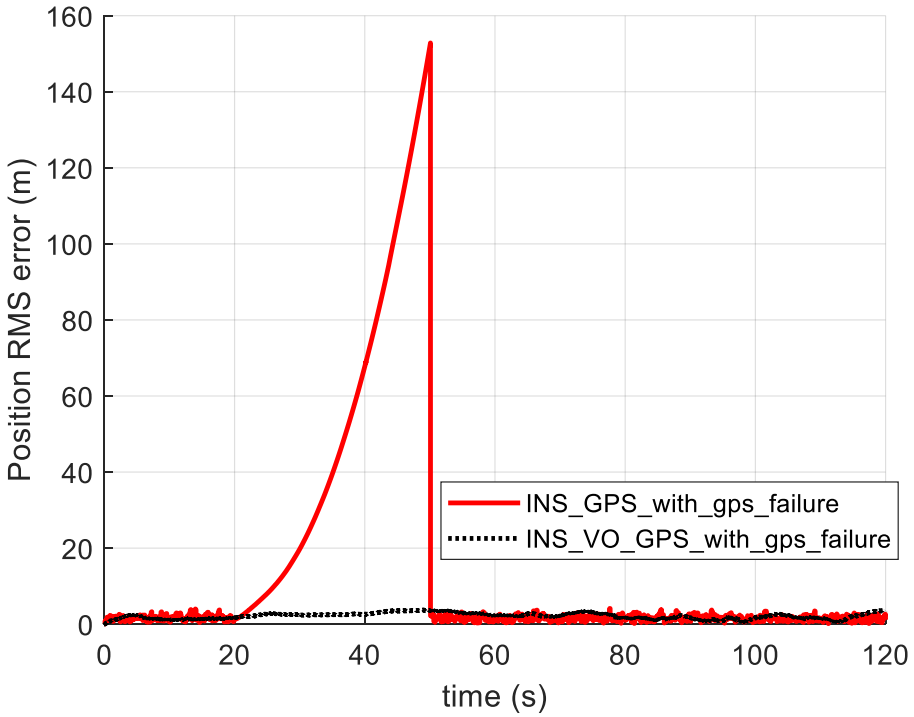

Figure 5-66 Position error plot comparison for INS/VO/GPS with GPS failure

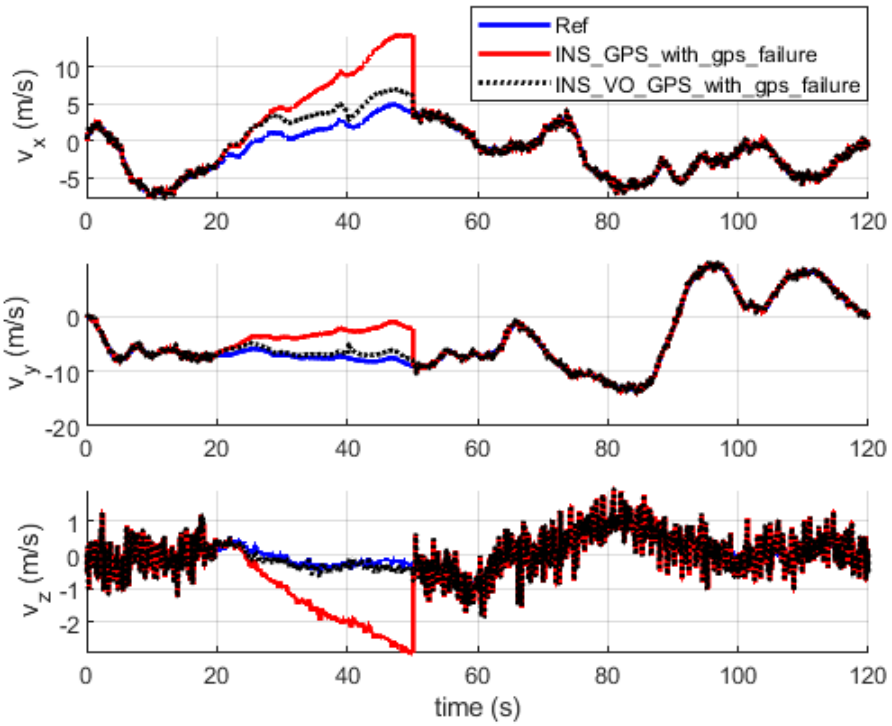

Figure 5-67 Velocity plot comparison for INS/VO/GPS with GPS failure

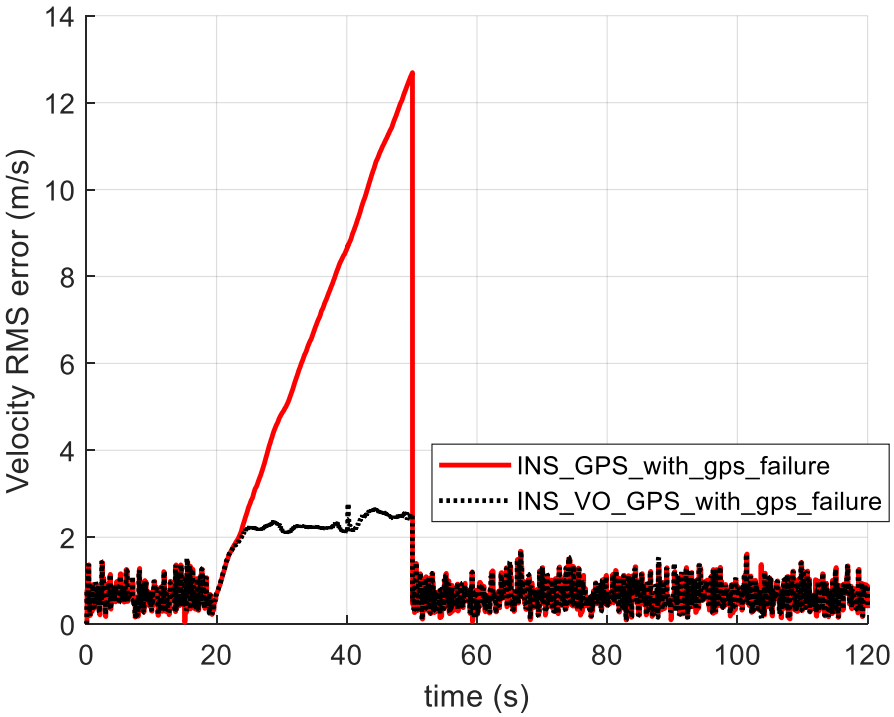

Figure 5-68 Velocity error plot comparison for INS/VO/GPS with GPS failure

It can be seen that GPS failure forces the INS/GPS navigation solution to utilize only the INS solution and as a result, the position and velocity state estimates quickly accumulate errors and start to drift from the reference trajectory and are not corrected until the GPS data is available. However, in case of INS/VO/GPS the effect of GPS failure is negated due to presence of VO

data, hence the INS solution continues to be corrected using VO measurements. This improvement can also be seen in the position and velocity RMS error averaged over the whole trajectory in the following table.

