EXTENDED KALMAN FILTER AND STEREOSCOPIC VISION BASED AUTONOMOUS FLYING SYSTEM FOR QUADCOPTERS

Amila Sandaruwan Somasiri

(149364C)

Degree of Master of Science

Department of Electrical Engineering

University of Moratuwa
Sri Lanka

June 2018

EXTENDED KALMAN FILTER AND STEREOSCOPIC VISION BASED AUTONOMOUS FLYING SYSTEM FOR QUADCOPTERS

Jahapu Appuhamilage Amila Sandaruwan Somasiri

(149364C)

Dissertation submitted in partial fulfilment of the requirements for the degree Master of Science in Industrial Automation

Department of Electrical Engineering

University of Moratuwa
Sri Lanka

June 2018

DECLARATION OF THE CANDIDATE & SUPERVISORS

I declare that this is my own investigation and this dissertation does not incorporate without acknowledgement any material previously submitted for a Degree or Diploma in any other University or institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Also, I hereby grant to University of Moratuwa the non-exclusive right to reproduce and distribute my thesis/dissertation, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as articles or books).

…….. ……...

J.A.A.S Somasiri Date

The above candidate has carried out research for the Masters Dissertation under our supervision.

…….. …….....

Dr. D.P. Chandima Date

…….. ……

Dr. A.G.B.P Jayasekara Date

ACKNOWLEDGMENTS

First of all, I would like to express my deepest gratitude to my research supervisors Dr. D. P Chandima and Dr. A. G. B. P Jayasekara for all the outstanding guidance, encouragement, advices and support, throughout the MSc thought course and related research.

I would like to express my deep appreciation to Head of the Department and academic staff of Department of Electrical Engineering, Faculty of Engineering, University of Moratuwa for their kind support and the guidance throughout the course.

Also, I owe special thanks to Dr. Sanjeeva Maithripala Senior Lecturer, Department of Mechanical Engineering, Faculty of Engineering, University of Peradeniya for his kind guidance throughout my master degree.

My sincere thanks also goes to my colleagues who are working with me and CodeGen (Pvt) Ltd for generous support throughout my master degree.

I would also like to thanks non-academic staff of Department of Electrical Engineering, Faculty of Engineering, University of Moratuwa for their kind support.

Furthermore, I would express my heartiest thanks my beloved parents, my beloved wife and beloved family members for their continuing love and support during my master degree.

Abstract

This thesis can be divided into two main modules. First module is implementation of an Extended Kalman filter and introduce into existing flight control algorithm which is used to control multi-rotor unmanned vehicles. Purpose of this implementation is to improve flight performance and reliability of the system. Second module is implementation of an obstacle avoidance system based on stereo vision and fuzzy logic for same flight control algorithm to avoid crashes and avoid obstacles during navigation. In this thesis Chapter 1 introduce basic modules of this implementations and explain about flight control algorithm and its major components which is used in here. This chapter also explains the theory behind the Extended Kalman Filters, stereo vision systems and fuzzy logic. Chapter 2 described literature survey about existing implementation of Extended Kalman filters on multi-rotor platforms, stereo vision system implementations and related obstacle avoidance implementations like artificial potential field and fuzzy logic. First section of chapter 3 focused into implementation details and experimenting results of Extended Kalman filter and also explained how Extended Kalman filter outputs are combined to Attitude and Position controllers of flight control algorithm. Second section of chapter 3 focused into implementation and experimenting results of the stereo vision system. This section explained detail implementation of stereo vision system like stereo camera calibration, image rectification, disparity map generation and depth calculation. Mainly OpenCV was used in this implementation. Third section of chapter 3 focused into explained implementation of fuzzy decision-making system. In here described deciding of fuzzy inputs and outputs using depth image, creation of fuzzy inference system, selection of membership functions and combined fuzzy decision-making system with flight control algorithm. Flight testing and experimental results of Extended Kalman filter and obstacle avoidance system were described in chapter 4, both systems were tested on outdoor environments and improvement of the performance and reliability was discussed in this chapter. Chapter 5 is the final chapter of this thesis and it includes conclusion of the thesis, recommendations and further works.

Keywords: Quadcopters, Kalman Filters, Obstacle Avoidance, Stereo Vision, Fuzzy Logic.

TABLE OF CONTENTS

DECLARATION OF THE CANDIDATE & SUPERVISORS i

ACKNOWLEDGMENTS ii

Abstract iii

TABLE OF CONTENTS iv

LIST OF FIGURES vi

LIST OF ABBREVIATIONS ix

LIST OF APPENDICES x

1 INTRODUCTION 1

1.1 Flight control algorithm 2
1.1.1 Hardware Abstraction Layer 2
1.1.2 State estimation 3
1.1.3 Attitude control & Position control 3
1.1.4 Communication protocol 3
1.1.5 Key features of the flight control algorithm 5

1.2 Kalman filter 6
1.2.1 Extended Kalman filter 7

1.3 Stereoscopic Vision 8
1.3.1 Camera calibration 8
1.3.2 Image rectification 10
1.3.3 Disparity 11
1.3.4 Pixel correlation (Block matching) 12
1.3.5 Disparity calculation (Stereo correspondence) algorithms 12
1.3.6 Depth calculation 14
1.3.7 Fundamental problems in stereo correspondence 15

1.4 Fuzzy Logic 16
1.4.1 Membership functions 16
1.4.2 Fuzzy inference system 16

2 LITERATURE REVIEW 17

2.1 Related quadcopter states estimator implementations 17

2.2 Related stereo vision implementations 17

2.3 Related obstacle avoidance implementations 18

3 METHODOLOGY 19

3.1 Development of the Extended Kalman filter 19
3.1.1 Inputs and outputs of the filter 19

3.1.2 Extended Kalman filter equations 20
3.1.3 Hardware selection and assembling of the Quadcopter 25
3.1.4 Experimental results of the filter 29
3.1.5 Introduce Extended Kalman filter outputs to flight control algorithm 34
3.2 Development of the stereo vision system 36
3.2.1 Selection of the stereo camera 36
3.2.2 Stereo camera calibration 38
3.2.3 Image rectification 40
3.2.4 Disparity calculation 40
3.2.5 Manual calibration of the depth image 44
3.2.6 Testing of stereo vision system on outdoor environment 45
3.3 Development of the fuzzy decision-making system 47
3.3.1 Analyzing the Depth Image 47
3.3.2 Calculation of size of sub regions 47
3.3.3 Calculation of normalize depth values of sub regions 49
3.3.4 Determination of fuzzy inputs and outputs 49
3.3.5 Determination of fuzzy rules 50
3.3.6 Selection of input and output membership functions 52
3.3.7 Selection of defuzziffication method 53
3.3.8 Introduce fuzzy inference system output to flight control algorithm 54
3.3.9 Testing of the fuzzy system on outdoor environment 55
4 EXPERIMENTAL RESULTS AND ANALYSIS 58
4.1 Flight test and result analyzing of Extended Kalman filter 58
4.1.1 Hardware setup 58
4.1.2 Experimental procedure 58
4.1.3 Results 59
4.1.4 Position controller error variation with Extended Kalman filter 64
4.1.5 Problems encountered during flight tests 67
4.2 Flight tests and results analyzing of obstacle avoidance system 68
4.2.1 Hardware setup 68
4.2.2 Experimental procedure 68
4.2.3 Results 69
4.2.4 Problems encountered during flight tests 70
5 CONCLUSIONS 71
6 RECOMMENDATIONS AND FURTHER WORKS 72
REFERENCES 74
APPENDICES 76

LIST OF FIGURES

Figure 1.1 Basic components of simple Multi-copter 4
Figure 1.2 MAVLink communication protocol block diagram 4
Figure 1.3 Graphical user interface use to communicate with quadcopter 5
Figure 1.4 Radial distortion 8
Figure 1.5 Tangential distortion 9
Figure 1.6 Rectification of a stereo pair. 10
Figure 1.7 The Tsukuba stereo image pair: 11
Figure 1.8 Relationship between depth of a point and disparity of same point 15
Figure 1.9 Block diagram of a fuzzy inference system. 16
Figure 3.1 Navio2 flight shield connected with Raspberrypi 3 computer board. 26
Figure 3.2 Selected 850KV brushless dc motor and 20Amp ESC 26
Figure 3.3 Selected transmitter and receiver 27
Figure 3.4 Selected Radio telemetry 27
Figure 3.5 Selected lithium-polymer battery 28
Figure 3.6 Actual hardware set-up of the system 28
Figure 3.7 Connections of the hardware components 29
Figure 3.8 GPS position vs EKF position for Latitude direction 30
Figure 3.9 Zoom version of Figure 3.8 30
Figure 3.10 GPS position vs EKF position for Longitude direction 31
Figure 3.11 Zoom version of Figure 3.10 31
Figure 3.12 GPS velocity vs EKF velocity for Latitude direction 32
Figure 3.13 GPS velocity vs EKF velocity for Longitude direction 32
Figure 3.14 Barometer height vs EKF height 33
Figure 3.15 Barometer climb rate vs EKF climb rate 34
Figure 3.16 Attitude control of quadcopter 35
Figure 3.17 Position control of quadcopter 35
Figure 3.18 Normal USB web camera. 36
Figure 3.19 Stereo camera build using two normal USB web camera 36
Figure 3.20 ZED stereo camera with Nvidia Jetson TX1 computing platform. 38
Figure 3.21 Stereo calibration using a 7 × 10 chessboard pattern 39

Figure 3.22 Raw images (Top) Rectified images (Bottom) 40
Figure 3.23 User interface for block matching algorithm 42
Figure 3.24 Effect of changing number of disparities. 42
Figure 3.25 Disparity maps for different SAD window sizes 43
Figure 3.26 Input raw images and corresponding depth map 44
Figure 3.27 Moment of manual calibration of depth image 45
Figure 3.28 Quality of the depth image in different scenarios 46
Figure 3.29 Depth map's division in nine windows. 47
Figure 3.30 Visible image plane of the stereo camera 48
Figure 3.31 Dimension of each sub regions 49
Figure 3.32 Navigation direction deciding based on depth image. 50
Figure 3.33 Input membership funcition of desired velocity input (Vin). 52
Figure 3.34 Input membership funcition of normalize depth value of one region (Dlu) 52
Figure 3.35 Output membership funcitions of Vx, Vz 53
Figure 3.36 Output membership funcition of Vy 53
Figure 3.37 Fuzzy inference system 53
Figure 3.38 Combination of stereo vision system, fuzzy inference system with flight control algorithm 54
Figure 3.39 Code architecture runs on Nvidia Jetson TX1 55
Figure 3.40 Outdoor testing of fuzzy decision-making system. 56
Figure 3.41 Variation of depth values of fuzzy decision-making system. 56
Figure 3.42 Variation of velocities of fuzzy decision-making system. 57
Figure 4.1 Final assembled quadcopter 58
Figure 4.2 Quadcopter flying on open field 59
Figure 4.3 Barometer measured height vs EKF estimated height 60
Figure 4.4 Barometer measured climb rate vs EKF estimated climb rate 60
Figure 4.5 GPS measured Latitude position vs EKF estimated Latitude position 61
Figure 4.6 GPS measured Longitude position vs EKF estimated 62
Figure 4.7 Zoom version of GPS measured Longitude position vs EKF estimated Longitude position between data point 6600 and 67000 62
Figure 4.8 GPS measured Latitude velocity vs EKF estimated Latitude velocity 63

Figure 4.9 GPS measured Longitude velocity vs EKF estimated 63
Figure 4.10 Actual height variation vs desired height variation 64
Figure 4.11 Height error variation 65
Figure 4.12 Actual position variation vs desired position variation in 65
Figure 4.13 Position error variation in Latitude direction 66
Figure 4.14 Actual position variation vs desired position variation in Longitude direction 66
Figure 4.15 Position error variation in Longitude direction 67
Figure 4.16 Overall hardware setup including stereo camera. 68
Figure 4.17 Quadcopter fly over tree 69
Figure 4.18 Quadcopter fly to a wall. 70

LIST OF ABBREVIATIONS

Abbreviation	Description
BM	Block matching
CPU	Central processing unit
EKF	Extended Kalman Filter
ESC	Electronic speed control
FIS	Fuzzy Inference system
GCS	Ground control station
GPU	Graphical processing unit
GPS	Global Positioning system
GUI	Graphical user interface
HAL	Hardware Abstraction Layer
MEMS	Microelectromechanical systems
NCC	Normalized cross correlation
OpenCV	Open source Computer Vision
PWM	Pulse Width Modulation
RTOS	Real-time operating system
SGBM	Semi-Global block matching
SAD	Sum of absolute difference
SSD	Sum of square difference
UAV	Unmanned Aerial Vehicle
UDP	User Datagram Protocol
USB	Universal Serial Bus

LIST OF APPENDICES

Appendix	Description
Appendix - A	MATLAB Symbolic Implementation of Extended Kalman Filter.
Appendix - B	C++ Implementation of Extended Kalman Filter.
Appendix - C	C++ Implementation of Stereo Vision System.
Appendix - D	C++ Implementation of Fuzzy Decision-making System.
Appendix - E	Extended Kalman filter ground test data.
Appendix - F	Extended Kalman filter flight test data.
Appendix - G	Obstacle avoidance system ground test data.
Appendix - H	Video evidence of system test.

Note: Appendices are available on the provided compact disk (CD).

1 INTRODUCTION

Basically, this research can be divided into two sections. Implementation of an Extended Kalman filter to improve the performance of multi-copter flight control algorithm presented in [1] and [2] and implementation of stereo vision and fuzzy logic-based obstacle avoidance algorithm for same flight control algorithm. There are several multi-rotor platform types available. These are categorized based on number of motors they have, like Tri-copters, Quad-copters, Hexa-copters and Octo-copters. In here Quad-copter was selected because of theirs less complexity compares to Hexa-copters or Octo-copters and relatively more stable than Tri-copters. To improve the flight performance of quadcopters, like improve robustness of position hold or autonomous flight to given waypoints, automatic take-off and landing, flight control algorithm should have accurate and robust attitude and position control system. To improve performance of these controllers, flight control algorithm should have reliable state estimation system. Therefore, in here Extended Kalman filter was proposed to the flight control algorithm presented in [1] and [2]. Extended Kalman filters are non-linear version of the linear Kalman filters. Due to non-linearity behaviour of multi-rotor vehicle attitude, we cannot use linear Kalman filter to estimate attitude of the system. Therefore, Extended Kalman filter was introduced to the system and it can estimate both attitude and position of the vehicle simultaneously. Unlike other filters like complementary filters or low pass filters over advantage of Kalman filter is we can fuse all available measurement at once. This will improve performance and reliability of the vehicle.

When flying quadcopters or other types of unmanned vehicle pilot needs more skill to avoid crashing into walls, tree or other obstacles. This is serious burden for untrained pilot and therefore, system should have reliable obstacle avoidance system. In many situations ultrasonic or Infar red sensors were used. This type of sensors suffers from lack of information around their environment and thus they give faulty readings most of the cases. As human use stereo vision to avoid obstacles around them when navigating, the same principal was used in here. Therefore, stereo vision and fuzzy logic based control method was proposed to detect and avoid obstacles accurately.

Rest of this chapter briefly explain the basic components of the flight control algorithm presented in [1] and [2] and theory behind the Extended Kalman filters, Stereo vision and Fuzzy Logic.

1.1 Flight control algorithm

Multi-Copters or Quad-copters are aerodynamically unstable and require an on-board flight control algorithm for stable flight. The flight controller algorithm abstract data from all on-board sensors like MEMS accelerometers, gyroscopes, barometers, and GPS and estimate system orientation and position and then theses information fed into to attitude and position controller to maintain quadcopter attitude and position. Figure 1.1 shows the basic components that include in most of multi-rotor platforms. Quad-copter can control its attitude or orientation (i.e. Roll, Pitch and Yaw) by changing its rotors speeds. Flight control algorithm presented in [1] and [2] introduce Globally stabilize intrinsic non-linear PID controller to stabilize any kind of multi-rotor system. This flight control algorithm presented in [1] and [2] was tested on several multi-rotor platforms like quad-copters, hexa-copters and octo-copters and it consists with following basic components.

1. Hardware Abstraction Layer.
2. State estimation.
3. Attitude & Position Control.
4. Communication Protocol.

1.1.1 Hardware Abstraction Layer

Hardware Abstraction Layer read sensor data and to send drive commands to motors. HAL is consisting with common driver set for peripheral devices. It also provides a device driver interface to communicate with the hardware. The main purpose of a HAL is to handle different hardware architectures from the operating system by providing a uniform interface to the system peripherals. Flight control algorithm presented in [1] and [2] supported, flight controller boards which are runs Linux based operating systems.

1.1.2 State estimation

To estimate the attitude of the multi-rotor, flight control algorithm presented in [1] and [2] consists with complementary filter implemented by Mahony et al presented in [5]. Currently, flight control algorithm does not have position estimation filter and GPS measured horizontal position and barometer measured height are directly fed into the position controller. Because of this reason performance of the position control of the vehicle is poor. So, vehicle needs reliable estate state estimation system to perform better for its autonomous operations.

1.1.3 Attitude control & Position control

The quadrotor control structure is usually divided into internal and external loops according to the characteristics of the system model. The internal control loop is used to control attitude (orientation) of the vehicle and the external control loop is used to control the position of the vehicle. Flight control algorithm presented in [1] and [2] used Globally stabilize intrinsic non-linear PID controller to control its attitude. To control position angle based linear PID control algorithm was used.

1.1.4 Communication protocol

It is important to maintain stable communication between flying machine and user. Because we have to give control commands through this communication protocol. In here MAVLink communication protocol was used. MAVLink is mostly used for communication between a ground control station and unmanned aerial vehicles. MAVLink protocol can be used to transmit the like vehicle position, velocity and orientation of the vehicle. It allows any type of communication devices such as Radio Telemetry, Wi-Fi or any other UDP device. Figure 1.2 shows working principal of MAVLINK using block diagrams. Figure 1.3 shows graphical user interface of flight control algorithm presented in [1] and [2] which is use to communicate with any kind of multi-rotor.

Figure 1.1 Basic components of simple Multi-copter

Reference: http://ardupilot.org/copter/docs/introduction.html

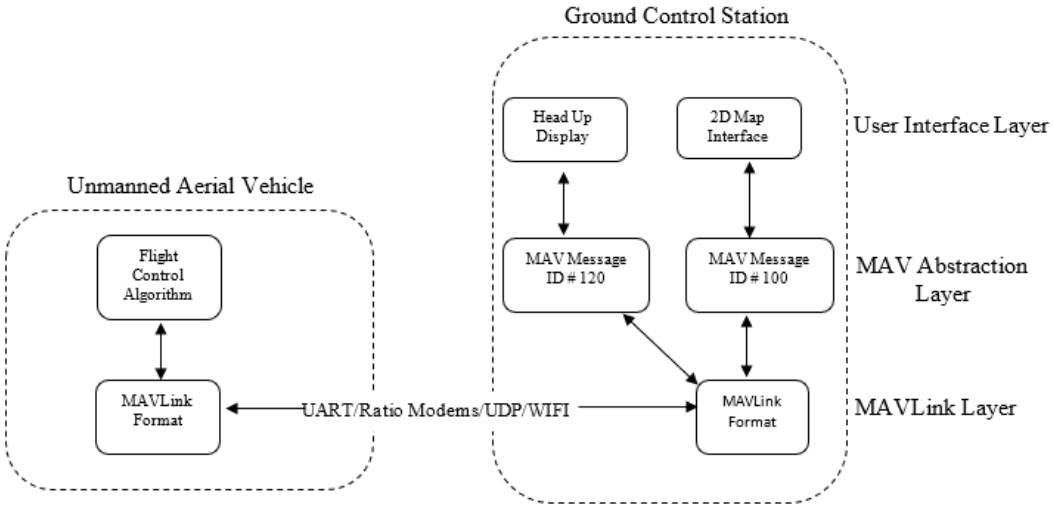

Figure 1.2 MAVLink communication protocol block diagram

Reference: https://www.slideshare.net/nicholasyoonchun/open-source-drone-sineverydaylife-66474415

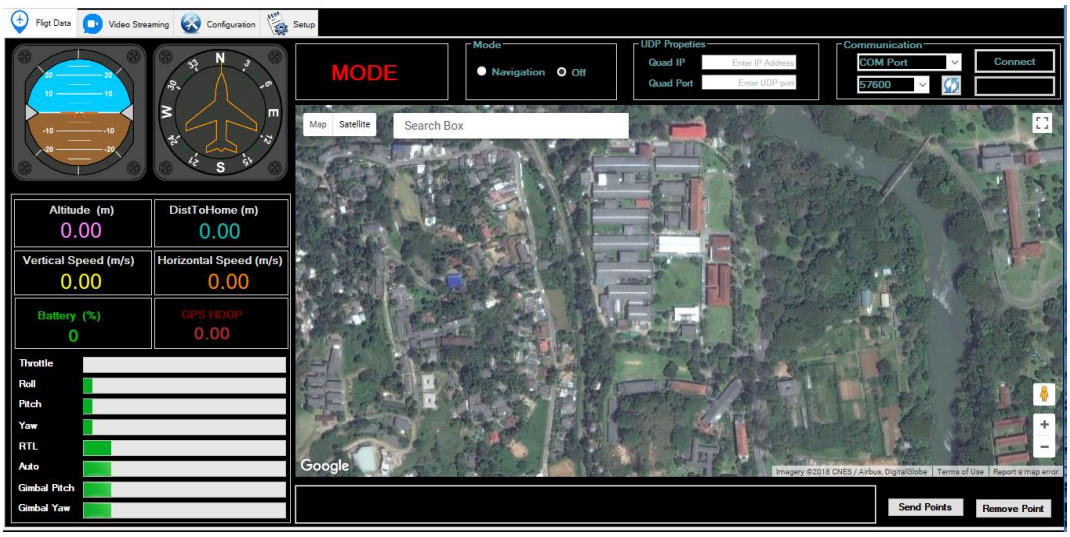

Figure 1.3 Graphical user interface use to communicate with quadcopter

1.1.5 Key features of the flight control algorithm

- High accurate aerobatic flight mode: Perform aggressive flips and stable under large deviation.
- Automatic Position Hold mode: The quadcopter holds its position using its GPS, accelerometers and barometer.
- Return to Home: Vehicle will be fly back to take-off position and land automatically.
- Autonomous missions: Using Graphical User Interface we can upload the GPS waypoints to vehicle. Then copter will fly automatically to these waypoints and come back to launch point without any human intervention.
- Failsafe: In case of loss of contact with the pilot, low battery or the vehicle goes outside a defined geofence copter will triggers an autonomous return-to-home.

