Sensor Fusion
For Navigation of Autonomous Ground Vehicles

by

Seyed Mohammad Moein Baghery Tabatabaei

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

Mechanical and Aerospace Engineering

Carleton University
Ottawa, Ontario

September 2022

Abstract

This thesis proposes an adaptive visual-inertial loosely-coupled sensor fusion method that uses an Error State Kalman Filter (ESKF) and Fuzzy Logic Controller (FLC). The method applies to GPS denied zones. In previous attempts, researchers either tried to tune the Kalman Filter in the most precise way possible, i.e., using the Genetic Algorithm (GA) to tune the Kalman Filter, or make an adaptive Kalman Filter to prevent the divergence problem from happening. This work aims to tune the Kalman Filter and makes it adaptive to overcome the disadvantages of previous methods and minimize the error of the estimated trajectories obtained by the Kalman Filter. The fuzzy system is trained via the Particle Swarm Optimization (PSO) algorithm to achieve this goal. A comparison is held between our tuned Kalman Filter, our adaptive system, and other methods previously used by other researchers. The results show that the proposed adaptive Kalman Filter improves the accuracy and outperforms other methods of tuning Kalman Filters. In addition, our proposed approach outperforms the conventional Extended Kalman Filter (EKF) methods. Our proposed structure can deal with differences between Inertial Measurement Unit (IMU) and camera data frequencies, which are assumed to be the same in most studies.

Acknowledgements

To begin with, I would like to thank and show my gratitude to my supervisor Professor Jurek Z. Sasiadek for giving me the opportunity to join the Aerospace Engineering program at Carleton University, and giving me the freedom I needed to do this thesis project. There are not enough words to describe my gratitude to his advice and help during this years of research.

I am forever thankful to all my friends who were always encouraging me to keep on working to finish this project.

Lastly, I would like to thank my family whose unconditional love and support have always given me the strength to go on and follow my passions.

Table of Contents

List of Tables

List of Figures

List of Acronyms

KF	Kalman Filter
FLC	Fuzzy Logic Controller
GA	Genetic Algorithm
PSO	Particle Swarm Optimization
EKF	Extend Kalman Filter
ESKF	Error-State Kalman Filter
IMU	Inertial Measurement Unit
FLAC	Fuzzy Logic Adaptive Controller
PPM	Perspective Projection Matrix
E	Essential Matrix
F	Fundamental Matrix
VO	Visual Odometry
VIO	Visual-Inertial Odometry
SAD	Sum-of-Absolute-Differences
INS	Inertial Navigation System
DCM	Direction Cosine Matrix
PF	Particle Filters
RANSAC	Random sample consensus
UKF	Unscented Kalman Filter
MAV	Micro Aerial Vehicle
TS	Takagi-Sugeno

Chapter 1

Introduction

In 1995, the NavLab 5 team, situated at Carnegie Mellon University, introduced an autonomous vehicle [1] that could drive from Pittsburgh to San Diego without human intervention. Although, autonomous vehicles were deployed to navigate through a complex environment of urban streets, further development was needed for the reliability and safety of the vehicles in all conditions. A significant contribution to autonomous vehicles came from the robotic technology developed for the DARPA challenge [2].

Autonomous vehicles navigate through their environment using onboard sensors to gather the required information for navigation purposes. In this scenario, failure of one sensor may result in the loss of track of the vehicle and cause serious consequences. Multiple sensors that can receive the same type of information (e.g., trajectory estimation and tracking) of the environment are used to circumvent the loss of track of the vehicle. Hence, the usage of multiple sensors and fusing their data is highly desirable for higher accuracy and safety reasons.

This thesis focuses on the problem of localization and navigation of autonomous ground vehicles in GPS denied zones. Navigation of autonomous ground vehicles in GPS denied zones implies that other sets of sensors are required to perform the same role as GPS. A system of stereo cameras and IMU can be a proper choice.

Kalman Filter (KF) can be used as an efficient method to estimate the state of the system and fuse the cameras and IMU data. KF is a computationally more efficient fusion algorithm compared to other common methods such as Unscented KF [3] and Particle Filter (PF) [3]- [4]. This thesis proposes a modified method to make Kalman Filter adaptive. In this method, a fuzzy system adjusts the tunable parameters of the Extended Kalman Filter (EKF). In order to find the best parameters, we utilize a heuristic optimization algorithm. Gessesse [24] also used a heuristic optimization algorithm, the Genetic Algorithm (GA), to tune their Kalman Filter. However, their method did not aim to make the EKF adaptive, but their method has shown a significant improvement in their results. It can be seen from comparing our results that our proposed method outperforms non-adaptive Kalman Filter methods like Gessesse [24] and conventional EKF.

Thesis Contributions

In this thesis, in order to minimize the error for navigational purposes an adaptive sensor fusion problem was considered. An EKF and FLC were implemented as an expert system that changes the EKF parameters. The goal was to minimize the errors of the estimated trajectory and bring it closer to the VO algorithm in the intervals that no new data is coming from cameras.

- Develop a novel method for tuning and optimizing EKF parameters and adaptive EKF that can handle the multi-rate Kalman Filter situation very well.
- Using Particle Swarm Optimization (PSO) algorithm to tune EKF and train the fuzzy system.

- Using an optimal gain scheduling Extended Kalman Filter to overcome dynamic changes during the operation. An FLC expert system is used to select the suitable Kalman gains in different situations.
- Although in the training stage we need offline data, the introduced trained fuzzy controller has the flexibility to be used in online applications.

Thesis Organization

This thesis is organized as follows:

- **Chapter 1:** This chapter includes the introduction.
- **Chapter 2:** This chapter provides the background and necessary information about the designed system. This background information consists of the camera model and visual odometry algorithm, Kalman Filter formulations, fuzzy logic, and adaptive Kalman Filter.
- **Chapter 3:** This chapter addresses different sensor fusion schemes and other discussions about them and provides our choice of method in this work.
- **Chapter 4:** This chapter includes the implementation details of our chosen method is discussed.
- **Chapter 5:** This chapter presents the details of the simulation experiments and the evaluation method. In addition, it presents the final results of those simulations. Furthermore, it includes the conclusion and the future work reference.

Chapter 2

System Overview and Background

This chapter begins with the introduction of the stereo camera system, camera calibration, and image rectification, which is then used to estimate the vehicle's motion.

Furthermore, the techniques of pose estimation using inertial data are presented. Afterward, methods for visual and inertial data fusion will be discussed, and the details of implementation will follow in Chapter 4.

2.1 Sensors

There are numerous methods of estimating a vehicle's motion in an unknown environment. There are many different sensors used in the navigation of autonomous vehicles. The sensors used and presented in this thesis are cameras and the Inertial Measurement Unit (IMU). The estimation of the vehicle's pose (position + orientation) using these sensors is also discussed in this section.

2.1.1 The Stereo Camera Setup

The Stereo Camera setup is referred to two cameras that are collinear to each other and are separated by a distance between the cameras' coordinate systems origins, which is known as the baseline. Stereo Camera Systems have the advantage of being scale-ambiguity-free compared to monocular systems. This advantage allows them to produce scene reconstruction in the form of point clouds up to a known scale (metric).

A conventional stereo system consists of two identical cameras that capture the left and right image sequences of the environment. The general configuration of the stereo vision system is shown in Figure 2.1. A point with unknown coordinates in the world coordinate system in three-dimensional (3D) Euclidean space is projected into the image plane of both cameras to obtain the two-dimensional (2D) points on the image plane coordinate systems [5]. In order to reconstruct the 3D position of the original point in world coordinates, we must first solve the problem known as the correspondence problem. Solving this problem means exact corresponding interest points on both image planes, left and right, must be found. Solving the correspondence problem requires calculating the disparity (distance between the projected point coordinates in the image coordinate system of the left and right cameras) of every image point must be computed. In order to calculate disparity, the Epipolar Geometry and related constraints on the stereo vision system should be applied. Epipolar Geometry defines the relationship between a 3D point and its projection onto two different image frames, as shown in Figure 2.1. In order to simplify the disparity calculation problem, procedures of camera calibration and image rectification must be performed. In the following section of this work, the author introduces the basis of the general camera calibration procedure.

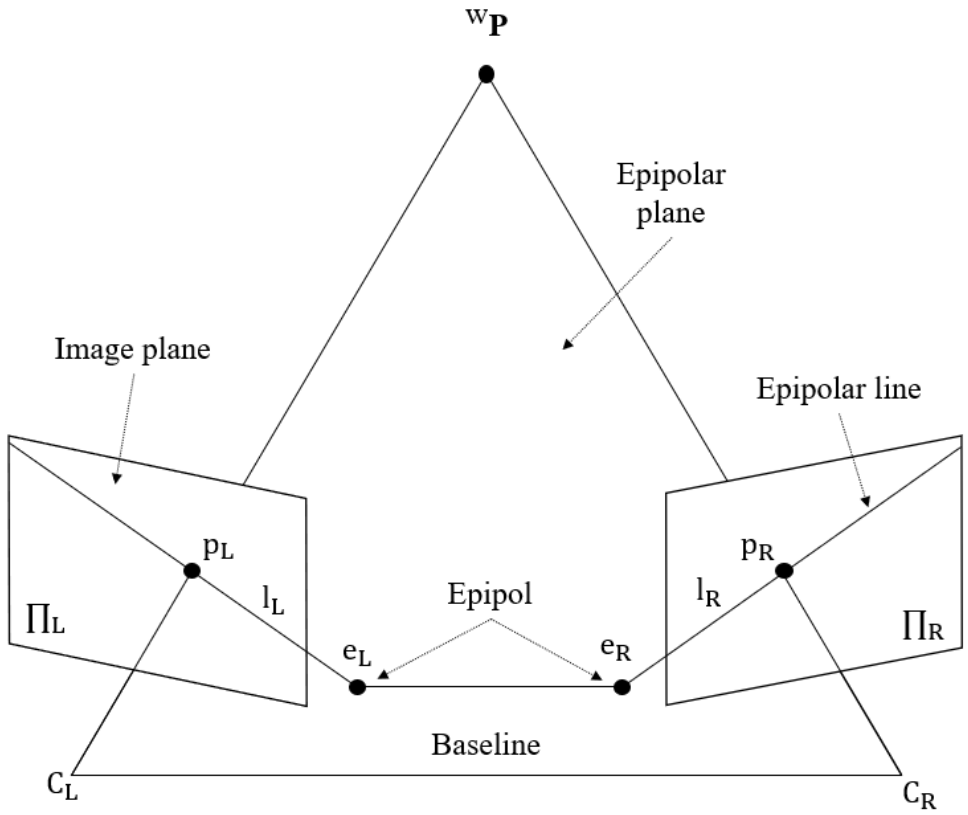

Figure 2.1. Epipolar Geometry Constraint [49].

2.1.2 Camera Calibration

The most crucial task for any machine vision application is determining the camera's intrinsic and extrinsic parameters. These parameters will be used later on to obtain the Essential matrix and Fundamental matrix to obtain a point's pose in the world coordinate system.

The *intrinsic* parameters describe the internal properties of the camera. These parameters include the focal length (f) in x and y direction, center of the image plane (c), radial (κ) and tangential (ρ) distortion coefficients, and a skew factor (γ), which is usually set to zero.

The extrinsic parameters describe the external properties of the camera system. These parameters express mathematical relations of the camera system in terms of location and orientation; i.e., for a stereo camera system, each camera has its coordinate system expressed in 6 Degree of Freedom (DoF), where conventionally, the left camera's origin is set as the whole system's origin. The right camera is then referenced with respect to the left one, and the distance between the camera's origins is established to be the baseline, as defined earlier.

In order to calibrate the stereo camera system, it is essential to consider a known coordinate system and use it as a reference. This coordinate system is known as the *world coordinate system*. Choosing the world coordinate frame allows the system to be referenced to a known coordinate system which will be used later on to determine the camera's own local coordinate system.

Although there is a built-in function in MATLAB that does the camera calibration job and considering the fact that KITTI Benchmark is used in this thesis, thus, camera calibration data is already existing, and the images are rectified; camera calibration procedures and image rectification will be discussed briefly.

The most used procedure for camera calibration was proposed by Zhang [6]. This procedure is done by taking several pictures of a known pattern (usually a chess board) with precise and known square dimensions, as shown in Figure 2.2. These images are then processed to look for the corners of the used pattern. Since the dimensions of the squares are known, it would be possible to compute the intrinsic and extrinsic parameters by following Zhang's work. This procedure is described in detail in [7].

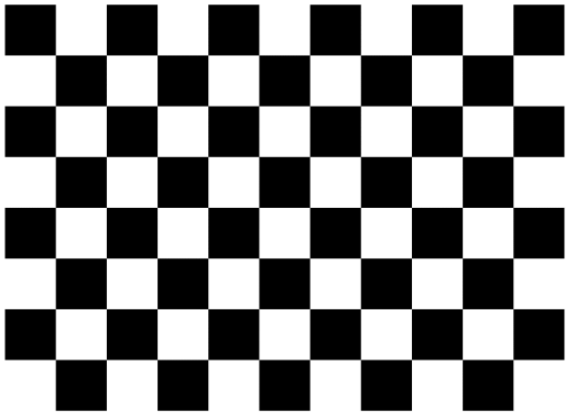

Figure 2.2. Conventional Chess Board Pattern Used in Camera Calibration [49].

The camera parameters are expressed in matrix form. Equation 2.1 shows the *intrinsic parameters matrix* $\boldsymbol{K}$, which contains all the internal parameters, thus:

$$\boldsymbol{K} = \begin{bmatrix} S_x & 0 & {}^{O_x}/_{f} \\ 0 & S_y & {}^{O_y}/_{f} \\ 0 & 0 & {}^{1}/_{f} \end{bmatrix} \tag{2.1}$$

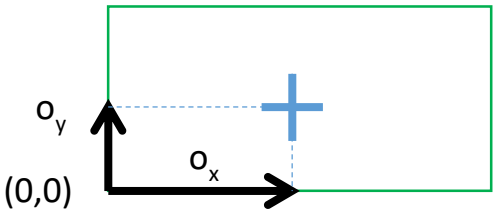

Figure 2.3. Image plane coordinate system

where, f is the focal length of camera, S_x and S_y are pixel scaling factors in x and y directions, as well as the offset of the image origin from the optical axis which are O_x and O_y.

The extrinsic parameters are then can be expressed by forming the 3×4 perspective projection matrix $\boldsymbol{M}$ as shown in equation 2.2 as shown below:

$$M_l = K_l[\, I \mid 0 \,]$$
$$M_R = K_R[\, R \mid -Rt \,] \tag{2.2}$$

where, M_l and M_R are the left and right cameras perspective projection matrices respectively, K_l and K_R are cameras intrinsic matrices. I is the 3×3 identity matrix representing the rotation of the left camera in degrees, R is a 3×3 matrix which expresses the rotation of right camera with respect to the left one, and finally, t is a 3×1 vector that represents the translation (in this case, it is the baseline) of the right camera with respect to the left camera.

A stereo system's *intrinsic* and *extrinsic* parameters are shown graphically in Figure 2.4. It can be observed that, as discussed earlier, the calibration Chess Board Pattern coordinate system is used as the world coordinate system of reference in the calibration process. Both cameras have their own coordinate system where typically, the left one is used as the origin of the whole system, so the right camera can be referenced to it. i.e., the right camera is translated by translation vector (t) and rotation matrix (R).

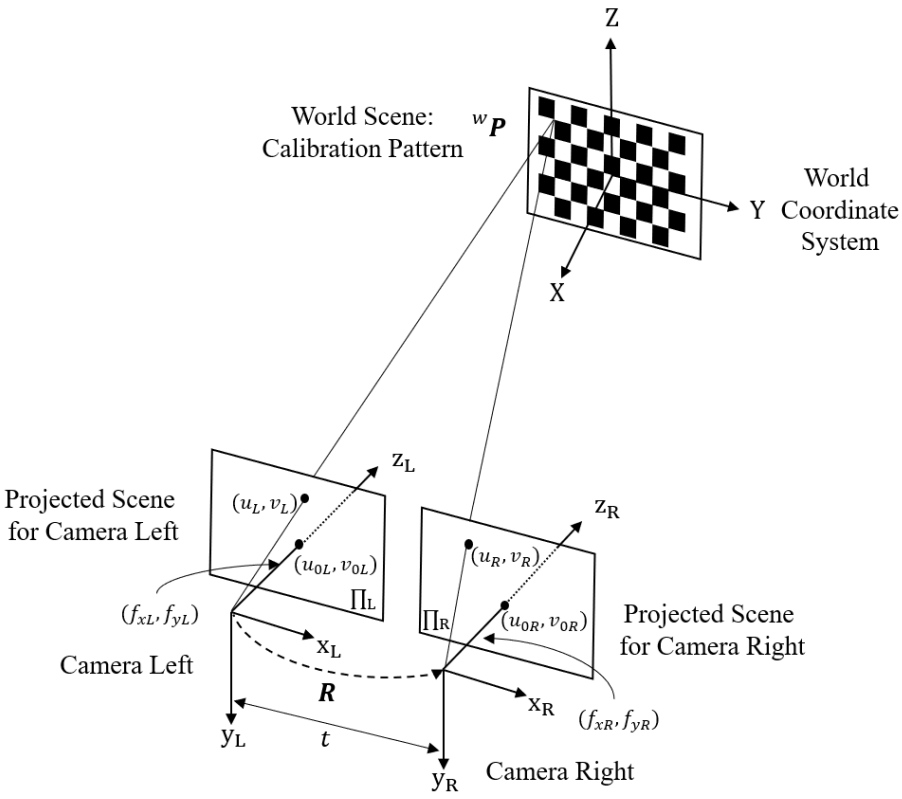

Figure 2.4. Stereo Vision Coordinate System [49].

2.1.3 Stereo Image Rectification and Epipolar Geometry

One of the crucial steps in visual odometry is *Stereo Image Rectification* or *Image Rectification*. Image rectification is the process that removes the radial and tangential distortion of the stereo images by performing 2D transformations over the input images. These parameters can be obtained from calibration parameters. Image rectification is done to simplify the process of matching and triangulation later. The distortion correction transformation corrects for lens distortion from the cameras, which warps the image near

the corners and can significantly reduce triangulation accuracy in these areas if not accounted for. The rectification transformation rotates the images such that the epipolar lines become horizontal. Making the epipolar lines horizontal makes matching of the stereo correspondences easier in a way that the search for finding correspondences becomes simply a search on a line. Doing this also speeds up the computations [8]. Considering the fact that KITTI Benchmark is used in this thesis, the images are already rectified and undistorted. Furthermore, there are MATLAB built-in functions for image rectification and undistorting. Further details on these processes can be found in [10]–[11].

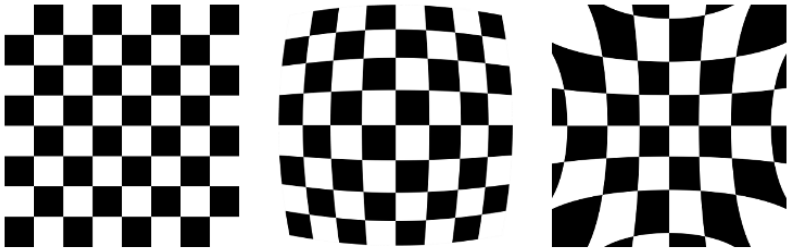

Figure 2.5. Rectified and Non-Rectified Images Examples [10]–[11]

In Figure 2.6, the rectified stereo image configurations can be seen. It can be observed that epipolar lines are aligned to the horizontal axis and meet the epipoles at infinity. The radial and tangential distortion are removed from the input stereo image pairs, and the center of the images is established to be at the same height. Image rectification considerably simplifies the correspondence problem, as mentioned earlier.

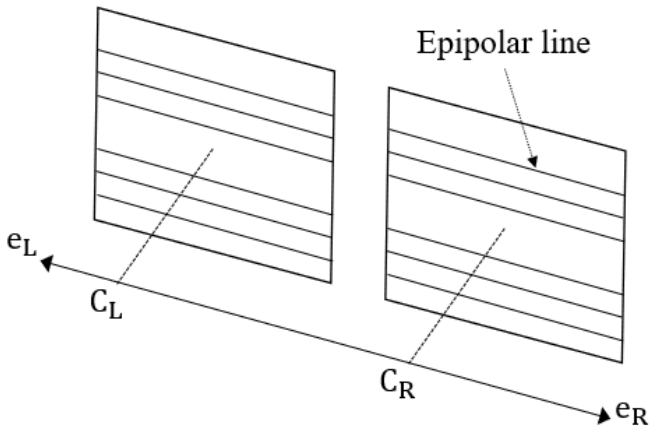

Figure 2.6. Rectified Stereo Configuration [49].

The Epipolar Geometry is also known as the two-view geometry. It is used to define the mathematical relations between a 3D point and its projection onto two different image frames, as shown in Figure 2.1.

The epipolar constraint is defined using the essential matrix $\boldsymbol{E}$, which relates the corresponding projections of a point in the world coordinate in two image frames.

Consider the position of a point P^w in the world coordinate system, its location in the left camera coordinate system is P_L^w with respect to the center of projection $\boldsymbol{C_L}$ and then it's location with respect to center of projection $\boldsymbol{C_R}$is P_R^w. The equation below would hold:

$$P_L^w = RP_R^w + t \tag{2.3}$$

where, R and t are the rotation and translation matrices between the coordinates of $\boldsymbol{C_L}$ and $\boldsymbol{C_R}$.

Note that when the projection of a 3D point P^w in one image is known to be p_L, then the

projection p_R on the second image will be restricted to the epipolar line shown in the figure 2.2 The 3D point P^w and the centers of projection $\boldsymbol{C_L}$ and $\boldsymbol{C_R}$. belong to the same plane. Knowing this, by Pre multiplying Equation (2.3) by $P_L^{W^T}[t]_\times$ results in:

$$P_L^{W^T}[t]_\times R P_R^w = 0 \tag{2.4}$$

where, $[t]_\times$ is the is skew symmetric matrix.

Thus, the essential matrix will be defined as $E = [t]_\times R$. This basically implies that:

$$P_L^{W^T} E P_R^w = 0 \tag{2.5}$$

This is also true for the projected image points on image plane, p_L and p_R, which gives:

$$p_L^T E p_R = 0 \tag{2.6}$$

The essential matrix and epipolar constraint will be helpful only if the cameras are calibrated. In the case of uncelebrated cameras, the intrinsic parameters of the cameras are required to determine the epipolar geometry. This will be given by the fundamental matrix $\boldsymbol{F}$.

$$F = (K_L^{-1})^T E K_R^{-1} \tag{2.7}$$

where, K_L and K_R are the camera intrinsic matrices.

Once the epipolar geometry is obtained, the relative pose (rotation and translation) estimation between two adjacent camera frames would be possible when a set of feature correspondences are known.

2.1.4 Visual Odometry and Motion Estimation

Visual odometry (VO) is referred to the process of estimating the trajectory of a moving agent (e.g., vehicle, human, or robot) by means of the input images that come from a single or multiple cameras attached to it. This method has been applied in robotics, wearable computing, augmented reality, and Autonomous vehicles. The term VO was introduced by Nister in his landmark paper [12]. A detailed review on the progress of visual odometry can be found in this two-part tutorial series [13]–[14].

The algorithm used in this work by the author is an adaptation of [15]. Its steps are outlined below:

Algorithm steps:

1. Capture images: I_l^t, I_r^t, I_l^{t+1}, I_r^{t+1}.
2. Undistort and rectify the above images.
3. Compute the disparity map D^t from I_l^t, I_r^t and the map D^{t+1} from I_r^{t+1}, I_r^{t+1}.

4. Use FAST algorithm to detect features in I_l^t, I_r^t and match them and track them in the next frame I_l^{t+1}, I_r^{t+1}.

5. Use the disparity maps D^t , D^{t+1} to calculate the 3D positions of the features detected in the previous steps. Two point clouds W^t, W^{t+1} will be obtained

6. Select a subset of points from the above point cloud such that all the matches are mutually compatible.

7. Estimate R, t from the inliers that were detected in the previous step.

   As mentioned earlier, because we are using KITTI Vision Benchmark Suite, the images are already undistorted and rectified, so we skipped this step.

   In step three, the disparity map, in general, can be calculated as follows. Assume a particular $3D$ point in the physical world P^w. Its projection's position in the left image plane is $(u_L. v_l)$. The same feature position in the right image plane is $(u_l + d. v_l)$, thus, the location $(u_L. v_l)$ on the disparity map holds the value *d*. It should be noted that the y-coordinates will be the same since the images have been rectified. Hence, the disparity at each point in the image plane will be defined as:

$$d = u_L - u_R \tag{2.8}$$

   The FAST [16] corner detector is used in the feature detection step. Consider a point ***P*** in the image plane, as shown in Figure 2.7, which is tested to see if it can be considered a corner. A circle of 16px circumference is drawn around this point. Suppose there exists a continuous set of pixels with an intensity more than that of the main pixel by at least factor **I**, and for another continuous set of pixels, the intensity is less than that of the main

pixel by at least the same factor **I.** In that case, point ***P*** is marked as a corner. A heuristic is used for rejecting the non-corners, in which the pixel at 1,9,5,13 are tested first, and at least three of them must have a higher intensity by at least the amount of **I** or must have an intensity lower by at least the same amount in order to the point be identified as a corner. This approach was chosen due to being computationally less expensive in comparison to other popular interest point detectors such as SIFT.

