AUTONOMOUS NAVIGATION AND LOCALIZATION OF A QUADROTOR MINI-UAV BY LANDMARKS IN INDOOR ENVIRONMENTS

BY

YOUNG MIN AHN

THESIS

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Mechanical Engineering in the Graduate College of the University of Illinois at Urbana-Champaign, 2011

Urbana, Illinois

Adviser:

Professor Ramavarapu Sreenivas
ABSTRACT

In this thesis, the problems of control, autonomous navigation, and localization in an indoor environment are considered for mini-UAV quadrotor. A commonly used localization methods such as GPS sensors or radar are strongly limited due to lack of signals in indoor. This thesis will present autonomous navigation by using vanishing point and inertial navigation methods. And indoor localization by using artificial landmarks in a known environment by utilizing vertical and horizontal cameras onboard of the quadrotor is proposed. The local data from the inertial measurement units onboard are combined with the proposed localization method.
To Mother and Father
ACKNOWLEDGMENTS

This thesis would not have been possible without the support of many people. I would like to acknowledge and thank my adviser, Professor Ramavarapu Sreenivas, for his guidance and encouragements. Many thanks to Daniel Block, head of the Controls Systems Laboratory for providing his expert advice, support, and lab space. Thank you to Hyongju Park and Seungho Lee for their personal advice, and friendship. Finally, I would like to thank my parents for their consistent encouragement, love, and support.
TABLE OF CONTENTS

LIST OF TABLES ........................................................................................................... vii

LIST OF FIGURES ......................................................................................................... viii

CHAPTER 1 INTRODUCTION .......................................................................................... 1

CHAPTER 2 SYSTEM HARDWARE AND SOFTWARE .................................................. 4
  2.1 AR.Drone Quadrotor .......................................................................................... 4
  2.2 Sensors ............................................................................................................... 5
  2.3 Software Development Environment .................................................................. 7
  2.4 Command and Data Communication ................................................................. 7

CHAPTER 3 MODELING ............................................................................................... 10
  3.1 Coordinate Frames ........................................................................................... 10
  3.2 Quadrotor Reference Frames ............................................................................ 12
  3.3 Quadrotor Simplified Dynamics ...................................................................... 15
  3.4 Inner-loop Dynamics ....................................................................................... 19
  3.5 Altitude Velocity Estimation and Lowpass Filter Design .............................. 21
  3.6 System Identification ....................................................................................... 24
  3.7 Simulation Model and Validation ..................................................................... 27

CHAPTER 4 CONTROLLER DESIGN ........................................................................... 32
  4.1 PID Controller ................................................................................................... 32
  4.2 Root Locus Design Method ............................................................................. 35
  4.3 Position Controller .......................................................................................... 41
  4.4 Velocity Controller .......................................................................................... 44
  4.5 Dead Reckoning ............................................................................................... 46
  4.6 Flight Path Tracking Control ............................................................................ 47
LIST OF TABLES

Table 4-1: Flight controller PID gains ................................................................. 36
Table 5-1: Results of camera calibration intrinsic parameters............................... 60
LIST OF FIGURES

Figure 1.1: Mobile robots, Samsung NaviBot (left) and iRobot Roomba (right) .................. 2

Figure 2.1: Parrot AR.Drone quadrotor, front and top view .................................................. 5
Figure 2.2: Plot of filtered and unfiltered angle orientations ...................................................... 6
Figure 2.3: Ultrasound sensor and camera location on AR.Drone ............................................. 6
Figure 2.4: User interface between the quadrotor and ground station ........................................ 8
Figure 2.5: Measured rate of inertial data packet received by ground station ............................. 8
Figure 2.6: Measured rate of video stream data packet received by ground station ................. 9
Figure 2.7: Block diagram representation of overall system architecture .................................... 9

Figure 3.1: Euler angles roll, pitch, and yaw representation ...................................................... 12
Figure 3.2: Representation of local frame .................................................................................... 13
Figure 3.3: The coordinate system of quadrotor for modeling .................................................. 14
Figure 3.4: Moment arm distance about the center of mass ....................................................... 16
Figure 3.5: Plot of noisy calculated altitude velocity ................................................................. 21
Figure 3.6: Plot of filtered estimation of altitude velocity .......................................................... 23
Figure 3.7: Input-output data set of yaw motion for system identification ................................. 24
Figure 3.8: Step input pitch% to the inner-loop controller and x-velocity response ............. 26
Figure 3.9: Within linear region of step input pitch% and x-velocity response ..................... 27
Figure 3.10: Yaw dynamics simulation model in Simulink with the time delay element ............ 28
Figure 3.11: Altitude dynamics simulation model in Simulink ..................................................... 28
Figure 3.12: Linear velocity of x and y dynamics model in Simulink .................................. 28
Figure 3.13: Comparison of simulated and measured yaw, yaw rate ...................................... 29
Figure 3.14: Comparison of simulated and measured altitude, altitude velocity ..................... 29
Figure 3.15: Comparison of simulated and measured linear x-velocity ................................. 30
Figure 3.16: Comparison of simulated and measured linear y-velocity ................................. 30
Figure 3.17: Comparison of MIMO measured and simulated position response .......... 31
Figure 3.18: Comparison of MIMO of measured and simulated velocity response .... 31

Figure 4.1: Block diagram of classical and modified PID controller ...................... 36
Figure 4.2: Root locus plot of yaw sub-system .................................................. 36
Figure 4.3: Root locus plot of altitude sub-system ......................................... 37
Figure 4.4: Root locus plot of velocity-x sub-system ....................................... 37
Figure 4.5: Root locus plot of velocity-y sub-system ....................................... 38
Figure 4.6: Complete decoupled sub-model of quadrotor in Simulink. ................. 38
Figure 4.7: The yaw PID controller performance simulated ................................ 39
Figure 4.8: The altitude PID controller performance simulated ........................ 39
Figure 4.9: The velocity-x PID controller performance simulated ..................... 40
Figure 4.10: The velocity-y PID controller performance simulated ................... 40
Figure 4.11: Yaw orientation PID controller structure in Simulink. ..................... 41
Figure 4.12: Altitude position PID controller structure in Simulink .................... 42
Figure 4.13: Quadrotor response to step reference yaw input ............................ 42
Figure 4.14: Quadrotor response to a step reference altitude input ................. 43
Figure 4.15: Simulink structure of anti-integral windup resetting-factor .............. 43
Figure 4.16: Linear velocity x and y PI controller structure in Simulink. ............. 44
Figure 4.17: Quadrotor response to a step reference velocity-x input ............... 45
Figure 4.18: Quadrotor response to a step reference velocity-y input ............... 45
Figure 4.19: A schematic of flight path controller ........................................... 47
Figure 4.20: Plot of 2-D flight trajectory path measured by IMU ..................... 48
Figure 4.21: Plot of 3-D flight trajectory ............................................................. 49
Figure 4.22: Plot of quadrotor linear velocities during 3-D flight trajectory ......... 49

Figure 5.1: Artificial landmarks. The circular shapes have radius of 6 centimeters. .... 51
Figure 5.2: Geometric representation of RGB and HSV .................................... 52
Figure 5.3: Plot of 2-D symmetric Gaussian kernel ........................................ 53
Figure 5.4: Pictures of detected landmarks taken by vertical camera ................. 53
Figure 5.5: Detected landmark picture taken by vertical camera while hovering .... 54
Figure 5.6: Plot of roll, pitch, corrected and uncorrected yaw attitude during hovering. 55
Figure 5.7: Picture of calibration grid.......................................................................................... 56
Figure 5.8: Schematic of the vertical camera system model. ...................................................... 57
Figure 5.9: Different orientations of paper box used to calibrate the vertical camera..... 59
Figure 5.10: Example plot of vanishing point calculation performed in Matlab................ 60
Figure 5.11: Test setup for camera intrinsic model validation .................................................... 61
Figure 5.12: Images of corridors in Mechanical Engineering Laboratory at UIUC.............. 62
Figure 5.13: MEL corridor image edge detection and Hough space representation. ...... 63
Figure 5.14: Images of edge detection and vanishing point detection............................. 65
Figure 5.15: Result of applying rotational matrix to image plane ........................................ 65
Figure 5.16: Schematic of optical center to $v_u$ .................................................................... 66
Figure 5.17: Landmarks in the MEL corridor............................................................ 67
Figure 5.18: Result showing successful localization at MEL............................................. 68
Figure 5.19: GUI layout displays the quadrotor in action. .................................................... 68
Figure 5.20: Four different landmarks used for corner turn navigation and localization. 70
Figure 5.21: Result showing successful autonomous navigation and localization........... 70

Figure A: Contents included in the supplemental files......................................................... 73
CHAPTER 1

INTRODUCTION

“For the execution of the voyage to the Indies, I did not make use of intelligence, mathematics or maps.”

- Christopher Columbus

Imagine yourself flying in an airplane. You have a good idea of where you are prior to take off but after few minutes, you become clueless. All you can see out the window are clouds. Few hours have passed, you look out the window then see the Sears Tower and you realize you are in Chicago.

When we navigate around the world we depend on known landmarks or maps to estimate where we are. We look at street signs, buildings, or addresses to update our location and make a new decision to get to our destination. Hence we can lose sense of our directions without a priori information about the environment.

Mobile robots that move freely around the world have the same problems as us, when navigating around the world. The mobile robots can be divided into two principle groups, which are global-navigational and local-navigational mobile robots. The global-navigational mobile robots can determine its position in absolute or map-referenced terms whereas the local-navigational mobile robots determine its position relative to stationary
or moving objects in the environment. The Samsung’s NaviBot (Figure 1.1) is an example of global-navigational mobile robot. It is a cleaning robot that can map its environment by using visual information. The iRobot’s Roomba (Figure 1.1) is an example of local-navigational mobile robot. It is a cleaning robot that interacts with its environment by sensing the bump against some object.

