DEVELOPMENT OF INTELLIGENT UNMANNED AERIAL VEHICLES WITH EFFECTIVE SENSE AND AVOID CAPABILITIES

ANG ZONG YAO, KEVIN

(*B.Eng.(Hons.), NTU*)

A THESIS SUBMITTED

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

ELECTRICAL AND COMPUTER ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2014

Declaration

I hereby declare that the thesis is my original work and it has been written by me in its entirety. I have duly acknowledged all the sources of information which have been used in the thesis.

This thesis has also not been submitted for any degree in any university previously.

[illegible]

Ang Zong Yao, Kevin
31st October 2014

Acknowledgments

First and foremost, I will like to express my upmost and sincere gratitude to my supervisor, Prof. Ben M. Chen, for his acceptance and motivation towards my Ph.D studies. He accepted me into my Ph.D studies even though I had been in the workforce for four years and have not been in contact with much of academia. His patience and knowledge inspired me to push hard for my studies and to achieve things I could only dream about.

I would also like to express my sincere thanks to my supervisors from Defense Science Organization (DSO) National Laboratories, Dr Poh Eng Kee, Dr Chen Chang and Dr Rodney Teo for their rich experience in national defence related projects and the guidance they offered me during my studies.

Special thanks are given to Prof Wang Biao, Prof Luo Delin, Dr Dong Miaobo, Dr Peng Kemao, Dr Lin Feng, Dr Cai Guowei and Dr Zhao Shiyu who are there whenever I had major theoretical problems in my project that needed solutions. Their perceptive views are enlightening and give me great motivation to implement different techniques in my research.

Dr Zheng Xiaolian and Bai Limiao who are my office mates provided me with much advice and knowledge everyday. They make my day really enjoyable so much so that I have lots of enthusiasm coming to work everyday.

I treat everyone in the NUS UAV Research Group as one big family and I will like to tell them my most heartfelt thanks. They are the ones that made my studies ever so enjoyable. They are Dr Dong Xiangxu, Dr Wang Fei, Phang Swee King, Cui Jinqiang, Li Kun, Lai Shupeng, Liu Peidong, Ke Yijie, Wang Kangli, Pang Tao, Bi Yingcai, Li Jiaxin Qing Hailong and Shan Mo. I will never forget all the international competitions that we have worked so hard for. The competitions are namely, DARPA UAVForge 2012 Competition in Georgia, USA. AVIC UAVGP 2013 in Beijing, China. IMAV Competition 2014 in Delft, the Netherlands.

I have two colleagues who have currently left the National University of Singapore for a brighter future in the USA and Canada. I will like to thank Ali Reza Partovi and Huang Rui who I worked with in the development of my quadrotor.

Lastly, I need to thank my parents, Mr Ang Taie Ping and Mrs Tan Hong Huay, and girlfriend Lin Jing who is always there for me when I needed support. They have worked really hard to take care of me during these long stretch of time. Without their understanding, kindness and care they gave me, it would be extremely difficult to finish my Ph.D studies.

Contents

Summary

This Ph.D thesis depicts the development of unmanned aerial vehicles (UAV) in hardware design as well as software algorithm development. The main UAV developed is a quadrotor and it has been thoroughly modeled and controlled through the onboard software system using our ground control station. The UAV is mounted with an advanced avionics system used for navigation and a stereo camera system used for obstacle sensing and navigational enhancements.

Obstacle detection capabilities are manifested through visual-based algorithms which allow general obstacles and specific obstacles such as power lines to be sensed. The algorithms proposed include stereo-based obstacle sensing that is capable of detecting obstacles to an accuracy of 10 cm within a 5 m range based on the camera's field of view. Visual-based navigation is also explored using visual odometry where the UAV's pose estimate is obtained by a fusion of visual-based odometry estimates and inertial measurement unit (IMU) readings using a Kalman filter. The proposed algorithm relies on the use of the Perspective-n-Points motion estimation and is shown to be more reliable than the Rigid Motion Computation as it computes relative motion estimation based on image feature points and their 3D world coordinate. The algorithm has shown to accumulate an error of less than 5% of the distance traveled.

An active stereo vision system has also been developed to operate in featureless environments such as indoor environments. The active stereo vision system makes use of a laser emitter to project features into an otherwise featureless environment. The stereo system then tracks these features and is able to generate a sparse 3D point cloud which could then be used for obstacle detection or navigational purposes.

In this thesis, novel ideas have been implemented in both hardware and software. In platform development, a hybrid reconfigurable UAV has been designed and built in hopes of having a more optimal platform to achieve navigation in urban environments. It is hoped that the visual-based algorithms could be ported to such an unconventional platform. As the platform could transform from a vertical VTOL form to that of a horizontal cruise form, having the vision

sensor switch its orientation could have interesting results. For example, if the vision sensor is facing forward during VTOL mode, it could be used for sensing obstacles or navigation. Then when the uav transforms into cruise flight mode, the vision sensor will be facing the ground and it could be used for image stitching to generate a 2D map of the area it flew over. An efficient onboard 2D map stitching algorithm has also been implemented in international competitions and will be covered in later chapters.

List of Tables

List of Figures

List of Symbols

Latin variables

acx_b,acy_b,acz_b	Body acceleration in body frame x,y,z axis
$\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}$	System matrices of a time-invariant linear system
A	Lift Reference Area
A_{chirp}	Amplitude of Chirp Signal
B	Stereo Camera Baseline
C_{x_l}	Left Camera Center
C_{x_r}	Right Camera Center
C_D	Drag Coefficient
C_L	Lift Coefficient
d	Disparity
D	Drag Force
e^x, e^y, e^z	North-East-Down inertial frame
e^{xb}, e^{yb}, e^{zb}	North-East-Down body frame
f_i	Motor Thrust
f_x, f_y	Focal Length in x & y direction
F^b	Force applied to body
$\mathbf{H}$	Homography Matrix
$I(x,\ y,\ t)$	Intensity at image point $(x,\ y)$ at time, t
J_{xx}, J_{yy}, J_{zz}	Rolling, pitching and yawing moment of inertia
$\mathbf{J}$	Moment of inertia of UAV
K	Camera Intrinsic Matrix
K_Ω , K_v	Motor constants
K_f	Rotor thrust lift coefficient

l, w	Length & Width of Centroid
L	Lift Force
m	Mass of UAV
M_{00}, M_{10}, M_{01}	First Moments
N	RANSAC Sample Size
p, q, r	Angular velocities in body frame
$P^C = [X, Y, Z]^T$	Coordinates of 3D point in camera frame
Q, R	Kalman Filter Covariance Matrix
Q_i	Reactive torque
Q	Image Point Set Q
$\mathbf{R}_{\text{b/g}}$	Rotation matrix from NED frame to body frame
$\mathbf{R}_{\text{g/b}}$	Rotation matrix from body frame to NED frame
R	Optimal Rotation Matrix
t	Optimal Translation Matrix
t	Time
u, v, w	Linear velocities in body frame
u_{a0}	Input trim value; aileron
u_{e0}	Input trim value; elevator
u_{th0}	Input trim value; throttle
u_{r0}	Input trim value; rudder
U_a	RC receiver input; aileron
U_e	RC receiver input; elevator
U_{th}	RC receiver input; throttle
U_r	RC receiver input; rudder
v_x^C	Linear velocity of camera frame in x-direction
v_y^C	Linear velocity of camera frame in y-direction
v_z^C	Linear velocity of camera frame in z-direction
x, y, z	Position coordinates in NED frame
x_c, y_c	Centroid Position

Greek variables

$\beta_x, \beta_y, \beta_z$	Accelerometer Bias
ρ	Density of air

ϕ, θ, ψ	Euler angles
Ω_i	Propeller's angular velocity
τ^b	Torque applied to body
$\tau_\phi, \tau_\theta, \tau_\psi$	roll, pitch and yaw torques
τ_{i}	Time constant of the motor dynamics
ω_x^C	Angular velocity of camera frame in x-direction
ω_y^C	Angular velocity of camera frame in y-direction
ω_z^C	Angular velocity of camera frame in z-direction

Acronyms

2-D	Two-Dimensional
3-D	Three-Dimensional
ABS	Acrylonitrile Butadiene Styrene
AHRS	Attitude and Heading Reference System
AOA	Angle of Attack
BM	Block Matching
CAD	Computer-aided Design
CFD	Computational Fluid Dynamics
CG	Center of Gravity
COTS	Commercial Off-the-Shelf
DLT	Direct Linear Transformation
DOF	Degrees-of-Freedom
DoG	Difference of Gaussian
EKF	Extended Kalman Filter
EPO	Expanded PolyOlefin
ESC	Electronic Speed Control
FEM	Finite Element Method
GCS	Ground Control System
GPS	Global Positioning System
GPS/INS	GPS-aided Inertial Navigation System
GUI	Graphic User Interface
IMU	Inertial Measurement Unit
KNN	Kth Nearest Neighbour

LoG	Laplacian of Gaussian
LQR	Linear Quadratic Regulation
LS	Least-squares
LSD	Line Segment Detector
LTI	Linear Time Invariant
MEMS	Micro-Electro-Mechanical Systems
NED	North-East-Down
NUS	National University of Singapore
OpenCV	Open Source Computer Vision
PCL	Point Cloud Library
PEM	Prediction Error Method
PNP	Perspective-n-Points
PPM	Pulse Position Modulation
PWM	Pulse Width Modulation
RANSAC	Random Sample Consensus
RC	Radio Control
RF	Radio Frequency
RPM	Revolutions Per Minute
SAD	Sum of Absolute Difference
SD	Secure Digital
SFM	Structure from Motion
SGBM	Semi-Global Block Matching
SIFT	Scale Invariant Feature Transform
SLAM	Simultaneous Localization and Mapping
SURF	Speeded Up Robust Features
SVD	Singular Value Decomposition
UAV	Unmanned Aerial Vehicle
VTOL	Vertical Take Off and Landing
WiFi	Wireless Fidelity

Chapter 1

Introduction

1.1 Motivation

In the recent few years, the role of autonomous robotics in human life has been getting more pivotal. In many situations, humans can be replaced by autonomous robots to deal with tasks in a more efficient and safer way. Among them, aerial robots with their ability to move easily in 3-dimensional (3D) space are potentially more viable in many applications where the robot's maneuverability is crucial. Autonomous aerial vehicles exhibit great potential to play in roles such as: data and image acquisition [1], localization of targets [2], surveillance, map building, target acquisition, search and rescue, multi-vehicle cooperative systems [3] [4] and others.

The rapid development of unmanned aerial vehicles resulted from the significant advancement of micro-electro-mechanical-system (MEMS) sensors and microprocessors, higher energy density Lithium Polymer batteries and more efficient and compact actuators, thus resulting in the rapid development of unmanned aerial vehicles. The vertical take-off and landing (VTOL) crafts due their capability for flying in many different flight missions have obtained more attention.

The helicopter as a VTOL aircraft, is able to take-off in a confined area, hover on the spot, perform horizontal flight movements and land vertically. However, besides these features, traditional helicopters have a complex architecture. The conventional helicopters requires a tail rotor to cancel the main rotor's reactive torque. They also typically need a large propeller and main rotor. Moreover, their flight control mechanism is relatively complicated. Other than the mechanical complexity of the main rotor cycle pitch mechanism, the helicopters' up and downwards motion control require the main rotor to maintain rotational speed while changing the pitch angle of rotor blades which needs a special mechanical configuration setup [5].

Although great success has been achieved, the development and applications of unmanned helicopters are still at its initial stage. It is attractive and necessary to investigate the potential of unmanned helicopters, and extend their applications in future. The capability of fully autonomous flying in an urban environment seems to point towards one of the main goals to a next generation Unmanned Aerial Vehicle (UAV). With advanced on-board sensors, a UAV is expected to see and avoid obstacles, as well as localize and navigate in city areas. These tasks are derived from both military and civilian requirements, such as giving soldiers in urban operation the ability to spot, identify, designate, and destroy targets effectively and keep them out of harm's way, or providing emergency relief workers a bird's-eye view of damage in search and rescue missions after natural disasters.

In this thesis, I will cover the development and verification of technologies enabling a UAV to operate in an urban environment with sense and avoid capabilities. The focus will be on UAV flight using vision-based technology to aid the development of UAV obstacle detection, navigation as well as mapping capabilities. The thesis covers the build-up of both simulation analysis as well as real-data testing. Studies will be performed to investigate various challenges facing UAV urban flight and provide potential solutions which are developed into functions that could be run onboard a UAV. Software simulations and flight demonstrations will then be conducted to verify the effectiveness of such algorithms.

It is undoubted that the latest trend in the unmanned aerial vehicles community is towards the creation of intelligent unmanned aerial vehicles, such as a sophisticated unmanned helicopter equipped with a vision enhanced navigation system [6], [7], [8]. Utilizing the maneuvering capabilities of the helicopter and the rich information of visual sensors, it aims to arrive at a versatile platform for a variety of applications such as navigation, surveillance, tracking, etc. More specifically, a vision system already becomes an indispensable part of a UAV system. In the last two decades, numerous vision systems for unmanned vehicles have been proposed by researchers world-wide to perform a wide range of tasks.

1.2 Literature Review on Non-active Range Sensing Technologies

In the last two decades, non-active range sensing technologies have gained much interests, especially the vision sensing technologies [9]. Compared to active sensing technologies, the non-active sensing technologies use passive sensors and do not emit any energy, which is important in special situations, such as a battleground. Although sophisticated sensors such as a radar or a laser scanner can provide accurate relative distance to objects in the surrounding environment, their cost and weight is not acceptable for low-cost and small-sized unmanned systems. Furthermore, all of them cannot identify targets and understand complicated environments.

Vision sensing technologies are employed by unmanned systems mainly due to their distinguishing advantages:

1. Vision systems can provide rich information on objects of interest and the surrounding environments, such as color, structure of scene and shape of objects;

2. Vision systems require natural light only and do not depend on any other signal source;

3. Vision systems are generally of low cost and light weight when compared to the other related sensing systems such as radars and laser scanners;

4. Vision systems use only passive sensors that do not emit any energy, so that the whole system is undetectable, and safer in special conditions, such as battle fields.

Although such integration of vision and the robots achieved remarkable success in the last two decades, the machine vision is still a challenge due to its inherent limitations [10]:

1. The way that biological vision works is still largely unknown and therefore hard to emulate on computers;

2. Attempts to ignore biological vision and to reinvent a sort of silicon-based vision has not been as successful as initially expected;

3. Computationally expensive for processing large image sequences.

Fortunately, thanks to the rapid growth of computer and electronic technologies, light-weight and powerful commercial processors become more and more feasible. In the following section, we will discuss the vision sensing technologies and their applications, as well as investigate the novel ideas, concepts and technologies behind these applications which we could implement in the goal for vision-based sensing used for UAVs.

1.2.1 Stereo Vision

According to the knowledge in computer vision, the most straightforward approach to measure the relative position is to use stereo vision technology. A stereo camera is a type of camera with two or more lenses which allows the camera system to simulate human binocular vision, and therefore gives it the ability to decipher depth information in a process known as stereo photography. The distance between the lenses in a stereo camera (the intra-axial distance) is about the distance between one's eyes (known as the intra-ocular distance) and is about 6.35 cm, though a longer base line (greater inter-camera distance) produces more extreme 3-dimensionality.

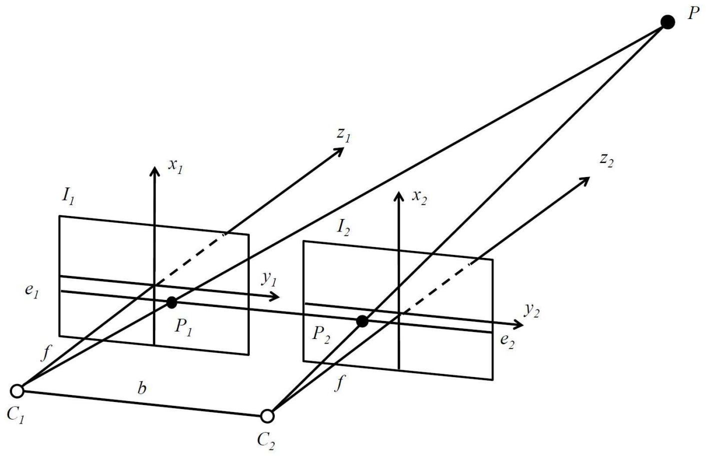


Figure 1.1: Stereo Vision Working Principle.

The fundamental idea behind stereo computer vision is that depth information can be computed when two points of reference are given for a single three-dimensional point. The method used to compute the depth of a point is called triangulation, which is illustrated in Fig. 1.1.

In order to understand triangulation using stereovision, let us look at a few definitions below first.

1. Epipolar plane: the plane defined by a 3D point and the optical centers. Or, equivalently, by an image point and the optical centers.
2. Epipolar line: the straight line of intersection of the epipolar plane with the image plane. It is the image in one camera of a ray through the optical center and image point in the other camera. All epipolar lines intersect at the epipole.

One of the key problems in stereo computation is finding corresponding points in the stereo images. Corresponding points are the projections in the two stereo images of a single point in the three-dimensional scene. The process to find those corresponding points is called “Stereo matching” or “Stereo Correspondence”.