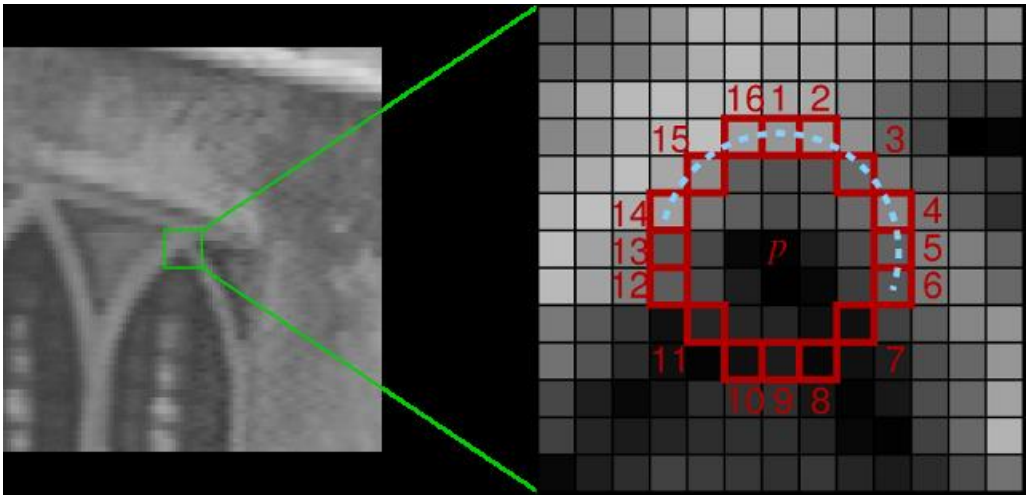

Figure 2.7. Image from original FAST paper [16]

For the next step, the detected features must be matched. Feature matching can be done through several existing matching algorithms (e.g., Belief propagation, Cooperative region, Graph cut, etc.). In this work, a Semi-Global Block Matching [17] is used, which is an advanced version of the Block-Matching algorithm. This algorithm works in a way that for every pixel in the left image, a 15×15 pixel wide window is generated around it, and the value of all the pixels in the windows is stored. Then the same window is constructed at

the same coordinate in the right image. This window will then slide horizontally until the Sum-of-Absolute-Differences (SAD) is minimized. Minimizing the SAD also gives us the disparity we will need later on in the triangulation step.

After the above is done, the detected corners must be tracked in the next frame. That means the corners detected in I_l^t are matched to the corner detected in next frame I_l^{t+1}. To do that a BRISK [18] descriptor (a binary feature vector) is used.

The next step is triangulation. Given that the images are already rectified, triangulation is a straightforward process using readily available information from the camera calibration. Knowing that the perspective projection matrix of the cameras, *M*, projects 3D points coordinates in the world coordinate system into the image plane coordinate system, a similar transformation matrix can be defined, which will bring the points in the rectified images back to the world coordinate system. This matrix is called the reprojection matrix Q, defined according to each camera's intrinsic properties, as well as the translation *T* between the cameras (baseline):

$$Q = \begin{bmatrix} 1 & 0 & 0 & -c_x \\ 0 & 1 & 0 & -c_y \\ 0 & 0 & 0 & f \\ 0 & 0 & -T_x^{-1} & 0 \end{bmatrix} \tag{2.9}$$

where, c_x, and c_y are coordinates of the optical center of the left camera (in pixels), f focal length of the left camera, and T_x is the x-coordinate of the right camera with respect to the

left one (in meters). The following equation is used to obtain the 3D coordinates of every feature in the image $\mathcal{F}_l^t$ and $\mathcal{F}_l^{t+1}$:

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = Q \begin{bmatrix} u_l \\ v_l \\ d \\ 1 \end{bmatrix} \tag{2.10}$$

Let the set of point clouds obtained from eq.(2.10) be referred to as W^t, W^{t+1}.

The next step in the algorithm outline is to select a subset of points from the above point cloud such that all the matches are mutually compatible. The algorithm used here is different from the most commonly used visual odometry algorithms in the sense that it does not have an outlier (aka., Anomaly) detection step like RANSAC [19], but it has an inlier detection step. The assumption here is that the scene is rigid, which means it must not change between two consecutive image frames. Scene rigidity implies that the distance between any two features in the point cloud W^t must stay the same as the distance between the corresponding points in W^{t+1}. If any such distance does not stay the same, it means either there is an error in the 3D triangulation of at least one of the two features, or we have triangulated a moving, which cannot be used in the next step. In order to have the largest set of consistent matches, the consistency matrix M is formed such that:

$$\mathrm{M}_{i.j} = \begin{cases} 1. \text{ if the distance between } i \text{ and } j \text{ points is same in both the point clouds} \\ 0. \text{ otherwise} \end{cases}$$

From the initial point clouds formed by triangulation, the largest subset of points is to be selected such that all the points in the subset are consistent with each other (every element

in the reduced consistency matrix is 1). This method is similar to the Maximum Clique Problem [19], with M as an adjacency matrix. A subset of a graph that only contains nodes that are all connected to each other is called a clique. This problem is known to be NP-complete, and thus an optimal solution cannot be found for any practical situation. Therefore, a greedy heuristic is employed that gives us a clique close to the optimal solution. The algorithm steps are listed as follows:

1. Select the node with the maximum degree, and initialize the clique to contain this node.
2. From the existing clique, determine the subset of nodes v, which are connected to all the nodes that exist in the clique.
3. From the set v, select a node connected to the maximum number of other nodes in v. Repeat from step 2 till no more nodes can be added to the clique.

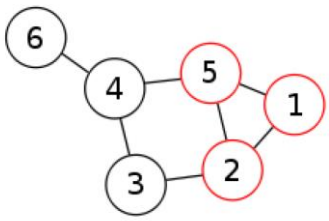

Figure 2.8. The largest clique in the above graph contains 1,5,2 [20]

The final step in the VO algorithm is to estimate $R.t$ from the inliers detected in the previous step. In order to determine the rotation matrix R and translation vector t, the

Levenberg-Marquardt non-linear least-squares minimization was used to minimize the following sum:

$$\epsilon = \sum_{\mathcal{F}_l^t . \mathcal{F}_l^{t+1}} (j_t - MTw_{t+1})^2 + (j_{t+1} - MT^{-1}w_t)^2 \quad (2.11)$$

in eq. (2.11), $\mathcal{F}_l^t . \mathcal{F}_l^{t+1}$, are the features in the left image at time t and $t+1$, j_t and j_{t+1} are 2D homogeneous coordinates of the features $\mathcal{F}_l^t$, $\mathcal{F}_l^{t+1}$, w_t and w_{t+1} are 3D homogeneous coordinates of the features $\mathcal{F}_l^t$, $\mathcal{F}_l^{t+1}$, M is 3×4 perspective projection matrix of the left camera, and T is the 4×4 homogeneous transformation matrix.

2.1.5 Pose from IMU

The Inertial Navigation System (INS) is composed of Inertial Measurement Unit (IMU) and a Processing Unit. An IMU measures the linear acceleration, angular velocity, and magnetic field acting on a body. By combining data from these accelerometers and gyroscopes, it is possible to determine a moving vehicle's position, velocity, and attitude.

An ordinary IMU sensor measurement can be modeled as:

$$\widehat{m}(t) = \big(1 + s(t)\big)m(t) + b(t) + n(t) \quad (2.12)$$

where, $\widehat{m}(t)$ is the noisy measurement, $s(t)$ is a scale factor, $b(t)$ is bias and $n(t)$ is Gaussian random noise. Thus error compensation can be done by:

$$m(t) = {(\widehat{m}(t) - b(t))}/{(1 + s(t))} \tag{2.13}$$

Accelerometer

An *accelerometer* is a sensor which measures linear accelerations. It has 3 degrees of freedom. The measured accelerations are influenced by noise and bias. Consider that $a_{SF}^{B} = \left[f_x \; f_y \; f_z \right]^T$ is the accelerometer measurements. Then the compensated acceleration after accounting for gravity and Coriolis effect can be written as:

$$\dot{v}^L = C_B^L a_{SF}^B + g^L - (\omega_{EL}^L + 2\omega_{IE}^L) \times v^L \tag{2.14}$$

where, C_B^L is the Direction Cosine Matrix (DCM) of the body frame with respect to local frame, g^L is the acceleration vector caused by gravity, ω_{EL}^L is the transport rate and ω_{IE}^L is earth rate. in eq. (2.14) a_{SF}^B can also be written as:

$$a_{SF}^B = \frac{\hat{a}_{IB}^B - b_a}{(1+s_a)} + w_a \tag{2.15}$$

where, $\hat{a}_{IB}^{B}$ is the accelerometer measurements, s_a is the accelerometer scale factor, b_a is accelerometer bias and w_a is zero mean random noise of accelerometer.

Gyroscope

A *gyroscope* is a sensor that measures angular velocities. It has 3 degrees of freedom. Suppose that ω_{LB}^{B} is the compensated gyroscope measurement and $\widehat{\omega}_{IB}^{B}$ is gyroscope raw measurement, then ω_{LB}^{B} and $\widehat{\omega}_{IB}^{B}$ are related by:

$$\omega_{LB}^{B} = \frac{\widehat{\omega}_{IB}^{B} - b_g}{(1+s_g)} - \omega_{IL}^{B} + w_g \tag{2.16}$$

where, w_g is zero mean random noise of gyroscope, s_g is scale factor, b_g is bias and ω_{IL}^{B} is the earth rate add to transport rate given by:

$$C_B^L \, \omega_{IL}^{L}$$

$$\omega_{IL}^{L} = \omega_{IE}^{L} + \omega_{EL}^{L} \tag{2.17}$$

with gyroscope's corrected data, we form the vector W_{LB}^{B} as below:

$$W_{LB}^{B} = \begin{bmatrix} 0 & \omega_{LBx}^{B} & \omega_{LBy}^{B} & \omega_{LBz}^{B} \end{bmatrix}^T \tag{2.18}$$

Coordinate frames

There are two coordinate frames used here, which should be distinguished from each other while working with IMU.

- Body frame (B): the body coordinate frame is rigidly attached to the IMU. When an IMU is attached to the vehicle, this coordinate frame moves with it so the motion of the vehicle can be calculated between each time step.
- Local Frame (L): The local frame is attached to the earth's surface. The origin of this coordinate frame is chosen as the starting point of the vehicle's motion. The total trajectory is then calculated in this coordinate system.

IMU motion equations

The motion of IMU can be modeled by having equations (2.12) to (2.18), a set of navigation equations. Consider the position of IMU in the local frame r^L, the velocity of IMU in the local frame v^L, the attitude quaternion describing the rotation from body frame to local frame q_B^L, and accelerometer and gyroscope biases b_a and b_g. Thus we will have the IMU motion equations as below:

$$\dot{q}_B^L = \frac{1}{2}\left(\left[W_{LB}^B - w_g\right]x\right)q_B^L \tag{2.19}$$

$$\dot{v}^L = C_B^L[\hat{a}_{IB}^B - b_a + w_a] + g^L \tag{2.20}$$

$$\dot{r}^L = v^L \tag{2.21}$$

$$\dot{b}_g = 0 + w_{b_g} \tag{2.22}$$

$$\dot{b}_a = 0 + w_{b_a} \tag{2.23}$$

where, $[W_{LB}^B - w_g]x$ is the skew symmetric matrix formed from vector $[W_{LB}^B - w_g]$.

2.2 Fusion of IMU and Vision Sensors

Sensor fusion is referred to methods used to estimate the state x of a system given observations z from different sensors and, in some cases, a control sequence u acting on the system. Depending on the system and the way the problem is formulated, several sensors can be used to gather information of interest regarding a system. The observations gathered by sensors are affected by noise. Therefore, it is not straightforward to determine a model that can accurately yield the state of the system given those noisy observations. Furthermore, it should be considered that the noises acting on the system and uncertainty of the model are of stochastic nature. An approximate system model is formulated to cope with noises and uncertainties. The parameters of this model are estimated such that the noise and uncertainty in the system are minimized.

2.2.1 Kalman Filter

The Kalman Filter is a special case of a recursive Bayesian filter for multivariate normal distributions with additive white Gaussian noise and discrete-time linear system. Generally, a Kalman Filter is applied to a system of order n, with m inputs and j measurement sources [19]. The filter algorithm consists of two parts, prediction and correction steps. The prediction step uses the system equations to determine a prior estimate of the state at the next time step. The correction step corrects the prior estimate based on observations made.

Now, consider the system below:

$$\dot{x}(t) = F(t)x(t) + G(t)w(t) \tag{2.24}$$

$$y(t) = H(t)x(t) + v(t) \tag{2.25}$$

here, F is the state transition model, and G is noise shaping matrix and H is the sensor observation model. In this notation, the control signal "u" is taken as a state and it is added to the state vector x. So, the deterministic part of "u" is modeled in F matrix. The random part of "u" is modeled in $G(t)w(t)$. The $w(t)$ and $v(t)$ are zero mean random noises signals with covariance matrices Q_{d_k} and R_k respectively.

The system described by eq. (2.24) and (2.25) can be written in discrete form as:

$$x_{k+1} = \Phi_k x_k + G_k w_k \tag{2.26}$$

$$y_k = H_k x_k + v_k \tag{2.27}$$

where,

$$\Phi \cong (I + FdT) \tag{2.28}$$

$$Q_{d_k} \cong \mathrm{GQ}G^T T \tag{2.29}$$

and T is the sampling period.

as mentioned above, $w(t)$ and $v(t)$ are zero mean random noises signals with covariance matrices Q_{d_k} and R_k, so we will have:

$$E\langle v_k \rangle = 0 \qquad var(v_k, v_l) = R_k$$

$$E\langle w_k \rangle = 0 \qquad var(w_k, w_l) = Q_{d_k}$$

in prediction step, noise signals will be ignored. So, we will have:

$$x_{k+1} = \Phi_k x_k \tag{2.30}$$

$$y_k = H_k x_k + v_k \tag{2.31}$$

Kalman Filter can be derived in least square sense by calculating the correction gain "K_k" to minimize the error in state estimation $x(t)$.

$$\hat{x}_k^+ = \hat{x}_k^- + K_k \delta y_k^- \tag{2.32}$$

$$\hat{y}_k^- = H\hat{x}_k^- \tag{2.33}$$

$$\delta y_k^- = y_k - \hat{y}_k^- \tag{2.34}$$

where, $\hat{x}_k^+$ is the corrected and updated state and $\hat{x}_k^-$ is predicted state. In eq. (2.34) δy_k^- is called residuals which is the difference between the observed noisy measurements y_k and predicted measurements $\hat{y}_k^-$.

in conclusion, the Kalman Filter algorithm can be summarized as:

- Initialization

$$\hat{x}_0^- = E\langle x_0 \rangle \tag{2.35}$$

$$P_{k-1}^+ = P_0^- = var(x_0^-) \tag{2.36}$$

- State Prediction

$$x_{k+1} = \Phi_k x_k \tag{2.37}$$

$$\hat{y}_k^- = H\hat{x}_k^- \tag{2.38}$$

$$P_k^- = \Phi_{k-1} P_{k-1}^+ \Phi_{k-1}^T + Q_{d_{k-1}} \tag{2.39}$$

- Measurement Update

$$(P_k^+)^{-1} = (P_k^-)^{-1} + H_k^T R_k^{-1} H_k \tag{2.40}$$

$$K_k = P_k^+ H_k^T R_k^{-1} \tag{2.41}$$

$$\delta y_k^- = y_k - \hat{y}_k^- \tag{2.42}$$

$$\hat{x}_k^+ = \hat{x}_k^- + K_k \delta y_k^- \tag{2.43}$$

$$\hat{y}_k^+ = H \hat{x}_k^+ \tag{2.44}$$

in equations above, P is the error state covariance, and $\hat{y}_k^+$ is the smoothed measurements.

2.2.2 Extended Kalman Filter

Kalman Filter has two main limitations, it works on linear dynamic systems and assumes the noises are zero-mean Gaussian. Most systems in real life are nonlinear and non-Gaussian. However, the linearity condition can be relaxed by linearizing the system about an initial guess using the first-order Taylor expansion for $f(.)$ and $h(.)$ and then using the updates to estimate the states. On the other hand, the Gaussian condition cannot be relaxed, so other estimators such as Particle Filters (PF) can be used [20].

Consider the system below:

$$\dot{x}(t) = f(x(t)\,, u(t) + w(t)) \tag{2.45}$$

$$y(t) = h(x(t)) + v(t) \tag{2.46}$$

using Tylor series, we will get:

$$F(t) = \frac{\partial f(x(t),u(t))}{\partial x}\Big|_{\tilde{x}} \tag{2.47}$$

$$H(t) = \frac{\partial h(x(t))}{\partial x}\Big|_{\tilde{x}} \tag{2.48}$$

the Extended Kalman Filter algorithm can be written as:

- Prediction Step

$$x_{k+1} = f(x_k,u_k) \tag{2.49}$$

$$P_{k+1} = (I + F_{k+1}T)P_k(I + F_{k+1}T)^T + Q_{k+1} \tag{2.50}$$

- Update Step

$$K_{k+1} = P_{k+1}H_{k+1}^{T}(H_{k+1}P_{k+1}H_{k+1}^{T} + R_{k+1})^{-1} \quad (2.51)$$

$$\hat{x}_{k+1}^{+} = \hat{x}_{k+1}^{-} + K_{k+1}[y_{k+1} - h(x_{k+1})] \quad (2.52)$$

$$P_{k+1} = (I - K_{k+1}H_{k+1})P_{k+1} \quad (2.53)$$

2.2.3 Error-State Kalman Filter

Error- State Kalman Filter (ESKF) is the most commonly used filter in applications involving pose estimation and navigation [21]. This is because it can overcome the drawbacks of a full-state Kalman Filter, which is discussed in detail in Roumeliotis et al. [22], introducing the error-state Kalman Filter discussion. The Error-State model is simply the Kalman Filter that uses the error between the predicted and the measurement values.

Most real-life systems consist of high-frequency non-linear dynamics, which a non-linear model describes. In addition, it is not possible to linearize these systems very accurately. However, because the Error-State Kalman Filter considers only the perturbations in states, it means that it can be linearly integrated close to zero mean. Hence, it is suitable for linear Gaussian filtering. So, being linearly integrated close to zero mean leads to better accuracy in Error-State Kalman Filters.

Furthermore, it should be noted that the equations for a Total State EKF are the same as the closed-loop ESKF model [23]. Therefore, the same behavior is expected from both approaches. The only difference remains in the implementation.

The Error-State Kalman Filter algorithm can be written as:

first, the matrices will be derived

$$\dot{x}(t) = f(x(t).u(t) + w(t)) \tag{2.54}$$

$$y(t) = h\big(x(t)\big) + v(t) \tag{2.55}$$

$$F(t) = \frac{\partial f(x(t).u(t)+w(t))}{\partial x}\Big|_{\tilde{x}} \tag{2.56}$$

$$G(t) = \frac{\partial f(x(t).u(t)+w(t))}{\partial w}\Big|_{\tilde{x}} \tag{2.57}$$

$$H(t) = \frac{\partial h(x(t))}{\partial x}\Big|_{\tilde{x}} \tag{2.58}$$

$$\delta\dot{x}(t) = F(t)\delta x(t) + G(t)w(t) \tag{2.59}$$

$$\delta y(t) = H\delta x(t) \tag{2.60}$$

- Prediction Step

$$P_{k+1} = (I + F_{k+1}T)P_k(I + F_{k+1}T)^T + G_{k+1}Q_{k+1}G_{k+1}^T T^2 \tag{2.61}$$

- Update Step

$$P_{k+1} = (I - K_{k+1}H_{k+1})P_{k+1} \tag{2.62}$$

$$K_{k+1} = P_{k+1}H_{k+1}^T(H_{k+1}P_{k+1}H_{k+1}^T + R_{k+1})^{-1} \tag{2.63}$$

$$\hat{x}_{k+1}^+ = \hat{x}_{k+1}^- + K_{k+1}\delta y \tag{2.64}$$

2.2.4 Exponential Weighted Kalman Filter

The Kalman Filter operates on the assumption that all of the system dynamics and noise processes are known, and the noise processes are zero mean white. Divergence problems will occur when the theoretical behavior of a filter and its actual behavior does not agree. The Divergence problem can be classified into two types [47]: Apparent divergence and True divergence. In apparent divergence, the actual error of the system remains bounded but in a larger bound than the predicted error covariance. In this situation, the filter will be suboptimal, leading to a solution that converges to a higher bound. In true divergence, the actual estimation covariance eventually becomes infinite. The actual estimation covariance becoming infinite means that the error of the system tends to infinity. This results from the system being unstable or when unstable states are present and are not modeled.

In Kalman Filter, the predicted error covariance matrix and the Kalman Gain go to values near zero over time (large k). This means that the KF relies on the estimations rather than measurements as time passes by.

The exponential data weighting method can be used to prevent divergence for large k [47]. This method prevents the Kalman Gain from going to zero by applying an exponential weighting in the process noise covariance matrix Q and measurement covariance matrix R.

Let us set the model covariance matrices as:

$$R_k = R\alpha^{-2(k+1)} \tag{2.65}$$

$$Q_k = Q\alpha^{-2(k+1)} \tag{2.66}$$

where, $\alpha \geq 1$. For $\alpha > 1$, as time k increases, the R and Q decrease, so that the most recent measurement is given higher weighting factor. If $\alpha = 1$, it makes the weighted filter works as a regular KF.

by applying 2.65 and 2.66 to the equations of the Error-state Kalman Filter model in section 2.2.3, the covariance matrix P and Kalman gain K become:

$$K_{k+1} = P_{k+1}H_{k+1}^T\left(H_{k+1}P_{k+1}H_{k+1}^T + R_{k+1}\alpha^{-2(k+1)}\right)^{-1} \quad (2.67)$$

$$P_{k+1} = (I + F_{k+1}T)P_k(I + F_{k+1}T)^T + G_{k+1}Q_{k+1}G_{k+1}^T T^2\alpha^{-2(k+1)} \quad (2.68)$$

2.3 Fuzzy Logic

Fuzzy Logic is referred to the act of mapping an input space to an output space. The inputs are mapped to a fuzzy space by means of membership functions. Each membership function maps its corresponding input to the interval of [0,1], which is called membership degree. Membership functions are connected to each other and the outputs by using some rules defined by an expert designer. Each rule can be represented as follows:

$$\text{Rule } l\text{: IF } X_1 = A_1 \text{ AND } X_2 = A_2 \ldots . X_n = A_n \text{ THEN } f_l = K_l$$

where, X_n is the n^{th} fuzzy variable, A_n is n^{th}input, f_l is the output function of rule l and K_l is the consequent parameter.

The output of a fuzzy system is as follows:

$$U(\tilde{x}) = \frac{\sum_l^M \left(\left(\prod_{i=1}^n \mu^{F_l^i(x_i)} \right) . c^l \right)}{\sum_{l=1}^M \left(\prod_{i=1}^n \mu^{F_l^i(x_i)} \right)} = \sum_{l=1}^M \Phi^l c^l \qquad (2.69)$$

where, c^l is the output parameter of rule l, Φ^l is the firing strength of rule l and F_l^i is the membership degree of rule l given input i and finally M is the total number of the rules.

2.4 Particle Swarm Optimization (PSO)

The Particle Swarm Optimization (PSO) algorithm was introduced in 1995, based on the social behavior of bird flocking and swarming theory in general [45]. This algorithm is based on the population, similar to a Genetic Algorithm (GA). In most of the PSO implementations, only two formulas (2.70) and (2.71) are used to describe the particles' behavior. In this algorithm, the members of the population, which are called particles, are moved based on the direction vector obtained from the best position of all particles, the best position of each particle by itself, and the direction vector of the previous movement.

$$V_{id}^{t+1} = \omega . V_{id}^{t} + c_1 r_1 (pBest_{id}^{t} - Z_{id}^{t}) + c_2 r_2 (gBest_{d}^{t} - Z_{id}^{t}) \quad (2.70)$$

$$Z_{id}^{t+1} = Z_{id}^{t} + V_{id}^{t+1} \ . \quad i = 1.2. \ldots . n \quad (2.71)$$

where, V_{id}^{t} is the velocity of the d^{th}dimension of i^{th} particle in the t^{th}iteration, $pBest_{id}^{t}$ is the best position of d^{th}dimension of i^{th} particle in the t^{th}iteration. $gBest_{d}^{t}$ is the d^{th}dimension of the best position amongst all of the particle till t^{th}iteration. Furthermore, $r_1$and r_2 are two random numbers. In (2.70) ω, c_1 and c_2 are scalers and act as a weighting factors in (2.70). In (2.71), using the velocity obtained from (2.70), yields new position of the particles. In this equation, Z_{id}^{t} is the position of d^{th}dimension of i^{th} particle in the t^{th}iteration. The flow chart of PSO algorithm is shown in the Figure 2.9.

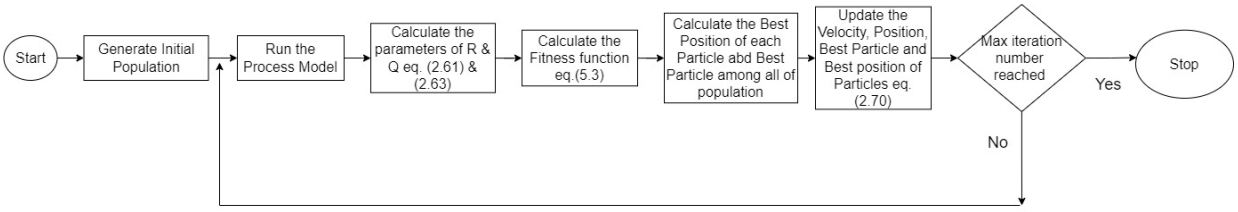

Figure 2.9. The PSO algorithm flow chart

2.5 Fuzzy Adaptive Kalman Filter

The measurement noise covariance matrix R and system noise covariance matrix Q are normally determined by the systems specifications and sensor characteristics and then are modified during the development phase of the system. There are several ways to tune a Kalman Filter, like using the Genetic Algorithm to find optimum Q and R matrices [24].

Despite our best effort, the optimum Kalman Filter tuning varies over time, and that causes the Kalman Filter to perform less accurately. An adaptive Kalman Filter may be used to correct the measurement and system noise covariance as the system operates. Yong Liu et al. [25] used an ensemble of three separate Kalman Filters that operate in different fusion intervals. Sasiadek et al. [26]- [27] used Fuzzy Logic to make Kalman Filter adaptive for INS and GPS signals sensor fusion. The Fuzzy Adaptive Kalman Filter is basically a standard KF that uses a feedback adaptation control with which the covariance matrices Q, R, or the Kalman gain are adjusted.