	Position RMSE	Velocity RMSE
INS/VO/GPS (With 30 sec GPS failure)	$\mathbf{2.31\ m}$	$\mathbf{1.29\ m/s}$
INS/GPS (With 30 sec GPS failure)	$34.45\ m$	$3.85\ m/s$

Table 5-13 RMSE position and velocity comparison for GPS failure case

These results validate the DEKF algorithm for INS/VO/GPS fusion by maintaining accuracy in case of GPS failure as well as VO failure.

5.5.2.3 INS/VO/GPS (without sensor failure)

This section shows the performance of INS/VO/GPS in case of no sensor failure in comparison to standalone INS, standalone GPS, and standalone VO with respect to the ground truth reference. It is expected that in case of no sensor failures, INS/VO/GPS should perform as well as INS/GPS, while with sensor failures it should outperform both INS/GPS and INS/VO. In the following figures (Figure 5-69 – Figure 5-75), '**Ref'** refers to ground truth reference, '**VO**' refers to the standalone VO solution, '**INS'** refers to the standalone INS solution, '**GPS**' refers to standalone GPS solution, and '**INS_VO_GPS**' refers to INS/VO/GPS integrated solution.

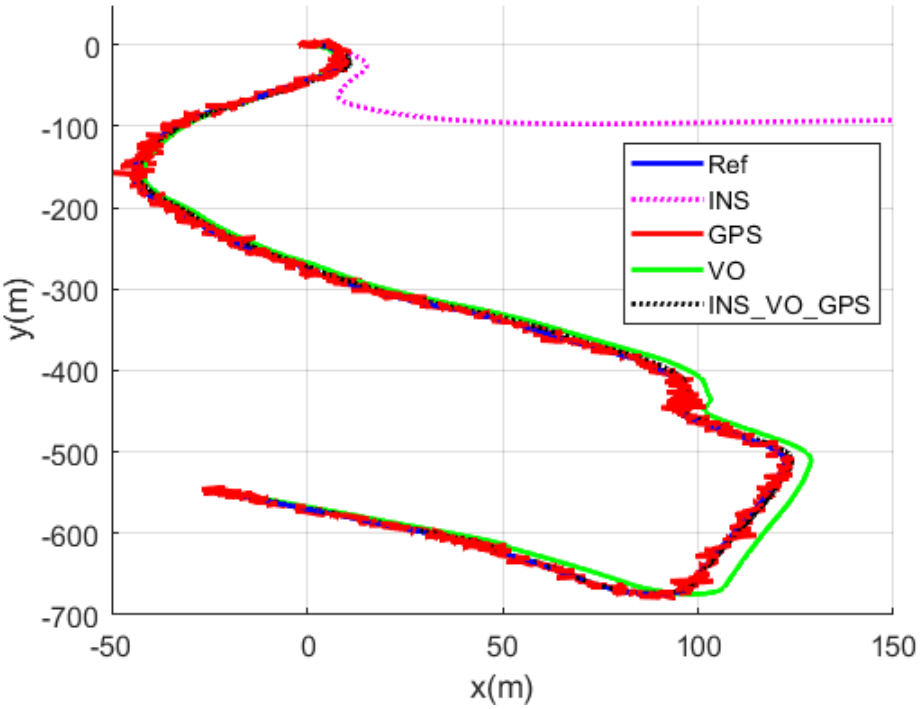

Figure 5-69 Trajectory plot comparison for INS/VO/GPS

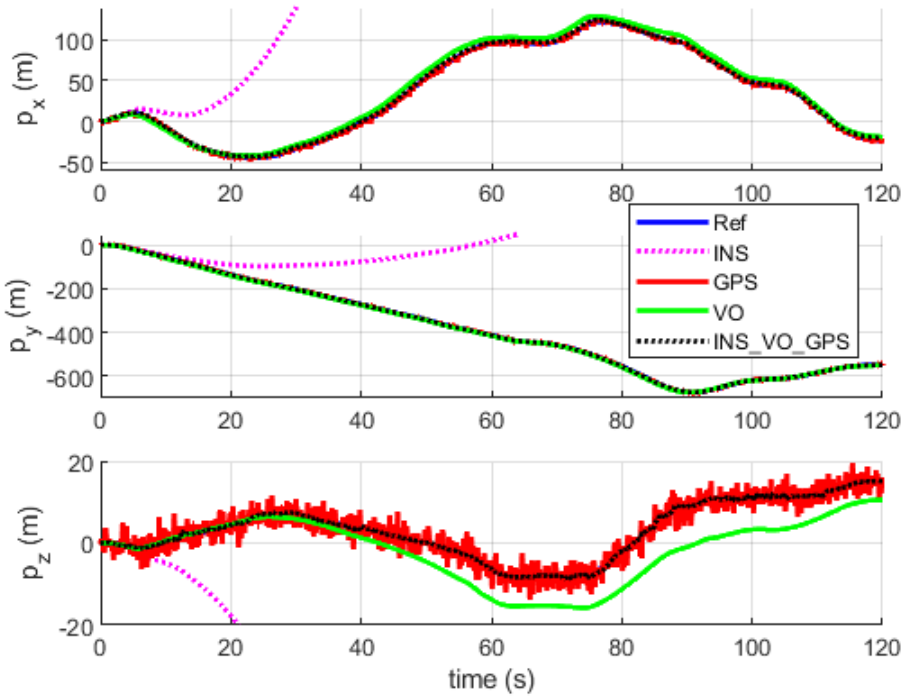

Figure 5-70 Position plot comparison for INS/VO/GPS

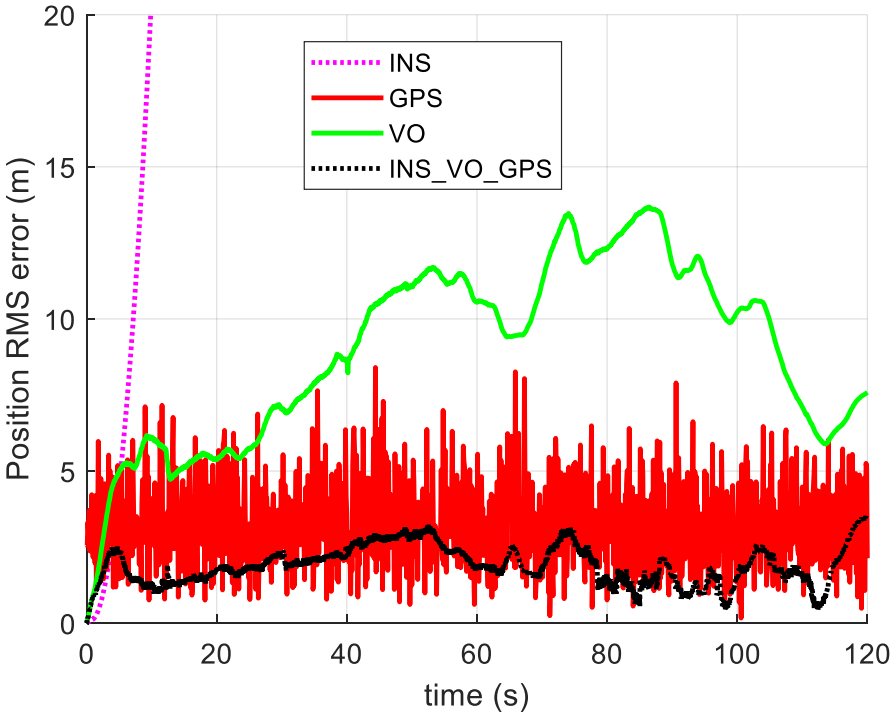

Figure 5-71 Position error comparison for INS/VO/GPS

The Figure 5-69 presents the trajectory comparison of INS, VO, GPS, and INS/VO/GPS with respect to the reference, Figure 5-70 shows the position estimate comparison, where p_x, p_y, p_z is the position estimate in x, y, z axis, and Figure 5-71 shows the position Root Mean Square (RMS) error plot. It can be seen that INS/VO/GPS follows the trajectory with higher accuracy as compared to all other standalone sensors. The Root Mean Square (RMS) error for position averaged over the whole trajectory is given in the following table.

