Vision Augmented State Estimation with Fault Tolerance

by

Rahul Chandail

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2018

Abstract

Obtaining accurate and reliable measurement data is really crucial for any vehicle system, especially if the system deals with maintaining safe operations of the vehicle. Conventional direct methods of obtaining such measurements by using sensors like GPS, INS, and Wheel Encoders are not reliable. The raw measurement obtained from these sensors is accompanied with bias and noise. Also, these sensors tend to underperform in certain conditions and faults, which adds to the inaccuracy and uncertainty of a system. Many different approaches exist, which intelligently fuse information from multiple sensors in order to produce reliable estimates. However, when used independently, these approaches suffer due to faults like plausibility faults, frequency faults, and error code faults, which can occur at the sensor level.

This thesis therefore takes into account the development and implementation of an architecture which merges a planned fault tolerance approach to an intelligent state estimation process. The main contribution of the developed prototype is that it demonstrates the effectiveness of the developed architecture in mitigation of various faults that occur at the sensor level. This prototype lays the ground for the use of VO as an alternate source of measurement to the conventional sensors. Starting from the implementation of sensor data specific integrity checking to the implementation of Extended Kalman Filter, fault tolerance methods have been employed at different levels. A 3 DoF dynamic vehicle model has been used to enable the Extended Kalman Filter predictions. Finally, test cases were devised to validate the effectiveness of the prototype.

Acknowledgements

I would like to thank all the little people who made this thesis possible.

Dedication

This is dedicated to the ones I love.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

From the advent of the motorized vehicle to the modern-day automobile, reliability and safety while driving has been a major concern for car manufacturers and scholars.

As the modern car has emerged, many different technological advancements have been made in order to reduce vehicle function failures and improve the overall stability and safety of the vehicle. Features like Electronic Stability Control (ESC), Antilock Brakes (ABS), and Traction Control provide added measure of safety and improve the control performance of the vehicle. These safety features rely on various information about the vehicle state to function sufficiently. The information required may include detailed state variables such as independent wheel speed and higher level information such as vehicle acceleration. However, measuring these state variables is not a trivial task. Whereas state variables like steering angle or the throttle torque are readily obtainable, accurately measuring slip angle and tire forces can be quite challenging due to technology limitations and the cost involved. Installing high cost sensors on vehicles suffices for research studies, but remains uneconomical for commercial or consumer vehicles.

To deal with these challenges, automotive engineers and researchers developed various state estimation methods. These methods tend to use the information from low cost sensors that are already present on the vehicle or can be discretely installed. Examples of such sensors include Global Positioning System (GPS) receivers, inertial navigation system (INS) sensors, and cameras. Different algorithmic techniques like machine learning, sensor fusion, and vehicle model analysis together or independently, provide multiple ways of estimating the state of the system. [19] These techniques differ in the working principle, but rely essentially on the information obtained from the sensors.

Therefore, it is crucial that these methods should be able to tolerate the anomalies or faults that might occur due to various factors, including malfunctioning of the sensors, external noise, and sensor failures. The reliability of the estimation provided by any system can be attributed to its ability to deliver a service that can justifiably be trusted [5], [20]. It is important that faults occurring at the sensor level be detected and their effect be mitigated in order to obtain the desirable estimations. Therefore, depending on the need, the complexity and the cost involved, different approaches for fault tolerance are used to make the system robust to such factors.

1.1 Motivation

Safety systems in the modern car rely heavily on accurate estimation of variables that define the state of the vehicle at a given instance in time. Speaking in broad terms, these state variables are available for a vehicle in two ways, i.e. in the form of direct and indirect measurements.

Direct measurements may include longitudinal vehicle velocity, lateral vehicle velocity, and acceleration. Such information can be easily obtained directly from the conventional group of sensors like GPS and INS, which are readily available on a modern vehicle. However, information or data obtained directly from the sensors can be plagued with uncertainties like random noise, drift, jumps, etc. Using raw information measured directly from low cost and relatively low performance sensors can lead to significant and undesired inaccuracies in safety systems. As a result, the systems might actuate functions prematurely, have a delayed response, and/or an incorrect response.

Indirect measurements refer to the variables that are derived from the available direct measurements. These measurements can include variables like tire forces, slip angle, and slip ratio. This information can be derived by using model equations that relate such variables. Vehicle models, like the kinematic bicycle model, 3 Degree of Freedom (DoF) bicycle force model, and 7 DoF vehicle model play a crucial role in predicting the variables that are not easily measured directly. They govern the motion of the vehicle with the help of dynamic equations, which can be used to calculate the non-measured variables from the variables that are directly measured. However, any model in itself is inacurate. A model always makes a number of assumptions in order to simplify the complex dynamics of the vehicle. These simplifying assumptions add uncertainty and potential bias to the output of the model. Additional errors may be introduced through numerical algorithms, such as integration.

Both standalone direct and indirect measurement methods are often insufficient to achieve high accuracy and reliability of the state estimates. An alternative approach is to work with an amalgamation of the directly available measurements and the prediction capability of the vehicle model equations to measure the state variables and reduce the overall uncertainty. Many different techniques are available to obtain these state variable estimates. One such technique that uses both direct measurements and the vehicle model is the Kalman Filter (KF). KF is widely used to deal with linear systems, whereas an extension of KF, called the Extended Kalman Filter (EKF), is used to deal with non-linear systems. Depending upon different needs and applications, the KF is further extended in to Unscented Kalman Filter (UKF), FKF (Fuzzy Kalman Filter), and many other variants in order to deal with complexity. [34]

For the purpose of vehicle state estimation, researchers have used the Kalman Filter or its variants, while typically relying on upon inputs from a conventional set of sensors like the GPS, INS, and Wheel Odometry. For example, Rezaei and Sengupta [29] estimated vehicle location using EKF, which fused the information provided by the sensors GPS and INS with the predictions made by the dynamic bicycle model. Such systems perform well under normal conditions. However, the sensors involved in such systems can underperform in some scenarios, which can cause inaccuracies in the state estimation. For example, GPS loses its tracking ability when passing through a tunnel, and wheel odometry suffers from slip condition especially when cornering and on slippery roads. Therefore, an added source of information is needed to complement the conventional set of sensors.

Visual Odometry (VO) is one such source. VO can be described as the process of determining the pose, i.e. the position and orientation, of a camera body based on the analysis of the associated camera images. In the case of a car, the images can be obtained from a camera mounted behind the front windshield. VO has certain advantages over other sensors. For example, VO does not suffer from tire slip, and it works in GPS denied environments. Visual odometry has been used to augment state estimation mostly for aerial vehicle or small size robots. [40] It's application on a full-scale vehicle for state estimation is a promising direction.

Even though an EKF mediates between the measurements and the predicted values to provide a best possible estimate, it is not immune to faults that occur at the measurement side or at the process level. Therefore, a system structure that can detect and adequately respond to faults occurring at the measurement input of the EKF is needed in order to improve the performance of the overall estimation system and make it more robust.

1.2 Problem Statement

Current state estimation methods involving EKF have shortcomings, especially when dealing with underperforming sensors. These sensors can provide implausible information, provide degraded data, or provide delayed information. Also, various faults can occur while processing the data. Sensors can provide biased data or stop working under certain environmental conditions. These issues can cause anomalies in the state estimation process and even lead to highly incorrect estimates. Thus, it is important to develop a system that is robust to such issues.

1.3 Contribution

As discussed earlier, there are various techniques for state estimation. One of the widely spread technique is the Kalman Filter approach, and it's variant, the Extended Kalman Filter approach, which allows one to deal with nonlinear systems. Existing work on vehicle state estimation using the EKF has focused on conventional sensors, such as INS, GPS, and wheel odometry, which can underperform under certain conditions. This thesis research contributes a state estimation system design for motor vehicles that augments the conventional sensors with vision input and a fault detection and response components in order to improve the fault tolerance of the overall state estimation system. The proposed design has the following two novel aspects:

- VO augmented state estimation: Using VO to complement the conventional sensors. Since VO is an added sensor, it can either replace or overlap some of the information that is obtained from the conventional sensors.
- Fault tolerance: Developing a fault tolerant stream for data throughout the overall state measurement and estimation system that detects and mitigates faults at an early stage. Since, faults can occur both at sensor level or at any time during the estimation process, the purpose of the fault-tolerant design is to moderate the impact of said faults on the systems estimates.

The thesis also presents a prototype implementation of the proposed design and its evaluation in simulation. The evaluation considers fault-injection scenarios and analyzes the improvement of the state estimate in these scenarios due to the proposed fault tolerant design.

1.4 Thesis Organization

The remainder of the thesis has the following chapters.

Chapter 2 details the necessary background information on state estimation and related topics like visual odometry, vehicle modeling, and Sensors Fusion.

Chapter 3 presents a summary of the related research work in the field of state estimation and fault tolerance. It gives a brief overview of the different methodologies that are being used to achieve state estimation with or without fault tolerance.

Chapter 4 provides a detailed overview of the components involved in the design proposed by the thesis. It describes the exploration of the components and their final configuration, overall structure of the design, its components, and the information flow through the structure. Finally, it describes the test setup used to perform the analysis under different fault conditions.

Chapter 5 displays the results obtained from the test setup. It analyzes the obtained data and discusses the contributions of the different parts of the architecture on the resulting estimates.

Chapter 6 summarizes the overall results and proposes future work.

Chapter 2

Background

The chapter provides the necessary background knowledge.

2.1 Basics

2.1.1 State

The state of a system is a vector of system variables that entirely defines the relevant aspects of the system at a specific instance in time. For example, for the sake of localization at a given instance of time, a vehicle can minimally be defined by its x, y, and z coordinates (state variable) in a three dimensional reference frame. Another example can be that of a temperature control system where state variables like inside temperature, humidity, and external temperature form the state.

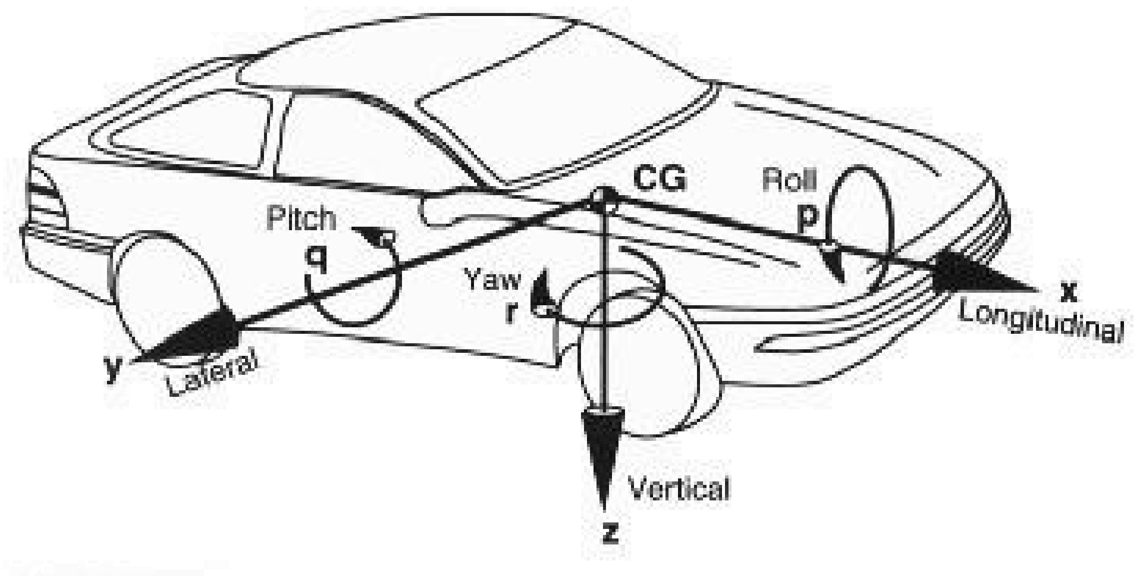


Figure 2.1: Different States of a Vehicle [9]

Similarly in a given reference frame, the state of a vehicle can be described by parameters like position, velocity, and orientation. A number of onboard active and passive safety technologies like traction and stability control require information of the vehicle's different states. For example, effective knowledge of the slip angle and tire forces can help prevent slips during unfavourable driving conditions. Or for a system like Front Collision Control, the knowledge of the vehicle speed and acceleration is of utmost importance.

2.1.2 State Estimation

State estimation can be described as the process of obtaining the accurate state of a body or a system from the available information and input. Ideally, the state of a vehicle can be directly measured by putting up different sensors onto the vehicle. However, due to physical, economical, and technological limitations, only a few of these states can actually be obtained by the means of direct sensor measurement. Even the states that are measured with the help of sensors have the tendency to be plagued with inconsistency and noise.

There are many methods that are used for state estimation. For example, Kalman Filter (and its derivatives), Particle filters, and Maximum Likelihood estimators. One of the most widely used estimators are Kalman filters, which will be discussed further in Section 2.4

2.1.3 State Vector

For the purpose of estimation process, a state vector carries all the state variables that are defined in the state.

$$StateVector = \begin{bmatrix} x \\ y \\ \psi \\ \vdots \\ etc. \end{bmatrix} = \begin{bmatrix} x\ position \\ y\ position \\ yaw\ angle \\ \vdots \\ etc. \end{bmatrix} \tag{2.1}$$

2.2 Visual Odometry

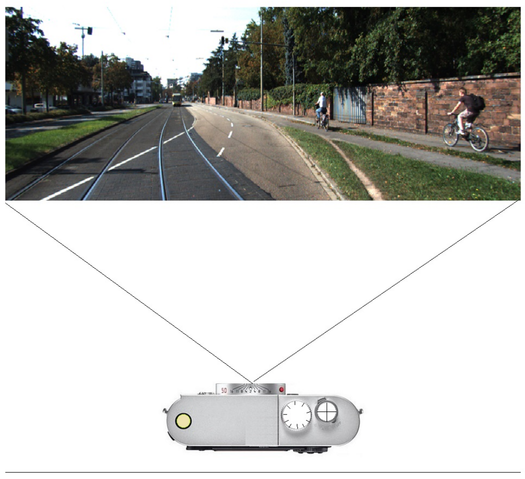
2.2.1 Overview

Visual Odometry (VO) is the process of estimating the egomotion of an agent (e.g., vehicle, human, and robot) using input from a single camera or multiple cameras attached to it. [30] Visual odometry relies on the input camera images to be sufficiently feature rich. The scenes need to have enough static features so that the relative motion of the vehicle (camera) can be extracted accurately. The term Visual Odometry was first coined by Nister et al. in 2004, where they presented a system that estimated the motion of a stereo head camera based on the video input. [26]

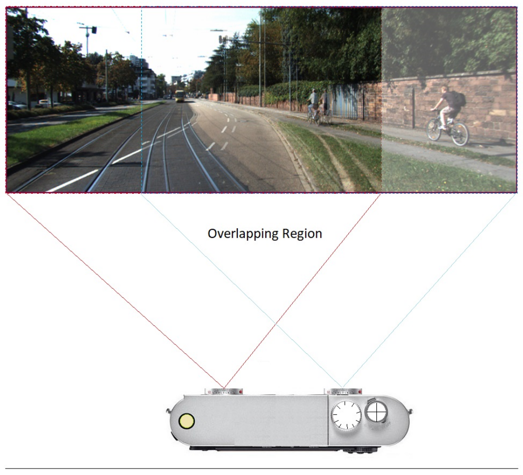
Visual Odometry has certain advantages over the conventional group of sensors. For example, the wheel odometer provides the motion of the vehicle by integrating the number of turns the wheel took over time. However, wheel odometery suffers from wheel slip on uneven or slippery surfaces and during braking or acceleration, and during cornering. Similarly, laser odometry suffers from timing and synchronization problems. In GPS-denied environments such as tunnels, GPS suffers from satellite signal issues and tends to stop functioning. VO is immune to these issues and acts as a good supplement for them. However, VO degrades under other conditions such as when not enough static features are visible due to obstructions or darkness.

2.2.2 VO Types

VO can be divided based on the type of camera. Superficially, if a monocular camera is used as an image source, then the odometry is called as Monocular Visual Odometry. And if the input image is a dual stream of images provided by a stereo camera, it is processed as Stereo Visual Odometry. Both monocular VO and stereo VO follow similar principle for egomotion estimation, but differ in terms of how factors like scale information, depth calculation, and image processing are handled. For the purpose of this thesis, due to several constraints, a monocular camera and hence monocular VO has been used.



(a) Monocular Camera Image



(b) Stereo Camera Image

Figure 2.2: Monocular Vision and Stereo Vision.

2.2.3 Generic VO Methodology

VO computes the path of the camera or agent incrementally, i.e. pose after pose. The first step is to obtain a sequence of images. In the case of a car, it is achieved with the help of a front facing camera mounted behind the front windshield.

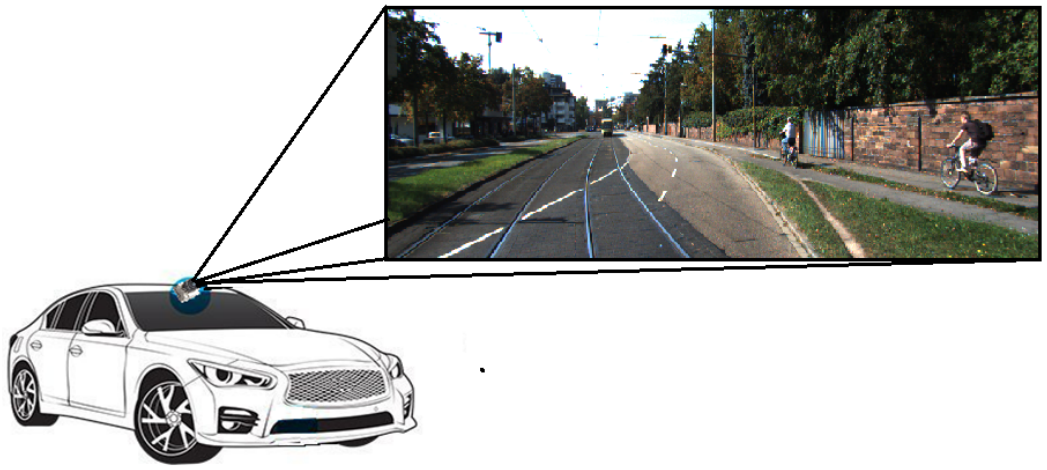


Figure 2.3: Front Facing Camera Mounted on Top of the Car.

The next step is to identify and extract certain features from the images like corners, edges, and blobs. For this purpose, a number of different techniques and their combinations can be used, like Sobel edge detector, Harris corner detector, and Hough transform. Figure 2.4 provides an example of the detected features.

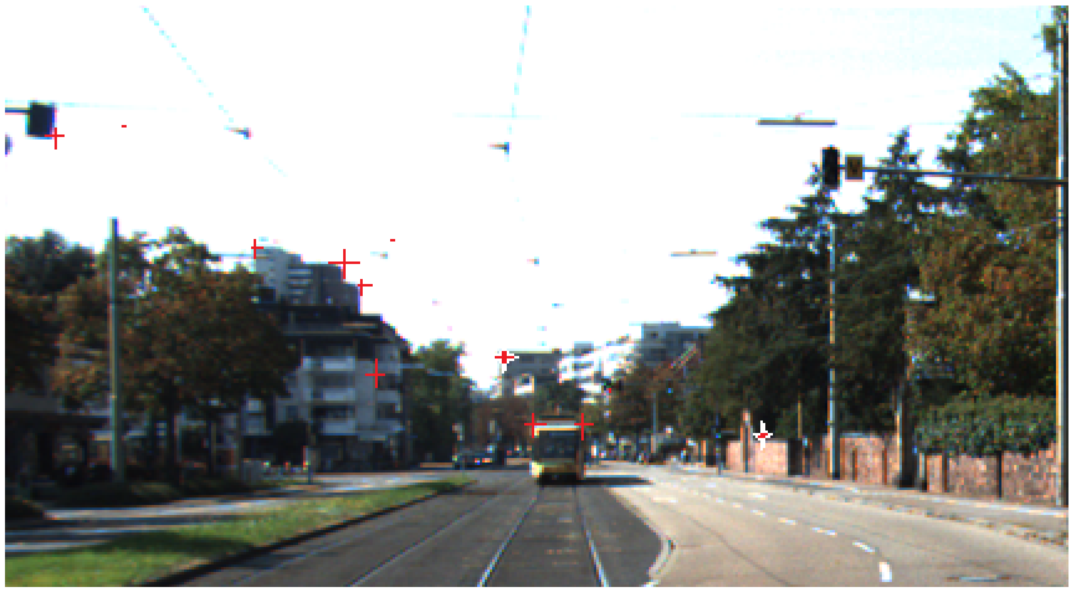


Figure 2.4: Detecting Features.