The advantage of having knowledge of its environment gives the global-navigational mobile robots to have potential to be used in security and surveillance missions. For example, unmanned aerial vehicle (UAV) such as MQ-1 Predator has been used in reconnaissance missions. Although the external sensors such as vision and radar are widely used in autonomous UAVs for localization, the global position system (GPS) and inertial navigation system (INS) integration method is most commonly used [1]. However, these methods are only effective in outdoor environment and they present many limitations in indoor. The GPS sensors or radar are strongly limited due to lack of signals in indoor and moreover, the positioning error of GPS receivers is considerably large compared to the small measured fields. In addition, the size of the UAV and the fixed wing flight is not suitable for indoor.

Figure 1.1: Mobile robots, Samsung NaviBot (left) and iRobot Roomba (right).
In this thesis, mini unmanned aerial vehicle (mini-UAV), specifically the quadrotor will be used as a platform to explore the challenges of modeling, control, autonomous flight, and localization in GPS-denied environment. The quadrotor makes excellent aerial vehicle for indoor flight since it has vertical takeoff and landing (VTOL) capability with excellent maneuverability and its small size. The quadrotor will use combinations of vision and inertial measurement units (IMU) for localization.

This thesis proceeds as follows. In Chapter 2, hardware and software used during the durations of this thesis is introduced. In Chapter 3, modeling of the quadrotor by grey-box approach is discussed. Then the modeled system is simulated and validated. Once the quadrotor model is establish, control design by root-locus method and performance results are presented in Chapter 4. In Chapter 5, computer vision techniques are utilized to enable the quadrotor to autonomously navigate through indoor environment and localize by observing the known landmarks. Finally, in Chapter 6, the results are summarized and future directions for this research are discussed.
CHAPTER 2

SYSTEM HARDWARE AND SOFTWARE

“A scientist discovers that which exists. An engineer creates that which never was.”

-Theodore von Karman

The following chapter will give an overview of hardware and software descriptions of the test platform used for this thesis.

2.1 AR.Drone Quadrotor

The AR.Drone quadrotor (Figure 2.1) is developed by a company named Parrot headquartered in Paris. The AR.Drone quadrotor is stabilized by an onboard controller that is equipped with 6 degrees of freedom (DOF) IMU, and two CMOS based cameras. The low-level control is regulated by proportional-integral-derivative (PID) type controller proprietary to Parrot. The controller takes four inputs from the user via WiFi signals which commands tilt angles, yawing rate, and altitude rate.

The AR.Drone quadrotor is highly stable system, and qualifies as an excellent testing platform.
2.2 Sensors

The inertial measurements of the quadrotor are obtained by the micro-electro-mechanical system (MEMS) devices. There are 3-axes accelerometer, 2-axes gyrometer, and 1-axis precision gyrometer. The 2-axes gyrometer measures roll rate and pitch rate, and the 1-axis gyrometer measures yaw rate on the quadrotor. Then the angular rates can be integrated over time to infer angular positions of the system. However, like all MEMS gyrometer, the bias offset will drift over time and the integrated values will fail to give an accurate angular position of the system. There are several filtering methods which utilize other system’s state to compensate for the bias offset error [2],[3]. The 3-axes accelerometer senses the tilt of the quadrotor which gives precise and absolute measurement of the roll and pitch attitude. Parrot has implemented a proprietary filtering method to compensate for the angular position error of roll and pitch by combining accelerometer and gyro measurements (Figure 2.2). However, drifting in angular position of yaw (Figure 2.2) cannot be compensated by using the same method. The acceleration vector of gravity is parallel to the yaw axis of the reference coordinate system hence absolute measurement of yaw attitude is not available. This issue will be addressed in chapter 5.

Figure 2.1: Parrot AR.Drone quadrotor, front and top view.
The altitude of the quadrotor is measured by the ultrasound sensor attached to the belly of the quadrotor. The emission frequency is 40 kHz with maximum reading range of 6 meters. Lastly, there are two CMOS cameras onboard. The horizontal camera has 93° wide-angle diagonal lens and vertical camera facing the ground has 64° diagonal lens.

Figure 2.2: Filtered and unfiltered angle orientations. The gyro bias offset compensated roll (top left) and pitch (top right) tracks true angle orientation over 45 minutes. As the bottom plot shows, uncompensated angle orientation of yaw (bottom) drifts away from its true angle orientation as much as 180 degree in 45 minutes. The measurements were conducted at room temperature.

The altitude of the quadrotor is measured by the ultrasound sensor attached to the belly of the quadrotor. The emission frequency is 40 kHz with maximum reading range of 6 meters. Lastly, there are two CMOS cameras onboard. The horizontal camera has 93° wide-angle diagonal lens and vertical camera facing the ground has 64° diagonal lens.

Figure 2.3: Ultrasound sensor and vertical camera location (left) and horizontal camera location on AR.Drone.
The locations of these sensors are shown in Figure 2.3.

According to Parrot’s website [4], vertical camera is used to calculate linear horizontal velocities of the quadrotor combined with IMU sensor readings. The algorithm uses an optical-flow[5] like method proprietary to Parrot which gives essentially a drift-less velocity estimations.

2.3 Software Development Environment

Parrot provides software development kit (SDK) for AR.Drone on many platforms including Linux operating system. However, there is no direct access to rewriting the embedded software. For this thesis, SDK version 1.6 and firmware 1.5.1 is used [4]. And for image processing, OpenCV v2.2 [6] is used. The project code is developed in Ubuntu10.04 LTS environment written in C/C++. The graphical user interface (GUI) is developed with GIMP tool kit (GTK) (Figure 2.4).

2.4 Command and Data Communication

AR.Drone is controlled from the ground station PC via WiFi ad-hoc connection while it is stabilized internally by the onboard controller. There are three main communication services. The control command is sent on datagram protocol (UDP) port 5556, which is sent over to the AR.Drone on a regular basis of approximately 30 times per second. The
inertial data of the AR.Drone is received by ground station pc on UDP port 5554 at approximately 200 times per second (Figure 2.5).

The video stream data is received by the ground station pc on UDP port 5555 at approximately 15 times per second (Figure 2.6). The image resolutions are 320 x 240 for horizontal camera and 176 x 144 for vertical camera where units are expressed in pixel.

![Figure 2.4: User interface between the quadrotor and ground station.](image)

![Figure 2.5: Measured rate of inertial data packet received by ground station PC. On average the inertial data packet is received every 5ms.](image)
The block diagram of complete system and an illustration of communications between the ground station and quadrotor is shown in Figure 2.7. The ground station PC is equipped with dual 2.80 GHz processor and 3.00 GB of RAM.

Figure 2.7: Block diagram representation of overall system architecture and wireless communication between the quadrotor and ground station. The wireless network is established in ad-hoc mode.
CHAPTER 3

MODELING

“All models are wrong, but some are useful.”

-George E. P. Box

It is important to have an accurate model of the system to design control algorithms. In this chapter, dynamic model of the quadrotor is derived. The exact description of inner-loop PID controller is not available, however model structures can be predicted. Hence the grey-box approach is used to estimate the values of the unknown parameters of the quadrotor using system identification technique. The system identification approach uses statistical methods to describe dynamic model of the system from measured input and output data.

3.1 Coordinate Frames

The coordinate frame is a collection of three orthogonal unit vectors \((\hat{i}, \hat{j}, \hat{k})\) that satisfy the right-hand-rule. One coordinate frame can be transformed into another through rotation and translation operations. The subscript will be used to distinguish between frames such as \((x_1, y_1, z_1)\) and \((x_2, y_2, z_2)\), also designated by the symbol \(f_1\) and \(f_2\). The
superscript will denote the coordinates of a vector $\mathbf{p} \in \mathbb{R}^3$ in frame 1 such as $\mathbf{p}^1$ or in frame 2 as $\mathbf{p}^2$. The $3 \times 3$ matrices $R \in SO(3)$, where $SO(3)$ denotes the special orthogonal group of order 3, can be used to represent the orientation of one coordinate frame with respect to another frame. Also, it can transform the coordinates of a point from one frame to another. Such matrices are called the rotation matrix. For example, coordinates in frame 1 respect to frame 2 can be represented by $\mathbf{p}^2 = R^2 \mathbf{p}^1$. And it is worth noting that all possible rotation matrices are the subset of $R \in \mathbb{R}^{3 \times 3}$ given by

$$SO(3) = \left\{ R \in \mathbb{R}^{3 \times 3} : R R^T = R^T R = I, \det R = 1 \right\}.$$ 

The Euler angles are three independent quantities that parameterize orientation of one frame to another. The roll, pitch, and yaw Euler angles shall be denoted by $\phi, \theta, \psi$ respectively (Figure 3.1). Then the rotations about each principal coordinate axes can be represented by Euler angles and rotation matrix as the followings

$$R_{x,\phi} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi) & \cos(\phi) \end{bmatrix} \quad (3.1)$$

$$R_{y,\theta} = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \quad (3.2)$$

$$R_{z,\psi} = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (3.3)$$
3.2 Quadrotor Reference Frames

To describe quadrotor dynamics, it is convenient to have several reference frames. Throughout the derivations, earth is assumed to be flat and not rotating which is a valid assumption for a mini-UAV.

- **The inertial frame, $f_i$**

  The location of the origin is fixed with respect to inertial space which is an earth fixed coordinate system. The initial takeoff point of the quadrotor will be defined as inertial frame.

- **The local frame, $f_l$**

  The location of the origin is fixed to the center of mass of the quadrotor. The axes are collinear to the inertial frame axes. That is, the local frame can only translate with respect to the inertial frame.