In order to perform the triangulation calculation, features in the left image need to be matched to corresponding features in the right image. Stereo matching is the process by which a match score is computed for a given pixel location in either the right or left image coordinate frame. Two types of correspondence matching techniques are used, namely area-based matching and feature-based matching.

The study on the range measurement for the navigation of robots has already been reported in [11]. A stereo vision system was employed to augment a traditional sensor system for a UAV. However, the fixed base-line of the stereo camera constrains the measurement range, and the computational cost of processing stereo images also limits its usages in the applications of UAVs with limited payload and space.

Stereo vision can be used to obtain the depth information directly, however, the computational cost and weight is high. A stereo vision system set in a forward looking position could be an option used to realize the navigation, guidance and obstacle avoidance an autonomous UAV needs.

1.2.2 Optical Flow Techniques

In stereo vision, the relative distance between the UAV and the objects in the environment are directly measured through triangulation. Besides direct measurement, the indirect method of estimating the relative speed of objects detected is also frequently used. Such a technique was presented in [12] to navigate a UAV through urban canyons. Both the optic-flow approach and stereo vision technique were employed to hold the UAV in the center of the canyons safely, and avoid obstacles detected. Although the optical flow method is suitable for the motion estimation of UAVs in the forward flight condition, it cannot estimate the absolute position, which is required in applications, such as the drift-free hover. Optical flow is also used as a means for tracking features that are detected in an image and could be used to estimate motion.

Optical flow is described as the pattern of apparent motion of brightness objects, surfaces, and edges in a visual scene caused by the relative motion between an observer, which could be an eye or a camera, and the scene. Ideally the optical flow is the projection of the 3-dimensional velocity on the image. The initial hypothesis in extracting optical flow is the brightness constancy assumption, i.e, the intensity structures of local time-varying image regions are approximately constant under motion for at least a short duration.

Let $I(x,\ y,\ t)$ denote the image intensity of $(x,\ y)$ at time t, the brightness constancy assumption is given by

$$I(x+\frac{dx}{dt}\delta t, y+\frac{dy}{dt}\delta t, t+\delta t) = I(x,\ y,\ t)\ , \tag{1.1}$$

which indicates that the intensity does not change in a short time period. With the small movement assumption, the following image intensity, $I(x+\frac{dx}{dt}\delta t,\ y+\frac{dy}{dt}\delta t,\ t+\delta t)$, can be expressed in Taylor series:

$$I(x+\frac{dx}{dt}\delta t, y+\frac{dy}{dt}\delta t, t+\delta t) = I(x,\ y,\ t) + \frac{\partial I}{\partial x}\frac{dx}{dt} + \frac{\partial I}{\partial y}\frac{dy}{dt} + \frac{\partial I}{\partial t}\ , \tag{1.2}$$

which with (1.2) yield

$$\frac{dI}{dt} = \frac{\partial I}{\partial x}\frac{dx}{dt} + \frac{\partial I}{\partial y}\frac{dy}{dt} + \frac{\partial I}{\partial t} = 0 \tag{1.3}$$

We define

$$u = \frac{dx}{dt}\ , \quad v = \frac{dy}{dt}\ , \tag{1.4}$$

$$I_x = \frac{\partial I}{\partial y}\ , \quad I_y = \frac{\partial I}{\partial y}\ , \quad I_t = \frac{\partial I}{\partial t} \tag{1.5}$$

which with (1.3) obtain

$$I_x u + I_y v + I_t = 0 \quad \text{or} \quad \nabla I(x,y,t)\cdot(v_x, v_y, 0) + I_t(x,y,t) = 0\ , \tag{1.6}$$

where u, v are the x and y component of the velocity or optical flow of $I(x,y,t)$, and $I_x,\ I_y,\ I_t$ are the derivatives of the image at (x, y, t) in the corresponding directions. Here we have one constrain (1.6), but there are two unknowns u, v. To find the optical flow, additional information must be added to the problem statement to clearly define a single stable solution. This problem is often referred to as the aperture problem as stated in [13].

Horn-Schunck Optical Flow

A global smoothness term was introduced to obtain a function for estimating optical flow together with the gradient constraint (1.6) [14]. The estimated velocity field, $v(x,t) = (u(x,t), v(x,t))$

is constrained to minimize (1.7):

$$\int_{\Omega} [(\nabla I \cdot v + I_t)^2 + \lambda\,^2(|\nabla u\|^2 + |\nabla v|^2)]\, d\vec{x} \tag{1.7}$$

where the magnitude of λ reflects the influence of the smoothness term, and larger λ leads to a smoother flow. Iterative equations are used to minimize (1.7), and obtain the image velocity:

$$u^{k+1} = \bar{u}^k - \frac{I_x[I_x\bar{u}^k + I_y\bar{v}^k + I_t]}{\alpha^2 + I_x^2 + I_y^2} \tag{1.8}$$

$$v^{k+1} = \bar{v}^k - \frac{I_y[I_x\bar{u}^k + I_y\bar{v}^k + I_t]}{\alpha^2 + I_x^2 + I_y^2}\,, \tag{1.9}$$

where k denotes the iteration number, $u^0,\ v^0$ denotes initial velocity estimates which are set to zero, and $\bar{u}^k,\ \bar{v}^k$ denote neighborhood average of u^k, v^k.

The Horn-Schunck algorithm yields a high density of flow vectors, i.e. the flow information missing in inner parts of homogeneous objects is filled in from the motion boundaries. However, it is more sensitive to noise than other local methods. Noise that appears in images will result in high intensity gradients. They serve as weights in the data term of the regularization functional Eqn. 1.7. Since the smoothness term has a constant weight λ, smoothness is relatively less important at locations with high intensity gradients than elsewhere. Consequently, flow fields are less regularized at noisy image structures.

Lucas-Kanade Optical Flow

Based on the assumption that the flow is essentially constant in a local neighborhood of the pixel, Lucas and Kanade proposed a flow estimation technique based on the first-order derivatives of the image sequence [15]. A weighted least-square (LS) fit of first-order constraints (1.6) in each small spatial neighborhood Ω is formulated to calculate the optical flow of all the pixels in that neighborhood by minimizing (1.10):

$$\min \sum_{\vec{x}\in\Omega} W^2(\vec{x})\, [\nabla I(\vec{x},t) \cdot \vec{v} + I_t(\vec{x},t)]^2 \tag{1.10}$$

where $W(\vec{x})$ denotes a window function that gives more influence to constraints at the center of the neighborhood than those at the periphery. The solution to this least square problem can be obtained by solving the following linear system for n points $x_i \in \Omega$ at a single time t,

$$A^T W^2 A\vec{v} = A^T W^2 \vec{b} \tag{1.11}$$

where

$$\begin{aligned} A &= [\nabla I(\vec{x}_1), ..., \nabla I(\vec{x}_n)]^T, \\ W &= diag[W(\vec{x}_1), ..., W(\vec{x}_n)], \\ \vec{b} &= -[I_t(\vec{x}_1), ..., I_t(\vec{x}_n)]^T. \end{aligned}$$

The closed form solution of the linear system in (1.11) is obtained as

$$\vec{v} = [A^T W^2 A]^{-1} A^T W \vec{b}$$

The solution is possible when $A^T W^2 A$ is nonsingular, which could be seen as a 2×2 matrix:

$$A^T W^2 A = \begin{bmatrix} \sum W^2 I_x^2 & \sum W^2 I_x I_y \\ \sum W^2 I_x I_y & \sum W^2 I_y^2 \end{bmatrix} \tag{1.12}$$

One important advantage of this approach over Horn and Schunk [14] is the existence of a confidence measure. The smallest eigenvalue, λ_1 , of (1.12) provides a measure to distinguish estimates of normal velocity from 2-Dimensional (2D) velocity. And by processing information in a neighborhood, the Lucas-Kanade method can often resolve the inherent ambiguity of the optical flow equation. Another advantage is that this method is less sensitive to image noise than point-wise methods. However, because of its local processing characteristics, it cannot provide flow information in the interior of uniform regions of the image.

1.2.3 Feature Detection, Description & Matching

Features in the image sequences have to be extracted first before stereo vision or optical flow could be calculated. Features in different images should be well matched by good correspondence measures. Therefore the methods to detect and describe point correspondence between images becomes important. This could be accomplished in three steps:

- Detection: "Interest points" are selected at distinctive locations, such as corners, blobs, and T-junctions. The interest point *detector* should be *repeatable* under different viewing conditions;

- Description: Every interest point is described as a *descriptor*, which has to be distinctive and robust to noise, detection displacement and geometric and photometric deformation;

- Matching: The descriptor vectors are matched between consequent images based a distance measures, such as Mahalanobis or Euclidean distance.

Feature Point Detectors

The most widely used detector is the one proposed by Harris and Stephens[16]. This combined corner and edge detector is based on the local auto-correlation function and it performs with good consistency on natural imagery. Even though this method is invariant to image rotation, it is not scale-invariant. To tackle the scale problem, Lindeberg came up with an automatic scale selection in [17]. This makes it possible to detect interest points in an image at different characteristic scale. Both the determinant of the Hessian matrix and the Laplacian are evaluated to detect blob-like structures. To further improve Lindeberg's method, Mikolajczyk and Schmid [18] created a robust and affine-invariant feature detectors with high repeatability, coined Harris-Laplace and Hessian-Laplace. A scale-adapted Harris measure or the determinant of the Hessian matrix is used to select the location, and the Laplacian to select the scale. This algorithm can simultaneously adopt location information as well as scale and shape of the point's neighborhood. Focusing on speed, Lowe proposed to approximate the Laplacian of Gaussian (LoG) by a Difference of Gaussian (DoG) filter.

Mikolajczyk and Schmid made a comparison of the available scale and affine-invariant detection techniques [19], they claimed that Hessian-based detectors were more stable and repeatable than their Harris-based counterparts. Moreover, using the determinant of the Hessian matrix rather than its trace (the Laplacian) seemed advantageous, as it triggered less on elongated, ill-localized structures.

A large variety of feature description techniques exists, but the best has been the one presented by David Lowe [20]. The method is called the Scale Invariant Feature Transform (SIFT). It transforms an image into a large collection of local feature vectors, each of which is invariant to image translation, scaling, and rotation, and partially invariant to illumination changes and affine or 3D projection. It computes a histogram of local oriented gradients around the interest point and stores the bins in a 128-dimensional vector (8 orientation bins for each of 4×4 location bins).

Herbert Bay et al [21] proposed a novel scale and rotation-invariant detector and descriptor, coined Speeded Up Robust Features (SURF). It is partly inspired by the SIFT descriptor. SURF is several times faster than SIFT and claimed by its authors to be more robust against different image transformations than SIFT. SURF relies on integral images for image convolutions to reduce computation time and builds on the strengths of the leading existing detectors and descriptors, using a fast Hessian matrix-based measure for the detector and a distribution-based descriptor. It describes a distribution of Haar wavelet responses within the interest point neighborhood. Integral images are used for speed and only 64 dimensions are used reducing the time for feature computation and matching. The indexing step is based on the sign of the Laplacian, which increases the matching speed and the robustness of the descriptor.

1.3 Contribution of the Thesis

The contributions of the thesis are segmented into three main topics that address the development of UAVs with sensing capabilities. They are namely platform development and modeling, obstacle sensing, and vision-based navigation and environment mapping. Each topic is then covered in detail throughout multiple chapters and they are summarized as below.

Chapter 2 studies the different type of possible platforms that are capable of performing autonomous navigation in an urban built-up environment. In the study, it was determined that the current platform that fits the our application is the Quadrotor UAV. Modeling of the Quadrotor UAV was then discussed in detail in Section 2.4. One of the disadvantages of VTOL UAVs is due to its limited flight endurance capabilities when compared to fixed-wing UAVs. The VTOL UAVs are not able to successfully complete missions where they are required to fly long distances before reaching their target operational area. This problem is addressed and tested in real-flight with the development of a hybrid unconventional UAV described in Section 2.6.

In an urban built-up environment, there exist an inherent need for the UAVs to exhibit obstacle sensing capabilities while operating autonomously. Chapter 3 covers the capabilities of vision sensors as a primary sensor to detect obstacles in the generic case as well as the specific case. Upon detection of these obstacles, there are many pursuing techniques that could be applied to track the target. Section 3.7 describes a vision-based algorithm developed for tracking these obstacles. Upon tracking obstacles, avoidance strategies such as that of potential field [22] could be used.

Chapter 4 explores the novel use of vision sensors used for navigation. It covers the novel implementation of active stereo vision which could be used for navigation and obstacle detection in environments that are devoid of features. Chapter 5 goes in detail on the use of vision to aid navigation for UAVs. It depicts a stereo vision based odometry calculation that uses 3D-to-3D correspondences and stereo vision based pose estimation that uses 3D-to-2D correspondences. Both techniques could be used for UAV navigation in urban environments but we cover the advantages and disadvantages of each.

Apart from the need for obstacle detection, operators will usually require maps of the urban environment to be built. 2D stitched maps taken from a bird's eye view could capture detailed surface structures. Chapter 6 depicts map building in 2D for use by operators. This method was also used in a recent international UAV competition which our UAV team won first place.

Finally, we conclude our work done and findings in the whole thesis in Chapter 7. It then covers the possible future work that we could expand from the work of this thesis.

Chapter 2

Platform Development

2.1 Introduction

Quadrotors as compared to conventional rotorcrafts have a simpler mechanical structure and flight mechanism. Using four rotors in a symmetrical configuration removes the need of a tail rotor. In fact, since each pair of rotors rotate in opposite directions, the generated reactive torque are inherently canceled. The quadrotor's fixed pitch blades reduces the platform's mechanical complexity, and the relatively small propeller size could decrease mechanical vibration. From the maneuverability point of view, quadrotors are generally able to offer better performance over traditional helicopters. Quadrotors have a symmetrical rigid structure which can be considered as an omnidirectional vehicle. They can be configured such that right, left, front and back would have a relative direction. In particular, this means it potentially is able to fly in any direction without maintaining its heading towards the desired direction [5]. Furthermore, quadrotors have a relatively simple flight control mechanism which is only based on individual propellers' rotational speed.

However, quadrotors as with most VTOL aircraft have low performance in aspects of forward flight speed, range, and endurance. Although new platforms have been proposed to increase the performance of VTOL aircraft and in specific quadrotors in horizontal flight, through a simple modification on its orientation control, the improvement to the quadrotor's maneuverability in horizontal displacement can be expected.

In fact, the standard quadrotor is constructed from four propeller-rotor sets in a plus style. The quadrotor flies in horizontal plane by changing its attitude. The desired attitude is obtained by differing the rotors speed of different pairs of rotors. Hence, only two rotors are involved

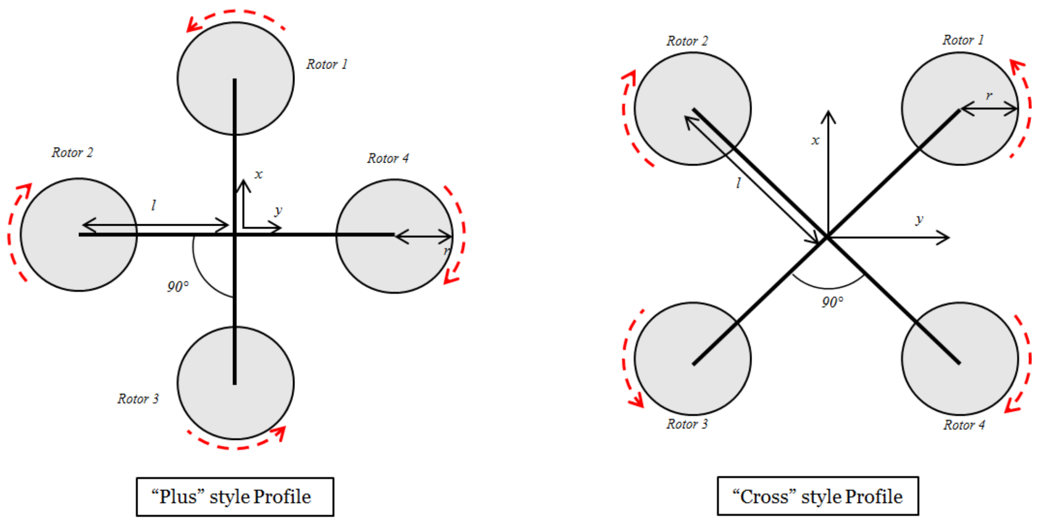


Figure 2.1: Quadrotor in plus and cross styles.

in horizontal movement which are defined along the body frame axes and lies on the quadrotor arms. By simply considering the quadrotor in a cross style comparing to the body frame (see Fig. 2.1), we are able to take advantage of using all four rotors in achieving horizontal displacement. In such a configuration all rotors participate together to rotate the platform around the desired axis of orientation. Thus for a quadrotor in the cross style when compared to the standard quadrotor, for the same desired motion, provides higher momentum which can increase the quadrotor's maneuverability performance.