1.2 Kalman filter

Multi-rotor position, orientation and velocities are critical states and need to be accurately estimated as mention in previous. The Kalman Filter has been applied to many aerospace applications and used in many navigation systems. Most popular states estimators are Kalman filters and its non-linear versions such as the Extended Kalman Filter. Standard Kalman filter is linear estimator like complementary filter and it can use to minimize the error variance. Unlike complementary filter gain Kalman filter itself optimize its filter gain and this gain calls Kalman gain. If system is linear then it can be mathematically represented as follows.

$$x_{k+1} = Ax_k + Bu_k + w_k \tag{1.1}$$

$$y_k = Hx_k + z_k \tag{1.2}$$

Where A is state matrix, B is input matrix and H is measurement matrix. x is state of the system, u is known input to the system, y is the measured output, w is a process noise and z is a measurement noise.

Kalman filter assumes that process noise and measurement noise are zero mean gaussian noises and they also not correlated. According to this assumption covariance matrices of process noise (Q) and measurement noise (R)can be written as follows.

$$Q_k = E(w_k, w_k^T) \tag{1.3}$$

$$R_k = E(z_k, z_k^T) \tag{1.4}$$

Kalman filter process can be divided into two stages. State and covariance prediction stage and state and covariance update stage. Theses stages can be mathematically represented by using equation (1.1) (1.2) (1.3) and (1.4).

Prediction stage

Predicted State $$\hat{x}_{k|k-1} = Ax_{k-1} + Bu_{k-1} \quad (1.5)$$

Predicted Covariance $$P_{k|k-1} = AP_{k|k-1}A^T + Q_k \quad (1.6)$$

Update stage

Innovation $$\tilde{y}_k = z_k - h(\hat{x}_{k|k-1}) \quad (1.7)$$

Innovation Covariance $$S_k = H_k P_{k|k-1} H_k^T + R_k \quad (1.8)$$

Kalman Gain $$K_k = P_{k|k-1} H_k^T S_k^{-1} \quad (1.9)$$

Update State $$\hat{x}_{k|k} = \hat{x}_{k|k-1} + k_k \tilde{y}_k \quad (1.10)$$

Update Covariance $$P_{k|k} = (I - K_k H_k) P_{k|k-1} \quad (1.11)$$

Where $\hat{x}_{k|k-1}$ is the predicted state estimation at actual time step, $P_{k|k-1}$ is the predicted estimated covariance matrix, $\hat{x}_{k|k}$ is updated state estimation from previous time step $P_{k|k}$ is the updated estimated covariance at previous time step and K_k is Kalman gain for actual time step.

1.2.1 Extended Kalman filter

When it comes to Extended Kalman filter system matrix (A) contains non-linear terms. As a result of this system state can be represent as equation (1.12). Therefore, system matrix (A) and measurement matrix (H) can be represented as equation (1.13) and (1.14). All the steps and equations are same as linear Kalman filter.

$$\hat{x}_{k|k-1} = f(\hat{x}_{k-1|k-1}, u_k) \quad (1.12)$$

Where $$A = \frac{\partial f}{\partial x}\Big|_{\hat{x}_{k-1|k-1}, u_k} \quad (1.13)$$

$$H = \frac{\partial h}{\partial x}\Big|_{\hat{x}_{k|k-1}} \quad (1.14)$$

1.3 Stereoscopic Vision

Using pair of stereo images which is acquired using stereo camera and calculating the distance to each pixel of the image is most important and difficult task in robotics and computer vision applications. Depth images can be extract using pixel wise matching between the left image and right image which is provided by stereo camera. Stereo vision is commonly use in fields of machine vision, computer vision, VR/AR (Virtual reality and Augmented reality), robot navigation, simultaneous localization and mapping, depth measurements and 3D environment reconstruction.

There are four main steps in stereoscopic vision process.

1. Due to Camera lens distortion first, we have to perform camera calibration to find camera distortion coefficients.
2. Due to camera misalignment, we have to perform Image rectification to get co-planner images.
3. Disparity Calculation.
4. Depth Calculation.

1.3.1 Camera calibration

Due to lens curvature and misalignment between image plane (Image sensor) and lens, images taken from cameras are always distorted. There two types of distortion.

Radial distortion: Due to curvature shape of the camera lens when light rays bend more near the edges of a lens than its optical center (Figure 1.4). Due to radial distortion, expected straight lines in the corner of the images are appear as curved. Its effect is increase we move away the camera from the center of image.

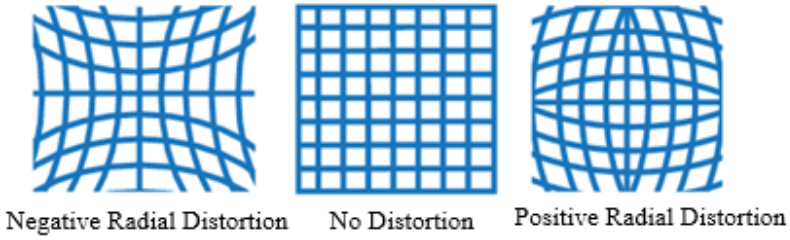

Figure 1.4 Radial distortion

Reference: https://www.mathworks.com/help/vision/ref/cameraintrinsics-class.html

Tangential distortion: Occurs due to misalignment of the image plane and lens as shown in Figure 1.5. Due to tangential distortion some areas in image may look nearer than expected.

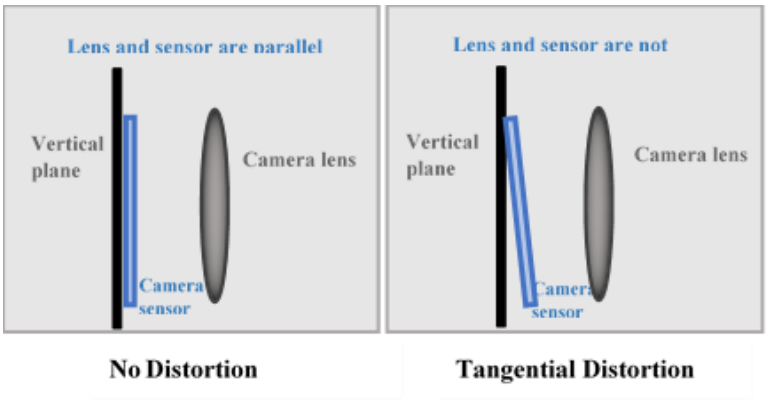

Figure 1.5 Tangential distortion

Reference: https://www.mathworks.com/help/vision/ug/camera-calibration.html

To eliminate this camera image distortion effect on depth calculation, we have to perform stereo camera calibration to find camera image distortion coefficients i.e. Radial distortion coefficient and Tangential distortion coefficient for both left and right camera.

The Radial distortion can be represented as follows:

$$x_{\text{distorted}} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \quad (1.15)$$

$$y_{\text{distorted}} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \quad (1.16)$$

The Tangential distortion can be represented as follows:

$$x_{\text{distorted}} = x + [2p_1 xy + p_2(r^2 + 2x^2)] \quad (1.17)$$

$$y_{\text{distorted}} = y + [p_1(r^2 + 2y^2) + 2p_2 xy] \quad (1.18)$$

So all together we need to find five distortion coefficients given by:

$$\text{Distortion Coefficients} = (k_1\ k_2\ k_3\ p_1\ p_2) \quad (1.19)$$

In addition to this, intrinsic parameters (Camera matrix) and extrinsic parameters of the stereo camera need be find. Intrinsic parameters are camera specific parameter and it includes focal length (f_x, f_y) and optical centers (c_x, c_y). These parameters can be stored as a 3x3 matrix:

$$\text{Camera Matrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{1.20}$$

The extrinsic parameters are rotation and translation vectors of the right camera compare to left camera.

1.3.2 Image rectification

In most of the situations cameras of the stereo camera are not perfectly aligned and also with high precision equipment it may be impractical to align these cameras. Due to this misalignment of those cameras it is difficult to find corresponding points of left camera image and right camera image. Therefore, we have to perform image rectification before going into next step. As shown in Figure 1.6 in the process of rectification will replace the initial images of both camera by using another projective equivalent pair. That means initial images are re-projected to common plane which parallel to the baseline of the stereo camera.

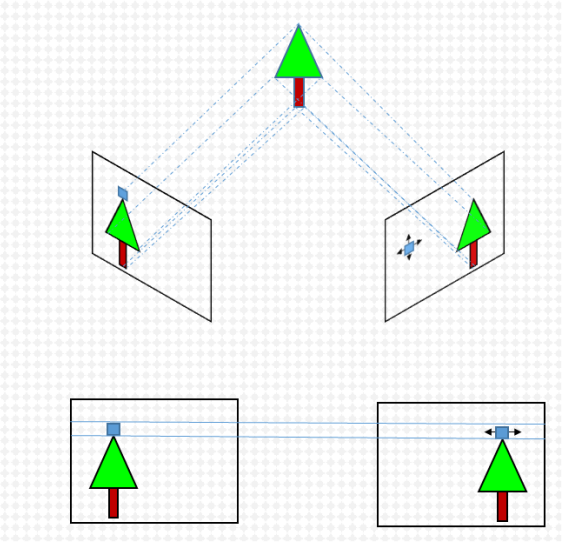

Figure 1.6 Rectification of a stereo pair.

Reference: https://sites.google.com/site/stereoimagesrectification/introduction

These Projection matrices $(P_l\ P_r)$ for left and right camera can be written using camera matrix (equation 1.20) and extrinsic parameters (Right camera translation and rotation compare to left camera) as shown in equation (1.21) and equation (1.22).

$$P_l = \begin{bmatrix} f_l & 0 & cx_l & 0 \\ 0 & f_l & cy_l & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{1.21}$$

$$P_r = \begin{bmatrix} f_r & 0 & cx_r & T_x \times f_r \\ 0 & f_r & cy_r & T_y \times f_r \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{1.22}$$

1.3.3 Disparity

The disparity is horizontal distance between two matching pixels relative to left camera optical centre or right camera optical centre (In most cases relative to left camera optical centre) and the disparity map defines a value of this horizontal pixel distance for each image pixel coordinate. Hence, it is a function of $d(x, y)$ where d is disparity value of x^{th} and y^{th} pixel. The Tsukuba stereo image pair is shown together with a disparity map in Figure 1.7.

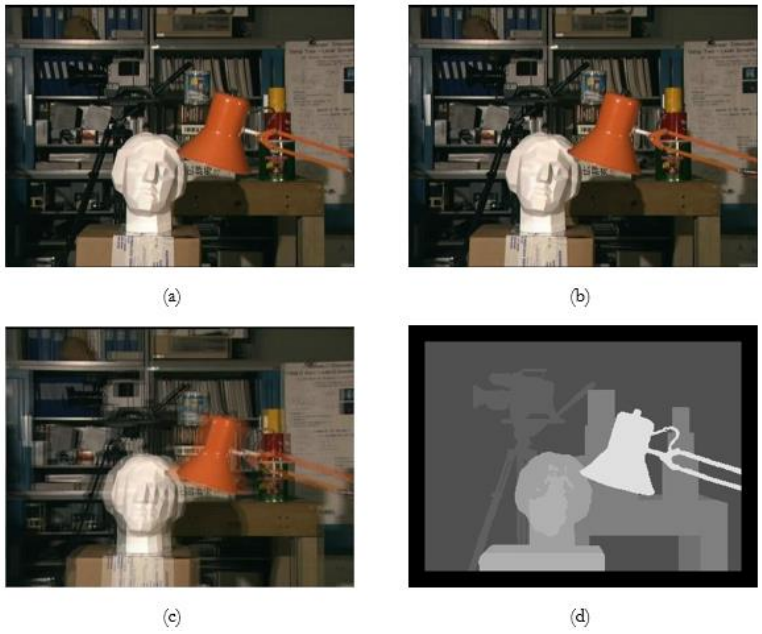

Figure 1.7 The Tsukuba stereo image pair:
a) Left view b) Right view c) Superimposed left and right view. d) Disparity map

1.3.4 Pixel correlation (Block matching)

To find disparity of each image pixel first we have to identify the conjugate pairs (blocks) in both left and right images. To identify two blocks as a conjugate pair, it is necessary to measure the similarity of these blocks. Several algorithms were proposed for this task and most of the algorithm uses a matching cost function to identify conjugate pair of pixels. The most common matching cost functions are sum of absolute differences (SAD), sum of squared differences (SSD) and the normalized cross correlation (NCC).

Sum of absolute differences (SAD) can be expressed as:

$$\mathrm{SAD}(x, y, d) = \sum_{x.y \epsilon W} |I_l(x, y) - I_r(x, y - d)| \tag{1.23}$$

Sum of squared differences (SSD) can be expressed as:

$$\mathrm{SSD}(x, y, d) = \sum_{x,y \epsilon W} (I_l(x, y) - I_r(x, y - d))^2 \tag{1.24}$$

Normalized cross correlation (NCC) can be expressed as:

$$\mathrm{NCC}(x, y, d) = \frac{\sum_{x,y \epsilon W} I_l(\mathrm{x}, \mathrm{y}) - I_r(\mathrm{x}, \mathrm{y} - \mathrm{d})}{\sqrt{\sum_{x,y \epsilon W} I_l^2(x, y) \,.\, \sum_{x,y \epsilon W} I_l^2(x, y - d)}} \tag{1.25}$$

Where I_l I_r are the intensity values in the left and right image, (x, y) are the pixel's coordinates, d is the given disparity value and W is the support region or block-size. Conjugate pairs are search in correspondence rows only. This is because images are rectified and this will eliminate the search of vertical conjugate pairs.

1.3.5 Disparity calculation (Stereo correspondence) algorithms

In above topic we already assigned random disparity values for each pixel based on matching cost algorithm SAD, SSD or NCC. Now we have to find disparity values in entire image. For this purpose, we used Stereo correspondence algorithms which are almost contains several optimization techniques.

Stereo correspondence algorithms can be divided into mainly two groups. First one is Dense Disparity algorithms and second one is Sparse Disparity algorithms. However Sparse disparity algorithms are low accurate and almost out of date.

Therefore, here I do not focus into sparse disparity algorithms. When it comes to dense disparity algorithms we can divide this into two groups based on their optimization techniques. First one is Local methods also called area-based methods or windows-based methods. In this method disparity computation of a given pixel depends only on the intensity value of the selected window. Second method is Global methods also called energy-based methods. Local methods are less accurate compare to Global methods because they only focus on local minima but considerably fast compare to Global methods. On the other hand, Global methods are more accurate because they focus on global minima but computationally extensive compare to local methods.

Local methods

Local methods only use information located in a close pixel in support window (block). In here we assign disparity value that minimizes the cost function for each individual pixel. This method is also called as Winner-Take-All (WTA) optimization and mathematical representation for this type of optimization is shown in equation (1.26).
When using local Winner-Take-All optimization, the disparity map is defined as:

$$d(x,y) = argmin\ C(x,y,d) \tag{1.26}$$

Where $d(x,y)$ is disparity value of the x^{th} and y^{th} pixel and $C(x,y,d)$ is matching cost function based on SAD, SSD or NCC as mention equation (1.23) (1.24) and (1.25).

Global methods

Global methods find disparities for all reference image pixels at once. This is achieved by minimizing an energy function using some optimization technique. The energy function used in stereo matching usually include two terms called correspondence data term and a smoothness term.

$$E(d) = E_{data}(d) + \boldsymbol{\lambda}. E_{smooth}(d) \tag{1.27}$$

In above $d = d(x, y)$ is the disparity value of each pixel of the reference image and λ is a parameter that adjusts smoothness of the result. The data term is normally based on sum of a matching cost function such as SAD SSD or NCC as mention equation (1.23) (1.24) and (1.25).

$$E_{data}(d) = \sum_{(x,y)\epsilon Im} C(x, y, d(x, y)) \tag{1.28}$$

The smoothness term is often a function depending on differences in disparity and intensity of neighboring pixels according to:

$$\begin{aligned} E_{smooth}(d) = \sum_{(x,y)\epsilon Im} & \rho[d(x, y) - d(x + 1, y),\ I(x, y) - I(x + 1, y)] \\ & + \rho[d(x, y) - d(x + 1, y), I(x, y) - I(x + 1, y)] \end{aligned} \tag{1.29}$$

Here ρ is tunable value and idea behind this value is increase monotonically with disparity difference to penalize a discontinuous result, but at the same time be able to reduce this penalty for disparity discontinuities located at color edges.

1.3.6 Depth calculation

Once we have derived the disparity ($d = x - x'$) for each image pixel (x, y) and camera matrices, the calculation of the depth for each pixel is very straightforward as shown in Figure 1.8. Once we know the disparity ($x - x'$) of each pixel then we can estimate how far away the object is. The relation between disparity and depth is given by

$$\text{Depth } (Z) = \frac{Bf}{x - x'} \tag{1.30}$$

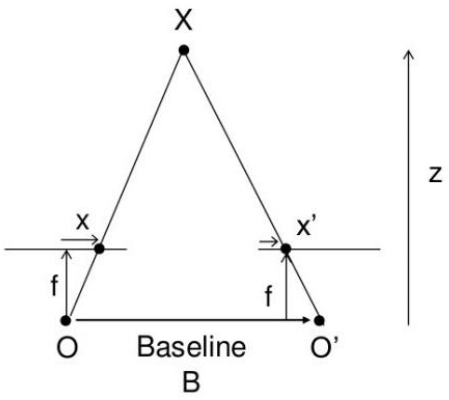

Figure 1.8 Relationship between depth of a point and disparity of same point

Where B is the distance between the optical centers of two cameras, also called the baseline and f is the focal length of the camera. According to above equation, disparity and depth of a point is inversely proportional.

1.3.7 Fundamental problems in stereo correspondence

Occlusion

Occlusion occurs when a point in the 3D space in front of the cameras gets depicted in one of the images and is blocked from being depicted in the other one. There is no general solution to the occlusion problem and as occlusion may cause to faulty matches.

Uniform texture and lack of texture

Another challenge is to handle surfaces with uniform texture and surfaces that lack features like white wall, glasses. When surfaces with these properties are encountered in a stereo image pair, it becomes complicated to decide which pixels in the left and right images that are corresponding to each other.

Sensor noise and bias

Camera sensor noise is another issue for stereo matching in that two matching pixels that should have the same intensities may not. This problem is especially occurring in poorly textured image regions or in images taken under poor lighting conditions. If two different cameras are used to capture the stereo images there might also be a difference in gain or a bias in the intensity of matching pixels in the left and right image.

1.4 Fuzzy Logic

Fuzzy logic is use in many applications such as robotics, image processing, industrial applications. Fuzzy logic has ability to replicate the human decision-making process. Basic concept in Fuzzy logic is the fuzzy if–then rule. Fuzzy logic can be used to model nonlinear systems and the most important thing is it can handle multi-input, multi-output system.

1.4.1 Membership functions

In fuzzy logic membership functions is a gaussian, trapezoidal, or any other shape of curve that are used to match inputs space to 0-1 degree of value. Each system has one or more input membership functions and output membership functions. According to variation of inputs and output variables shape of membership function will be decided.

1.4.2 Fuzzy inference system

Fuzzy inference system (FIS) is mapping of input data and output data using fuzzy rules. Mainly FIS has four main components, fuzzifier, inference engine, rule base or knowledge base, and defuzzifier. Fuzzifier maps input space into fuzzy membership functions. The rule base or knowledge base consists with if-then rules that are provided by human expertise. The inference engine maps input fuzzy sets into output fuzzy sets and determine satisfied combination of each rules. The defuzzifier maps output fuzzy sets into a single number. There are many defuzzification methods available. The most commonly used method is the centroid and other methods are Bisector, Mean of maximum, Sum of maximum and Largest of maximum etc. A general schematic of a FIS is shown in Figure 1.9.

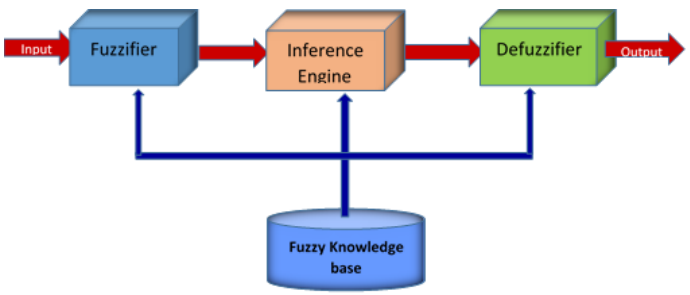

Figure 1.9 Block diagram of a fuzzy inference system.

2 LITERATURE REVIEW

2.1 Related quadcopter states estimator implementations

Mahony et al [5] presented attitude estimator for multirotor vehicles using low cost inertial measurement unit and observer has global convergence property for any initial state. Madgwick [6] presented attitude estimator which very similar to observer described in [5], however, this filter uses gradient-descent algorithm to compute the direction of the gyroscope measurement and also include magnetic bias and gyroscope bias compensation. Sabatini [7] presented quaternion-based Extended Kalman Filter (EKF) for estimating the attitude of a rigid body which can be applied to multirotor vehicle attitude estimation and filter also includes magnetic bias estimation and gyroscopic measurement bias estimation. Madinehi [8] presented several Kalman filter derivatives like Multiplicative EKF, Additive EKF, Unscented Kalman Filters and Invariant Kalman Filters, which can be applied to multirotor vehicle attitude estimation.

2.2 Related stereo vision implementations

Lazaros et al [9] proposed novel stereo vision algorithm using Laplacian of Gaussian and edge detection. It can achieve 10 frames per second for 19 x 19-pixel images. Muhlmann et al [10] presented a local optimization stereo correspondence algorithm which uses the sum of absolute differences (SAD) and it can achieve 20 frames per second, for 160 × 120-pixel images. Di Stefano et al [11] presented another local optimization SAD based stereo correspondence algorithm. It achieves 39.59 frames per second for 320×240-pixel images. Hirshmuller [12] presented stereo vision algorithm which is already implemented in OpenCV call Semi-Global block matching technique. Mutual information based matching cost function is used for pixel-wise matching of images and this algorithm is mostly used in stereo vision applications. Hernandez et al [13] implemented this Semi-Global block matching algorithm in a GPU processor and it achieves 42 frames per second for 640 x 480-pixel images.

2.3 Related obstacle avoidance implementations

Lazaros et al [9] proposed stereo vision and fuzzy logic-based obstacle avoidance method for mobile robot. In here depth image is divided into three separate portion and calculate normalize depth value in each region, then these values fed into fuzzy inference system as inputs. Burschkal et al [14] proposed stereo vision-based obstacle avoidance method for mobile robots using potential field method. Proposed method tested only in indoors and achieved significant results. Borenstein [15] has proposed the virtual force field method for robot obstacle avoidance. Kyung et al [16] presented an elastic strip method for mobile robots to avoid obstacles.

3 METHODOLOGY

Development of the Extended Kalman filter and obstacle avoidance system can be divided into four separate modules for implementation purpose.

1. Implementation and Testing of an Extended Kalman filter for quadcopter and introduce it to flight control algorithm presented in [1] and [2].
   In this section explained about deciding of inputs and outputs of the filter, system equations and filter equations, hardware selection and pre-flight test of the filter.
2. Implementation of the Stereo vision system.
   In this section focused into stereo camera selection, camera calibration and image rectification using OpenCV, depth calculation and indoor and outdoor test of the stereo vision system.
3. Implementation of the Fuzzy decision-making system.
   In this section explained about analysing of depth map, deciding of fuzzy inputs and outputs, creation of fuzzy table and outdoor test result of the fuzzy system with stereo vision system.
4. Testing of flight control algorithm with above implemented Extended Kalman filter and obstacle avoidance system.