![Euler angles roll, pitch, and yaw representation.](image)
• **The local frame-1, \( f_{i-1} \)**

The location of the origin is fixed to the center of mass of the quadrotor. The local frame-1 is rotated about the \( x_i \) axis of the local frame (Figure 3.2) where the coordinate vector of local frame-1 is defined by the relation \( \mathbf{p}^{i-1} = R_{x_i} \mathbf{p}^i \).

• **The local frame-2, \( f_{i-2} \)**

The location of the origin is fixed to the center of mass of the quadrotor. The local frame-2 is rotated about the \( y_i \) axis of the local frame (Figure 3.2) where the coordinate vector of local frame-2 is defined by the relation \( \mathbf{p}^{i-2} = R_{y_i} \mathbf{p}^i \).

• **The local frame-3, \( f_{i-3} \)**

The location of the origin is fixed to the center of mass of the quadrotor. The local frame-3 is rotated about the \( z_i \) axis of the local frame (Figure 3.2) where the coordinate vector of local frame-3 is defined by the relation \( \mathbf{p}^{i-3} = R_{z_i} \mathbf{p}^i \).

---

![Figure 3.2: Representation of local frame \( f_{i-1}, f_{i-2}, f_{i-3} \) from left to right respectively.](image)


• **The body frame,** \(f_b\)

The location of the origin is fixed to the center of mass of the quadrotor. The body frame is rotated about the \(x_i\) axis, \(y_i\) axis, and then \(z_i\) axis of the local frame in that order. This is known as the roll-pitch-yaw rotation. The coordinate vector of body frame is defined by the relation \(\mathbf{p}^b = R \mathbf{p}^i\). Where \(R\) is the resulting rotational transformation of the roll-pitch-yaw sequence product, \(R_{x,\phi} R_{y,\theta} R_{z,\psi}\) defined by

\[
R = \begin{bmatrix}
c_{\phi}c_{\psi} & -c_{\phi}s_{\psi} & -s_{\phi} \\
c_{\phi}s_{\psi} + s_{\phi}s_{\theta}c_{\psi} & c_{\phi}s_{\psi} - s_{\phi}s_{\theta}s_{\psi} & -c_{\phi}c_{\theta} \\
s_{\phi}s_{\psi} - c_{\phi}s_{\theta}c_{\psi} & s_{\phi}s_{\psi} + c_{\phi}s_{\theta}s_{\psi} & c_{\phi}c_{\theta}
\end{bmatrix}
\] (3.4)

![Diagram of quadrotor coordinate system](image)

Figure 3.3: The coordinate system of quadrotor for modeling.
3.3 Quadrotor Simplified Dynamics

For an autonomous flight, it is desired to control the altitude, heading, and translational velocity of the quadrotor. In this section, we obtain a model for the altitude, heading and translational velocity, coupled with the inner-PID model that is used for the grey-box modeling approach. The quadrotor is assumed to be single rigid body with 6 DOF, symmetric, and that the forces and moments are dominantly from four rotors and gravity.

The dynamics of the quadrotor consist of a point mass \( m \in \mathbb{R} \), coordinate vector \( \mathbf{p} \in \mathbb{R}^3 \), angular moment of inertia \( \mathbf{J} \in \mathbb{R}^{3 \times 3} \), and angular orientation \( \mathbf{\rho} \in \mathbb{R}^3 \). Then the total force \( \mathbf{F} \in \mathbb{R}^3 \) and total moment \( \mathbf{M} \in \mathbb{R}^3 \) acting on the quadrotor is given by

\[
\mathbf{F} = m \ddot{\mathbf{p}}
\]

\[
\mathbf{M} = \mathbf{J} \ddot{\mathbf{\rho}} + \dot{\mathbf{\rho}} \times \mathbf{J} \dot{\mathbf{\rho}}
\]

The quadrotor’s state is changed by varying the thrust of the four rotors. The two pairs of propellers (1, 2) and (3, 4) turn in opposite directions (Figure 3.3). The \( F_k \) and \( \tau_k \) will denote the force and torque at the \( k^{th} \) rotor, respectively. The force acting on the quadrotor from the thrust in body frame is given by

\[
F_i = \sum_{k=1}^{4} F_k
\]

The rolling moment is generated by

\[
M_\phi = I \left( (F_1 + F_3) - (F_2 + F_4) \right)
\]
in body frame, where \( l \) is the moment arm (Figure 3.4), and constraint by the conditions

\[
0 = \sum_{k=1}^{4} \tau_k \quad \text{and} \quad F_1 = F_3, \ F_2 = F_4.
\]

And similarly, pitching moment is generated by

\[
M_\theta = l \left( (F_1 + F_4) - (F_2 + F_3) \right)
\]

in body frame, which is constraint by

\[
0 = \sum_{k=1}^{4} \tau_k \quad \text{and} \quad F_1 = F_4, \ F_2 = F_3.
\]

The yawing moment is caused by the difference in rotor torques as follows

\[
M_\psi = (\tau_1 + \tau_3) - (\tau_2 + \tau_4)
\]

in body frame.

For the quadrotor’s altitude to be constant, the total thrust \( F_i \) and gravitational force acting on the body must be equal, meaning \( F_i \cdot z_i = mg \). Then by Newton’s law, using equation (3.5), the sum of all the forces on the quadrotor in inertial frame is given by the following
\[
\begin{bmatrix}
\ddot{x}_i \\
\ddot{y}_i \\
\ddot{z}_i
\end{bmatrix} = \begin{bmatrix} 0 & 0 \\
0 & 0 \\
-mg & 0
\end{bmatrix} + R^T \begin{bmatrix} 0 \\
0 \\
F_i
\end{bmatrix}
\]

(3.11)

Where \( R^T \) is the transpose of the roll-pitch-yaw matrix from equation (3.4) which maps the body frame to local frame, \( R^T : f_b \rightarrow f_i \).

For an indoor flight, the roll and pitch orientations will be small hence \( \phi \approx 0 \), \( \theta \approx 0 \). Then the angular velocity of inertial frame and body frame are related by

\[
\begin{bmatrix} \dot{\phi}_i & \dot{\theta}_i & \dot{\psi}_i \end{bmatrix}^T \approx \begin{bmatrix} \dot{\phi}_b & \dot{\theta}_b & \dot{\psi}_b \end{bmatrix}^T.
\]

Moreover, the Coriolis term, which is small for nominal flight will be neglected. Then by combining small angle assumption with equation (3.6) and knowing \( J^i \approx J^b \), the sum of all the moments on the quadrotor in inertial frame is given by the simple form

\[
\begin{bmatrix}
J_{xx} & 0 & 0 \\
0 & J_{yy} & 0 \\
0 & 0 & J_{zz}
\end{bmatrix} \begin{bmatrix} \dot{\phi}_i \\
\dot{\theta}_i \\
\dot{\psi}_i
\end{bmatrix} = \begin{bmatrix} M_{\phi} \\
M_{\theta} \\
M_{\psi}
\end{bmatrix}
\]

(3.12)

Hence the simplified dynamics of quadrotor are

\[
\begin{aligned}
\ddot{x}_i &= \left( s_{\phi} s_{\psi} - c_{\phi} s_{\psi} s_{\phi} \right) \frac{F_i}{m} \\
\ddot{y}_i &= \left( s_{\phi} c_{\psi} + c_{\phi} s_{\psi} s_{\phi} \right) \frac{F_i}{m} \\
\ddot{z}_i &= -g + \left( c_{\phi} c_{\theta} \right) \frac{F_i}{m}
\end{aligned}
\]

(3.13)
\begin{align*}
\dot{\phi}_i &= \frac{l}{I_{xx}} \left( (F_1 + F_3) - (F_2 + F_4) \right) \\
\dot{\theta}_i &= \frac{l}{I_{yy}} \left( (F_1 + F_4) - (F_2 + F_3) \right) \\
\dot{\psi}_i &= \frac{1}{I_{zz}} \left( \tau_1 + \tau_3 \right) - \left( \tau_2 + \tau_4 \right) 
\end{align*}

(3.14)

The translational and rotational dynamics are given in the inertial frame. However, the control of velocity is more convenient with respect to quadrotor’s heading. In other words, expressing the translational dynamics in local frame-3 will make velocity quantities irrelevant to heading angle. Then by using equations (3.1) and (3.2), translational dynamics expressed in local frame-3 we obtain the following expression

\begin{align*}
\begin{bmatrix}
\ddot{x}_{i,3} \\
\ddot{y}_{i,3} \\
\ddot{z}_{i,3}
\end{bmatrix} &=
\begin{bmatrix}
0 \\
0 \\
-mg
\end{bmatrix} +
\begin{bmatrix}
R_{x,\phi} & R_{y,\phi} & T
\end{bmatrix}^T
\begin{bmatrix}
0 \\
0 \\
F_t
\end{bmatrix}
\end{align*}

(3.15)

which can be written of each components as

\begin{align*}
\ddot{x}_{i,3} &= (-s_{\phi}c_{\phi}) \frac{F_t}{m} \\
\ddot{y}_{i,3} &= (s_{\phi}) \frac{F_t}{m} \\
\ddot{z}_{i,3} &= -g + (c_{\phi}c_{\theta}) \frac{F_t}{m}
\end{align*}

(3.16)

Now we have an expression where the translational state of the quadrotor is only affected by the thrust force and roll, pitch orientations.
3.4 Inner-loop Dynamics

The inner-loop controller in on-board of the quadrotor consists of PID type which must be included in the dynamics model. The details of PID controller will be discussed in chapter 4. The general structure of PID controller in frequency domain is

\[
U(s) = K_p + \frac{K_I}{s} + K_D \cdot s
\]  

(3.17)