Once feature extraction process is complete, these features are tracked from one image frame to the next one. This is called as feature matching. Figure 2.5 shows the features being tracked from one frame to the next frame.

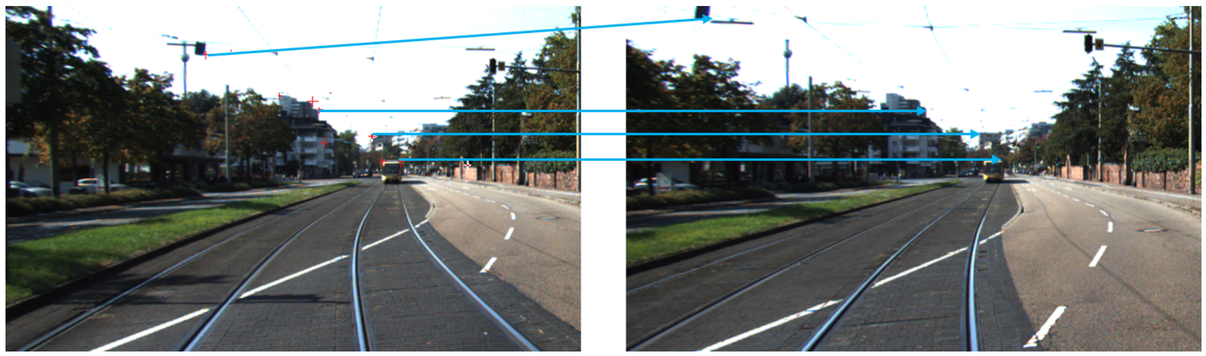


Figure 2.5: Tracking Features in Consecutive Image Frames.

Finally based on the motion of these features relative to the frame of the camera, the motion of the camera or the agent is estimated. Some of the approaches for VO are presented in [8], [16], and [13]

2.3 Vehicle Model

A vehicle model can be explained as a set of constraints that describe the state evolution of the vehicle. The constraints restrict the motion of a mechanical system. Such constraints can be represented with the help of equations, which can be formed out of both the holonomic and nonholonomic constraints of a vehicle system. [38] For the purpose of state estimation via. EKF, a vehicle model is of utmost importance. It enables the estimator to make predictions about the state that it tries to estimate.

A vehicle model needs to be adequate enough to be able to model a 4-wheel car with a certain degree of accuracy. It can be as simple as a 2-wheel bicycle kinematic model or as complex as a 4-wheel 14 DoF dynamic model [32]. The Figure 2.6 below shows a simple 2-wheel bicycle kinematic model.

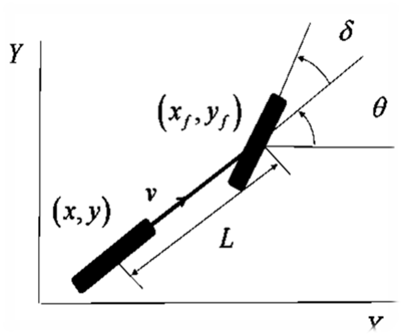


Figure 2.6: 2-Wheel Bicycle Kinematic Model.

2.4 Kalman Filter and the Extended Kalman Filter

A Kalman Filter is an optimal estimator, i.e. it infers parameters of interest from indirect, inaccurate and uncertain observations. [21] It has a recursive nature, which means that at every iteration new measurements are processed as they are received. The filter is considered optimal in the sense that if all the noise is Gaussian, the filter minimizes the mean square error of the estimated parameters. [21]

The following concepts are involved in a Kalman Filter:

- State Vector: Described in Section 2.1.3
- Vehicle Model: Described in Section 2.3
- Process Error Covariance (P): The process error covariance can be described as trust matrix. It describes the trust in state variables at every iteration step. It is represented by the character symbol P. If it is known that the initial state is accurate, then the P matrix is initialized with low values on its diagonal. If there is no trust in the initialized state, then the matrix is initialized with high values on its diagonal.
- Process Noise Covariance (Q): This covariance matrix depicts the uncertainty present in the prediction process. It is represented by the character symbol Q. If the vehicle model chosen for the prediction is adept and there is high confidence in its output, the matrix values on its diagonal are set low. If the confidence is low, these matrix values are set high.
- Measurement Noise Covariance (R): This covariance matrix depicts the uncertainty present in the sensor measurements. It is represented by the character symbol R. If the confidence

in sensor measurements is high, the matrix is initialized with low values on its diagonal. If the confidence is low, these matrix values are set high.

- Kalman Gain (K): The Kalman gain is the most critical parameter of the Kalman Filter. The gain value is used to update the state predictions. If the calculated gain value is high, it represents that the filter trusts the update measurements more than the state prediction. Thus, the predicted state is allowed to be influenced by the measurement update. The measurement update is directly proportional to the calculated gain value. If the gain value is low, it represents that the state prediction confidence is much higher than the measurement trust. Thus, the measurement update has very low influence on the predicted state.

- Jacobian Matrix: Jacobian matrix is the linearization of the nonlinear system of equations that form the vehicle model. It is the matrix of all first order partial derivatives of a nonlinear function. Jacobian matrix defines a linear map i.e. a linear approximation of a function near a certain point.

The following subsections provide a brief overview of the various steps involved in the functioning of a Kalman Filter.

2.4.1 Kalman Filter

Linear System

A linear system can be given by:

$$x_{t+1} = Ax_t + Bu_{t+1} + w_{t+1} \tag{2.2}$$
$$y_{t+1} = Cx_t + v_{t+1} \tag{2.3}$$

where $x_{t+1} \in \mathbb{R}^n$ is state, $y_{t+1} \in \mathbb{R}^m$ is measurement, u_{t+1} is the control input, t is the time step, w_{t+1} and v_{t+1} are the process noise and measurement noise respectively. The noise is usually modelled as additive Gaussian noise.

Kalman Filter functions as a two-step process. The first step is the prediction step, which is followed by the correction step.

Before the beginning of the Kalman process the x_0 state vector and the P, Q, R matrices are initialized as per the initial conditions.

Prediction:

Prediction step calculates two parameters: State vector prediction $\bar{x}_{t+1}$ and the process covariance prediction $\bar{P}_{t+1}$.

$$\bar{x}_{t+1} = Ax_t + Bu_t + w_t \tag{2.4}$$

$$\bar{P}_{t+1} = AP_tA^T + Q_{t+1} \tag{2.5}$$

Correction:

Correction step calculates the Kalman gain K_{t+1}. Based on the Kalman gain, the predicted state vector and the process error covariance goes through the needed corrections as shown in the equations below:

$$K_{t+1} = \frac{\bar{P}_{t+1}C^T}{C\bar{P}_{t+1}C^T + R_{t+1}} \tag{2.6}$$

$$x_{t+1} = \bar{x}_{t+1} + K_{t+1}(y_{t+1} - C\bar{x}_{t+1}) \tag{2.7}$$

$$P_{t+1} = (I - K_{t+1}C)\bar{P}_{t+1} \tag{2.8}$$

However, a KF comes with a limitation. A Kalman Filter works only for linear systems. Therefore, the Kalman Filter was further extended into an Extended Kalman Filter (EKF). The EKF is capable of dealing with non-linear systems. EKF linearizes the non-linear system near a given point. It makes use of Jacobians to perform the linearization.

2.4.2 Extended Kalman Filter

Non Linear System

A non linear system can be given by:

$$x_{t+1} = g(x_t, u_{t+1}) + w_{t+1} \tag{2.9}$$

$$y_{t+1} = h(x_t) + v_{t+1} \tag{2.10}$$

The Extended Kalman Filter steps can be written as follows:

Prediction:

$$G_{t+1} = \frac{\partial}{\partial x_t} g(x_t, u_{t+1}) \tag{2.11}$$

$$\bar{x}_{t+1} = g(x_t, u_{t+1}) \tag{2.12}$$

$$\bar{P}_{t+1} = G_{t+1} P_t G_{t+1}^T + Q_{t+1} \tag{2.13}$$

where I is the identity matrix and G_{t+1} is the Jacobian calculated with the help of the vehicle model.

Correction:

$$H_{t+1} = \frac{\partial}{\partial x_t} h(\bar{x}_{t+1}) \tag{2.14}$$

$$K_{t+1} = \frac{\bar{P}_{t+1} H_{t+1}^T}{H_{t+1} \bar{P}_{t+1} H_{t+1}^T + R_{t+1}} \tag{2.15}$$

$$x_{t+1} = \bar{x}_{t+1} + K_{t+1}(y_{t+1} - h(\bar{x}_{t+1})) \tag{2.16}$$

$$P_{t+1} = (I - K_{t+1} H_{t+1}) \bar{P}_{t+1} \tag{2.17}$$

where I is the identity matrix and H_{t+1} is the Jacobian.

2.5 Fault Tolerance

EKF deals effectively with random process and sensor noise with the help of its prediction and correction capability. However, this capability does not make EKF immune to input faults like plausibility faults, frequency faults, or sensor inherent faults. The overall system needs to be made tolerable to input faults that can occur outside its scope. Therefore, the aspect of fault tolerance is important as it aims to ensure proper delivery of a system's services despite the presence of faults [6]. Fault tolerance is the ability of a system to continue performing its intended function in spite of faults. [12] A system can be made fault tolerable at software and hardware level. Fault tolerance includes fault detection and fault mitigation.

Fault detection is the first step towards fault tolerance. Fault detection's aim is to identify the anomaly in the system before faults spread and result in a system shutdown. Bader et al. [6] identifies three major methods:

- Duplication-comparison: consists in comparing results from at least two redundant units that are independent of the faults to tolerate and provide the same service.
- Temporal watchdog: consists in checking a temporal error in a system by controlling its response time, which should not exceed a maximum value (timeout).
- Likelihood checks: seeks to detect errors by checking against aberrant values in the system state.

Once a fault has been detected, the next step is to eliminate the fault (possible via redundancy) or to mitigate the impact of the fault. Many different methods can be utilized to achieve fault mitigation. Bader et al. [6] describe fault correction or system recovery that allows an error-free state to be substituted in place of an erroneous state, which can be made in three ways:

- Recovery restores the system to a correct state that was encountered before the error occurred. This correct state must previously have been saved by the system.
- Pursuit seeks a new state from which the system can function properly (possibly in a degraded mode).
- Error compensation considers that the erroneous state contains enough redundancy to allow it to be transformed into a correct state.

Chapter 3

Literature Review

This chapter reviews the literature related to the major components of the fault tolerant state estimation prototype presented in this thesis. It explores the different avenues that can be adopted to achieve the desired functionality and briefly explains why certain approaches were preferred. Section 3.1 describes the different approaches for state estimation on a high level and offers relevant perception about each approach. Section 3.2 describes various vehicle models that can be used for the process of state estimation. It highlights the degrees of freedom that are involved in vehicle modelling. Section 3.3 presents the different approaches for visual odometry. Section 3.4 discusses some of the different methodologies present for making a system tolerant to different types of faults.

3.1 State Estimation and EKF

This section deals with the classification of estimation methods. As mentioned in Section 2.1.2, state estimation can be broadly categorized into direct and indirect measurement methods. W. Deng and H. Zhang [10] have taken this categorization to the next step according to which the state estimation methods can be classified into direct sensor approach, indirect sensor approach, and model-based approach. A fourth approach can be devised, which intelligently makes use of these approaches while following a certain algorithm to produce adept estimates. These methods can be described as follows:

3.1.1 Direct State Measurement

As the name suggests, this approach refers to the state variables that can be measured directly with sensors. It delivers valuable data about the dynamic state of the vehicle without any integration. An example is that INS measures linear acceleration and angular speed directly. A drawback of this approach would be the inadept handling of throughput of the sensors to ensure smooth fast loop control. Without any fault tolerance, the sensor measurements have a high chance of being plagued by faults like frequency faults, plausibility faults, and noise. Sensors like GPS are improving as the time progresses and provide highly accurate measurements. However, the cost of improvement is quite high, thus making such sensors unviable.

3.1.2 Indirect State Measurement

Indirect state measurement is a broad term and can encompass any approach that is not direct measurement. Regardless, W. Deng and H. Zhang [10] limit the scope of indirect measurement approach of state variables to obtaining a state variable by integrating its rate of change as outputted by existing sensor over time. Examples include yaw from yaw rate, position from vehicle speed, speed from acceleration, etc. Such an approach allows one to obtain states that are not readily available. However, this approach faces certain challenges which adds uncertainty to its results. One of the most prominent challenges is the accumulation of the integration error. Due to the presence of certain factors like noise and drift, every estimate produced by this approach is bound to be plagued with a certain degree of error. This error, if not corrected at every iteration, keeps accumulating over the complete integration period. As such, the indirect state measurement method becomes unviable over a long term. Nevertheless, for short time intervals, using sensors like INS allows accurate predictions with the help of indirect measurement approach.

3.1.3 Model Based State Measurement

A model based approach refers to the estimation of the state variables with the help of vehicle model equations. It uses measurements available with the help of sensors to make predictions of the different state variables. This approach is quite accurate as the vehicle model is less sensitive to sensor noise, thus becoming a more reliable and cost-effective source of information. However, the model based approach is plagued with model inaccuracy. While developing a vehicle model, it is hard to model all factors that can affect the dynamics of a vehicle. Therefore, certain assumptions are made that add to the imprecision of the vehicle model predictions. Also, this method can suffer from process noise and faults.

3.1.4 Intelligent State Measurement:

The intelligent state measurement approaches involve using one or more of the four measurement approaches along with a suitable algorithm in order to produce the best possible set of state variable estimates. A few examples are highlighted below.

- Machine Learning: Machine learning or automated learning refers to the methods by which a computer can be programmed to ”learn” from input available to it. Learning can be defined as the process of converting experince into expertise knowledge. [33] The input to a learning algorithm is training data, representing experience, and the output is some expertise, which usually takes the form of another computer program that can perform some task. [33] There are many different techniques of machine learning, for instance: Principal Component Analysis, Neural Networks, and Linear Discriminant Analysis. However, machine learning can suffer from issues like subpar feature extraction and extensive training periods. Developing successful machine learning applications requires a substantial amount of black art that is hard to find. [11]

- EKF: An EKF is an extended version of the Kalman filter as discussed in Section 2.4. A Kalman filter is the de-facto optimal estimator, which is relatively easy to comprehend and implement, leading to its popularity. EKF has variety of use cases in different environments, which range from large scale aerospace applications to small scale home robotics projects. One of the major development and application of an extended Kalman filter came from the National Aeronautics and Space Administration in 1962 [36] by Smith et al. The authors of the technical report explore the application of statistical filter theory for the estimation of position and velocity on board a circumlunar vehicle. The problem faced by the authors pertained to the measurement of the desired observables (state variables). The measurements were plagued by additive errors and noise, thus were unfit for the determining the position and velocity perfectly. The authors employed a state transition scheme, which is regarded as a dynamical time varying filter that weights the incoming observations in an optimal sense for use in producing an up-to-date optimal estimate of position and velocity. [36] The report provides an implementation of statistical filter theory that lays down the grounds for the successful development and implementation of EKF.

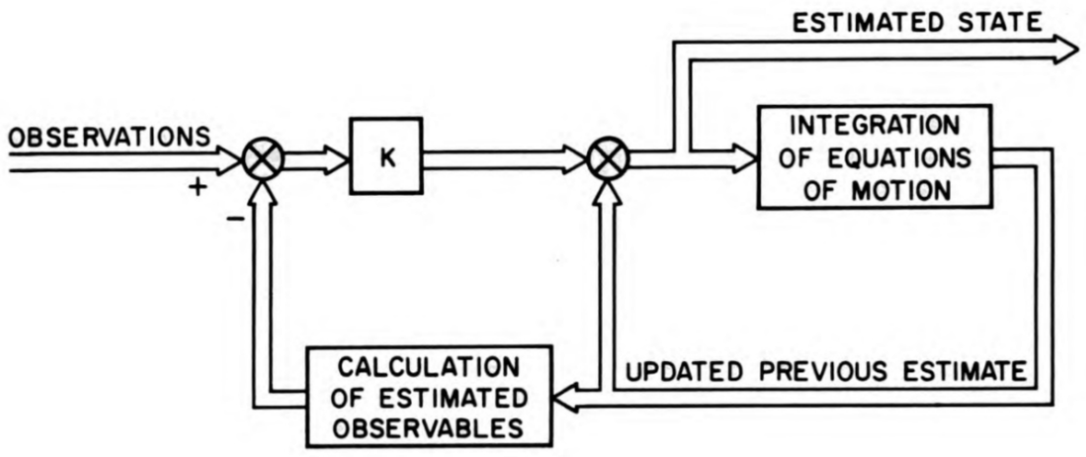

ESTIMATED STATE
OBSERVATIONS
+
−
K
INTEGRATION OF EQUATIONS OF MOTION
CALCULATION OF ESTIMATED OBSERVABLES
UPDATED PREVIOUS ESTIMATE

Figure 3.1: Statistical Theory Estimation Flow. [36]

Figure 3.1 displays the generic sketch from the report, which shows the process of estimating state that was employed. First the integration of equations of motion is initialized to be used as the updated previous estimate or the prediction. This state estimate is used to calculate the current observable measurements. Then the difference between the observed measurements and the calculated measurements is calculated. The difference is then multiplied by the matrix K (known as Kalman Gain in EKF), just before it is allowed to increment the predictions. The new estimate is a new set of starting conditions used for integration of the model equations until the time of the next observation. [36]

- Fuzzy-adapted Kalman Filtering: As optimal as an EKF is, it is not flawless. For example, work done by R. Mehra [22] and by R. Fitzgeraldin [15] depicts the problem of filter divergence. Divergence can generally be stated as the fail situation when the EKF estimate diverge to an incorrect solution. It happens due to complex non-linearity present in the prediction model or the measurement model or both. In another paper, R. Mehra [23] provides a methodology to adapt the measurement noise covariance matrix and prevent filter divergence, i.e. the fuzzy-adapted Kalman filter. The technique of covariance matching is used, which follows the idea that the actual covariance value of the Kalman filter residual is consistent with its theoretical value. [14] Figure 3.2 shows the architecture of the Fuzzy-adapted Kalman Filter.

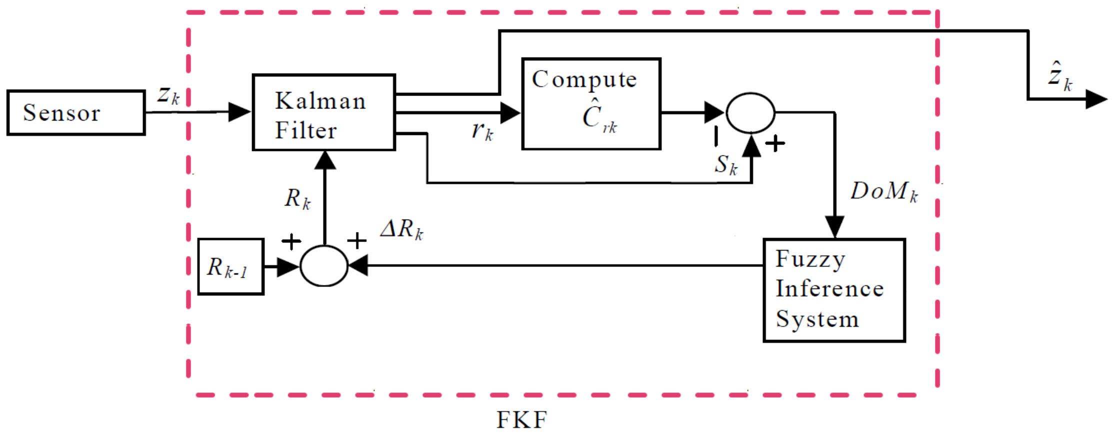


Figure 3.2: Fuzzy Kalman Filter.[14]

The residual r_k is calculated based on the difference between the observed measurements recorded by sensors and the calculated measurements obtained by the measurement model.

$$r_k = (y_k - H_k \bar{x}_k) \tag{3.1}$$

where $k = t$ is the time period, H_k is the measurement Jacobian, $\bar{x}_k$ is the predicted state vector, and y_k is the measurement model vector.