3.1 Development of the Extended Kalman filter

3.1.1 Inputs and outputs of the filter

In order to control attitude and position of the quadcopter, it is necessary to estimate actual attitude, position, and velocity of the system. Position and velocity of the quadcopter can be expressed as linear model however attitude of the system is nonlinear. Therefore, Extended Kalman filter was used to estimate attitude, position, and velocity of the quadcopter.

Advantage of the use of Kalman filter is we can fuse all available sensors (measurements) to estimate the attitude, position, and velocity of the system. However, we can estimate attitude of the system only using accelerometer and magnetometer, but results of this estimations are corrupted due to accelerometer noise, magnetometer

measurement delay and noise. We can also estimate the quadcopter position and velocity by only using GPS. But GPS measurements suffer from low update rate (less than 10Hz) and measurement lag compare Inertial measurement unit (IMU) (more than 250ms measurement delay in GPS compare to IMU). Therefore, Extended Kalman filter was used to estimate the vehicle states and filter simultaneously estimates both attitude and position of the vehicle.

Inputs:

- Body frame (Quadcopter frame) reference acceleration (Accelerometer measurements).
- Body frame reference angular rates (Gyroscopic measurements).
- Body frame reference magnetic flux measurements (Magnetometer measurements).
- Earth frame reference GPS position and velocity.
- Earth frame reference Barometer height measurement.

Outputs (States):

- Body frame (Quad-copter frame) reference attitude quaternions (q_1, q_2, q_3, q_4).
- North, East, Down earth frame reference velocities (v_n, v_e, v_d).
- North, East, Down earth frame reference positions (p_n, p_e, p_d).
- Body frame reference gyro biases $(\omega_{xb}, \omega_{yb}, \omega_{zb})$.
- Body frame reference accelerometer bias offsets (a_{xb}, a_{yb}, a_{zb}).

3.1.2 Extended Kalman filter equations

Angular rate vector: $$\omega = (\omega_x; \omega_y; \omega_z) \quad (3.1)$$

Acceleration vector: $$a = (a_x; a_y; a_z) \quad (3.2)$$

Angular rate bias vector: $$\omega_b = (\omega_{xb}; \omega_{vb}; \omega_{zb}) \quad (3.3)$$

Acceleration bias vector: $$a_b = (a_{xb}; a_{yb}; a_{zb}) \quad (3.4)$$

Body frame to earth frame rotation matrix:

$$R = \begin{bmatrix} q_1^2 + q_2^2 - q_3^2 - q_4^2, 2(q_2q_3 - q_1q_4), 2(q_2q_4 + q_1q_3) \\ 2(q_2q_3 + q_1q_4), q_1^2 - q_2^2 + q_3^2 - q_4^2, 2(q_3q_4 - q_1q_2) \\ 2(q_2q_4 - q_1q_3), 2(q_3q_4 + q_1q_2), q_1^2 - q_2^2 - q_3^2 + q_4^2 \end{bmatrix} \quad (3.5)$$

Rate of change of quaternion vector:

$$dq = (1; 0.5(\omega_x - \omega_{xb})dt; 0.5(\omega_y - \omega_{yb})dt; 0.5(\omega_z - \omega_{zb})dt) \quad (3.6)$$

State prediction:

Using quaternion vector and equation (3.6) quaternion update equation can be written as follows:

$$q_{New} = ((q_1dq_1 - q_2dq_2 - q_3dq_3 - q_4dq_4), (q_1dq_2 + q_2dq_1 + q_3dq_4 - q_4dq_3), (q_1dq_3 + q_1dq_1 + q_4dq_2 - q_2dq_4), (q_1dq_4 + q_4dq_1 + q_2dq_3 - q_3dq_1)) \quad (3.7)$$

Using velocity vector, gravity vector, equation (3.2) and equation (3.5) velocity update equation can be written as follows:

$$v_{New} = [v_n; v_e; v_d] + R((a - a_b)dt + [g_n; g_e; g_d]dt \quad (3.8)$$

By integrating above velocity, position update equation can be written as follows:

$$p_{New} = [p_n; p_e; p_d] + [v_n; v_e; v_d]dt \quad (3.9)$$

Angular rate bias update equation and acceleration bias update equation can be written as follows:

$$\omega_{bnew} = (\omega_{xb}; \omega_{vb}; \omega_{zb}) \quad (3.10)$$

$$a_{bnew} = (a_{xb}; a_{yb}; a_{zb}) \quad (3.11)$$

Covariance prediction:

Process equation: $f = (q_{new}; v_{New}; p_{New}; \omega_{bnew}; a_{bnew})$ (3.12)

State vector:

$$x = (q_1; q_2; q_3; q_4; v_n; v_e; v_d; p_n; p_e; p_d; \omega_{xb}; \omega_{yb}; \omega_{zb}; a_{xb}; a_{yb}; a_{zb}) \quad (3.13)$$

According to equation (1.13) and using equation (3.12) and (3.13) state transition matrix can be written as follows:

Therefore, State transition matrix: $F = \frac{\partial f}{\partial x}$ (3.14)

Control vector: $u = (\omega;\ a)$ (3.15)

Control influence matrix: $G = \frac{\partial f}{\partial u}$ (3.16)

Noise vector (IMU noise):

$$N = diag(daxCov\ dayCov\ dazCov\ dvxCov\ dvyCov\ dvzCov) \quad (3.17)$$

Process noise: $Q = GNG^T$ (3.18)

According to equation (1.6) and using equation (3.14), (3.15), (3.16), (3.17) and (3.18) covariance matrix can be written as follows:

Therefore, covariance matrix: $P_{new} = FPF^T + Q$ (3.19)

Update equations:

Filter state correction step was explained in here. Velocity and position states were going to be corrected first. There are six direct measurements (observations) available in this system, GPS North and East velocities and positions measurements and Barometer height and climb rate measurements. Therefore, unlike attitude, velocity and position innovation can be found directly. However, due to measurement lag on GPS and the Barometer measurement fusion cannot be done sequentially. Therefore, we store predicted states with corresponding time-span in an array until new measurement is available. If new GPS/Barometer measurement is available then we call to an array

which stored in previously estimated states and select a state which should have close timespan to corresponding measurement. States correction for velocity and position will be explained in here only for one axis (North axis) and states correction for other axis (East and Down) will be identical to these steps.

GPS measurements vector: $m_G = (V_X, V_Y, P_X, P_Y)$ (3.20)

Barometer measurements vector: $m_B = (V_z, P_Z)$ (3.21)

State Correction step (measurement fusion) for v_n and only GPS measurement V_X is used in here.

According to equation (1.7) innovation update equation can be written as follows:

Innovation: $y_v = V_X - v_n$ (3.22)

According to equation (1.14) measurement Jacobian can be written as follows:

Measurement Jacobian: $H_v = \frac{\partial v_n}{\partial x}$ (3.23)

According to equation (1.9) and using equation (3.23) innovation covariance update equation can be written as follows:

Innovation covariance:

$S_v = H_v P H_v^T + R_v$(R_v = GPS Velocity observation noise) (3.24)

According to equation (1.10) and using equation (3.19), (3.23) and (3.24) Kalman Gain can be calculated as follows:

Kalman Gain: $K_v = P H_v^T S_v$ (3.25)

According to equation (1.11) (1.12) and using equation (3.19), (3.22) and (3.25) update state and covariance can be written as follows:

Update state: $$x_{new} = x + K_v y_v \quad (3.26)$$

Update covariance: $$P_{new} = (I - K_v H_v)P \quad (3.27)$$

Attitude measurements (Observations) are not directly available. Therefore, to correct attitude (quaternion), body frame reference predicted magnetic flux value was estimated and compare it with magnetometer flux value to find innovation.

Body frame reference predicted magnetic flux value:

$$predictedMag = R^T\ (magN; magE; magD) \quad (3.28)$$

$magN, magE, magD$ are earth frame reference magnetic fields and they can be found using initial body frame reference magnetic field and initial R.

According to equation (1.8) and using equation (3.28) innovation update equation for attitude can be written as follows:

Innovation: $$y_{mag} = mesuredMag - predictedMag \quad (3.29)$$

According to equation (1.14) and using equation (3.28) measurement Jacobian can be written as follows:

Measurement Jacobian: $$H_{mag} = \frac{\partial predictedMag}{\partial x} \quad (3.30)$$

According to equation (1.9) and using equation (3.30) innovation covariance update equation can be written as follows

Innovation covariance: $$S_{mag} = H_{mag} P H_{mag}^T + R_{mag} \quad (3.31)$$

(R_{mag} = Magnetometer observation noise)

According to equation (1.10) and using equation (3.19), (3.30) and (3.31) Kalman Gain can be calculated as follows:

Kalman Gain: $$K_{mag} = PH_{mag}^{T}S_{mag} \quad (3.32)$$

According to equation (1.11) (1.12) and using equation (3.19), (3.29) and (3.32) update state and covariance can be written as follows:

Update state: $$x_{new} = x + K_{mag}y_{mag} \quad (3.33)$$

Update covariance: $$P_{new} = \left(I - K_{mag}H_{mag}\right)P \quad (3.34)$$

These equations are initially simulated using MATLAB symbolic Tool-box and then it converted to C++ language which can be run on flight controller board with Linux operating system. Then the filter was tested in real-time using actual sensor signals. Results of these experimentations are described in next section.

3.1.3 Hardware selection and assembling of the Quadcopter

There are several components consists with quadcopter. Mainly flight control board, Motors, Motor speed control unit (ESC), Propellers, Radio frequency communication unit and Radio control unit (Remote Controller). Selected component and reason to select those components are described in below.

Flight controller

When it comes to flight controller board it is the brain of the quadrotor system. This consists with main processing unit which is run flight control algorithm and also it consists with necessary sensors like Inertial measurement unit (IMU) which includes 3-axis mems accelerometer measures accelerations, 3-axis mems gyroscope measures angular rates, 3-aixs magnetometer also called compass which is sense the earth magnetic field, Barometer which is sense the atmospheric pressure, GPS Receiver and also includes serial ports, I2C ports, SPI Ports to attach necessary external sensors. In here I selected Navio2 flight shield with Raspberrypi 3 small computer which is run Linux real-time operating system. Navio2 flight shield includes MPU 9250 sensor with 3-axis accelerometer, 3-axis gyroscope, 3 axis magnetometer MS5611 barometer, M8N GPS Receiver, and also includes PWM output pins for motor control, I2C port and

Serial port. However, Raspberrypi board responsible for run all control algorithms, read sensor data from Navio2 flight sensor shield and output commands to the motors.

Figure 3.1 Navio2 flight shield connected with Raspberrypi 3 computer board.

Motors and Electronic speed control unit (ESC)

Motors used in most multi-rotors are brushless dc motors. These motors are categorized based on their KV rates. KV rates means increment of the rpm of the motor due to change of 1 volt. Normally motors with higher KV rates can produce low thrust and higher rpm while motors with lower KV rates can produce high thrust and low rpm. Therefore, high KV rate motors are suitable for heavy vehicle while low KV rate motors are suitable for light weight situations. When it comes to ESC they are categorized based on their current they can handle. ESC with higher current rates are suitable for motors with low KV rates. ESC converts DC voltage into 3-phase AC voltage using MOSFET. In here KV rate with 850 brushless dc motor with 20Amp ESC was selected.

Figure 3.2 Selected 850KV brushless dc motor and 20Amp ESC

Remote controller

To give control commands to quadcopter like Throttle Roll Pitch Yaw command we need remote controller. Most of the time these are working with radio frequency range around 2.4GHz. These are categorized based on operating channels like 4 channel radios, 8 channel radios etc. In here 8 channel radio was selected. It has 4 main channels (Throttle, Roll, Pitch and YAW) and four auxiliary channels. Remote controller work as a transmitter and to capture this transmitted signal quadcopter needs a receiver. Receiver also has eight output channels.

Figure 3.3 Selected transmitter and receiver

Radio telemetry

To receive data like position velocity and attitude information or to send control commands from quadcopter to laptop, tablet or vice versa we need this radio module. Unlike Remote controller, this device is capable of two-way communication. That means one radio module can transmit data and at the same time, it can handle received data. Most of the time working frequency of those modules is 433MHz.

Figure 3.4 Selected Radio telemetry

Battery

Most of the time batteries used in multi-rotors are lithium-polymer. Because they can handle high discharge rates. Batteries categorized based on their number of cells like 2cell 3cell etc. In here I used 4cell 14.8V with capacity 3300mAh battery.

Figure 3.5 Selected lithium-polymer battery

Final assembly of the quadcopter

Figure 3.6 and Figure 3.7 shows final assembly of the system. Motors were connected to ESCs and then ESCs are powered using battery. PWM signal wires of the ESCs are connected to PWM output pin of the Navio2 flight shield and also remote-control receiver was connected to PWM input pin. Finally, radio telemetry module was connected via serial port of the flight shield and external antenna also connected to GPS receiver.

Figure 3.6 Actual hardware set-up of the system

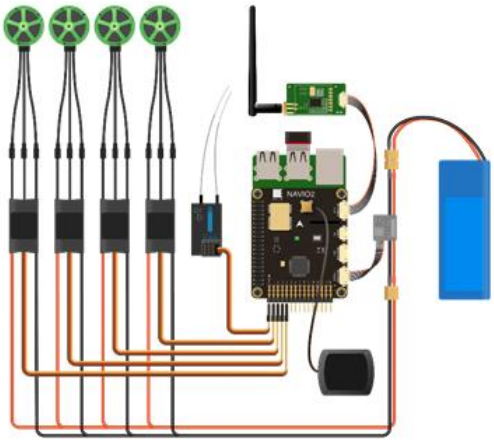

Figure 3.7 Connections of the hardware components

3.1.4 Experimental results of the filter

In here Kalman Filter estimated position and velocity compared with GPS position and velocity. However, Kalman Filter estimated attitude cannot be compare because quad-copter attitude cannot be measured directly. Experiment was done in outdoor while moving the setup, the GPS position of Latitude and Longitude (North-East axis is positive) directions and Kalman filter estimated position of Latitude and Longitude (North-East axis is positive) directions were recorded every 5ms and plotted as shown in Figure 3.8, Figure 3.10. Several experiments were conducted while changing the process noise (Accelerometer noise) and measurement noise (GPS noise). Only most successful experiment result was shown in below. According to Figure 3.8 and Figure 3.10 Kalman filter estimated position is followed by GPS measured position. However, Kalman filter estimated position is considerably lead compare to GPS measured position according to Figure 3.9 and Figure 3.11 which are zoom version of the Figure 3.8 and Figure 3.10. In here GPS position update rate is 5Hz but Kalman Filter position update rate is 400Hz because IMU update frequency is also 400Hz. Therefore, Kalman Filter estimated position is 80times faster than GPS position. According to Figure 3.9 and Figure 3.11 position variation of the Kalman filter is smoother than GPS position estimation. However, there are little jumps on Kalman filter estimated position due to fusion step perform in every 200ms.

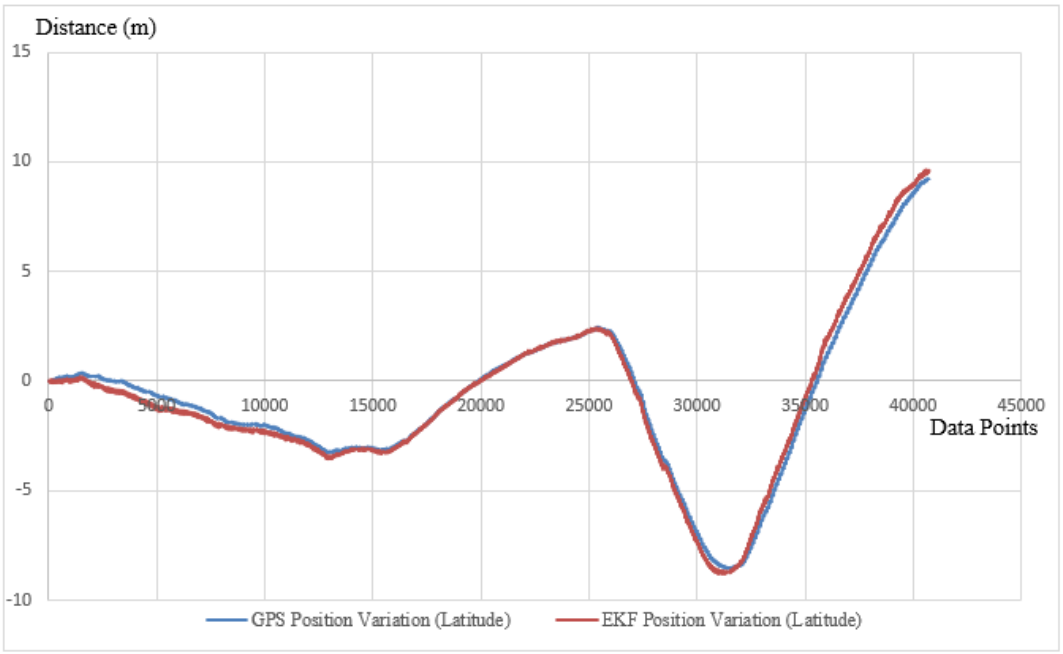

Figure 3.8 GPS position vs EKF position for Latitude direction
(Accelerometer process noise =0.25m/s^2 and GPS position noise= 0.5m)

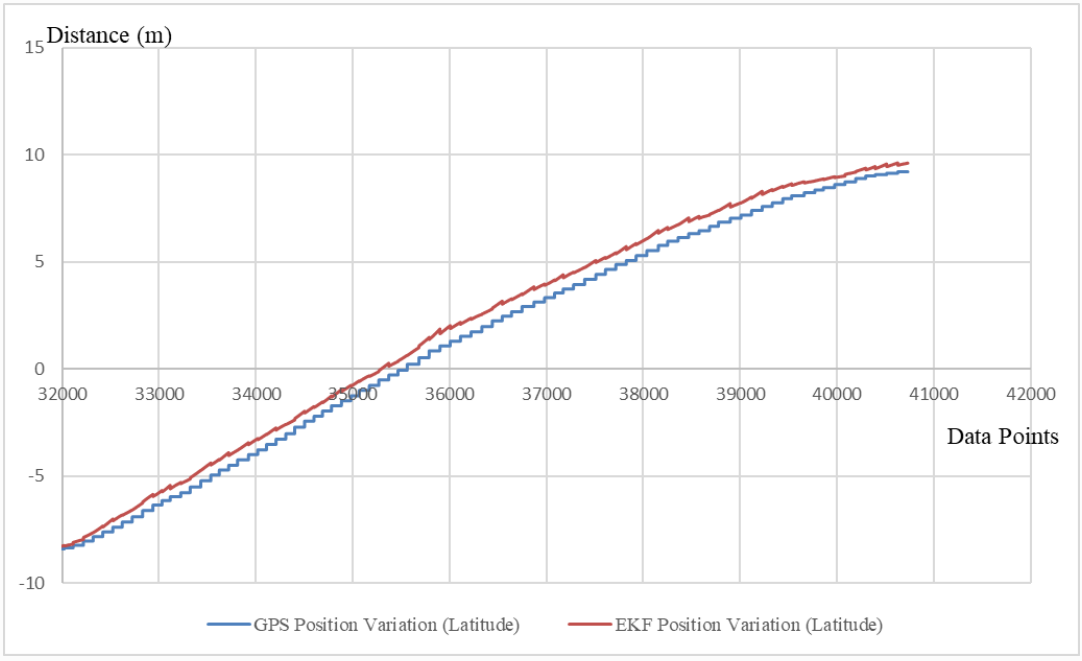

Figure 3.9 Zoom version of Figure 3.8
GPS position vs EKF position for Latitude direction
(Accelerometer process noise =0.25m/s^2 and GPS position noise= 0.5m)

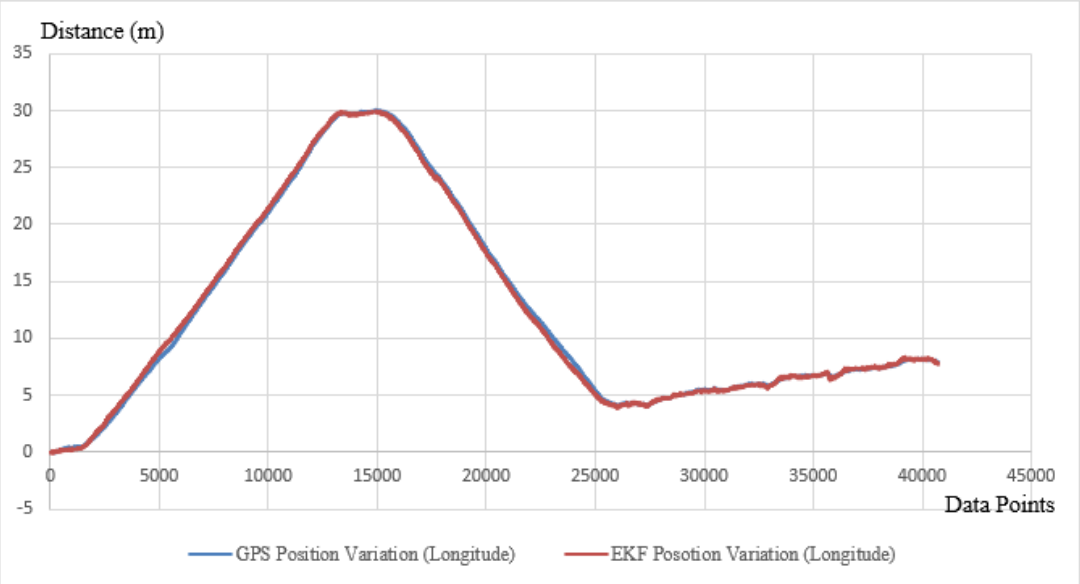

Figure 3.10 GPS position vs EKF position for Longitude direction
(Accelerometer process noise =0.25m/s^2 and GPS position noise= 0.5m)

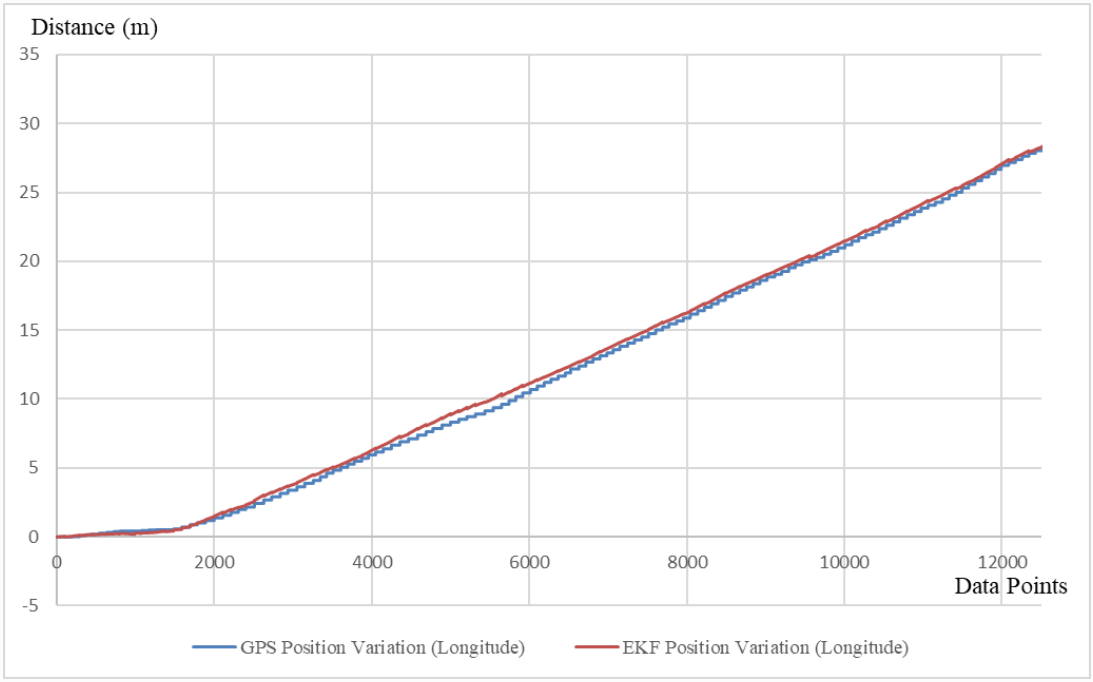

Figure 3.11 Zoom version of Figure 3.10
GPS position vs EKF position for Latitude direction
(Accelerometer process noise =0.25m/s^2 and GPS position noise= 0.5m)

Velocity comparison experiment was done by applying sudden disturbance to vehicle. Same as position comparison experiment in here also data was recorded every 5ms and then plotted. Figure 3.12 and Figure 3.13 shows how EKF estimated Velocity variation with GPS measured velocity. According to Figures, EKF estimated velocity lead compare to GPS velocity. Same noise covariances were used in here and EKF velocity update rate is 400Hz. Fusion step only applied when new GPS data is available.