From equation (3.16), the input to the system will be defined as the followings

\[
\begin{align*}
 u_x & \triangleq \left(-s_\theta c_\phi\right) \frac{F_i}{m} \\
 u_y & \triangleq \left(s_\theta\right) \frac{F_i}{m} \\
 u_z & \triangleq -g + \left(c_\phi c_\theta\right) \frac{F_i}{m}
\end{align*}
\]

(3.18)

then the translational dynamics in local frame-3 has the simple form

\[
\begin{align*}
 \ddot{x}_{t-3} &= u_x \\
 \ddot{y}_{t-3} &= u_y \\
 \ddot{z}_{t-3} &= u_z
\end{align*}
\]

(3.19)

and the Laplace transform of equation (3.19) yields

\[
\begin{align*}
 s^2 X(s)_{t-3} &= U(s)_x \\
 s^2 Y(s)_{t-3} &= U(s)_y \\
 s^2 Z(s)_{t-3} &= U(s)_z
\end{align*}
\]

(3.20)

Similarly for the rotational dynamics from equation (3.14), the input to the system will be defined as the followings
\[
\begin{align*}
\lambda_{\phi} & \triangleq \frac{l}{J_{SS}} \left( (F_1 + F_3) - (F_2 + F_4) \right) \\
\lambda_{\theta} & \triangleq \frac{l}{J_{YY}} \left( (F_1 + F_4) - (F_2 + F_3) \right) \\
\lambda_{\psi} & \triangleq \frac{1}{J_{ZZ}} \left( (\tau_1 + \tau_3) - (\tau_2 + \tau_4) \right)
\end{align*}
\]

(3.21)

then the rotational dynamics in inertial frame has the simple form

\[
\begin{align*}
\dot{\phi}_i &= u_{\phi} \\
\dot{\theta}_i &= u_{\theta} \\
\dot{\psi}_i &= u_{\psi}
\end{align*}
\]

(3.22)

Assuming the sensor dynamics are constant and closing the loop with the PID controller, we have a candidate model of the quadrotor as the following transfer function for both translational and rotational form as a third order system

\[
T_{cl} = \frac{K_p s^2 + K_p s + K_I}{s^3 + K_p s^2 + K_p s + K_I}
\]

(3.23)

In more general form, to include unknown free parameters, candidate transfer function model will be defined as the

\[
T_{cl} = \frac{\alpha s^2 + \beta s + \gamma}{s^3 + \rho s^2 + \xi s + \sigma}
\]

(3.24)

where the coefficients are \( \alpha, \beta, \gamma, \rho, \xi, \sigma \in \mathbb{R} \). 

20
3.5 Altitude Velocity Estimation and Lowpass Filter Design

The altitude velocity of quadrotor is desired for both the simulation design and control design which is discussed in a later chapter. However, the altitude velocity is not available to the ground station. So velocity is estimated by the position measurement from ultrasound sensor. The discrete measurement of position is differentiated by the following equation

\[
\dot{z}_k = \frac{z_k - z_{k-1}}{h}
\]

where \(z_k\) is the \(k^{th}\) ultrasound sensor’s data sample and \(h\) is the sampling interval. The sampling period is relatively large for ultrasound sensor since there is a delay between transducer and the receiver. Hence estimating the velocity with large sampling interval with equation (3.25) will yield undesired high frequency noise amplified as shown in Figure 3.5. Hence the velocity signal must be filtered to be useful.

Figure 3.5: Plot of square wave input response of measured altitude position and noisy calculated altitude velocity.
A lowpass filter is used to smooth out high frequency noise in velocity signal. Specifically, a discrete first order lowpass filter is chosen for its simplicity, performance, and minimal computational cost. General model of a first order lowpass filter is

\[ H(s) = \frac{Y(s)}{U(s)} = \frac{1}{\tau \cdot s + 1} \]  

(3.26)

where \( \tau \) is a time-constant and \( U(s) \) and \( Y(s) \) are filter input and output, respectively.

Cross multiplying and expanding equation (3.26) yields

\[ \tau \cdot s \cdot Y(s) + Y(s) = U(s) \]  

(3.27)

Then taking the inverse Laplace of equation (3.27) to time domain yields

\[ \tau \cdot \dot{y}(t) + y(t) = u(t) \]  

(3.28)

Then in discrete representation, and substituting backward differentiation as derived above, we arrive at

\[ \tau \cdot \frac{y_k - y_{k-1}}{h} + y_k = u_k \]  

(3.29)

where \( y_k \) is the \( k^{th} \) sample and \( h \) is the sampling interval. Finally, solving for \( y_k \)

\[ y_k = (1 - \alpha) \cdot y_{k-1} + \alpha \cdot u_k \]  

(3.30)

where \( \alpha = \frac{h}{\tau + h} \) is the filter parameter.
As seen in equation (3.30), filter output is a function of input and its previous output. Hence it is an iterative algorithm that is simple to implement in any programming language with a very low computational cost.

It is important that the filter sampling interval is considerably smaller than the filter time-constant in order for the lowpass filter to suppress high frequency noise with minimal phase delays. Hence with general rule of thumb $5 \cdot h \leq \tau$, time-constant factor is iteratively found by collecting filtered altitude velocity data with varying time-constant factors. As shown in Figure 3.6, the filtered altitude velocity is a good estimation with no high frequency noise. However, some phase delay in the signal can be observed which is nominal. The time-constant factor is chosen as $\tau = 0.25$ which effectively filters the calculated noisy altitude velocity.

![Figure 3.6: Plot of sinusoidal input response of measured altitude position and filtered estimation of altitude velocity at different time constant factors. Phase delay can be observed at different time constant.](image)
3.6 System Identification

The dynamics of quadrotor has cross-coupling effects between parameters. However, cross-coupling effects are negligible during nominal flight. Moreover, the model assumes linear dynamics model that is valid only during nominal flight.

The decoupled linear sub-models are identified using the system identification toolbox in MATLAB version R2009a by MathWorks. For each sub-models, multiple set of input and output flight data is collected in Armory building at University of Illinois, which is a large indoor environment. For example, input of varying yawing rate to the quadrotor and its responses are collected as shown in Figure 3.7 in the time domain. Then the collected input-output data are processed using the prediction error method (PEM). This process is repeated for linear velocities of x and y, and altitude to identify the transfer function models that describe these dynamics. The PEM is an iterative nonlinear least-square method which uses optimization to minimize a cost function defined as

Figure 3.7: Multiple set of input-output data set of yaw motion for system identification.
where vector $e(t)$ is the difference between the measured output and predicted output of the model. In the cost function above, the subscript $N$ indicates that scalar $V_N$ is a function of number of data samples that become more accurate for larger values of $N$. For interested readers, the details on PEM can be found in [7].

To develop an accurate model of the system, time delay components must be considered. Time delays frequently arise in control systems. It comes from both the delays in process itself and from the processing of sensed signals. Since the quadrotor receives and sends data over WiFi signals that gets processed by a computer at discrete intervals, time delays cannot be neglected. Moreover, time delay always reduces the stability of a system; hence, it is crucial to add the time delay components in the dynamics model to predict its effect combined with the system. Then by equation (3.24), and including the time delay element, final decoupled sub-model transfer functions of the system are the followings

\[
T_\psi = \frac{-29.2024 \cdot s - 1.9742}{1.089 \cdot s^3 + 16.74 \cdot s^2 + s} \cdot e^{-0.051761 \cdot s} 
\]  
\[
(3.32)
\]

\[
T_z = \frac{22.6485 \cdot s + 0.3951}{1.187 \cdot s^3 + 27.3 \cdot s^2 + s} \cdot e^{-0.325 \cdot s} 
\]  
\[
(3.33)
\]

\[
T_\xi = \frac{-39422 \cdot s - 14.24}{1120 \cdot s^3 + 9549 \cdot s^2 + 3049 \cdot s + 1} \cdot e^{-0.016415 \cdot s} 
\]  
\[
(3.34)
\]
$$T_y = \frac{-257082 \cdot s - 4.3079}{7537 \cdot s^3 + 46068 \cdot s^2 + 33559 \cdot s + 1} \cdot e^{-0.064365 \cdot s}$$ (3.35)

In software, the inner-loop controller takes input value that ranges from -1 to 1 in floating number. It takes four arguments that commands yawing rate, altitude rate, and tilt angles. The max value 1 represents 100% of reference point limit. For instance, if the pitch angle was limited to 10 degrees then input value of 0.5 will make the quadrotor to track 5 degrees of pitch. The limits of set points are embedded in the onboard controller.

Due to non-linearity in tilt dynamics, as expected, its affect associated with the translational motions can be seen. The linear velocities in x and y directions do not increase linearly with varying step inputs in linear manner as shown in Figure 3.8. To justify the linear model, saturation block was implemented to limit the input amplitude inside the linear region. The saturation limit is set at ±0.5 for both the roll and pitch reference inputs to the inner-loop controllers.

![Plot of vx at varying step input](image)

Figure 3.8: Step input pitch% to the inner-loop controller and x-velocity response.
As shown in Figure 3.9, increasing step inputs within the linear region which are below the saturation limit indicates that the velocity responses along the x-axis increase in linear fashion.

### 3.7 Simulation Model and Validation

The simulation model of the quadrotor is developed in MATLAB/Simulink version R2009a by MathWorks. The decoupled sub-models of quadrotor in Simulink are shown in Figure 3.10, Figure 3.11, and Figure 3.12. The system simulation data is then compared to recorded flight data to validate the accuracy of the simulation model. The yaw and yaw rate model validation plot is shown in Figure 3.13. And the altitude, altitude rate model validation plot is shown in Figure 3.14.
Finally, the model validation plots for velocity-x and velocity-y are shown in Figure 3.15 and Figure 3.16, respectively.

Figure 3.10: Yaw dynamics simulation model in Simulink with the time delay element.