	Position RMSE
INS/VO/GPS	$\mathbf{2.00\ m}$
GPS only	$3.52\ m$
VO only	$9.42\ m$
INS only	$1454.5\ m$

Table 5-14 RMSE position comparison for INS/VO/GPS

The INS/VO/GPS system shows position error reduction by 42.96% compared to GPS, and 78.67% compared to VO. This shows that the INS/VO/GPS fusion provides significant improvement in position estimates, with results being close to those of INS/GPS.

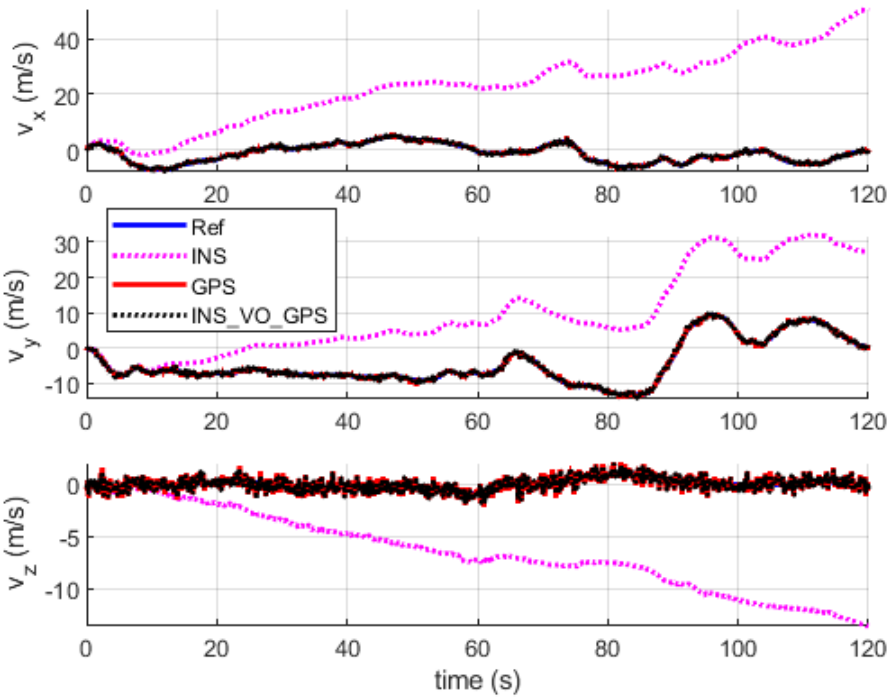

Figure 5-72 Velocity plot comparison for INS/VO/GPS

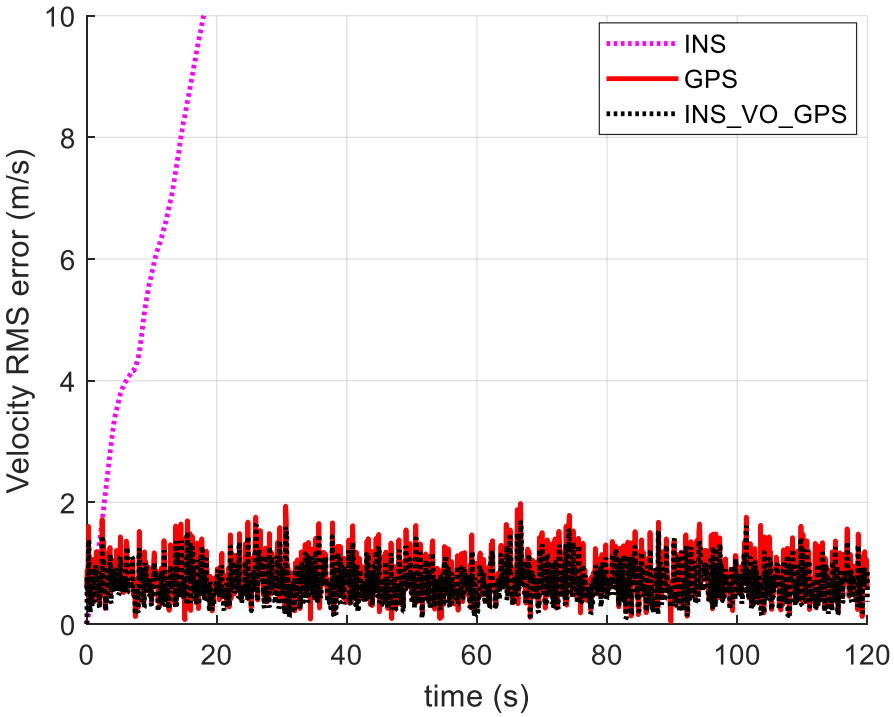

Figure 5-73 Velocity error comparison for INS/VO/GPS

Figure 5-72 shows the velocity comparison of INS, GPS, and INS/VO/GPS with respect to the reference, where v_x, v_y, v_z is that velocity estimate in x, y, z axis, and Figure 5-73 shows the velocity Root Mean Square (RMS) error plot. It should be noted that the INS solution in Figure 5-71 and Figure 5-73 is the same as shown in Figure 5-21 and Figure 5-23, respectively. The

y-axis of the plots here are limited to better observe the difference between GPS and INS/VO/GPS estimates. The reason for INS drift, as stated earlier in section 5.4.2, is because it relies on the previous state estimates to obtain the new state. However, due to the noisy sensor data, the state estimates computed by integration during INS computation are erroneous. These erroneous estimates are used to obtain the new estimates resulting in an accumulation of error. If there is no correction, then the INS solution diverges from the ground truth. The Root Mean Square (RMS) error for velocity averaged over the whole trajectory is given in the following table.