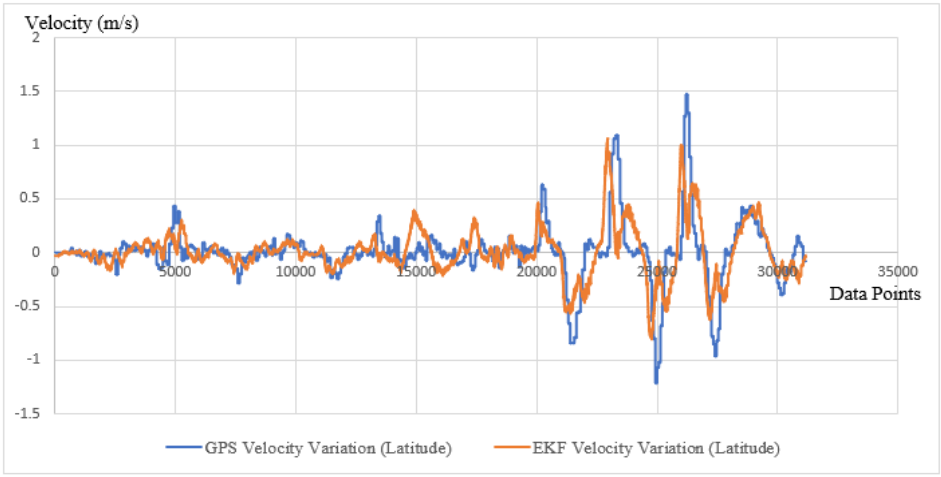

Figure 3.12 GPS velocity vs EKF velocity for Latitude direction
(Accelerometer process noise =0.25m/s^2 and GPS velocity noise= 0.5m/s)

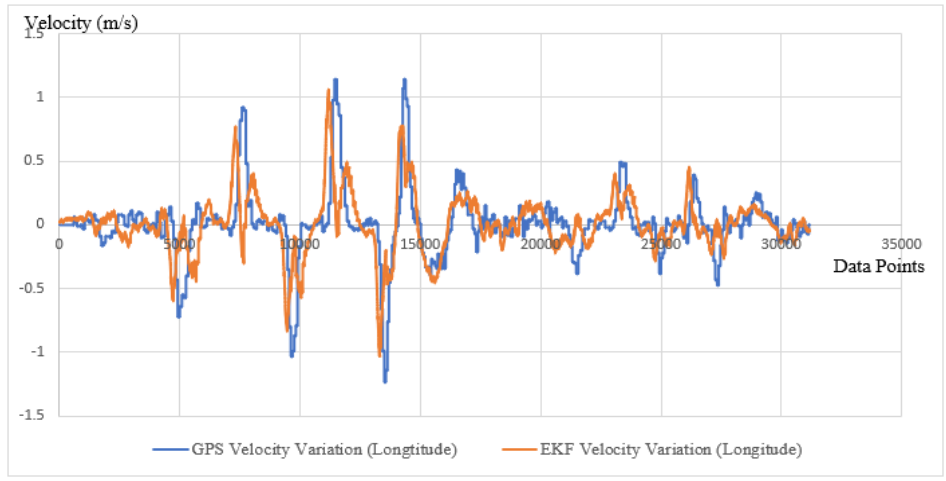

Figure 3.13 GPS velocity vs EKF velocity for Longitude direction
(Accelerometer process noise =0.25m/s^2 and GPS velocity noise= 0.5m/s)

Vertical velocity and height variation of the EKF was plotted against barometer climb rate and height as shown in Figure 3.14 and Figure 3.15. Unlike GPS, barometers are measures atmospheric pressure, because of this it contains highly noisy measurements and also measurement lag is around 50ms and its low compare to GPS. However, EKF estimated height and climb rate overcome these problems and update rate was increased from 10Hz to 400Hz.

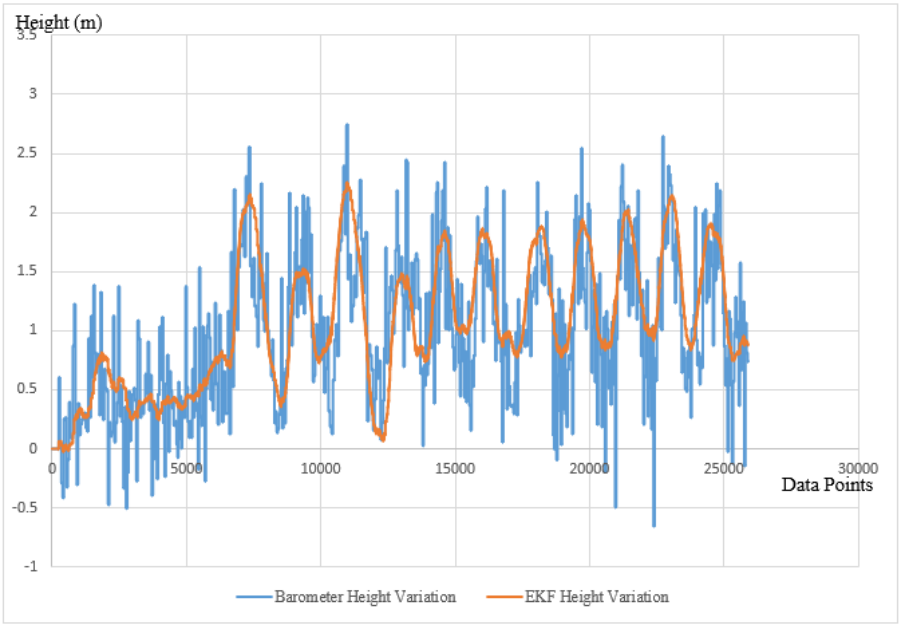

Figure 3.14 Barometer height vs EKF height

(Accelerometer process noise =0.25m/s^2 and Barometer height noise= 2m)

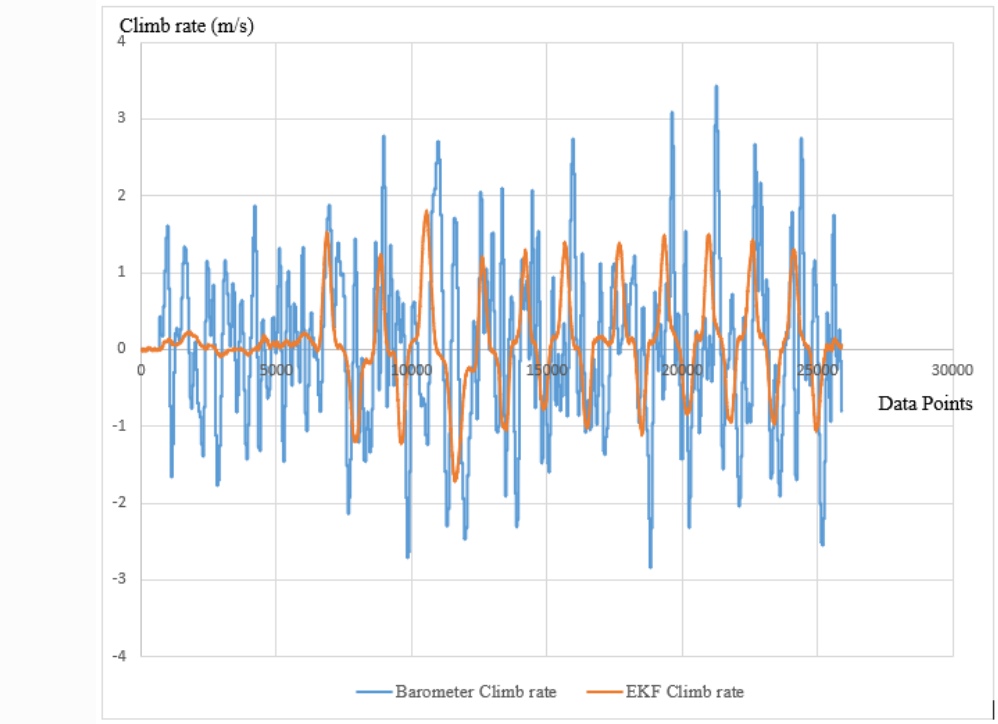

Figure 3.15 Barometer climb rate vs EKF climb rate

(Accelerometer process noise =0.25m/s^2 and climb rate noise= 1.5m/s)

3.1.5 Introduce Extended Kalman filter outputs to flight control algorithm

The chosen Attitude and Position control architectures are shown in Figure 3.16 and Figure 3.17. Theoretical development of this attitude controller was described in this paper [2] and implementation and experimental results were described in this paper [1]. When it comes to attitude control of the system, actual attitude information was extracted from the Extended Kalman filter and desired attitude information were provided by the pilot using a remote controller. Error between these two are the attitude error which fed into the attitude PID controller. PID controller then generates the thrust which should produce by each of the motors.

Both controllers work as a cascade system as shown in Figure 3.17. Quadcopters are underactuated systems, therefore they cannot be control using single controller. Manual flight only requires attitude controller but when it comes to autonomous flight of the vehicle there must be need of position controller.

Input to the position controller is actual position and velocities which extracted from EKF and desired position information given by pilot via remote controller. Output of the position controller is desired Roll angle, Pitch angle and Thrust. Then this information is fed into the attitude controller to maintain the desired position of the vehicle.

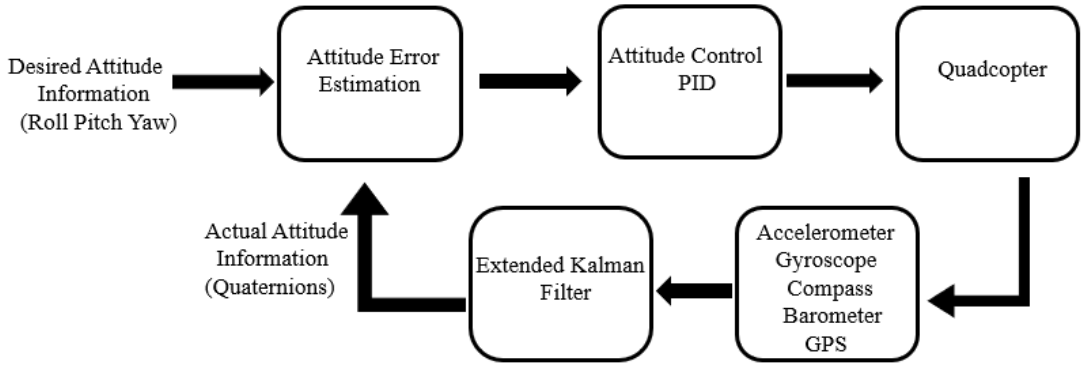

Figure 3.16 Attitude control of quadcopter

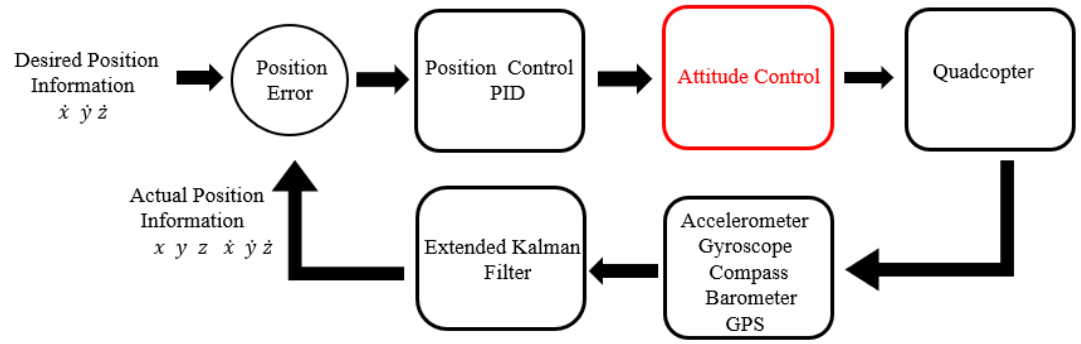

Figure 3.17 Position control of quadcopter

3.2 Development of the stereo vision system

3.2.1 Selection of the stereo camera

Stereo camera consists with two similar cameras. To get better performance from the system camera should be identical and camera alignment should be perfect. But in reality, cameras even with same brand won't be identical. However, in here two normal USB web cameras as shown Figure 3.18 were used to build the stereo camera as my first experiment. This camera can provide up to 720-pixel (1280x720) video resolution with 30 frames per second also view angle is only up to 60° with manual focus lens. Most of the situations distance between optical centers (Baseline distance) of the stereo camera is in the range between 10-15cm therefore, here I placed both cameras 12cm apart as shown Figure 3.18. Then both cameras were connected to Navio2 + Raspber-rypi flight controller as shown in Figure 3.19.

Figure 3.18 Normal USB web camera.

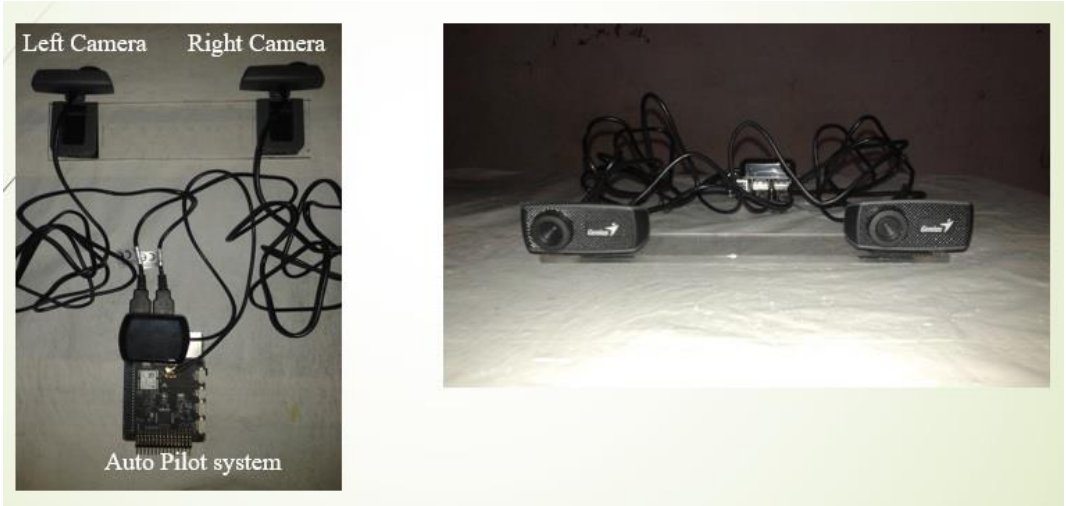

Figure 3.19 Stereo camera build using two normal USB web camera

However, when creating depth images using above mention stereo camera, several problems occurred as follows:

1. Camera Misalignment.
2. Image synchronization problem.
3. Narrow View Angle (lower than 60°).
4. Low frame rate (VGA video 30fps).
5. Low computing power. (Depth calculation using VGA resolution approximately takes 250ms).

Due to manual installation of cameras, there are severe camera misalignment occurred which can't be compensated using camera calibration and image rectification. This camera communicates with flight controller using two USB wires and due to this, most of the time image buffering problem occurred either one of cameras. Therefore, depth image calculation goes wrong on due to this problem. Narrow view angle creates depth images with less information about obstacles. Depth calculation using VGA resolution approximately takes 250ms. This also caused more serious problem. Because this refresh rate cannot be acceptable in highly dynamic robots like quadcopters.

To get rid of above mention problems then I moved to stereo camera called ZED and computing platform called Nvidia Jetson TX1 which has lot of advantages as follows:

1. No Camera Misalignment.
2. No Image synchronization problem.
3. Wide View Angle (Up to 110° View Angle).
4. Fast frame rate (Capture 720p video with 60fps or VGA video with 100fps).
5. More computing power with GPU. (NVIDIA Maxwell™ GPU with 256 NVIDIA® CUDA® Cores).
6. Quad-core ARM® Cortex®-A57Processor. (Depth calculation with 720p resolution approximately takes 75ms).

Compared to other USB web cameras, the ZED has a larger CMOS sensor that is ideal for low-light environments. It also has a low-latency USB performance which is suitable for the stereo vision application.

The ZED also has fixed focus lens with 110° field of view and the distance between the cameras is 12 cm. This will make information rich depth maps. When it comes to the Jetson TX1 Developer Kit, it is an ideal development platform for image processing. It is consists with a Linux operating system call Ubuntu and this makes Jetson TX1 ideal for computer vision applications such as computation of depth using stereo images.

Figure 3.20 ZED stereo camera with Nvidia Jetson TX1 computing platform.

3.2.2 Stereo camera calibration

According to Chapter 1, we have to perform Stereo camera calibration before computing the depth. Because almost all stereo cameras subject to camera lens distortion and misalignment as mentioned in Chapter 1. In stereo camera calibration we hope to find camera distortion coefficient as mention in equation (1.19), camera intrinsic parameters (Camera Matrix) as mentioned in equation (1.20) and camera extrinsic parameters. These parameters are expected to use in Image rectification and depth calculation steps. To stereo camera calibration, OpenCV (Open Source Computer Vision) was used. OpenCV is free open source software for computer vision applications. However, OpenCV is registered under BSD license therefore, it is free to use in research applications as well as commercial applications.

To calibrate the stereo camera OpenCV “StereoCalibrate” function was used. First, a 7× 10 chessboard pattern was printed on A3 sized paper and it was attached to a white color background.

Then place the chessboard pattern in front of the cameras so that stereo camera can read all 70 corners of the chess board. Once the calibration starts, we start to move and rotate the chessboard, allowing both cameras to see the chessboard from different viewpoints as shown in Figure 3.21. Before calling the function “StereoCalibrate”, “FindChessboardCorners” function was called to locate the corners’ of the chess board.

Finally, “StereoCalibrate” function take chess board information as input and compute distortion coefficient and intrinsic parameters of each camera and saved in a .XML file which was used in image rectification and depth calculations. Also, compute extrinsic parameters (rotation matrix and translation vector of the right camera compare to left camera) and saved in a separate .XML file. Figure 3.21 shows one moment of the stereo calibration process.

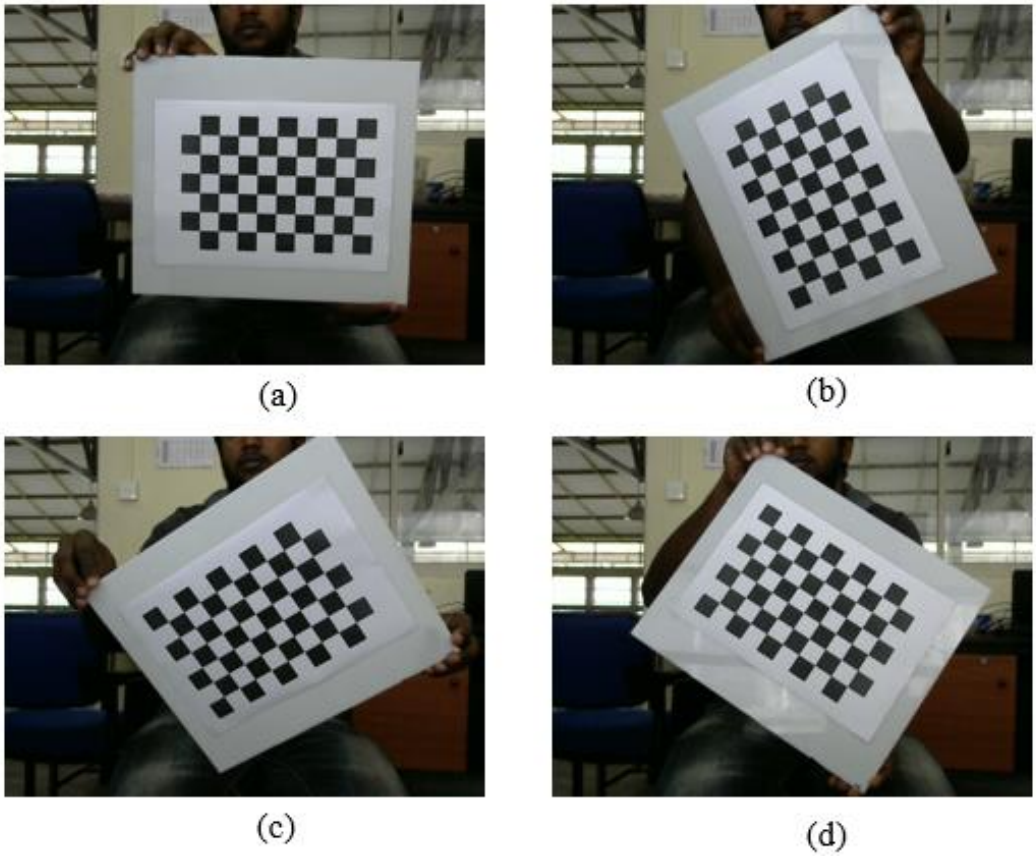

Figure 3.21 Stereo calibration using a 7 × 10 chessboard pattern

3.2.3 Image rectification

When it comes to image rectification process, OpenCV "StereoRectify" was used. Which takes, raw images, camera matrix and extrinsic Parameters (rotation matrix and translation vector) that are computed during the stereo camera calibration process as inputs and returns the rectification projections matrices P_l P_r for left and right camera as mention in equation (1.21) and equation (1.22). Then, instead of performing this calculation every time, OpenCV "Remap" function was used to rectify the images by using previously calculated projection matrices. Figure 3.22 shows the comparison between raw input images and rectified images, here we can see that raw images suffer from lens distortion effect. That means straight lines in real world appears as curved in raw images. However, after rectification is performed, distortion effect was removed and objects appear in rectified images closer to what in real world.