Figure 3.11: Altitude dynamics simulation model in Simulink with the time delay element. The frame of reference is defined in local frame-3.

Figure 3.12: Linear velocity of x and y dynamics model in Simulink with the time delay element. The frame of reference is defined in local frame-3 for the velocity state.
Figure 3.13: Comparison plots of simulated and measured yaw, yaw rate in inertial frame. As it can be seen from the plots, the simulation model predicts yaw and yaw rate with high accuracy.

Figure 3.14: Comparison plots of simulated and measured altitude, altitude velocity in local frame-3. Altitude plot shows that gravity acts as constant disturbance to the system. And altitude velocity shows slight delay due to low-pass filter dynamic. The simulation model accurately predicts both the altitude and altitude velocity response.
Figure 3.15: Comparison plots of simulated and measured linear x-velocity in local frame-3. Two different square wave inputs with amplitude within linear dynamics region show simulation model accurately predicts the x-velocity response.

Figure 3.16: Comparison plots of simulated and measured linear y-velocity in local frame-3. Two different square wave inputs with amplitude within linear dynamics region show simulation model accurately predicts the y-velocity response.
As mentioned previously, cross-coupling effects are negligible during nominal flight. To validate this claim, multiple inputs of square waves with nominal amplitudes are simultaneously sent to the quadrotor as shown in Figure 3.17 and Figure 3.18. The position and velocity responses are well predicted by the decoupled sub-model simulation.

Figure 3.17: Comparison plot of multiple input and output of measured and simulated position response. Plot shows during nominal flight, cross-coupling effects are negligible.

Figure 3.18: Comparison plot of multiple input and output of measured and simulated velocity response. Plot shows during nominal flight, cross-coupling effects are negligible.
CHAPTER 4

CONTROLLER DESIGN

“If everything seems under control, you’re just not going fast enough.”

-Mario Andretti

For an autonomous flight, quadrotor’s position and velocity must be controlled. In this chapter, decoupled sub-system models are utilized to design the outer-loop control. The root-locus design method is used to design the PID controller for all outer-loop control. The outer-loop will regulate the heading, altitude, and linear velocity of the quadrotor.

4.1 PID Controller

The classical PID controller (Figure 4.1) expressed in continuous time domain is

\[ u(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d \dot{e}(t) \]  \hspace{1cm} (4.1)

where \( u(t) \) is the control effort or input signal to the plant. The error signal \( e(t) \) is defined by the difference between the reference signal and output signal as the following

\[ e(t) = r(t) - y(t) \] \hspace{1cm} (4.2)
And the tuning parameters are $K_p, K_I, K_D \in \mathbb{R}$. The implementation of classical PID controller such as equation (4.1) on a continuous system is well behaved under normal operation. That is, if there is no sudden change in reference signal or the output signal. However, if sudden change in the signal is presented, then the derivative of error will have a sudden spike. Meaning, the slope of each discrete step in signal is infinitely large. This is more prevalent in discrete system since all signals have delays between each sampling period that can cause undesirable transient response of the controlled variable. Consequently, the PI controller is commonly used to avoid this issue. However, the derivative term is useful to slow down the rate of change of the controller output. The derivative term is effective in reducing the magnitude of the overshoot produced by the integral component.

In order to implement the derivative component, classical PID controller must be modified to avoid sudden spikes in the control effort signal. The modified PID controller will take the derivative of the measurement output $\dot{y}(t)$ rather than taking the derivative of an error signal $\dot{e}(t)$. Then the modified PID controller (Figure 4.1) takes the form

$$u(t) = K_p e(t) + K_I \int_0^t e(\tau) d\tau - K_D \dot{y}(t) \quad (4.3)$$

At steady state, the error term becomes constant so the derivative of the error signal becomes zero. Although the modified PID controller has a slightly different form than the classical, the behaviors of both controllers are identical near steady state.

In theory, an arbitrarily fast response and small steady state error to a constant disturbance can be achieved by increasing the gains in the PID controller. However, any
real system has limitation on how much output signal it can generate. Hence real systems have saturation limit on its output. So the saturation factor must be included in the control design. Another important factor that must be considered is the integral windup. When the controlled variable saturates, the integral term can grow without bound. For example, suppose some PID controlled motor has maximum output speed of 100 rpm and for few seconds it was commanded to spin at 150 rpm. For that few seconds, the feedback loop is broken and the system runs as an open-loop because the motor will remain at its speed limit independently of the process output. And within this period, the integral control will sum the error signals without bound which is known as the integral windup. Following this, if the new speed command was set within the saturation limit, it will take some time for the system to track the new set point since the integral component has to burn off the remaining summed error signals from the previous command. Thus integral windup can effectively slow down the transient response of the system significantly. To mitigate this issue, anti-integral windup algorithm is implemented, namely the resetting-factor. If the control effort signal is at its saturation limit, then the resetting-factor of 0.99 is multiplied to the integral component signal until the total control effort is below the saturation limit.

With the incorporation of modified PID structure and anti-integral windup component, the outer-loop controller is implemented to regulate the heading and altitude of the quadrotor. However, due to highly noisy readings for acceleration sensors, instead, PI controller is implemented for the velocity regulation. Nonetheless, performance of PI controllers are satisfactory. The details of the controller design and performance will be discussed in the following sections.
4.2 Root Locus Design Method

The root locus design method is a graphical way of representing the change in roots of the system with variation of some system parameter, which is the PID gain for our case.

For both position and velocity controllers, the main goal of design is to avoid overshoot with sufficient rise time and settling time. For a reasonable performance, the system design general objectives were set as the following:

- Rise time (10%-90%) of no more than ~1 second.
- Settling time (5%) of no more than ~2 second.
- Overshoot of no more than ~5%.
-Magnitude of control effort is below saturation limit.

The root locus design is implemented with **rltool** in MATLAB version R2009a by MathWorks. The gains are iteratively varied with design goal and input saturation limit in mind. The root locus plots are shown in Figure 4.2, Figure 4.3, Figure 4.4, and Figure 4.5 for yaw, altitude, velocity-x, and velocity-y respectively. And the gain parameters found are shown in Table 4-1. The gain parameters were tested with simulation model (Figure 4.6) prior to implementing the controllers on AR.Drone, which takes account for discretized signals. The result from the simulation model’s closed-loop step response and the design criteria were compared and validated as shown in Figure 4.7, Figure 4.8, Figure 4.9, and Figure 4.10. As it can be seen, the velocity controllers without the D component are predicted to perform well within the general design goal objectives and that performance of the controller was not sacrificed.
Figure 4.1: Block diagram of classical (left) and modified PID controller (right).

Table 4-1: Flight controller PID gains.

<table>
<thead>
<tr>
<th></th>
<th>Yaw</th>
<th>Altitude</th>
<th>Velocity-x</th>
<th>Velocity-y</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K_P$</td>
<td>3</td>
<td>1.5</td>
<td>0.4</td>
<td>0.25</td>
</tr>
<tr>
<td>$K_I$</td>
<td>0.03</td>
<td>0.025</td>
<td>0.18</td>
<td>0.2</td>
</tr>
<tr>
<td>$K_D$</td>
<td>0.3</td>
<td>0.1</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Figure 4.2: Root locus plot of yaw sub-system with PID gains within design goal criteria. The PI component is shown on left and D component is shown on the right.
Figure 4.3: Root locus plot of altitude sub-system with PID gains within design goal criteria. The PI component is shown on left and D component is shown on the right.

Figure 4.4: Root locus plot of velocity-x sub-system with PI gains within design goal criteria.
Figure 4.5: Root locus plot of velocity-y sub-system with PI gains within design goal criteria.

Figure 4.6: Complete decoupled sub-model of quadrotor simulator setup in Simulink.
Figure 4.7: The yaw PID controller performance simulated with gains obtained from root locus method.

Figure 4.8: The altitude PID controller performance simulated with gains obtained from root locus method.
Figure 4.9: The velocity-x PID controller performance simulated with gains obtained from root locus method.

Figure 4.10: The velocity-y PID controller performance simulated with gains obtained from root locus method.
4.3 Position Controller

The yaw PID controller shown in Figure 4.11 takes output value of yaw and yaw rate signals from the quadrotor and generates an error signal which is inputted back to the quadrotor thus closing the outer-loop of the system. The integration of error signal in software is done with the trapezoidal rule for each time step. To avoid integral windup, resetting-factor block is added. The resetting-factor block is activated when control effort signal is above the saturation limit of ±1, which then multiplies the integral component signal by 0.99. The detail of the resetting-factor component is shown in Figure 4.15.

![Figure 4.11: Yaw orientation PID controller structure in Simulink.](image)

For the altitude PID controller, as shown in Figure 4.12, takes output value of altitude sent from the quadrotor, however, the altitude rate is calculated on the ground-station. Then, the altitude and altitude rate are fed into the PID controller closing the outer-loop. Similarly, to negate integral-windup issue, the resetting-factor is set at 0.99.

The position controllers are implemented in software by following the simulation structure and are given a step input. The resulting reference yaw step response is shown in Figure 4.13. The simulation model and measured data are identical which demonstrates the accuracy of the simulation model and validates the control gains designed from the
root locus method. The reference altitude step response is shown in Figure 4.13. The measured response of the altitude has slightly more delay than the simulated response. This may be an indication that there exists a higher order component than what is modeled. Nonetheless, the performance of the altitude regulation is satisfactory and falls within the system design general objectives. The rise time is sufficiently fast with no overshoot.

Figure 4.12: Altitude position PID controller structure in Simulink.

Figure 4.13: Quadrotor response to step reference yaw input. It shows that simulated and measured models are identical and the performance falls within the design criteria.
As mentioned previously, the resetting-factor block is crucial so it is integrated with all outer-loop controllers to avoid the integral windup. The block representation of resetting-factor below is implemented in software for all outer-loop controllers.