Chapter 3

Multi Sensor Fusion

In this chapter, several visual odometry and visual-inertial odometry techniques are introduced. Furthermore, the necessity for visual-inertial fusion to overcome the drawbacks of a standalone VO system is discussed, and the superiority of these techniques is surveyed. Also, the sensor fusion schemes that can be used are briefly argued. These will provide motivation for the implemented method.

3.1 Visual Odometry

As was discussed in the previous chapter, estimating the motion of a moving vehicle in an environment using data provided by cameras attached to it is called visual odometry (VO). This technique has applications in the field of Autonomous vehicles for localization and mapping. In this chapter, we briefly discuss three approaches to Visual odometry.

3.1.1 Feature-based Approach

The feature-based method is a conventional approach to the visual odometry problem. These methods rely on extracting unique interest points in the image, aka “features,” as illustrated in Figure 3.1. These feature points are tracked and matched over consecutive image frames. The algorithm outline for this was discussed in Section 2.1.4.

The feature-based methods can handle large frame-to-frame motions, and they are accurate because they use Efficient optimization of structure and motion (e.g., Bundle Adjustment). The main drawbacks of these methods can be mentioned as being Slow due to costly feature extraction and matching. Also, despite our best effort, an outlier matching can happen – if, for example, RANSAC is used- which affects the accuracy of the system in a long run.

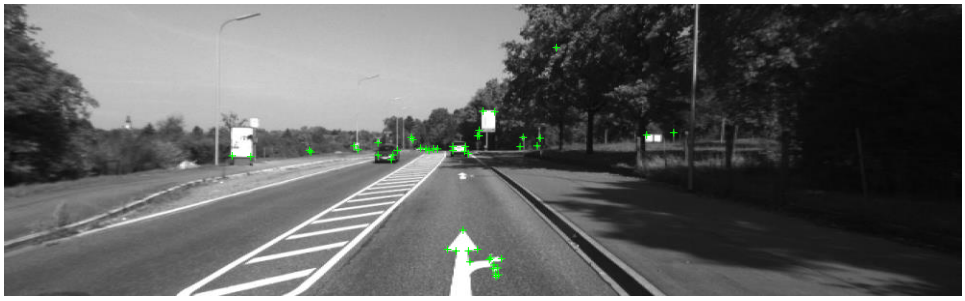

Figure 3.1. Detected FAST corners

As mentioned in the previous chapter, the algorithm used in this work by the author is an adaptation of an algorithm introduced by Howard [15], which is a feature-based method. This method was described by Howard [15] as inlier detection rather than outlier rejection (e.g., RANSAC) which has several advantages. To begin with, it is extremely robust to

false matches. This aspect allows us to simplify the matching stage of the algorithm, so we can trade accuracy for speed. Second, it was mentioned that the algorithm is well-equipped to handle dynamic scenes since each independent motion gives rise to a separate set of self-consistent features. Although the dynamic scenes are not discussed and are not considered in our assumptions in this work, it is worthy of mentioning that this algorithm is not confused when the agent is driving into its own shadow, a known problem for many algorithms of such.

3.1.2 Direct Approach

Feature-based methods include a process of feature detection, extraction, and matching. While the direct methods are the ones that compare pixels directly, therefore, the feature extraction step is skipped. Direct methods estimate the motion between two frames using the pixel intensity information. This motion estimation is achieved through using photometric error minimization such as the one done in Engel et al. [28]. This photometric error minimization implies that every pixel places a constraint on the optimization problem, which leads to an accurate and robust pose estimation in addition to a dense map of the environment.

The main assumption in these methods is that the projection of a point in both frames has the same intensity. This assumption is likely to fail oftentimes due to changes in lighting, sensor noises, pose errors, and moving objects. Direct methods require a high frame rate to compensate for and minimize the changes in intensity. Furthermore,

increasing the camera frame rate reduces the computational cost per frame. Another way proposed to handle the light intensity is, normalizing the global intensity, Gonçalves et al. [29]. Another major issue in this approach is being computationally expensive due to the use of all pixels over all frames. One way to solve this is by only using pixels with sufficient information, which are the ones with high-intensity gradients. The details can be found in Engel et al. [28]. Doing this, a semi-dense map that mostly contains edges and corners, as is shown in Figure 3.2, will be obtained. Concha et al. [30] proposed minimizing the photometric error accompanied by the image intensity error regularization. This technique will result in a smooth solution and helps to reconstruct low-gradient pixels. Generally, it can be said that direct methods are more sensitive to camera calibration in comparison with feature-based methods [31].

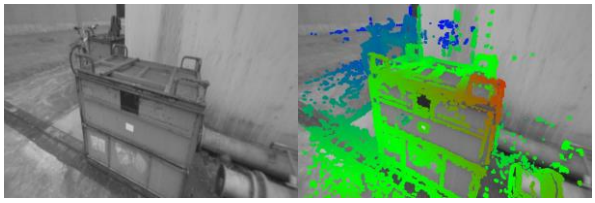

Figure 3.2. Extraction of pixel intensity data at corners and edges. (Reprinted from ORB-SLAM [28]).

3.1.3 Hybrid Approach

A hybrid approach (a.k.a. semi-dense or semi-direct) is a method that basically benefits from the advantages of feature-based and direct methods. Hybrid methods are less computationally expensive than direct methods, and they are capable of handling large inter-frame motions and also producing a semi-dense map. Forster et al. [32] used small but numerous patches around features, this increased robustness and allowed neglecting the patch normally. The next step in their work was minimizing the photometric error of these feature patches, and then the camera pose was estimated.

A two-layered approach that combined feature-based tracking with semi-dense direct image alignment for Stereo visual odometry was proposed by Krombach et al. [33]. In their method, a semi-dense map was estimated from the motion obtained by a fast and robust feature point-based method. They implemented an LSD-SLAM using the stereo camera to circumvent the scale ambiguity of the monocular version of it. In scenarios of a fast or large motion, the direct method was used to assign a key-frame which then was optimized using stereo measurements and also the propagated depth from the previous key-frame. Their method focuses on the fusion of depth estimates from motion between key-frames of the direct method with instantaneous stereo depth estimates to overcome the issues created due to large motions.

3.2 Sensor fusion methods

Visual odometry methods are very well researched, but they suffer from several drawbacks that the main ones can be addressed as:

- Loss of Tracking

  Visual odometry will perform best when there is sufficient overlap of features between two images, and this overlap requires a high frame rate. A high frame rate makes the process computationally more expensive, so either a high computation speed is required, or we will need more powerful processors, hence making the system less practical. However, by using a lower frame rate, it is possible to circumvent the issue of heavy computation to some extent; when there is a sudden movement of the camera, there might not be enough overlap of features due to the low frame rate, and this might result in loss of tracking.

- Scale Ambiguity

  In the case of monocular visual odometry, the scale of the scene is unknown. Thus, it will generate a trajectory that may seem accurate but not scaled correctly. The scale here is referred to the ratio between the distances that the vehicle has traveled and the estimated distances obtained from visual odometry. Although, it should be noted that we will not face this problem in stereo visual odometry.

- Lack of high frequency data feedbacks

  Assume that the designed odometry system is supposed to be used in an autonomous car or robot. To be able to successfully design a practical control plant system to control the

vehicle's trajectory, we will need high-frequency feedback data from sensors. As discussed above, it is not the case for cameras.

One solution to overcome the issues mentioned above is fusing the visual data with an IMU. Due to its high frequency, an IMU can provide good information for large sudden movements in a small time interval. Furthermore, the information from IMU is also scaled correctly. Considering these factors has motivated researchers to use an Inertial Measurement Unit (IMU) along with the visual sensor. Also, it is worth mentioning that IMU data can be very noisy, so it will be prone to false trajectory calculations and drift. Ergo, the fusion of Visual and Inertial sensors seems to complement each other drawbacks.

Based on the type of visual observations utilized, two main IMU/Visual fusion schemes exist; loosely coupled and tightly coupled. These two methods will be briefly discussed as follows.

3.2.1 Loosely-coupled method

The loosely-coupled approach means that visual and inertial frameworks are treated as independent black boxes in which only the outputs are of interest. Salim Sırtkaya et al. [34] proposed a loosely-coupled approach based on the error propagation model using the indirect Kalman Filter. In their proposed method, to overcome colored noises of VO, they tried to average the measurements over a Kalman period. To achieve robustness, they used a sigma-test within the filter. Weiss and Siegwart [35] proposed a loosely-coupled implementation in which the estimation of metric scale is independent of the visual algorithm.

Kelly and Sukhatme [36] have used the Unscented Kalman Filter (UKF) in their implementation of the visual-inertial approach due to the UKF's ability to handle non-linearity in the system better than the Extended Kalman Filter (EKF). In their solution, the agent is localizing itself and simultaneously mapping the environment and self-calibrating the transformation between camera and IMU. Lynen et al. [37] implemented a multi-sensor fusion using EKF on a Micro Aerial Vehicle (MAV), which was robust in localization in long-term missions. As mentioned earlier, all of these filter-based loosely-coupled sensor fusion schemes treat the visual odometry algorithm as a black box; hence the pose and covariance resulting from VO are used.

The loosely-coupled sensor fusion scheme is efficient and less computationally expensive in comparison to the tightly-coupled scheme. Furthermore, it is easier to implement; ergo, it is more suited for real-time tasks. It is also more facile to expand the system if more sensors are needed. However, in terms of accuracy, it is less accurate in comparison to a tightly-coupled framework since cross-coupling between visual and inertial parameters is neglected.

The general framework for a loosely-coupled scheme is shown in Figure 3.3.

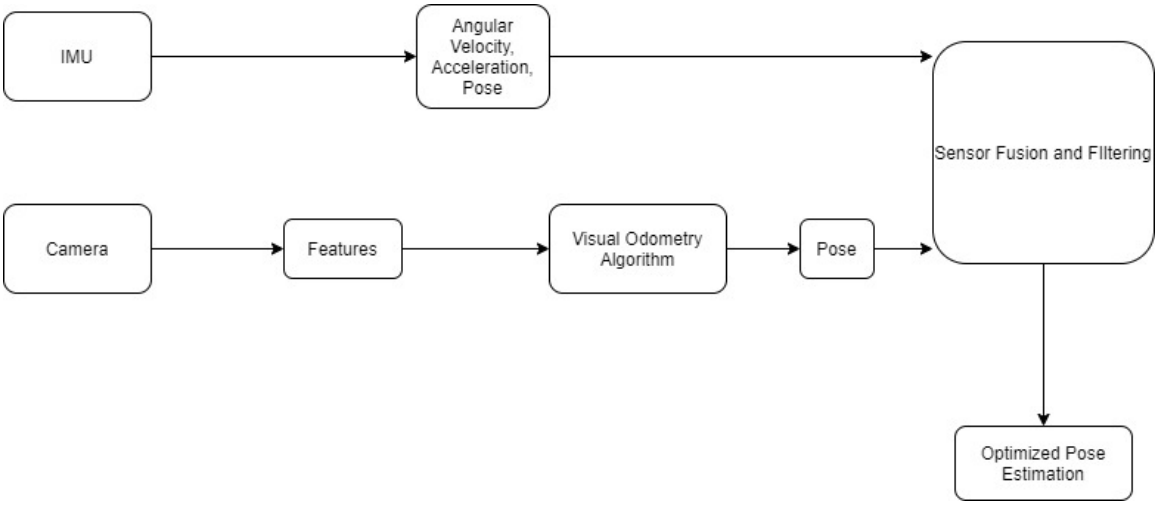

Figure 3.3. General framework for loosely-coupled visual inertial odometry

3.2.2 Tightly-coupled method

In the tightly-coupled approach for visual-inertial sensor fusion, the visual and inertial parameters and raw measurements are combined and then fed to a single optimization function so that their states can be jointly estimated. Leutenegger et al. [38] proposed this way of formulation as below:

$$J(x) = \sum_{i=1}^{I}\sum_{k=1}^{K}\sum_{j\in\mathcal{J}(i.k)} e_r^{i.j.k} W_r^{i.j.k} e_r^{i.j.k} + \sum_{k=1}^{K-1} e_s^k W_s^k e_s^k \qquad (3.1)$$

where, $J(x)$ is the cost function which includes the weighted reprojection errors e_r and weighted temporal errors from IMU e_s. Here, i is the camera index, k is the frame index and j is the image feature index.

This approach can be implemented in direct visual odometry methods as proposed by Concha et al. [39]. The optimization function minimized the IMU residual errors, IMU pose errors, and photometric errors. The assumption in the optimization function is that between two consecutive frames, the first frame is considered to be constant, so the second frame is adjusted based on that. Usenko et al. [40] also proposed a similar energy function that takes into account the IMU and Image errors to estimate the agent's pose and translational velocity in addition to IMU biases.

A tightly-coupled approach results can be more accurate than a loosely-coupled approach by considering the correlation between both sensors. However, it is hard to implement and debug, and it is computationally more expensive than the loosely-coupled method.

The general framework for a loosely-coupled method is shown in Figure 3.4.

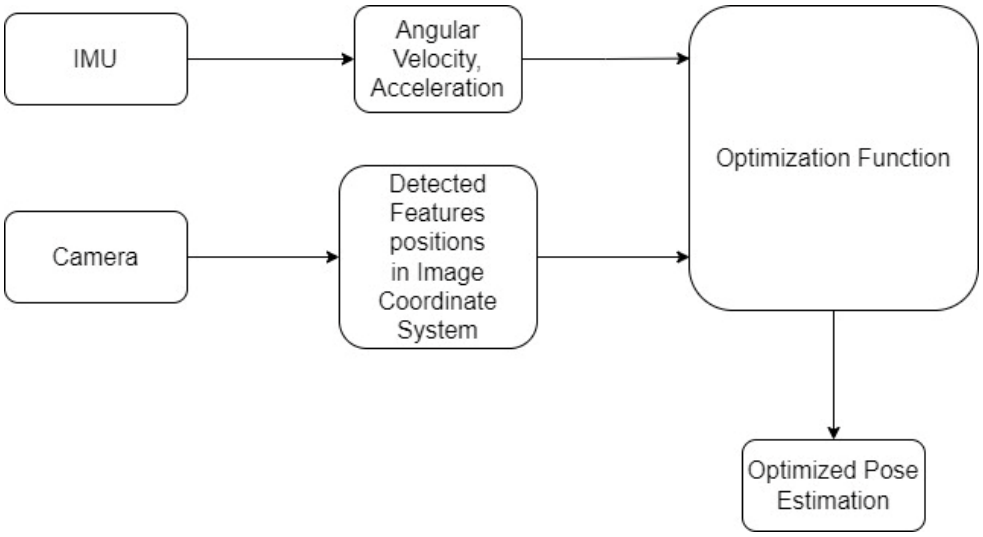

Figure 3.4. General framework for tightly-coupled visual inertial odometry

Choice of method

A loosely-coupled method for the fusion of visual odometry and inertial data of IMU is chosen due to its flexibility and being computationally less expensive. In this framework, a feature-based method is chosen over the direct method for the visual odometry part. The reason is that the direct method is computationally expensive, harder to implement, and unable to handle large frame-to-frame motions. Because this work is assumed to be implemented on ground vehicles, and because these applications involve high speeds, it is logical to choose a visual odometry algorithm that can handle fast movements.

Chapter 4

Implementation

This chapter discusses the implementation of visual-inertial odometry using an error state Kalman Filter and the fuzzy logic to make the filter adaptive. This chapter begins with a brief description of coordinate frames used by the *KITTI vision benchmark suit*. It continues to go on to the details of the inertial and visual sensor data and the fuzzy engine which has been designed and used in this work.

4.1 Coordinate Frames

As illustrated in Figures 4.1 and 4.2, the Visual-Inertial setup, which has been used here, consists of three major coordinate frames. Differentiating these frames and understanding the transformations between them is necessary for designing and executing a successful Visual-Inertial algorithm and filter design.

- IMU Frame

  The IMU frame (a.k.a., body frame) is the coordinate system that is rigidly attached to the IMU body and moves along with it. The raw measurements of the IMU are taken in this coordinate system. The raw IMU measurements should be transformed into the NED frame using a transformation matrix. In the presented system, the IMU coordinate frame is defined as +X forward, +Y left, +Z up.

- Camera Frame

  The Camera Frame is referred to the coordinate system of the left camera, which has been defined in chapter 2. This frame is attached to the agent and moves along with it. The calculated DCM rotation matrix and translation vector of the vehicle's movement are expressed in this coordinate system frame; then, these obtained data are used to calculate the agent's pose and transformed into the NED frame. For the system in this work, the Camera coordinate frame is defined as +X right, +Y down, +Z forward.

- NED Frame

  The NED frame (a.k.a., world or local frame) is a static reference frame with an arbitrary Origin used to estimate the vehicle's motion in that coordinate system. The Origin of this coordinate frame is normally chosen to be the starting point of the vehicle's movement. The NED frame is defined as +X right, +Y forward, +Z down.

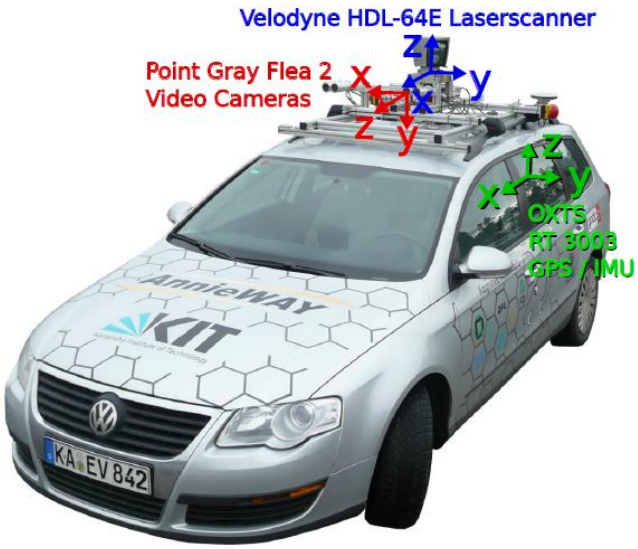

Figure 4.1. Autonomous Driving Platform AnnieWAY used in KITTI vision benchmark suit [41].

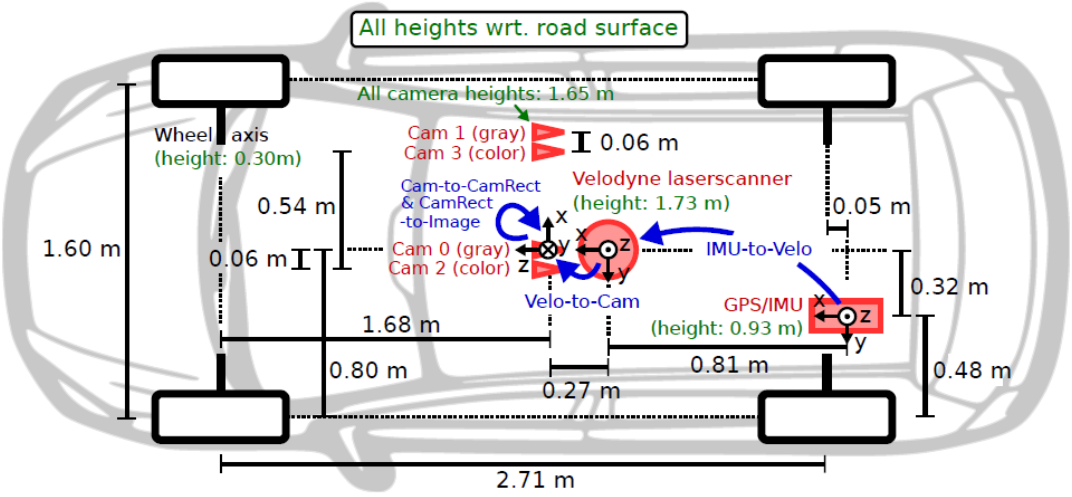

Figure 4.2. Autonomous Driving Platform AnnieWAY used in KITTI vision benchmark suit [41].

4.2 Error-State Extended Kalman Filter design

To design the EKF, first, we need the define the error state based on the IMU motion equation described in sections 2.15, (2.19) through (2.23):

$$\delta x = \begin{bmatrix} \delta r^L & \delta v^L & \delta q_b^l & \delta\, b_a & \delta b_g \end{bmatrix}^T \tag{4.1}$$

having IMU motion equations and error state defined, we need to linearize the system equations to obtain F and G matrices. To do so, we start with the quaternion error model $\delta \dot{q}_B^L$. The equation (2.19) can be rewritten as:

$$\delta \dot{q}_B^L = \frac{\partial}{\partial x}\left[\frac{1}{2}\left(\left[\widehat{W}_{LB}^B - B_g + W_g\right]x\right)q_B^L\right] \tag{4.2}$$

it can be seen that $\dot{q}_B^L$ is a function of q_B^L and b_g so we need to do the partial derivative of $\dot{q}_B^L$ with respect to these two parameters. This will yield the quaternion error model $\delta \dot{q}_B^L$ as follows:

$$\delta \dot{q}_B^L = F_q q_B^L - \frac{1}{2}\Omega(q_B^L)\delta b_g \tag{4.3}$$

where, F_q is,

$$F_q = \begin{bmatrix} 0 & -(\widehat{\omega}_{LBx}^B - b_{gx}) & -(\widehat{\omega}_{LBy}^B - b_{gy}) & -(\widehat{\omega}_{LBz}^B - b_{gz}) \\ (\widehat{\omega}_{LBx}^B - b_{gx}) & 0 & (\omega_{LBz}^B) & -(\widehat{\omega}_{LBy}^B - b_{gy}) \\ (\widehat{\omega}_{LBy}^B - b_{gy}) & -(\widehat{\omega}_{LBz}^B - b_{gz}) & 0 & (\widehat{\omega}_{LBx}^B - b_{gx}) \\ (\widehat{\omega}_{LBz}^B - b_{gz}) & (\widehat{\omega}_{LBy}^B - b_{gy}) & -(\widehat{\omega}_{LBx}^B - b_{gx}) & 0 \end{bmatrix} \tag{4.4}$$

and $\Omega(q_B^L)$ is,

$$\Omega(q_B^L) = \begin{bmatrix} -q_1 & -q_2 & -q_3 \\ q_0 & -q_3 & q_2 \\ q_3 & q_0 & -q_1 \\ -q_2 & q_1 & q_0 \end{bmatrix} \tag{4.5}$$

$$q_B^L = [q_0 \quad q_1 \quad q_2 \quad q_3]^T \tag{4.6}$$

to calculate the velocity error model $\delta\dot{v}^L$, using (2.20) we can write:

$$\delta\dot{v}^L = \frac{\partial}{\partial x}[C_B^L[\hat{a}_{IB}^B - b_a + w_a] + g^L] \tag{4.7}$$

here, $\dot{v}^L$ is a function of C_B^L and b_a states and C_B^L(direction cosine matrix) is a function the quaternion vector q_B^L, hence we need to do the partial derivative of $\dot{v}^L$ with respect to only q_B^L and b_a.

$$C_B^L(q_B^L) = \begin{bmatrix} R_{1.1} & R_{1.2} & R_{1.3} \\ R_{2.1} & R_{2.2} & R_{2.3} \\ R_{3.1} & R_{3.2} & R_{3.3} \end{bmatrix} \tag{4.8}$$

where,

$$\begin{cases} R_{1.1} = q_0^2 + q_1^2 - q_2^2 - q_3^2 \\ R_{1.2} = 2(q_1q_2 + q_0q_3) \\ R_{1.3} = 2(q_1q_3 - q_0q_2) \\ R_{2.1} = 2(q_1q_2 - q_0q_3) \\ R_{2.2} = q_0^2 - q_1^2 + q_2^2 - q_3^2 \\ R_{2.3} = 2(q_2q_3 + q_0q_1) \\ R_{3.1} = 2(q_1q_3 + q_0q_2) \\ R_{3.2} = 2(q_2q_3 - q_0q_1) \\ R_{3.3} = q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{cases} \tag{4.9}$$

knowing the above, the velocity error model can be written as:

$$\delta \dot{v}^L = F_{vq} \delta q_B^L + C_B^L \delta b_a \tag{4.10}$$

where,

$$F_{vq} = \begin{bmatrix} F_{vq0} & F_{vq1} & F_{vq2} & F_{vq3} \\ -F_{vq3} & -F_{vq2} & F_{vq1} & F_{vq0} \\ F_{vq2} & -F_{vq3} & -F_{vq0} & F_{vq1} \end{bmatrix} \tag{4.11}$$

where, the F_{vq}'s elements can be written as:

$$\begin{cases} F_{vq0} = 2\big(q_0[\hat{a}_{IBx}^B - b_{ax}] - q_3\big[\hat{a}_{IBy}^B - b_{ay}\big] + q_2[\hat{a}_{IBz}^B - b_{az}]\big) \\ F_{vq1} = 2\big(q_1[\hat{a}_{IBx}^B - b_{ax}] - q_2\big[\hat{a}_{IBy}^B - b_{ay}\big] + q_3[\hat{a}_{IBz}^B - b_{az}]\big) \\ F_{vq2} = 2\big(-q_2[\hat{a}_{IBx}^B - b_{ax}] + q_1\big[\hat{a}_{IBy}^B - b_{ay}\big] + q_0[\hat{a}_{IBz}^B - b_{az}]\big) \\ F_{vq3} = 2\big(-q_3[\hat{a}_{IBx}^B - b_{ax}] - q_0\big[\hat{a}_{IBy}^B - b_{ay}\big] + q_1[\hat{a}_{IBz}^B - b_{az}]\big) \end{cases} \tag{4.12}$$

for the position error model $\delta \dot{r}^L$, using (2.21) we can write:

$$\delta \dot{r}^L = \frac{\partial}{\partial x}[v^L] \tag{4.13}$$

here, $\dot{r}^L$ is a function in v^L states so we need to do the partial derivative of $\dot{r}^L$ with respect to only v^L. This will give us the following velocity error model:

$$\delta \dot{r}^L = I_{3\times 3} \delta v^L \tag{4.14}$$

where, $I_{3\times 3}$ is the identity matrix.

for the biases error model, we already have them as linear random walk model so $\delta\dot{b}_g$ and $\delta\dot{b}_a$ are zeros.