2.2 Platform Selection

A thorough review of the available platforms and their problems surfaced after some research. Current market survey of existing platforms capable of flight in outdoor environments show that the platforms will require a minimum payload of near to 1 kg for onboard systems and sensors to realize fully autonomous control. As with outdoor environments, having the capability to resist wind is also a significant issue. Urban canyons frequently have large drafts as wind pass through them. Generally, smaller unmanned aerial vehicles (UAVs) are unable to fly outdoors due to this reason. Therefore, platform selection lies with selecting a platform that has the desired payload capability of 1 kg, have a large enough dimension and thrust to resist outdoor wind drafts and a decent flight endurance. One of our desired specifications of a small outdoor UAV is to have it as light-weight as possible to facilitate transportation. This will be kept in mind and achieved as much as possible.

A few frequently used platforms that show potential are mentioned below. But each of them have their limitations.

2.2.1 ESky Big Lama Co-axial Helicopter

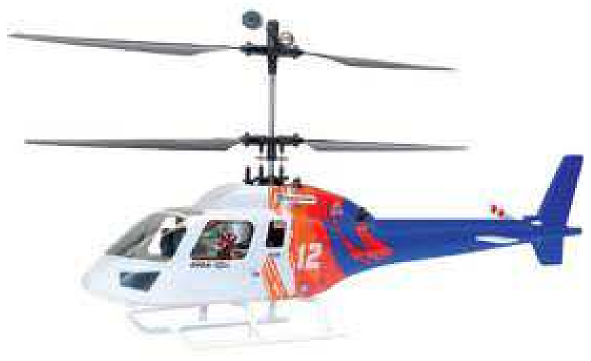


Figure 2.2: ESky Big Lama Co-axial Helicopter.

The ESky Big Lama Co-axial Helicopter shown in Fig. 2.2 is a light-weight platform which has a co-axial pair of propellers. The propellers rotate in contra-rotating directions to generate thrust and is mechanically stable due to the mechanical stabilizer above the propellers. The co-axial helicopter has a relatively good payload of about 400 g. However, with a basic onboard system, it would not be able to take-off comfortably. The Big Lama is also not capable of flying outdoors when the propulsion system is already taxed to its maximum due to the high payload and will not be able to resist strong winds.

2.2.2 Align T-Rex 450 Conventional Helicopter

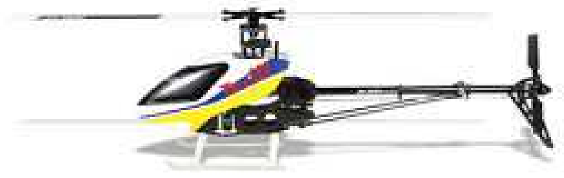


Figure 2.3: Align T-Rex 450 Conventional Helicopter.

The Align T-Rex 450 shown in Fig. 2.3 is one of the smallest hobby grade conventional helicopters available and it is built with outdoor flight in mind. It has a fuselage weight of 850 g but with only a payload of about 300 g. Thus, with its limited payload capabilities, it will not be possible to mount the sensor suite and onboard systems required for urban outdoor flight.

Therefore, as a start, platform selection will be geared towards having a platform to test and evaluate different approaches to the urban navigation problem before tackling the weight limitation issue. The quadrotor, being easy to model and having very good payload is chosen to be the platform of choice.

2.2.3 XAircraft X650 Quadrotor platform

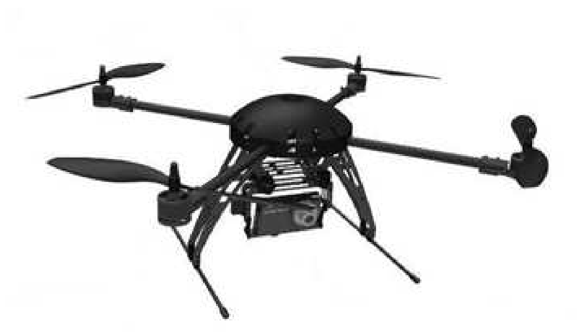


Figure 2.4: XAircraft X650 Carbon Fibre Quadrotor.

The X650 Quadrotor has a payload capability of up to 1 kg and able to fly in two different modes. Namely, the “Plus” style and the “Cross” style. The “Cross” style is chosen as the optimal choice as it has higher capabilities in aggressive, high velocity flight as well as allowing obstruction free sensor placement. The X650 Quadrotor has a programmable gyro onboard which covers the aspects of motor mixing and inner-loop stability in manual flight. Lastly, the quadrotor is chosen as the platform as it is easily scalable to include additions to its onboard system and sensors. The final state of the platform is to enable it to achieve autonomous flight in an urban environment. The platform has been modified to mount the onboard systems and sensor suite. The onboard system will house a Gumstix processor for control and the sensor suite will be made up of Point Grey vision sensors, SBG Systems Inertial Measurement Unit (IMU) and the controllable inner loop control board.

2.2.4 Related Work on Quadrotors

Many groups have worked on standard quadrotors and realized various controllers and scenarios based on these platforms. The STARMAC is one of the more successful outdoor platform from Stanford University. They build up their quadrotor based on the Dragonflyer III platform and developed their own avionics system. One of the first progress of this project is presented in [23]. They took the advantages of a sliding mode controller to stabilize the height and Linear Quadratic Regulation (LQR) method to control the attitude. An extensive dynamic modeling including aerodynamic analysis was done on STARMAC [24]. By using differential Global Positioning System (GPS), STARMAC succeeded in performing autonomous hovering flight with a duration of up to two minutes and within a $3\ \mathrm{m}$ circle [25]. In addition, an outdoor autonomous trajectory tracking were realized based on the STARMAC quadrotor [25] as well as modeling of the flapping propeller [26].

A research group in Australian National University developed a large scale quadrotor. In the preliminary study, platform design, fabrication and hardware development of the first design, MARK I, was described in [27]. The issue of insufficient thrust margin and unstable dynamic behavior led them to design the second platform MARK II [1]. The complete dynamic modeling and aerodynamic analysis is presented in [28]. Through this work, a discrete proportional-integral-derivative (PID) controller is used to realize the attitude control.

Another outdoor flying quadrotor was introduced in [29]. This group developed an advanced powerful avionic system based on the gumstix processor and cross bow IMU. A nonlinear hierarchical flight controller and a complete stability analysis is presented in this work. Based on the proposed controller, they succeeded in achieving autonomous take-off and landing, and an outstanding attitude and path tracking performance.

The indoor quadrotor also attracted many groups. A truly successful indoor quadrotor was built in MIT [30]. The goal of this platform is to realize a single or multi-vehicle health management system. The development of advanced equipped quadrotor for detection of target observation under indoor calamity environment was discussed in [31]. More advanced controllers are also implement on this type of helicopter. A robust adaptive and back stepping control was proposed and validated experimentally on a quadrotor in [32], [33].

2.3 Hardware and Software Development

2.3.1 Platform Design

This section aims to illustrate the systematic design and development of the quadrotor platform. The mechanical structure of the quadrotor platform will be introduced and the configuration of the rotor and propeller will be discussed. Development of the avionics board as a electronic heart of the vehicle will be extensively described with specifications and features of the autopilot components highlighted.

The body of the quadrotor should satisfy some important structural features. It should be strong enough to withstand all the forces produced by the rotors, the hardware and battery payload, and sufficiently lightweight to allow the mounting of extra sensors or using higher capacity batteries. It also should be relatively flexible to absorb the impact of a hard landing. Among various types of commercial quadrotor frame, we chose the X650 carbon fiber quadrotor from XAircraft. The platform body frame made from carbon fiber is strong and lightweight. It has four sets of rotors with efficient high strength $12\ \mathrm{in}$ propellers, high frequency $500\ \mathrm{Hz}$ Ultra-PWM (Pulse-width modulation) supported motor drivers and a programmable three axes gyro. The total weight of platform excluding the battery, receiver and avionic board is about $800\ \mathrm{g}$. We added one layer of carbon fiber mounting board on the center of the frame where the avionic board, Radio Control (RC) receiver, GPS module and battery regulator are placed. To ensure that the center of gravity is near the central axis, the battery is mounted under the center of the frame to counter the weight caused by adding the onboard system. Below shows the assembled platform.

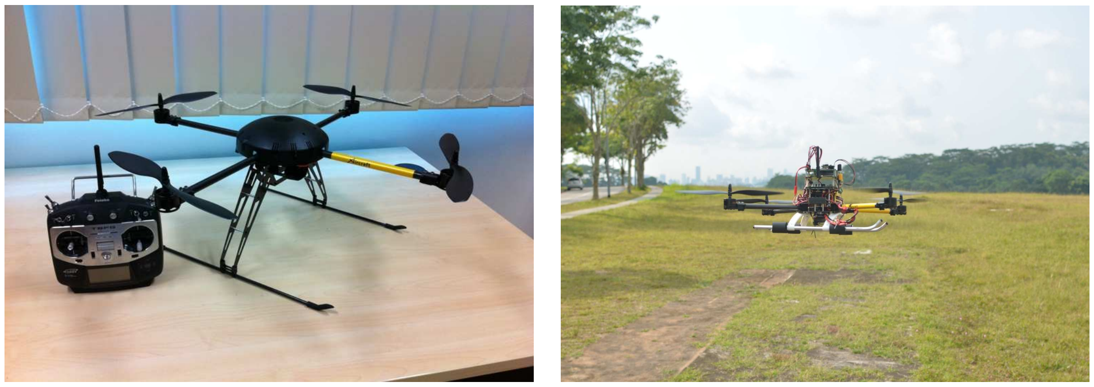


Figure 2.5: Left: X650 quadrotor assembled frame, Right: The developed quadrotor hovering.

2.3.2 Avionics System Design

In this project, we aim to design and development a lightweight, powerful, cheap and expandable avionics system. Since the goal is to use this setup for different types of mini-UAVs, it is essential to choose standard, low-cost and high quality components for each part. In fact, the current on-board system's weight does not exceed 100 g, which could be appropriate for most mini helicopters and airplanes. The block diagram of the on-board system is depicted in Fig. 2.6. The system is governed by the main computer board which is chosen to be the Gumstix Overo Fire processor supported by the Summit extension board, which is sufficiently powerful for normal control processing tasks. This embedded computer has enough communication ports to drive the other avionics boards. Also, it has an embedded Wi-Fi module which can be used as a communication link to the ground station or other UAVs' board. In addition, the Gumstix processor supports micro SD cards, which is useful for logging flight data.

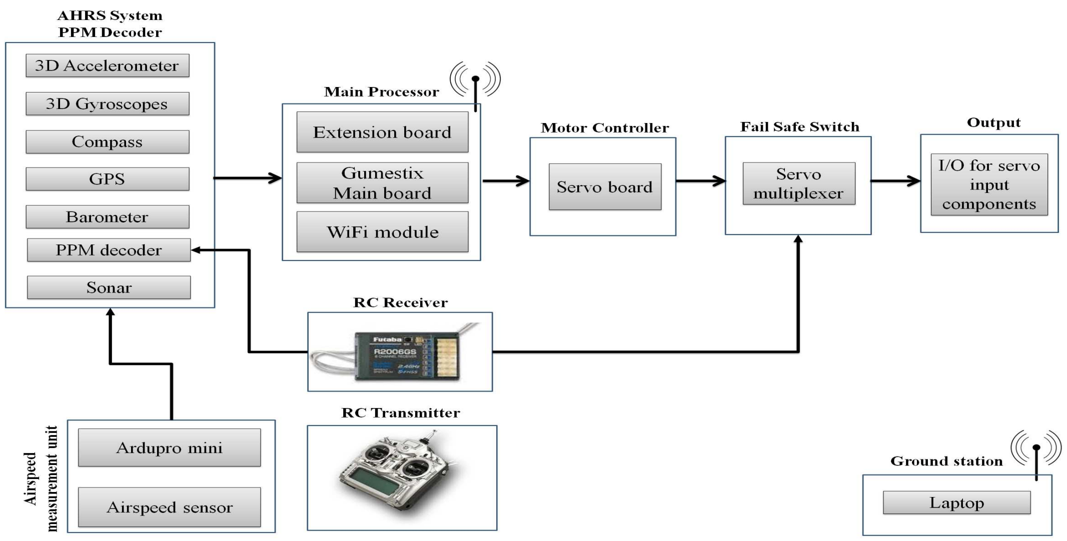


Figure 2.6: Avionics system block diagram.

The sensing core of the avionics setup is an inertial navigation system (INS). It consists of an IMU, GPS, barometer and ultrasonic sonar. Usually this component is most expensive item in avionics systems. However, since we are trying to achieve a low cost design, ArduIMU, one the cheapest and lightest IMU in the current market was selected. On contrary to its prices, the ArduIMU can provide accurate orientation data. Moreover, this IMU's programming is open source, which gave us the opportunity to modify the code based on own design demands. The GPS unit is responsible for ground position and velocity data and the Ublox GPS is chosen as an appropriate unit for our design as it can be easily matched with ArduIMU. However, the

GPS position it gives is not very accurate, specifically, altitude is quite inaccurate. To help fuse the GPS data, a barometer and ultrasonic sonar are added to the INS. The barometer gives air pressure and it is possible to estimate the relative height based on this information. Furthermore, for automatic take-off and landing scenarios, precise height measurement is necessary. The sonar is mounted under the platform facing ground and through the ultrasonic signal reflection analysis, the sonar provides quite accurate relative altitude data for INS. We also modified the IMU firmware to read the RC receiver Pulse-position modulation (PPM) signal and send it to the main processor. These data are necessary for model identification. The standard actuators in UAVs such as servo motors and rotor-speed controllers are driven by the PWM signal. A servo controller board is added to the setup to derive the actuators. A Pololu micro serial servo controller, due to it's compact design and light weight is a very suitable candidate for our design. One of the important part of any avionic system is the fail safe switch. Basically, during any autonomous flight, there are potentially dangerous situations where we need a reliable option to switch back to manual flight. The fail safe board is a hardware switch that acts as a servo multiplexer. By assigning one channel of RC receiver to the fail safe channel selection input, the pilot is able to switch back to manual flight mode overriding the autopilot system. In our setup, the data monitoring and sending of necessary commands are done by a ground control station. In fact, a laptop is linked to the on-board computer through Wi-Fi. The flight data are continuously sent to the ground station which is displayed for the user. The user during autonomous flight can also send predefined commands to the avionics system. Similar to all hobby aircrafts we have a Radio Frequency (RF) transmitter and receiver module for manual flight.

The aforementioned board's hardware and firmware have been developed from several years of consistent work in the NUS UAV group. [34], [35], [36], [37], [38].

Table 2.1: The software framework thread description.

Thread name	**Task description**
IMU	INS data fusion and servo reading
DLG	Data logging
CTL	Control
SVO	Servo driving
CMM	Communication to control ground station

The INS data fusion is handled in the IMU thread. The software activates this thread to read

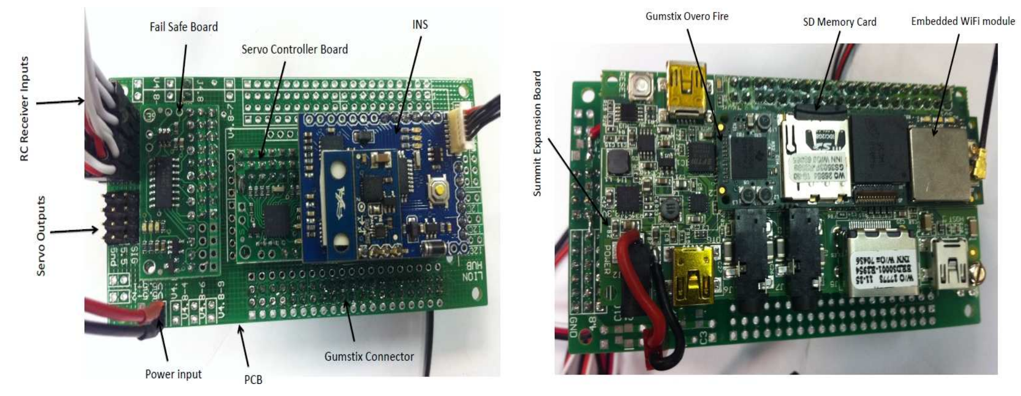
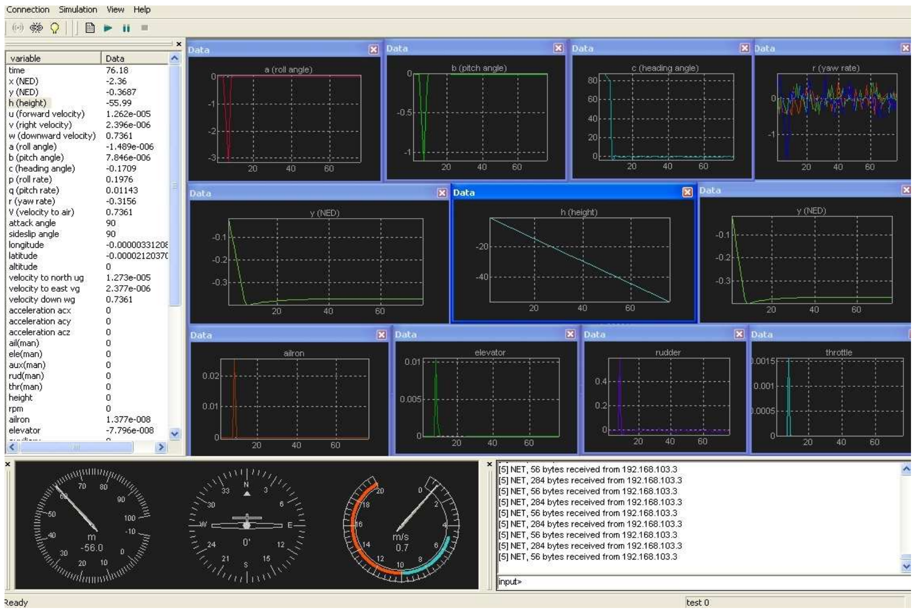


Figure 2.7: Avionics board.

the INS output at a refreshing rate of 50 Hz which in our setup also contains the RC receiver PPM data. The DLG thread helps by recording the flight data during every flight. Sampling time, sensors raw and filtered data, controller outputs and user defined variables are logged in a 2 GB SD memory card. The data are written in a text file format which can be easily read in a computer and imported to MATLAB or excel programs for post-flight analysis. The CTL thread realizes the flight controllers where the flight scheduling and hierarchical control algorithms are covered. In real flight, the controller reads the states from the IMU thread, but when hardware-in-the-loop simulation is applied, states are obtained from the identified dynamic model. Driving the servo and platform actuators is done through the SVO thread. This part gets the controller output from the CTL thread and converts it to the servo board input range. As it is mentioned previously, this board is mainly added to our setup when reading the gyro output is desired. Therefore, the SVO thread's main roles are sending the controller output to the servo board, reading the gyro output and scaling the received data accordingly for logging thread. The communication with ground control station is realized in the CMM thread where the quadrotor states, RC transmitter data are sent to ground station via Wi-Fi. The preliminary received command processing also is done in this thread.