![Simulink structure of anti-integral windup resetting-factor component.](image)

Figure 4.15: Simulink structure of anti-integral windup resetting-factor component. It activates when the control effort is beyond the saturation limit which then multiplies the integral error component by 0.99 to reduce the control effort back to below saturation limit.

Figure 4.14: Quadrotor response to a step reference altitude input. The plot shows that simulated and measured responses are slightly offset by more time delay than modeled. But it demonstrated satisfactory performance which falls within the design criteria.
4.4 Velocity Controller

For both the velocity in x and y directions, the PI controller takes the structure shown in Figure 4.16. The velocity controllers take output values of linear velocity sent from the quadrotor. The error signal is generated by comparing the reference input, which is then used to generate a control effort input to the quadrotor thus closing the outer-loop of the system. The reference velocity-x step response is shown in Figure 4.17. Although the rise time is well predicted compared to the simulation model, there is a greater overshoot than anticipated. The main reason for this can be due to noisy value of linear velocity onboard. As mentioned before, the velocity is calculated onboard using the combinations of IMU and optical flow type algorithm. In addition, it can be due to non-linearity of the dynamics model combined with the un-modeled higher order component. However, the steady state response is satisfactory and overall performance is acceptable.

Similarly for velocity along y-axis, reference step response shows slightly different response than the simulated response, namely the rise time (Figure 4.18). The discrepancy can be due to same reasoning as discussed above. Again, the overall performance and steady state response is acceptable.

Figure 4.16: Linear velocity x and y PI controller structure in Simulink.
Figure 4.17: Quadrotor response to a step reference velocity-x input. The plot shows that simulated and measured responses are slightly offset by more overshoot than modeled.

Figure 4.18: Quadrotor response to a step reference velocity-y input. The plot shows that simulated and measured responses are slightly offset by slower rise time than modeled.
4.5 Dead Reckoning

In order to design a path tracking control, the position of quadrotor must be known. The altitude information comes from the ultrasound sensor that measures distance between the quadrotor and ground. However, the translation position of x and y is not directly measured. The information of quadrotor’s position respect to the starting point, or the origin in the world frame can be obtained by dead reckoning. The dead reckoning is a process of estimating one’s position based on the knowledge of previously determined position. The method works well in simulation and in practice for relatively short amount of distances. However, it is an open-loop system which degrades over time with unboundedly increasing error.

The error comes from many sources and it varies for different systems. For the quadrotor, the velocity readings in local frame-3 are integrated over time. Then it is projected back to the world frame by rotational matrix from equation (3.4) to obtain the quadrotor’s position. Since the velocity readings are processed in discrete time, performing a numerical integration accumulates error over time which is known as the truncation error. Moreover, if the velocity sensor reading contains slightest error, then the error signals are integrated along with the correct readings. These are some factors that cause drifting of the position estimations by dead reckoning.

However, as mentioned, dead reckoning method is reasonably accurate for short amount of distance but must be corrected before the readings become unreliable. The detail of correcting the quadrotor’s position is discussed in the Chapter 5.
4.6 Flight Path Tracking Control

Given complete control over the quadrotor’s heading, altitude, and velocity, the flight path can now be generated and tracked. A trajectory controller described in [8] generates a line segment that connects points $P_n^d$ to $P_{n+1}^d$ where $P \in \mathbb{R}^3$. But here, we take a different approach. Given a desired waypoint $P \in \mathbb{R}^3$, at every time step, the quadrotor generates new heading angle $\alpha(t)$ (Figure 4.19). This approach is more passive in a sense that the quadrotor does not try to catch up to the line path when disturbed but instead, generates a new heading angle from current position to the waypoint. Then with the current position of quadrotor in world frame, and the desired velocity of travel $\dot{x}_n^d$ expressed in local frame-3, the error signals can be generated as the following

$$
\begin{align*}
    e_\varphi &= \alpha(t) \\
    e_k &= (P_{n+1}^d - P(t)) \cdot \hat{k}_n \\
    e_i &= (\dot{x}_n^d - \dot{x}(t)) \cdot \hat{i}_n \\
    e_j &= (0 - \dot{y}(t)) \cdot \hat{j}_n
\end{align*}
$$

(4.4)

The error signals are fed back into the position and velocity PID controllers designed from previous sections. Hence, the flight path controller consists of piecewise PI and PID control which are expressed as

Figure 4.19: A schematic of flight path controller. The heading angle is continuously updated between the waypoints.
\[ u_y(t) = K_p e_y(t) + K_\tau \int_0^t e_y(\tau) d\tau - K_D \dot{\psi}(t) \]
\[ u_z(t) = K_p e_z(t) + K_\tau \int_0^t \dot{e}_z(\tau) d\tau - K_D \dot{z}(t) \]
\[ u_x(t) = K_p e_x(t) + K_\tau \int_0^t e_x(\tau) d\tau \]
\[ u_y(t) = K_p e_y(t) + K_\tau \int_0^t e_y(\tau) d\tau \]

(4.5)

The next flight path is scheduled if the following condition is satisfied

\[ r - \|P(t)\| \leq \varepsilon \] (4.6)

where \( r \) is a radius of sphere centered about the waypoint and \( \varepsilon \in \mathbb{R} \). The path controller fixes the heading angle and desired altitude prior to taking off to the next scheduled waypoint.

Performance of the flight path controller is shown in Figure 4.20 and Figure 4.21.

Figure 4.20: Plot of 2-D flight trajectory path measured by IMU.
The path shown in both plots were given desired heading velocity of 0.3 m/s with the goal radius of 0.15 meters. For the 3-D flight path, the velocities are show in Figure 4.22. The heading velocity input of 0.3 m/s is well tracked in between each waypoints as well as the velocity of y at nearly 0.0 m/s.

Figure 4.21: Plot of 3-D flight trajectory path with roll, pitch, and yaw represented by 3-D rendered model airplane path measured by IMU.

Figure 4.22: Plot of quadrotor linear velocities in local frame-3 during 3-D flight trajectory. The change in velocity-z indicates each waypoint.
The AR.Drone is equipped with two onboard cameras, one vertical and horizontal. These cameras can be used for more than capturing images and videos. In this chapter, using computer vision techniques, implementations of yaw drifting correction, localization by landmarks, and autonomous flight by using vanishing point are discussed. The flight environment is GPS denied indoor where the mapping of the environment is known.

5.1 Landmark Detection

Many methods of vision-based localization by landmarks utilize either a natural or artificial landmarks. Natural landmarks are proper for both indoor and outdoor environments. However, detecting natural landmarks requires intensive image processing and training prior to the implementation stage [9]. In contrast, an artificial landmark is simple yet powerful tool for localization in indoor environment.
The landmarks are constructed from a pair of circular color objects as shown in Figure 5.1. The image data sent from the quadrotor is in Red-Green-Blue (RGB) color space. To threshold each image, RGB color space is transformed into Hue-Saturation-Value (HSV) color space for convenience. As shown in Figure 5.2, geometrically, RGB is represented as a cube whereas HSV is a cone. So RGB space is essentially Euclidean space, which makes it difficult to threshold certain range of colors. For instance, if colors close to purple is desired, the color range that will satisfy purple in RGB will include points along the red and blue axis. In contrast, in HSV space, the hue angle which represents the color can be picked initially. Then the brightness and purity can be varied, hence in HSV space, unwanted colors can be avoided when thresholding for certain color.

The HSV image is further processed by applying 2-D Gaussian filter. A symmetric Gaussian kernel in general form is defined as the following

$$G_x(x, y) = \frac{1}{2\pi \sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

(5.1)

This smoothing kernel creates a weighted average that weights pixels at its center much more than at its boundaries. That is, it suppresses noise by forcing neighboring pixels to look like each other.

Figure 5.1: Artificial landmarks. The circular shapes have radius of 6 centimeters.
The kernel window of 5x5 is chosen with $\sigma = 1.25$ depicted in Figure 5.3. The filter parameters are empirically determined by comparing suppression of high frequency noises in binary threshold image.

By pairing two circular shapes as one landmark, absolute orientation of the landmark is obtained. First, moments of the binary threshold image is found. The $i, j$ moment for $k^{th}$ marker is defined as the following

$$m_{ij}(k) = \sum_{r,c} r^i c^j I_k(r,c)$$

where $r$ and $c$ is the row and column values in pixel coordinate frame and $I(r,c)$ is the binary image value. Then for each $k^{th}$ marker, the centroid is obtained by

$$\begin{align*}
\bar{r}_k &= \frac{\sum_{r,c} r I_k(r,c)}{\sum_{r,c} I_k(r,c)} \\
\bar{c}_k &= \frac{\sum_{r,c} c I_k(r,c)}{\sum_{r,c} I_k(r,c)}
\end{align*}$$

Figure 5.2: Geometric representation of RGB(left) and HSV(right).
Knowing the centroids of each circular object, orientation of the landmark is obtained by connecting the centroids. Hence the absolute orientation of the landmark respect to the world frame is defined by assigning one of the circular shapes any arbitrary direction. The detected landmarks are shown in Figure 5.4.

Figure 5.4: 176x144 resolution pictures of detected landmarks taken by vertical camera. Each circular color object is smoothed by Gaussian filter, and then threshold is applied for each object in HSV color space. The result of threshold image is displayed in binary image where picture (a) is orange, (b) is pink, (d) is green, and (e) is blue. Picture (c) is the original image before processing, and picture (f) shows the centroids for each object and orientation line.
5.2 Yaw Drift Correction by Landmark

As mentioned previously, the measurements of yaw attitude readings suffer from bias offset drift. Although the readings are accurate for short amount of time, drift effect cannot be ignored as time increases. The roll and pitch attitudes are compensated by sensing the gravitational acceleration vector. However, this is not possible for yaw attitude due to gravitational acceleration vector being parallel to the yaw axis of the reference coordinate system.