Its theoretical covariance S_k is then defined by:

$$S_k = H_k \bar{P}_k H_k^T + R_k \tag{3.2}$$

where $\bar{P}_k$ is the predicted process covariance, and R_k is the measurement noise covariance.

The actual covariance $\hat{C}_{rk}$ is defined by:

$$\hat{C}_{rk} = \frac{1}{N} \Sigma_{i=i_0}^{N} r_i r_i^T \tag{3.3}$$

where N is the window size that is chosen empirically, and $i_0 = k - N + 1$.

Based on the discrepancy between the theoretical and the actual covariance of the residual, a new parameter known as Degree of Matching (DoM) is defined:

$$DoM_k = S_k - \hat{C}_{rk} \tag{3.4}$$

Since the measurement covariance R influences the theoretical covariance S, it is used to reduce the discrepancy and bring DoM as close to 0 as possible. The R matrix is adjusted based on DoM value as follows:

- If $DoM \;=\; 0$ (this means S and $\hat{C}_r$ match almost perfectly) then maintain R unchanged.
- If $DoM > 0$ (this means S is greater than its actual value $\hat{C}_r$) then decrease R.
- If $DoM < 0$ (this means S is smaller than its actual value $\hat{C}_r$) then increase R.

- Particle Filter: The particle filter is a Sequential Monte Carlo method, where unknown states samples from the state space are used to approximate the system states. These unknown state samples are associated with certain probability mass and density that allows the system to approximate the subsequent filtered distribution. Hence, the particle filter relies on estimation of the sample distribution and not on linearization of the model equations, which is essential because the sampling methods used by particle filter enable the particles to propagate. Different sampling techniques can be used to serve this purpose: regularized particle filter, sampling importance resampling, and even Kalman filter. Using different sampling techniques allows the particles to be assigned a different weights and probability densities. As the time progress, the particles start to converge to the desired probability distribution. However, when compared to KF or EKF, a particle filter is computationally extensive. The greater the number of particles, the higher the accuracy and computational power.

  Regardless particle filter is considered a good substitute to the EKF as stated by Djuric et al [7]. One of particle filters major advantage, apart from the ability to handle any non-linear system, is that it is capable of dealing with non-Gaussian distributions. Comparisons made by [42] and many other studies conclude PF's superiority over EKF.

3.2 Vehicle Model

A vehicle model is the crux of a good vehicle state estimation approach. The more complete the vehicle model, more precise predictions. However, developing a vehicle model that caters all the factors that affect the dynamics of the vehicle accurately is hard. Regardless, models have been developed that make relatively accurate predictions about the state of the vehicle. Depending upon the need of the system, different models suffice the prediction process for the required states. A few models have been discussed in the sections below.

3.2.1 Bicycle Model

The bicycle model is a 2-wheel (front and rear) representation of a 4-wheel car. The aim of the bicycle model is to reduce the computational complexity of the 4-wheel car. It merges the two front tires as one and the two rear tires as one, thus eliminating the axel in between. As a result of the merger the calculation of the effective torque at the centre of gravity becomes simpler. However, bicycle model itself can vary from a simple 2-wheel kinematic model to a complex 3 DoF dynamic model. It can further be elaborated with adding sprung mass force equations, independent tire motion equations, pitch motion, etc. A few of these models have been discussed below.

- Kinematic Model: The simple kinematic model shown in Figure 2.6 consists of equations that describe the pose of the vehicle in a 2-dimensional inertial reference frame. The model consists of three states: x position, y position, and the yaw angle. Though, the kinematic model is simple, it is widely used in the industry due to its low computational complexity. The effectiveness of such a model in predicting the required states has always been a matter of concern. Philip Polack et al. in their paper [27] discuss the effectiveness of such a model in comparison to a higher fidelity 9 DoF model. They highlight the modelling errors and limitations of the kinematic model and specify the efficient consistency criterion to validate the use of the kinematic model.

$$\begin{bmatrix} x_t \\ y_t \\ \theta_t \end{bmatrix} = \begin{bmatrix} x_{t-1} + v_t \cos\theta_{t-1} + \delta_t dt \\ y_{t-1} + v_t \sin\theta_{t-1} + \delta_t dt \\ \theta_{t-1} + \frac{v_t \tan\delta_t}{L} dt \end{bmatrix} \tag{3.5}$$

  where,

$$\delta_t = \begin{cases} 0, \text{rear wheel} \\ \delta_t, \text{front wheel} \end{cases} \tag{3.6}$$

  x, y are the x and y coordinates in the inertial plane, θ is the yaw angle, δ is the heading angle, v is the vehicle velocity, and dt is the time step.

- Bicycle 3 DoF dynamic model: The 3 DoF dynamic bicycle model is an upgraded version of the simple kinematic bicycle model. The three degrees of freedom are lateral motion, longitudinal motion, and yaw motion. Equations below describe the model in the respective degrees. The 3 DoF model is widely used where the dynamics of the vehicle are concerned. For example, the 3 DoF bicycle model was used for slip angle and yaw rate estimation by Song et al. [37].

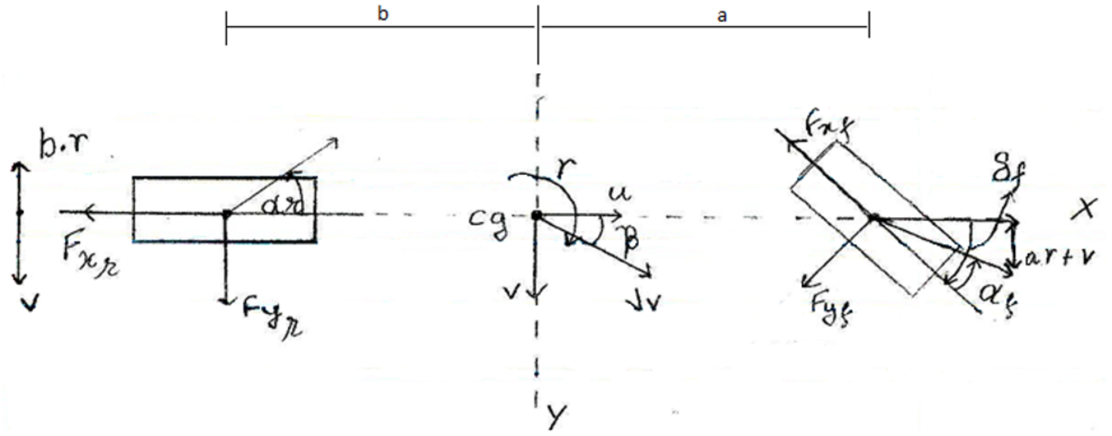


Figure 3.3: Bicycle 3 DoF Dynamic Model.

The primary equations that can be derived from this model are represented below.

The longitudinal motion of the vehicle can be represented by:

$$m(\dot{u} - \dot{\psi}v) = -F_{xf}\cos\delta_f - F_{yf}\sin\delta f - F_{xr} \tag{3.7}$$

The lateral motion of the vehicle can be represented by:

$$m(\dot{v} - \dot{\psi}u) = F_{yf}\cos\delta_f - F_{xf}\sin\delta_f + F_{yr} \tag{3.8}$$

The yawl motion i.e. motion about the yaw axis of the vehicle can be represented by:

$$I_z\ddot{\psi} = a(F_{yf}\cos\delta_f - F_{xf}\sin\delta_f) - b(F_{yr}) \tag{3.9}$$

where,

m = Mass of the vehicle,

u, v = Longitudinal and Lateral velocity of the vehicle,

δ = Steering (heading) angle,

I_z = Moment of inertia of the vehicle about the Z axis,

a, b = Distance of centre of gravity (c.g.) of the vehicle from the front and rear axle respectively,

$F_{xf}, F_{xr}, F_{yf}, F_{yr}$ = Tire forces front x, rear x, front y, rear y,

$\dot{\psi} = r$ = Yaw rate

3.2.2 3 DoF Dynamic Model

A 3 DoF dynamic bicycle model can be expanded into a regular 4-wheel 3 DoF dynamic model. This expansion adds in certain force components to the longitudinal, lateral and yaw equations that were ignored in the bicycle model. The 4-wheel motion model along with its equation is described is referenced in [4].

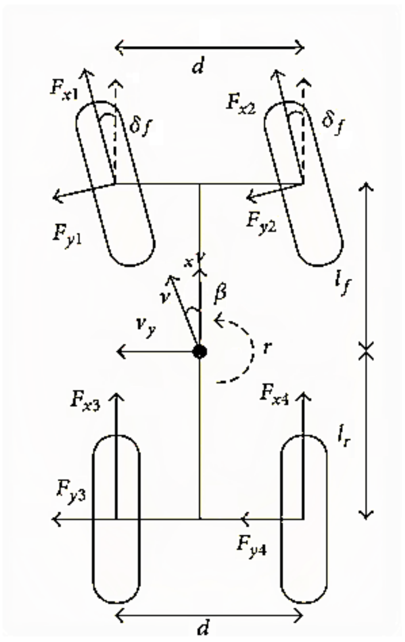


Figure 3.4: 3 DoF Dynamic Model [4].

The equations of motion are quite similar to the bicycle model. The additional forces that were neglected in the Bicycle 3 DoF Dynamic Model are represented in the equations below.

Longitudinal Motion:

$$m(\dot{v}_x - rv_y) = (F_{x1} + F_{x2})\cos\delta_f + F_{x3} + F_{x4} - (f_{y1} + F_{y2})\sin\delta_f \tag{3.10}$$

Lateral Motion:

$$m(\dot{v}_y + rv_x) = (F_{x1} + F_{x2})\sin\delta_f + (F_{y1} + F_{y2})\cos\delta_f + F_{y3} + F_{y4} \tag{3.11}$$

Yaw Motion:

$$I_z\dot{r} = l_f(F_{y1}\cos\delta_f + F_{y2}\cos\delta_f + F_{x1}\sin\delta_f + F_{x2}\sin\delta_f) - l_r(F_{y3} + F_{y4}) \tag{3.12}$$

where,

m = Mass of the vehicle,

v_x, v_y = Longitudinal and Lateral velocity of the vehicle,

δ = Steering (heading) angle,

I_z = Moment of inertia of the vehicle about the Z axis,

l_f, l_r = Distance of centre of gravity (c.g.) of the vehicle from the front and rear axle respectively,

$F_{x1}, F_{x2}, F_{x3}, F_{x4}$ = Longitudinal tire forces of tire 1,2,3,4,

$F_{y1}, F_{y2}, F_{y3}, F_{y4}$ = Lateral tire forces of tire 1,2,3,4,

r = Yaw rate

3.2.3 7 DoF Dynamic Model

A 7 DoF model is essentially a 3 DoF model with added equations that govern the independent rotation motion of the 4 wheels, which is further depicted below.

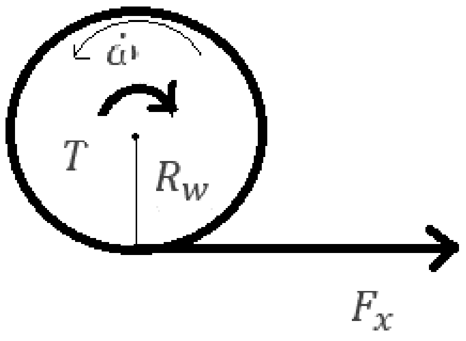


Figure 3.5: Independent Motion of Tire.

The Longitudinal, Lateral, and Yaw motion equations have already been represented in Equation 3.10, 3.11, and 3.12. The 7 DoF model adds in additional independent motion of the tire that can be represented by:

$$I_{w1}\dot{\omega}_1 = T_1 - F_{x1}R_w \tag{3.13}$$
$$I_{w2}\dot{\omega}_2 = T_2 - F_{x2}R_w \tag{3.14}$$
$$I_{w3}\dot{\omega}_3 = T_3 - F_{x3}R_w \tag{3.15}$$
$$I_{w4}\dot{\omega}_4 = T_4 - F_{x4}R_w \tag{3.16}$$

where,

$I_{w1/2/3/4}$ = Moment of inertia of the tire 1, 2, 3, 4

$\omega_{1/2/3/4}$ = Angular velocity of the tire 1, 2, 3, 4

$T_{1/2/3/4}$ = Torque acting on the tire 1, 2, 3, 4

$F_{x1/2/3/4}$ = Total longitudinal force acting on the tire 1, 2, 3, 4

R_w = Radius of the tire

3.2.4 9 DoF and Higher Models

The 7-DoF model can be expanded with higher fidelity by adding in more degrees of freedom. For example, a 9 DoF model adds in roll and pitch motion equations to the model. Further expansion can take place by adding in sprung mass forces, un-sprung mass forces and many more.

The complexity of the model is decided by the requirements of the project. A higher fidelity model will try to provide better and more accurate predictions. But these models are computationally expensive, which might be undesirable given that a fast-iterative rate is desired for vehicle state estimations. Also, a higher DoF does not necessarily ascertain accurate predictions. Every equation governing a DoF and in turn every parameter involved in the equation contains a certain amount of error. When the number of parameters grows, the uncertainty caused by this error also grows, thus increasing the unreliability of the model. Therefore, the model fidelity is chosen based on the constraints of the project and on the reliability of the model.

3.3 Visual Odometry

Visual odometry as discussed in Section 2.2 computes the relative evolution of the pose of the vehicle based on a sequence of images obtained via. a camera mounted on the vehicle. The different types of VO based on the type of camera i.e. monocular and stereo VO have already been discussed briefly in Section 2.2.2 This section will primarily focus on monocular VO as it is the one used in this thesis. Monocular VO was primarily chosen due to its simplicity.

A general categorization can be made in order to classify monocular VO methods into feature-based methods, appearance-based methods, and hybrid methods. [30] Feature-based methods use features that are identified and traced over subsequent image frames. Appearance-based methods rely on the intensity and density information of all the pixels that comprise the image. A hybrid method tends to combine both the feature-based and appearance-based methods.

Much work has been done in the field of feature based methods for visual odometry. Nister et al. in their work [26] initiated the development of visual odometry by demonstrating a major implementation of visual odometry in real time. For the purpose of pose calculation, a 3D to 2D estimation technique was used. Most of the present-day techniques rely on an eight-point minimal solver RANSAC for detecting outliers. However, Nister et al. used a five-point solver RANSAC to filter out any undesired outliers. Another novel approach for visual odometry was presented by Mouragnon et al.[25]. They made use of bundle adjustment along with a five-point RANSAC over a local window to recover the body motion.

Many approaches were developed, which focused on using the majority of the pixel information available from a frame rather than only looking at specific features. Milford and Wyeth [24] presented a method to extract approximate rotational and translational velocity information from a single perspective camera mounted on a car. Scaramuzza and Siegwart [31] in their work made use of a hybrid combination of appearance based and feature based approaches. The feature based approach was used to estimate the translation and scale, whereas the rotation of the vehicle was estimated by the use of appearance based approach.

3.4 Fautl Tolerance

This section summarizes related work which augments estimators with fault tolerance.

3.4.1 Hybrid Kalman Filter-Fuzzy Logic Architecture For Multisensor Data Fusion

Escamilla and Mort [14] introduce a novel approach where the capabilities of a Fuzzy Kalman Filter are used to obtain fused measurement data that determines the parameter being measured as precisely as possible. Figure 3.6 displays a high-level view of the architecture.

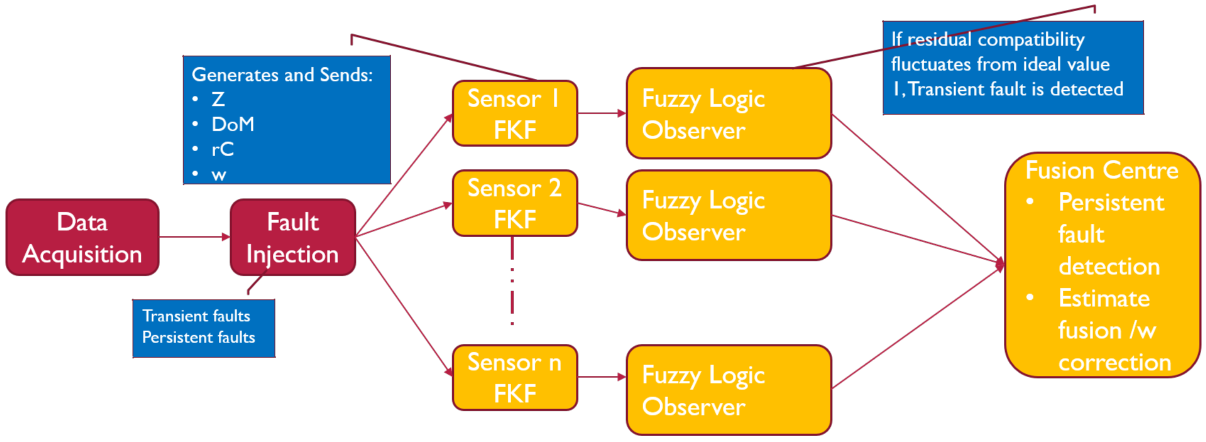


Figure 3.6: Hybrid Kalman Filter Architecture.

This architecture relies on hardware redundancy of the source of a particular measurement. Measurements of each sensor are introduced to an independent FKF. Each FKF outputs the associated estimated state Z, the DoM, and the Residual Compatibility rC. The DoM and rC along with the measurement covariance matrix is then forwarded to the Fuzzy Logic Observer (FLO). The parameter rC is defined as:

$$rC = \frac{|r|}{\sqrt{S}} \tag{3.17}$$

where,

r = Residual,

S = Theoretical covariance

The FLO associates weight w as a measure of trust with the estimated state output. The range of the weight lies between the interval [0,1], which helps in the detection of transient faults. The weight w, weighted estimate wZ and the estimate Z are then forwarded to the Fusion Centre block. The fusion centre uses a voting scheme to determine if any of the received estimates are faulty. If any estimate differs from the other estimates above a certain threshold, then that estimate is deemed faulty and a persistent sensor fault is declared. The fusion centre ignores this estimate while performing the final fusion.

3.4.2 Fault Tolerant Architecture for Data Fusion Targeting Hardware and Software Faults

Another approach is highlighted by Bader et al. [18]. This approach again relies on the hardware redundancy, but follows a different detection and correction scheme. A high-level block diagram for the architecture is shown in Figure 3.7 below.

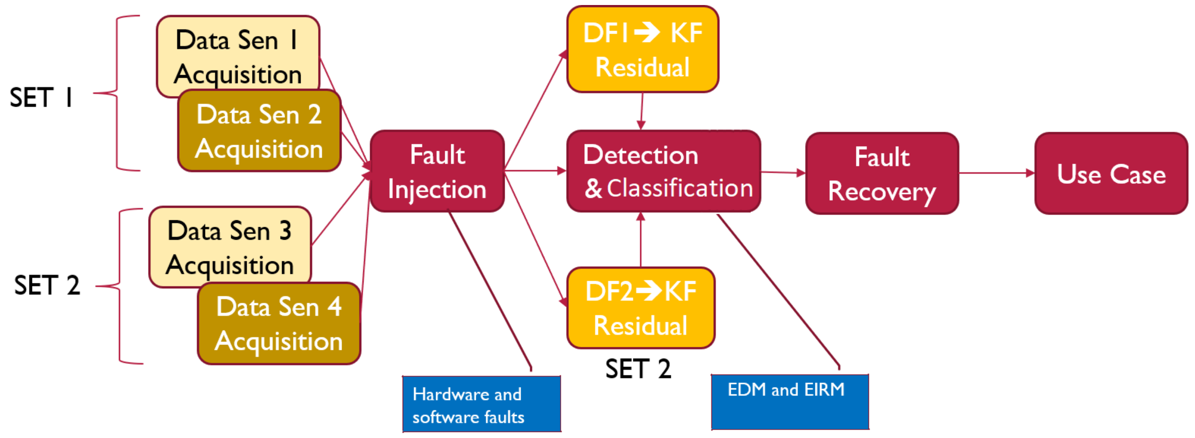


Figure 3.7: Architecture Focused on Hardware Redundancy.