Given the above linearized error equations, we will be able to writhe the F and G matrices as follows:

$$F = \begin{bmatrix} 0_{6x3} & I_{3x3} & 0_{3x10} \\ & 0_{3x3} & F_{vq\ 3x4} & 0_{3x3} & C_B^L{}_{3x3} \\ 0_{4x6} & F_{q\ 4x4} & -\frac{1}{2}\Omega(q_B^L)\ 4x3 & 0_{4x3} \\ 0_{6x16} \end{bmatrix}_{16 \times 16} \quad (4.15)$$

The derivation of the matrix G is similar to matrix F except that in matrix G we do the partial derivative with respect to the random zero-mean stochastic noise term "w" not the states "x". Thus:

$$G(t) = \frac{\partial f(x(t).\bar{u}(t)+w(t))}{\partial w} \quad (4.16)$$

so the matrix G can be written as:

$$G = \begin{bmatrix} 0_{3x14} \\ 0_{3x3} \quad C_B{}^L{}_{3x3} \quad 0_{3x8} \\ 0_{4x6} \quad -\frac{1}{2}\Omega(q_B^L)\,4x3 \quad 0_{4x5} \\ 0_{3x8} \quad I_{3x3} \quad 0_{3x3} \\ 0_{3x11} \quad I_{3x3} \end{bmatrix}_{16 \times 14} \tag{4.17}$$

having F and G matrices, now we need to define the measurement model to be able to fuse VO data with IMU. Generally, the measurement model can be written as:

$$y(t) = h\big(x(t)\big) + v(t) \tag{4.18}$$

in loosely-coupled sensor fusion the measurement model is linear and is defined as:

$$\delta y(t) = H\delta x(t) \tag{4.19}$$

the above can be written as:

$$\delta y = \begin{bmatrix} N_{VO} - N_{IMU} \\ E_{VO} - E_{IMU} \\ D_{VO} - D_{IMU} \\ v_{nVO} - v_{nIMU} \\ v_{eVO} - v_{eIMU} \\ v_{dVO} - v_{dIMU} \end{bmatrix} =$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \delta N \\ \delta E \\ \delta D \\ \delta v_n \\ \delta v_e \\ \delta v_d \\ \delta q_0 \\ \delta q_1 \\ \delta q_2 \\ \delta q_3 \\ \delta b_{gx} \\ \delta b_{gy} \\ \delta b_{gz} \\ \delta b_{ax} \\ \delta b_{ay} \\ \delta b_{az} \end{bmatrix} \tag{4.20}$$

where, δy is called innovation sequence, which is the difference between the obtained agent's pose and velocity from the VO algorithm and its predicted pose by IMU measurements.

Having all the necessary elements of the Kalman Filter, now we can use the mentioned algorithm in section 2.2.2 from equations (2.54) throughout (2.64).

Since the system is loosely-coupled, the aim of the sensor fusion step becomes getting the final and corrected results as close as possible to the VO data. The reason for that is we are assuming the VO data have absolute certainty. However, in reality, this assumption does not hold. The errors that are occurring will cause the whole system to drift in the long runs.

Furthermore, when we implement the Kalman Filter, we consider that the frequency of data coming from IMU is different from that of VO. This problem is known as the multi-rate Kalman Filter and is considered a research field. Numerous research has been done in this field [42]–[43]. On the other hand, knowing that the final corrected data will be fed to the vehicle's control system, we need accurate and high-frequency data. In this work, to be able to obtain the corrected results from the filter successfully, we simplify the problem by propagating the error states at a base sample rate (IMU rate) and propagating the covariance at all of the time steps. Then, once a measurement is observed, we perform the Kalman correction step and update the covariance and error state.

The general configuration of the system is illustrated in Figure 4.3.

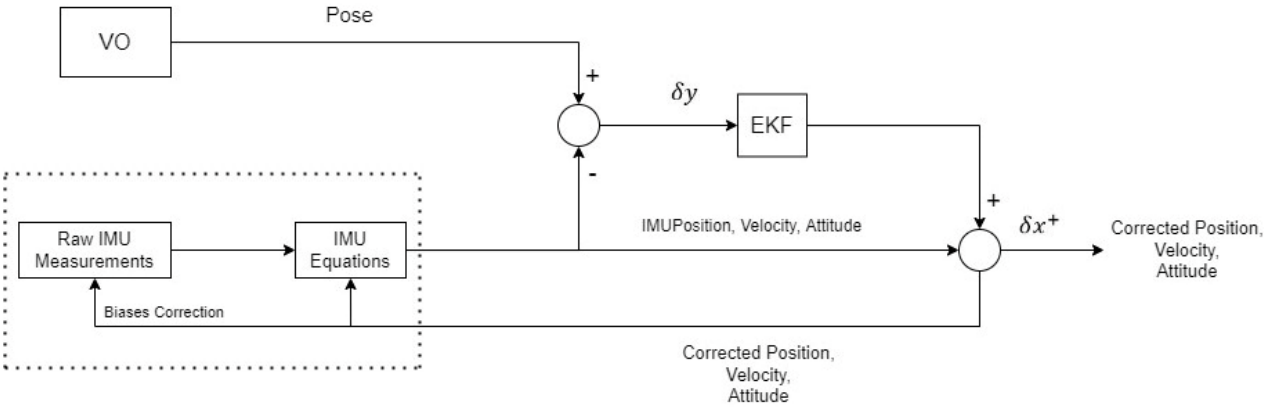

Figure 4.3. General system configuration

where, δx^+ is the corrected states produced by Kalman Filter after the update step is performed.

The next step in designing the sensor fusion system is tuning the EKF. There are few known conventional methods for tuning the noise parameters in EKF. The nominal design parameters in matrix Q, can be found by using Gauss-Markov (GM) random process [52] and Allan Variance (AV) method [53]. However, the nominal design parameters in matrix R are found based on observation of measurements. Next, we apply the Design of Experiment (DoE) approach similar to what Gessesse [24] used to study the effect of varying the EKF noise parameters on performance. The concept of the DoE that we used can be explained as follows. We assume the elements of matrix R of the EKF are given by $R = diag([R_r \quad R_P \quad R_h])$. The effect of changing these parameters can be measured using the fitness function defined in the equation (5.3). We vary the parameters of R around the previously guessed nominal design points (e.g. $diag([0.5 \quad 0.5 \quad 0.5])$). We refer to the increased value of parameters by (+) and the decreased values by (-). Since there are three elements, we need eight experiments for a full factorial experiment. In each test, we vary the parameters of matrix R and run EKF and calculate the fitness function. The results can be shown in the form of the table below:

Table 4.1 Effect of changing R matrix parameters

Experiments	R_r	R_p	R_h	Fitness function
1	0.05 (-)	0.05	0.05	a_1
2	0.05 (-)	0.05	0.95	a_2
3	0.05 (-)	0.95	0.05	a_3
4	0.05 (-)	0.95	0.95	a_4
5	0.95 (+)	0.05	0.05	a_5
6	0.95 (+)	0.05	0.95	a_6
7	0.95 (+)	0.95	0.05	a_7
8	0.95 (+)	0.95	0.95	a_8

To study the effect of changing each parameter (e.g. R_r), we calculate two sets of averages of the fitness function, one for increased values and one for decreased values, as below:

$$Average\ fitness\ function\ (-)_{R_r} = \frac{a_1+a_2+a_3+a_4}{4} = b_1 \quad (4.21)$$

$$Average\ fitness\ function\ (+)_{R_r} = \frac{a_5+a_6+a_7+a_8}{4} = b_2 \quad (4.22)$$

then the overall average effect of the parameter R_r can be written as;

$$S_{R_r} = b_2 - b_1 = c_1 \quad (4.23)$$

by applying the same procedure for the other parameters of matrix R, the effect of changing each parameter can be observed by comparing the values of c_1 for each parameters.

4.3 Optimized Kalman Filter

In section 4.2, we discussed the classic implementation of EKF. We propose to find the suitable variables for R and Q matrices for the Kalman Filter by using the particle swarm optimization algorithm (PSO). Not only the PSO algorithm returns a set of parameters for R and Q, but also these parameters are optimal concerning a defined fitness function. It should be noted that the difference between sections 4.2 and 4.1 is how we find and tune the Kalman Filter's parameters. In sections 4.1 and 4.2, the parameters inside the R and Q matrices are constant. Therefore for any state, the same R and Q are assigned to the system.

The covariance matrices, R and Q, are diagonal matrices. We assume that the diagonal value of these matrices is equal. In other words, a constant is multiplied by identity matrix. Considering multiple values for covariance matrices diagonals is an interesting study for future research.

4.4 Adaptive Kalman Filter

It is observed that the values of R and Q have a significant impact on the estimation performance. As discussed in chapter 2, a filter with adaptive values for R and Q outperforms a filter with constant values of R and Q [24]. In this study, the usage of Fuzzy

Logic Controllers (FLC) is proposed to make the filter adaptive. The FLC intakes some of the performance indicators and returns suitable values for R and Q. Now, the master question in this thesis is designing a setting for FLC output parameters.

The heuristic algorithms are tools that not only can find a proper set of parameters for an application but also those parameters are the optimal solutions concerning a fitness function. In this work, we use the Particle Swarm Optimization (PSO) algorithm to find the Output parameter of the fuzzy system.

The structure of the proposed method is depicted in Figure 4.4.

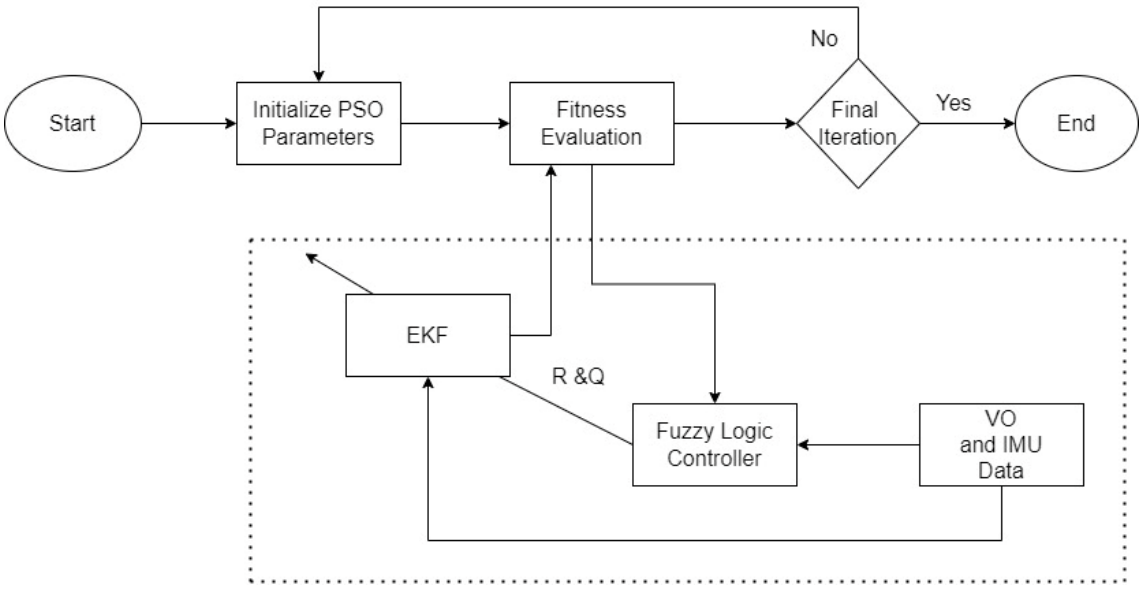

Figure 4.4. Fuzzy adaptive system

where, start is referring to starting the calculations in the training stage.

4.5 Fuzzy Implementation

In this section, we are going to describe the Fuzzy Logic Controller's (FLC) structure. We are using a Takagi-Sugeno (TS) fuzzy system [44]. The proposed method includes two inputs. The inputs are described as follows:

$$Input\ 1\ (e) = X_{VO}(i) - X_{fused}(i).\ \ i = 1.2. \ldots . N$$

$$Input\ 2\ (\dot{e}) = \frac{(X_{VO}(i+1) - X_{fused}(i+1)) - (X_{VO}(i) - X_{fused}(i))}{\Delta t_d}.$$

$$i = 1.2. \ldots . N \tag{4.24}$$

where, $X_{VO}(i)$ is the i^{th} pose data obtained by odometry standalone algorithm, $X_{fused}(i)$ is i^{th} corrected estimation of agent's pose after fusion and Δt_d is the time step of the odometry data.

There are five membership functions for each input. The membership functions are triangular and uniformly distributed in the interval of [0,1]. The inputs in (5.21) are mapped to the interval of [0,1] before being used in the fuzzy engine. The membership functions are depicted in Figure 4.5. We chose the triangular membership function over the other forms of membership functions like Gaussian membership functions for the following reasons. Although the Gaussian membership function can be more precise, they are considerably slower in simulation time due to computational complexity. On the other hand, it is observed that nearly the same precision can be achieved by increasing the

number of triangular membership functions in our application while keeping the computational cost considerably lower in comparison to the scenarios in which the Gaussian membership function is used. Hence, the triangular membership function makes the system faster in simulation with nearly the same precision as a system using the Gaussian membership function. Furthermore, triangular membership functions are easier to implement.

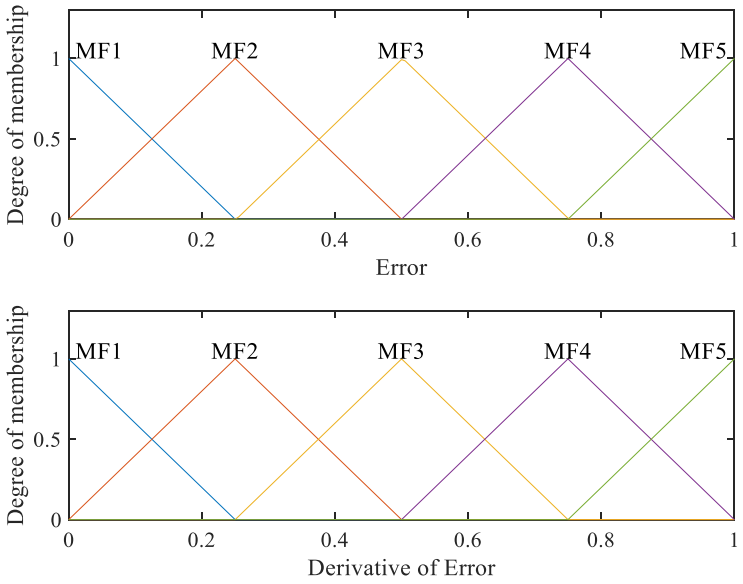

Figure 4.5. Membership functions

In order to map the errors into the interval of [0,1], the maximum and minimum errors must be known. The maximum error can be varied for each sequence of the KITTI benchmark. Thus, for each scenario, first, we have to determine the error bounds.

There are two inputs and five membership functions for each input, so there are a total number of 25 rules for the FLC. In other words, there are 25 output parameters that must be set. To decrease the computational complexity, we only take the fired rules into account to calculate the FLC output.

4.6 Fuzzy Adaptive Weighted EKF

Sasiadek et al. [47] used Mamdani fuzzy inference system [50] to adjust the exponential weighting factor , α, of a weighted EKF to prevent the Kalman Filter from divergence. The fuzzy logic adaptive controller (FLAC) will continually adjust the noise strengths in the filter's internal model and tune the filter as well as possible. The details of this system can be found in [47].

Chapter 5

Results and Discussion

The designed system is tested by a visual odometry dataset referred to as the *KITTI vision benchmark suit* [41], and the results are presented in this chapter. To begin with, the KITTI dataset is briefly introduced, followed by a description of the experimental setup, and the results are illustrated and discussed afterward.

5.1 KITTI Vision Benchmark Suit

The KITTI benchmark suit is a collection of 22 data sequences from various sensors such as Light Detection And Ranging (LiDAR), Global Navigation Satellite System (GNSS), IMU, and two stereo systems, Red Green Blue Alpha (RGBA) and grayscale cameras. These sensors were mounted on an autonomous driving platform called Annieway [41], as shown in Figures 4.1 and 4.2. In this benchmark, datasets of scene flow, depth estimation, tracking, visual odometry, and raw measurements of IMU are included. The KITTI data are provided in raw and rectified formats. It includes the camera calibration parameters as well as the projection matrices of the stereo system.

5.2 Experimental Setup

The sensor used in this work are as follows [41]:

- 2 × PointGray Flea2 grayscale cameras (FL2-14S3M-C), 1.4 Megapixels, 1/2" Sony ICX267 CCD, global shutter.
- 1 × OXTS RT3003 inertial and GPS navigation system, 6 axes, 100 Hz, L1/L2 RTK, resolution: 0.02m / 0.1°.

5.3 Evaluation Method

As discussed in section 4.2, the data collection frequencies of IMU and cameras are not the same. Thus, the idea is to test the designed system to see how close the results of optimized EKF and adaptive EKF can become to the standalone VO results and compare the improvement in the results with the conventional EKF method.

To evaluate the results, first, we need to define the translational and velocity Root Mean Square Errors (RMSE) as follows:

$$Translational_{RSME} =$$

$$\sqrt{\frac{1}{N}\sum_{i=0}^{N}\left[\left((X_{VO}(i) - X_{fused}(i)\right)^2 + \left((Y_{VO}(i) - Y_{fused}(i)\right)^2 + \left((Z_{VO}(i) - Z_{fused}(i)\right)^2\right]}$$

(5.1)

$$Velocity_{RSME} =$$

$$\sqrt{\frac{1}{N}\sum_{i=0}^{N}\left[\left(\left(\dot{X}_{VO}(i)-\dot{X}_{fused}(i)\right)^2+\left(\left(\dot{Y}_{VO}(i)-\dot{Y}_{fused}(i)\right)^2+\left(\left(\dot{Z}_{VO}(i)-\dot{Z}_{fused}(i)\right)^2\right]\right.\right.}$$

(5.2)

A second criterion for evaluating the results can be defined as a fitness function calculated as below. This fitness function is used for a comparison between the optimized EKF by PSO and adaptive EKF:

$$Fitness\ function = Translational_{RSME}{}^2 + Velocity_{RSME}{}^2 \qquad (5.3)$$

The reason why the square of RSMEs is used is to intensify the effect of optimizing large errors in comparison to smaller ones. The concept is similar to the Integral of Squared Error (ISE) which is used in control systems.

In addition, the 3D RMSE against the grand truth data for position and velocity is done using the same formulas as (5.3) and (5.2), but instead of VO data, we use the ground truth data.

5.4.1 KITTI Dataset−Sequence 02 Results

In sequence 02, the vehicle starts on a highway and then turns into a residential area in which there are several turns, ups and downs, and numerous trees, which has caused a considerable change in lighting conditions. Furthermore, cars and motorcycles were moving ahead of our vehicle. All of these conditions can cause error and drift, but it can be

seen that our proposed method has performed adequately. In addition, by considering the fitness function in the optimized Kalman Filter and adaptive Kalman Filter in Table 1, it is observed that the results have become about 37% more accurate in the adaptive Kalman Filter. Furthermore, comparing the fitness function for adaptive weighted EKF and the proposed adaptive Kalman Filter, it is observed that the adaptive Kalman filter is 15% more accurate. Table 1 shows the fitness functions obtained by the PSO algorithm and Table 2 shows the position and velocity RMSE obtained using ground truth data. A summary of the results is presented in Figures 5.1 through 5.8.

Figure 5.1 depicts the trajectory obtained by the stereo visual odometry and the trajectory obtained from IMU for KITTI Sequence 02 which starts at 0,0 and ends in the vicinity of 370 m East and 180 m North. It was observed that the estimated trajectory by IMU starts to drift too much. It was observed that the IMU data available in the KITTI benchmark has a low frequency of data collection and is not synced with the images; in some cases, there are some gaps in the data, or the IMU data is largely out-of-order. All of these reasons can lead to significant drifts. More details regarding IMU drifts can be found in section 5.5. In addition, Figures 5.2 through 5.4 show the IMU measurements in the time domain and how IMU's results start to drift over time. In Figures 5.5 and 5.7, the convergence of the fitness function to the final value for the optimized Kalman Filter and the adaptive Kaman Filter are shown, respectively. As shown in Figure 5.5, after 50 iterations, the parameters converge. In Figure 5.7, convergence happens after roughly 350 iterations. Although, as shown in Figure 5.7, the fitness function value is still dropping after roughly 475 iterations, because the rate of changes is very small from iteration 300, the final value of the fitness function is acceptable. Figures 5.6 and 5.8 show the estimated trajectories obtained by the

discussed methods which are conventional Kalman Filter, optimized Kalman Filter, weighted EKF and our adaptive Kalman Filter method. Although a precise comparison of these methods' accuracy can be made by studying Table 1 and 2, it can be seen how the errors of the proposed adaptive Kalman Filter are reduced. Figure 5.8 shows that in the coordinates 600 m to 700 m East and 100 m to 200 m North and also 600 m to 700 m East, and 330 m to 550 m North, the proposed adaptive Kalman Filter is more robust because it showed less fluctuation around the trajectory estimated by stereo visual odometry.

Table 5.1. KITTI Sequence 02 Results

Case	Fitness function
Conventional Kalman Filter	4.6352×10^4
Optimized Kalman Filter by PSO	3.9174×10^4
Adaptive Weighted EKF	3.2730×10^4
Adaptive Kalman Filter	2.8516×10^4

Table 5.2. KITTI Sequence 02 Results

Case	Position RMSE	Velocity RMSE
Conventional Kalman Filter	$91.26\ m$	$0.182\ m/s$
Optimized Kalman Filter by PSO	$88.86\ m$	$0.149\ m/s$
Adaptive Weighted EKF	$87.43\ m$	$0.147\ m/s$
Adaptive Kalman Filter	$77.01\ m$	$0.127\ m/s$

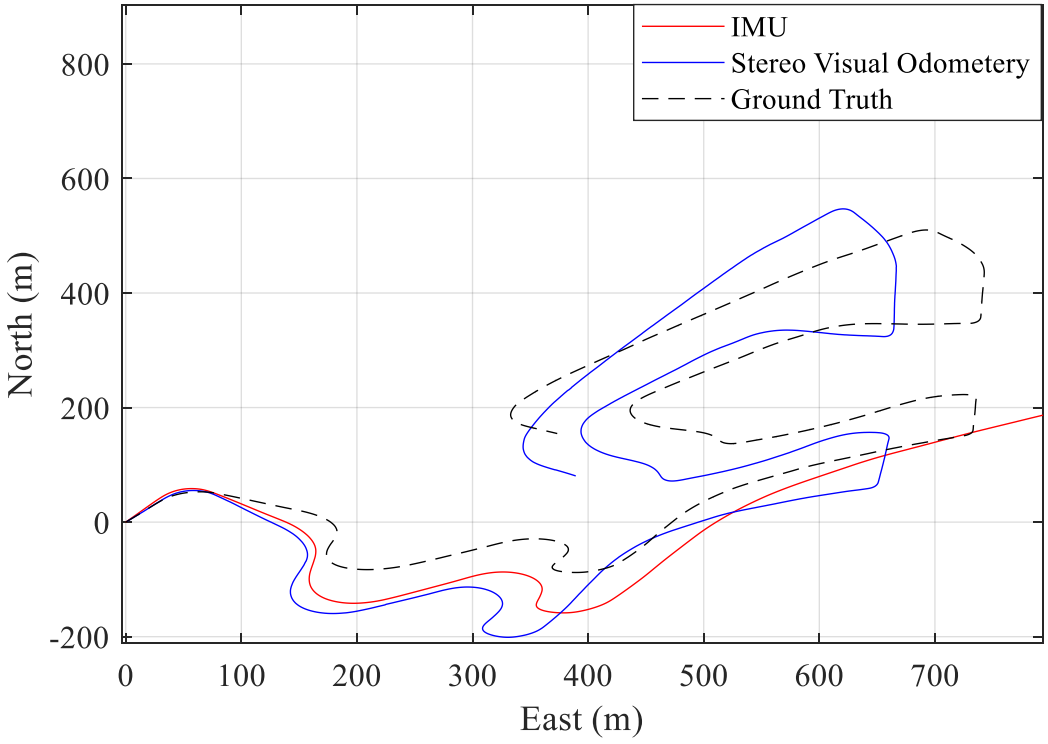

Figure 5.1. VO and IMU estimated trajectory before sensor fusion-KITTI Sequence 02