	Velocity RMSE
INS/VO/GPS	$\mathbf{0.74\ m/s}$
GPS only	$0.86\ m/s$
VO only	–
INS only	$32.44\ m/s$

Table 5-15 RMSE velocity comparison for INS/VO/GPS

For velocity, it can be seen that INS/VO/GPS provides a minor improvement of 14.68% when comparing to GPS only solution, and significant improvement of 97.73% when compared to INS only solution. Therefore, in velocity estimates INS/VO/GPS performance is parable to INS/GPS performance.

.

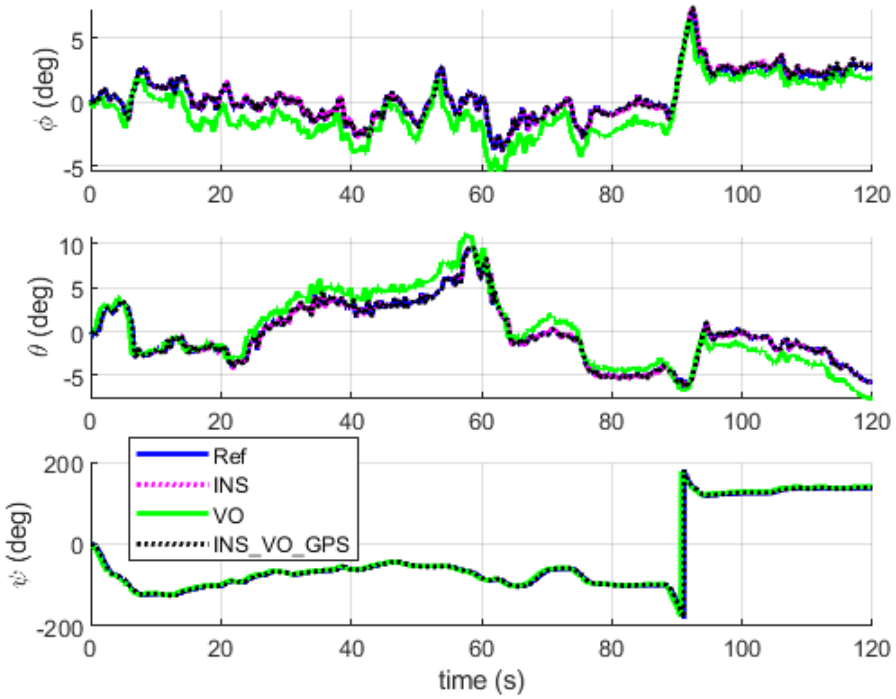

Figure 5-74 Attitude plot comparison for INS/VO/GNSS

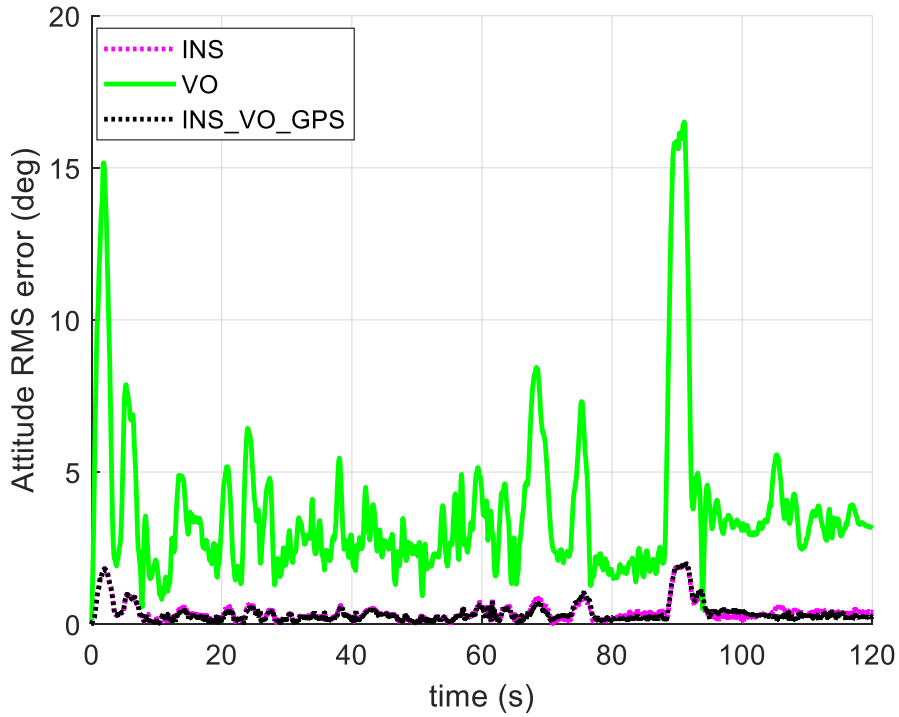

Figure 5-75 Attitude error comparison for INS/VO/GNSS

The Figure 5-74 shows the attitude comparison between INS, VO, and INS/VO/GPS with respect to ground truth reference, where ϕ, θ, ψ is the roll, pitch, and yaw angle about the x, y, z axis,Figure 5-75 shows the attitude Root Mean Square (RMS) error plot. The Root Mean Square (RMS) error for attitude averaged over the whole trajectory is given in Table 5-6.

	Attitude RMSE
INS/VO/GPS	$\mathbf{0.50°}$
GPS only	–
VO only	4.49°
INS only	0.52°

Table 5-16 RMSE attitude comparison for INS/VO/GPS

The INS/VO/GPS algorithm, compared to VO only solution, reduces the attitude error by 88.86%. With respect to INS, the INS/VO algorithm shows 2.988 % improvement. It can be seen that INS/VO/GPS integrated solution provides a significant improvement in in attitude, when compared to VO estimate, and a minor improvement when compared to INS estimate.