Hardware redundancy dictates that more than one source of each measured variable be available. Set 1 consists of a sensor set providing data like position from GPS and velocity from wheel speed sensor. Set 2 outputs same information from different sensors, such as position information from Laser Odometry and velocity information from INS. This information is then forwarded to their respective data fusion blocks DF1 and DF2 for producing respective state estimates and residual information. The state estimate information is then passed on to the Error Detection Module (EDM), where based on the discrepancy between the two estimates, the EDM decides whether an error is present. If an error is detected, the Error Identification and Recovery Module (EIRM) is initialized, which tries to determine if the fault or error that has occurred is a software or a hardware fault. The EIRM checks for discrepancy between the identical measured variables from both data sets. If both measurements are identical, a software fault is declared. If the discrepancy is above a certain threshold, a hardware fault is declared. In the case of a hardware fault, the sensors are the culprits. The EIRM performs residual checks and looks at the residual information generated by the DF blocks. Whichever block has the highest residual, it is deemed as the faulty sensor branch. In the case of a software fault, additional branches can be installed that take input sets formed from any combination of sensors. A majority wins scheme is then applied to detect and reject the faulty software branch. This approach effectively eliminates the faults that the system encounters. However, it is very resource heavy.

3.4.3 Approach 3

Another example of a diverse method is the knowledge based system approach that has been introduced by Silva et al.[35]. Their approach uses machine learning to recognize the faults. The high-level architecture is highlighted in the Figure 3.8 below.

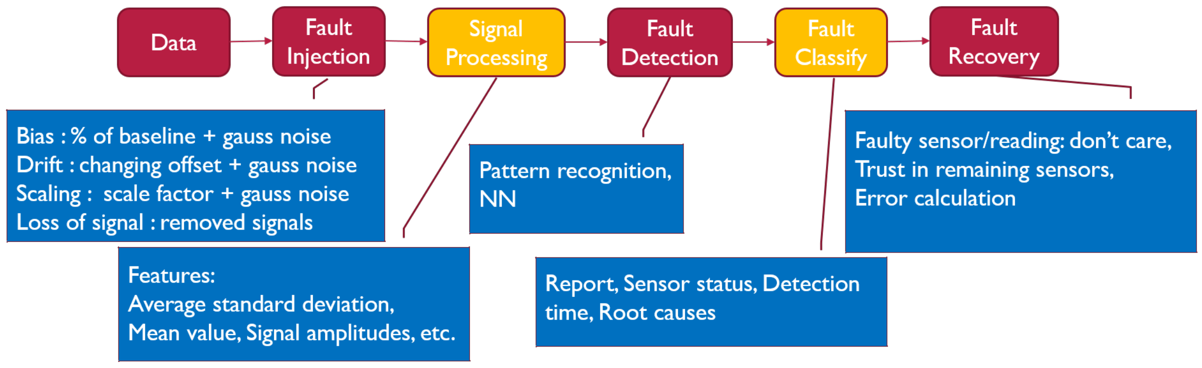

Data
Fault Injection
Signal Processing
Fault Detection
Fault Classify
Fault Recovery
Bias : % of baseline + gauss noise
Drift : changing offset + gauss noise
Scaling : scale factor + gauss noise
Loss of signal : removed signals
Features:
Average standard deviation,
Mean value, Signal amplitudes, etc.
Pattern recognition,
NN
Report, Sensor status, Detection time, Root causes
Faulty sensor/reading: don't care,
Trust in remaining sensors,
Error calculation

Figure 3.8: Architecture that employs Machine Learning.

The measured data from the sensors is introduced to the system as signals. In the following block the process of feature extraction takes place. Features like average standard deviation, mean value, signal amplitudes, etc. are extracted over fixed n sized window. The extracted feature information is then subjected to machine learning techniques like Neural Networks, which have been trained on similar datasets. The machine learning algorithms are trained to detect patterns of bias, drift, scale or drop. Based on the training, the technique tends to recognize any faults that it has been trained for. Once a fault is detected the system generates a report of the fault, the sensor status, detection, time, and root causes. For the purpose of correcting fault, two approaches are proposed. First is the don't care approach, where the faulty sensor is ignored/-dropped and the system imbues trust in remaining sensors. The secondary approach deals with error calculation and subtracting it from the original signal to obtain the corrected signal.

Chapter 4

Components, Architecture, and Approach

This chapter describes the overall approach to the fault-tolerant vision-augmented localization. Section 4.1 gives an overview of the involved components. Section 4.2 describes the architecture. Section 4.3 outlines the test process used to evaluate the architecture.

4.1 Components

4.1.1 Visual Odometry

This work relies on monocular visual odometry. As discussed in Section 3.3 there are different methods including feature based method, appearance based methods, and hybrid methods for performing visual odometry with a monocular camera. Out of these three methods, the feature based approach was chosen. An off the shelf implementation package libvisio2 [17] of monocular visual odometry provided by the Karlsruhe Institute of Technology and Toyota Technological Institute (KITTI) has been used. The KITTI infrastructure provides prerecorded image sets of different scenarios, which are accompanied with data sets containing sensor information like GPS, INS, and camera(s). Another advantage of using the libvisio2 package is that it can be directly used with Robotic Operating System (ROS) by the means of a ROS wrapper [41]. The MATLAB implementation of the libvisio2 package simplified integration and testing of visual odometry with the state estimation process.

The steps followed for monocular odometry in libvisio2 package are:

1. Consecutive set of image frames is obtained,

2. If the images are raw, they are rectified,
3. Features are detected with the help of methods like Harris corner detector and tracked over consecutive image frames,
4. Once the desired number of point correspondences are obtained, the fundamental matrix F is calculated using RANSAC and 8-point algorithm,
5. Next the essential matrix E is calculated with the help of the available camera calibration parameters,
6. The essential matrix is then used to obtain the rotation and translational information $R|t$,
7. The 3-D points are also computed from the 2-D bearing data,
8. Based on the computed 3-D points, the ground plane is estimated. Estimation of the ground plane provides information about camera height and camera pitch,
9. Camera height and pitch are then used to scale the $R|t$ information,

All estimates are relative to some unknown scaling factor; libviso2 overcomes this shortcoming by assuming a fixed transformation from the ground plane to the camera (parameters camera height and camera pitch); To obtain these values, in each iteration the ground plane has to be estimated. [17]

The rotation and translation information (R—t) is accumulated to estimate the vehicle trajectory.

The base libvisio2 package is modified with custom message implementation to maintain consistency throughout the project.

Different type of information is available from visual odometry, like position, orientation, and velocity. For the purpose of state estimation, velocity information is used as measurement to the EKF.

4.1.2 Vehicle Model

A number of different vehicle models were tested with different configurations for the development of the prototype. The 3 DoF 2- wheel bicycle dynamic model is shown in Figure 3.3. The model equations are described in Section 3.2.1

The estimation model of the vehicle dynamics neglects suspension dynamics in order to keep the size of estimation model and sensing requirements to a minimum. These equations form the first portion of the state vector that will be used in the EKF process. This state vector is further augmented to include differential equations for each force. As per [28], an integrated random walk is chosen to model each force, allowing the forces to vary with time as necessary.

$$\begin{bmatrix} \dot{f}_0 \\ \dot{f}_1 \\ \dot{f}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \\ f_2 \end{bmatrix} + w_y \tag{4.1}$$

f_0 represents the force to be estimated, f_1 and f_2 are the first and second time derivatives of the force, and w_y is random white noise, which is Gaussian distributed in our case. These equations are then replicated for each of the considered forces (four in total) and appended to the original state equations set. The final vehicle model state vector to be used in EKF can then be described as:

$$X_{t+1} = \begin{bmatrix} x_{t+1} \\ y_{t+1} \\ \psi_{t+1} \\ v_{x_{t+1}} \\ v_{y_{t+1}} \\ \dot{\psi}_{t+1} \\ F_{xf_{t+1}} \\ \dot{F}_{xf_{t+1}} \\ \ddot{F}_{xf_{t+1}} \\ F_{xr_{t+1}} \\ \dot{F}_{xr_{t+1}} \\ \ddot{F}_{xr_{t+1}} \\ F_{yf_{t+1}} \\ \dot{F}_{yf_{t+1}} \\ \ddot{F}_{yf_{t+1}} \\ F_{yr_{t+1}} \\ \dot{F}_{yr_{t+1}} \\ \ddot{F}_{yr_{t+1}} \end{bmatrix} = \begin{bmatrix} x_t + (v_{x_t}\cos\psi - v_{y_t}\sin\psi)dts \\ y_t + (v_{x_t}\sin\psi + v_{y_t}\cos\psi)dts \\ \psi_t + (\tan\delta\frac{v_{x_t}}{(a+b)})dts \\ v_{x_t} + (v_{y_t}\dot{\psi}_t + \frac{1}{m}(F_{xf_t}\cos\delta - F_{yf_t}\sin\delta + F_{xr_t}))dts \\ v_{y_t} + (v_{x_t}\dot{\psi}_t + \frac{1}{m}(F_{yf_t}\cos\delta + F_{xf_t}\sin\delta + F_{yr_t}))dts \\ \dot{\psi}_t + (\frac{1}{I_z}(aF_{yf_t}\cos\delta + aF_{xf_t}\sin\delta - bF_{yr_t}))dts \\ F_{xf_t} + \dot{F}_{xf_t}dts \\ \dot{F}_{xf_t} + \ddot{F}_{xf_t}dts \\ \ddot{F}_{xf_t} \\ F_{xr_t} + \dot{F}_{xr_t}dts \\ \dot{F}_{xr_t} + \ddot{F}_{xr_t}dts \\ F\ddot{x}r_t \\ F_{yf_t} + \dot{F}_{yf_t}dts \\ \dot{F}_{yf_t} + \ddot{F}_{yf_t}dts \\ F\ddot{y}f_t \\ F_{yrt} + \dot{F}_{yr_t}dts \\ \dot{F}_{yr_t} + \ddot{F}_{yr_t}dts \\ \ddot{F}_{yr_t} \end{bmatrix} \tag{4.2}$$

The measurement model is a simple direct input of the measurement state variables.

$$Y_{t+1} = \begin{bmatrix} \bar{x}_{t+1} \\ \bar{y}_{t+1} \\ \bar{\psi}_{t+1} \\ \bar{v}_{x_{t+1}} \\ \bar{v}_{y_{t+1}} \\ \bar{\dot{\psi}}_{t+1} \end{bmatrix} \tag{4.3}$$

4.1.3 Sensors

Different sensors can be used to provide measurement inputs required for the EKF process. The conventional set of sensors includes GPS, INS, Laserscanner, and wheel odometry. Due to simulation limitations and prototype requirements, only GPS and INS are used to provide measurements to the EKF process.

- GPS: GPS provides position information. Specifically, the x and y coordinates are used. It is assumed that there is no motion in the z direction. These coordinates are with respect to the odom frame, which is a world fixed frame.
- INS: Provides the orientation information of the vehicle. Specifically, the yaw angle and yaw rate are used. The orientation is obtained with respect to the odom frame.
- Camera and VO: Camera provides the image sequence necessary for the construction of visual odometry. VO acts as an added sensor to provide longitudinal and lateral velocity measurement information to the EKF process.

4.1.4 Faults

Primarily two types of faults were injected for the purpose of observation.

- *Error Code Faults:* Error code faults are those reported by the sensor hardware and software. The sensors being used to obtain the measurement data can malfunction, break, or underperform under several conditions. To aid the end user diagnosis, manufacturers equip the sensors with mechanisms to indicate faults. These mechanisms help in identifying the extent to which the data might have been corrupted. A common measure employed is to set Diagnostic Trouble Codes or Error Codes once some anomaly occurs.

- *Plausibility Faults:*Plausibility faults refer to the faults that make the data unbelievable for the current instance of time. Different types of plausibility faults used in this thesis will be discussed in Section 4.2.2

4.1.5 Fault Tolerant Strategy

Based on the type of fault detected, the severity of the fault is determined by the Integrity Checking (IC) node. The IC makes the following changes to the covariance of the data for their associated severity:

Table 4.1: Integrity Checker Actions.

Severity	Transient	Permanent
Normal	No change	No change
Degraded	$Cov \times 2/3/4$ i.e. mult factor can change	$Cov \times 2or3or4$ i.e. mult factor is fixed
Unusable	$Cov \times 10000$	$Cov \times 10000$

For a transient degraded fault, based of certain factors, the multiplicator for a particular type of data stream can change over time even when the fault remains the same. For example, if a fault persists in a data stream and crosses a fixed threshold of the number of consequtive messages, the multiplactor value will be increased from its initialized value to indicate the increase in unreliability. However, for a permament degraded fault, in a similar scenario, the mulitplicator value does not changes from its initialized value. The IC node also modifies the status field of the associated data with the tags Normal, Degraded, and Unusable. Based on these changes, the EKF node reacts accordingly by adjusting its measurement covariance matrix R or by triggering its residual check gateway. This approach helps in minimizing the effect of the faulty measurements.

4.2 Architecture

Section 4.2.1 describes the complete proposed architecture to include fault-tolerance into the estimation process. Section 4.2.2 presents an implementation of simplified version of the architecture that was used in the experimental evaluation.

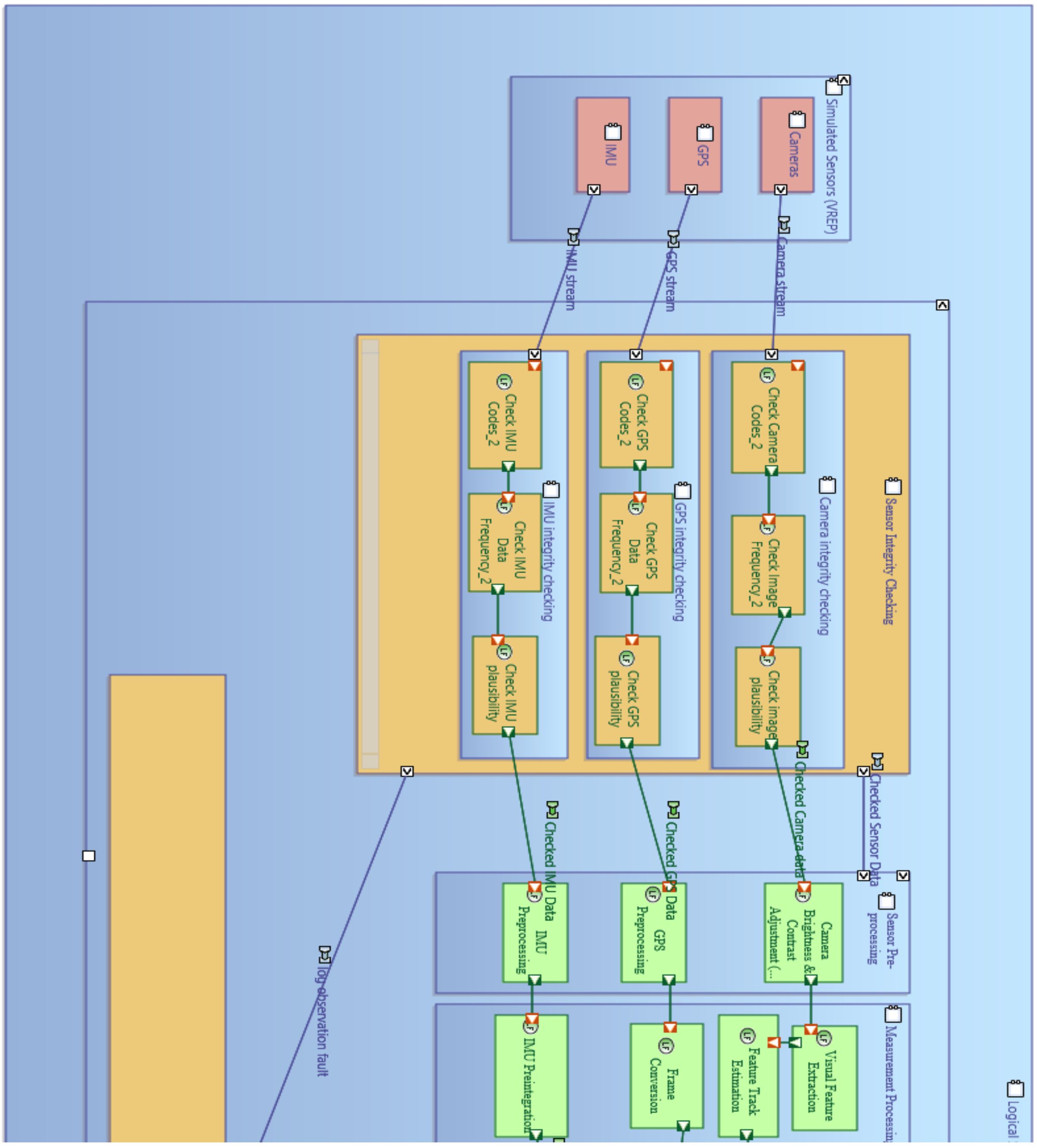


Figure 4.1: Proposed State Estimation with Fault Tolerance Architecture Part 1.

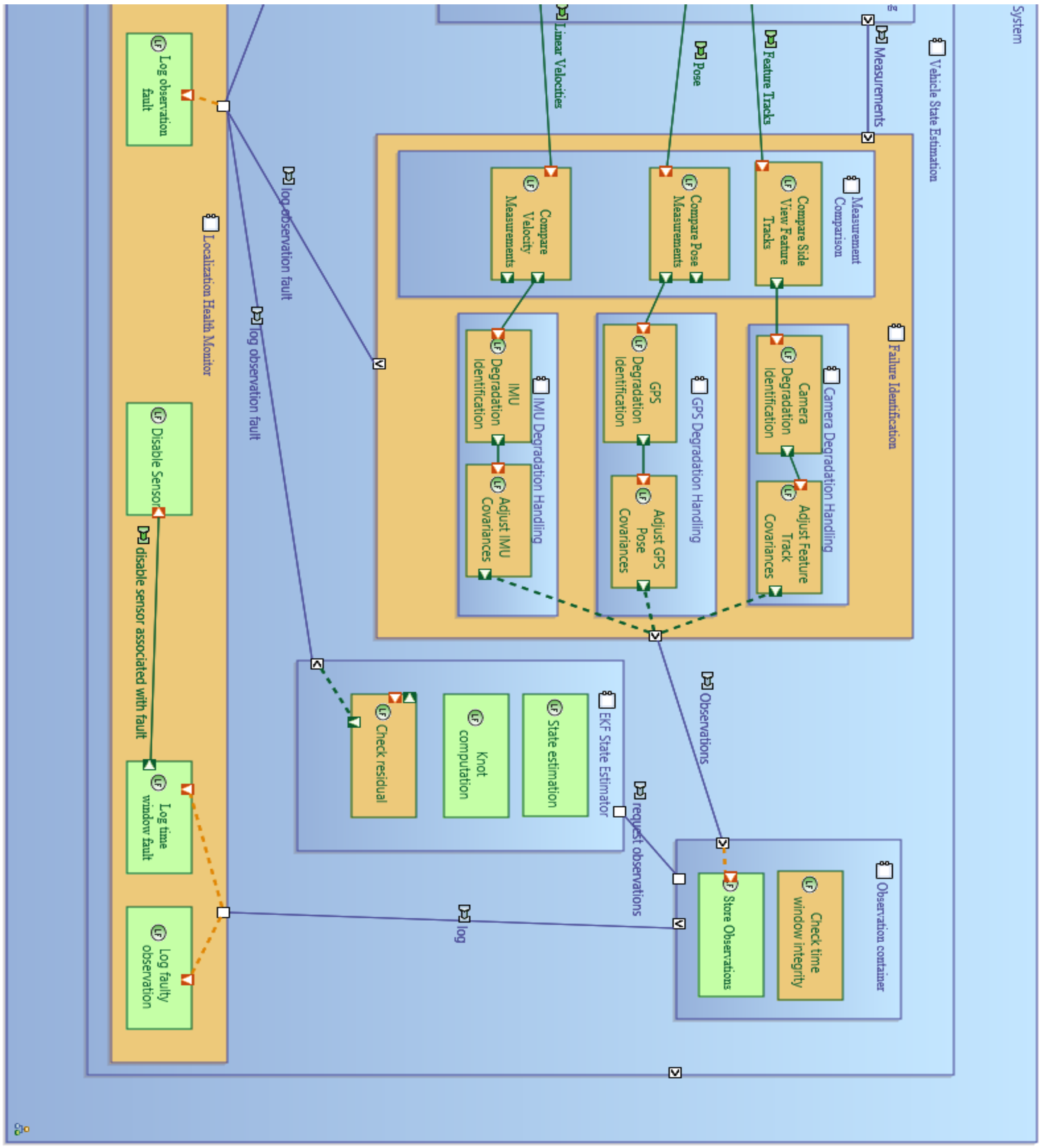


Figure 4.2: Proposed State Estimation with Fault Tolerance Architecture Part 2.