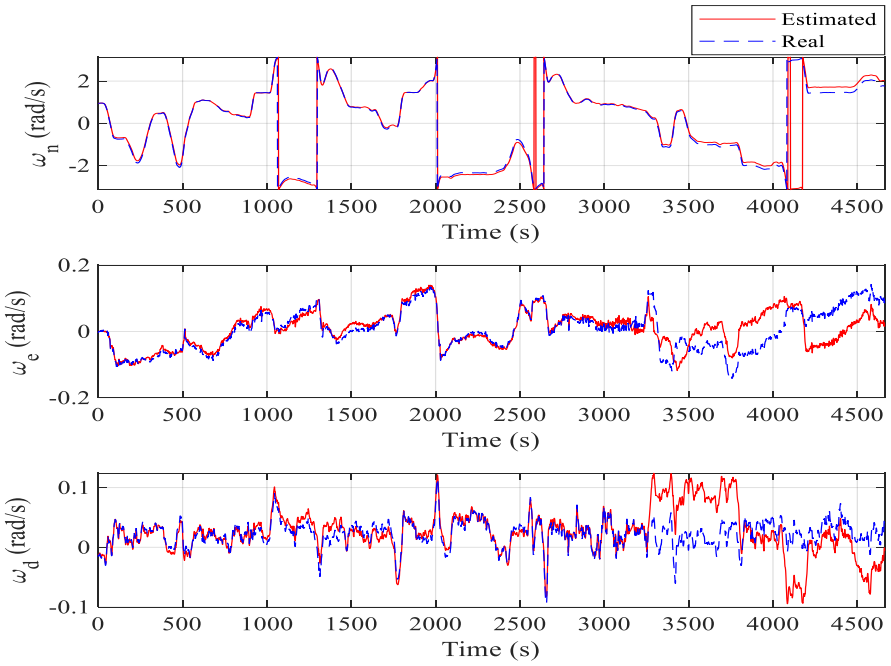

Figure 5.2. Angular velocities estimated by IMU-KITTI Sequence 02

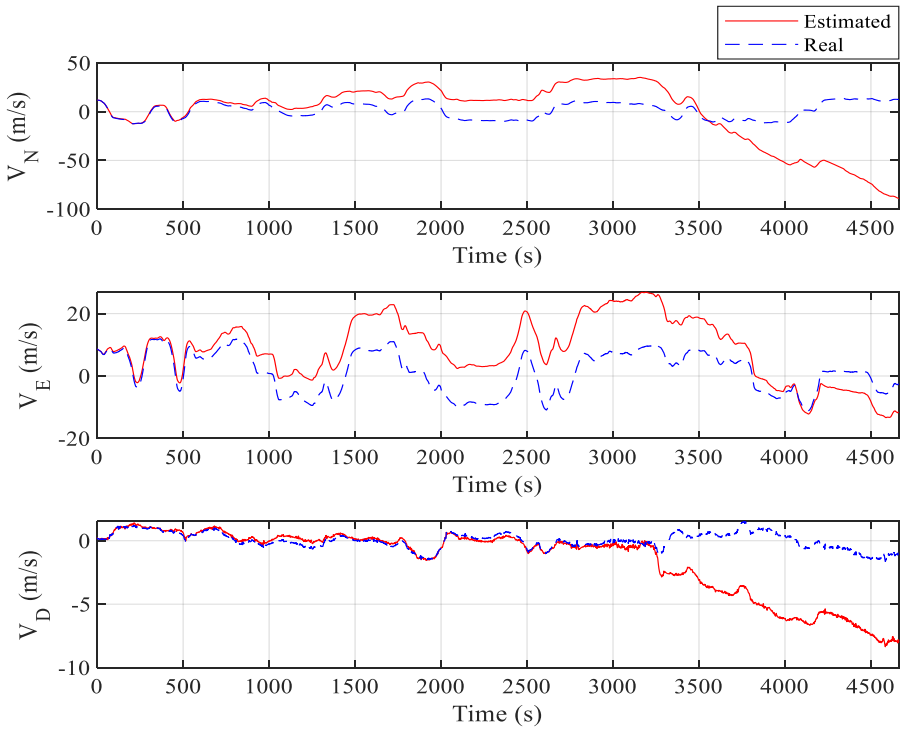

Figure 5.3. Linear velocities estimated by IMU-KITTI Sequence 02

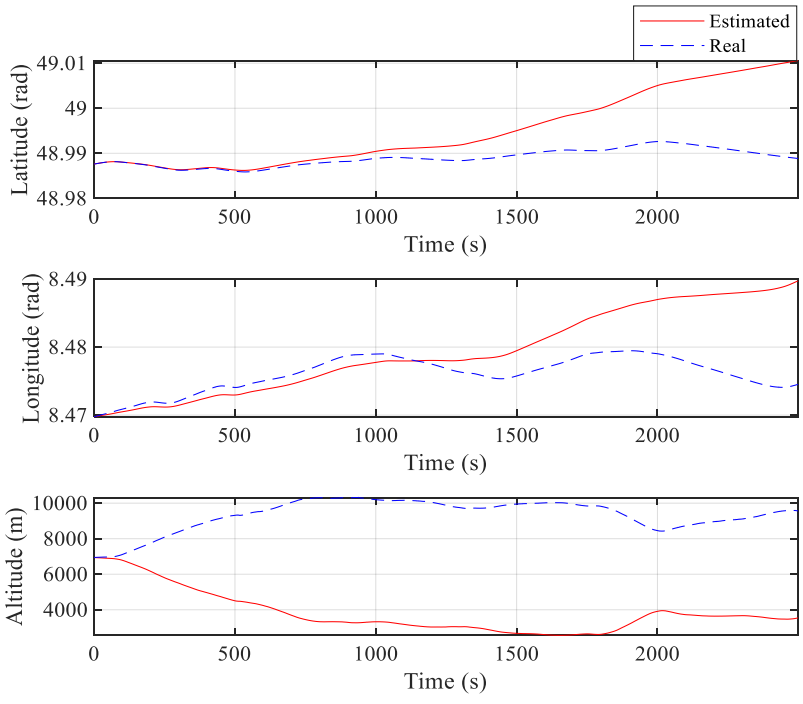

Figure 5.4. Vehicle's position estimated by IMU-KITTI Sequence 02

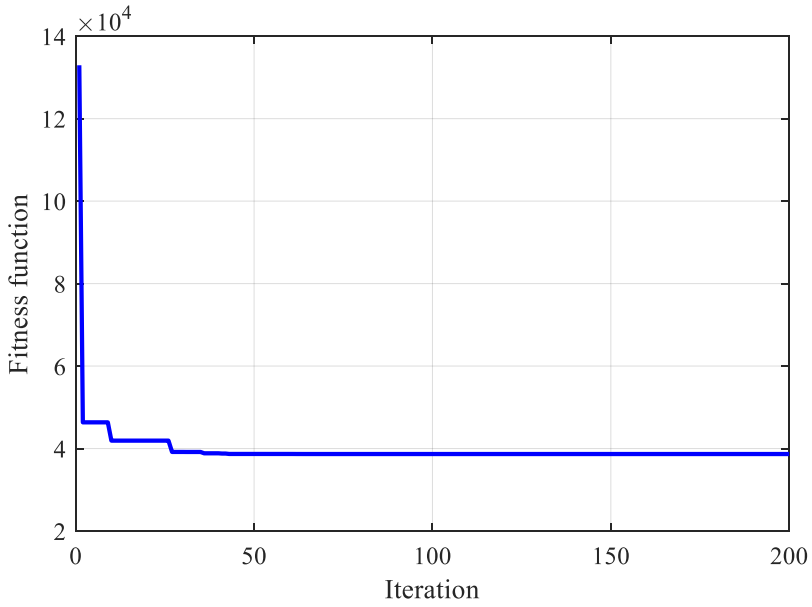

Figure 5.5. Calculated fitness function by PSO algorithm in optimized Kalman Filter-KITTI Sequence 02

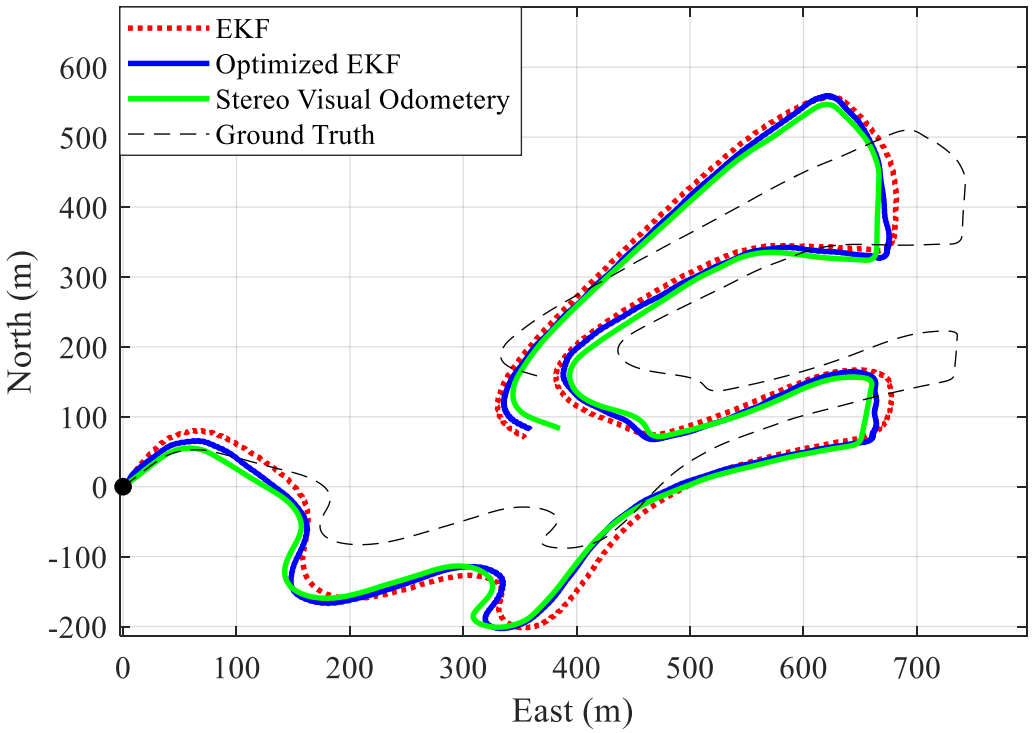

Figure 5.6. Estimated trajectory by conventional and optimized Kalman Filters-KITTI Sequence 02

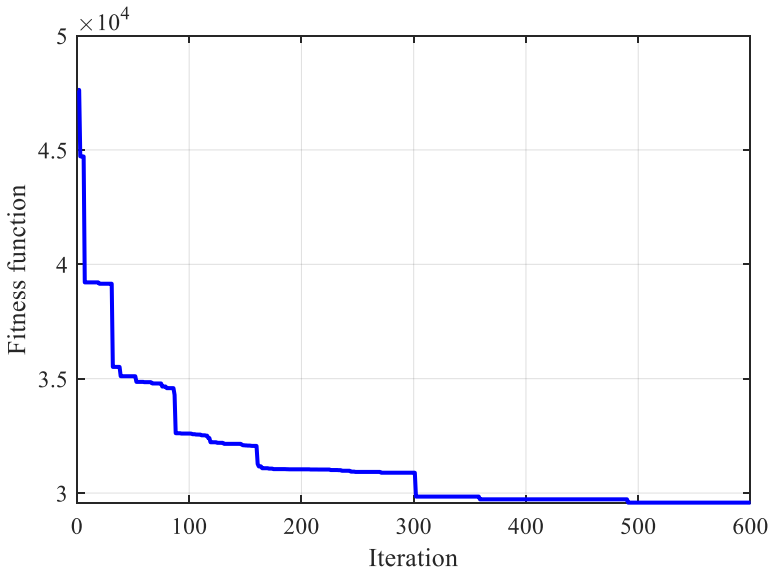

Figure 5.7. Calculated fitness function by PSO algorithm for adaptive Kalman Filter-KITTI Sequence 02

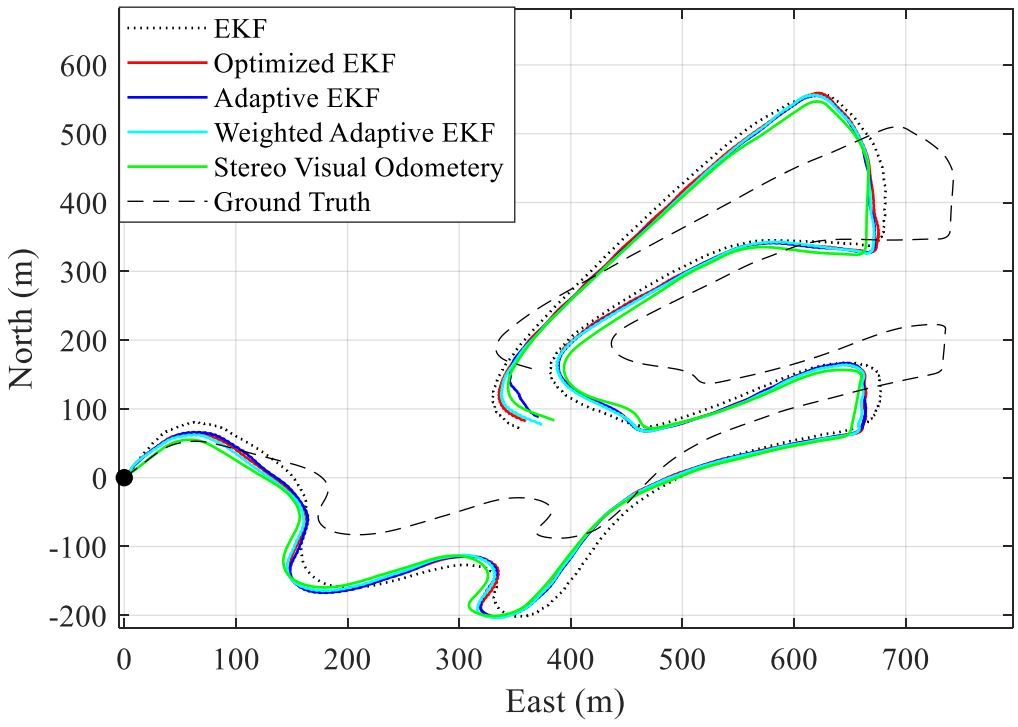

Figure 5.8. Estimated trajectory by optimized and adaptive Kalman Filters-KITTI Sequence 02

5.4.2 KITTI Dataset−Sequence 05 Results

Sequence 05 shows the vehicle moves in a residential area with several turns and less changes in lighting situation compared to sequence 02. In addition, car traffic and pedestrians walking could be observed. Hence, the data coming from cameras are less noisy.

By considering the fitness function in the optimized Kalman Filter and the adaptive Kalman Filter presented in Table 3, it is observed that the results have become about 20% more accurate in the adaptive Kalman Filter. Furthermore, comparing the fitness function for adaptive weighted EKF and the proposed adaptive Kalman Filter, it is observed that the adaptive Kalman filter is 10% more accurate. Table 3 shows the fitness function values and Table 4 shows the position and velocity RMSE obtained using ground truth data. Furthermore, a summary of the results is presented in Figures 5.9 through 5.16.

Figure 5.9 depicts the trajectory obtained by the stereo visual odometry and the trajectory obtained from IMU for KITTI Sequence 05. It was observed that the estimated trajectory by IMU starts to drift too much. It was observed that the IMU data available in the KITTI benchmark has a low frequency of data collection and is not synced with the images; in some cases, there are some gaps in the data, or the IMU data is largely out-of-order. All of these reasons can lead to significant drifts. More details regarding IMU drifts can be found in section 5.5. In addition, Figures 5.10 through 5.12 shows the IMU measurements in the time domain. In Figures 5.13 and 5.15, the convergence characteristic of the fitness functions for the optimized Kalman Filter and the adaptive Kaman Filter are shown, respectively. As shown in Figure 5.13, after roughly 45 iterations the fitness function

converges, and in Figure 5.15, the fitness function converges after roughly 150 iterations. Figures 5.14 and 5.16 show the estimated trajectories obtained by the discussed methods which are conventional Kalman Filter, optimized Kalman Filter, weighted EKF and our adaptive Kalman Filter method. Although a precise comparison of these methods' accuracy can be made by studying Table 3 and 4, it can be seen how the errors of the proposed adaptive Kalman Filter are reduced.

Table 5.3. KITTI Sequence 05 Results

Case	Fitness function
Conventional Kalman Filter	3.5507×10^4
Optimized Kalman Filter by PSO	2.8669×10^4
Adaptive Weighted EKF	2.6126×10^4
Adaptive Kalman Filter	2.406×10^4

Table 5.4. KITTI Sequence 05 Results

Case	Position RMSE	Velocity RMSE
Conventional Kalman Filter	$84.70\ m$	$0.579\ m/s$
Optimized Kalman Filter by PSO	$82.79\ m$	$0.564\ m/s$
Adaptive Weighted EKF	$81.07\ m$	$0.541\ m/s$
Adaptive Kalman Filter	$75.33\ m$	$0.530\ m/s$

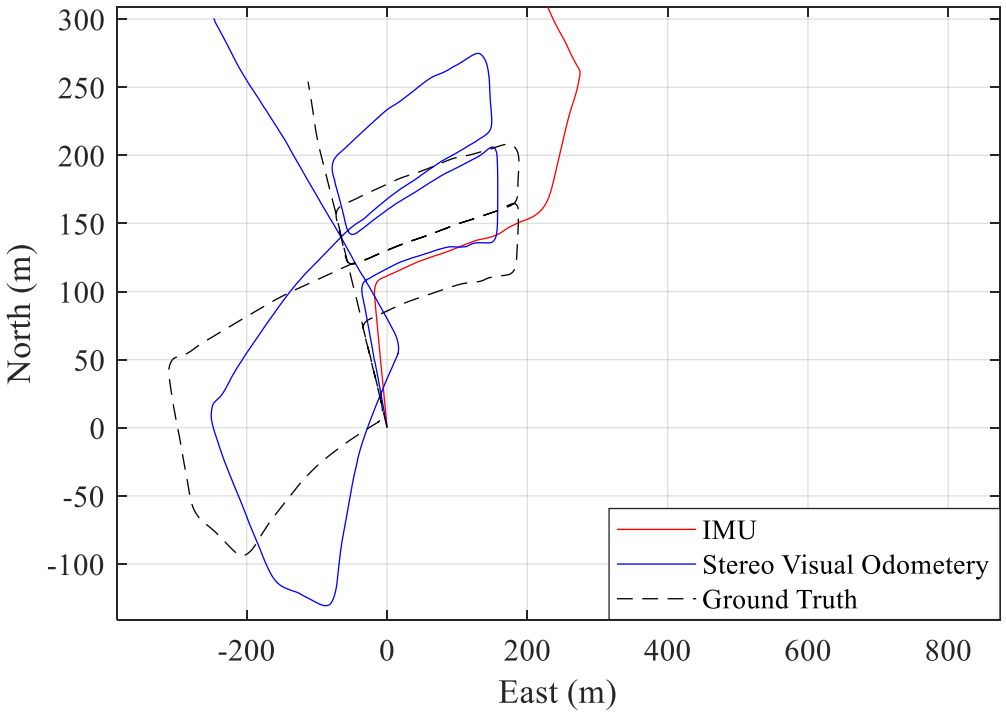

Figure 5.9. VO and IMU estimated trajectory before sensor fusion-KITTI Sequence 05

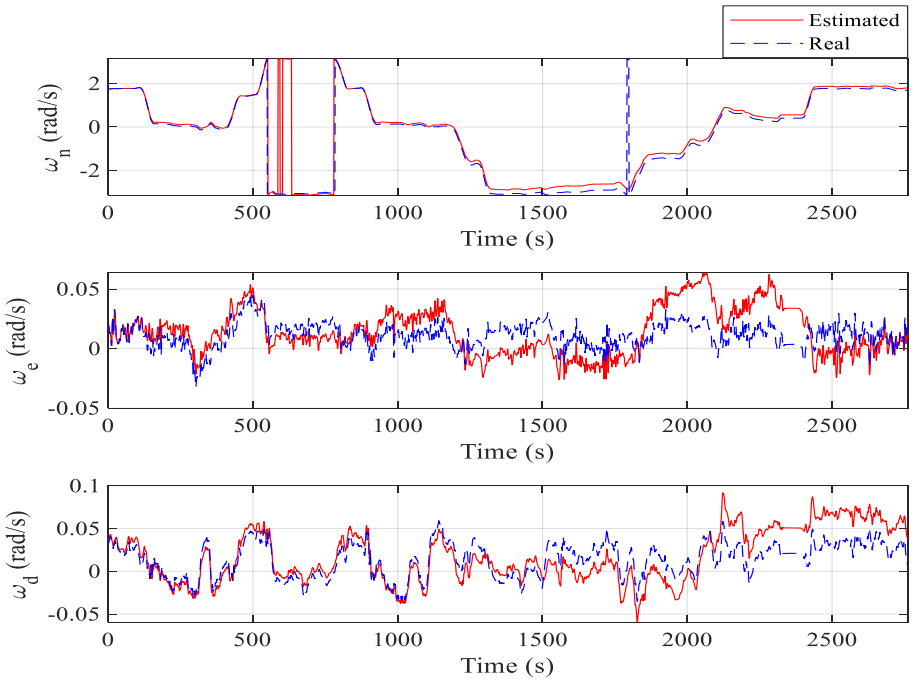

Figure 5.10. Angular velocities estimated by IMU-KITTI Sequence 05

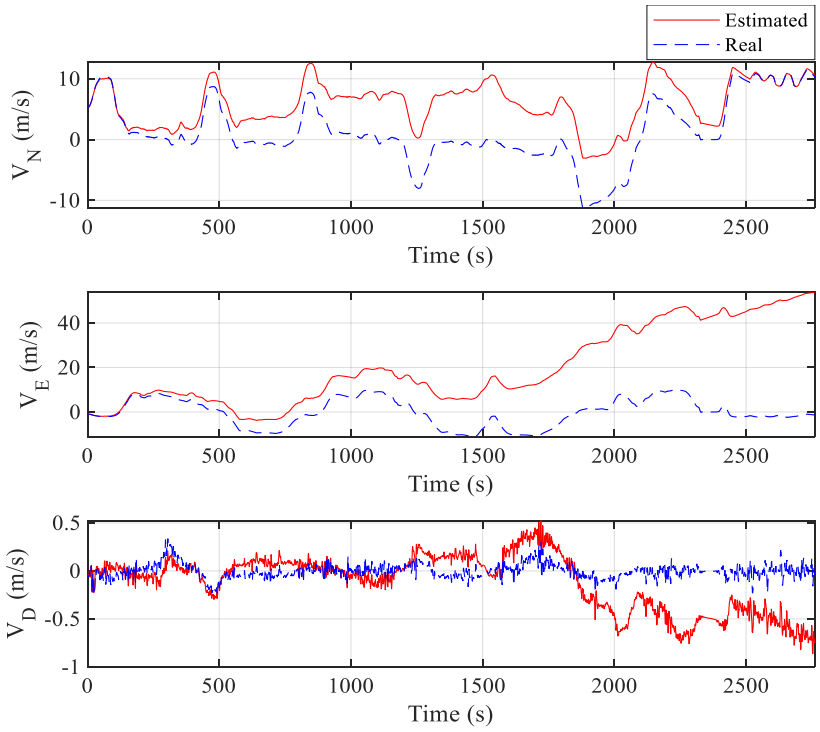

Figure 5.11. Linear velocities estimated by IMU-KITTI Sequence 05

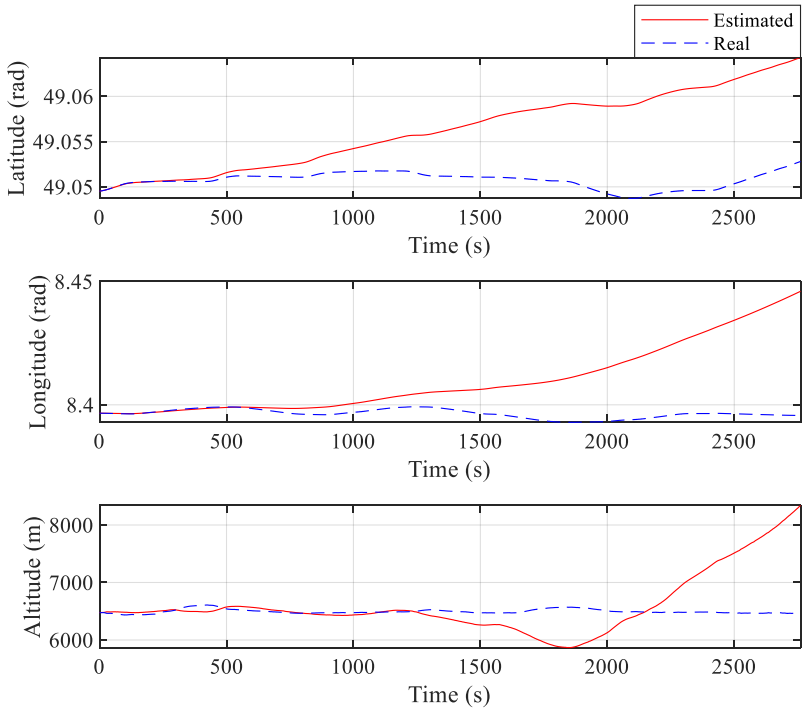

Figure 5.12. Vehicle's position estimated by IMU-KITTI Sequence 05

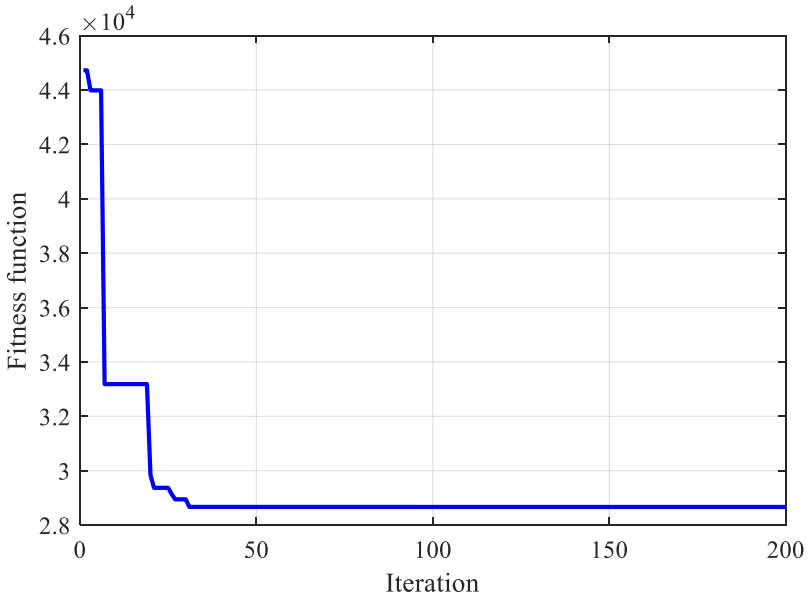

Figure 5.13. Calculated fitness function by PSO algorithm in optimized Kalman Filter-KITTI Sequence 05