Figure 3.22 Raw images (Top) Rectified images (Bottom)

3.2.4 Disparity calculation

OpenCV inbuilt functions were used to calculate disparity images. OpenCV has mainly two stereo correspondence algorithms, stereo block matching (StereoBM) and Semi-Global block matching (StereoSGBM).

As mention in Chapter 1, the stereo block matching algorithm (StereoBM) is a local stereo correspondence algorithm that was first developed by Kurt Konolige produces a less accurate but very fast disparity map. The Semi-Global block matching algorithm (StereoSGBM) is a global stereo correspondence algorithm that was first developed by Heiko Hirschmuller [12]. Although it produces a highly accurate disparity map, it is somewhat computationally extensive but ideal for real-time purposes. Since we use Nvidia Jetson TX1 as our computing platform for image processing, StereoSGBM was selected as stereo correspondence algorithm. SGBM algorithm is use sum of absolute differences (SAD) matching cost function as mention in equation (1.23) When using SGBM algorithm, key parameters that need to be tuned are summarized as below:

SAD window size: This value controls the size of both the support window in the left image and the corresponding window in the right image. Window sizes are odd numbers like 3, 5 7 9, etc. A smaller window size decrease quality of the disparity map while reducing computational power and larger numbers provide smooth disparity maps while increasing the computational power.

Uniqueness ratio: This value controls the matching cost function margin value which is separate disparity values of two matching blocks.

Pre-filter size: This value controls filter value of image pixels intensity filter. Larger values give more smooth disparity image but the response is going to be slow.

Number of disparities: Difference between maximum and minimum disparity values. If we want to see closer objects clearly then we have to use larger value of number of Disparities and if we want to see things far away objects then we have to select smaller values of number of Disparities.

User interface

To understand the effect of above parameters and fine tuning of the disparity image OpenCV “CreateTrackbar” function was used. It creates a slider for each parameter and value can be change using those sliders. Figure 3.23 shows the user interface together with the disparity map computed using the SGBM algorithm.

Disparity values are varying between 0 – 255. 0 means closer objects and 255 means far-away objects. Changes of pre-filter size and uniqueness ratio not much affect to quality of the disparity image but changing SAD window size and Number of disparities gives significant results as mention in below.

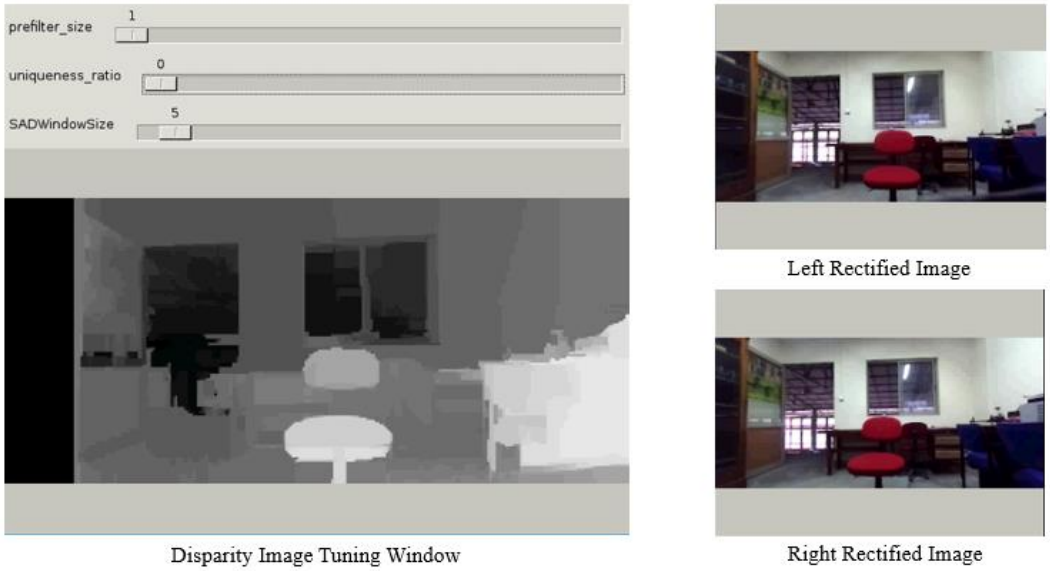

Figure 3.23 User interface for block matching algorithm

Effect of changing number of disparities

Lower value caused far-away objects noisy but higher value caused far-away objects smoother.

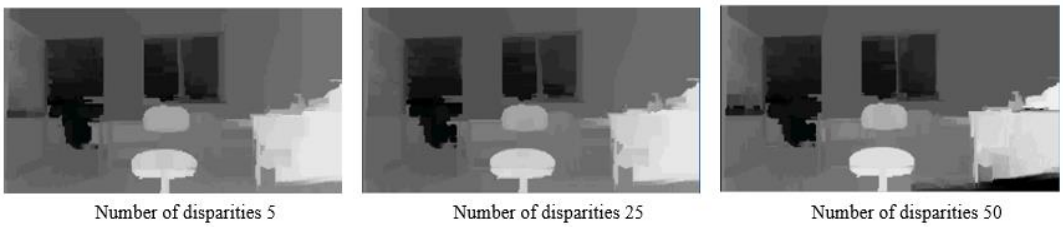

Figure 3.24 Effect of changing number of disparities.

Effect of changing SAD window size

Smaller SAD gives more noise but captures object contours better. Bigger SAD window size gives smoother image but loses contour recognition of objects.

Figure 3.25 Disparity maps for different SAD window sizes

When we have disparity map, calculation of the depth image is straight forward. Depth was calculated according equation (1.30) which is mention in the Chapter 1. According equation (1.30) to calculate the depth of a point we need three parameters. Disparity value of that point, focal length of the left or right camera (In here left camera frame considered as reference) and baseline distance of stereo camera. Disparity values were calculated in previous section, focal length of the each camera was found in camera calibration process as intrinsic parameters and finally baseline distance of camera can be calculated using optical centre values which are also found on camera calibration process. Therefore we know all three parameters and depth values for each pixels were calculated. Update rate of depth image with VGA (640x480) resolution raw images takes average 40ms and with HD (1280x720) resolution raw images take average 75ms.

In this case VGA resolution was selected because it is considerably fast compared to HD resolution and quality of depth image was enough to get information about obstacles. Rectified raw images with VGA resolution and corresponding depth image was shown in Figure 3.26.

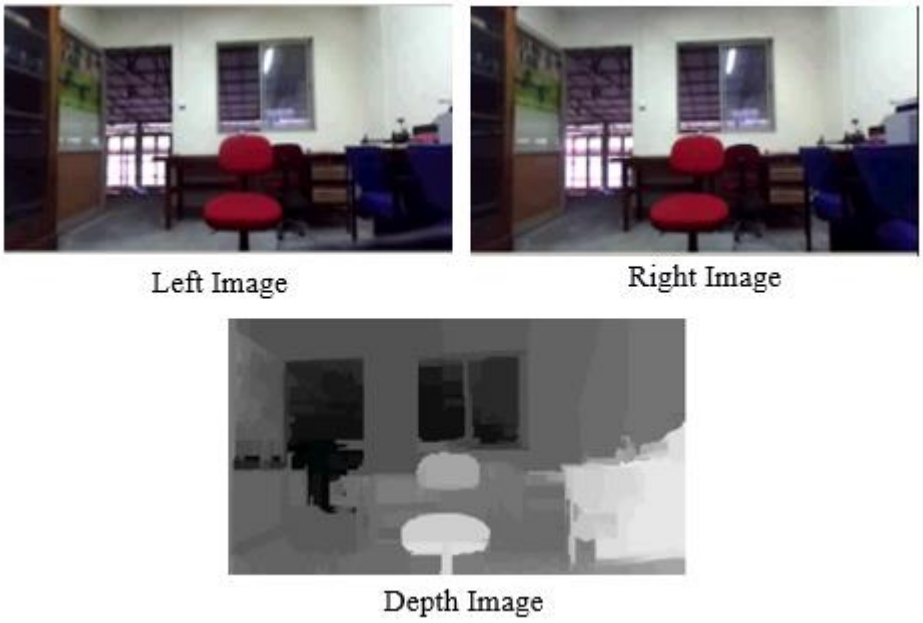

Figure 3.26 Input raw images and corresponding depth map

3.2.5 Manual calibration of the depth image

However, calcultated depth in above step may not represent the true depth values observed in the real world due to slight deviation of focal lengths and optical center distance of each camera from real value. Therefore manual calibration step was performed as shown in Figure 3.27. A series of images were recorded using the stereo camera and OpenCV function was used to get depth of the point which we interest by clicking on the raw image. This point is marked in red colour as shown in Figure 3.27(a). True distance between this marked point and the stereo camera was also measured. This procedure was performed for several points and average factor was found. This factor was then used to correct computed depth. Moment of this process shown in Figure 3.27.

According to Figure 3.27(a) average distance value from camera to marked red dotted point is 1.55m, but according to Figure 3.27(b) measured depth is 1.6m. According to Figure 3.27(c) average distance given by depth image was 2.50m and according to Figure 3.27(d) measured depth was 2.57m. So this procedure was followed by ten different situations and found an average scale factor to correct the depth values of the depth image.

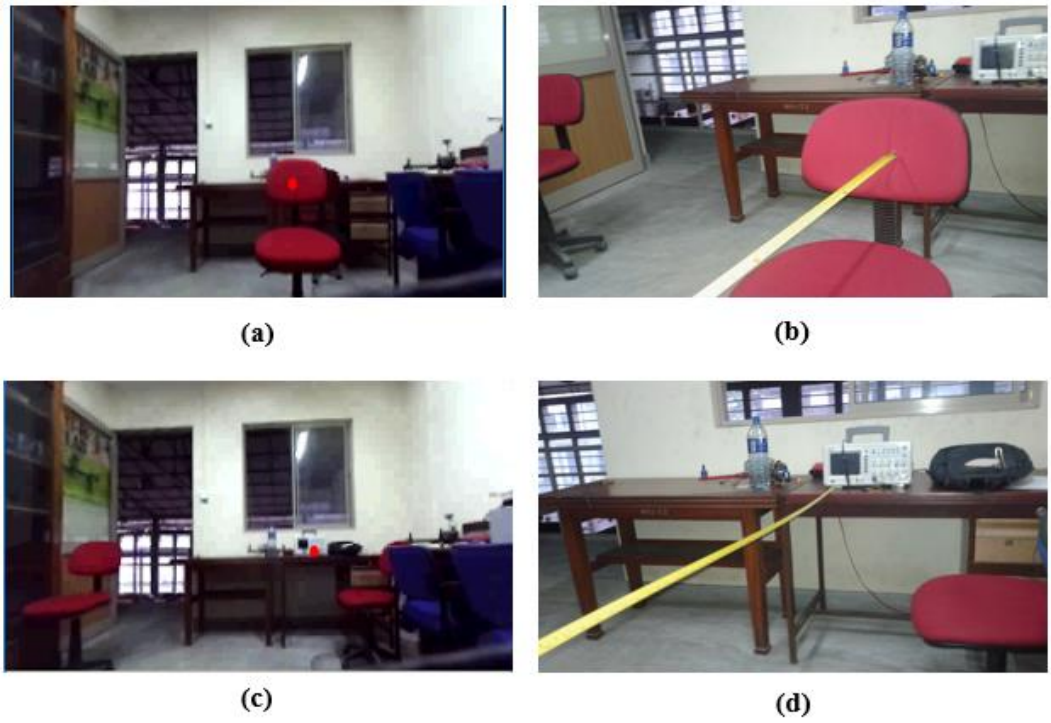

Figure 3.27 Moment of manual calibration of depth image

3.2.6 Testing of stereo vision system on outdoor environment

Unlike indoors, outdoor environments are subjected to varies lightning conditions. Due to variations of the sunlight may cause problems in depth image. Therefore, stereo vision system was tested in outdoor and parameters are being tuned in varies lightning conditions to get better results. Some results are described in this section.

Here I focused only on changing number of disparities and SAD window size in outdoor environment. As I described in previous, changing of per-filter size and uniqueness ratio in outdoor environments not much effect on quality of depth image. But changing number of disparities caused to somewhat clear image in far-away objects. But it also caused to reduce the quality of the closer objects. When it comes to SAD window size, increasing its size caused to reduce the noise of the depth image but reduce the quality of objects contours. In this application we interested only closer objects therefore, low number of disparities and low SAD window size were selected and these values are not much differed from indoor environment.

Therefore, same values were decided to use in both indoor and outdoor environments. Figure 3.28 shows some experimentation on outdoor environment considering different situation.

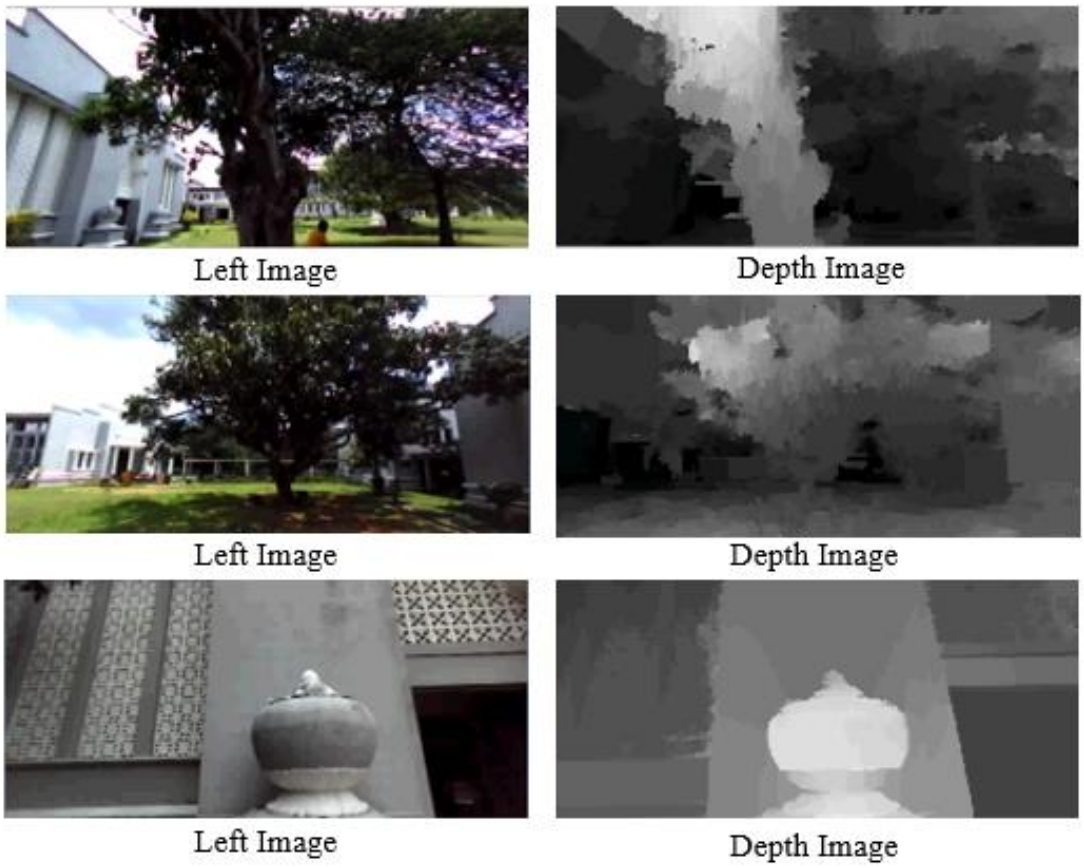

Figure 3.28 Quality of the depth image in different scenarios

3.3 Development of the fuzzy decision-making system

3.3.1 Analyzing the Depth Image

Lazaros et al [9] presented an obstacle avoidance method which is successfully applied to mobile robot by dividing depth map into three separate regions. However mobile robot only have two degrees of freedom therefore, dividing depth map into three separate regions is enough but when it comes to quadcopter it has three degrees of freedom simply up-down left-right and forward-backward. Therefore depth map was divided into nine separate regions namely, Centre, Centre-Up, Centre-Down, Center-Left, Centre-Right, Up-Left, Up-Right, Down-Left and Down-Right as shown in Figure 3.29.

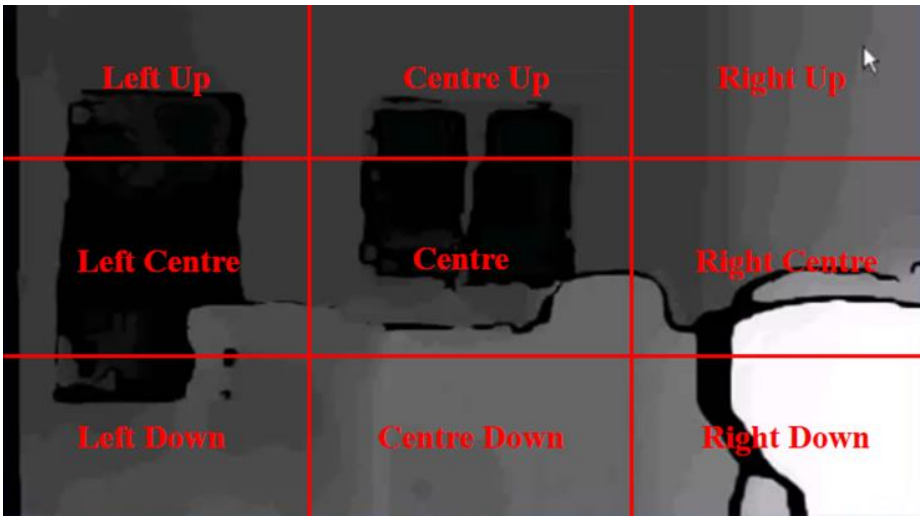

Figure 3.29 Depth map's division in nine windows.

3.3.2 Calculation of size of sub regions

Size of each region depends on the stereo camera field of view, height and width of the quadcopter as shown in Figure 3.30. According to equation (3.35), if we know camera horizontal field of view (θ) distance between camera and image plane (distance of camera to end of blind region can be neglected because baseline distance is smaller than 1.0m) then we can calculate width (x) and height (h) of visible plane. According to camera data sheet, horizontal field of view is $110°$ for one camera and image width to height ratio is 16:9.

Although camera has 20m visible distance according to its data sheet here I only considered visible image plane with 5m distance (d) from camera. Since the maximum speed of the quadcopter is 5m/s therefore, it takes one second to move current image plan at its maximum speed. Processing speed of the fuzzy decision around 10ms and as mention in previous chapter processing time of the depth image around 40ms so total processing time is around 50ms. Therefore, 5m of distance (d) is enough for this situation.

Using these information, width and height of visible image plane was calculated and width is 14.28m and height is 8.01m. Therefore each sub region width is 4.76m and height is 2.67m as shown in Figure 3.31. However quadcopter width is 0.75m and height is 0.5m. Therefore, selected size for centre region is approximately five times larger than quadcopter size. Therefore, quadcopter can be safely move through centre region.

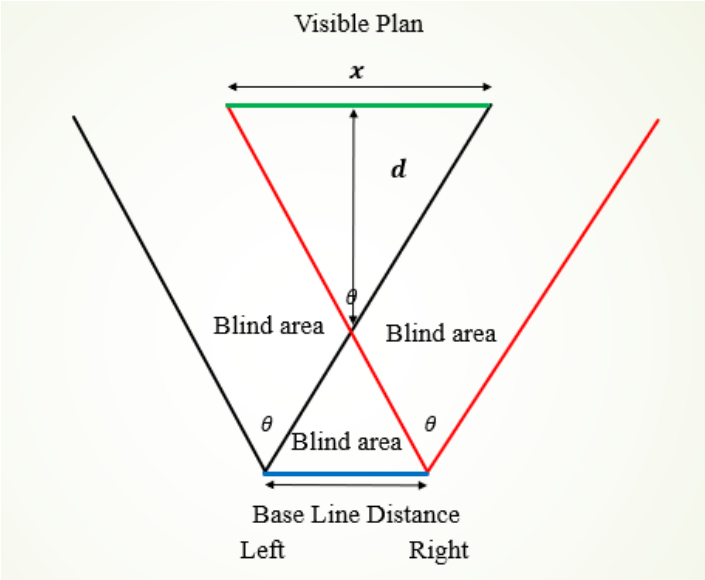

Figure 3.30 Visible image plane of the stereo camera

$$x = 2\ d\ tan(\frac{\theta}{2}) \tag{3.35}$$

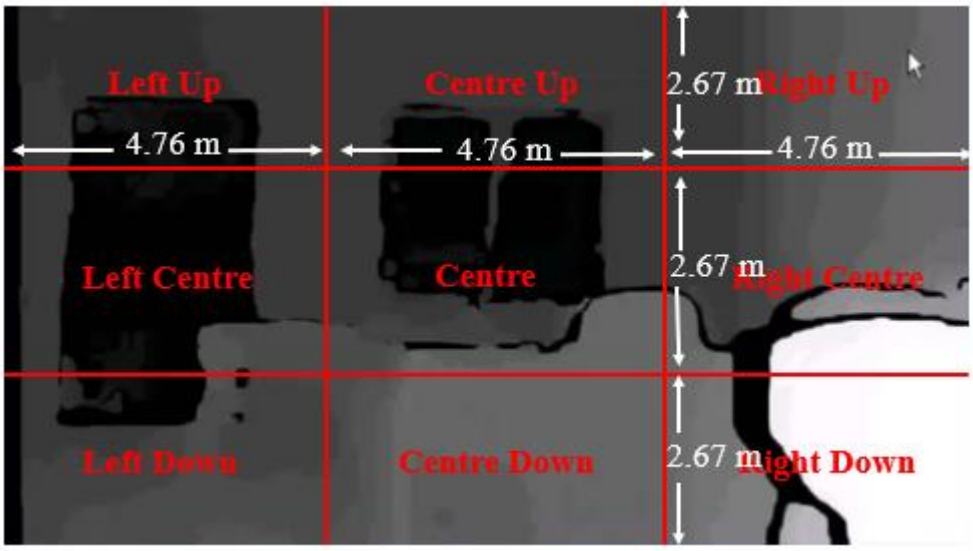

Figure 3.31 Dimension of each sub regions

3.3.3 Calculation of normalize depth values of sub regions

Considering quadcopter maximum speed 5m/s depth threshold was selected as 5m. That means objects smaller than 5m are considered as threats to the system. Reason to select this value is to improve the robustness of the system. If we select very smaller threshold value noisy signal can affect to decision-making system. If we select too large threshold value robustness of the system will decrease. However, each pixel in sub regions contains depth value and this value compare with threshold value. If depth value is greater than the threshold value, value of that pixel considered as one and otherwise zero. This procedure was then applied to all pixels in sub regions and take the summation of all values. This total value then normalizes throughout the sub window. This gives one single integer for each sub region and altogether nine values can be calculated. Then these nine values are fed into the fuzzy inference system as inputs.