4.2.1 Proposed Architecture

The proposed architecture was developed with the goal to prototype fault tolerant localization strategy. The architecture entails the merger of the fault tolerant strategy with the state estimation approach in a seamless way. In the architecture, the input data message is accompanied with a status field that describes the health of the particular data message. Detailed message format can be found in the appendix A.1.

This data is then processed through an integrity checking layer, where it is checked for faults like error codes, frequency faults, and plausibility faults. If a fault is detected, the integrity checking layer alters the status field of the particular message in correspondence to the type of fault.

Next the data goes through a preprocessing layer where data conversions, specific data extraction, and transformations take place. Once the desired measurement data is available, it is subjected to a failure identification/isolation layer. This layer alters the covariance information of each of the faulty messages in accordance with the nature of the fault and the status of the message. At the end, the failure identification layer passes the data to the container, which stores all the data that it receives. The container node also performs a time window check to detect if a certain type of fault persists.

The container is then be polled by the EKF process to receive the latest measurement data. The EKF node adds in an additional failure check by making use of a residual check. Throughout the process the necessary meta information associated with a message and any changes made by the layers are preserved till the very end.

Every layer is connected with the health monitor, which collects logs of any events related to faults and alterations made by the different layers. The health monitor also has the capability to turn off certain sensors in case of undesired conditions.

4.2.2 Implemented Architecture

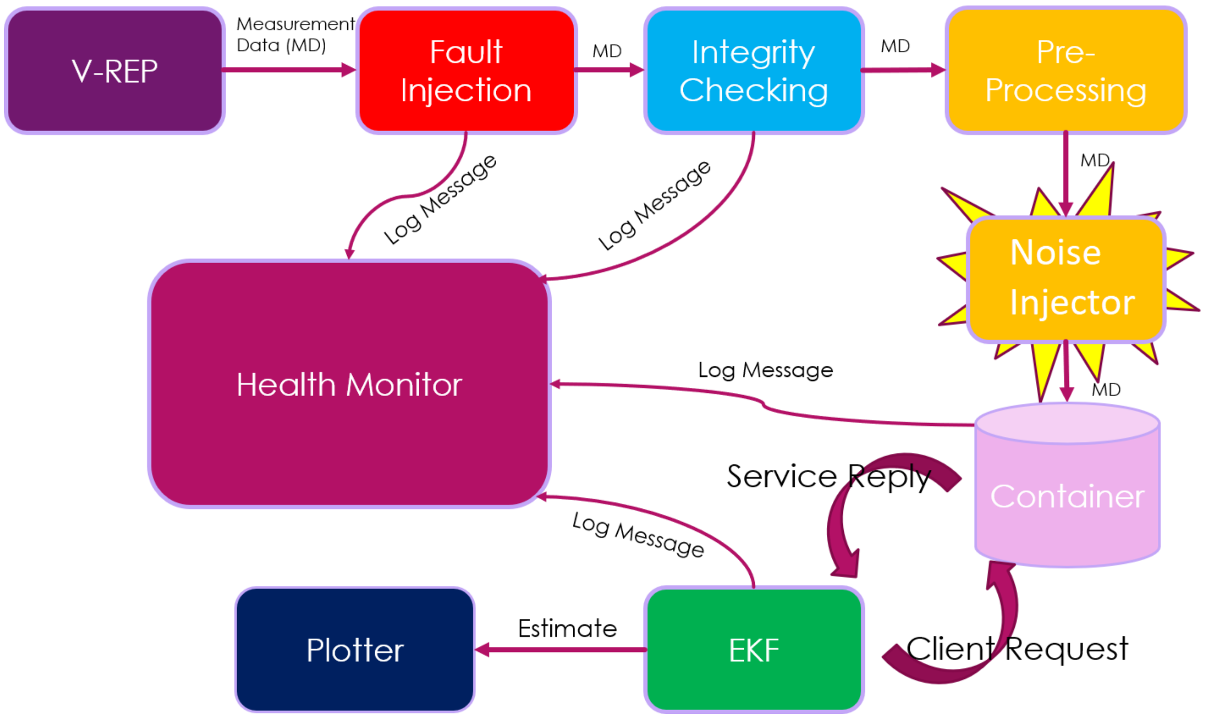


Figure 4.3: Implemented High Level Architecture.

Figure 4.2 provides an overview of the sample implementation of the architecture from Figure 4.1. The implementation includes components for fault and noise injection that were used in the experimental evaluation. Each of the components was implemented as a ROS node.

Data Source

The source of all the measurement data is a simulation environment developed inside the simulator VREP. The simulation makes use of a vehicle object that can be driven freely in the virtual world. The simulator allows the addition of different objects like trees, boards, walls, and buildings in order to construct and alter the simulation world as desired. VREP comes with a number of built in or custom sensor objects that can be added to the desired object. These sensor objects include vision sensors, proximity sensors, and GPS.

The vehicle object in simulation extended with a number of different sensor objects. The sample implementation uses GPS, IMU, and Vision Sensor to obtain the desired information.

The data from these sensors is published in the ROS message format, which is accompanied by the standard header meta data. Figure 4.4 shows a screenshot of the simulation environment.

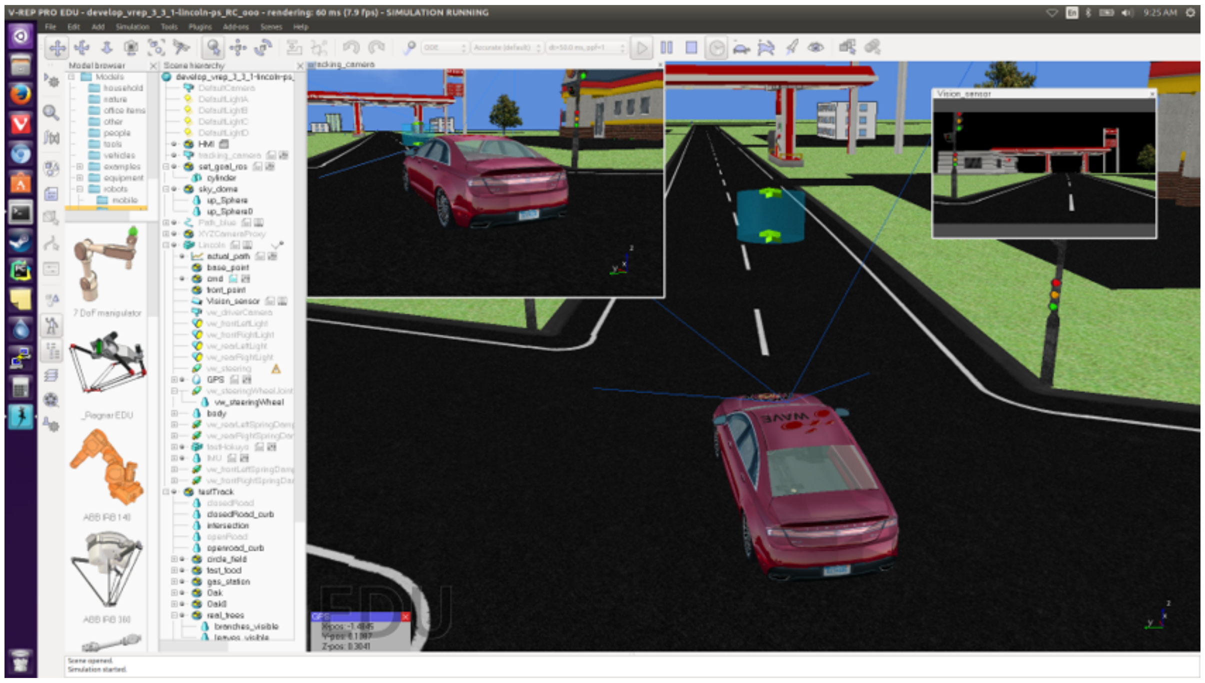


Figure 4.4: VREP Simulation Scene.

Fault Injection (FI) Node

Once VREP publishes data on particular topics, the fault injection node subscribes to the said topics to obtain the measurement data. FI node essentially performs two duties.

- Fault Injection: Based on a predetermined list of faults and scenarios, the fault injector injects faults inside the data stream of the sensor. These faults and scenarios can be of any nature as discussed in Section 4.1.4.
- Data Encapsulation: Once the fault is injected, the individual data messages are encapsulated into a custom message format. This message format includes additional meta data information like status, GUID, and sequence.

Finally, at the end, the FI node publishes out the custom messages onto new topics.

Integrity Check (IC) Node

The integrity check node subscribes to the FI node published topics. Once these messages are received by the IC node, it performs three main tasks:

- Error code detection: If an error occurred at the sensor level, the message from the said sensor is accompanied with an error code. Depending upon the nature of the error code, the IC node can alter different information like the status and the covariance of the said message.
- Frequency check: The IC node looks for any delays in the incoming stream by making use of the timestamp information present in the meta data. If any of the sensor message at a given time deviates from the other messages above a certain threshold, a frequency delay or drop is detected. To recover from this anomaly, the IC node decides to send the last known good message. The covariance accompanying the message is modified to reflect that the message reliability has decreased. If a message is absent over more than 10 iterations, the IC node generates a dummy message. The status of the dummy message is then set to unusable.
- Plausibility check: The IC node also performs certain plausibility checks. It looks for any inconsistencies in the measurement stream. Inconsistencies are checked in the form of plausibility faults, which are:
  - Out of bounds: If a sensor message contains a value that is out of bounds of the range for that particular type of message, a plausibility fault is detected. For example, if the RGB values of any of the pixels of the image is greater than 255 and less than 0, a plausibility fault is detected.
  - Jumps or discontinuities: The slope formed by the consecutive sensor messages is also checked by the IC. If there are any abrupt changes in the slope. For example, if for a particular GPS message, the x coordinate is 10 and in the next message the coordinate jumps to a value of 1000, a plausibility fault is detected.

  If any plausibility fault is detected for a measurement, the associated status is set to unusable. Any time a message is set to unusable, the covariance associated with the message is set very high i.e. 10000 to indicate that the said measurement is untrustworthy.

Finally, IC publishes out the messages over new topics, which are available for other nodes to subscribe.

Preprocessing Node(s)

Preprocessing node layer consists of different nodes that are required to obtain the desired data from the data that is received. This process involves:

- Extract the desired data: In this process, the desired data is extracted from the raw data. For example, the VO node subscribes to the data published by the IC, which is in the form of images. These images individually are of no use to the EKF estimation process. Therefore, the VO node processes these images to extract the pose and velocity of the vehicle.
- Data Transformation: Data available from the IC node can be in different formats and in different reference frames. For example, the GPS data available from the sensors is in the LLA format. For this data to be of any use in the EKF estimation process, it must be converted to the odometry frame. Such preprocessing aligns the data with the data available from all the sources, and allows for consistency and accuracy in the estimation process.

Noise Injector Node

Since the data available originates from ideal simulated sensors, the noise injector node allows adding noise to the data to simulate realistic signals. This noise is white Gaussian noise, which is modeled with the variance obtained from the covariance attached with each sensor message.

Container Node

The container subscribes to the topics published by the noise injector node. It receives the messages and stores them in a tabulated structure, which can be accessed based on the timestamp.

The container also runs a ROS service that allows any node to query the container for the latest available message set.

A major function performed by the Container node is the time window check. If the container receives 20 or more subsequent messages (from the same sensor) that are tagged with the unusable status, the container sets a flag for the sensor and sends out a request to the health monitor to turn off the faulty sensor.

EKF Node

The EKF is the core of the estimation process. It houses a client that polls the container for the latest available data on each iteration of the EKF. The Container node runs the service, which answers to the clients call for the latest data. If a response from the service is received, the call is deemed successful. Once the EKF receives the data, it starts off by isolating the important information from the encapsulated messages. The necessary measurement data that EKF needs to complete its iteration for a given time instance includes x and y coordinates, yaw angle, yaw rate, and x and y velocity along with the measurement covariance values for each of the said measurements. This data aids in construction of the measurement noise covariance matrix R as well as the measurement vector Y.

The process covariance matrix P is initialized with mid values for the known states and high values for the unknown states. Even though we initialize the EKF with relatively accurate information, setting the P matrix to low values is unfavourable. Some scenarios dictate the injection of faults into the measurement data at random. If the system is initialized with a measurement containing fault, the EKF estimate will be flawed. Setting the P matrix values to high levels prevents this scenario. Also, within a few iterations, the P matrix converges to lower values depending upon the estimate and measurement reliability. The matrix is as shown below:

$$P = \begin{bmatrix} 10.0 & 0.0 & 0.0 & \dots & 0.0 \\ 0.0 & 10.0 & 0.0 & \dots & 0.0 \\ 0.0 & 0.0 & 10.0 & \dots & 0.0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0.0 & 0.0 & 0.0 & \dots & 100.0 \end{bmatrix}_{18\times 18} \tag{4.4}$$

The process noise covariance matrix Q is initialized to relatively low values for each of the states. This initialization is done to show that the system trusts the model in use, but does not overly relies on it to produce estimates. The Q matrix is as shown below:

$$Q = \begin{bmatrix} 0.01 & 0.0 & 0.0 & \dots & 0.0 \\ 0.0 & 0.01 & 0.0 & \dots & 0.0 \\ 0.0 & 0.0 & 0.001 & \dots & 0.0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0.0 & 0.0 & 0.0 & \dots & 1.0 \end{bmatrix}_{18\times 18} \tag{4.5}$$

The measurement noise covariance matrix R is initialized with the pre-recorded default values of each sensor. The R matrix is updated as each measurement observation is received with the attached covariance values. This default R matrix is specifically used for Phase 3 (Section 4.3.3), where the faults have not been detected and hence every measurement observation carries their measurement noise covariance values. The default values can only be changed at the place of origin, which is at the sensor or V-REP in this case. The default matrix is shown below:

$$R = \begin{bmatrix} 1.0 & 0.0 & 0.0 & \dots & 0.0 \\ 0.0 & 1.0 & 0.0 & \dots & 0.0 \\ 0.0 & 0.0 & 0.1 & \dots & 0.0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0.0 & 0.0 & 0.0 & \dots & 0.1 \end{bmatrix}_{6\times 6} \tag{4.6}$$

As described in Section 2.4, EKF makes use of all of this information along with the vehicle model to produce state estimates.

On the first iteration, the state vector x_0 is initialized with the first set of measurements received and arbitrary values for the unknown forces. The values (100) in the P matrix that correspond to the force values, ensure that the impact of these arbitrary values is minimized. These values keep improving as a result of the estimation process with each iteration.

The prediction step uses the vehicle model described in Equation 4.2 to predict the states. Next the Jacobian matrix G is calculated using the vehicle model. This Jacobian is used to make prediction for the P matrix.

$$G = \frac{\partial}{\partial(x_t, y_t, \psi_t, \dots, \ddot{F}_{yr_t})^T} \begin{bmatrix} x_t + (v_{x_t} \cos\psi - v_{y_t} \sin\psi)dts \\ y_t + (v_{x_t} \sin\psi + v_{y_t} \cos\psi)dts \\ \psi_t + (\tan\delta \frac{v_{x_t}}{(a+b)})dts \\ \vdots \\ \ddot{Fyr_t} + 0.0dts \end{bmatrix} \tag{4.7}$$

For the correction step, the Kalman gain is calculated by making use of Jacobian H calculated from the measurement model described in Equation 4.3. Along with the Jacobian H, the predicted P matrix and the R matrix are used to update the predictions with the corrections.

$$H = \frac{\partial}{\partial(x_t, y_t, \psi_t, \ldots, \dot{\psi}_t)^T} \begin{bmatrix} x_t \\ y_t \\ \psi_t \\ \vdots \\ \dot{\psi}_t \end{bmatrix} \tag{4.8}$$

Before the corrections are made, it is important to remove any measurements that have been deemed as unusable by the Integrity Checking node. Any observations carrying the ”unusable” tag are not allowed to impact the correction update.

The residual is calculated by subtracting the measurement model measurements from the sensor measurements:

$$Residual = Z - H\bar{X} \tag{4.9}$$

The step to the estimation process is the residual check. The system checks if the obtained residual is greater or less than twice the variance of the measurement parameters. If it is greater or less than twice the variance, then the said measurement is deemed unusable and is not allowed to impact the correction update.

Finally, the state and the process covariance correction update take place based on Equations 2.16 and 2.17. Once the estimates are available, the EKF node publishes out the estimates, which are then subscribed and stored by the Container.

Plotter

The estimates published by the EKF node are also collected by the plotter node, which uses them to create desired plots.

Health Monitor

At each fault critical step, every node publishes out information regarding the fault event in the form of logs. It is then subscribed by the Health Monitor node, which keeps track of each faulty measurement from the point of injection to the point of estimation.

4.3 Test Approach

The goal of the approach evaluation is to assess the impact of the fault tolerance strategy on the state estimation process. Extending the state estimation data path with the associated fault tolerance data path should ideally improve the estimates in the presence of faults in the measurement data stream. Therefore, to test the impact of the fault strategy and the effectiveness of the architecture, the test approach is divided into three phases.

4.3.1 Phase 1: No faults are injected

In this phase, no faults are injected by the FI node, i.e., fault injection is turned off. The noise injector node injects noise. Rest of the process chain works as normal. Since faults are absent, this run can be considered as the ideal or the clean run where no anomaly occurs.

4.3.2 Phase 2: Fault Injection is ON and Integrity Checking is ON

Phase 2 comprises of the run in which every part of the process chain is turned ON. Faults are injected by the FI node. Next the IC node detects the faults and takes appropriate action based on the type of fault detected. As a result, the rest of the process stream is aware of the detected faults. The noise injector node injects noise. The purpose of this run is to produce results that reflect the effectiveness of the complete stream when compared to the Phase 1 and Phase 3.

4.3.3 Phase 3: Fault Injection is ON and Integrity Checking is OFF

Phase 3 runs a scenario in which the faults are being injected in the stream at the FI node. The noise injector node injects noise. However, the rest of the stream is blind to any of these faults. No actions are taken to curb these faults or their effects. EKF stream functions as normal with the default covariances. Since IC is OFF, the covariance matrices are not modified based on the detection of faults.

Chapter 5

Results

The test approach mentioned in Section 4.3 has been iterated on multiple scenarios to produce metrics suitable for comparison. Scenarios mostly comprised of injecting and studying the effects of plausibility and error code faults. The effects of 5 different scenarios are discussed:

- Injecting plausibility faults in GPS data only
- Injecting error code faults in GPS data only
- Injecting plausibility faults in IMU data only
- Injecting error code faults in IMU data only
- Injecting plausibility faults in Camera data only

The improvement due to the additional fault-tolerance components is assessed using the following metrics.

- Fault Injection and Correction (FIC) - Mean absolute percentage error of Phase 2 relative to clean Phase 1
- Fault Injection Only (FIO) - Mean absolute percentage error of Phase 3 relative to clean Phase 1
- Improvement (I) = $FIC - FIO$

5.1 Overall Expectations

In an ideal scenario, the following are the trends that are to be expected.