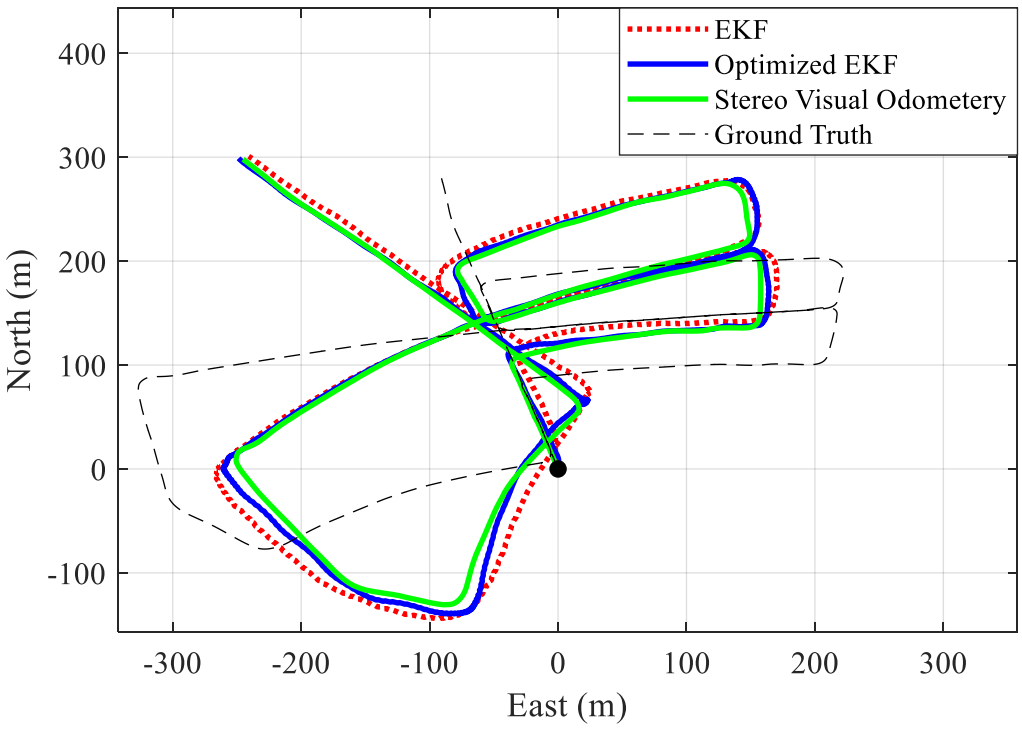

Figure 5.14. Estimated trajectory by common and optimized Kalman Filters-KITTI Sequence 05

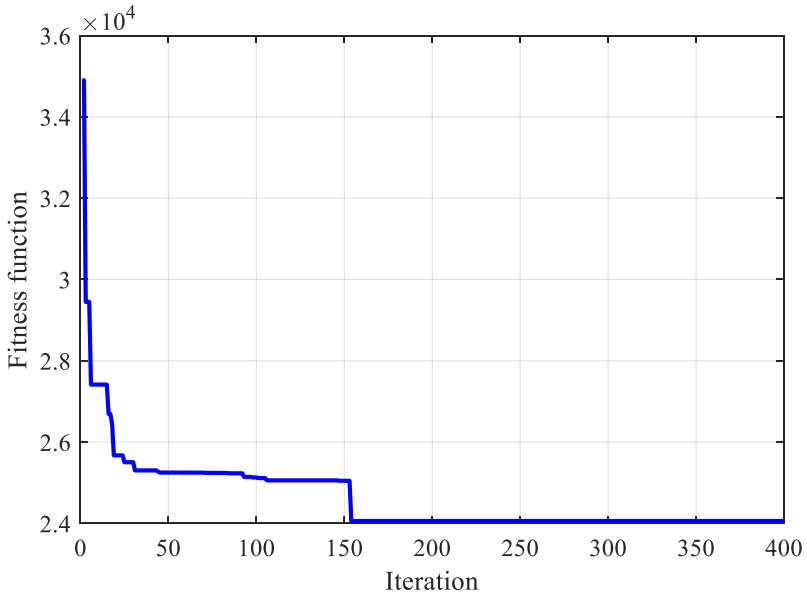

Figure 5.15. Calculated fitness function by PSO algorithm for adaptive Kalman Filter-KITTI Sequence 05

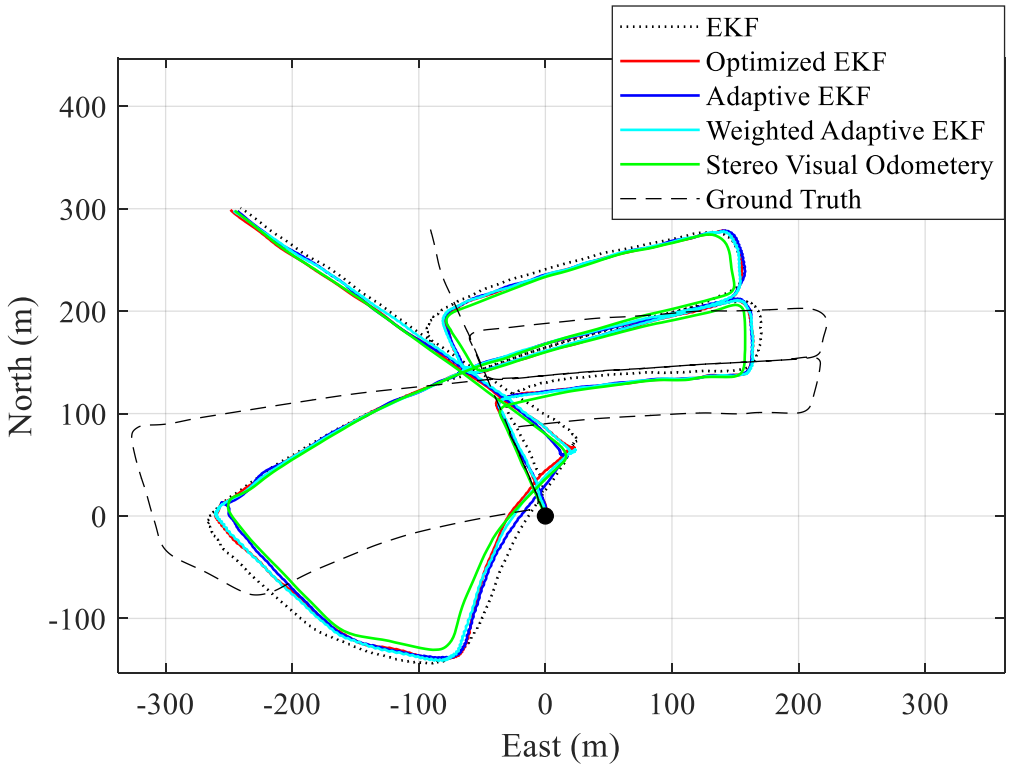

Figure 5.16. Estimated trajectory by optimized and adaptive Kalman Filters-KITTI Sequence 05

5.4.3 KITTI Dataset−Sequence 07 Results

In sequence 07 vehicle drives in a residential area with a busy and dynamic environment. Comparing the fitness function in the optimized Kalman Filter and adaptive Kalman Filter presented in Table 5, it is observed that the adaptive Kalman Filter has proven to be about 15% more accurate than the optimized Kalman Filter. Furthermore, comparing the fitness function for adaptive weighted EKF and the proposed adaptive Kalman Filter, it is observed that the adaptive Kalman filter is 10% more accurate. Table 5 shows the fitness function values, and Table 6 shows the position and velocity RMSE obtained using ground truth data. A summary of the results is presented in Figures 5.17 through 5.24.

Figure 5.17 depicts the trajectory obtained by the stereo visual odometry and the trajectory obtained from IMU for KITTI Sequence 07. In addition, Figures 5.18 through 5.20 show the IMU measurements in the time domain. In Figures 5.21 and 5.23, the convergence characteristic of the fitness functions for the optimized Kalman Filter and the adaptive Kaman Filter are shown, respectively. As shown in Figure 5.21, after roughly 10 iterations, the fitness function converges, and in Figure 5.23, after roughly 90 iterations, the fitness function converges. Figures 5.22 and 5.24 show the estimated trajectories obtained by the discussed methods which are conventional Kalman Filter, optimized Kalman Filter, weighted EKF and our adaptive Kalman Filter method. Although a precise comparison of these methods' accuracy can be made by studying Table 5 and Table 6, it can be seen how the errors of the proposed adaptive Kalman Filter are reduced and how it has less fluctuation around the estimated trajectory by stereo visual odometry.

Table 5.5. KITTI Sequence 07 Results

Case	Fitness function
Conventional Kalman Filter	9792.8
Optimized Kalman Filter by PSO	7861
Adaptive Weighted EKF	7548
Adaptive Kalman Filter	6900

Table 5.6. KITTI Sequence 07 Results

Case	Position RMSE	Velocity RMSE
Conventional Kalman Filter	28.86 m	0.156 m/s
Optimized Kalman Filter by PSO	23.49 m	0.150 m/s
Adaptive Weighted EKF	22.69 m	0.147 m/s
Adaptive Kalman Filter	21.03 m	0.084 m/s

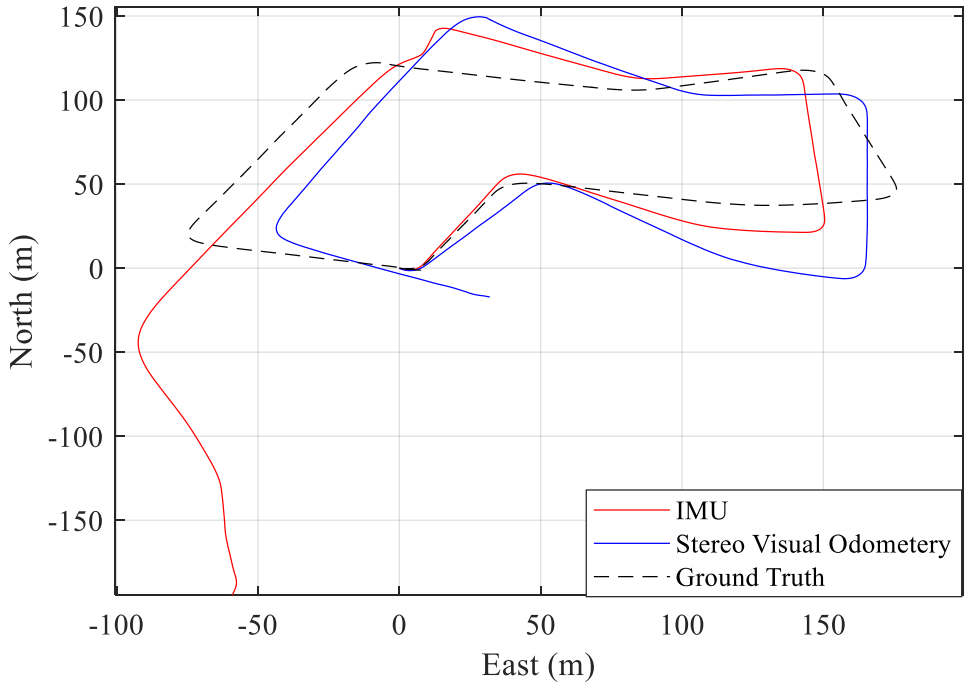

Figure 5.17. VO and IMU estimated trajectory before sensor fusion-KITTI Sequence 07

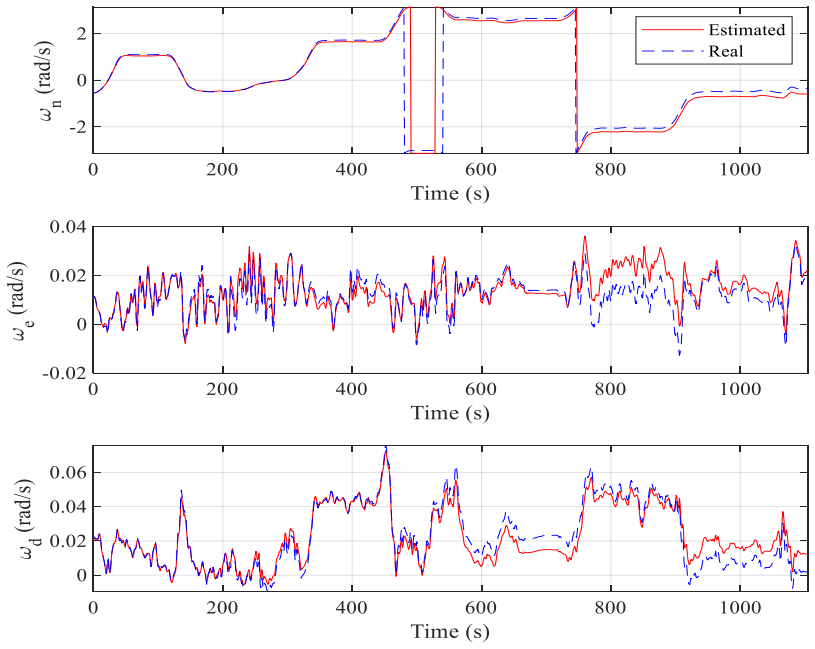

Figure 5.18. Angular velocities estimated by IMU-KITTI Sequence 07

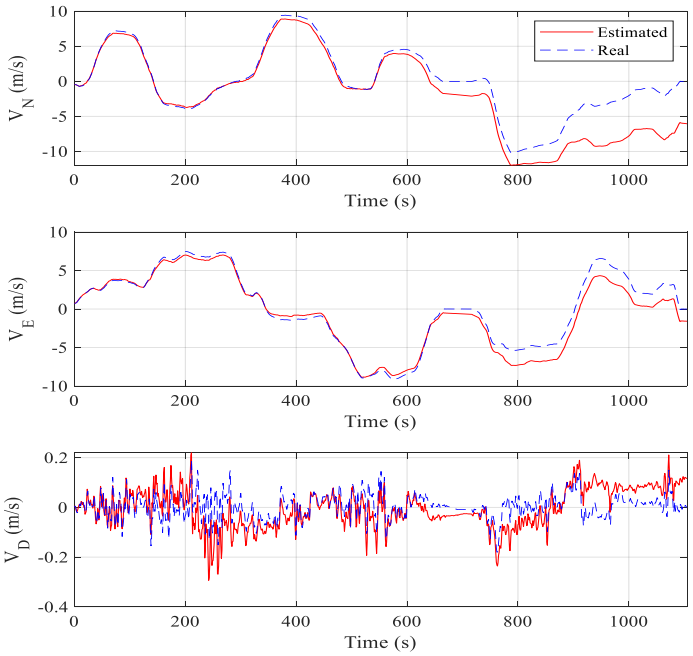

Figure 5.19. Linear velocities estimated by IMU-KITTI Sequence 07

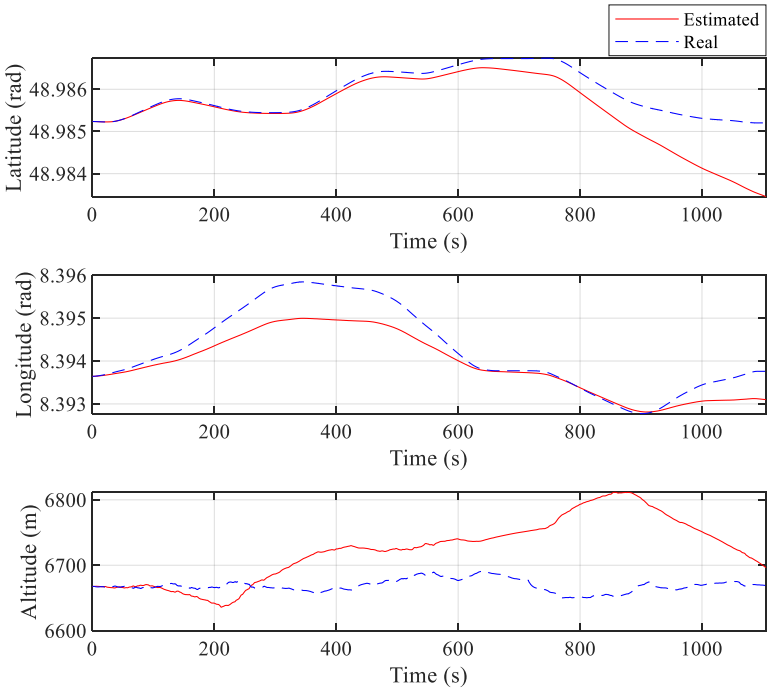

Figure 5.20. Vehicle's position estimated by IMU-KITTI Sequence 07

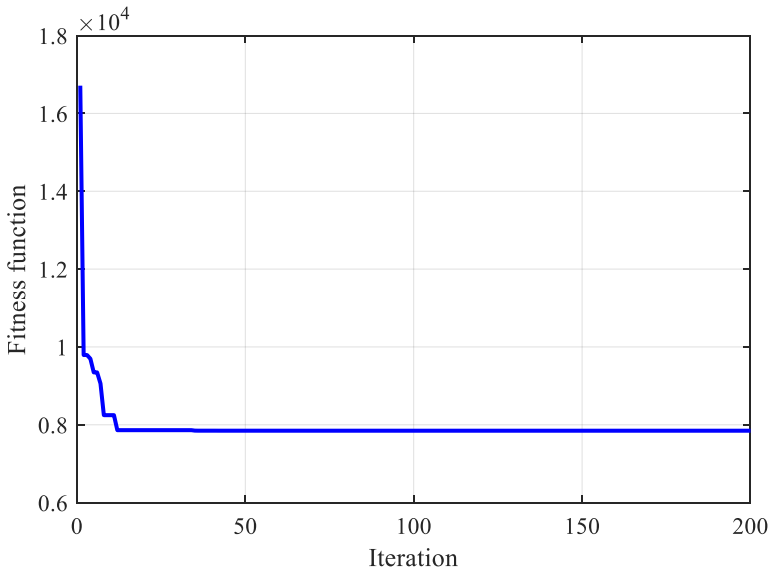

Figure 5.21. Calculated fitness function by PSO algorithm in optimized Kalman Filter-KITTI Sequence 07

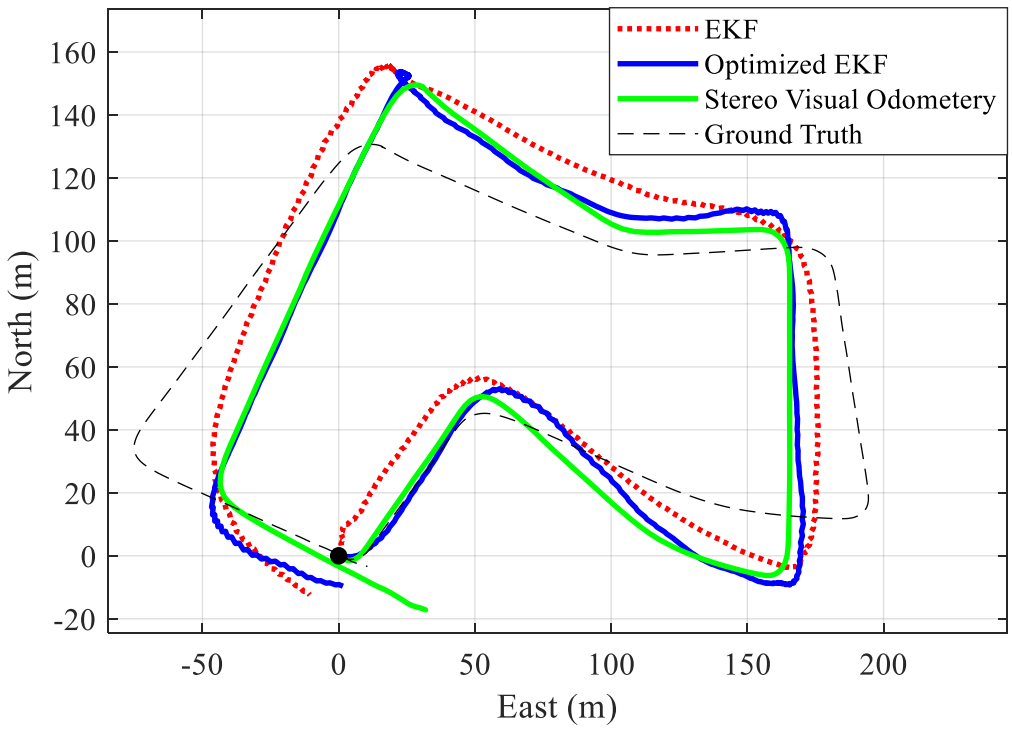

Figure 5.22. Estimated trajectory by common and optimized Kalman Filters-KITTI Sequence 07

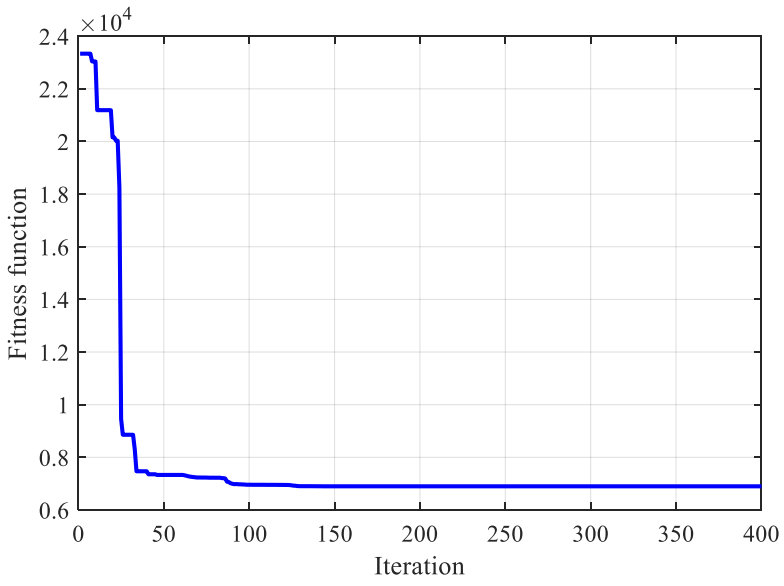

Figure 5.23. Calculated fitness function by PSO algorithm for adaptive Kalman Filter-KITTI Sequence 07

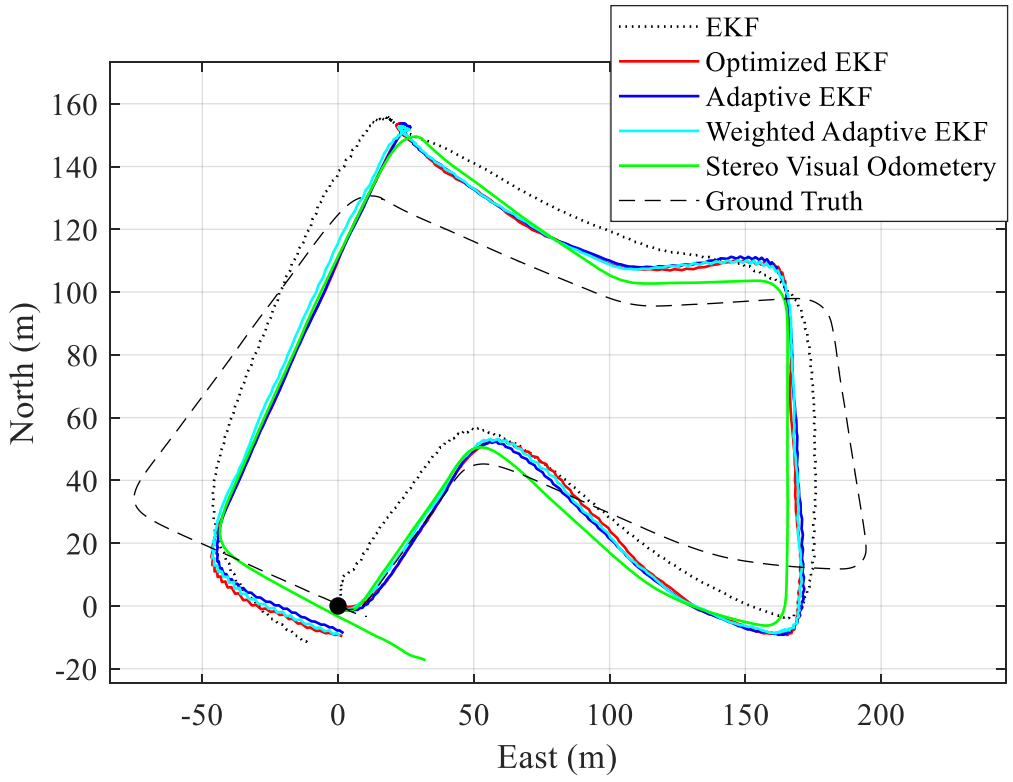

Figure 5.24. Estimated trajectory by optimized and adaptive Kalman Filters-KITTI Sequence 07

5.4.4 KITTI Dataset−Sequence 09 Results

In sequence 09, the vehicle moves in a hybrid environment in a forest area and suburban residential area with several turns and ups and downs and considerable changes in the lighting situation. Considering the fitness function in the optimized Kalman Filter and adaptive Kalman Filter presented in Table 7, it can be seen that the results of the adaptive Kalman filter are 44% more accurate in comparison with the optimized Kalman Filter. Furthermore, comparing the fitness function for adaptive weighted EKF and the proposed adaptive Kalman Filter, it is observed that the adaptive Kalman filter is 14% more accurate. Table 7 shows the fitness function amounts and Table 8 shows the position and velocity RMSE obtained using ground truth data. A summary of the results is presented in Figures 5.25 through 5.32.

Figure 5.25 depicts the trajectory obtained by the stereo visual odometry and the trajectory obtained from IMU for KITTI Sequence 09. In addition, Figures 5.26 through 5.28 shows the IMU measurements in the time domain. In Figures 5.29 and 5.31, the convergence characteristic of the fitness functions for the optimized Kalman Filter and the adaptive Kaman Filter are shown, respectively. As shown in Figure 5.29, after roughly 25 iterations, the fitness function converges, and in Figure 5.31, after roughly 270 iterations, the fitness function converges. Figures 5.30 and 5.32 show the estimated trajectories obtained by the discussed methods which are conventional Kalman Filter, optimized Kalman Filter, weighted EKF and our adaptive Kalman Filter method. Although a precise comparison of these methods' accuracy can be made by studying Table 7 and Table 8, the robustness of the proposed adaptive Kalman Filter can be seen. Also, it can be seen how the errors are

reduced and how it has less fluctuation around the estimated trajectory by stereo visual odometry.