The landmarks can be positioned perpendicular to the yaw axis of the reference coordinate system by laying them on the floor. And knowing the orientation of the landmark, yaw drift can be compensated by constantly updating the bias offset when landmarks are visible to the quadrotor’s vertical camera. The Figure 5.5 shows a picture of the landmark on the floor captured by the quadrotor while hovering over it. The quadrotor’s heading is referenced at zero degree and regulated by the PID controller. The result of comparison between compensated and uncompensated yaw attitude is shown in Figure 5.6. As shown, just over 14 minutes of hovering, the uncompensated yaw attitude is drifted to 110 degrees.

Figure 5.5: Detected landmark picture taken by vertical camera while in hovering flight.
This method gives precise correction of yaw attitude. Similarly, the x and y position of the quadrotor can be corrected. The altitude of the quadrotor is estimated by the ultrasound sensor attached to the bottom of the quadrotor hence the altitude readings do not suffer from any drift issues. Then the localization problem is simplified to 2-D problem where the coordinates of x and y for quadrotor in the world coordinates of frame must be estimated. This can be done by having a priori knowledge of the 2-D coordinates of landmarks in the world frame. To solve this problem, one must be able to relate the landmark coordinates in the world frame to pixel coordinates in the image frame and to the quadrotor’s body coordinates, and then back to the world frame.

This is possible by obtaining intrinsic and extrinsic parameters of the camera which is discussed in the following section.

Figure 5.6: Plot of roll, pitch, corrected and uncorrected yaw attitude during hovering flight. The heading is fixed at zero degree by the yaw PID controller. The uncorrected yaw attitude drifts from its true yaw attitude of 0 to 110 degrees within 15 minutes.
5.3 Camera Calibration

In order to infer any coordinate information from the landmarks on the floor, the intrinsic and the extrinsic parameters of the vertical camera must be known.

There are several methods of camera calibration that include linear and non-linear methods [10], [11]. The non-linear methods are used to model a detail structure of the intrinsic matrix of the camera. However, for a low-cost camera with low resolutions such as equipped on AR.Drone, linear calibration method will suffice. We use method described in [10] to model the intrinsic matrix of the vertical camera. First, the unit aspect ratio is checked by using the calibration grid as shown in Figure 5.7. The unit aspect ratio is determined to be approximately 1. Then with assumption of zero skew, and modeling the camera as a pinhole lens, we obtain the following equations

\[
\begin{align*}
    u &= u_o - f \frac{x}{z} \\
    v &= v_o - f \frac{y}{z}
\end{align*}
\]  

(5.4)

where \(u, v\) are coordinates in the image plane, and \(x, y, z\) are coordinate in camera reference frame that describe a point in the world frame. The schematic of the camera system model is shown in Figure 5.8.

![Figure 5.7: Picture of calibration grid used to compute the ratio of focal length in y-pixel and in x-pixel. The aspect ratio computed is determined to be approximately 1.](image)
The intrinsic parameters of the camera can be expressed in the matrix form as

$$K = \begin{bmatrix} f & 0 & u_0 \\ 0 & f & v_0 \\ 0 & 0 & 1 \end{bmatrix} \quad (5.5)$$

where $u_0, v_0$ are the Origins in the image plane that is collinear with the camera frame’s origin through the optical axis. There are three unknowns presented in equation (5.5) which requires system of three equations. Three mutually orthogonal vanishing points in the image can be used to solve for the three unknowns by utilizing properties of orthogonality. By definition, a vanishing point is a point in a perspective image to which

Figure 5.8: Schematic of the vertical camera system model.
parallel lines that is not parallel to the image plane appear to converge. And orthogonal vectors in Euclidean space has the property defined by the following

\[
\begin{align*}
P_i^T P_j &= 0 \\
P_j^T P_k &= 0 \\
P_i^T P_k &= 0
\end{align*}
\]

(5.6)

where the subscripts \(i, j, k\) indicate unit vectors that are mutually orthogonal to each other. Now the image coordinates can be related to the point in world coordinate respect to the camera frame by the following equations

\[
\begin{align*}
p_i &= K R P_i \\
p_j &= K R P_j \\
p_k &= K R P_k
\end{align*}
\]

(5.7)

The matrix \(R\) is the rotational matrix that relate camera frame to the world frame. Taking the inverse of the intrinsic matrix and taking the transpose of expression we see that equation (5.7) can be described as

\[
\begin{align*}
p_i^T \left( K^{-1} \right)^T &= P_i^T R^T \\
p_j^T \left( K^{-1} \right)^T &= P_j^T R^T \\
p_k^T \left( K^{-1} \right)^T &= P_k^T R^T
\end{align*}
\]

(5.8)

and since the rotational matrix belongs special orthogonal group of order 3, equation (5.8) can be expressed as
\[
\begin{align*}
&\, p^*_i\left(K^{-1}\right)^TR = P^F_i \\
&\, p^*_j\left(K^{-1}\right)^TR = P^F_j \\
&\, p^*_k\left(K^{-1}\right)^TR = P^F_k
\end{align*}
\] (5.9)

Then multiplying both side by orthogonal vectors yield

\[
\begin{align*}
&\, p^*_i\left(K^{-1}\right)^TRP = 0 \\
&\, p^*_j\left(K^{-1}\right)^TRP = 0 \\
&\, p^*_k\left(K^{-1}\right)^TRP = 0
\end{align*}
\] (5.10)

and by substituting in equation (5.7), finally we have

\[
\begin{align*}
&\, p^*_i\left(K^{-1}\right)^TR^{-1}p = 0 \\
&\, p^*_j\left(K^{-1}\right)^TR^{-1}p = 0 \\
&\, p^*_k\left(K^{-1}\right)^TR^{-1}p = 0
\end{align*}
\] (5.11)

The three vanishing points expressed in homogeneous coordinates are obtained by capturing a photo of a paper box with gridlines and calculating the three vanishing points in the image. Several different orientations are taken (Figure 5.9) to average the result of the calculations which can minimize measuring error from each image.

Figure 5.9: Different orientations of paper box used to calibrate the vertical camera.
Table 5-1: Results of camera calibration intrinsic parameters. The units are in pixel length.

<table>
<thead>
<tr>
<th></th>
<th>Setup 1</th>
<th>Setup 2</th>
<th>Setup 3</th>
<th>Setup 4</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>( u_0 )</td>
<td>86.071</td>
<td>98.766</td>
<td>83.832</td>
<td>83.340</td>
<td>88.003</td>
</tr>
<tr>
<td>( v_0 )</td>
<td>77.218</td>
<td>55.782</td>
<td>76.424</td>
<td>69.209</td>
<td>69.66</td>
</tr>
<tr>
<td>( f )</td>
<td>205.336</td>
<td>209.469</td>
<td>201.682</td>
<td>197.645</td>
<td>203.533</td>
</tr>
</tbody>
</table>

The vanishing point calculation is performed by using Matlab as a graphical tool (Figure 5.10). The vanishing point found minimizes square distance error from all intersection points found in each direction. The result is shown in Table 5-1 where all the units are expressed in pixel length.

The camera frame’s origin is assumed to be at the origin of the body frame. From equation (5.4), since we know the altitude \( z \), focal length \( f \), object in image coordinate \( u,v \), and the optical center \( u_o,v_o \) at any given time, we can solve the following

\[
x = (u - u_o) \frac{z}{f} \\
y = (v - v_o) \frac{z}{f}
\]  

(5.12)

where \( x, y \) are the coordinate point of detected object’s centroid expressed in camera coordinate frame. Then by using the extrinsic equation of the camera we have
\[
O_c^w = P^w - R_c^w p^r
\]  
(5.13)

where the values of rotational matrix \( R_c^w \) can be obtained by roll, pitch, and yaw readings from the onboard sensors, and \( p^r \) can be solved by the equation (5.12), and finally, the coordinates of the \( P^w \) is given as the location of the landmark in the world frame. Hence we can obtain the \( x, y \) coordinates of the quadrotor respect to the world frame given that the quadrotor observes the landmarks.

To test the accuracy of intrinsic model, the quadrotor is attached to a fixture above ground where the vertical camera’s optical axis is perpendicular to the surface plane as shown in Figure 5.11. Then small object is placed on the plane which is detected by the camera and the centroid location is calculated by the method described above. After taking measurements, the ultrasound sensor gave readings with approximately \( \pm 30 \text{mm} \) error, and the detected object coordinates had approximately \( \pm 15 \text{mm} \) for both along the x and y axis.

Now, with a good model of the camera, quadrotor can be localized by observing the known landmarks in an indoor environment.

Figure 5.11: Test setup for camera intrinsic model validation. The blue dot indicates object’s centroid detected by the vertical camera. Measurements are taken with a scale.
5.4 Vanishing Point Detection

When flying down a long corridor (Figure 5.12), the flight path controller developed in previous chapter has limitations. As discussed previously, the IMU suffers from drifting. Then the slightest error in initial heading direction will propagate over distance and will fly the quadrotor towards the wall. This issue can be negated by providing multiple landmarks along the track to correct the heading, but the number of landmarks required will proportionally increase as the corridor’s distance increase. Another approach to negate this issue is by utilizing a proximity sensor such as infrared sensor. However, mini-UAVs such as AR.Drone have limited load capacity so adding on extra sensors will reduce flight time.

The onboard cameras can be used to obtain useful information from the scene in the corridor. As described in [12], straight lines in the corridor can be used to navigate the robot by detecting vanishing points in the scene. In theory, vanishing points in the corridor should be located at the center, but because of noisy response from the image, further processing is required to estimate the true vanishing point in the scene.