- GPS sensor: GPS provides position information in the form of x and y coordinates. Therefore, ideally the x and y coordinate estimates should be affected directly in cases where GPS faults are introduced. However, other variables can have indirect impact due the vehicle model dependencies.
- INS sensor: INS provides the orientation information and the yaw rate. Therefore, ideally the yaw angle and the yaw rate estimate should be directly affected in cases where INS faults are introduced. However, other variables can have indirect impact due the vehicle model dependencies.
- Camera sensor: Camera provides the velocity information. Therefore, ideally the lateral and longitudinal velocity estimates should be directly affected in cases where Camera faults are introduced. However, other variables can have indirect impact due the vehicle model dependencies.
- Number of faults: As the number of faults increases in a given time period, the FIO and FIC values should increase. I should overall increase because more faults are being caught.
- Time period: Time period refers to the total number of observations acquired in a given time period. Since the acquisition rate is constant, the larger the time period the larger the observation set and vice-versa. As the time period grows and the number of faults remain the same, the FIO and FIC values should decrease since the mean error should go down. I should remain positive indicating that the fault correction is working as intended. However, it should decrease since the effect of detection will diminish over when averaged over a larger number of observations.
- Plausibility and Error code: With fault correction turned OFF, ideally, the disruption caused due to plausibility faults should be higher than the disruption caused by error code faults. This is because the plausibility faults have the tendency of providing data that is out of bounds of the range or nature of a particular data type. This data over consequtive frames and over a long period of time, produces faulty estimates. However, for the sake of simplicity, the setup and the nature of the plausibility faults avoids extreme scenarios. The purpose of introducing the faults individually is to see if the system is able to tolerate both types of faults.

5.2 Tests

The tests were performed for each of the scenarios in two sets:

- Test 1: Number of faults over a fixed period of time was increased in subsequent iterations

Table 5.1: Test 1.

Number of faults:	10	20	30	40	50	60
Observation set:	100	100	100	100	100	100

- Test 2: Constant number of faults were spread over an increasing number of observations

Table 5.2: Test 2.

Number of faults:	20	20	20	20	20	20
Observation set:	100	200	300	400	500	600

The results are presented in the following sections.

5.3 Graphs and Analysis

The graphs in the following subsections show the plots that reflect the trend of the mean absolute percentage error (FIC and FIO). These plots are generated for each of the primary state estimates that are produced at the end of the EKF process: x position, y position, yaw, x velocity, y velocity, and yaw velocity. The mean absolute percentage error (MAPE) is calculated based on the following equation:

$$\text{MAPE} = \frac{100}{n} \sum_{t=1}^{n} \left| \frac{A_t - F_t}{A_t} \right|, \tag{5.1}$$

where A_t is the actual value obtained from Phase 1 and F_t is the estimated value obtained from Phase 2 or Phase 3.

MAPE helps in analyzing the average error present in the estimation process in the situations when the fault detection and correction is turned ON and when it is turned OFF, with respect to the situation when no faults are injected. This information helps to assess the overall effectiveness of the fault tolerance stream and the estimation process. Another advantage of using MAPE is that it reduces each of the different states to a common mesure value, which helps in making inter-state comparisons, regardless of the type of the state.

The legend represented in graphs in the subsequent sections is as follows:

- $DetectionON = FIC$,
- $DetectionOFF = FIO$, and

One of the primary objectives of the scenarios presented in the following sections was to analyze and understand the spread of MAPE. Running the test scenario iteratively helps in determining the precision of the setup in determining the percentage error and produces conclusive results. Due to the long simulation time and system resources required to run multiple iterations, the iterative test runs were conducted for limited scenarios. The faults injected have been referenced in appendix B

5.3.1 Injecting Plausibility Faults in GPS Data Only

Test 1

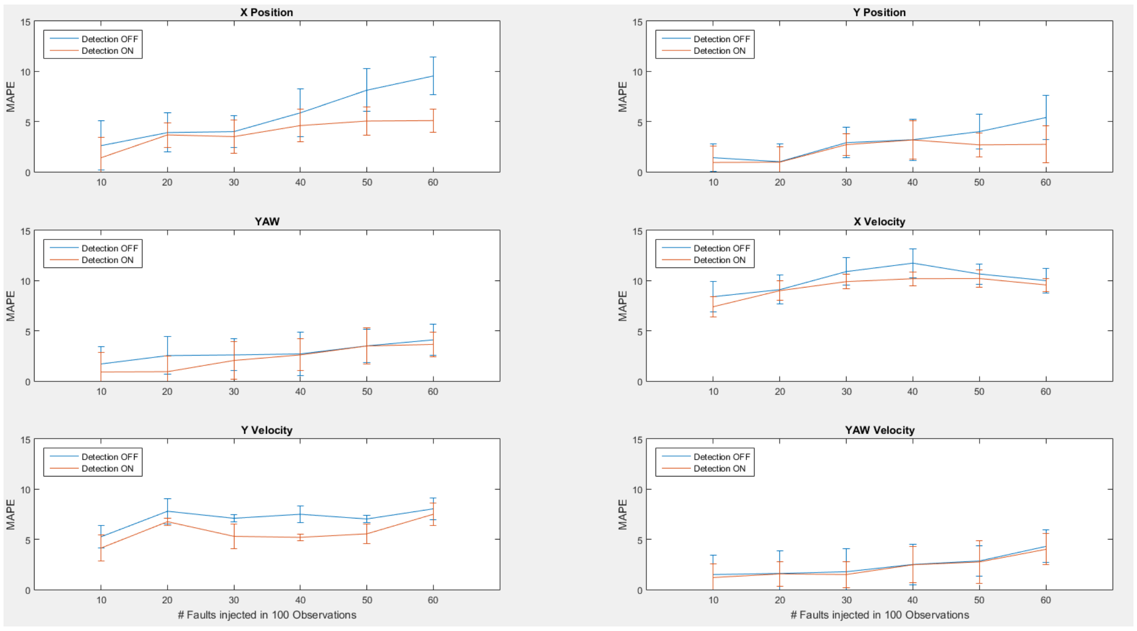


Figure 5.1: Test 1 GPS Plausibility MAPE Plots.

Figure 5.3.1 represents the plots of the estimates of each of primary states in terms of MAPE. These plots were generated for the scenario where only plausibility faults were injected randomly into the GPS measurement data stream. The faults were injected over a fixed time window in subsequent iterations and with each iteration the number of faults was steadily increased. In six iterations, [10,20,30,40,50,60] faults were injected into each window of 100 observations. Finally each of the scenarios was repeated atleast 16 times. The average of 16 runs is plotted along with the standard deviation.

Since the GPS corresponds to the position measurements, x position and the y position are the directly affected estimates. However, other variables like yaw and velocity are also indirectly affected as a result of model dependencies. The correction or the FIC curve, for example the FIC yaw curve, shows that as the faults are mitigated overall, the indirect impact on the secondary variables also reduces.

- One observation from Figure 5.3.1 is that as the number of faults increases in the fixed window, the MAPE increases. This trend can be seen from the plot where fault correction is turned OFF (blue) as well as from where the fault correction is turned ON (red).

- Even though the MAPE increases, the scenarios which have fault correction measures enabled mitigate the faults. This trend can be seen from the lower FIC values (red) as compared to the FIO values (blue) throughout the multiple iterations and across the different states.
- A major observation is that the system effectively estimates the estimation error within an average standard deviation of $\pm 2.91\%$ for the X Position and Y Position curves, and with an overall average standard deviation of $\pm 2.43\%$.
- The plots clearly show that the system effectively mitigates faults injected into the GPS measurement stream whith an average MAPE difference between FIC and FIO of 2.33%, while mitigating the indirect impacts.

Another observation that can be made is that in certain plots, like the YAW Velocity plot, there are fluctuations in the estimations error, where the FIC plots MAPE value is closer or a little higher than the FIO plot MAPE value. At low velocity, certain states have a very low value, almost close to zero. Therefore, a small negative or positive value for the lateral velocity, occurring due to process noise or measurement noise, can create fluctuations in the trend. Similar observations are seen in a few upcoming scenarios.

Test 2

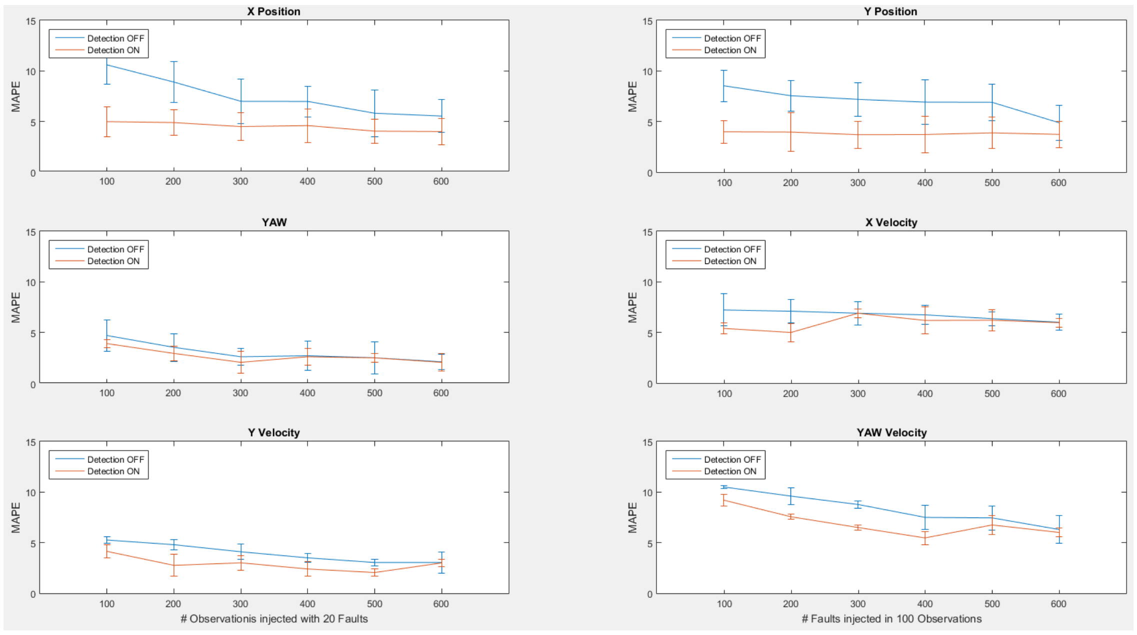


Figure 5.2: Test 2 GPS Plausibility MAPE Plots.

Figure 5.3.1 represents the plots of the estimates of each of primary states in terms of MAPE. These plots were generated for the scenario where only plausibility faults were injected into the GPS measurement data stream. In this scenario a fixed number of GPS plausibility faults were injected over an increasing time window in subsequent iterations. The time window length was increased steadily with each iteration. In an increasing time-window of size [100,200,300,400,500,600], 20 faults were injected in each window. Finally each of the scenarios was repeated atleast 16 times. The average of 16 runs is plotted along with the standard deviation.

Since the GPS corresponds to the position measurements, x position and the y position are the primary affected estimates as a result of the fault injections. However, other variables are also indirectly affected as a result of model dependencies as reflected in the graphs.

- One observation from Figure 5.3.1 is that as the time window increases with the number of faults remaining constant, the MAPE tends to decrease. This trend can be seen from the plot where fault correction is turned OFF (blue) as well as from where the fault correction is turned ON (red).

- Even though the MAPE decreases, the scenarios which have fault correction measures enabled mitigate the faults. This trend can be seen from the lower FIC values (red) as compared to the FIO values (blue) throughout the multiple iterations and across the different states.
- A major observation is that the system effectively estimates the estimation error within an average standard deviation of ±3.01% for the X Position and Y Position curves, and with an overall average standard deviation of ±2.51%.
- The plots clearly show that the system effectively mitigates faults injected into the GPS measurement stream whith an average MAPE difference between FIC and FIO of 2.84%, while mitigating the indirect impacts.

5.3.2 Injecting Error Code Faults in GPS Data Only

Test 1

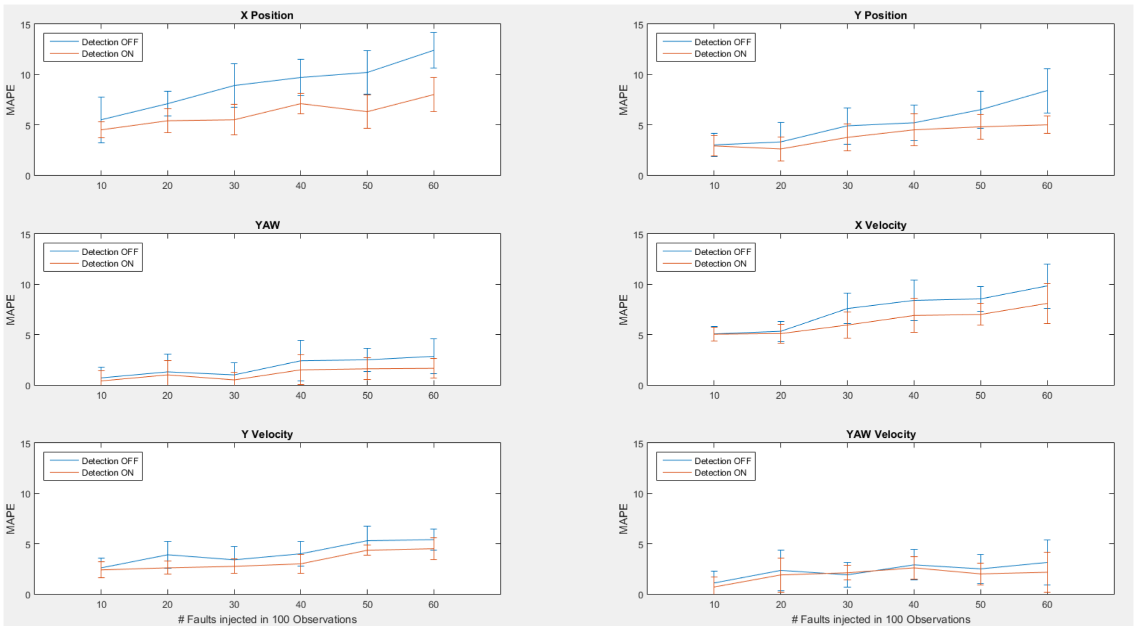


Figure 5.3: Test 1 GPS Error Code MAPE Plots.

Similar to Section 5.3.1, for this scenario error code faults were injected into the GPS measurement data stream. The aim is to test the effect of error code faults on the state estimation process and also to observe the effectiveness of the fault tolerant architecture in mitigating these faults. These faults were injected over a fixed time window in subsequent iterations and with each iteration the number of faults was steadily increased. In six iterations, [10,20,30,40,50,60] faults were injected into 100 observations. Finally each of the scenarios was repeated atleast 16 times. The average of 16 runs is plotted along with the standard deviation.

Since the GPS corresponds to the position measurements, x position and the y position are the primary affected estimates as a result of the fault injections. However, other variables are also indirectly affected as a result of model dependencies as reflected in the graphs. Even though the MAPE values might differ, trends similar to Section 5.3.1 were observed:

- One observation from Figure 5.3.2 is that as the number of faults increases in the fixed window, the MAPE increases. This trend can be seen from the plot where fault correction is turned OFF (blue) as well as from where the fault correction is turned ON (red).
- Even though the MAPE increases, the scenarios which have fault correction measures enabled mitigate the faults. This can be seen from the lower FIC values (red) as compared to the FIO values (blue) throughout the multiple iterations and across the different states.
- A major observation is that the system effectively estimates the estimation error within an average standard deviation of ±2.39% for the X Position and Y Position curves, and with an overall average standard deviation of ±2.03%.
- The plots clearly show that the system effectively mitigates faults injected into the GPS measurement stream whith an average MAPE difference between FIC and FIO of 2.31%, while mitigating the indirect impacts.

It can be observed from the error bars that sometimes the estimation error is relatively high. One of the reasons behind this occurrence is the consecutive nature of the faults. Since the faults are injected randomly, the faults can stack back to back. When faulst are encountered consecutively over a long observation window, it becomes difficult for the system to compensate for the repeated faulty measurements, thus leading to increase in estimation error.

Even though the x and y curves show fault correction, the improvement is significantly low at some instances. This can be attributed to the low covariance measures set by the integrity checker node. Setting relatively higher covariance values for the particular error code should yield better improvement.

Test 2

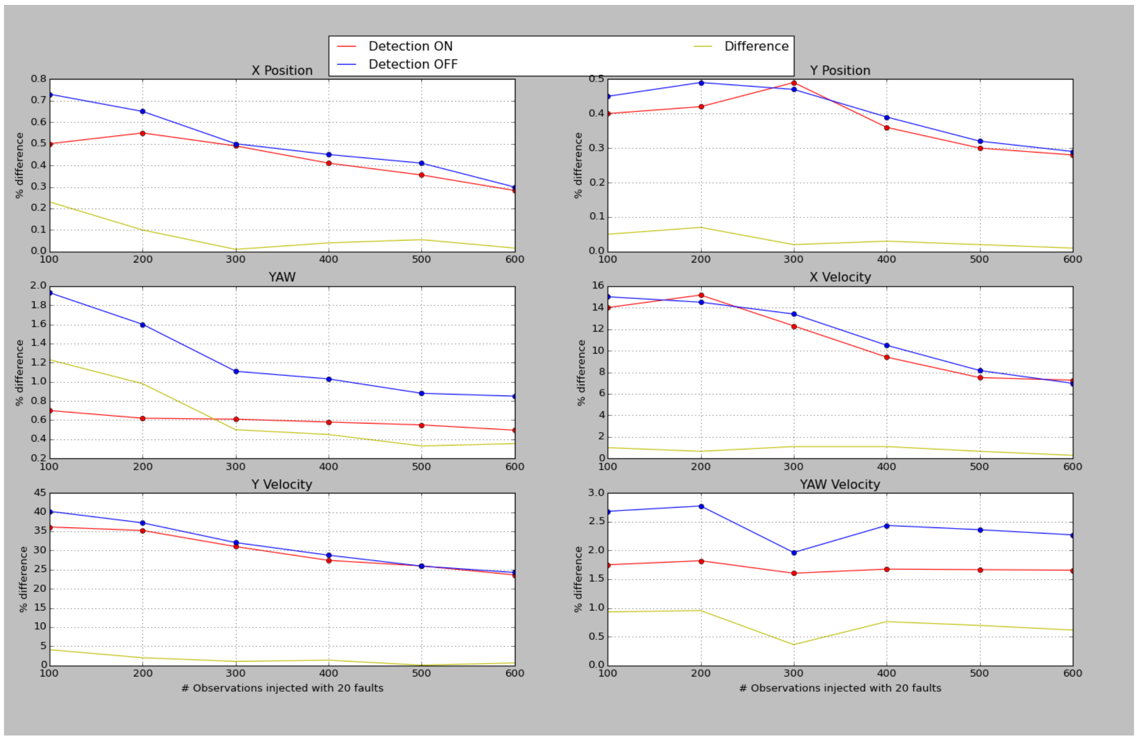


Figure 5.4: Test 2 GPS Error Code MAPE Plots.

Similar to Section 5.3.1, for this scenario error code faults were injected into the GPS measurement data stream. In this scenario a fixed number of GPS error code faults were injected over an increasing time window in subsequent iterations. The time window length was increased steadily with each iteration. In an increasing time-window of size [100,200,300,400,500,600], 20 faults were injected. This run is only a singular iteration of the test. Therefore, it is not conclusive or representative of the actual result.

Since the GPS corresponds to the position measurements, x position and the y position are the primary affected estimates as a result of the fault injections. However, other variables like yaw are also indirectly affected as a result of model dependencies as reflected in the graphs. Even though the MAPE values might differ, trends similar to Section 5.3.1 were observed:

- One observation from Figure 5.3.2 is that as the time window increases with the number of faults remaining constant, the MAPE tends to decrease. This trend can be seen from the plot where fault correction is turned OFF (blue) as well as from where the fault correction is turned ON (red).

- Even though the MAPE decreases, the scenarios which have fault correction measures enabled mitigate the faults. This trend can be seen from the lower FIC values (red) as compared to the FIO values (blue) throughout the multiple iterations and across the different states.