The results in all three cases of experimental evaluation section provide confidence in INS/VO/GPS integrated system to provide navigation solution with accuracy similar to the INS/GPS in case of no sensor failure, as well as outperform INS/GPS and INS/VO in case of either GPS failure or VO failure.

Chapter 6: Future Research

The work done in this research can be furthered in the future in the following aspects:

1. Physical testing may be carried out using IMU, GPS, and optical cameras with routes that will deliberately cause VO and GPS failures in order to test the robustness of INS/VO/GPS.
2. Tightly coupled integration may explored for Dual Extended Kalman Filter (DEKF).
3. Other sensor fusion techniques, such as Particle Filter, Unscented Kalman Filter, or Adaptive Kalman Filter, can be employed in order to improve estimates when white noise is not guaranteed.
4. Monocular VO with scale estimation can be employed thereby reducing the computational requirement as well as cost.
5. Other sensors such as wheel odometry, LiDAR, RADAR, may be considered to further improve the robustness in case of sensor failure.

Chapter 7: Conclusion

The objective of this research was to develop a multi-sensor fusion system to obtain navigation solution in order to determine position, velocity, and attitude of a ground vehicle using INS (Inertial Navigation System), VO (Visual Odometry), and GPS (Global Positioning System), which maintains accuracy in case of GPS or VO failure.

In the first phase of this research, the Visual Odometry (VO) system was developed and tested. In this phase, the traditional Stereo Visual Odometry (SVO) approach was analyzed and implemented, and then improvement to this approach was presented called the Modified Stereo Visual Odometry (ModSVO). For SVO, the SURF (Speeded-Up Robust Features) algorithm was used for detection of interest points and SSD (Sum of Squared Differences) was used to match the interest point descriptor vectors. The essential matrix was estimated using the 5-point algorithm along with RANSAC. This was further utilized to remove outliers based on the epipolar constraints. The 3D point location was computed using the Direct Linear Transformation (DLT) method. Pose estimation was carried out by solving the Perspective-3-Point (P3P) problem. It was observed that the estimated rotation matrix was not guaranteed orthonormal, therefore the estimated rotation matrix and translation vector was optimized by minimizing the reprojection error function using Levenberg Marquardt algorithm. The ModSVO improved on this algorithm by utilizing essential matrix decomposition to obtain the pose estimate, rather than solving P3P. As the resulting rotation matrix was reasonably accurate, only the translation vector is optimized by minimizing the reprojection error. This resulted in improved accuracy, while reducing the requirements for computational resources. The ModSVO and SVO algorithms were compared on the KITTI (Karlsruhe Institute of Technology and Toyota Technological Institute) real-world dataset, sequence 10. It was observed that the position RMSE (Root Mean Square Error) reduced from $38.00m$ in SVO case to $9.42m$ in ModSVO case. The attitude of RMSE reduced from $12.16°$ in SVO case to

4.49° in ModSVO case. These results showed that ModSVO is a significant improvement over the traditional SVO.

In the second phase, integrated navigation systems of INS/VO and INS/GPS were developed. It was observed that INS (Inertial Navigation System) is affected by error accumulation due to double integration over time during computation making it unsuitable as a standalone navigation solution. For GPS (Global Positioning System) the sensor noise, low frequency, and potential signal loss caused by blockage of line of sight of receiver and satellite due to tall trees, buildings, or tunnels rendered it as not a suitable standalone solution. Similarly, VO failure may also occur due to direct sunlight to the camera lens, blocking of camera field of view by some large vehicle, or unfavorable lighting conditions, so VO is also unsuitable as a standalone navigation solution. This provides the requirement for development of the integrated system.

The INS/VO (Inertial Navigation System / Visual Odometry) was developed by fusing INS data and VO data through an Extended Kalman Filter algorithm. The VO data was obtained using the ModSVO algorithm discussed above. In this structure, in addition to the traditional EKF structure, the EKF updated states were provided as a delayed feedback to the VO pose update segment. Furthermore, the VO uncertainty was computed by the standard deviation of the reprojection error residual over all the inliers at each frame. Therefore, the sensor noise covariance matrix was updated with each VO update. The algorithm was tested using the KITTI sequence 10 dataset. It was observed that the position RMSE was reduced from $9.42m$ (with standalone VO) to $4.63m$ (with INS/VO), the attitude RMSE reduced from 4.49° (with standalone VO) to 0.5° (with INS/VO), and the velocity RMSE reduced from $32.44m/s$ (with standalone INS) to $2.12m/s$ (with INS/VO). These results show that INS/VO integration presented a significant improvement over standalone INS, and standalone VO solutions. However, it was observed that if a VO failure occurred then the INS/VO solution significantly deteriorates to $46.56m$ position RMSE and $12.19m/s$ velocity RMSE.

The INS/GPS (Inertial Navigation System / Global Positioning System) was developed by fusing INS data and GPS data through an Extended Kalman Filter algorithm. The algorithm was tested using the KITTI sequence 10 dataset. It was observed that the position RMSE was reduced from $3.52m$ (with standalone GPS) to $1.77m$ (with INS/GPS), the velocity RMSE was reduced from $0.86m/s$ (with standalone GPS) to $0.73m/s$ (with INS/GPS). These results show that INS/GPS integration presented a significant improvement over standalone INS, and standalone GPS solutions, as well as the INS/VO integrated system because of the higher accuracy of the GPS in comparison to VO. However, it was observed that if a GPS failure occurred then the INS/GPS solution significantly deteriorates to $34.45m$ position RMSE and $3.85m/s$ velocity RMSE.

The observation of accuracy deterioration in case of sensor failure motivated the third phase of the research, in which an INS/VO/GPS integrated system was proposed using Dual Extended Kalman Filter (DEKF). DEKF utilizes both VO and GPS measurements as means to correct the INS state estimates. The input from accelerometer and gyroscope is used to compute the INS state estimates, and the inputs images from left and right cameras are used to compute the VO state estimates, VO uncertainty, and an indicator of occurrence of VO failure. This flag is true if the VO algorithm was unable to converge to a solution. Otherwise, in presence of valid VO solution, the flag is set to false. The VO estimates and INS estimates are fused using the EKF structure similar to INS/VO. The output is the state estimate error, which is used to correct the INS solution and provide an updated estimate of position, velocity, and attitude. These are fused with the GPS measurements using the EKF structure similar to INS/GPS. The solution from the second EKF is used to update the states from the first EKF. The solution is also fed back to the VO pose update sequence, similar to INS/VO structure. An indicator for connection of less than 4 GPS satellites is defined. A true value for this indicator means that due to signal blockage the GPS is unable to provide a reliable position estimate. In case the VO failure

indicator is true, then the VO measurements are bypassed, and if GPS failure indicator is true, then the GPS measurements are bypassed.