The ground station has been developed by NUS UAV team [37]. The ground control station software is mainly used for monitoring the aerial vehicle's behavior, sending the flight task command and hardware-in-the-loop simulation. Fig. 2.8 shows the graphic user interface (GUI) of the ground control station. In left side of the software, the received states are listed; the user can choose one or several signals to be plotted in the graph windows. The monitored state variables mainly are the position, body and ground velocities, ground acceleration, attitude angles and angular rate, manual and auto servo outputs and GPS raw position data. At the bottom of



variable	Data
time	76.18
x (NED)	-2.36
y (NED)	-0.3687
h (height)	-55.99
u (forward velocity)	1.262e-005
v (right velocity)	2.396e-006
w (downward velocity)	0.7361
a (roll angle)	-1.489e-006
b (pitch angle)	7.846e-006
c (heading angle)	-0.1709
p (roll rate)	0.1976
q (pitch rate)	0.01143
r (yaw rate)	-0.3156
V (velocity to air)	0.7361
attack angle	90
sideslip angle	90
longitude	-0.00000331208
latitude	-0.00002120370
altitude	0
velocity to north ug	1.273e-005
velocity to east vg	2.377e-006
velocity down wg	0.7361
acceleration acx	0
acceleration acy	0
acceleration acz	0
ail(man)	0
ele(man)	0
aux(man)	0
rud(man)	0
thr(man)	0
height	0
rpm	0
ailron	1.377e-008
elevator	-7.796e-008

```
[5] NET, 56 bytes received from 192.168.103.3
[5] NET, 284 bytes received from 192.168.103.3
[5] NET, 56 bytes received from 192.168.103.3
[5] NET, 284 bytes received from 192.168.103.3
[5] NET, 56 bytes received from 192.168.103.3
[5] NET, 284 bytes received from 192.168.103.3
[5] NET, 56 bytes received from 192.168.103.3
[5] NET, 284 bytes received from 192.168.103.3
[5] NET, 56 bytes received from 192.168.103.3
input>
```

Figure 2.8: Ground control station (GCS).

the GUI, the command bar input is designed to obtain the command string from the user. The command strings are defined in the embedded onboard firmware as a task command. For instance, sending "hover" will be processed in the onboard system as a hover flight scenario, the task management layer will obtain the current position as a reference and call the outer-loop and inner-loop hovering controller functions to hold the quadrotor at its current position. However, some commands have two parts, the command name and the input argument. Trajectory tracking command and the desired path number as the command argument, is an example of this structure. Onboard, when a command is received it will be processed in several steps. First it is validated, and then extracted if it contains extra input. If the new command request a change of flight task, the corresponding flight controllers are executed in the control loop, the relevant variables such as references may be reloaded based on the current scenario, and depending on the flight condition the output trim values can be adjusted.

2.4 Quadrotor Modeling

2.4.1 Quadrotor Flight Mechanism

The quadrotor proposed in this paper is developed in a cross style(see Fig. 2.7) which has a different flight mechanism as compared to standard quadrotors as shown in Fig. 2.9. As with standard quadrotors, the basic motion of the quadrotor is realized by adjusting individual propeller's speed. The propellers have a fixed-pitch and their air flows point downwards to produce an upward thrust. On each end of the horizontal arms, a rotating motor is placed. The whole quadrotor setup consists of two clockwise rotating motors and two counter-clockwise rotating motors.

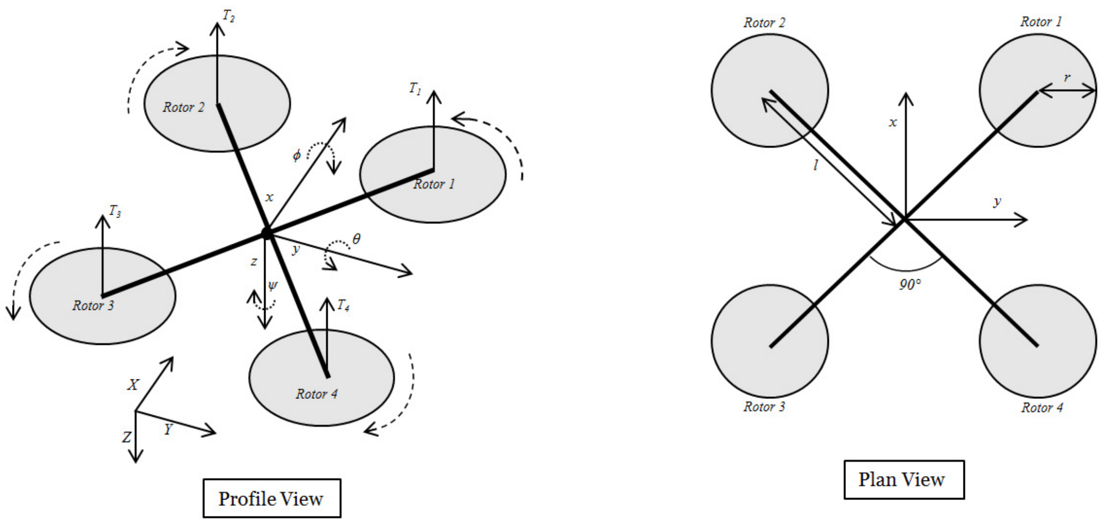


Figure 2.9: Freebody diagram of Quadrotor.

The quadrotor's translational motion requires tilting the platform towards the desired axis. Hence, similar to the traditional helicopters, the translational and rotational motion are coupled. Basically, changing the speed of one motor can cause a motion in three degrees of freedom (DOF). This phenomena is the reason that allows the quadrotor with six DOF to be controlled by only four inputs [39].

The thrust in the vertical direction is produced by the summation of all the rotors' force. Changing all four rotors' rotational speed by the same amount generates a vertical acceleration. In manual flight, this motion is controlled by the throttle channel. The pitching motion is produced by applying torque around the y axis. Changing the speed of the front pair of rotors against the back pair of rotors will result in the platform tilting around the y axis. The elevator channel controls this motion in manual flight. The rolling motion exhibits the same mechanism as pitching but around the x axis. The difference of the left and right pair of rotors provides rolling which is around the x axis. The pilot uses the aileron channel to induce this motion. Yawing motion is introduced by exerting torque around the z axis. Applying different rotational speed to the pair of counter rotating motors, causes a change of heading. In fact, the yaw motion is generated by the rotors' reactive torque. During manual flight, the rudder channel is used to

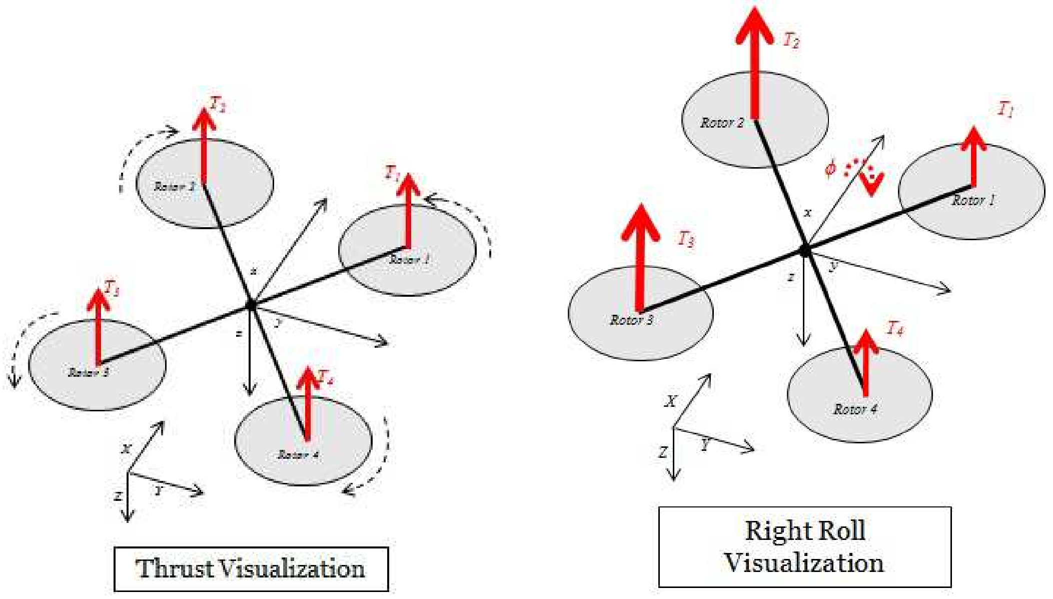


Figure 2.10: Thrust and Roll visualization.

change the quadrotor's heading. See Fig. 2.10 and Fig. 2.11 for the visualization.

Hence for hovering flight, all the rotors' thrust must be equivalent. Slight differences may cause the quadrotor to tilt and be unstable. Although the quadrotor flight control mechanism may seem simple, in real flight, it is almost humanly impossible to fly it without the help of a controller. For the manual flight, in order to assist the pilot, a commercial gyro will be used to control the quadrotor's attitude.

2.4.2 Quadrotor Dynamics

In this section, we aim to derive the mathematical model of the quadrotor in the cross style. The quadrotor dynamics are obtained through combining aerodynamics forces and blade theory. The quadrotor has four motors with propellers mounted on four arms extended at 90° to each other. Power is applied through the attached battery to create torque on the rotor shaft which in turn creates thrust for each rotor. Each of the rotors also creates torque about the blade area which affects the whole quadrotor's dynamics. An initial modeling of the Quadrotor is done while considering a limited flight envelope. Only hovering or low velocity flight are considered during the derivation of the dynamics shown below.

To achieve this goal, we first define the quadrotor dynamic and kinematic equations. The dynamic model describes how applied forces and torques result in translational and rotational accelerations. The relation between the vehicle's position and velocity are described through its

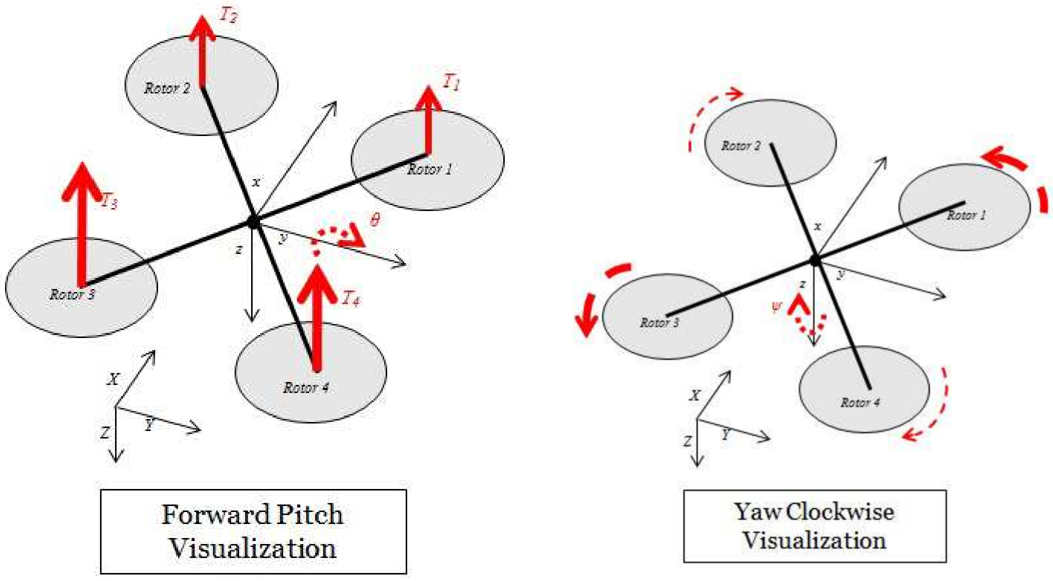


Figure 2.11: Pitch and Yaw visualization.

kinematic equations [40]. The inertial and body frames are introduced as shown in Fig. 2.12. The origin of the body frame is assumed to be at the center of gravity with the following notations defined. The North-East-Down inertial frame and body frame is defined as $F^I = (e^x, e^y, e^z)$ and $F^B = (e^{xb}, e^{yb}, e^{zb})$ respectively. Body orientation with respect to NED frame is defined as $\eta \triangleq [\phi, \theta, \psi]^T$ where we can also define the respective roll, pitch and yaw, body frame angular rates; $\omega^b \triangleq [p, q, r]^T$ as rolling rate, pitching rate and yawing rate. Position in the inertia frame is defined as $\xi \triangleq [x, y, z]^T$ and body frame velocities as $v^b \triangleq [u, v, w]^T$.

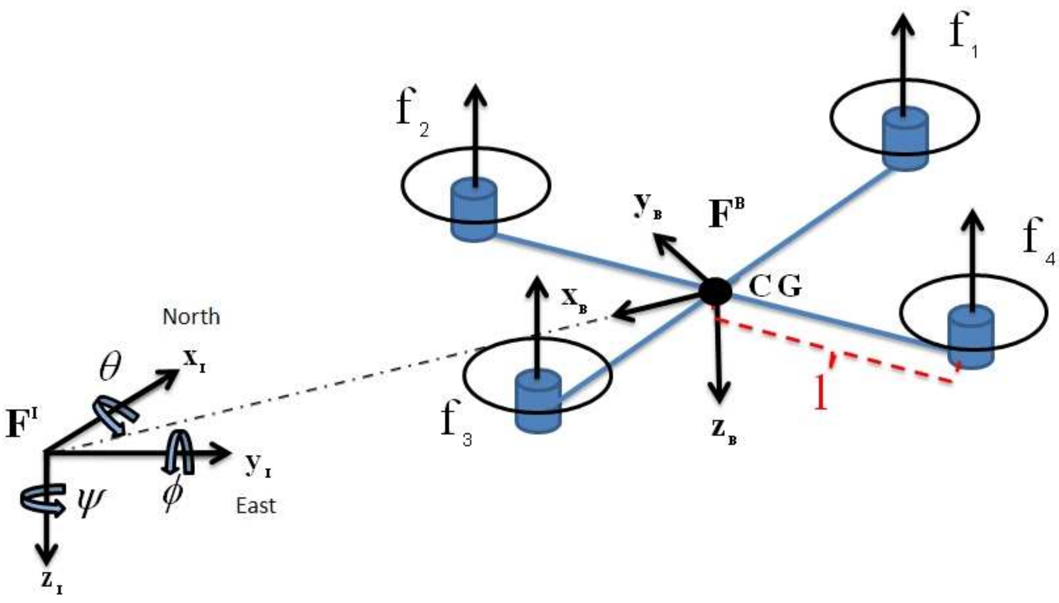


Figure 2.12: Inertia and body coordinate system

The rigid body dynamic equation subjected to acting forces and torques in the body-fixed frame and in Newton-Euler scheme is:

$$\begin{bmatrix} mI_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & J \end{bmatrix} \begin{pmatrix} \dot{v}^b \\ \dot{\omega}^b \end{pmatrix} + \begin{bmatrix} \omega^b \times (mv^b) \\ \omega^b \times (J\omega^b) \end{bmatrix} = \begin{pmatrix} F^b \\ \tau^b \end{pmatrix}, \tag{2.1}$$

where m[kg] is the mass of platform, I is an identity matrix, J is the moment of inertia matrix for the quadrotor platform. The platform has a symmetric body structure and most of its mass is located around the CG, hence we can assume the inertia matrix has a diagonal form, $J = diag(J_{xx}, J_{yy}, J_{zz})$. F^b and τ^b are total force and torque applied to the aircraft body respectively. The body velocity is projected to its inertial reference frame through the rotational matrix, $\dot{\xi} = Rv^b$, given as

$$R = \begin{pmatrix} c_\theta c_\psi & c_\psi s_\theta s_\phi - s_\psi c_\phi & c_\psi s_\theta s_\phi + s_\psi c_\phi \\ s_\psi c_\theta & s_\psi s_\theta s_\phi + c_\psi c_\phi & s_\psi s_\theta c_\phi - c_\psi s_\phi \\ -s_\psi & c_\theta s_\phi & c_\theta c_\phi \end{pmatrix}, \tag{2.2}$$

where $c_k = \cos(k)$, $s_k = \sin(k)$. The transformation matrix from $\dot{\eta}$ to ω^b also is $\dot{\eta} = \Gamma\omega^b$, where

$$\Gamma = \begin{pmatrix} 1 & \sin(\phi)\tan(\theta) & \cos(\phi)\tan(\theta) \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \frac{\sin(\phi)}{\cos(\theta)} & \frac{\cos(\phi)}{\cos(\theta)} \end{pmatrix}. \tag{2.3}$$

The quadrotor exhibits an under-actuated dynamic system where there are six DOF but only four control inputs. The main acting forces on the quadrotor are its weight, rotor thrusts and body drag force [39]. The motor thrust f_i is proportional to square of the propeller's angular velocity Ω_i and acts along the e^z as define in Fig. 2.12. The rotor rotational speed is controlled through the input voltage v_i.

$$f_i = K_\Omega \Omega_i^2 \tag{2.4}$$

$$\Omega_i = \frac{K_v}{1 + \tau_f s} v_i \tag{2.5}$$

where K_Ω and K_v are the constant coefficients, and τ_f is the response delay of the rotor thrust.