3.3.4 Determination of fuzzy inputs and outputs

It is clear that the normalize depth values ($D_{lu}\ D_{lc}\ D_{ld}\ D_{cu}\ D_{cc}\ D_{cd}\ D_{ru}\ D_{rc}\ D_{rd}$) of the nine regions and vehicle desired speed (V_{in}) are input to the fuzzy inference system as shown in Figure 3.32. According to chapter 3 inputs to the position controller are desired positions of North-East-UP axis system.

However, instead of desired position, desired velocity in body frame can be estimated and then it can be converted into earth fame desired velocity. Integrating these desired velocities gives desired positions for position controller. Therefore, the output of the fuzzy system should be body frame velocity ($V_x, V_{y,} V_z$).

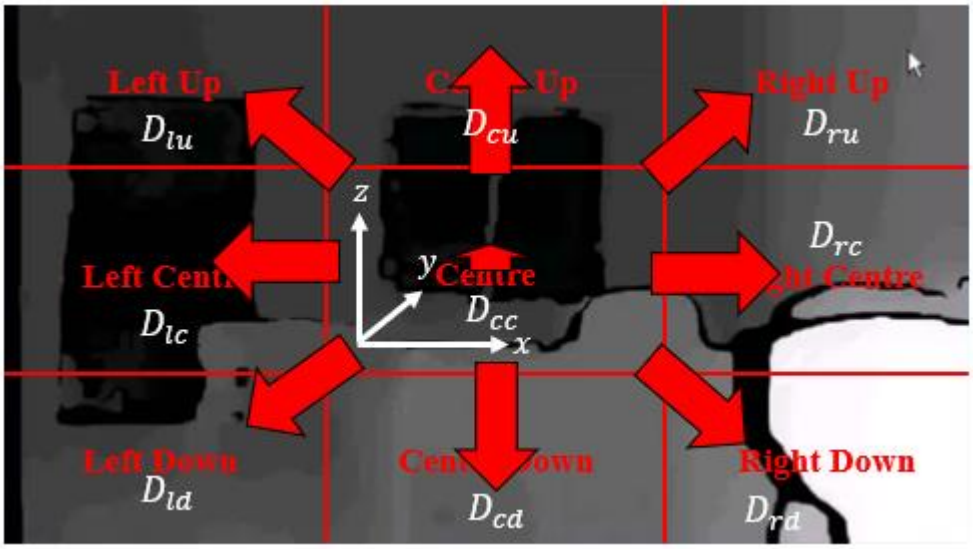

Figure 3.32 Navigation direction deciding based on depth image.

3.3.5 Determination of fuzzy rules

Next, fuzzy rules were determined which combined inputs and outputs of fuzzy system. Rules were defined based on values of each sub windows, quadcopter desired input velocity and output velocities. Overall there 18 rules were defined for the fuzzy controller as shown in Table 3.1.

Table 3.1: Fuzzy rule table

V_{in}	D_{lu}	D_{lc}	D_{ld}	D_{cu}	D_{cc}	D_{cd}	D_{ru}	D_{rc}	D_{rd}	V_x	V_y	V_z
Low	-	-	-	-	Low	-	-	-	-	Zero	Zero	Zero
Low	High	Low	Low	Low	High	Low	Low	Low	Low	SP	SN	SN
Low	Low	High	Low	Low	High	Low	Low	Low	Low	SP	SN	Zero
Low	Low	Low	High	Low	High	Low	Low	Low	Low	SP	SN	SP
Low	Low	Low	Low	High	High	Low	Low	Low	Low	Zero	SN	SN
Low	Low	Low	Low	Low	High	High	Low	Low	Low	Zero	SN	SP
Low	Low	Low	Low	Low	High	Low	High	Low	Low	SN	SN	SN
Low	Low	Low	Low	Low	High	Low	Low	High	Low	SN	SN	Zero
Low	Low	Low	Low	Low	High	Low	Low	Low	High	SN	SN	SP
High	-	-	-	-	Low	-	-	-	-	Zero	Zero	Zero
High	High	Low	Low	Low	High	Low	Low	Low	Low	LP	LN	LN
High	Low	High	Low	Low	High	Low	Low	Low	Low	LP	LN	Zero
High	Low	Low	High	Low	High	Low	Low	Low	Low	LP	LN	LP
High	Low	Low	Low	High	High	Low	Low	Low	Low	Zero	LN	LN
High	Low	Low	Low	Low	High	High	Low	Low	Low	Zero	LN	LP
High	Low	Low	Low	Low	High	Low	High	Low	Low	LN	LN	LN
High	Low	Low	Low	Low	High	Low	Low	High	Low	LN	LN	Zero
High	Low	Low	Low	Low	High	Low	Low	Low	High	LP	LN	LP

SN – Small Negative

SP – Small Positive

LN – Large Negative

LP – Large Positive

3.3.6 Selection of input and output membership functions

To guarantee smoothness of the fuzzy system gaussian type input and output membership functions were used in here. Input membership functions of Normalize depth values are defined as Low and High states which is vary between 0:1 as shown in Figure 3.33. Vehicle input desired speed also defined as Low and High states which is also vary between 0:1 as shown in Figure 3.34. Output membership functions of V_x and V_z are defined as Large Negative, Small Negative, Zero, Small Positive, Large Positive states which is vary between -1:1 as shown in Figure 3.35. Due to absence of positive values of V_y, membership function of V_y was defined as Zero, Small Negative and Large Negative as shown in Figure 3.36.

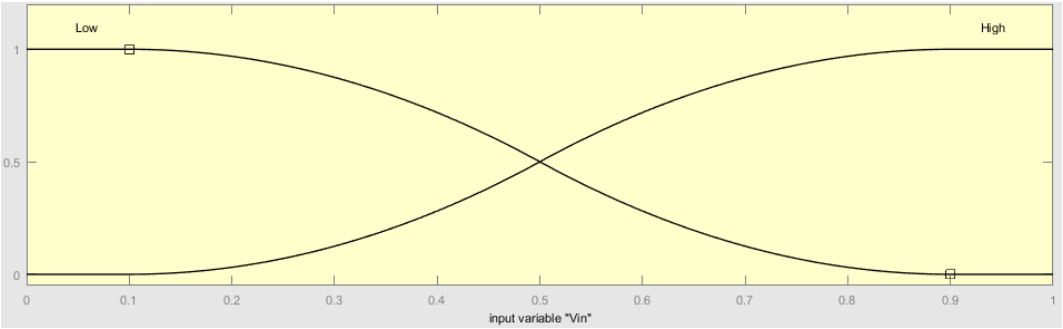

Figure 3.33 Input membership funcition of desired velocity input (V_{in}).

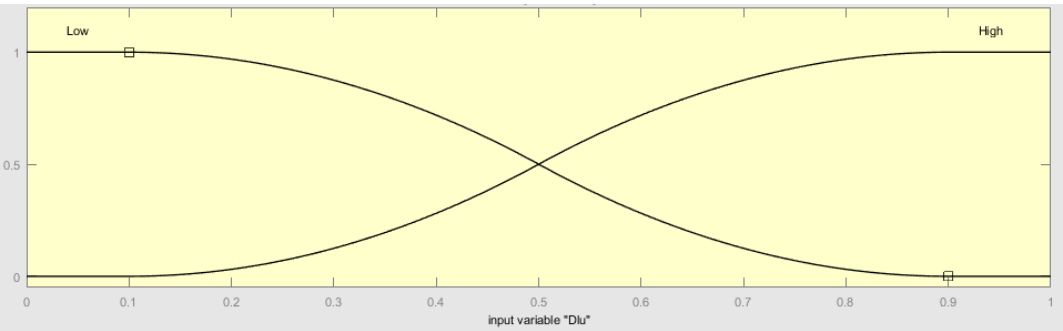

Figure 3.34 Input membership funcition of normalize depth value of one region (D_{lu})

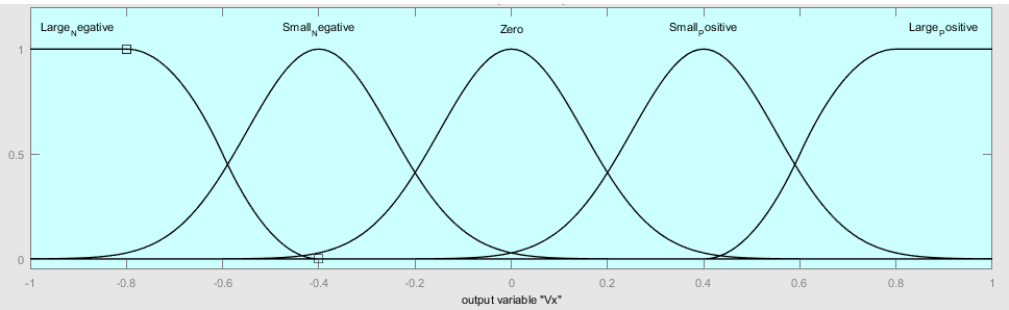

Figure 3.35 Output membership funcitions of V_x, V_z

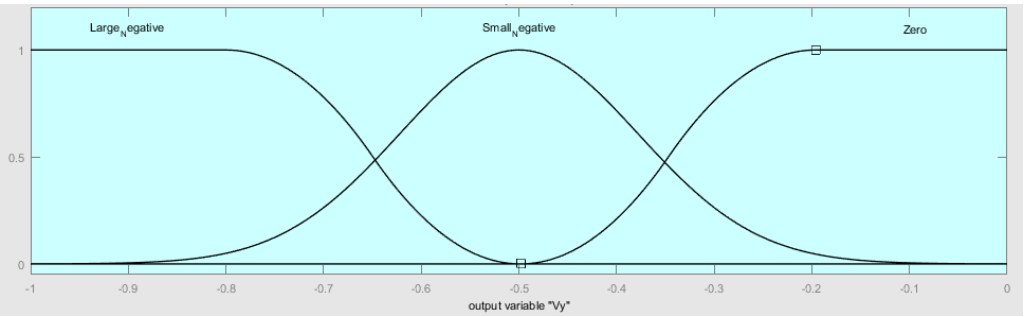

Figure 3.36 Output membership funcition of V_y

3.3.7 Selection of defuzziffication method

In here centroid, bisector, mean of maximum, sum of maximum and largest of maximum defuzzification methods were tried. Most significant results were provided by centroid method. Therefore, centroid method was used as defuzzify outputs. Finally, overall fuzzy inference system can be shown as Figure 3.37.

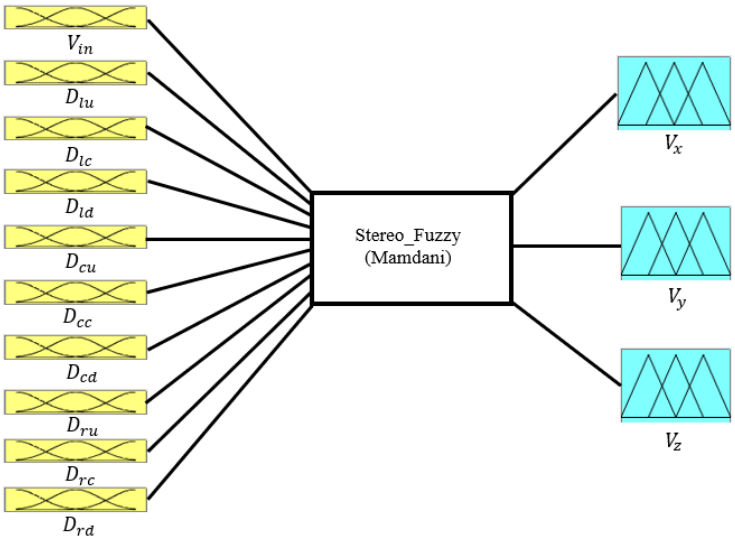

Figure 3.37 Fuzzy inference system

3.3.8 Introduce fuzzy inference system output to flight control algorithm

As shown in Figure 3.38 section drawn by red color is the obstacle avoidance system which is going to introduce to the flight control algorithm presented in [1] and [2]. Output of the stereo vision system is depth image and this depth image divide into nine sections and find corresponding normalize depth value for each sub region as mention in previous section. These nine values are inputs to the fuzzy inference system and according fuzzy rule base direction and magnitude of output velocities were decided. Then these body frame velocities were converted to earth frame velocities and fed into the position controller as desired velocities and by integrating these velocities, desired positions were estimated in North-East and up directions.

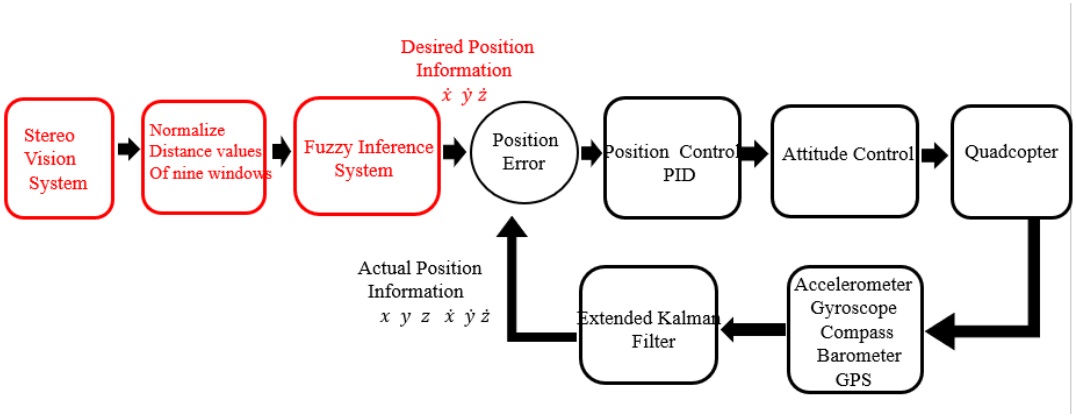

Figure 3.38 Combination of stereo vision system, fuzzy inference system with flight control algorithm

Fuzzy inference system initially developed and simulated using MATLAB Fuzzy Logic Tool box. MATLAB fuzzy system can't be run on Nvidia Jetson TX1 because Jetson TX1 runs on Linux operating system. So using fuzzylite library it was converted into the C++ environment which can be run on Linux environment. Entire stereo vision system runs on OpenCV which is compatible with Linux operating system. So finally both systems are runs on Nvidia Jetson TX1 as separate threads.

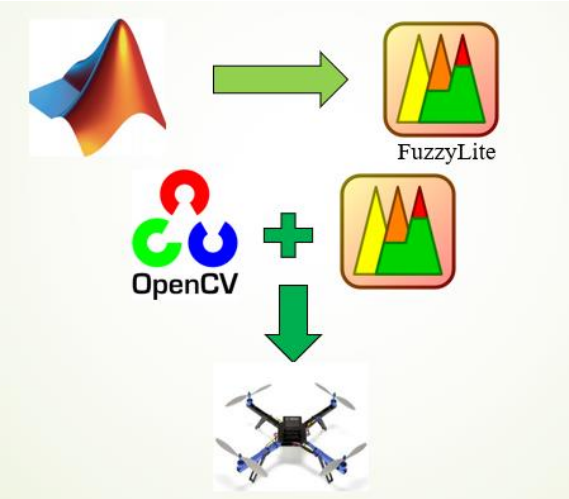

Figure 3.39 Code architecture runs on Nvidia Jetson TX1

3.3.9 Testing of the fuzzy system on outdoor environment

The Fuzzy decision-making system was tested on outdoor environment. Vehicle was gradually taken into an obstacle (tree) as shown in Figure 3.40 and simultaneously recorded normalize depth values of nine regions and output velocities of fuzzy decision-making system. Results were plotted as shown in Figure 3.41 and Figure 3.42 (Normalize depth values were plotted only for main sub regions in here). According to the Figure 3.41 high normalize depth values were appeared in all most all regions around 1050 data point and onward, at the same time vehicle forward velocity (V_y) responded as negative velocity. This can be expected because situation shown in Figure 3.40 almost impossible to avoid. Therefore, other two velocities (V_x V_z) almost zero during this time. V_z shows some spikes on the plot because noise occurred in normalize depth values during the test. To eliminate this problem a low pass filter was introduced to the system later. At the last stage of this test (data point around 1300 and onward) sudden decrease of left side normalize depth value (D_{lc}) caused V_x was responded as negative. But in this situation, it is impossible to avoid the obstacle (tree) and navigate around it. Therefore, these types of false situations need to be corrected in future developments.

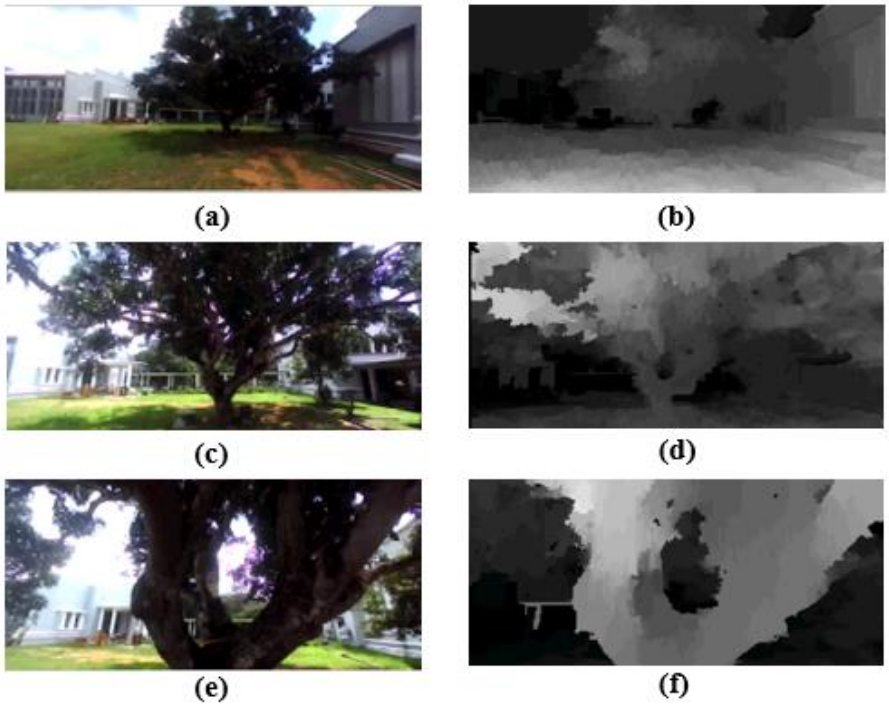

Figure 3.40 Outdoor testing of fuzzy decision-making system.

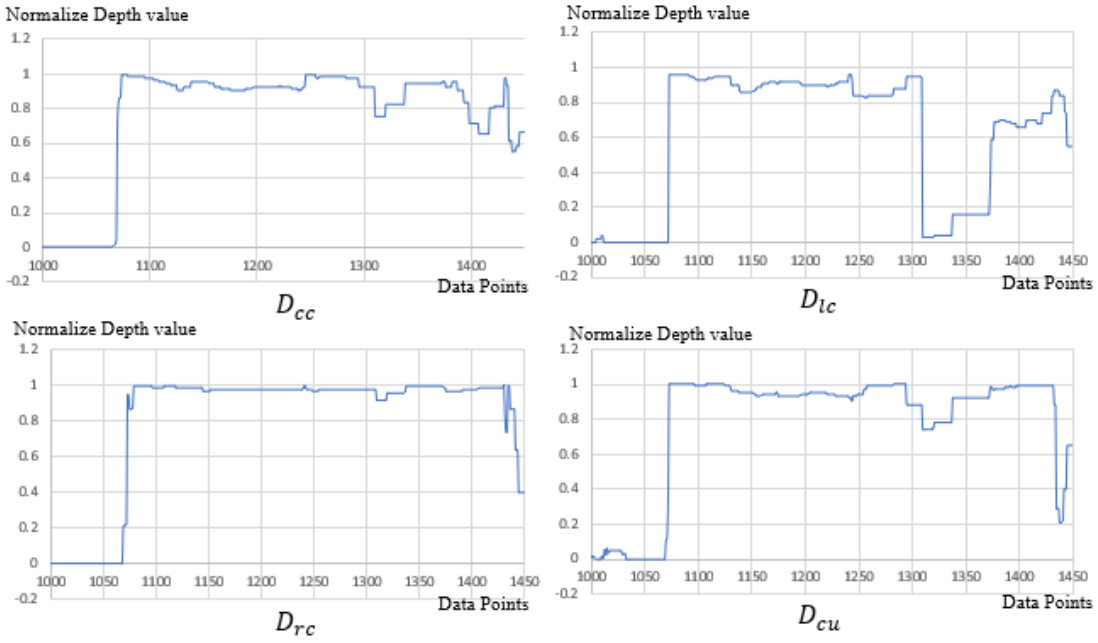

Figure 3.41 Variation of depth values of fuzzy decision-making system.

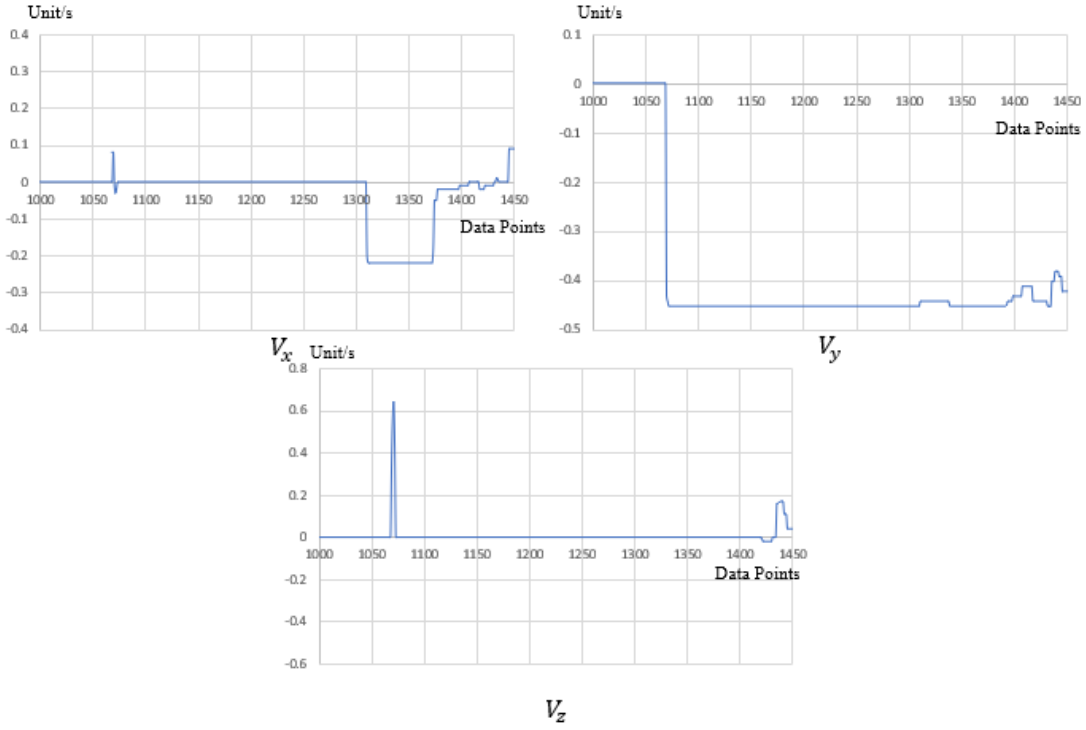

Figure 3.42 Variation of velocities of fuzzy decision-making system.