The INS/VO/GPS DEKF fusion was tested using the KITTI sequence 10 dataset. It was observed that in case of no sensor failure, INS/VO/GPS performed as well as the INS/GPS, however in case of GPS failure it significantly outperforms the INS/GPS solution, reducing the position RMSE from $34.45m$ (with INS/GPS) to $2.31m$ (with INS/VO/GPS), and reducing the velocity RMSE from $3.85m/s$ (with INS/GPS) to $1.29m/s$ (with INS/VO/GPS). In case of VO failure, INS/VO/GPS reduces the position RMSE from $46.56m$ (with INS/VO) to $1.76m$ (with INS/VO/GPS), and the velocity RMSE from $12.19m/s$ (with INS/VO) to $0.74m/s$ (with INS/VO/GPS).

These results show that in case of VO failure or GPS failure, the INS/VO/GPS DEKF fusion significantly outperforms the INS/VO and INS/GPS integrated systems. Therefore, the INS/VO/GPS DEKF fusion scheme is an auspicious contribution towards the end goal of autonomous robotics and self-driving cars.

References

[1] P. D. Groves, *Principles of GNSS, inertial, and multisensor integrated navigation systems*, 2nd ed. Boston: Artech House, 2013.

[2] U.S. government, "GPS.gov: GPS Overview." https://www.gps.gov/systems/gps/ (accessed Oct. 19, 2021).

[3] Y. Zhao, "GPS/IMU integrated system for land vehicle navigation based on MEMS," PhD Thesis, KTH Royal Institute of Technology, 2011.

[4] D. Gingras, "An overview of positioning and data fusion techniques applied to land vehicle navigation systems," *Automot. Inform. Commun. Syst. Princ. Veh. Netw. Data Exch.*, pp. 219–246, 2009.

[5] H. Min, X. Wu, C. Cheng, and X. Zhao, "Kinematic and Dynamic Vehicle Model-Assisted Global Positioning Method for Autonomous Vehicles with Low-Cost GPS/Camera/In-Vehicle Sensors," *Sensors*, vol. 19, no. 24, pp. 5430-, 2019, doi: 10.3390/s19245430.

[6] P. Zhang, J. Gu, E. E. Milios, and P. Huynh, "Navigation with IMU/GPS/digital compass with unscented Kalman filter," in *IEEE International Conference Mechatronics and Automation, 2005*, Jul. 2005, vol. 3, pp. 1497-1502 Vol. 3. doi: 10.1109/ICMA.2005.1626777.

[7] D. Nister, O. Naroditsky, and J. Bergen, "Visual odometry," in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, Jun. 2004, vol. 1, p. I–I. doi: 10.1109/CVPR.2004.1315094.

[8] D. Zhou, Y. Dai, and H. Li, "Ground-Plane-Based Absolute Scale Estimation for Monocular Visual Odometry," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 2, pp. 791–802, 2020, doi: 10.1109/TITS.2019.2900330.

[9] M. O. Aqel, A, M. H. Marhaban, M. I. Saripan, and N. B. Ismail, "Review of visual odometry: types, approaches, challenges, and applications," *SpringerPlus*, vol. 5, no. 1, pp. 1–26, Oct. 2016, doi: http://dx.doi.org/10.1186/s40064-016-3573-7.

[10] M. B. Alatise and G. P. Hancke, "Pose Estimation of a Mobile Robot Based on Fusion of IMU Data and Vision Data Using an Extended Kalman Filter," *Sensors*, vol. 17, no. 10, pp. 2164-, 2017, doi: 10.3390/s17102164.

[11] K.-W. Chiang, T. T. Duong, and J.-K. Liao, "The performance analysis of a real-time integrated INS/GPS vehicle navigation system with abnormal GPS measurement elimination," *Sensors*, vol. 13, no. 8, pp. 10599–10622, 2013, doi: 10.3390/s130810599.

[12] D. Scaramuzza and F. Fraundorfer, “Visual Odometry [Tutorial],” *IEEE Robot. Autom. Mag.*, vol. 18, no. 4, pp. 80–92, Dec. 2011, doi: 10.1109/MRA.2011.943233.

[13] D. Nister, “An efficient solution to the five-point relative pose problem,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 6, pp. 756–770, Jun. 2004, doi: 10.1109/TPAMI.2004.17.

[14] J.-P. Tardif, M. George, M. Laverne, A. Kelly, and A. Stentz, “A new approach to vision-aided inertial navigation,” in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. 2010, pp. 4161–4168. doi: 10.1109/IROS.2010.5651059.

[15] “MathWorks - Makers of MATLAB and Simulink - MATLAB & Simulink.” https://www.mathworks.com/?s_tid=gn_logo (accessed Oct. 26, 2021).

[16] C. Harris and M. Stephens, “A combined corner and edge detector,” in *Alvey vision conference*, 1988, vol. 15, no. 50, pp. 10–5244.

[17] G. Lowe, “SIFT-The Scale Invariant Feature Transform,” *Int J*, vol. 2, no. 91–110, p. 2, 2004.

[18] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, “Speeded-Up Robust Features (SURF),” *Comput. Vis. Image Underst.*, vol. 110, no. 3, pp. 346–359, Jun. 2008, doi: 10.1016/j.cviu.2007.09.014.

[19] Y. Cheng, M. W. Maimone, and L. Matthies, “Visual odometry on the Mars exploration rovers - a tool to ensure accurate driving and science imaging,” *IEEE Robot. Autom. Mag.*, vol. 13, no. 2, pp. 54–62, Jun. 2006, doi: 10.1109/MRA.2006.1638016.

[20] B. Kitt, A. Geiger, and H. Lategahn, “Visual odometry based on stereo image sequences with RANSAC-based outlier rejection scheme,” in *2010 IEEE Intelligent Vehicles Symposium*, Jun. 2010, pp. 486–492. doi: 10.1109/IVS.2010.5548123.

[21] K. Cao, X. Yang, S. Gao, C. Chen, J. Huang, and X. Song, “Visual Odometry Based on 3D-3D and 3D-2D Motion Estimation Method,” in *2018 Chinese Automation Congress (CAC)*, Nov. 2018, pp. 3643–3648. doi: 10.1109/CAC.2018.8623237.

[22] A. de la Escalera, E. Izquierdo, D. Martín, F. García, and J. M. Armingol, “Stereo Visual Odometry for Urban Vehicles Using Ground Features.” Springer International Publishing, Cham, pp. 385–397, 2015. doi: 10.1007/978-3-319-27146-0_30.