As the platform moves through the air, it produces drag, however since our platform's motion as a small scaled VTOL aircraft is slow, the generated drag is quite small. The body drag force can be written as

$$D = -\frac{1}{2}\rho C_d \left| v^b \right| v^b, \tag{2.6}$$

where ρ is the density of air and C_d is the drag coefficient.

Gravity also exerts a force on the quadrotor. This force acts on the Center of Gravity (CG) and in inertial frame and is given by:

$$f_g^I = \begin{pmatrix} 0 & 0 & mg \end{pmatrix}^T. \tag{2.7}$$

However, since we are defining all the acting forces in the body frame, the gravity force must be transferred to the body frame. The gravity force in the body frame is then obtained following [41].

$$f_g^b = R^T f_g^I = \begin{pmatrix} -mg\sin(\theta) \\ mg\cos(\theta)\sin(\phi) \\ mg\cos(\theta)\cos(\phi) \end{pmatrix} \tag{2.8}$$

As a result, the total acting force in body frame is obtained as follows.

$$F^b = -\frac{1}{2}\rho C_d \left| v^b \right| v^b + f_g^b - \sum_{i=1}^{4} f_i e^z \tag{2.9}$$

The roll, pitch and yaw $(\tau_\phi, \tau_\theta, \tau_\psi)$ torques acting on the body frame are mainly produced by the rotors' differential thrust. Changing the motors' thrust results in roll and pitch torques. Yaw torque is mainly generated by differential thrust between the two pairs of counter-rotating motors. The yaw moment produced by the rotors' reactive torque Q is given as,

$$Q_i = d\Omega_i, \tag{2.10}$$

where d is a rotor induced torque drag coefficient which mainly depends on the propeller's specification. Thus, the acting torques on body fixed frame are as shown.

$$\begin{aligned}\tau_\phi &= \tfrac{\sqrt{2}}{2} l[(f_2 + f_3) - (f_1 + f_4)] \\ \tau_\theta &= \tfrac{\sqrt{2}}{2} l[(f_1 + f_2) - (f_3 + f_4)] \\ \tau_\psi &= Q_1 - Q_2 + Q_3 - Q_4\end{aligned} \tag{2.11}$$

The quadrotor complete dynamic model can be derived by substituting equations (2.9), (2.11) into (2.1) and transferring the body states to inertial frame with the transformation matrices (2.2) and (2.3). The next step is to determine the unknown model parameters and conduct identification experiments to obtain the numerical representation of this model.

2.4.3 Model Parameter Identification

In this section, we present the model parameter identification procedure. Based on the model structure which was derived in the previous section, the unknown parameters are determined. However, not all parameters are necessary to be identified. For instance, the body drag force due to the low velocity of the platform does not significantly influence the quadrotor's dynamic and thus we will not consider this term. The unknown parameters are identified through static experiments and recorded flight data. The ground experiments are mainly suited for identifying the parameters that can be measured in static condition such as platform weight, center of gravity position, moment of inertia and platform dimensions.

For those parameters that should be identified based on the flight data analysis, we require data logging hardware which was designed and implemented in real flight tests. The developed avionic system is able to record the RC receiver inputs, IMU and GPS outputs which respectively contain joystick signal, quadrotor attitude and position data. Additionally, only for model identification purposes, a PWM measurement board is added to system to record the gyro output (motors inputs).

2.4.4 Static Tests

Moment of Inertial Measurement

The platform moment of inertia can also be measured through the static experiments. We follow the same method used in [42] which is theoretically based on the Trifilar Pendulum theory.

Using the Trifilar Pendulum theory [43], by the setup of a simple experiment, the moment of inertia of a compound object such as the quadrotor with onboard systems can be measured.

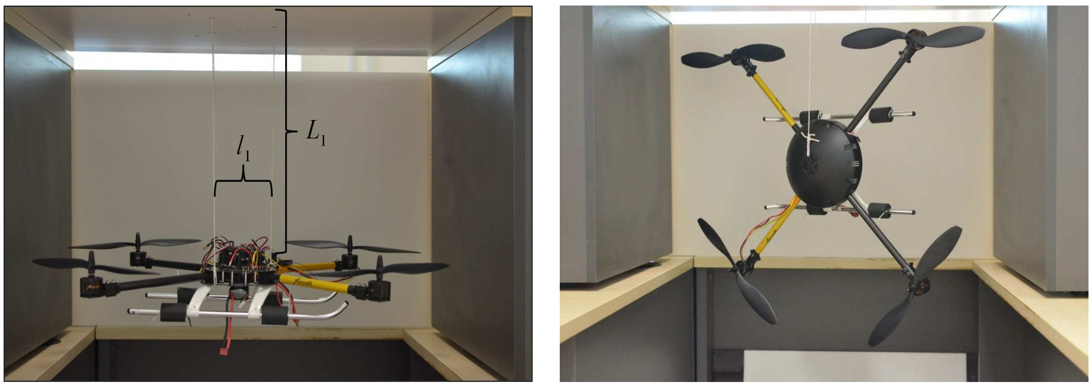


Figure 2.13: Left: J_{zz} measurement, Right : J_{xx},J_{yy} measurement.

Since quadrotors are structurally symmetric, only the moment about each axis is required to be identified. Also, the moment of inertial in the x and y direction is expected to almost similar due to the symmetrical setup.

The platform in this case is suspended by three lines with equal length to create a torsional pendulum as shown in Fig. 2.13. The suspended quadrotor is excited minutely in the axis of interest. The period of oscillation is recorded and using Eqn. 2.12, the rolling, pitching and yawing moment of inertia could all be calculated [42].

$$J_{xx,yy,zz} = \frac{mgl_1l_2l_3t^2}{4\pi^2L}\frac{l_1\sin\alpha_1 + l_2\sin\alpha_2 + l_3\sin\alpha_3}{l_2l_3\sin\alpha_1 + l_1l_2\sin\alpha_2 + l_1l_2\sin\alpha_3} \tag{2.12}$$

where α_i is the angle between the three strings which for our setup all are $120°$. l_i is distance between the strings and L is length of them. t is the platform oscillation period around the axis of measured moment of inertia.

In order to obtain the period, first the platform should be suspended from three points around the desired axis and then excited gently. The platform oscillation is recorded by a camera for several periods. Each perturbation experiment is done three times to improved accuracy and reliability of the results. Captured video is analyzed in a computer and the average of periods for all experiments is calculated. This procedure is repeated to obtain the moment of inertia about each axis. All axes measured moment of inertia are given in Table 2.2.

Center of Gravity Location

Finding and adjusting the center of gravity is a very important step to have in order to obtain a stable platform. To find the CG location, in at least three experiments, the platform should be suspended from one arbitrary point. For simplicity, we attached the string to the tip of quadrotor rotor arm. For each plane $(x-y, x-z, y-z)$, the platform is suspended from an appropriate point such that the desired plane faces the camera in which a high resolution picture is captured. In the computer, a straight line connects the attached points. The intersection of these lines gives the location of center of gravity. For our quadrotor the CG obtained is very close to the center of platform which verifies the correct placement of our avionics system.

Rotor Thrust Measurement

The rotor thrust measurement stand consist of a force meter, tachometer for measuring the propeller rotational speed, servo control board which drives the motors and, current and voltage monitoring systems. Fig. 2.14 shows the assembled setup.

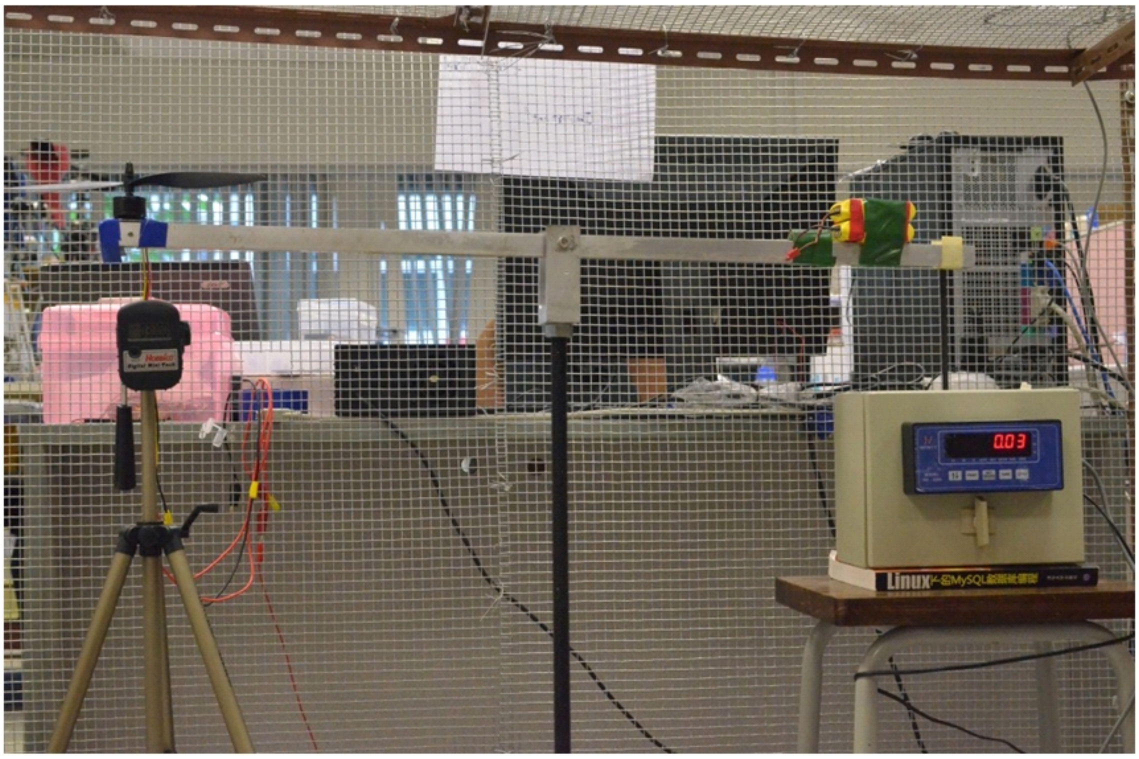


Figure 2.14: Thrust measurement experiment.

Before each experiment, the force measurement unit and tachometer are carefully adjusted and calibrated. This experiment is mainly used for modeling the rotors force dynamics. The current and voltage data will be used to estimate total power consumption as well.

Since quadrotors mainly operates in hovering condition and at low velocities, it would be reasonable to observe the rotor dynamic through a ground experiment. In fact, it is assumed that the UAV will fly in low wind conditions and not very aggressively, therefore the rotor-propeller's indoor behavior will replicate a real outdoor flight. In order to identify the rotor dynamic, a thrust force experimental setup was designed. For a full range of input from minimum to full throttle, rotor thrust, propeller angular velocity, rotor current and applied voltage are measured. In order to improve the accuracy of the result, this experiment was repeated at least three times. Fig. 2.15 shows the experiment's collected data.

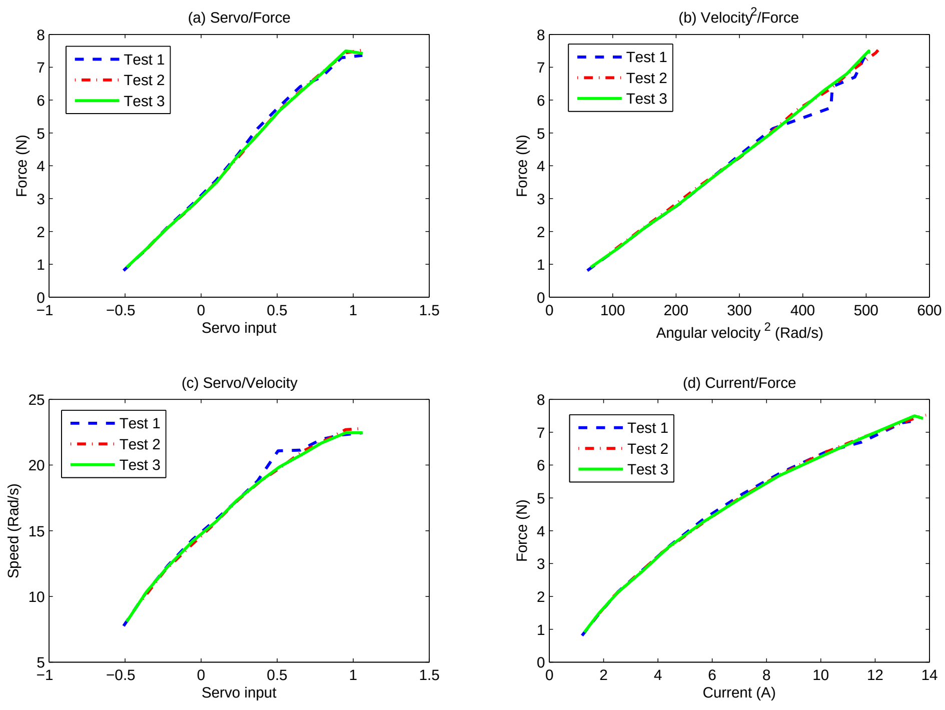


Figure 2.15: Rotor-Propeller thrust experiment's collected data. (a) Servo input to rotor thrust, (b) Square of propeller angular velocity to rotor thrust, (c) Servo input to propeller angular velocity, (d) Consumption current to rotor thrust.

Fig. 2.15(b) shows that Ω^2 is proportional to the thrust. However, from the Fig. 2.15(c) it can be observed that the propeller angular velocity is not linearly proportional to the input voltage. This phenomena seems to be due to the rotor speed controller characteristic. The electronic speed controller (ESC) controls the rotor rotational speed to regulate the output power.

Hence, as Fig. 2.15(a) illustrates, the rotor thrust is approximately linearly proportional to the servo input. From the control aspect, only this relationship is necessary. As a result the Eqn. 2.13 is valid for our setup.

$$f_i = \frac{K_f}{1 + \tau_f s} v_i, \tag{2.13}$$

where K_f is the constant thrust lift coefficient.

To obtain the force response time delay, a square wave is applied to the motor input and the corresponding thrust response is observed by force measurement unit. Meanwhile, voltage of the battery as a power source is monitored by a oscilloscope. Since, the force measurement unit only shows the steady thrust, the time delay is estimated from battery voltage drop, which depends on the rotor current and correspondingly output thrust.

The rotor dynamic coefficients are obtained based on a least square curve fitting. Fig. 2.16 to Fig. 2.18 shows the curve fitting results.

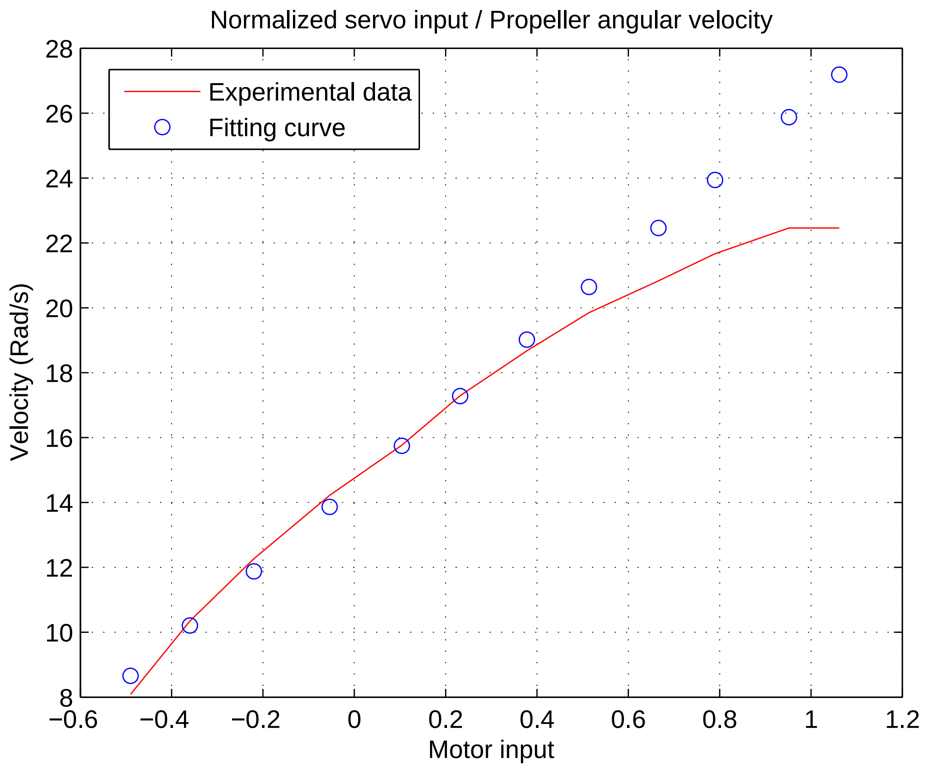


Figure 2.16: Rotor thrust to angular velocity coefficient (K_Ω) result.