![Images of corridors in Mechanical Engineering Laboratory at UIUC.](image)

Figure 5.12: Images of corridors in Mechanical Engineering Laboratory at UIUC.
The edges in the scene are obtained by using *Canny Edge Detection* algorithm [13]. Then, straight lines in the detected edge image are found by Hough transform method [14]. The Hough transform method uses voting scheme to find a line. The detected edge image is in binary form, so each pixel in the binary image can be part of some set of possible lines. The line is parameterized in polar coordinates by

\[ \rho = x \cos(\theta) + y \sin(\theta) \]  

(5.14)

where \( \rho \) is the perpendicular distance to the line respect to the origin and \( \theta \) is the angle of \( \rho \). Then for each point in the binary image, two parameters that describe the point are plotted in the accumulator plane also known as the Hough space. Then appropriate bin sizes are chosen and threshold value is set to obtain lines in the binary image (Figure 5.13). There are many tunable parameters to consider when working with canny edge detection and Hough transform. These values are obtained empirically by running captured video frames of the corridor offline and tuning the parameters prior to

Figure 5.13: MEL corridor image edge detection and Hough space representation.
implementing it on the quadrotor. The parameters used for this thesis are included in the Appendix: Companion DVD inside the ‘Source Code/’ folder.

Finally, the vanishing point is obtained by calculating the intersection point of the lines found in the image by taking the cross products. As mentioned briefly, because of noise in the image, the intersection points do not lie in the same coordinates. There are many different methods proposed to estimate the true vanishing point. For a real-time application, the method should be computational cheap and fast. In [15], voting scheme similar to Hough transform is used to estimate the vanishing point. We take a different approach here. To estimate the vanishing point

- Ignore extreme lines in the current image frame $F_i$ that are close to vertical or horizontal.
- Find the intersection points by $I_k \times I_{k+1}$ where $k$ is the total number of lines found in the current image frame $F_i$.
- Take the median value of the intersection points calculated.
- Take the weighted average (lowpass filter) with median point found from the previous image frame $F_{i-1}$ and median point found in current image frame $F_i$.

The result of the edge and vanishing point detection in the corridor is shown in Figure 5.14. Unlike ground robots, when the quadrotor is in flight, the image plane will rotate from pitching and rolling motions. Hence in order to filter out vertical or horizontal lines in world frame, the image plane must be rotated back to its original frame. The rotational matrix in equation (3.1) and (3.2) is applied to column vector $[u, v, 0]^T$. 
The result of the image plane rotation and filtering out vertical and horizontal lines is shown in Figure 5.15.

Figure 5.14: Images of edge detection and vanishing point detection in the corridor during real-time flight. The yellow circle indicates result of median value in the current frame, and the green circle indicates the weighted average of the vanishing point with the previous frame’s vanishing point.

Figure 5.15: Result of applying rotational matrix to image plane. The blue vertical line in image (b) represents true vertical lines in the world frame. The rotated images (a) and (c) from positive and negative roll orientation, the true vertical lines are ignored. Only the red lines are used for vanishing point calculation.
5.5 Vanishing Point Navigation

The vanishing point in the image frame can generate an input signal to the quadrotor to enable autonomous navigation through the corridor. The heading of the quadrotor is regulated by PD controller, and the error signal is defined by

\[ e_v = (0 - v_a) \]  

(5.15)

where \( v_a \) is the normalized pixel distance between the vanishing point and image center (Figure 5.16). The altitude and velocities are regulated by the same method as described in equation (4.4). Where velocity of x and altitude in local frame-3 takes user defined reference and velocity of y in local frame-3 is regulated to zero. So we have a vanishing point navigation controller defined by

\[
\begin{align*}
    u_v(t) &= K_p e_v(t) - K_D \dot{\psi}(t)_k \\
    u_z(t) &= K_p e_z(t) + K_I \int_0^t \dot{e}_z(\tau) d\tau - K_D \dot{z}(t)_k \\
    u_x(t) &= K_p e_x(t) + K_I \int_0^t e_x(\tau) d\tau \\
    u_y(t) &= K_p e_y(t) + K_I \int_0^t e_y(\tau) d\tau
\end{align*}
\]

(5.16)

Figure 5.16: The center line represents optical center and the \( v_a \) is the pixel distance of vanishing point in image plane to the optical center along the horizontal direction.
5.6 Localization by Landmarks

The combinations of vanishing point navigation controller with object detection ability, the quadrotor is able to navigate through a corridor and localize itself. The experiment is done at Mechanical Engineering Laboratory (MEL) corridor at UIUC. As shown in Figure 5.17, two landmarks are placed in the corridor. When the landmark is detected by the horizontal camera, the quadrotor tracks the landmark by generating an error signal shaped by a constant to convert the normalized landmark’s centroid pixel location to take units in radians. The horizontal camera has wide view angle of 93° so $\frac{\pi}{46.5}$ is multiplied to the error signal and a simple proportional controller is implemented. When the tracked landmark goes out of sight, the camera is immediately switched to the vertical camera. Then the quadrotor sweeps over the landmark and is able to localize given the landmark’s coordinate and orientation in the world frame. The result of localization through the corridor is shown in Figure 5.18 and snap shot of real-time flight GUI is shown in Figure 5.19. There are two videos of this flight included in Appendix Companion DVD inside ‘Media/Corridor_Straight_Flight/’ folder.

![Figure 5.17: Landmarks in the MEL corridor.](image-url)
Figure 5.18: Result showing successful localization and dead reckoning error at corridor of MEL. The quadrotor uses vanishing points to navigate through the corridor autonomously. The dead reckoning position error grows to approximately 11 meters over the course of this experiment. Two landmarks are used over a 20 meters of distance traveled down the corridor.

Figure 5.19: GUI layout displays the quadrotor in action.
5.7 Autonomous Flight and Corner Turn

The vanishing point disappears as the quadrotor gets close to the end of the corridor. Hence the vanishing point navigation alone cannot control the quadrotor to turn the corners in the corridor. By combining flight path controller developed previously with the vanishing point navigation controller, corner turn is achieved.

Because the dead reckoning position estimate is unreliable, at least one landmark per 10 meters of travel is required for reasonable performance. To navigate through two corridors that span over 40 meters of distance, four landmarks are used (Figure 5.20). Two landmarks per corridor are used where one of them is placed at near the corner turn. Given accurate current pose near the corner, flight path controller can rely on inertial sensor data to turn the corner and proceed with vanishing point navigation. The result is shown in Figure 5.21 with each landmark locations plotted. Prior to take off, the yaw orientation is drifted to -9.17 degrees, which propagates the position estimation error until it is corrected. This demonstrates sensitivity of dead reckoning method to initial conditions. In addition, there is discrepancy between true ending position and assumed ending position which is due to absence of landmarks in the 3rd corridor. Of course, this can be corrected with placing extra landmarks. Nevertheless, the experiment of corner turn navigation is successfully executed by combining two controllers with minimal number of landmarks placed.

There are two videos of this flight included in Appendix Companion DVD inside ‘Media/Corridor_Full_Flight/’ folder.
Figure 5.20: Four different landmarks used for corner turn navigation and localization.

Figure 5.21: Result showing successful autonomous navigation and localization through multiple corridors. The dead reckoning position and uncorrected yaw orientation accumulates significant amount of error throughout the course of this experiment.
CHAPTER 6
CONCLUSION

In this Chapter, contents of this thesis and the results are summarized. Then, the future directions of this research are discussed.

6.1 Summary

In this thesis, modeling, design of controllers, autonomous navigation, and solving localization problem for mini-UAV quadrotor in indoor environments was considered. While deriving a full dynamical model of a quadrotor is complicated by many factors including coupling effect, non-linearity, multivariable, aerodynamic effect, and friction, a simplified approach can be used to find an approximate sub-models of the system one desires to control. We showed that the decoupled sub-models of a system can be derived from the grey-box modeling approach with a system identification method. And that decoupled sub-models can be appropriate for nominal flight application such as for indoor flight. We discussed PID control design by root locus method and modifications necessary to implement the controllers on the quadrotor. Then, new flight path tracking controller was proposed which enabled the quadrotor to fly to desired waypoints in passive manner.
We introduced new and simpler method of estimating the vanishing point in the corridor with satisfactory performance. This method is computational cheap and is appropriate for real-time applications. Then we showed that using a simple artificial landmark consisting of two colored circular objects can be used to localize the quadrotor including yaw orientation drift correction. Lastly, we showed that total of four landmarks were enough for the quadrotor to localize itself in the corridor of over 40 meters in distance traveled.

All of the work above combined enabled the mini-UAV quadrotor to successfully navigate autonomously and localize itself by using landmarks in an indoor environment.

6.2 Future Work

Future work will extend to use of more robust landmarks detection method. That is, although the artificial landmarks worked well inside MEL, in an indoor environment where there may be many colored objects that overlap with the marker’s color, will present difficulties of implementing current method.

One possible approach to deal with this issue is to use more sophisticated object detection method such as scale-invariant feature method [16]. This method utilizes keypoints in the object and not sorely just on the color of objects.

Lastly, the velocity controls can be improved. One possible approach is to design a digital filter based on probabilistic approach to get better velocity estimation.
The Matlab simulation model and source code used for this thesis along with the videos of the quadrotor in flight are included in the supplemental files. Figure A below lists the contents.

Figure A: Contents included in the supplemental files. The ‘Media’ folder contains videos of the quadrotor in flight which include autonomous navigation and localization in the MEL corridor from both the quadrotor’s view and 3rd person’s point of view. And some extra videos are included as well. The ‘Source Code’ folder contains code written in C/C++ for all implementations including the code for GUI. The ‘System Simulation Model’ folder contains Matlab code and Simulink file of the decoupled sub-system simulation model for the quadrotor.
REFERENCES