Table 5.7. KITTI Sequence 09 Results

Case	Fitness function
Conventional Kalman Filter	4.0785×10^4
Optimized Kalman Filter by PSO	3.3015×10^4
Adaptive Weighted EKF	2.5915×10^4
Adaptive Kalman Filter	2.2872×10^4

Table 5.8. KITTI Sequence 09 Results

Case	Position RMSE	Velocity RMSE
Conventional Kalman Filter	$87.29\ m$	$0.253\ m/s$
Optimized Kalman Filter by PSO	$75.98\ m$	$0.232\ m/s$
Adaptive Weighted EKF	$68.85\ m$	$0.198\ m/s$
Adaptive Kalman Filter	$66.30\ m$	$0.182\ m/s$

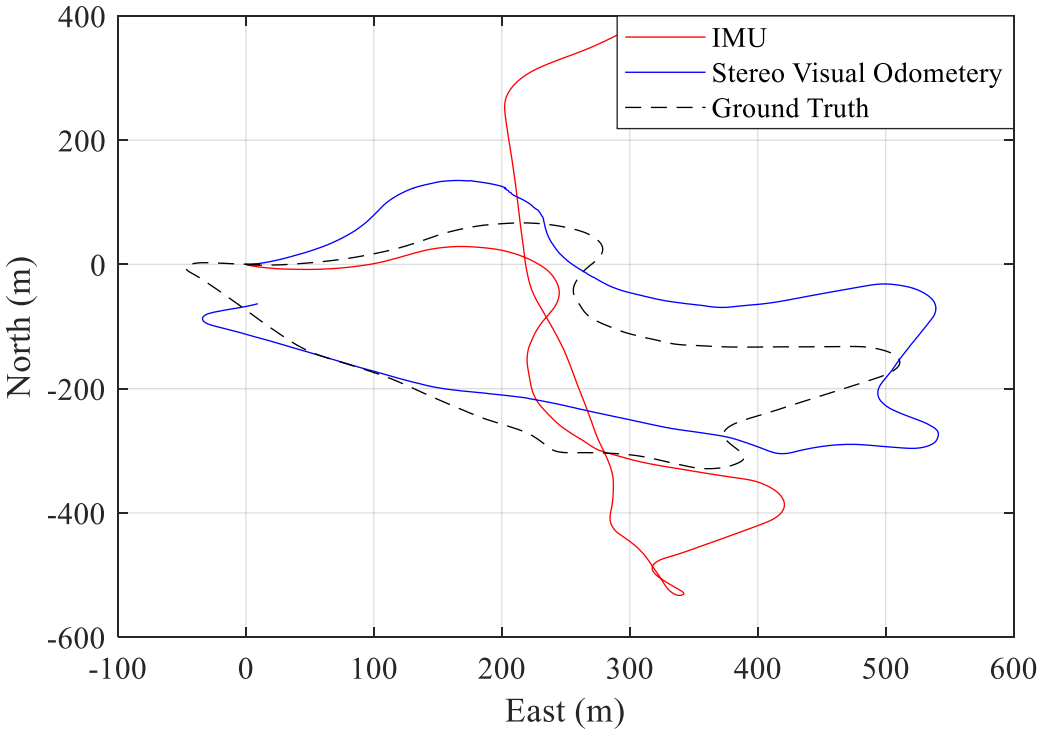

Figure 5.25. VO and IMU estimated trajectory before sensor fusion-KITTI Sequence 09

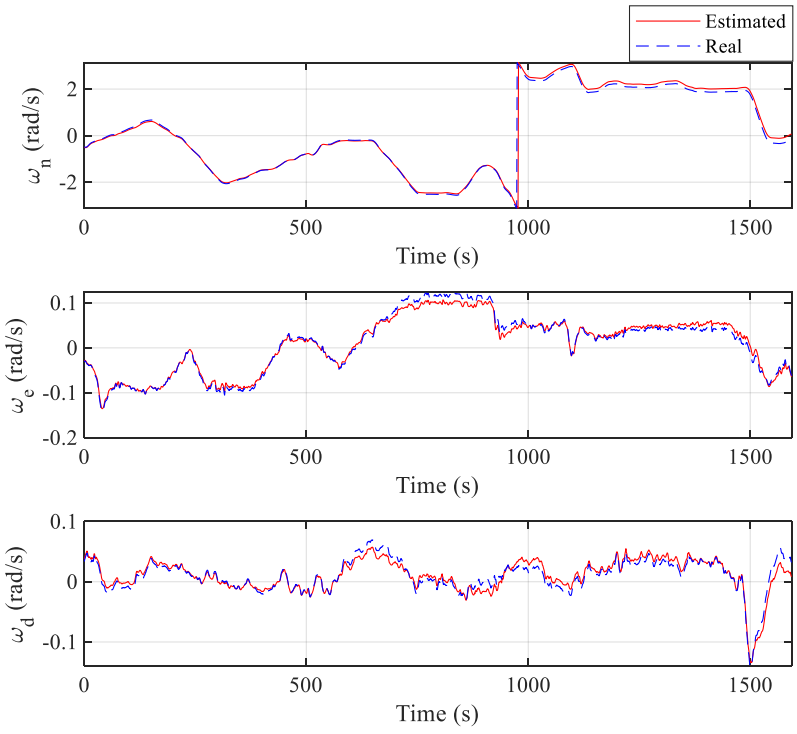

Figure 5.26. Angular velocities estimated by IMU-KITTI Sequence 09

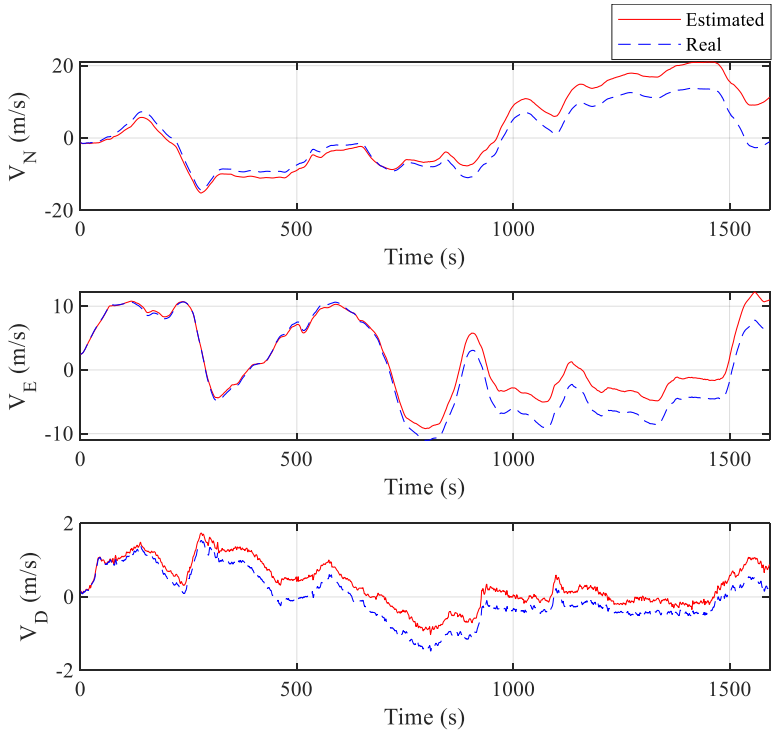

Figure 5.27. Linear velocities estimated by IMU-KITTI Sequence 09

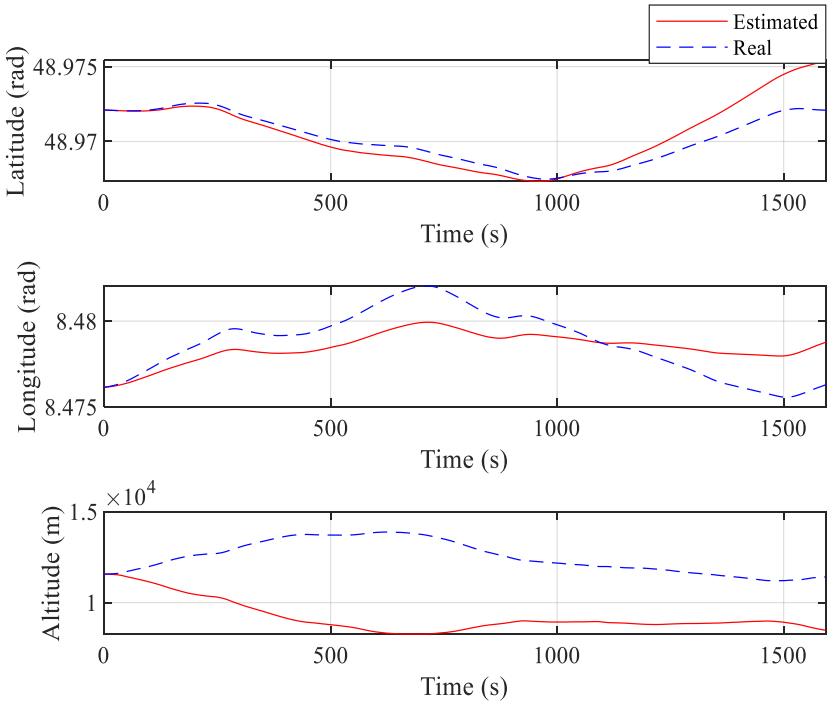

Figure 5.28. Vehicle's position estimated by IMU-KITTI Sequence 09

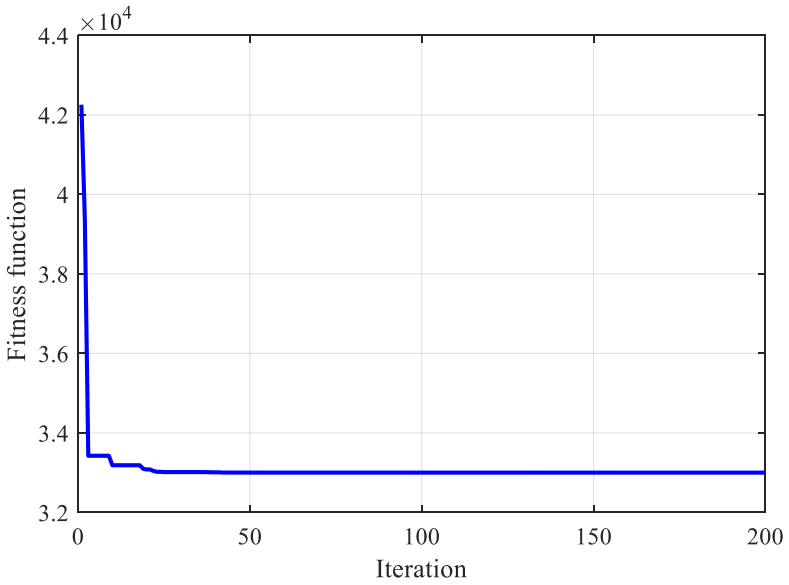

Figure 5.29. Calculated fitness function by PSO algorithm in optimized Kalman Filter-KITTI Sequence 09

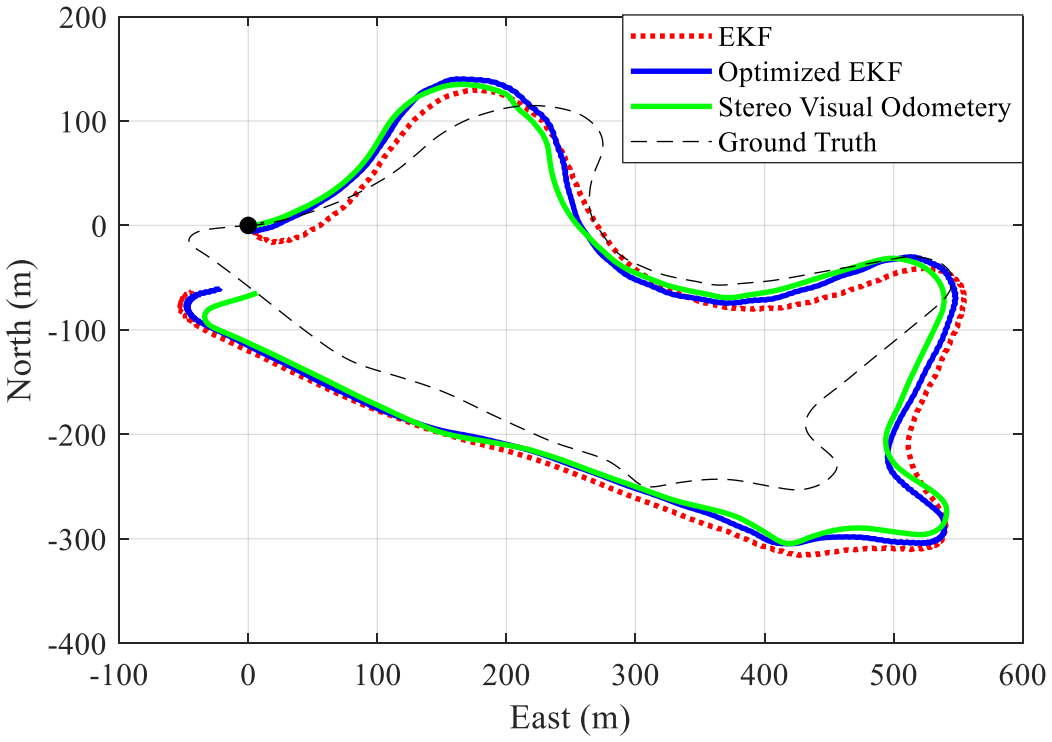

Figure 5.30. Estimated trajectory by common and optimized Kalman Filters-KITTI Sequence 09

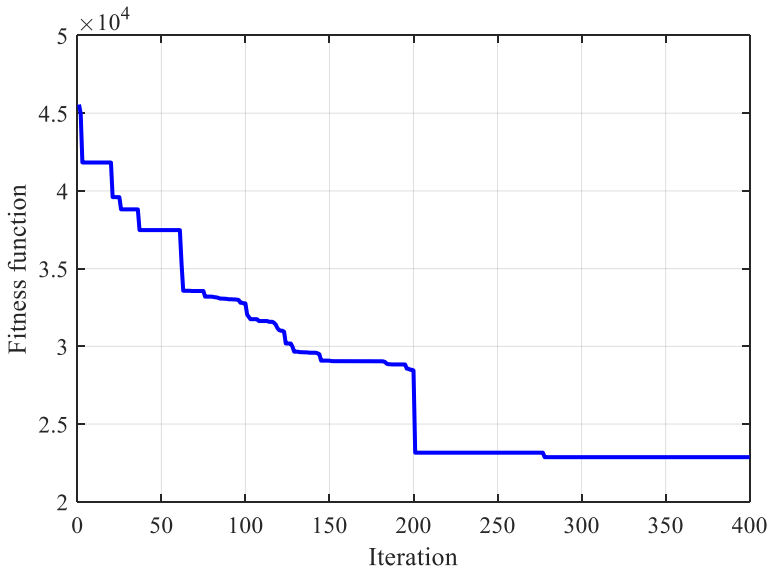

Figure 5.31. Calculated fitness function by PSO algorithm for adaptive Kalman Filter-KITTI Sequence 09

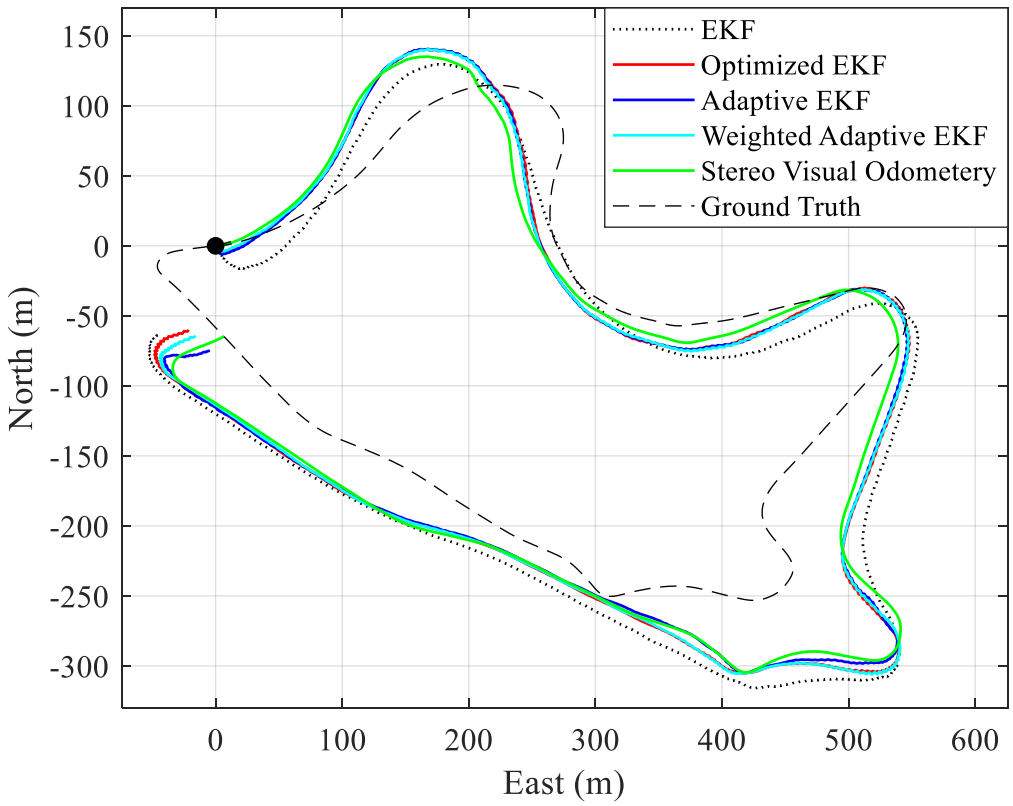

Figure 5.32. Estimated trajectory by optimized and adaptive Kalman Filters-KITTI Sequence 09

5.4.5 KITTI Dataset−Sequence 10 Results

In sequence 10, the vehicle similarly drives through a hybrid environment in a forest area and suburban residential area. By comparing the values of the fitness functions of adaptive and optimized Kalman Filters presented in Table 9, it is observed that the adaptive Kalman Filter shows to be roughly 20% more accurate than the optimized Kalman Filter. Furthermore, comparing the fitness function for the adaptive weighted EKF and the proposed adaptive Kalman Filter, it is observed that the adaptive Kalman filter is 10% more accurate. Table 9 shows the fitness functions obtained by both algorithms and Table 10 shows the position and velocity RMSE obtained using ground truth data. A summary of the results is depicted in Figures 5.33 through 5.40.

Figure 5.33 depicts the trajectory obtained by the stereo visual odometry and the trajectory obtained from IMU for KITTI Sequence 10. In addition, Figures 5.34 through 5.36 shows the IMU measurements in the time domain. In Figures 5.37 and 5.39, the convergence characteristic of the fitness functions for the optimized Kalman Filter and the adaptive Kaman Filter are shown, respectively. As shown in Figure 5.37, after roughly 25 iterations, the fitness function converges, and in Figure 5.39, after roughly 400 iterations, the fitness function converges. Although, as shown in Figure 5.39, the fitness function value is still dropping, after roughly 375 iterations, the rate of changes is very small. Figures 5.38 and 5.40 show the estimated trajectories obtained by the discussed methods which are conventional Kalman Filter, optimized Kalman Filter, weighted EKF and our adaptive Kalman Filter method. Although a precise comparison of these methods' accuracy can be made by studying Table 9 and Table 10, the robustness of the proposed adaptive Kalman

Filter can be seen. In addition, it can be seen how the errors are reduced and how it has less fluctuation around the estimated trajectory by stereo visual odometry.

Table 5.9. KITTI Sequence 10 Results

Case	Fitness function
Conventional Kalman Filter	7967
Optimized Kalman Filter by PSO	6334
Adaptive Weighted EKF	5791
Adaptive Kalman Filter	5300

Table 5.10. KITTI Sequence 10 Results

Case	Position RMSE	Velocity RMSE
Conventional Kalman Filter	61.73 m	0.078 m/s
Optimized Kalman Filter by PSO	56.10 m	0.070 m/s
Adaptive Weighted EKF	50.02 m	0.063 m/s
Adaptive Kalman Filter	47.69 m	0.060 m/s

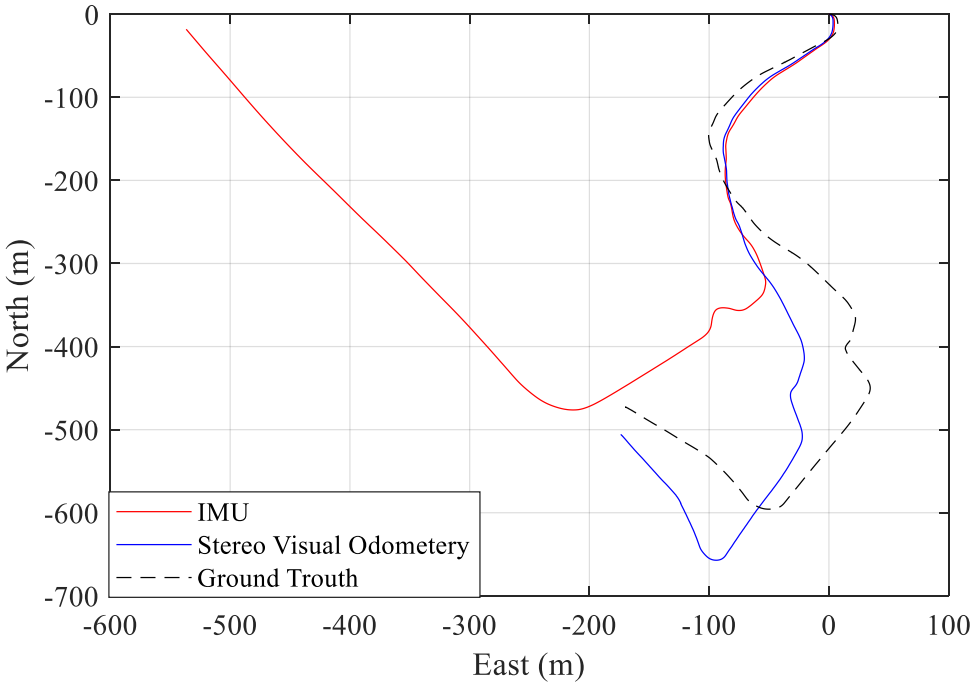

Figure 5.33. VO and IMU estimated trajectory before sensor fusion-KITTI Sequence 10

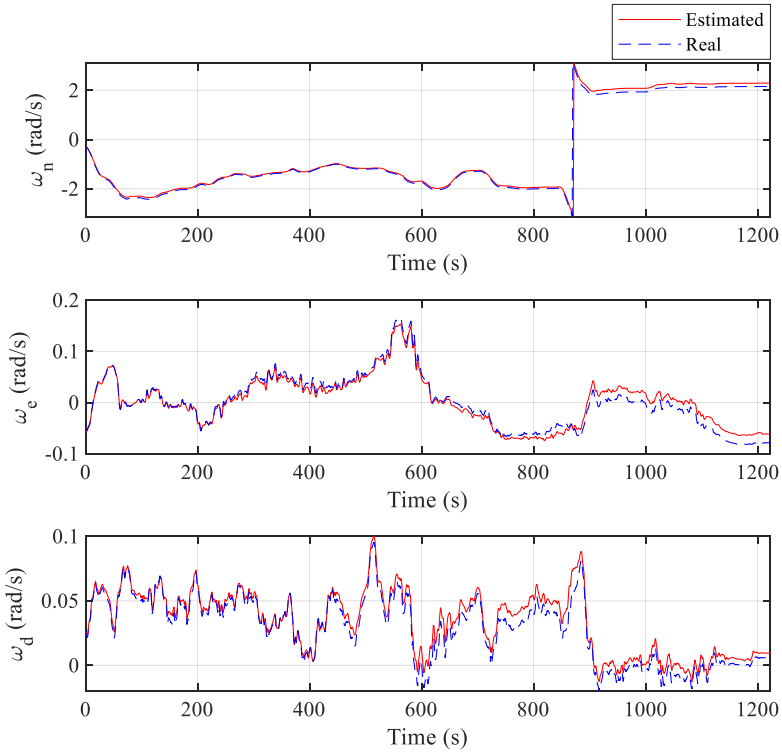

Figure 5.34. Angular velocities estimated by IMU-KITTI Sequence 10

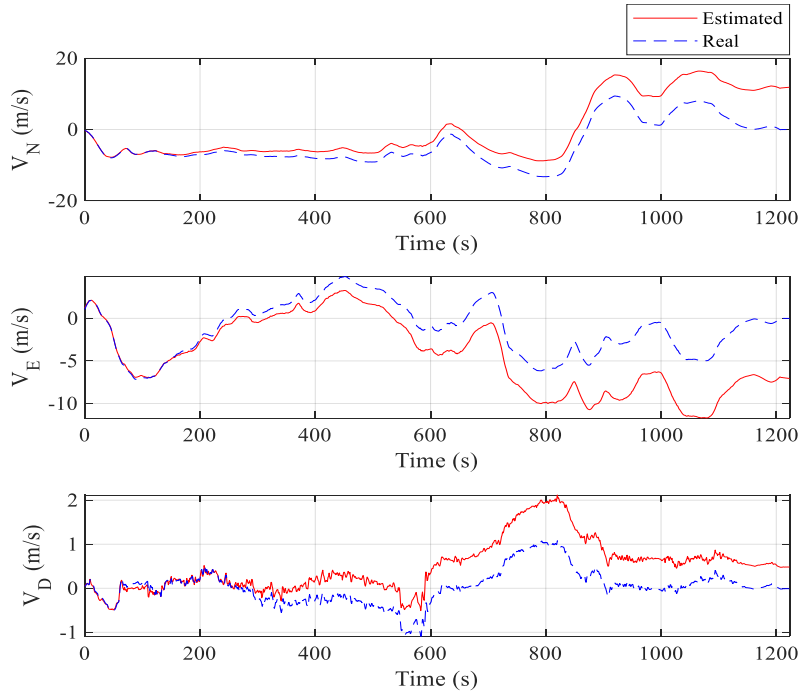

Figure 5.35. Linear velocities estimated by IMU-KITTI Sequence 10

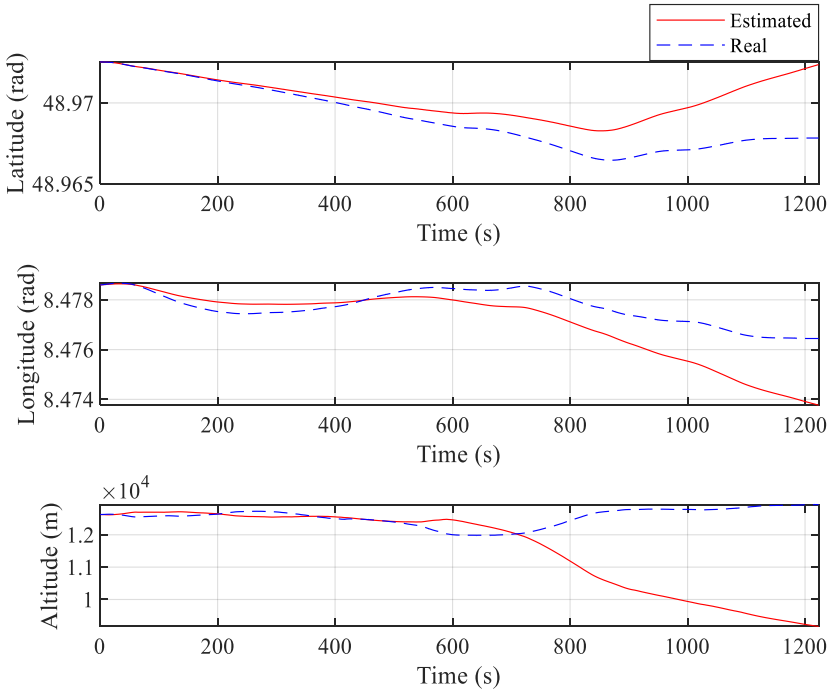

Figure 5.36. Vehicle's position estimated by IMU-KITTI Sequence 10

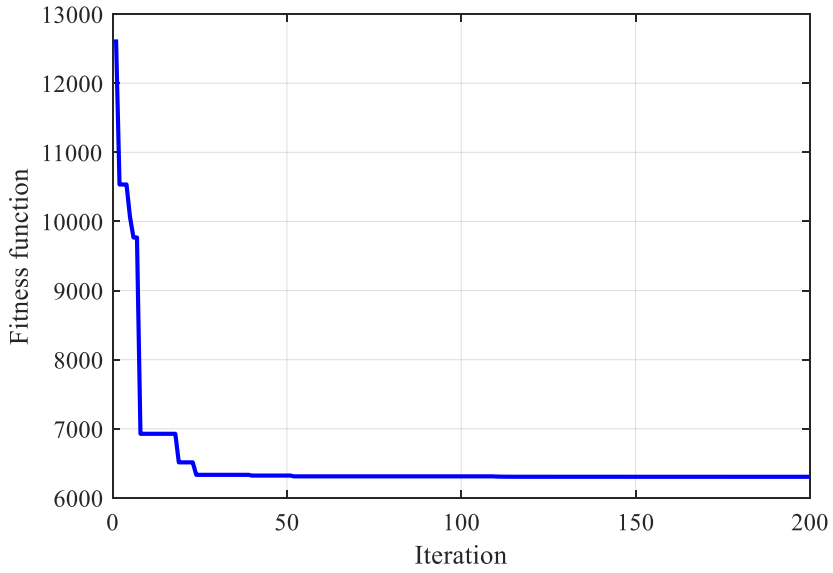

Figure 5.37. Calculated fitness function by PSO algorithm in optimized Kalman Filter-KITTI Sequence 10