[23] R. Gonzalez, J. I. Giribet, and H. D. Patino, “NaveGo: A simulation framework for low-cost integrated navigation systems,” *J. Control Eng. Appl. Inform.*, vol. 17, no. 2, pp. 110–120, 2015.

[24] C. Yang, W. Shi, and W. Chen, “Correlational inference-based adaptive unscented Kalman filter with application in GNSS/IMU-integrated navigation,” *GPS Solut.*, vol. 22, no. 4, p. 100, Jul. 2018, doi: 10.1007/s10291-018-0766-2.

[25] P. J. Glavine, O. De Silva, G. Mann, and R. Gosine, “GPS Integrated Inertial Navigation System Using Interactive Multiple Model Extended Kalman Filtering,” 2018, pp. 414–419. doi: 10.1109/MERCon.2018.8421936.

[26] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, “Vision meets robotics: The KITTI dataset,” *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1231–1237, 2013, doi: 10.1177/0278364913491297.

[27] B. Suwandi, T. Kitasuka, and M. Aritsugi, “Low-cost IMU and GPS fusion strategy for apron vehicle positioning,” in *TENCON 2017 - 2017 IEEE Region 10 Conference*, Nov. 2017, pp. 449–454. doi: 10.1109/TENCON.2017.8227906.

[28] D. I. B. Randeniya, S. Sarkar, and M. Gunaratne, “Vision–IMU Integration Using a Slow-Frame-Rate Monocular Vision System in an Actual Roadway Setting,” *IEEE Trans. Intell. Transp. Syst.*, vol. 11, no. 2, pp. 256–266, Jun. 2010, doi: 10.1109/TITS.2009.2038276.

[29] G. Nützi, S. Weiss, D. Scaramuzza, and R. Siegwart, “Fusion of IMU and Vision for Absolute Scale Estimation in Monocular SLAM,” *J. Intell. Robot. Syst.*, vol. 61, no. 1–4, pp. 287–299, Jan. 2011, doi: http://dx.doi.org/10.1007/s10846-010-9490-z.

[30] V. Peretroukhin, L. Clement, M. Giamou, and J. Kelly, “PROBE: Predictive robust estimation for visual-inertial navigation,” in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2015, pp. 3668–3675. doi: 10.1109/IROS.2015.7353890.

[31] M. Bloesch, M. Burri, S. Omari, M. Hutter, and R. Siegwart, “Iterated extended Kalman filter based visual-inertial odometry using direct photometric feedback,” *Int. J. Robot. Res.*, vol. 36, no. 10, pp. 1053–1072, 2017, doi: 10.1177/0278364917728574.

[32] A. Martinelli, “Vision and IMU Data Fusion: Closed-Form Solutions for Attitude, Speed, Absolute Scale, and Bias Determination,” *IEEE Trans. Robot.*, vol. 28, no. 1, pp. 44–60, Feb. 2012, doi: 10.1109/TRO.2011.2160468.

[33] Y. Shen, Z. Zhu, and E. Mao, “Double-Fuzzy Kalman Filter Based on GPS/IMU/MV Sensor Fusion for Tractor Autonomous Guidance,” in *2007 IEEE International Conference on Automation and Logistics*, Aug. 2007, pp. 61–65. doi: 10.1109/ICAL.2007.4338531.

[34] Z. Yue, B. Lian, C. Tang, and K. Tong, “A novel adaptive federated filter for GNSS/INS/VO integrated navigation system,” *Meas. Sci. Technol.*, vol. 31, no. 8, p. 085102, May 2020, doi: 10.1088/1361-6501/ab78c2.

[35] R. Sun, Y. Yang, K.-W. Chiang, T.-T. Duong, K.-Y. Lin, and G.-J. Tsai, “Robust IMU/GPS/VO Integration for Vehicle Navigation in GNSS Degraded Urban Areas,” *IEEE Sens. J.*, vol. 20, no. 17, pp. 10110–10122, Sep. 2020, doi: 10.1109/JSEN.2020.2989332.

[36] Y. Song, S. Nuske, and S. Scherer, “A Multi-Sensor Fusion MAV State Estimation from Long-Range Stereo, IMU, GPS and Barometric Sensors,” *Sensors*, vol. 17, no. 1, p. 11, 2017, doi: http://dx.doi.org/10.3390/s17010011.

[37] J.-L. Blanco-Claraco, F.-Á. Moreno-Dueñas, and J. González-Jiménez, “The Málaga urban dataset: High-rate stereo and LiDAR in a realistic urban scenario,” *Int. J. Robot. Res.*, vol. 33, no. 2, pp. 207–214, Feb. 2014, doi: 10.1177/0278364913507326.

[38] W. Maddern, G. Pascoe, C. Linegar, and P. Newman, “1 year, 1000 km: The Oxford RobotCar dataset,” *Int. J. Robot. Res.*, vol. 36, no. 1, pp. 3–15, Jan. 2017, doi: 10.1177/0278364916679498.

[39] G. Pandey, J. R. McBride, and R. M. Eustice, “Ford Campus vision and lidar data set,” *Int. J. Robot. Res.*, vol. 30, no. 13, pp. 1543–1552, Nov. 2011, doi: 10.1177/0278364911400640.

[40] M. Arbabmir and M. Ebrahimi, “Visual–inertial state estimation with camera and camera–IMU calibration,” *Robot. Auton. Syst.*, vol. 120, p. 103249, Oct. 2019, doi: 10.1016/j.robot.2019.103249.

[41] C. Bamann and P. Henkel, “Visual-Inertial Odometry with Sparse Map Constraints for Planetary Swarm Exploration,” in *2019 IEEE International Conference on Industrial Cyber Physical Systems (ICPS)*, May 2019, pp. 290–295. doi: 10.1109/ICPHYS.2019.8780342.

[42] G. He, Q. Cao, X. Zhu, and H. Miao, “Visual-IMU State Estimation with GPS and OpenStreetMap for Vehicles on a Smartphone,” in *2020 16th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, Dec. 2020, pp. 516–521. doi: 10.1109/ICARCV50220.2020.9305386.

[43] P. Kim, H. Lim, and H. J. Kim, “Visual Inertial Odometry with Pentafocal Geometric Constraints,” *Int. J. Control Autom. Syst. IJCAS*, vol. 16, no. 4, pp. 1962–1970, Aug. 2018, doi: http://dx.doi.org/10.1007/s12555-017-0200-5.