4 EXPERIMENTAL RESULTS AND ANALYSIS

4.1 Flight test and result analyzing of Extended Kalman filter

4.1.1 Hardware setup

Quad-copter was assembled as shown in Figure 4.1. Flight controller board, Motors and Electronic speed control units (ESC), Radio Telemetry module, Radio control receiver, Battery and GPS antenna were included in here as described in Chapter 3. All sensors, motor sequence and all other components and were checked before the flight test.

Figure 4.1 Final assembled quadcopter

4.1.2 Experimental procedure

Quad-copter was flying open field as shown in Figure 4.2. Quad-copter was Take-off manually then flew few meters in open field and finally land where it Take-off. While flying, data was collected every 2.5 milli-seconds through Wi-Fi communication module. GPS Horizontal Positions and Velocities of Latitude and Longitude directions, Barometer Height and Climb rate, EKF Positions and Velocities of Latitude Longitude and Up-Down directions were recorded. Quad-copter was not fly too far from the pilot because of loss of data due to low communication range of Wi-Fi modules.

Figure 4.2 Quadcopter flying on open field

4.1.3 Results

Barometer measured height with EKF estimated height and Barometer measured climb rate with EKF estimated climb rate were plotted as shown in Figure 4.3 and Figure 4.4. Initially, EKF measured height and climb rate were little bit drift due to accelerometer Z axis bias. But within very short period, it is converged into measured value as shown in figures. Normally barometers are very highly noise sensors because they measured changes in atmospheric pressure. But according to figures EKF estimated values have low noise compare to measured values. In here 2.0m measurement covariance was used for height measurements and 1.5m/s measurement covariance was used for climb rate measurement due to high noise of climb rate measurements. 0.25m/s^2 Accelerometer process noise was used in this situation.

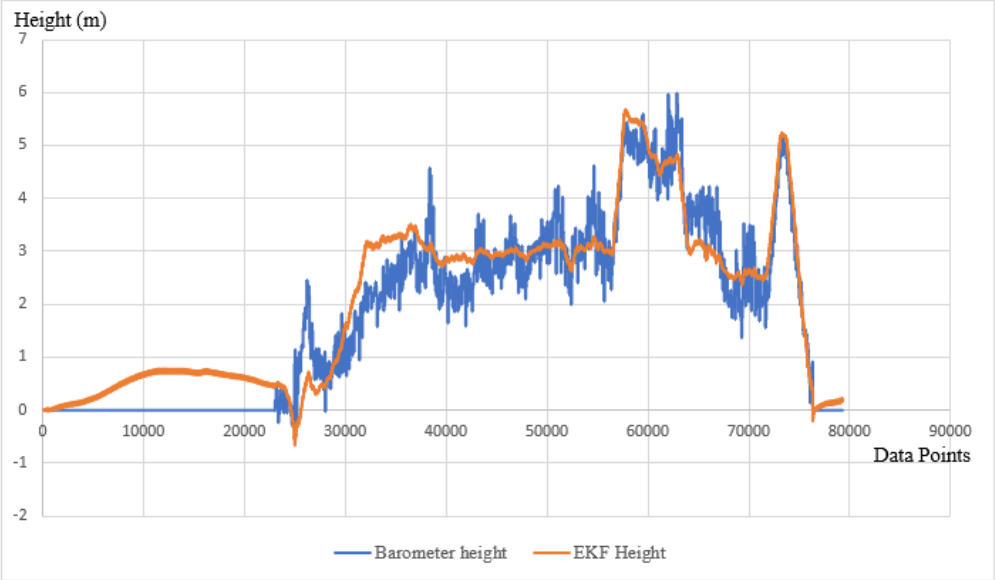

Figure 4.3 Barometer measured height vs EKF estimated height

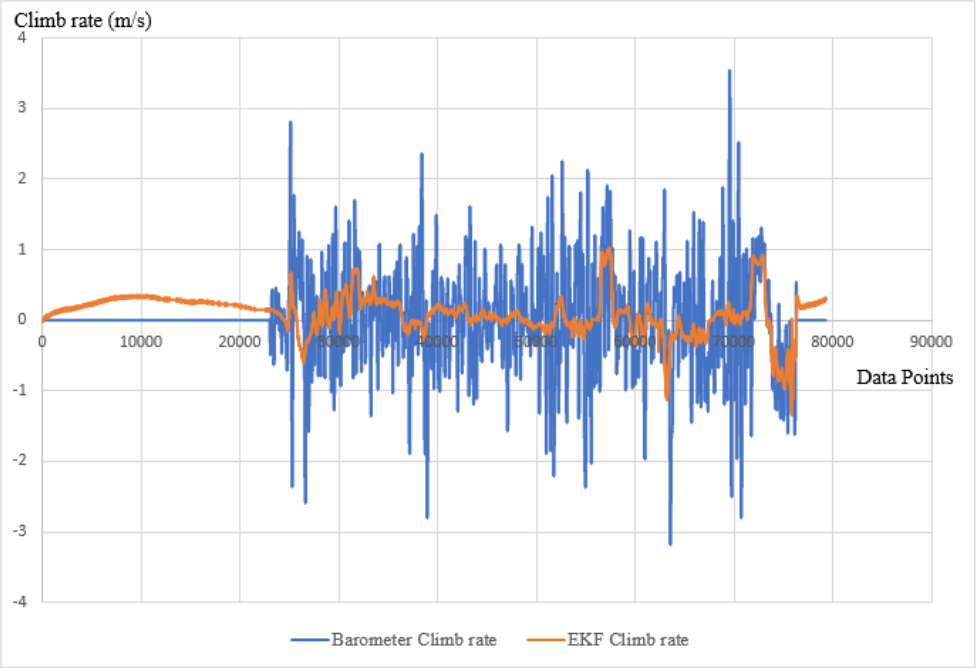

Figure 4.4 Barometer measured climb rate vs EKF estimated climb rate

In here GPS horizontal positions and velocities for Latitude and Longitude directions were plotted with EKF estimated horizontal position and velocities for Latitude and Longitude directions as shown in Figure 4.5, Figure 4.6, Figure 4.8 and Figure 4.9. Unlike above initial bias of EKF estimated horizontal position and velocity are almost zero due to negligible bias in X and Y axis of accelerometer.

According to Figure 4.5 and Figure 4.6, there is not much difference in GPS measured position and EKF estimated position. Because GPS measured position has normally low noise. But update frequency of GPS measured position is 5Hz and this is very low update frequency compare to EKF estimated positions which have 400Hz update frequency. This is very clearly indicated in Figure 4.7 (which is zoom version of Figure 4.6) as step changes in GPS measured position. Also, GPS measured position is normally lag 200 milli-second compares to inertial measurement unit. In here 220 milli-second GPS measurement lag was assumed. Figure 4.7 clearly shows EKF estimated position is 220 milli-second leads compare to GPS measured position. According to Figure 4.7 to reach 10m, EKF estimated position takes only 66500 data points, but to reach same position GPS measured position takes 66580 data points, Therefore, the difference between these two are 80 data points. Since data were collected every 2.5 milli-second total time between these two points are 220 milli-second. In here GPS measured position covariance consider as 0.5m and accelerometer process noise consider as $0.25 m/s^2$. According to Figure 4.8 and Figure 4.9 compare to EKF estimated velocity, GPS velocity is noisier. GPS velocity also suffers from low update frequency as mention above and measurements are lag compare to inertial measurement unit. These disadvantages are overcome by using Extended Kalman Filter.

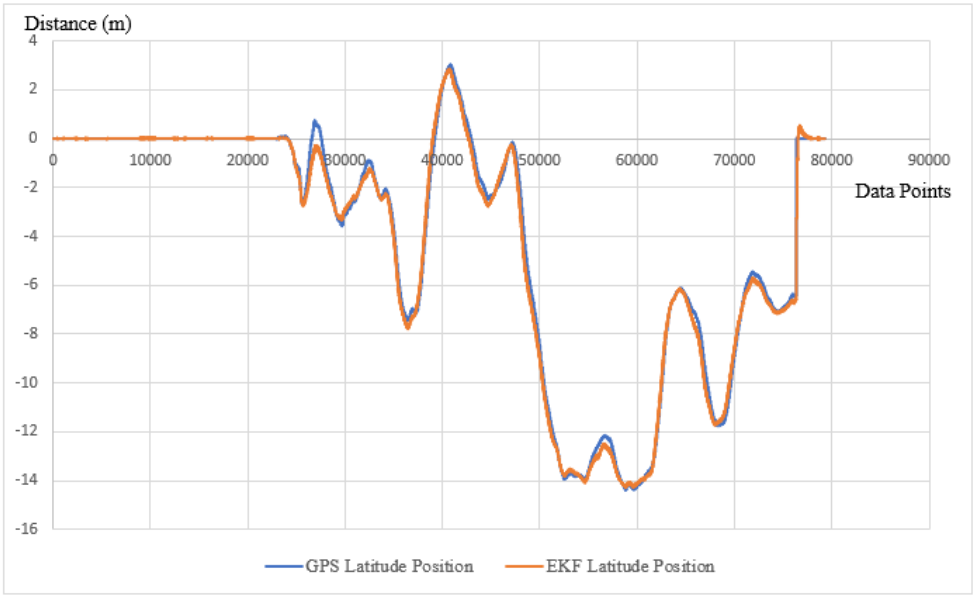

Figure 4.5 GPS measured Latitude position vs EKF estimated Latitude position

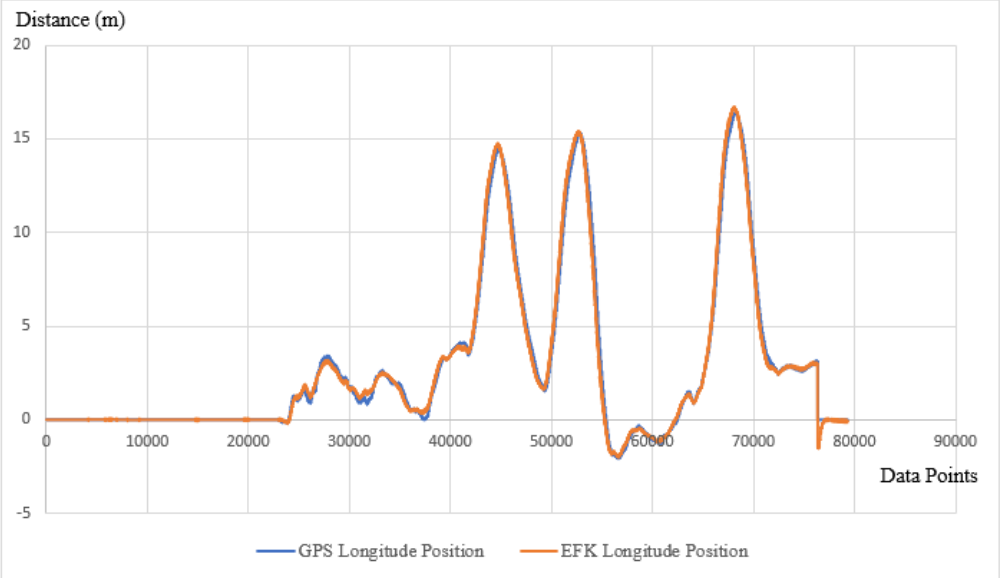

Figure 4.6 GPS measured Longitude position vs EKF estimated Longitude position

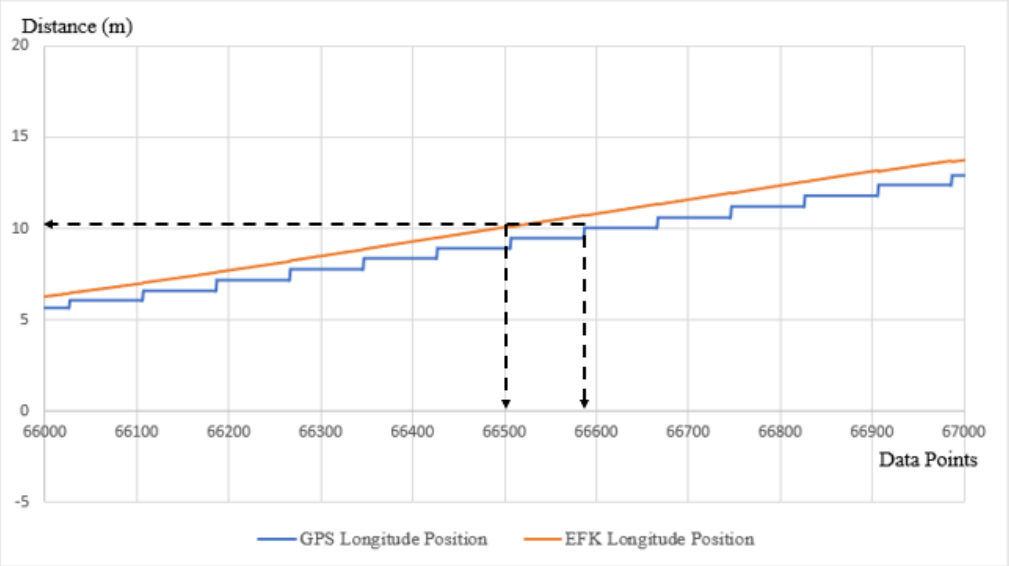

Figure 4.7 Zoom version of GPS measured Longitude position vs EKF estimated Longitude position between data point 6600 and 67000

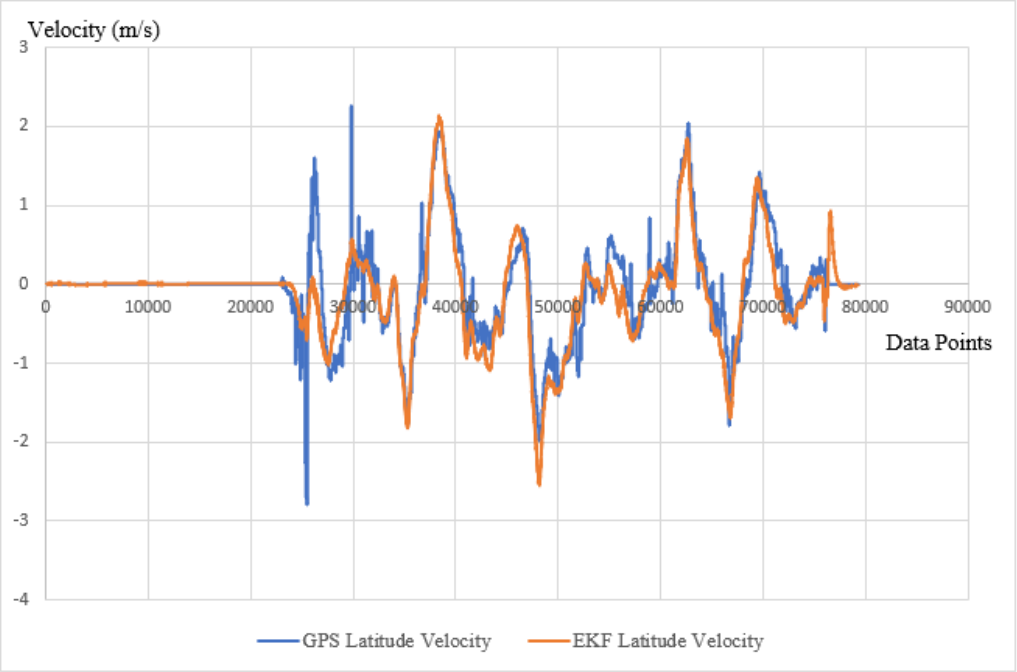

Figure 4.8 GPS measured Latitude velocity vs EKF estimated Latitude velocity

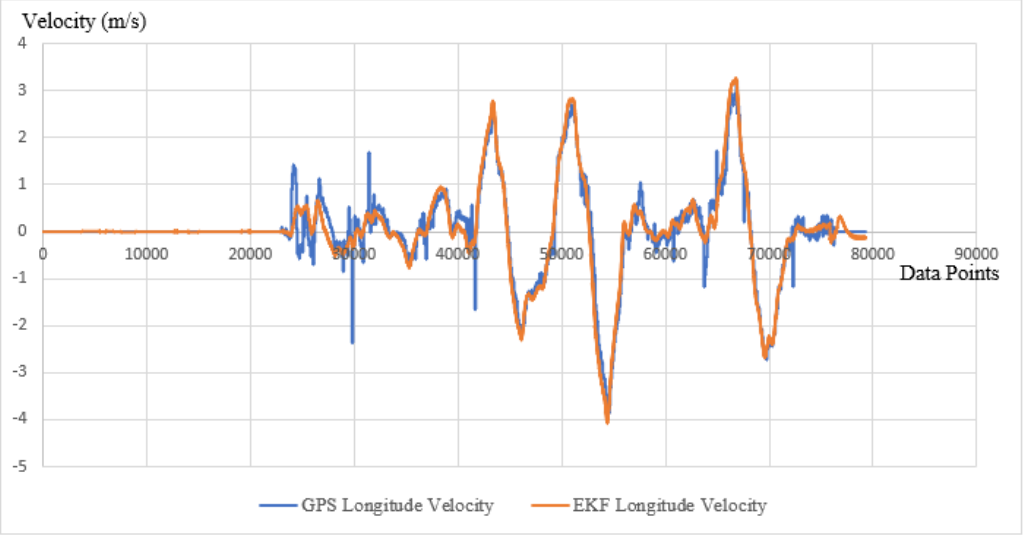

Figure 4.9 GPS measured Longitude velocity vs EKF estimated Longitude velocity

4.1.4 Position controller error variation with Extended Kalman filter

Behavior of position controller with Extended Kalman filter was shown in Figure 4.10, 4.11, 4.12, 4.13, 4.14, 4.15. Data were collected every 2.5ms as mentioned in previous chapter. In Figure 4.10 Actual height and desired height were plotted in same plot. According to these results, actual height variation of the system was almost tracked the desired height variation with small steady state error as shown in Figure 4.11. Initially, this error is somewhat large value, but when time passes this error is close to zero and almost all the time, height error of the system is less than 0.5m except initial convergence situation. Figure 4.12 and Figure 4.14 shows Actual and Desired position variations for Latitude and Longitude directions. In these two directions system was tracked desired position but with little overshoots as shown Figure 4.13 and Figure 4.15. These overshoots mostly occur when system is start to move, but almost all the time position error is less than 2m. Therefore, in all directions, position errors, steady state errors and overshoots are in acceptable values.

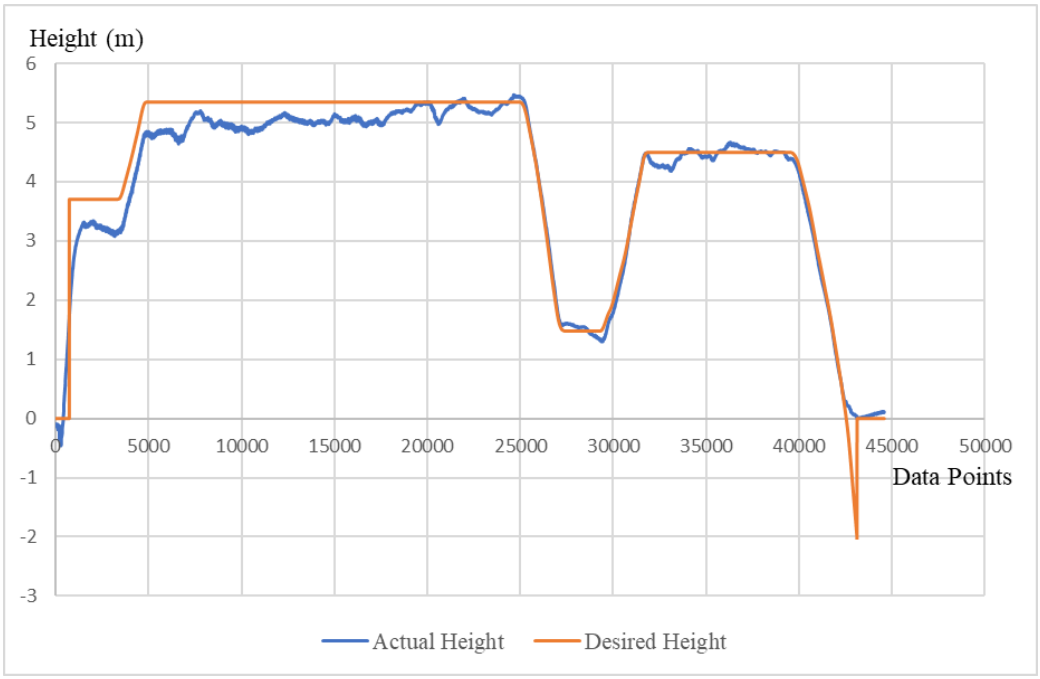

Figure 4.10 Actual height variation vs desired height variation

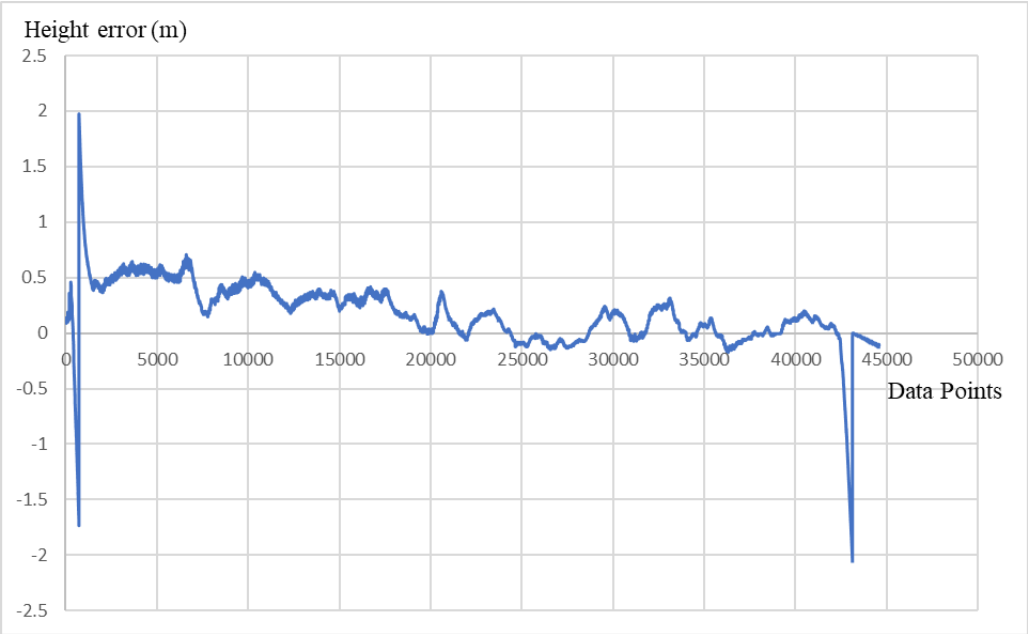

Figure 4.11 Height error variation

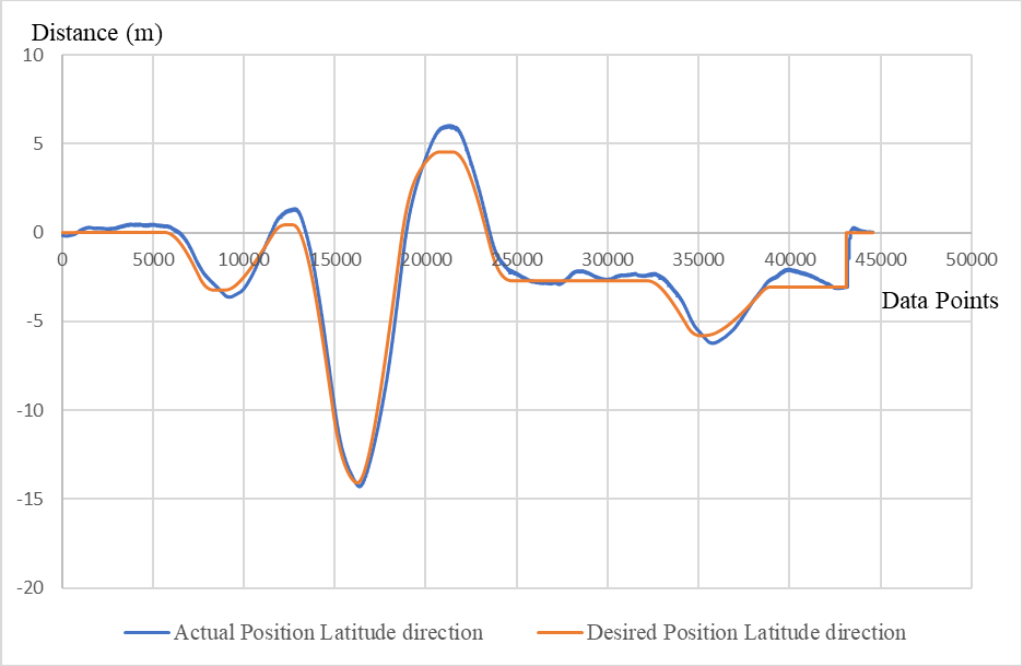

Figure 4.12 Actual position variation vs desired position variation in Latitude direction