All the parameters that are identified by the static experiments are listed in Table 2.2.

2.4.5 Flight Experiments

In this part, we plan to use experiment data obtained from flight tests to identify the remaining unknown model parameters. The obtained parameters from the static experiments can also be tuned or validated through this method. In this approach the pilot will be asked to excite a requested dynamic of the platform in a certain way. The avionic board through the INS system will observe the response of the UAV and record the sensor data on the onboard memory card. We chose a time domain model identification method and will use MATLAB IDENT toolbox

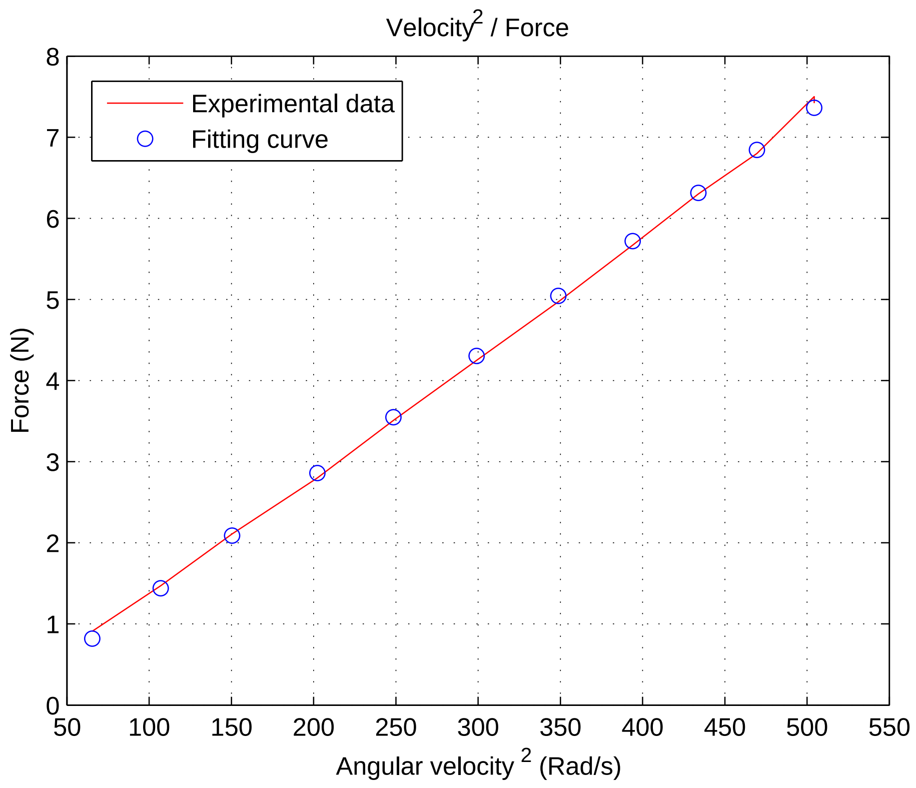


Figure 2.17: Rotor rotational velocity coefficient (K_v) result.

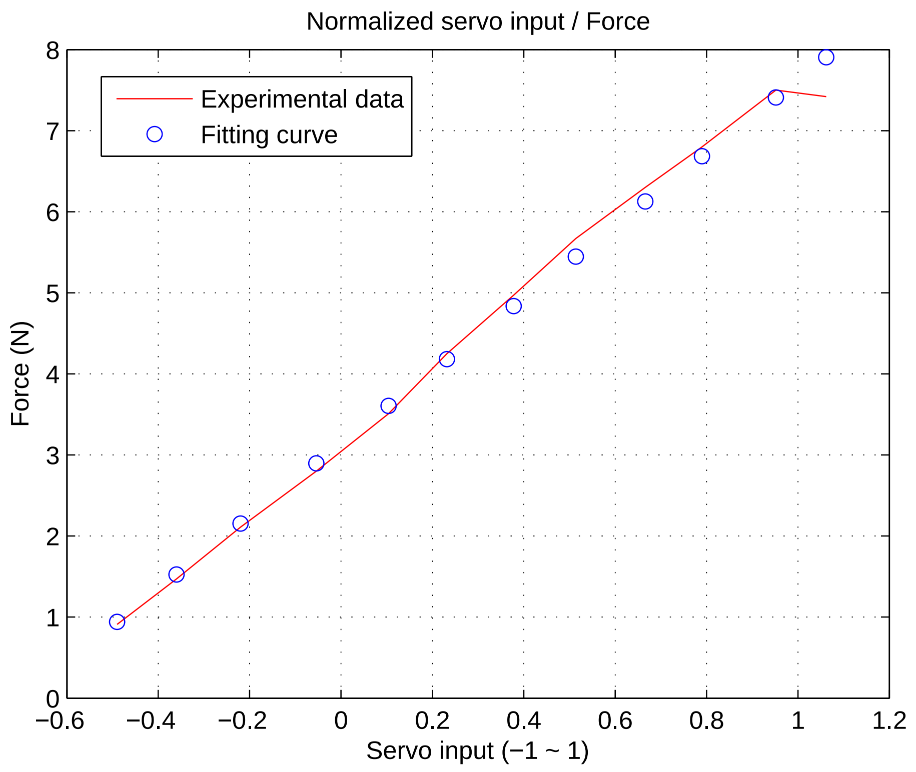


Figure 2.18: Rotor thrust lift coefficient (K_f) result.

Table 2.2: Numerical value of identified parameters from ground experiment.

Description	**Parameter**	**Value / Units**
Rolling, pitching, yawing moment of inertia	J_{xx},J_{yy},J_{zz}	0.03356, 0.03122, 0.05423 $kg.m^2$
Rotor thrust to rotational velocity coefficient	K_Ω	0.059
Rotor rotational velocity coefficient	K_v	9.2994
Rotor thrust lift coefficient	K_f	4.4861
Rotor thrust response time delay	τ_f	0.06 s
Rotor to CG distance	l	0.325 m
Total platform mass	m	1.37 kg

to analyze the collected data. We describe the implementation procedure of this approach in the following steps.

Flight Data Collection

It is clear that the dynamic input-output data is necessary. The model states and their measurability status are given in Table 2.3. Recording RC receiver outputs $U_{rc} = (U_a, U_e, U_{th}, U_r)$ is useful when we have the gyro in the control loop or as a input source for gyro model identification where more details will be covered in Section 2.5. For non-measurable variables highlighted in Table 2.3, it is possible to observe them based on the mathematical relationship expressed in Eqn. 2.1 and Eqn. 2.2. In fact, we have all the dynamic states and inputs data for model identification.

Table 2.3: Dynamic states and inputs description and measurability status.

Variable	**Physical expression**	**Units**	**Direct measurability**
x,y,z	Position in inertial frame x,y,z axis	m	Yes
u,v,w	Velocity in body frame x,y,z axis	m/s	No
ϕ,θ,ψ	Roll, Pitch, Yaw angle	rad	Yes
p,q,r	Roll, Pitch, Yaw angular rate	rad/s	Yes
acx_b,acy_b,acz_b	Body acceleration in body frame x,y,z axis	m/s^2	Yes
M$_i$	Normalized rotor$_i$ input $(-1 \sim 1)$	NA	Yes
U$_a$	Normalized aileron servo input $(-1 \sim 1)$	NA	Yes
U$_e$	Normalized elevator servo input $(-1 \sim 1)$	NA	Yes
U$_{th}$	Normalized throttle servo input $(-1 \sim 1)$	NA	Yes
U$_r$	Normalized rudder servo input $(-1 \sim 1)$	NA	Yes

Model Identification Input Signal

Since Chirp signals contains a variety of frequency components, it is an appropriate input signal for model identification or validation purposes [42]. A typical linear Chirp signal is mathematically expressed as Eqn. 2.14 [44].

$$u_{chirp}(t) = A_{chirp} \sin\left[2\pi(f_0 + \frac{k_{chirp}}{2}t)t\right], \tag{2.14}$$

where A_{chirp} is amplitude, f_0 initial frequency, and k_{chirp} is the rate of frequency increase.

In order to appropriately excite the desired dynamic, selecting the Chirp frequency range is a crucial step. Furthermore, since this signal is applied manually by the pilot, preparing some simulation based flight tests for training purpose is highly recommended. The initial frequency is preferred to be as low as possible, however it depends on pilot's sight and maneuverability. The maximum frequency should be chosen to avoid exciting undesired dynamic nonlinearities, mechanical vibration and actuators' rate constrains.

The Chirp ratio is preferred to be linearly increasing. In each flight experiment, the pilot initially apply two sinusoid long period inputs to the desire channel [44]. Basically, this is to ensure that the low frequency is completely perturbed [42]. Thereafter, the input frequency should be increased smoothly to the desired maximum frequency. It is worth mentioning that the input amplitude does not necessarily have to be constant, typically keep the variation in the range of $\pm 10 - 20\%$ is acceptable [38].

During the flight test, the quadrotor is expected to exhibit abnormal behavior due to the chirp input and its nonlinear dynamic. Therefore, it would be better to do this experiment in a wide and open area. Since, we mainly plan to control the quadrotor not very aggressively, the UAV should be perturbed around hovering. Hence, firstly the quadrotor hovers at a appropriate point with a good eyesight, then the pilot excites only one desired channel as a chirp signal style. Meanwhile the onboard system records all the desired data. For each channel the perturbation is repeated several times to ensure enough qualified data is recorded.

Between each perturbation, the pilot is requested to keep the quadrotor in a hovering flight for several seconds. This part later would help to easily identify each set of data, and also verify the trim values. Furthermore, because of dynamic coupling and disturbances, when one channel is perturbed, the platform may drift in different directions or orientations. Yet, the pilot is asked to not touch the other channels to correct the quadrotor, unless in emergency situations. This method is quite important, specifically when we want to decoupled the dynamic model and identified them separately.

Quadrotor Hovering Trim Value

For any further analysis on the flight data, trim values of states and inputs are necessary. In fact, the quadrotor is modeled around the trim values which physically is the signal's value in a good hovering condition. We asked the pilot to keep the quadrotor in a hovering mode and adjust the joystick trim inputs to achieve a good hovering flight with minimum input correction.

Meanwhile, the onboard system records all the measurable signals. After this experiment, the data are analyzed and a period of best hovering flight is extracted. Averaging this part's data, gives the trim values. The results are presented in Table 2.4. Regarding to the inputs' trim $(u_{a0}, u_{e0}, u_{th0}, u_{r0})$, even though mathematically it is expected to have zero inputs for hovering, due to the imbalance rotor thrust and payload, and minor nonsymmetric mechanical structure, small inputs' bias are almost always necessary. From the control point of view, these trim values correspond to the zero controller output.

In the model identification process, the required parameters will be identified around these trim values.

Table 2.4: States and inputs trim value in hover condition.

Variable	**Trim value in hover condition**	**Units**
u_{a0}	0.0660	NA
u_{e0}	0.0660	NA
u_{th0}	0.1800	NA
u_{r0}	-0.0130	NA
u_0, v_0, w_0	$-0.0739, -0.1759, -0.5610$	$\mathrm{m/s}$
ϕ_0, θ_0, ψ_0	$-0.0381, 0.0035, -0.0288$	rad
p_0, q_0, r_0	$0.00092, 0.00095, -0.0023$	$\mathrm{rad/s}$
$acx_{b0}, acy_{b0}, acz_{b0}$	$-0.0023, -0.0024, -0.1139$	$\mathrm{m/s^2}$

Parameter identification

After preprocessing the raw data, we are ready to identify the desired unknown parameters. We use the MATLAB IDENT toolbox [45] software and prediction error method (PEM) as an identification algorithm.

The rotor thrust dynamic can be identified based on the vertical body acceleration. The total rotor thrust can be obtained from Eqn. 2.15. In fact, if body z axis acceleration is perturbed in a zero attitude condition, we can assume all the rotor thrust are approximately equal and throttle input is equivalently distributed to the rotors' input. Hence, individual rotor thrust can be observable.

$$\sum_{i}^{4} f_i = m\left(-\dot{w} + qu - pv + g\cos\theta\cos\phi\right) \tag{2.15}$$

After data de-trending based on the signal trim values, the input-output data source is ready for identification processing.

The data is then fed into the Matlab IDENT toolbox with 20 ms sampling rate. The model structure is assigned as Eqn. 2.13 and initial guess values are chosen as ground static experiment results. The other set of throttle perturbation is used for model validation. The toolbox estimates the model based on the given structure and through PEM method. Fig. 2.19 shows a comparison between the identified rotor thrust model from static ground experiment and that based on flight data. As the figure illustrates, the two estimated models are quite identical and can closely track the observed output force.

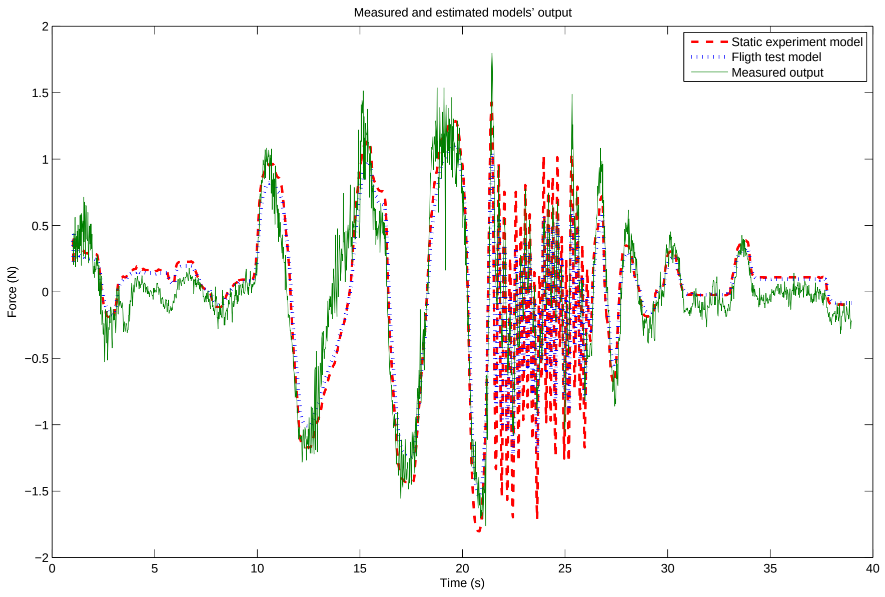


Figure 2.19: Rotor thrust dynamic model comparison.

The induced torque coefficient d, can be estimated based on yaw dynamic. From Eqn. 2.10 and Eqn. 2.11, the yaw ratio dynamic with respect to motors voltage input can be written as:

$$\begin{cases} \dot{r} = \frac{J_{xx}-J_{yy}}{J_{zz}} pq + \frac{d}{J_{zz}} \Omega \\ \Omega(s) = \frac{K_v}{1+\tau_f s} \sum_{i=1}^{4} (-1)^{i+1} v_i \end{cases} \tag{2.16}$$

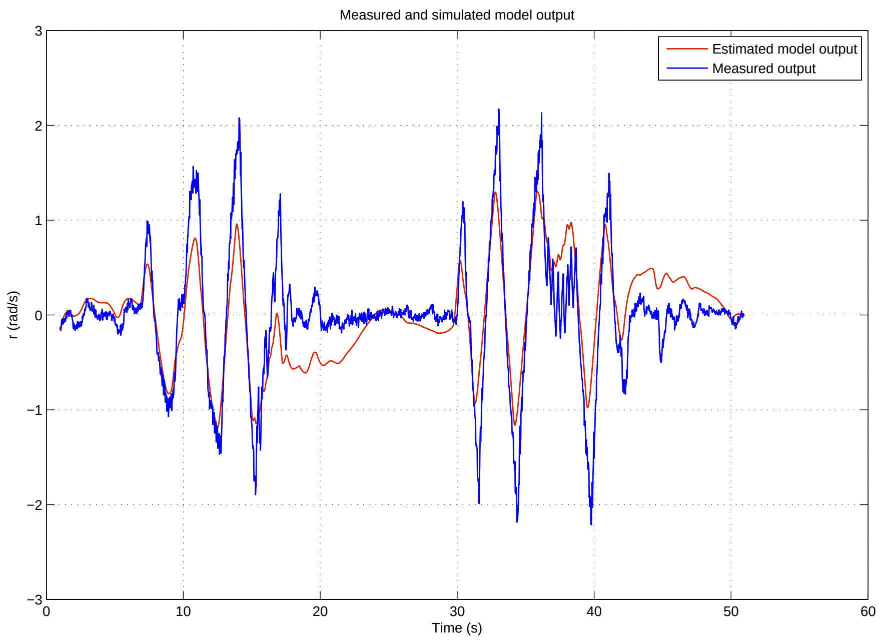


Figure 2.20: Yaw ratio dynamic estimated and measured outputs.