5.3.3 Injecting Plausibility Faults in IMU Data Only

Test 1

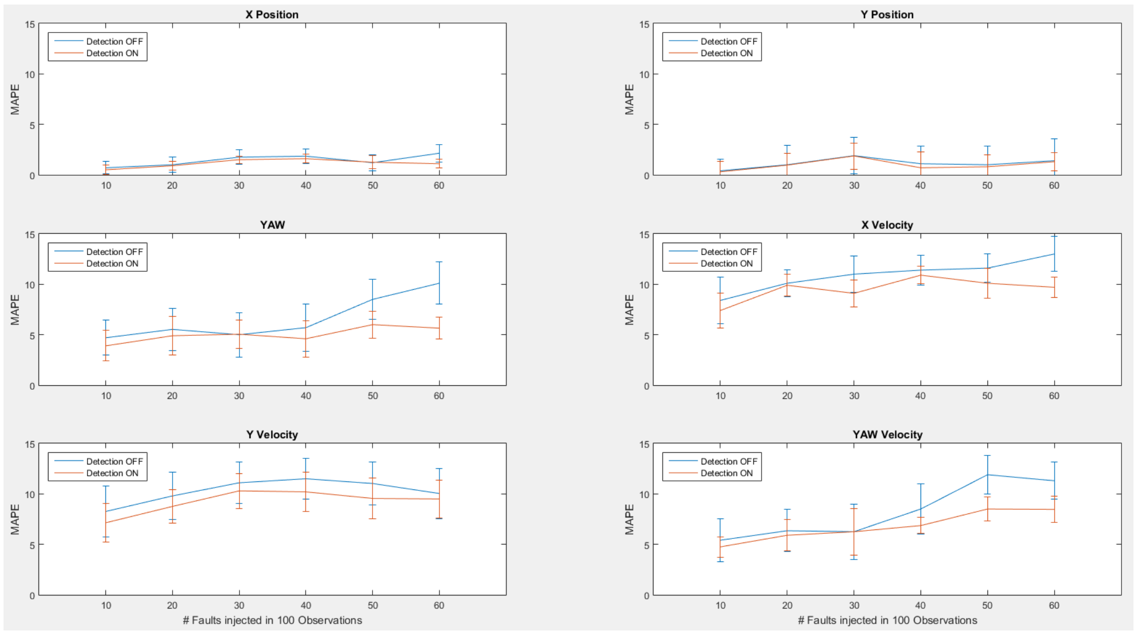


Figure 5.5: Test 1 IMU Plausibility MAPE Plots.

IMU data primarily affects the orientation measurements. Thus, plaguing the IMU measurement data with any type of faults is tend to cause anomalies in the orientation state estimates, i.e. the yaw angle and yaw rate. As a result of the vehicle model, other model equations like the velocity depend upon orientation information. Therefore, IMU faults would indirectly impact other state estimates as well. The above figure represents the plots of the estimates of each state in terms of MAPE. These faults were injected over a fixed time window in subsequent iterations and with each iteration the number of faults was steadily increased. In six iterations, [10,20,30,40,50,60] faults were injected into 100 observations. Finally each of the scenarios was repeated atleast 16 times. The average of 16 runs is plotted along with the standard deviation.

- One observation from Figure 5.3.3 is that as the number of faults increases in the fixed window, the MAPE increases. This trend can be seen from the plot where fault correction is turned OFF (blue) as well as from where the fault correction is turned ON (red).
- Even though the MAPE increases, the scenarios which have fault correction measures enabled mitigate the faults. This trend can be seen from the lower FIC values (red) as compared to the FIO values (blue) throughout the multiple iterations and across the different states.
- Another observation is that the average Improvement index (yellow) per state stays positive across the different states reflecting the improvement brought by the implementation of the fault tolerance architecture.
- A major observation is that the system effectively estimates the estimation error within an average standard deviation of ±2.62% for the YAW and YAW Velocity curves, and with an overall average standard deviation of ±2.37%.
- It can also be seen that the X Velocity and Y Velocity is also affected indirectly with an average difference of 1.81% between the MAPE FIC and FIO curves.
- The plots clearly show that the system effectively mitigates faults injected into the IMU measurement stream whith an average MAPE difference between FIC and FIO of 1.96%, while mitigating the indirect impacts.

It can be observed that the MAPE spread is almost condensed for the X Position and Y Position plots. However, for the YAW and the velocity plots, the MAPE spread is a little wide with the standard deviation averaging to about ±2.62%. This outcome is because of the nature of the faults injected. On an average, out of all the runs, 92.43% faults were detected successfully. Some data discontinuity faults are hard to detect given the low threshold of discontinuity parameter. Any values close yet above to the threshold slope limit are flagged by the IC, leading to the possibility of misclassification. Also, successive discontinuous data along with the misclassified data, increases the estimation error, thus increasing the MAPE value.

Test 2

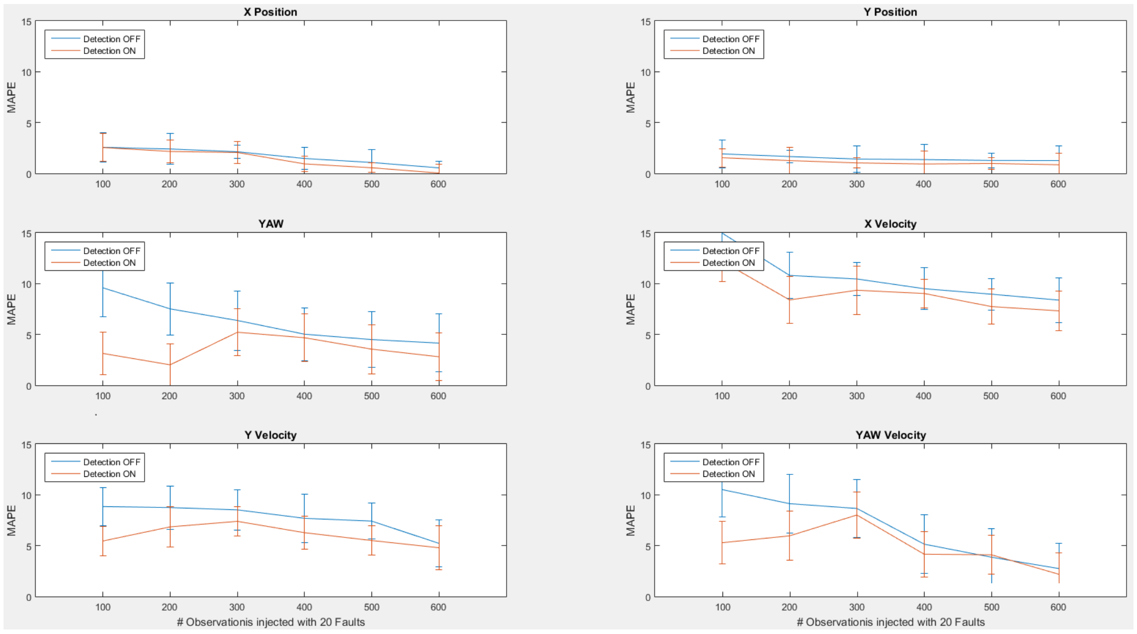


Figure 5.6: Test 2 IMU Plausibility MAPE Plots.

IMU data primarily affects the orientation measurements. Thus, plaguing the IMU measurement data with any type of faults is tend to cause anomalies in the orientation state estimates, i.e. the yaw angle and yaw rate. As a result of the vehicle model, other model equations like the velocity depend upon orientation information. Therefore, IMU faults would indirectly impact other state estimates as well. The above figure represents the plots of the estimates of each of primary states in terms of MAPE. The time window length was increased steadily with each iteration. In an increasing time-window of size [100,200,300,400,500,600], 20 faults were injected. Finally each of the scenarios was repeated atleast 16 times. The average of 16 runs is plotted along with the standard deviation.

- One observation from Figure 5.3.3 is that as the time window increases with the number of faults remaining constant, the MAPE tends to decrease. This trend can be seen from the plot where fault correction is turned OFF (blue) as well as from where the fault correction is turned ON (red).
- Even though the MAPE decreases, the scenarios which have fault correction measures enabled mitigate the faults. This trend can be seen from the lower FIC values (red) as com-

pared to the FIO values (blue) throughout the multiple iterations and across the different states.

- A major observation is that the system effectively estimates the estimation error within an average standard deviation of ±3.17% for the YAW and YAW Velocity curves, and with an overall average standard deviation of ±2.07%.
- It can also be seen that the X Velocity and Y Velocity is also affected indirectly with an average difference of 2.21% between the MAPE FIC and FIO curves.
- The plots clearly show that the system effectively mitigates faults injected into the IMU measurement stream whith an average MAPE difference between FIC and FIO of 2.56%, while mitigating the indirect impacts.

5.3.4 Injecting Error Code Faults in IMU Data Only

Test 1

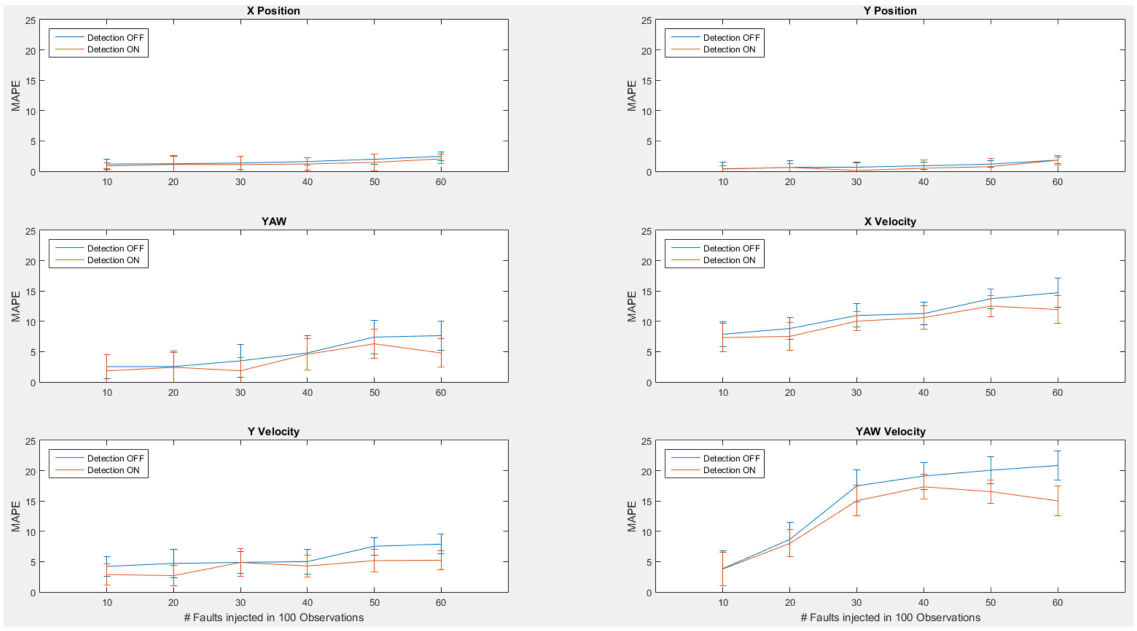


Figure 5.7: Test 1 IMU Error Code MAPE Plots.

Similar to Section 5.3.3, instead of plausibility faults, different error code faults were injected in the IMU measurement stream. This test was conducted to observe the effects of IMU error code faults on the estimation process and to observe the effectiveness of the fault tolerance architecture. The trend of the results obtained are mostly consistent with the results of section 5.3.3 In six iterations, [10,20,30,40,50,60] faults were injected into 100 observations. Finally each of the scenarios was repeated atleast 16 times. The average of 16 runs is plotted along with the standard deviation.

- One observation from Figure 5.3.4 is that as the number of faults increases in the fixed window, the MAPE increases. This trend can be seen from the plot where fault correction is turned OFF (blue) as well as from where the fault correction is turned ON (red).
- Even though the MAPE increases, the scenarios which have fault correction measures enabled mitigate the faults. This trend can be seen from the lower FIC values (red) as compared to the FIO values (blue) throughout the multiple iterations and across the different states.
- A major observation is that the system effectively estimates the estimation error within an average standard deviation of $\pm 2.17\%$ for the YAW and YAW Velocity curves, and with an overall average standard deviation of $\pm 1.62\%$.
- It can also be seen that the X Velocity and Y Velocity is also affected indirectly with an average difference of 2.05% between the MAPE FIC and FIO curves.
- The plots clearly show that the system effectively mitigates faults injected into the IMU measurement stream whith an average MAPE difference between FIC and FIO of 2.29%, while mitigating the indirect impacts.

A minor yet distinguishable change in trend is visible in these plots. The Y velocity MAPE FIO curve (blue) has a slightly negative slope from 10 to 30 faults, which turns positive subsequently. One of the reasons that can partially influence this behavior is the vehicle model. The vehicle model velocity equations depend upon orientation rate and anomalies in the orientation rate can cause indirect fluctuations in the velocity estimates. There is also the fact that when traveling in a straight line, the lateral velocity of the vehilce is close to zero. Thus process and measurement noise can cause dips in the Y Velocity. Regardless of this behavior, it is quite evident from the FIC curve (red) that the estimation process combined with fault tolerance reduces the impact of the faults.

Test 2

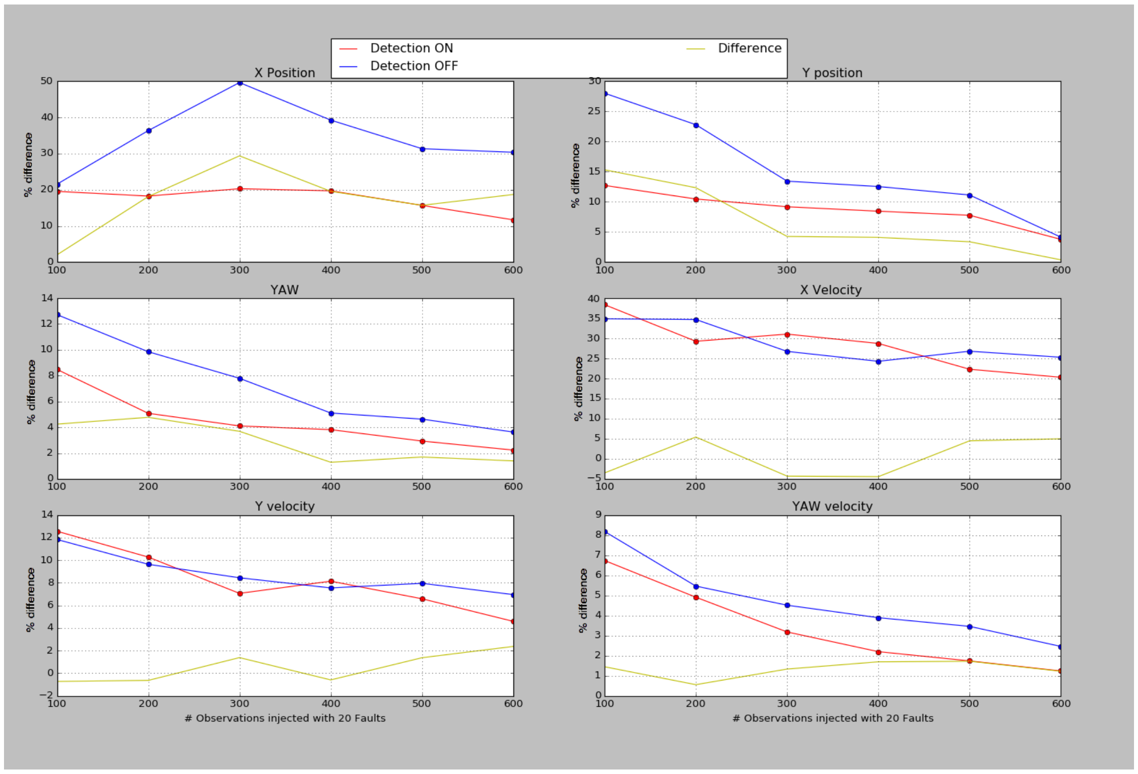


Figure 5.8: Test 2 IMU Error Code MAPE Plots.

Similar to Section 5.3.3, instead of plausibility faults, different error code faults were injected in the IMU measurement stream. This test was conducted to observe the effects of IMU error code faults on the estimation process and to observe the effectiveness of the fault tolerance architecture. The trend of the results obtained are mostly consistent with the results of section 5.3.3 The time window length was increased steadily with each iteration. In an increasing time-window of size [100,200,300,400,500,600], 20 faults were injected in multiple iterations. This run is only a singular iteration of the test. Therefore, it is not conclusive or representative of the actual result.

- One observation from Figure 5.3.4 is that as the number of faults increases in the fixed window, the MAPE increases. This rend can be seen from the plot where fault correction is turned OFF (blue) as well as from where the fault correction is turned ON (red).
- Even though the MAPE increases, the scenarios which have fault correction measures enabled mitigate the faults. This trend can be seen from the lower FIC values (red) as com-

pared to the FIO values (blue) throughout the multiple iterations and across the different states.

Another indirect impact of the IMU faults which was discussed in Section 5.3.4 is witnessed in Figure 5.3.4. The X position curve rises steeply till the 300-observation iteration and then falls. There are also abrupt and unexpected rise and falls in the X and Y velocity curves. Again, the vehicle model can partially impact this nature as these state equations can be affected by the abnormal changes in the orientation information. Once again, the fault tolerant run mitigates the effect of the injected faults and performs relatively well. However, it can be seen from the X velocity curve that the fault tolerance was less effective. This anomaly can be attributed to the magnitude of the effect caused by the injected fault.

5.3.5 Injecting Plausibility Faults in Camera Data Only

Since camera provides images rather than measurement information in raw format, the impact of any anomalies or faults caused due to degradation is relatively hard to judge. Images are first processed by visual odometry to obtain information about the pose or velocity of the vehicle. Due to time contraints, only plausibility faults results are presented in this section.

Test 1

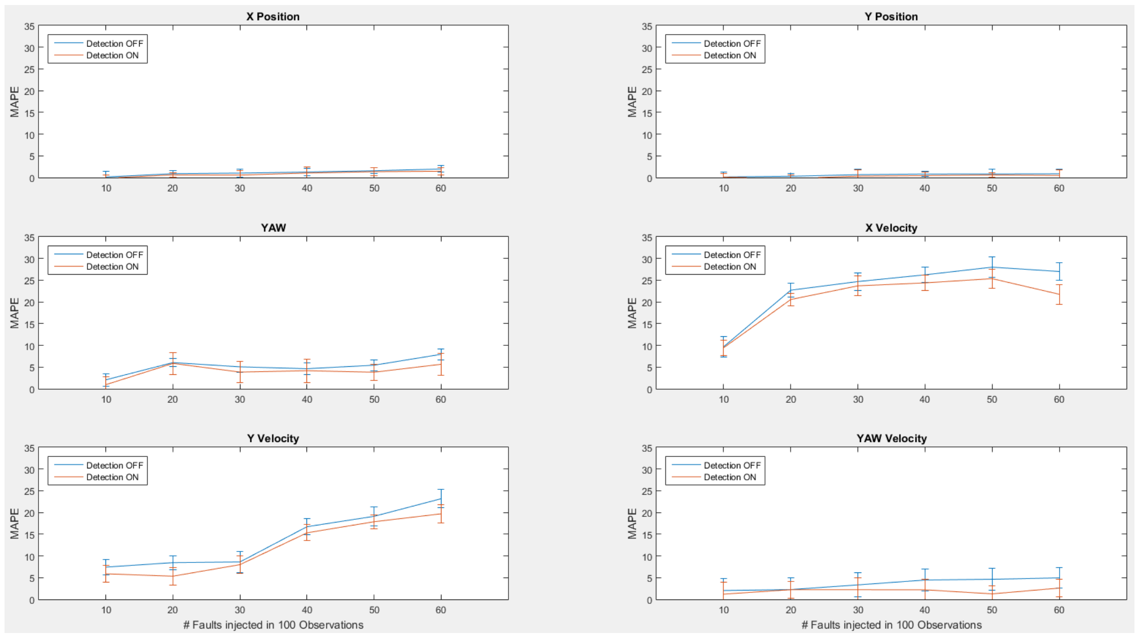


Figure 5.9: Test 1 Camera Plausibility MAPE Plots.