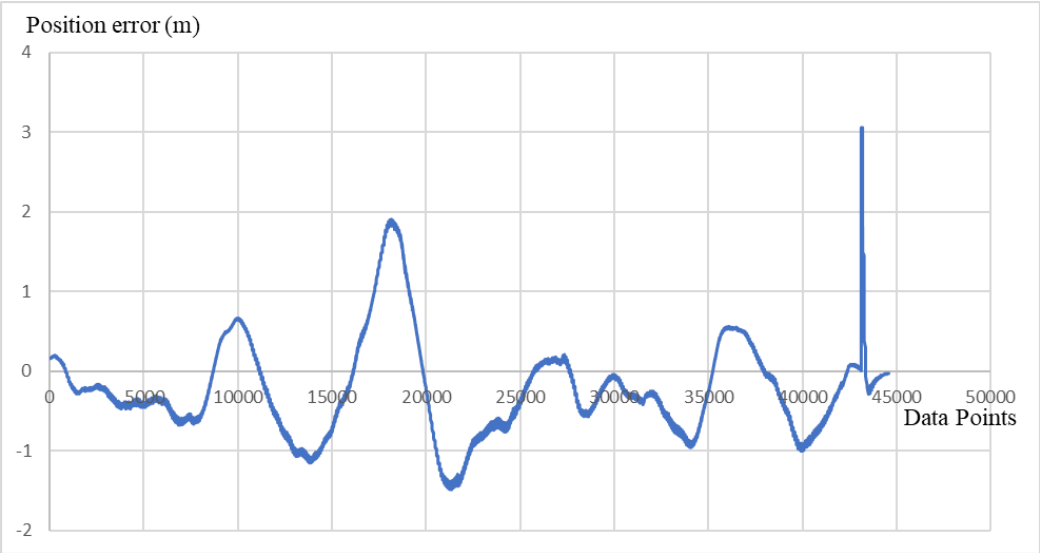

Figure 4.13 Position error variation in Latitude direction

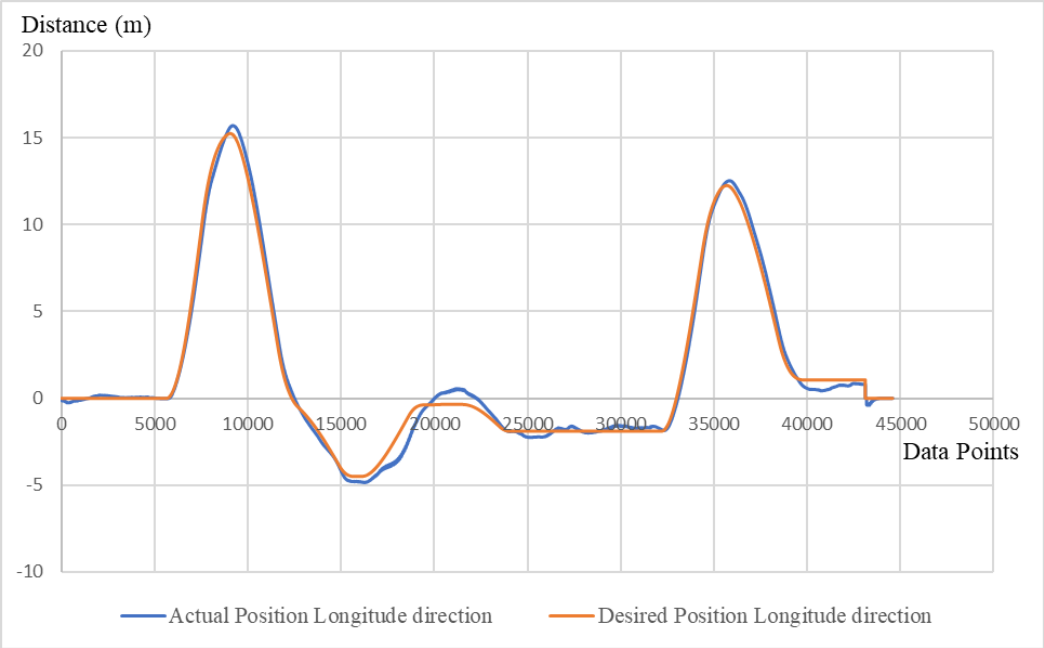

Figure 4.14 Actual position variation vs desired position variation in Longitude direction

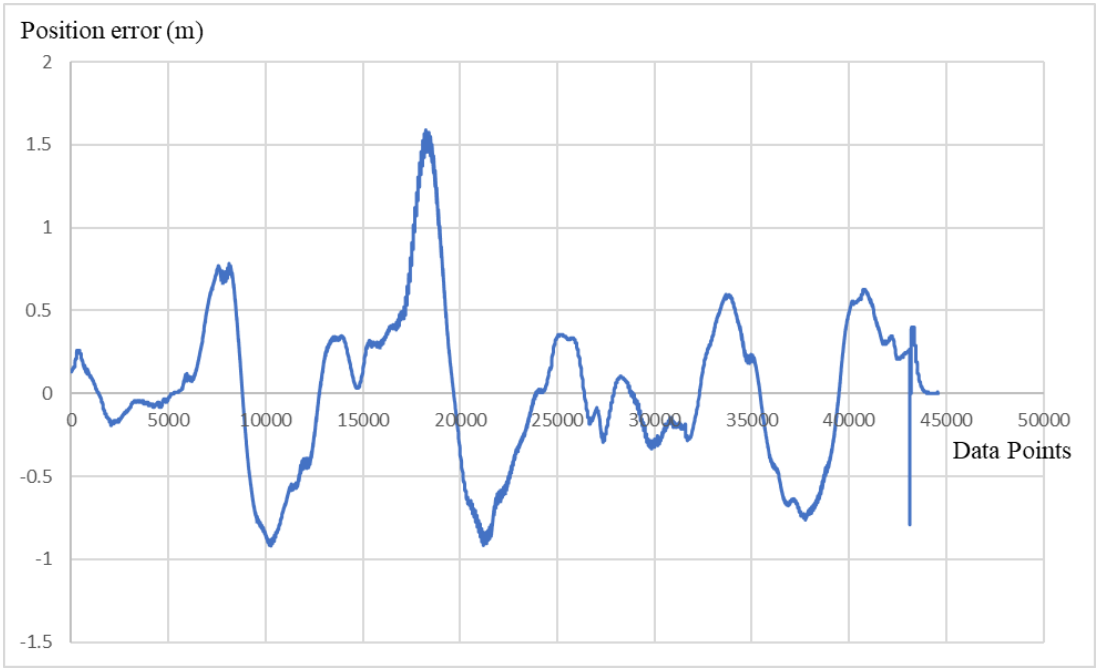

Figure 4.15 Position error variation in Longitude direction

4.1.5 Problems encountered during flight tests

Most of the time communication range of Wi-Fi modules cause problems due to low communication range. Due to same 2.4GHz Communication frequency in Wi-Fi module and Remote controlling unit caused communication confliction.

4.2 Flight tests and results analyzing of obstacle avoidance system

4.2.1 Hardware setup

Figure 4.16 shows overall hardware setup for test obstacle avoidance system. In here stereo camera is connected to Nvidia Jetson TX1 computer using USB 3.0 port. All image processing stuff and flight control algorithm runs on this Jetson TX1. Normally Nvidia Jetson TX1 doesn't have sensors like inertial measurement unit (Accelerometer + Gyroscope), Magnetometer, Barometer and GPS. Therefore, these sensors were externally connected to Jetson TX1 using available SPI, I2C and Serial ports. All other components are same as above mention Quad-copter. All sensors, motor sequence and all other components and were checked before the flight test.

Figure 4.16 Overall hardware setup including stereo camera.

4.2.2 Experimental procedure

Flight experimentation was done in two different situations. First one is vehicle was fly into a large tree which have feature rich scene. Second one is vehicle was fly into a wall which have less features. Both experiments were done using same vehicle and during flight tests environment had almost same lightning condition. Results of this tests were presented in next section.

4.2.3 Results

Test 1: Quad-copter fly to a tree

In here Quad-copter Take-off and autonomously fly into a large tree as shown in Figure 4.17. Figure 4.17(a) shows quad-copter fly into a large tree. Then Figure 4.17(b) shows changing its height to avoid collision with the tree and navigate above the tree. Figure 4.17(c) and (d) shows final stages of avoiding the tree safely. In this situation vehicle was fly in the feature rich scene. Therefore, avoiding this type of obstacles is possible because creation of depth images not much difficult in this type of situations.

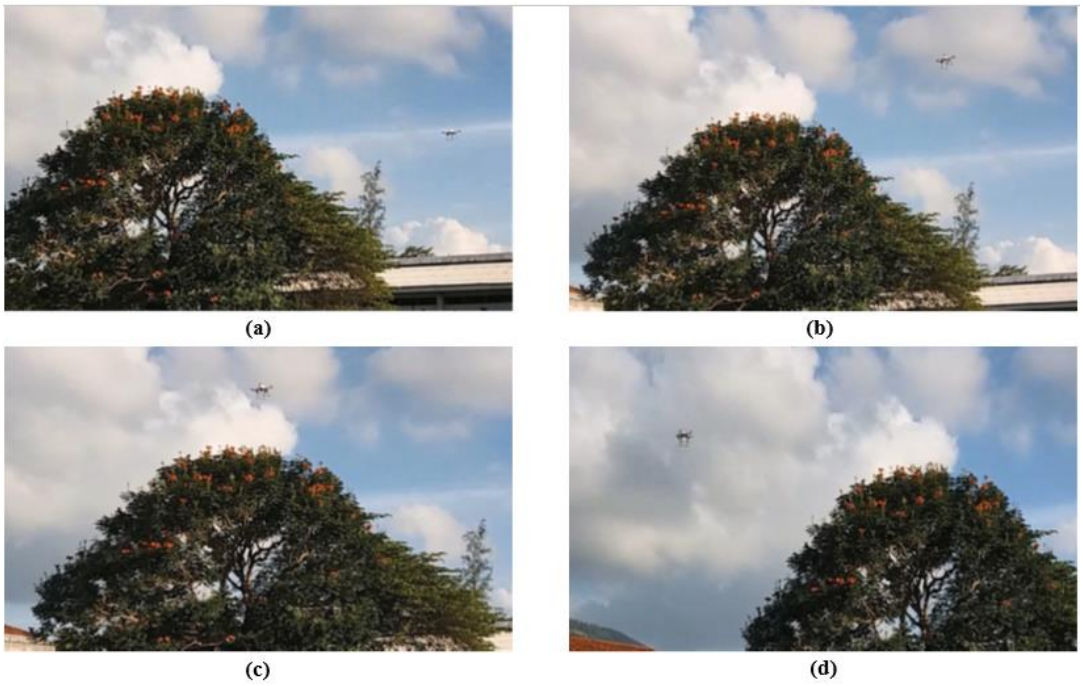

Figure 4.17 Quadcopter fly over tree

Test 2: Quad-copter fly to a solid wall

In here Quad-copter Take-off and fly into a wall as shown in Figure 4.18(a). However, quadcopter cannot see the wall and go into crash as shown in Figure 4.18(b). Because wall doesn't contain any identifiable features like corners or edges. This is called uniform texture problem in stereo vision as mention in chapter1. Therefore, stereo camera cannot be created depth maps and wall cannot be identified as an obstacle. If wall contain with feature rich surface, then it can be identified.

(a) (b)

Figure 4.18 Quadcopter fly to a wall.

4.2.4 Problems encountered during flight tests

Unlike Extended Kalman filter test, here data saving is not possible. Because flying distance is too long compared to previous experiments and therefore, communication loss occurred while data saving. Some situations, stereo camera facing directly to sun-light caused blind situations. These situations caused crashed vehicle into obstacles. Due to low lightning camera caused to blind and these situations obstacle avoidance is also not possible using stereo camera.

5 CONCLUSIONS

In here sixteen states Extended Kalman filter and stereo vision and fuzzy-logic based obstacle avoidance system were implemented. Both systems were tested in real-time and results were presented in here. Attitude, Position and Velocity of the quadcopter were estimated using Extended Kalman filter and experimental results were compared with raw sensor measurements of GPS horizontal position and velocity, Barometer height and climb rate. Performance and reliability of vehicle state estimation process was increased due to usage of Extended Kalman filter and experimental results also proved this. Horizontal position and velocity update frequency of the quad-copter was 5Hz without using Extended Kalman filter and its increased to 400Hz after implementing the filter and height and climb rate estimation frequency was also increased to 10Hz to 400Hz. GPS Measurements lag also caused to reduce the performance of the system and, by using Extended Kalman filter reduce this measurement lag problem and experimental results clearly showed this improvement. Barometer height and climb rate measurement are very noisy and Extended Kalman filter reduces this barometer noise problem as shown in results.

Stereo vision system was implemented using OpenCV and tested with different lightning conditions in indoor and outdoor environments. OpenCV functions were used to calibrate the stereo camera and image rectification process. OpenCV global optimization technique called Semi-Global block matching stereo correspondence algorithm was used to create depth maps because of its high accuracy compare to local optimization algorithms. Fuzzy logic was used to implement the obstacle avoidance decision-making system. Depth maps created by stereo vision system was inputs to the fuzzy decision-making system. Depth map is divided into equal nine portions and normalize depth values of each region were fed into fuzzy decision-making system as inputs. Output of the fuzzy system is desired navigation direction of the vehicle. Outputs of the fuzzy decision-making system were fed into flight control algorithm to avoid the obstacles and navigate around it. Obstacle avoidance system failed to avoid obstacles in some situations due to blind scenarios occurred in stereo vision algorithm and in future expected to resolve these problems.

6 RECOMMENDATIONS AND FURTHER WORKS

Extended Kalman filter experimental results were compared with GPS and barometer raw measurements instead of using ground truth measurement. Because it is hard to find ground truth like these kinds of situations. However, here I can recommend some alternatives to find a ground truth to compare experimental results of the EKF. First, I recommended to use commercially available GPS/INS module like Xsense MTi-10. Xsense MTi-10 is an industry standard MEMS based GPS/INS navigation module which is use for many industrial applications like drone stabilizations, control and stabilizations of industrial equipment, Navigation and control of cargo ships etc. According to data sheet of this module it can provide attitude, velocity and position information up to 2KHz and measurement latency is less than 2ms. It has GUI for visualize the outputs such as velocity, position. It also includes examples for C/C++ and MATLAB and Windows and Linux are supported operating systems. Second option is we can use simulated vehicle model with EKF to compare filter values. However, I hesitate to recommend this as an option because it is hard to mathematically model all those dynamics of the system and this will generate faulty results. But if you have very accurate system model like including wind dynamics, propeller dynamics then I will recommend to use this option. As a third option I recommended to use vision position estimation using AR tag. AR tag is image pattern which can be used to estimate position using cameras. However, this method is more suitable for indoor environments because lightning variations are cause problems in position estimation. This method also suffers from measurement lag (like GPS) because there is a time delay on image sensing process and calculation of the outputs. However, here I recommended to use high processing power laptop or Nvidia Jetson TX1 for image processing because EKF estimation and AR tag position estimation should be synchronize.

When it comes to obstacle avoidance system, vehicle will stop if all nine sub-regions are occupied by obstacles during navigation. However, I recommended if this kind of situation is occurred then vehicle continually go upwards until obstacle is disappear from the scene and then continue the navigation. This can be done by giving constant positive earth frame desired velocity for vertical axis until obstacle is disappear from

the scene. By integrating this earth frame desired velocity, we can get earth frame desired position for vertical axis and then this position can be feed to position controller to control the altitude. In this situation desired horizontal positions should not be changed until obstacle is disappear from the depth image. This scenario is most useful for when vehicle will be face to building like structures.

REFERENCES

1. J. A. A. S. Somasiri, K. G. B. Gamagedara, D. H. S. Maithripala and J. M. Berg, "Implementation of an almost globally stable intrinsic nonlinear PID controller for attitude stabilization of a quadrotor," 2015 IEEE 10th International Conference on Industrial and Information Systems (ICIIS), Peradeniya, 2015, pp. 425-428.
2. Maithripala, D. H. S., and Berg, J. M., 2014. "An Intrinsic Robust PID Controller on Lie Groups". Automatica, pp. 5606–5611 (2015).
3. Silvere Bonnabel, Philippe Martin, Erwan Salaun. Invariant Extended Kalman Filter: theory and application to a velocity-aided attitude estimation problem. 48th IEEE Conference on Decision and Control, Dec 2009, Shanghai, China. pp.1297-1304, 2009,
4. D. H. S. Maithripala, W. P. Dayawansa, and J. M. BERG, "Intrinsic observer-based stabilization for simple mechanical systems on Lie groups," SIAM J. Control and Optim., vol. 44, pp. 1691–1711, 2005.
5. R. Mahony, Tarek Hamel, Jean-Michel Pflimlin. Nonlinear Complementary Filters on the Special Orthogonal Group. IEEE Transactions on Automatic Control, Institute of Electrical and Electronics Engineers, 2008, 53 (5), pp.1203-1217.
6. S. O. H. Madgwick, A. J. L. Harrison and R. Vaidyanathan, "Estimation of IMU and MARG orientation using a gradient descent algorithm," 2011 IEEE International Conference on Rehabilitation Robotics, Zurich, 2011, pp. 1-7.
7. Sabatini, Angelo. (2011). Kalman-Filter-Based Orientation Determination Using Inertial/Magnetic Sensors: Observability Analysis and Performance Evaluation. Sensors (Basel, Switzerland). 11. 9182-206. 10.3390/s111009182.
8. Madinehi, Nojan, "Rigid Body Attitude Estimation: An Overview and Comparative Study" (2013). Electronic Thesis and Dissertation Repository. Paper 1259.
9. Lazaros Nalpantidis., Antonios Gasteratos.,"Stereo Vision-based Fuzzy Obstacle Avoidance Method".International Journal of Humonoid Robotics Vol. 8, No. 1 (2011)169-183.
10. K. Muhlmann, D. Maier, J. Hesser and R. Manner, Calculating dense disparity mapsfrom color stereo images, an efficient implementation, International Journal of Computer Vision **47**(1–3) (2002) 79–88.

11. L. Di Stefano, M. Marchionni and S. Mattoccia, A fast area-based stereo matchingalgorithm, Image and Vision Computing **22**(12) (2004) 983–1005.
12. H. Hirschmuller, "Stereo Processing by Semiglobal Matching and Mutual Information," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 30, no. 2, pp. 328-341, Feb. 2008.
13. D. Hernandez-Juarez, A. Chac´on, A. Espinosa, D. V´azquez, J. C. Moure, and A. M. L´opez, “Embedded real-time stereo estimation via Semi-Global Matching on the GPU” The International Conference on Computational Science, Volume 80, 2016, Pages 143–153.
14. D. Burschkal, S. Lee and G. Hager, "Stereo-based obstacle avoidance in indoor environments with active sensor re-calibration," Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292), 2002, pp. 2066-2072 vol.2.
15. J. Borenstein and Y. Koren, The vector field histogram-fast obstacle avoidance formobile robot, IEEE Transactions on Robotics and Automation **7**(3) (1991) 278–288.
16. C. Kyung Hyun, N. Minh Ngoc and R. M. Asif Ali, A real time collision avoidancealgorithm for mobile robot based on elastic force, International Journal of Mechanical, Industrial and Aerospace Engineering **2**(4) (2008) 230–233.

APPENDICES

Appendix - A	MATLAB Symbolic Implementation of Extended Kalman Filter.
Appendix - B	C++ Implementation of Extended Kalman Filter.
Appendix - C	C++ Implementation of Stereo Vision System.
Appendix - D	C++ Implementation of Fuzzy Decision-making System.
Appendix - E	Extended Kalman filter ground test data.
Appendix - F	Extended Kalman filter flight test data.
Appendix - G	Obstacle avoidance system ground test data.
Appendix - H	Video evidence of system test.

Note: Appendices are available on the provided compact disk (CD).