In a similar manner, a flight experiment is conducted to collect the necessary data to identify the yaw dynamic parameter. The rudder channel is excited in a Chirp style, and the sensors data are recorded correspondingly. Due to the small attitude dynamic coupling, and unavoidable pilot correction applied on the other channels, we observed small changes on the pitch and roll outputs. Based on the model structure from Eqn. 2.16 and collected data, MATLAB identified the induced torque coefficient d. The estimated model output is compared with real measurements and the result are given in Fig. 2.20.

The identified parameters based on flight tests are presented in Table 2.5.

Table 2.5: Numerical value of identified parameters from flight data.

Parameter	**Value**	**Description**
K_f	3.8738	Rotor thrust lift coefficient
τ_f	0.095583	Rotor thrust response time delay
d	0.013	Induced torque coefficient

2.5 Quadrotor Model with Gyro-in-the-Loop

One way to control the quadrotor is to use the gyro in both manual and autopilot flight modes. In this approach, the autopilot system acts as a pilot and apply the control outputs to the gyro instead of directly controlling the motors. As Fig. 2.21 illustrates, the gyro gets the control signals from the autopilot and accordingly distributes them to the motors. The gyro itself is also able to control the quadrotor attitude angles or angular ratios. In fact adding the gyro in the automatic flight control loop makes the overall controller simpler, since the gyro can help to stabilize the attitude. However, it requires additional dynamic identification to model the close loop attitude dynamic with having the gyro as a attitude stabilizer.

Furthermore, from a safety aspect, this structure can be helpful specifically during preliminary autopilot tests. In the rare case when the autopilot system goes wrong and the pilot is forced to switch back to manual mode, the rotor inputs do not experience this change, since the gyros have already been driving the motors.

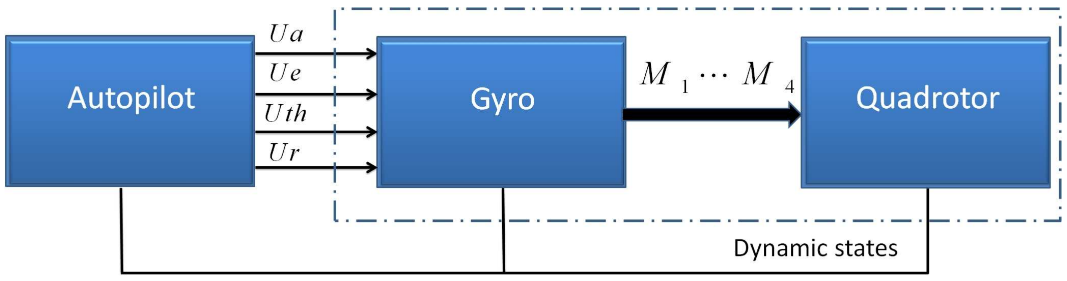


Figure 2.21: Gyro in the loop control structure.

The gyro used in our system has two working configurations: Hover mode and Cruise mode. In the hover mode, the gyro stabilizes quadrotor attitude based on the control inputs. The aircraft with gyro in hover mode, is inherently quite stable. As hover mode mainly is for hovering flight, the gyro in cruise mode is more suitable for cruise flight. Based on our tests, in the cruise mode, the gyro is controlling attitude angular ratio and the quadrotor behaves more aggressively. We have conducted several flight tests to observe the gyro characteristic and model the closed loop attitude dynamics. In the following sections we will identify the model of quadrotor attitude dynamic when the gyro is in the control loop. Since, the gyro to some extent controls the attitude states, we will assume the attitude axes dynamic are decoupled. Moreover, the motors input are linearly proportional to throttle channel, thus the gyro does not affect the rotor thrust dynamic.

2.5.1 Gyro in Hover Mode

We set the gyro in hover mode and asked the pilot to perturbed all the channels around the hovering condition. The perturbation style is the same as that stated in Section 2.4.5. The input joystick signal, together with attitude angles and their ratio are analyzed. It is observed that the gyro in this mode, stabilizes angles of pitch and roll axes and angular ratio of heading. In order to have an initial guess for the model order, it can be argued that, each attitude dynamic naturally is a double integrator and the gyro have a PID controller inside, hence the closed loop system should behave as a third order system with two stable zeros. However, during model identification analysis, it is found that a simple second order system can reasonably match the output signals. Eqn. 2.17 to Eqn. 2.19 shows the attitude dynamic model structure and Fig. 2.22 to Fig. 2.24 illustrates the accuracy of identified models.

$$\phi(s) = \frac{K_\phi}{(1+\tau_{\phi 1}s)\,(1+\tau_{\phi 1}s)} e^{-T_\phi} u_a \tag{2.17}$$

$$\theta(s) = \frac{K_\theta}{(1+\tau_{\theta 1}s)\,(1+\tau_{\theta 1}s)} e^{-T_\theta} u_e \tag{2.18}$$

$$r(s) = \frac{K_r}{(1+\tau_r s)} u_r \tag{2.19}$$

In this mode, since the gyro directly controls the pitch and roll angles, these attitude dynamics has a considerable delay which has to be taken into account. It worth to note that, it is possible to increase the model order to obtain a perfect output matching, but it will increase the complexity of controller and correspondingly increase computation cost. From the control point of view, since the closed loop is stable, for roll and pitch dynamics the gyro can take the responsibility of inner loop tracking control and for the heading, we can consider the yaw angle dynamic as a simple linear second order system with input u_r from Eqn. 2.19.

2.5.2 Gyro in Cruise Mode

In this mode, the gyro stabilizes the attitude angular ratio and hence the platform behavior is not as smooth as the gyro in hover mode. In a similar flight experiment manner, we have done several flight tests for each individual channel identification. From the collected flight data, it can be observed that, a first order system can quite perfectly emulate each attitude dynamic. Hence, the attitude angular ratio structures are estimated as presented in Eqn. 2.20 to Eqn. 2.22.

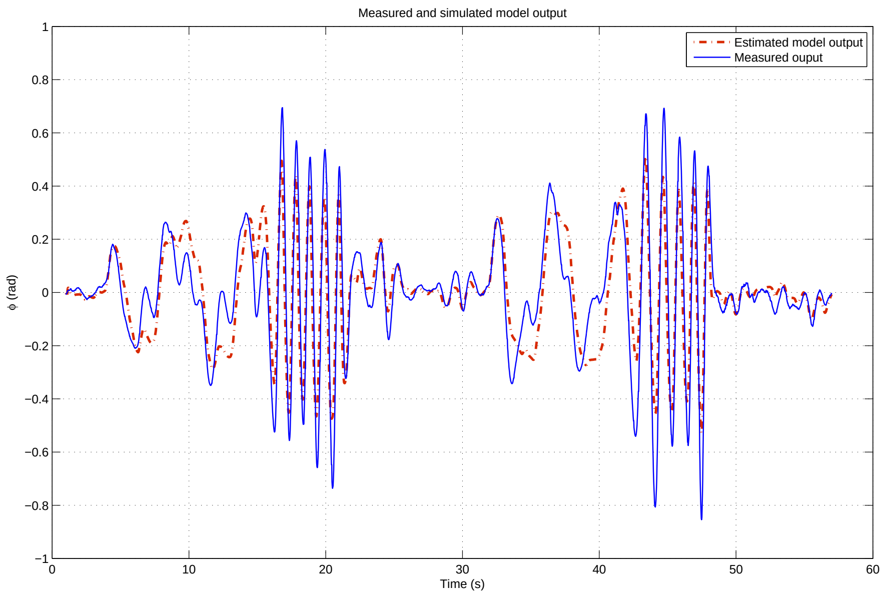


Figure 2.22: Gyro in hover mode, rolling identified model.

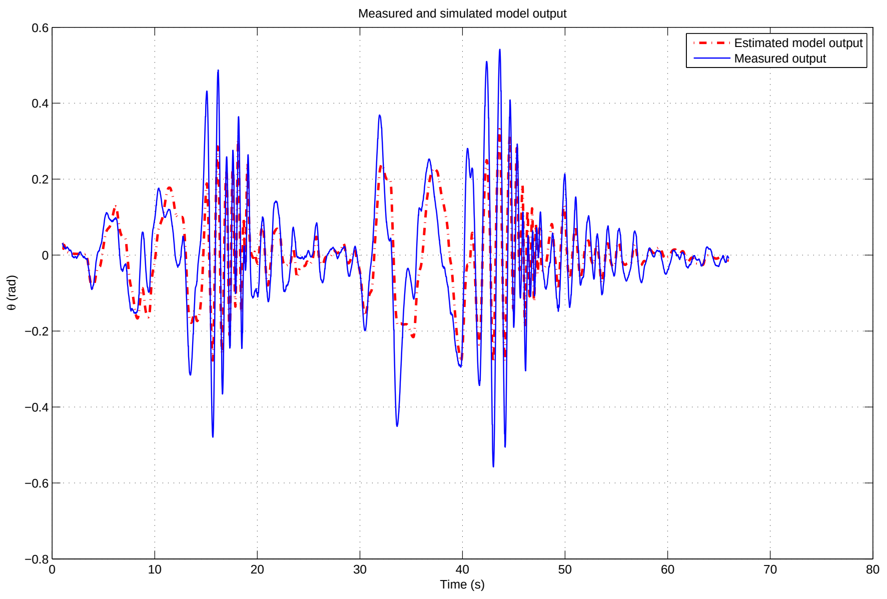


Figure 2.23: Gyro in hover mode, pitching identified model.

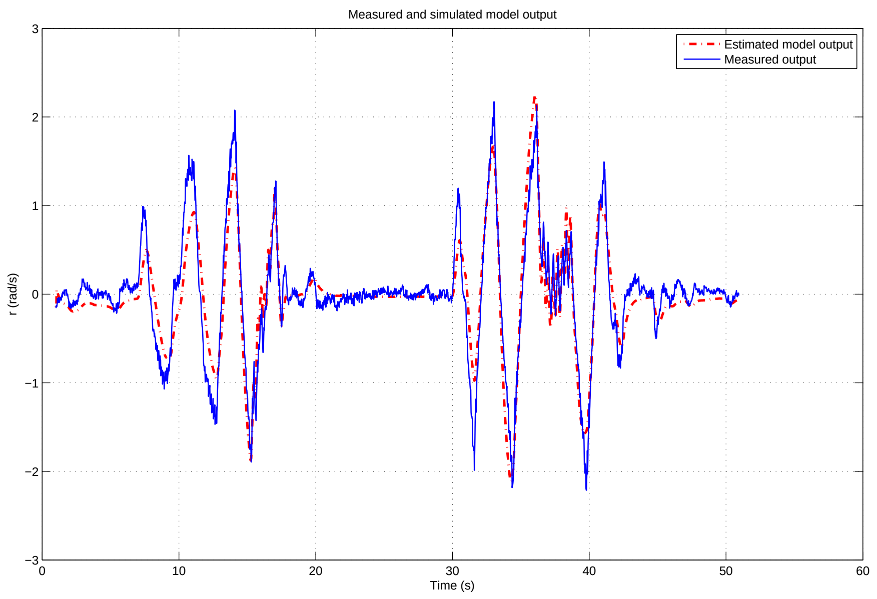


Figure 2.24: Gyro in hover mode, yaw angle ratio identified model.

The heading dynamic basically is the same as the gyro in hover mode. With such a model structure the unknown parameters are also identified. Fig. 2.25 to Fig. 2.27 show the estimated model fidelity.

$$p(s) = \frac{K_p}{(1+\tau_p s)} u_a \tag{2.20}$$

$$q(s) = \frac{K_q}{(1+\tau_q s)} u_e \tag{2.21}$$

$$r(s) = \frac{K_r}{(1+\tau_r s)} u_r \tag{2.22}$$

The numerical value of identified model parameters for both gyro operation modes are listed in Table 2.6.

2.5.3 Model Validation

In this section we aim to verify the fidelity of the identified dynamic model. To do so, we have used the collected data from previous flight tests. For perturbation of each input channel, the measured body axes acceleration and angular rate ratios are compared with simulated model outputs. Since for further control purpose, we plan to mainly use the gyro (cruise mode) in the loop configuration, the model validation result of this configuration is only presented here.

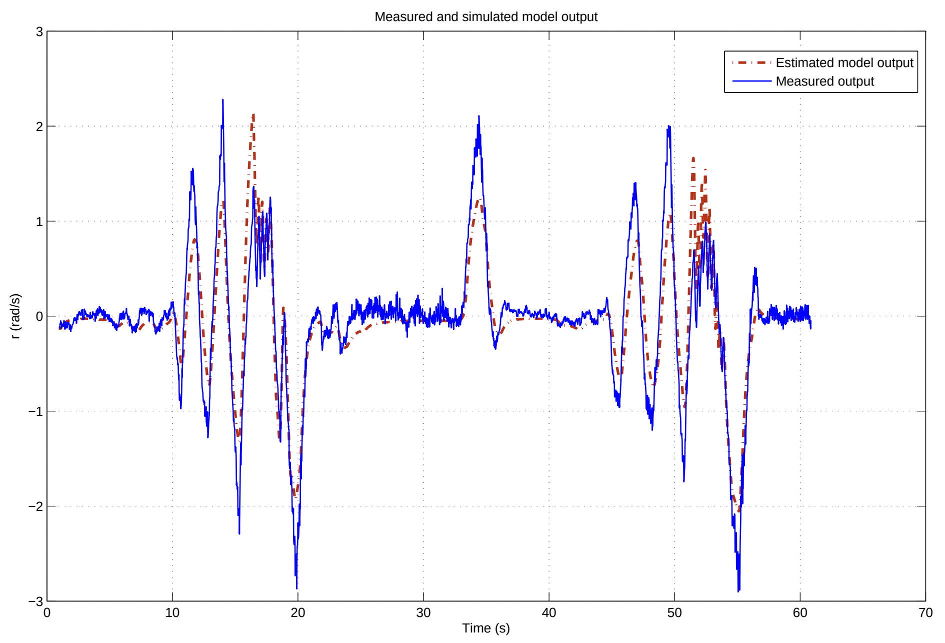


Figure 2.25: Gyro in cruise mode, roll angle ratio identified model.

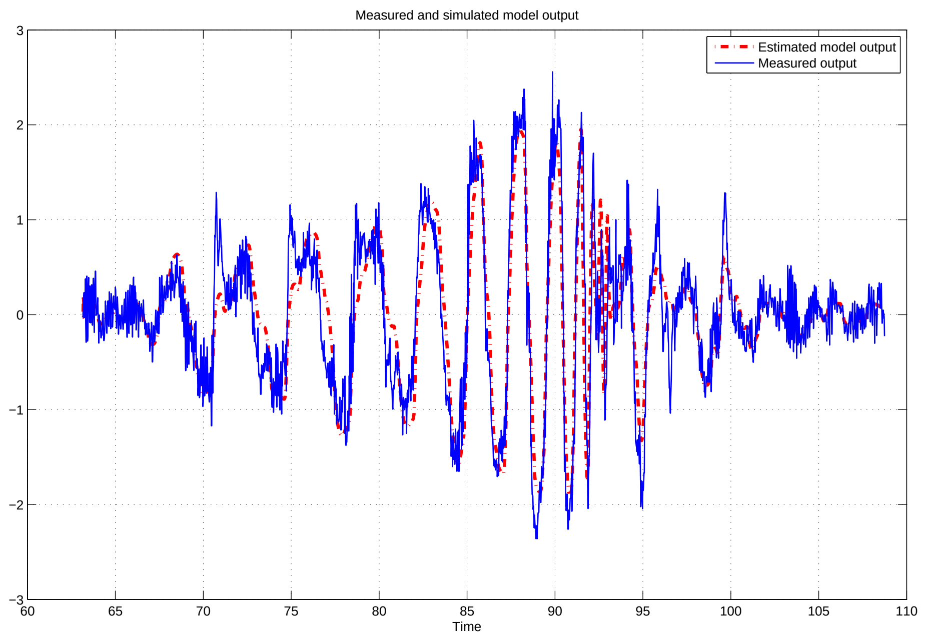


Figure 2.26: Gyro in cruise mode, pitch angle ratio identified model.