Camera plausibility faults were injected in this scenario. Fault injected forced the pixel RGB values to fall out of 0-255 range or fixed a set of pixels with the same RGB value. Different number of faults were injected in subsequent iterations. However, other variables are also indirectly affected as a result of model dependencies as reflected in the graphs. In six iterations, [10,20,30,40,50,60] faults were injected into each window of 100 observations. Finally each of the scenarios was repeated atleast 16 times. The average of 16 runs is plotted along with the standard deviation.

Camera and VO data primarily affects the velocity measurements. Thus, plaguing the images with any type of faults is tend to cause anomalies in the velocity state estimates, i.e., the lateral and longitudinal velocities. As a result of the vehicle model, other variables get affected.

- One observation from Figure 5.3.5 is that as the number of faults increases in the fixed window, the MAPE increases. This trend can be seen from the plot where fault correction is turned OFF (blue) as well as from where the fault correction is turned ON (red).
- Even though the MAPE increases, the scenarios which have fault correction measures enabled slightly mitigate the faults. This trend can be seen from the lower FIC values

(red) as compared to the FIO values (blue) throughout the multiple iterations and across the different states.

- A major observation is that the system effectively estimates the estimation error within an average standard deviation of ±1.57% for the X Velocity and Y Velocity curves, and with an overall average standard deviation of ±1.12%.
- The plots clearly show that the system effectively mitigates faults injected into the camera measurement stream whith an average MAPE difference between FIC and FIO of 1.43%, while mitigating the indirect impacts.

Test 2

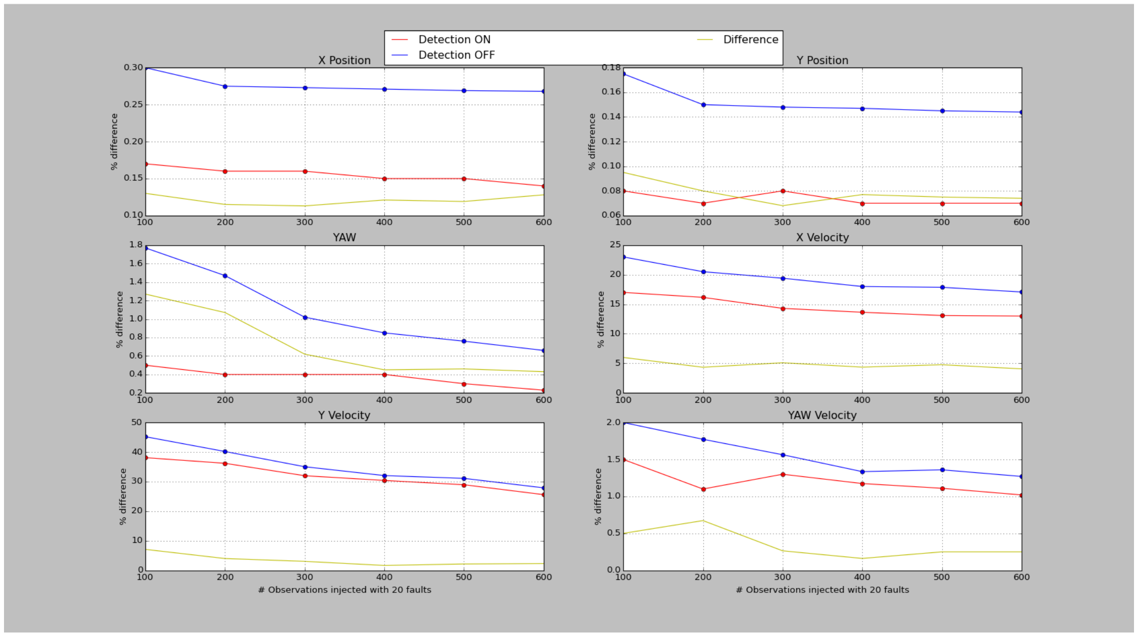


Figure 5.10: Test 2 Camera Plausibility MAPE Plots.

In an increasing time-window of size [100,200,300,400,500,600], 20 faults were injected in multiple iterations. This run is only a singular iteration of the test. Therefore, it is not conclusive or representative of the actual result.

- One observation from Figure 5.3.5 is that as the number of faults increases in the fixed window, the MAPE increases. This trend can be seen from the plot where fault correction is turned OFF (blue) as well as from where the fault correction is turned ON (red).

- Even though the MAPE increases, the scenarios which have fault correction measures enabled try to mitigate the faults. This trend can be seen from the lower FIC values (red) as compared to the FIO values (blue) throughout the multiple iterations and across the different states.
- Another observation is that the average Improvement index (yellow) per state stays positive across the different states reflecting the improvement brought by the implementation of the fault tolerance architecture.

It is interesting to note that the x and y position curve slopes remain close to zero and consistent.

5.4 Result vs Expectations

The tests were conducted with the aim to check the effectiveness of the fault tolerance stream over the state estimation process. The iterative analysis conducted provides conclusive results of the effectiveness of the complete setup. The expectations laid out in Section 5.1 were made in order to guide the test layout. The results obtained for each of the tests are very promising. They indicate that in all the scenarios, having fault tolerance turned ON provided better state estimates than having fault tolerance turned OFF. All the expectations laid out in Section 5.1 were realized. As an effect of introducing the faults in the data stream of individual sensors, the results show that the primary state variables x and y for GPS, yaw and yaw rate for INS, and lateral and longitudinal velocity for camera are directly impacted. Also, as per expectation, for each test run, non-primary variables were indirectly impacted. However, in some scenarios the indirect impact on the non-primary variables is higher or close to the direct impact on the primary variables. This can be attributed to the vehicle model inaccuracy and state variable covariance inaccuracies. As per the vehicle model, the state variables are not independent. They depend on prior knowledge of other state variables. Hence, any impact on one state variable is reflected on the dependent state variables. Static process noise covariances can also be a reason behind the different anomalies that occurred in Section 5.3. Having the same confidence over the vehicle model equation output, when it is known that the prior state estimate was affected with faults, can cause fault to propagate through future iterations. The improvement, which is the difference between FIC and FIO MAPE curves remained positive for all the scenarios. The effect of steadily increasing the time period and the number of faults is also as expected. As the time period rose, with the number of faults remaining the same, the FIO and FIC values decreased. Similarly, as the number of faults rose, with the time period remaining the same, the FIO and FIC values increased.

Chapter 6

Conclusion & Future Work

This thesis explored the process of state estimation boosted by fault tolerance by the means of development and implementation of a unified architecture. The contribution of the developed prototype lies in validation of the fault tolerant architecture and the use of visual odometry as an alternate to conventional sensors, for the production of improved state estimates in the presence of faults. The complete architecture can be split into two primary functions, fault tolerance and state estimation.

The components involved in the fault tolerance architecture have performed promisingly. The methods employed by the integrity check successfully detected and flagged the presence of faults. As a result, the estimation process was able to recognize faults with the help of the meta data information appended to each measurement message. Different methods were employed by the integrity checker. Frequency checks, plausibility checks, and error code checks performed really well in identifying the faults. Time window check employed at the container level helped in identifying any persistent faults. Adaptive measurement noise covariance adjustment used by the EKF helped in curbing the impact of the faulty measurements. The residual check employed at the EKF level minimized the impact of any unusable data or any erroneous measurement prediction. However, the methods employed in the fault tolerance stream are not exhaustive. Based on the literature review of the different architectures, a number of different approaches can be amalgamated into the integrity checker and fault tolerance stream as a whole to improve the detection and correction of the faults. For example, adding in software and hardware redundancy can help the system in eliminating faults rather than mitigating their impact.

On the state estimation side, the visual odometry was successfully used as a source of measurement input to the estimation process. This visual odometry setup in the estimation architecture can be extended to test and employ more advanced visual odometry methods including hybrid algorithms based on a combination feature based and appearance-based methods. VO usage for the localization of a vehicle was performed at low speed, which lays ground for the future research on implementation and testing of different VO algorithms at low as well as high speeds. The 3- DoF dynamic vehicle model employed by the EKF made acceptable predictions of the state variables. It represented the simulated vehicle with reasonable reliability in view of the available inputs. However, this model is quite limited in its scope. With the availability of measurements like wheel speed and torque inputs, and constants like the longitudinal and lateral tire stiffness, damping and spring coefficients, more advanced high-fidelity vehicle models can be tested out to obtain better state predictions. EKF is the biggest part of the state estimation process. EKF is considered to be an optimal estimator. EKF accounted for the different faults in the measurement inputs and provided the best possible results with the available information. However, a few anomalies in the state estimates still occurred, which deviated from the expected outcome. These anomalies are not EKFs fault and can be attributed to the incomplete or inaccurate information provided to the EKF, as well as to the vehicle model inaccuracy. Regardless, it has been proven that particle filters can provide better results than an EKF. As a future improvement, the particle filter efficiency can be tested and implemented in place of an EKF.

Overall the complete architecture formed from the amalgamation of the fault tolerance stream and the state estimation process was successful. A number of tests focused on testing the effects of the injected faults were conducted on the architecture implementation to test its validity. It is quite evident from the results obtained in Section 5.3 that the fault tolerant architecture is able to reduce the effect of the faults injected into the measurement stream. Empirically speaking, with the exception of a few incidents, the scenarios with fault tolerance turned ON have produced lower error percentages throughout the different state estimates. The improvements have been significant in some cases and insignificant in others, which makes the implementation of the fault tolerance stream seem questionable. However, it also lays the ground for improvement by implementation and testing of more robust methods. The experimental setup employed to test the prototype is quite reliable in terms of theory and simulation. Simulated data including the input measurements, faults, and environment has been used to test the fault tolerant estimation system. However, the outcome for the real world might be different.

In conclusion, despite of the various limitations at the simulation level, the proposed architecture looks very promising as an alternative to the different architectures that have already been explored and provides diverse approach of obtaining state estimation with fault tolerance.

References

[1] Creating a ros msg and srv. http://wiki.ros.org/ROS/Tutorials/CreatingMsgAndSrv#Creating_a_msg. Accessed: 2017-09-30.

[2] Gps error codes. http://www.roborealm.com/help/XIMEA_Camera.php. Accessed: 2017-05-30.

[3] Imu error codes. http://x-io.co.uk/downloads/x-IMU-User-Manual-v5.2.pdf. Accessed: 2017-05-30.

[4] M. K. Aripin, Yahaya Md Sam, Kumeresan A. Danapalasingam, Kemao Peng, N. Hamzah, and M. F. Ismail. A review of active yaw control system for vehicle handling and stability enhancement. *International Journal of Vehicular Technology*, 2014, 2014.

[5] Algirdas Avizienis, Jean-Claude Laprie, Brian Randell, and Carl Landwehr. Basic concepts and taxonomy of dependable and secure computing. *IEEE Trans. Dependable Secur. Comput.*, 1(1):11–33, January 2004.

[6] Kaci Bader, Benjamin Lussier, and Walter Schön. A fault tolerant architecture for data fusion: A real application of kalman filters for mobile robot localization. *Robotics and Autonomous Systems*, 88:11–23, 2017.

[7] Mnica F. Bugallo, Shanshan Xu, and Petar M. Djuri. Performance comparison of ekf and particle filtering methods for maneuvering targets. *Digital Signal Processing*, 17(4):774 – 786, 2007.

[8] C. Chuanqi, H. Xiangyang, Z. Zhenjie, and Z. Mandan. Monocular visual odometry based on optical flow and feature matching. In *2017 29th Chinese Control And Decision Conference (CCDC)*, pages 4554–4558, May 2017.

[9] Vehicle Dynamics Standards Committee. Society of automotive engineers(sae), vehicle dynamics terminology. 2008.

[10] Weiwen Deng and Haicen Zhang. Rls-based online estimation on vehicle linear sideslip. In *2006 American Control Conference*, pages 6 pp.–, June 2006.

[11] Pedro Domingos. A few useful things to know about machine learning. *Commun. ACM*, 55(10):78–87, October 2012.

[12] Elena Dubrova. Introduction. pages 1–2, 08 2013.

[13] J. Engel, J. Sturm, and D. Cremers. Semi-dense visual odometry for a monocular camera. In *2013 IEEE International Conference on Computer Vision*, pages 1449–1456, Dec 2013.

[14] P. J. Escamilla-Ambrosio and N. Mort. A hybrid kalman filter-fuzzy logic architecture for multisensor data fusion. In *Intelligent Control, 2001. (ISIC '01). Proceedings of the 2001 IEEE International Symposium on*, pages 364–369, 2001.

[15] R. Fitzgerald. Divergence of the kalman filter. *IEEE Transactions on Automatic Control*, 16(6):736–747, Dec 1971.

[16] Z. Fu, Y. Quo, Z. Lin, and W. An. Fsvo: Semi-direct monocular visual odometry using fixed maps. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 2553–2557, Sept 2017.

[17] Andreas Geiger, Julius Ziegler, and Christoph Stiller. Stereoscan: Dense 3d reconstruction in real-time. In *Intelligent Vehicles Symposium (IV)*, 2011.

[18] B. Kaci, L. Benjamin, and S. Walter. A fault tolerant architecture for data fusion targeting hardware and software faults. In *2014 IEEE 20th Pacific Rim International Symposium on Dependable Computing*, pages 1–10, Nov 2014.

[19] Muhammad Altamash Khan. Specifications and strategies for state estimation of vehicle and platoon. Master's thesis, KTH, Signal Processing, 2011.

[20] J. C. Laprie. *Dependability: Basic Concepts and Terminology*, pages 3–245. Springer Vienna, Vienna, 1992.

[21] B.D. M. *Advances in Genetic Statistics: Law of Hardy Weinberg Equilibrium Revisited.* Ebooks2go Incorporated, 2017.

[22] R. Mehra. On the identification of variances and adaptive kalman filtering. *IEEE Transactions on Automatic Control*, 15(2):175–184, April 1970.

[23] R. Mehra. Approaches to adaptive filtering. *IEEE Transactions on Automatic Control*, 17(5):693–698, Oct 1972.

[24] M. J. Milford and G. F. Wyeth. Single camera vision-only slam on a suburban road network. In *2008 IEEE International Conference on Robotics and Automation*, pages 3684–3689, May 2008.

[25] E. Mouragnon, M. Lhuillier, M. Dhome, F. Dekeyser, and P. Sayd. Real time localization and 3d reconstruction. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 1, pages 363–370, 2006.

[26] D. Nister, O. Naroditsky, and J. Bergen. Visual odometry. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 1, pages I–652–I–659 Vol.1, June 2004.

[27] Philip Polack, Florent Altché, Brigitte D'Andréa-Novel, and Arnaud De La Fortelle. The Kinematic Bicycle Model: a Consistent Model for Planning Feasible Trajectories for Autonomous Vehicles? In *IEEE Intelligent Vehicles Symposium (IV)* , Los Angeles, United States, June 2017.

[28] L. R. Ray. Nonlinear state and tire force estimation for advanced vehicle control. *IEEE Transactions on Control Systems Technology*, 3(1):117–124, Mar 1995.

[29] S. Rezaei and R. Sengupta. Kalman filter-based integration of dgps and vehicle sensors for localization. *IEEE Transactions on Control Systems Technology*, 15(6):1080–1088, Nov 2007.

[30] D. Scaramuzza and F. Fraundorfer. Visual odometry [tutorial]. *IEEE Robotics Automation Magazine*, 18(4):80–92, Dec 2011.

[31] D. Scaramuzza and R. Siegwart. Appearance-guided monocular omnidirectional visual odometry for outdoor ground vehicles. *IEEE Transactions on Robotics*, 24(5):1015–1026, Oct 2008.

[32] J. D. Setiawan, M. Safarudin, and A. Singh. Modeling, simulation and validation of 14 dof full vehicle model. In *International Conference on Instrumentation, Communication, Information Technology, and Biomedical Engineering 2009*, pages 1–6, Nov 2009.

[33] S. Shalev-Shwartz and S. Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Understanding Machine Learning: From Theory to Algorithms. Cambridge University Press, 2014.

[34] J. Shen, Y. Liu, S. Wang, and Z. Sun. Evaluation of unscented kalman filter and extended kalman filter for radar tracking data filtering. In *2014 European Modelling Symposium*, pages 190–194, Oct 2014.

[35] Jonny Carlos da Silva, Abhinav Saxena, Edward Balaban, and Kai Goebel. A knowledge-based system approach for sensor fault modeling, detection and mitigation. *Expert Syst. Appl.*, 39(12):10977–10989, September 2012.

[36] G.L. Smith, S.F. Schmidt, L.A. McGee, United States. National Aeronautics, and Space Administration. *Application of Statistical Filter Theory to the Optimal Estimation of Position and Velocity on Board a Circumlunar Vehicle*. NASA technical report. National Aeronautics and Space Administration, 1962.

[37] Sirui Song, Michael Chi Kam Chun, Jan Paul Huissoon, and Steven Lake Waslander. Pneumatic trail based slip angle observer with dugoff tire model. *2014 IEEE Intelligent Vehicles Symposium Proceedings*, pages 1127–1132, 2014.

[38] Martin Swaczyna. Several examples of nonholonomic mechanical systems. *Communications in Mathematics*, 19(1):27–56, 2011.

[39] King Tin Leung, James Whidborne, David Purdy, and Phil Barber. Road vehicle state estimation using low-cost gps/ins. 25:1988–2004, 08 2011.

[40] Stephan Weiss, Markus W. Achtelik, Simon Lynen, Michael C. Achtelik, Laurent Kneip, Margarita Chli, and Roland Siegwart. Monocular vision for long-term micro aerial vehicle state estimation: A compendium. *Journal of Field Robotics*, 30(5):803–831, 2013.

[41] Stephan Wirth. Ros wrapper for libvisio2. http://wiki.ros.org/viso2?distro=indigo. Accessed: 2017-09-30.

[42] Changjiang Yang, Ramani Duraiswami, and Larry Davis. Fast multiple object tracking via a hierarchical particle filter. In *Proceedings of the Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1 - Volume 01*, ICCV '05, pages 212–219, Washington, DC, USA, 2005. IEEE Computer Society.

APPENDICES

Appendix A

Message Format of Custom ROS Measurement Messages

The custom messages were created to facilitate the detection, tracking and correction of the faults that were injected into the system. These messages were created in the form of a ".msg" file and are based on the guidelines presented in [1]

A.1 Message Definition

Even though independent custom messages for the measurement data from each sensor were created, they all followed a generic format. The format is as follows:

```
uint32 seq          % Senquence number of the measurement received
Header header          % Header information of the source sensor
int32 GUID          % ID of the sensor
string state          % Health state of the message
sensor_msgs/*Sensor*_Data           % Actual data
float64[36] covmat          % Covariance of the data
```

Appendix B

Injected Faults

B.1 GPS Error Codes

The error codes were referenced from [2]

- *Error Code = 1: $TEMP\_STATUS$*
  - Data becomes noisy because the temperature is too high
  - Noise model modifier: variance is default x 3
- *Error Code = 2: $LOW\_VOLTAGE$*
  - Data becomes noisy because the voltage it too low
  - Noise model modifier: variance is default x 2
- *Error Code = 6: $ANTENNA\_SHORT$*
  - Antenna shorted
  - Permanent degraded data
  - Noise model modifier: variance is default x 4
- *Error Code = 7: $CPU\_OVERLOAD$*
  - CPU is overloaded
  - Unusable data

B.2 IMU Error Codes

The error codes were referenced from [3]

- *Error Code = 02: $Lowbattery$*
  - Data becomes noisy at low battery
  - Noise model modifier: variance is default x 2
- *Error Code = 12: $Gyroscopenotstationary$*
  - Data becomes noisy
  - Noise model modifier: variance is default x 3
- *Error Code = 13: $Accelerometeraxismisaligned$*
  - Data reliability is decreased
  - Noise model modifier: variance is default x 4
- *Error Code = 08: $Toofewbytesinpacket$*
  - Data becomes unusable
  - Unusable data

B.3 Plausibility faults

B.3.1 Common to all streams

- *Data mismatch:* Incoherent data is passed
- *Frequency drops:* Messages are delayed or dropped
- *Data discontinuity:* High jumps in the data are injected or the polarity is reversed

B.3.2 GPS faults

- *Out of bound data:* Exceed latitude range of -90°to 90°or longitude range of -180°to 180°

B.3.3 IMU faults

- *Out of bound data:* Exceed yaw range of -180°to 180°