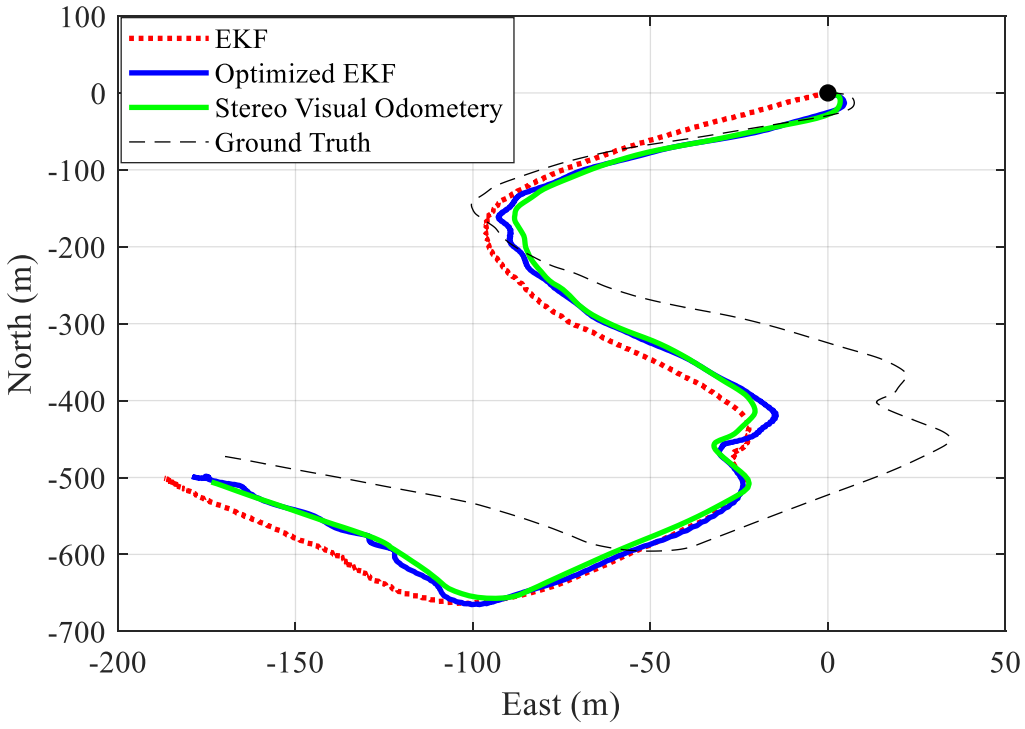

Figure 5.38. Estimated trajectory by common and optimized Kalman Filters-KITTI Sequence 10

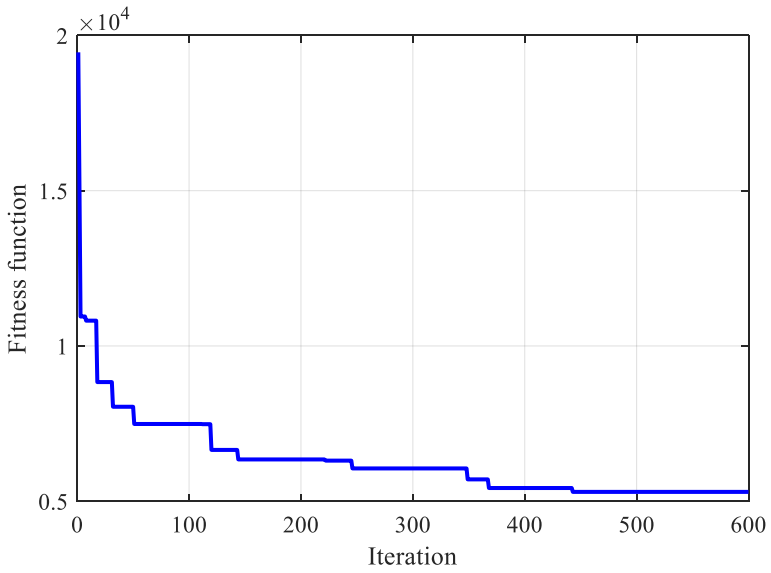

Figure 5.39. Calculated fitness function by PSO algorithm for adaptive Kalman Filter-KITTI Sequence 10

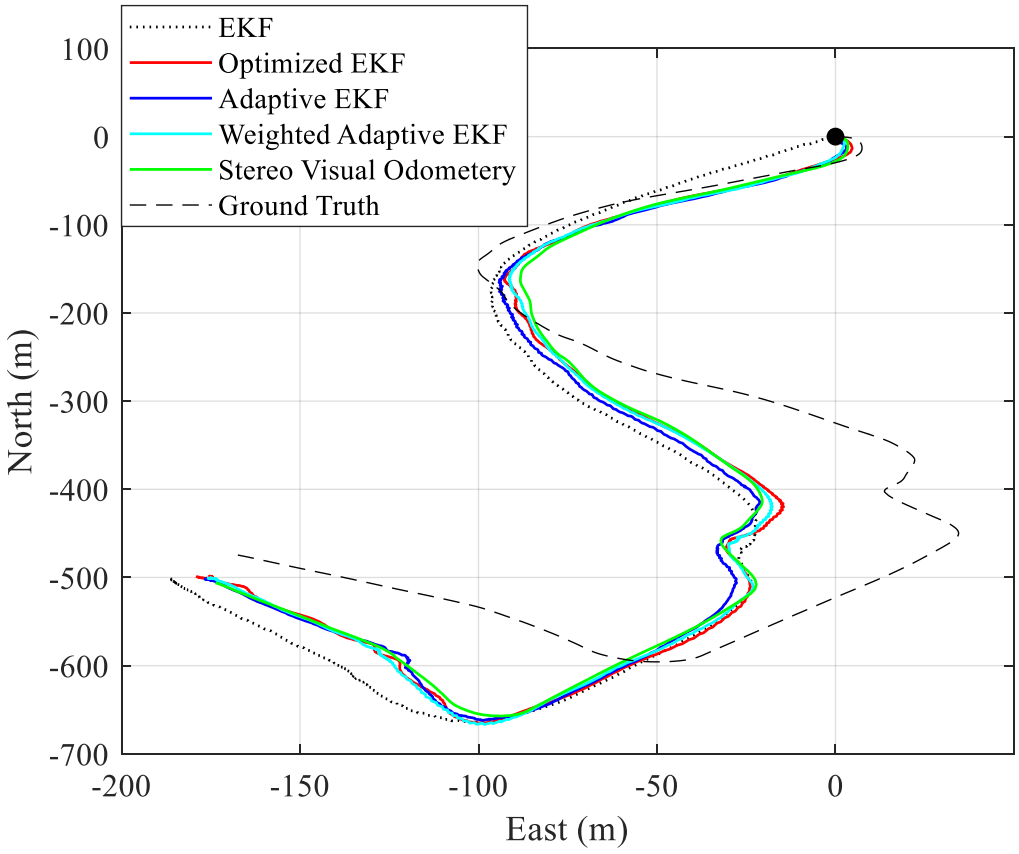

Figure 5.40. Estimated trajectory by optimized and adaptive Kalman Filters-KITTI Sequence 10

5.5 Discussion

A few considerations regarding the results presented in this chapter need to be addressed. Firstly, it should be mentioned that the main difference between our proposed fuzzy adaptive method and Sasiadek et al. [27] lies in the tuning method. In our proposed method, the PSO algorithm was used to optimize the fuzzy output parameters so that the Kalman Filter can use optimal parameters for matrices R and Q in each time step, while in Sasiadek's method, they make use of conventional methods to tune the Kalman Filter. Hence that method's accuracy depends on how well the researcher has tuned those parameters. Furthermore, the introduced method by Sasiadek et al. uses an exponential data weighting method that prevents divergence for large k [47]. This method prevents the Kalman Gain from going to zero by applying an exponential weighting in the process noise covariance matrix Q and measurement covariance matrix R.

Secondly, it is worth mentioning that due to the dependency of Sasiadek's method on the ability of the researcher to tune the Kalman Filter parameters, the reported accuracy improvements of our proposed method in comparison to sasiadek's and other methods in the next chapter are specified to this work and the scenarios in which we tuned the Filters.

Thirdly, it was observed that the estimated trajectory by IMU in two cases starts to drift too much. It was observed that the IMU data available in the KITTI benchmark has a low frequency of data collection and is not synced with the images; in some cases, there are some gaps in the data, or the IMU data is largely out-of-order. So, before using the provided data, we need to try to synchronize IMU data and image sequences. In addition, we need to interpolate some parts of missing data or eliminate the out-of-order data, and despite our

best effort, there might be some errors left in those data. These factors can lead to significant errors. In addition, it was observed that other researchers [51]–[54] faced the same problem.

Lastly, a comparison between the accuracy of the mentioned methods for sensor fusion can be made using Tables 2, 4, 6, 8, and 10. However, a more accurate comparison between these methods can be made using Tables 1, 3, 5, 7, and 9. Since we used the defined fitness function to optimize the parameters of the matrices R and Q, the resulted fitness function presented in these tables can show more accurately how our proposed methods produce more accurate results.

Chapter 6

Future work and Conclusions

6.1 Future work

For future work, we propose that the covariance matrices, R, and Q, will be diagonal matrices with multiple values for diagonal elements is an interesting study for future research. In such a structure, each state will have its own covariance values. Furthermore, in this thesis, each output parameter corresponds to a particular sequence. In other words, different driving situations change the output parameters. A particular output parameter cannot be applied to other sequences. In future work, we propose finding a single output parameter that can handle a variety of situations. For such a study, a fuzzy system needs to have more information about the operation region; for instance, the fuzzy controller must know the operation category (e.g., driving on a highway, residential area, or an off-road trail).

6.2 Conclusion

In this thesis, an adaptive sensor fusion problem was considered to be solved by utilizing an Extended Kalman Filter (EKF) and Fuzzy Logic Controller (FLC). The KITTI benchmark was used as our study's simulation and evaluation platform. In this thesis, we strived to establish a framework to adapt the EKF parameters concerning the performance characteristic. An FLC was used as an expert system that changes the EKF parameters. The fuzzy system is trained via the Particle Swarm Optimization (PSO) algorithm. The goal was to minimize the error of the estimated trajectory and bring it closer to the Visual Odometry (VO) algorithm in the intervals during which no new data is coming from cameras. The results showed that the optimized and tuned EKF, via PSO, performs 20% to 25% better than conventionally tuned EKF.

Furthermore, the adaptive Kalman Filter performs 15% to 44% better than the optimized Kalman Filter. Finally, if compared, it is observed that the introduced adaptive Kalman Filter performs 41% to 79% better than the conventional EKF method. Gessesse has reached 40% to 60% better accuracy using the Genetic Algorithm (GA) to tune their Kalman Filter. However, their method is not adaptive; hence, the optimized parameters they have obtained cannot be used or cannot perform as well in all situations. The Weighted EKF method introduced by Sasiadek et al. has reached 29% to 57% better accuracy than conventional EKF. Although they have developed a fuzzy adaptive weighted Kalman Filtering method, which, unlike Gessesse, is able to adapt the filter parameters in different situations, it relies on being tuned perfectly beforehand. The optimal tuning is not easily feasible by conventional methods. On the other hand, our proposed method benefits from both being

well-tuned and being adaptive; thus, it outperforms those two methods, and this can be observed by comparing our results and Gessesse and Sasiadek et al. However, it is worth mentioning that due to the dependency of Sasiadek's method on the ability of the researcher to tune the Kalman Filter parameters, the reported accuracy improvements of our proposed method in comparison to sasiadek's and other methods are specified to this work and the scenarios in which we tuned the Filters.

References

[1]. D. Pomerleau, "RALPH: Rapidly adapting lateral position handler." In Proceedings of the IEEE Intelligent Vehicles' 95. Symposium, pp. 506-511, 1995.

[2]. S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong et al. "Stanley: The robot that won the darpa grand challenge." In The DARPA Grand Challenge, pp. 1-43. Springer, Berlin, Heidelberg, 2007.

[3]. MH. Moghari, P. Abolmaesumi. "Comparing unscented and extended Kalman filter algorithms in the rigid-body point-based registration." In International Conference of the IEEE Engineering in Medicine and Biology Society, pp. 497-500, 2006.

[4]. J. Zhou, S. Knedlik, O. Loffeld. "INS/GPS tightly-coupled integration using adaptive unscented particle filter." The Journal of Navigation 63, no. 3, pp. 491-511, 2010.

[5]. A. Monjazeb. "Autonomous mobile robot positioning using unscented HybridSLAM." PhD Thesis, Carleton University, Ottawa, Canada, 2014.

[6]. L. Ekstrand, Y. Wang, N. Karpinsky, S. Zhang. "Superfast 3D profilometry with digital fringe projection and phase-shifting techniques." Handbook of 3-D Machine Vision: Optical Metrology and Imaging, 2013.

[7]. Z. Zhang, "Flexible camera calibration by viewing a plane from unknown orientations." In Proceedings of the 7th IEEE International Conference on Computer Vision, vol. 1, pp. 666-673, 1999.

[8]. Z. Zhang, "A flexible new technique for camera calibration." IEEE Transactions on Pattern Analysis and Machine Intelligence 22, no. 11, pp. 1330-1334, 2000.

[9]. M. Sonka, V. Hlavac, R. Boyle. "Image pre-processing." In Image Processing, Analysis and Machine Vision, pp. 56-111. Springer, Boston, MA, 1993.

[10]. AW. Fitzgibbon, "Simultaneous linear estimation of multiple view geometry and lens distortion." In Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001, vol. 1, pp. I-I, 2001.

[11]. CB. Duane, "Close-range camera calibration." Photogram. Eng 37, no. 8, pp.855-866 1971.

[12]. D. Nistér, O. Naroditsky, J. Bergen. "Visual odometry for ground vehicle applications." Journal of Field Robotics 23, no. 1, pp. 3-20, 2006.

[13]. D. Scaramuzza, F. Fraundorfer. "Visual odometry [tutorial]." IEEE Robotics & Automation Magazine 18, no. 4, pp. 80-92, 2011.

[14]. F. Fraundorfer, D. Scaramuzza. "Visual odometry: Part ii: Matching, robustness, optimization, and applications." IEEE Robotics & Automation Magazine 19, no. 2, pp. 78-90, 2012.

[15]. A. Howard, "Real-time stereo visual odometry for autonomous ground vehicles." In 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 3946-3952, 2008.

[16]. E. Rosten, T. Drummond. "Machine learning for high-speed corner detection." In European Conference on Computer Vision, pp. 430-443. Springer, Berlin, Heidelberg, 2006.

[17]. H. Hirschmuller, "Accurate and efficient stereo processing by semi-global matching and mutual information." In IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), vol. 2, pp. 807-814, 2005.

[18]. S. Leutenegger, M. Chli, RY. Siegwart. "BRISK: Binary robust invariant scalable keypoints." In IEEE International Conference on Computer Vision, pp. 2548-2555, 2011.

[19]. PHS. Torr, A. Zisserman. "MLESAC: A new robust estimator with application to estimating image geometry." Computer Vision and Image Understanding 78, no. 1, pp. 138-156, 2000.

[20]. RE. Burkard, E. Cela. "Linear assignment problems and extensions." In Handbook of Combinatorial Optimization, pp. 75-149. Springer, Boston, MA, 1999.

[21]. PD. Groves, "Principles of GNSS, inertial, and multisensor integrated navigation systems, " IEEE Aerospace and Electronic Systems Magazine 30, no. 2, pp. 26-27, 2015.

[22]. SI. Roumeliotis, GS. Sukhatme, GA. Bekey. "Circumventing dynamic modeling: Evaluation of the error-state Kalman filter applied to mobile robot localization." In Proceedings IEEE International Conference on Robotics and Automation (Cat. No. 99CH36288C), vol. 2, pp. 1656-1663, 1999.

[23]. GG. de Cunto, "Sensor fusion INS/GNSS based on fuzzy logic adaptive error-state Kalman filter and unscented Kalman filter." Master Thesis, Carleton University, Ottawa, Canada, 2020.

[24]. M.H.Gessesse, "Multi-sensor Attitude and Heading Reference System Design using Genetically Optimized Kalman Filter." Master Thesis, Carleton University, Ottawa, Canada, 2018.

[25]. Y. Liu, R. Xiong, Y. Wang, H. Huang, X. Xie, X. Liu, G. Zhang. "Stereo visual-inertial odometry with multiple Kalman filters ensemble." IEEE Transactions on Industrial Electronics 63, no. 10, pp. 6205-6216, 2016.

[26]. S. Yazdkhasti, J.Z. Sasiadek, S. Ulrich. "Performance enhancement for GPS/INS fusion by using a fuzzy adaptive unscented Kalman filter." In 21st IEEE International Conference on Methods and Models in Automation and Robotics (MMAR), pp. 1194-1199, 2016.

[27]. J. Z. Sasiadek, Q. Wang, MB. Zeremba, "Fuzzy adaptive Kalman filtering for INS/GPS data fusion." In Proceedings of the IEEE International Symposium on Intelligent Control. Held jointly with the 8th IEEE Mediterranean Conference on Control and Automation (Cat. No. 00CH37147), pp. 181-186, 2000.

[28]. J. Engel, T. Schöps, D. Cremers. "LSD-SLAM: Large-scale direct monocular SLAM." In European Conference on Computer Vision, pp. 834-849. Springer, Cham, 2014.

[29]. T. Gonçalves, AI. Comport. "Real-time direct tracking of color images in the presence of illumination variation." In IEEE International Conference on Robotics and Automation, pp. 4417-4422, 2011.

[30]. A. Concha, J. Civera. "DPPTAM: Dense piecewise planar tracking and mapping from a monocular sequence." In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 5686-5693, 2015.

[31]. J. Engel, V. Koltun, D. Cremers. "Direct sparse odometry." IEEE Transactions on Pattern Analysis and Machine Intelligence 40, no. 3, pp.611-625, 2017.

[32]. C. Forster, M. Pizzoli, D. Scaramuzza. "SVO: Fast semi-direct monocular visual odometry." In IEEE International Conference on Robotics and Automation (ICRA), pp. 15-22, 2014.

[33]. N. Krombach, D. Droeschel, S. Behnke. "Combining feature-based and direct methods for semi-dense real-time stereo visual odometry." In International Conference on Intelligent Autonomous Systems, pp. 855-868. Springer, Cham, 2016.

[34]. S. Sirtkaya, B. Seymen, A.A. Alatan. "Loosely coupled Kalman filtering for fusion of Visual Odometry and inertial navigation." In Proceedings of the 16th IEEE International Conference on Information Fusion, pp. 219-226, 2013.

[35]. S. Weiss, R. Siegwart. "Real-time metric state estimation for modular vision-inertial systems." In IEEE International Conference on Robotics and Automation, pp. 4531-4537, 2011.

[36]. J. Kelly, GS. Sukhatme. "Visual-inertial sensor fusion: Localization, mapping and sensor-to-sensor self-calibration." The International Journal of Robotics Research 30, no. 1, pp.56-79, 2011.

[37]. S. Lynen, M.W. Achtelik, S. Weiss, M. Chli, R. Siegwart. "A robust and modular multi-sensor fusion approach applied to mav navigation." In IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 3923-3929, 2013.

[38]. S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, P. Furgale. "Keyframe-based visual–inertial odometry using nonlinear optimization." The International Journal of Robotics Research 34, no. 3, pp. 314-334, 2015.

[39]. A. Concha, G. Loianno, V. Kumar, J. Civera. "Visual-inertial direct SLAM." In IEEE International Conference on Robotics and Automation (ICRA), pp. 1331-1338, 2016.

[40]. V. Usenko, J. Engel, J. Stückler, D. Cremers. "Direct visual-inertial odometry with stereo cameras." In IEEE International Conference on Robotics and Automation (ICRA), pp. 1885-1892, 2016.

[41]. A. Geiger, P. Lenz, C. Stiller, R. Urtasun. "The KITTI vision benchmark suite." URL http://www. cvlibs. net/datasets/kitti 2, 2015.

[42]. M. Kordestani, M. Dehghani, B. Moshiri, M. Saif. "A new fusion estimation method for multi-rate multi-sensor systems with missing measurements." IEEE Access 8, pp. 47522-47532, 2020.

[43]. R. Cristi, M. Tummala. "Multirate, multiresolution, recursive Kalman filter." Signal Processing 80, no. 9, pp. 1945-1958, 2000.

[44]. PP. Angelov, D. P. Filev. "An approach to online identification of Takagi-Sugeno fuzzy models." IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 34, no. 1, pp. 484-498, 2004.

[45]. R. Eberhat, J. Kennedy. "A new optimizer using particle swarm theory." In 6th International Symposium on Micro Machine and Human Science, Piscataway, pp. 39-43, 1995.

[46]. S. N. Givigi, H. M. Schwartz, X. Lu. "A reinforcement learning adaptive fuzzy controller for differential games." Journal of Intelligent and Robotic Systems 59, no. 1, pp. 3-30, 2010.

[47]. J. Z. Sasiadek, Q. Wang. "Low-cost automation using INS/GPS data fusion for accurate positioning." Robotica 21, no. 3, pp. 255-260, 2003.

[48]. “Camera Calibration and 3D Reconstruction — OpenCV 2.4.13.7 Documentation,”docs.opencv.org,2019.[Online].Available:https://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reco nstruction.html. [Accessed: 07-Mar-2019].

[49]. EF. Aguilar Calzadillas, "Sparse Stereo Visual Odometry with Local Non-Linear Least-Squares Optimization for Navigation of Autonomous Vehicles.", Master Thesis, Carleton University, Ottawa, Canada, December 2019.

[50]. EH. Mamdani, S. Assilian. "An experiment in linguistic synthesis with a fuzzy logic controller." International Journal of Man-Machine Studies 7, no. 1, pp. 1-13, 1975.

[51]. M. Brossard, A. Barrau, S. Bonnabel, "AI-IMU dead-reckoning. " ArXiv Preprint ArXiv:1904.06064, 2019.

[52]. P. Aggarwal, "MEMS-based integrated navigation. " Artech House, 2010.

[53]. E.S. Naser, H. Haiying and N. Xiaoji, "Analysis and Modeling of Inertial Sensors Using Allan Variance," IEEE Transactions on Instrumentation and Measurement, vol. 57, no. 1, pp. 140-149, 2008.

[54]. C. Huang, Y. Jiang & K. O'Keefe, "Wheel Odometry aided Visual-Inertial Odometry for Land Vehicle Navigation in Winter Urban Environments, " In Proceedings of the 33rd International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS+ 2020) (pp. 2237-2251), 2020.