[44] X. Kong, W. Wu, L. Zhang, and Y. Wang, “Tightly-Coupled Stereo Visual-Inertial Navigation Using Point and Line Features,” *Sensors*, vol. 15, no. 6, pp. 12816–12833, 2015, doi: http://dx.doi.org/10.3390/s150612816.

[45] A. Geiger, J. Ziegler, and C. Stiller, “StereoScan: Dense 3d reconstruction in real-time,” in *2011 IEEE Intelligent Vehicles Symposium (IV)*, Jun. 2011, pp. 963–968. doi: 10.1109/IVS.2011.5940405.

[46] Y. Liu *et al.*, “Stereo Visual-Inertial Odometry With Multiple Kalman Filters Ensemble,” *IEEE Trans. Ind. Electron.*, vol. 63, no. 10, pp. 6205–6216, Oct. 2016, doi: 10.1109/TIE.2016.2573765.

[47] C. Zhai, M. Wang, Y. Yang, and K. Shen, “Robust Vision-Aided Inertial Navigation System for Protection Against Ego-Motion Uncertainty of Unmanned Ground Vehicle,” *IEEE Trans. Ind. Electron.*, vol. 68, no. 12, pp. 12462–12471, Dec. 2021, doi: 10.1109/TIE.2020.3044802.

[48] Z. Yue, B. Lian, and Y. Gao, “Robust adaptive filter using fuzzy logic for tightly-coupled visual inertial odometry navigation system,” *IET Radar Sonar Navig.*, vol. 14, no. 3, pp. 364–371, 2020, doi: 10.1049/iet-rsn.2019.0390.

[49] J. H. Jung *et al.*, “Monocular Visual-Inertial-Wheel Odometry Using Low-Grade IMU in Urban Areas,” *IEEE Trans. Intell. Transp. Syst.*, pp. 1–14, 2020, doi: 10.1109/TITS.2020.3018167.

[50] X. Dong, B. He, X. Dong, and J. Dong, “Monocular visual-IMU odometry using multi-channel image patch exemplars,” *Multimed. Tools Appl.*, vol. 76, no. 9, pp. 11975–12003, May 2017, doi: http://dx.doi.org/10.1007/s11042-016-3927-8.

[51] Y. B. Sarvrood and Y. Gao, “Loosely-Coupled Stereo Vision-Aided 3D Reduced Inertial Sensor and GPS for Land Vehicle Localization,” in *Proceedings of the 28th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2015)*, 2015, pp. 2152–2160.

[52] M. Aladem and S. A. Rawashdeh, “Lightweight Visual Odometry for Autonomous Mobile Robots,” *Sensors*, vol. 18, no. 9, pp. 2837-, 2018, doi: 10.3390/s18092837.

[53] V. Dan, K. V. Vinay, R. G. Rohan, and G. D. Shilpa, “Integration of GPS and IMU using Kalman filter for terrestrial vehicle tracking”.

[54] K. U. Pavan, M. P. V. Sahul, and B. T. V. Murthy, “Implementation of stereo visual odometry estimation for ground vehicles,” in *2017 2nd IEEE International Conference on Recent Trends in Electronics, Information Communication Technology (RTEICT)*, May 2017, pp. 1173–1177. doi: 10.1109/RTEICT.2017.8256783.

[55] F. Liu, Y. B. Sarvrood, and Y. Gao, “Implementation and Analysis of Tightly Integrated INS/Stereo VO for Land Vehicle Navigation,” *J. Navig.*, vol. 71, no. 1, pp. 83–99, Jan. 2018, doi: 10.1017/S037346331700056X.

[56] P. J. Glavine, O. De Silva, G. Mann, and R. Gosine, “GPS Integrated Inertial Navigation System Using Interactive Multiple Model Extended Kalman Filtering,” in *2018 Moratuwa Engineering Research Conference (MERCon)*, May 2018, pp. 414–419. doi: 10.1109/MERCon.2018.8421936.

[57] Y. Sung-Joo and T. Kim, “Development of Stereo Visual Odometry Based on Photogrammetric Feature Optimization,” *Remote Sens.*, vol. 11, no. 1, Jan. 2019, doi: http://dx.doi.org/10.3390/rs11010067.

[58] A. Geiger, P. Lenz, and R. Urtasun, “Are we ready for autonomous driving? The KITTI vision benchmark suite,” in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2012, pp. 3354–3361. doi: 10.1109/CVPR.2012.6248074.

[59] Oxford Technical Solutions, “RTv2 GNSS-aided Inertial Measurement Systems User Manual.” https://www.oxts.com/ (accessed Jul. 09, 2021).

[60] W. Lu, S. A. Rodríguez F, E. Seignez, and R. Reynaud, “Lane Marking-Based Vehicle Localization Using Low-Cost GPS and Open Source Map,” *Unmanned Syst. Singap.*, vol. 3, no. 4, pp. 239–251, 2015, doi: 10.1142/S2301385015400014.

[61] P. Osborne, “Mercator,” Technical report, Edinburgh, 2013.

[62] D. Titterton and J. Weston, *Strapdown Inertial Navigation Technology*, vol. 17. Stevenage: The Institution of Engineering and Technology, 2004.

[63] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge: University Press, 2004. doi: 10.1017/CBO9780511811685.

[64] M. A. Fischler and R. C. Bolles, “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography,” *Commun. ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[65] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical recipes in C (2nd ed.): the art of scientific computing*. USA: Cambridge University Press, 1992.

[66] D. Nistér, “Preemptive RANSAC for live structure and motion estimation,” *Mach. Vis. Appl.*, vol. 16, no. 5, pp. 321–329, Dec. 2005, doi: 10.1007/s00138-005-0006-y.

[67] J. Nocedal, *Numerical Optimization*, 2nd ed. 2006. New York, NY: Springer New York, 2006. doi: 10.1007/978-0-387-40065-5.

[68] R. E. Kalman, “A New Approach to Linear Filtering and Prediction Problems,” *J. Basic Eng.*, vol. 82, no. 1, pp. 35–45, Mar. 1960, doi: 10.1115/1.3662552.

[69] 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.

[70] J. Z. Sasiadek, M. J. Walker, and A. Krzyzak, “Feature matching for UAV navigation in urban environments,” in *2010 15th International Conference on Methods and Models in Automation and Robotics*, Aug. 2010, pp. 164–169. doi: 10.1109/MMAR.2010.5587244.

[71] A. Monjazeb, J. Z. Sasiadek, and D. Necsulescu, “Autonomous navigation among large number of nearby landmarks using FastSLAM and EKF-SLAM - A comparative study,” in *2011 16th International Conference on Methods Models in Automation Robotics*, Aug. 2011, pp. 369–374. doi: 10.1109/MMAR.2011.6031375.