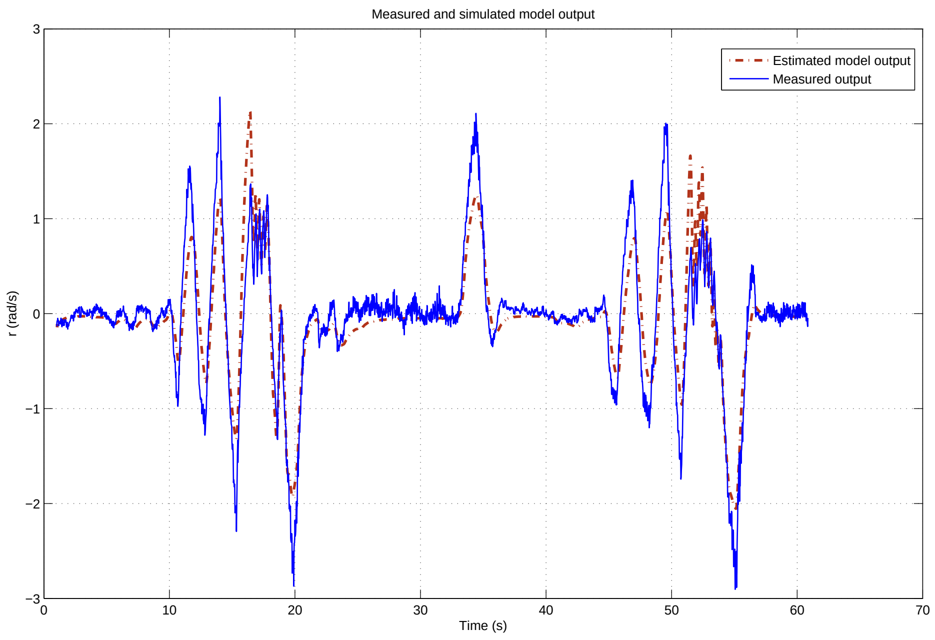


Figure 2.27: Gyro in cruise mode, yaw angle ratio identified model.

Table 2.6: Quadrotor with gyro in the loop attitude dynamic identified parameters.

Parameter	**Value**	**Parameter**	**Value**
K_ϕ	-0.8628	$\tau_{\theta 1}$	0.079119
K_θ	-0.59151	$\tau_{\theta 2}$	0.079119
K_r	-4.8121	τ_p	0.033855
K_p	-2.3167	τ_q	0.046672
K_q	-2.4967	τ_r	0.37029
$\tau_{\phi 1}$	0.076615	T_ϕ	0.1 s
$\tau_{\phi 2}$	0.076615	T_θ	0.1 s

Fig. 2.28 to Fig. 2.31 illustrates this comparison. As it can be observed from the figures, the identified quadrotor model is able to track the measured states with acceptable accuracy. We will use this model for further control design and simulation analysis.

The details for modeling a cross style quadrotor has been described through this modeling section. Through the modeling and physical experiments conducted, we are able to obtain all the parameters for the cross style quadrotor. Both simulations and real flight tests have been successfully conducted to prove the performance and model of the quadrotor. A cross style quadrotor boasts great maneuverability to execute aggressive flight patterns and enable the unobstructed placement of sensors such as cameras or scanning laser rangefinder.

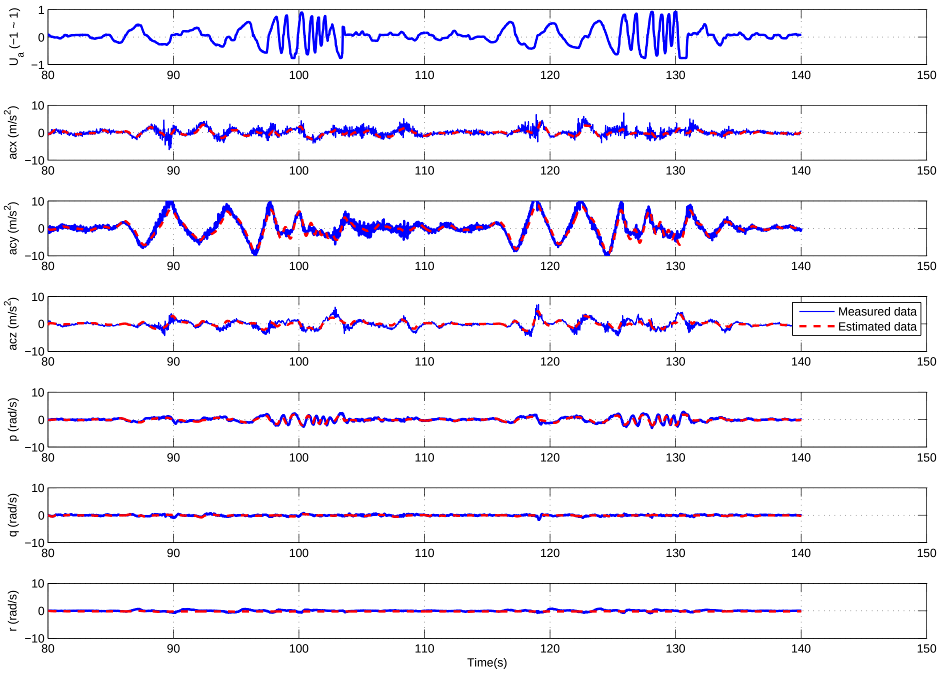


Figure 2.28: Model verification using aileron perturbation.

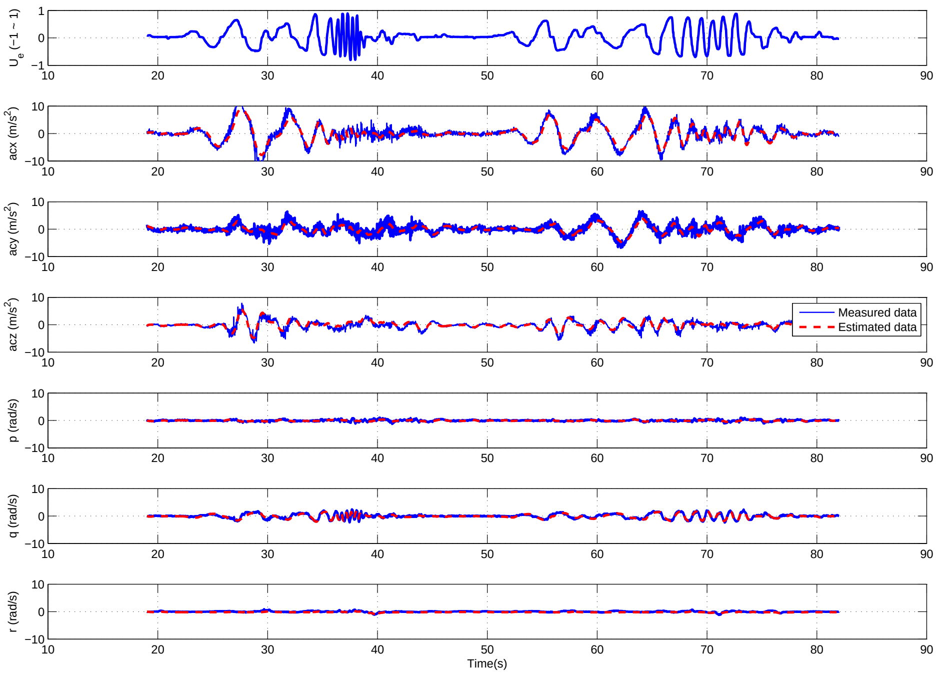


Figure 2.29: Model verification using elevator perturbation.

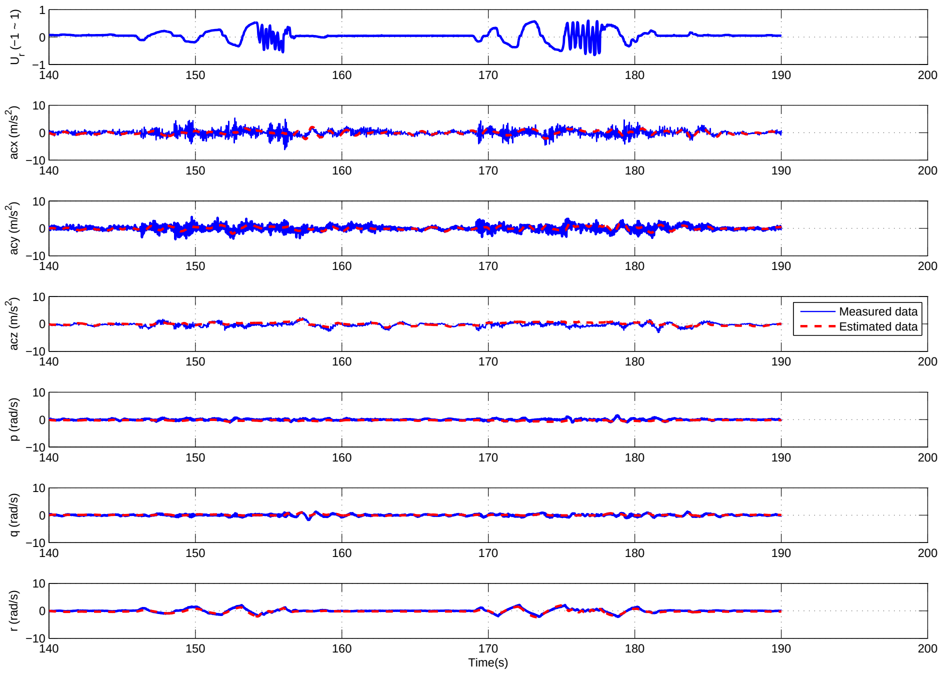


Figure 2.30: Model verification using rudder perturbation.

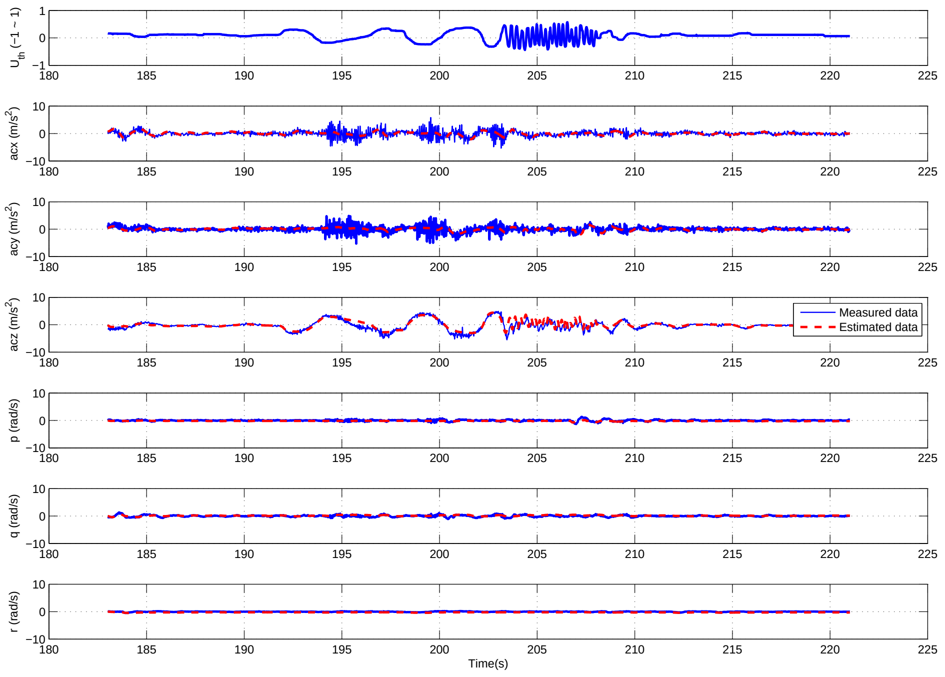


Figure 2.31: Model verification using throttle perturbation.

2.6 Development of an Unconventional UAV

Development of unconventional UAVs have made major progress due to the huge improvements in research areas such as UAV flight control theories, MEMS, electronic devices and material science. Unconventional UAVs have great potential applications in military and civilian operations, especially where there are severe constraints in their operating environment. In tasks such as outdoor surveillance, the ability to perform long endurance or long distance missions usually requires the UAV to perform cruise flight. Efficient cruise flight is usually achieved with UAVs having airfoils that have good lift to drag ratios. However, there are cases where there are insufficient take-off space available for standard airplane-type UAVs to be launched and recovered. On the other hand, the capability of VTOL and hovering is beneficial to most surveillance missions as it allows for a close up static surveillance view of the intended target instead of circling around the target site as with most fixed-wing UAVs. A hybrid UAV with both cruise flight and VTOL capability will be very useful in such tasks. A few prototypes of hybrid aircrafts fitting these tasks have been designed in the past, such as the Tail-sitter airplanes developed by the University of Sydney and the University of Compiegne [46], the hybrid UAV developed by icarusLabs [47] and the tilting-wing quadrotor–SUAVI [48], developed by Hancer and his teammates. However, these existing designs are usually only optimal in one mode of flight and largely sacrifice the performance of the other mode. For example, the main structural design of SUAVI ensures its stable flight in the VTOL mode. However, its airfoil structure designed for the cruise mode is just four small pieces of airfoils attached to the tilting wings. Its cruise mode is far more inefficient when compared to conventional airplanes.

The ideology for a true hybrid UAV was one that allows the UAV to achieve optimal flight performances in both VTOL and cruise mode. In the rest of this chapter, we highlight the design and development methodology that enabled us to rapidly develop our unconventional UAV, "U–Lion". U–Lion is the preliminary prototype developed which not only has two modes of flight, but is also capable of restructuring the platform shape by folding or expanding its wings. This special design aims to achieve stable and efficient flight in both VTOL and cruise modes.

2.6.1 Design Methodology

Our design methodology with reference from Cai et al. [38] was an iterative way of four steps shown in Fig. 2.32. The first step was overall layout design, which includes the brainstorming

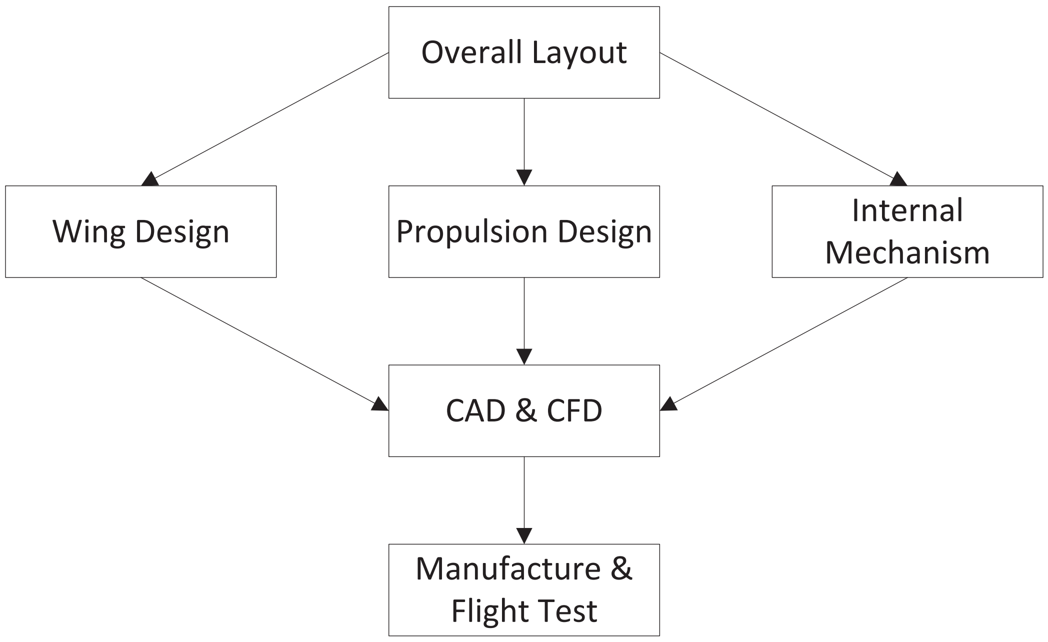


Figure 2.32: U–Lion Design Methodology.

process, aircraft configuration establishment and weight estimation. The second step includes: (I) Wing design, in which we designed the wing based on parameters that can satisfy our requirement for the lift force. (II) Propulsion design, in which we designed the propulsion system comprising rotors and gimbals. (III) Internal mechanisms design, where several mechanisms were designed to realize reconfigurable wings, adaptive CG and canard wings. The third step was generating the computer-aided design (CAD) and computational fluid dynamics (CFD) analysis. This step allowed us to realize and validate our design virtually. We designed all the components and parts for our UAV in detail, and exported the geometry to the CFD software to validate their aerodynamic performance. We may have several iterations in this step until all of the design specifications are satisfied. The last step is implementation and flight test. Carbon fibre and Expanded PolyOlefin (EPO) foam were used as the materials for U–Lion because of their low weight and good structural characteristics. Flight tests were then conducted for our U–Lion including VTOL tests and fixed-wing tests until the platform's flight performance meets our expectation.

The design of this unconventional UAV was motivated by the requirements of unconventional UAVs with the capabilities of VTOL, hovering, as well as long-range, high-speed flight. The requirement of VTOL leads to our design incorporating the features of rotary propulsion instead of the very popular flapping-wing mechanism found in recent research [49], [50], [51].

For the rotary propulsion approach, single-rotor helicopter is the most conventional and mature platform, applied to both research and practical cases. The rotating torque produced by the main rotor is countered by the tail rotor mounted with a specific distance from the main rotor. However, the complexity of the swash-plate mechanism makes the single-rotor propulsion d-ifficult to be implemented. Another approach is the use of co-axial rotary propulsion, which makes use of two contra-rotating motors to counter the torque, providing a clean and vectorized thrust. The direction of the vector thrust is controlled by a gimbal mechanism activated by two electric servos, whereby the pitch and roll channels are mechanically decoupled. The yawing motion is achieved by the difference of the upper and lower rotor rotational rates. There are also some other hybrid aircrafts adopting multi-rotor system as their propulsion methods. The Tactical Utility TU-150 from Rheinmetall Airborne Systems, with two propeller blades mounted on its wing tips, is capable of hovering and cruising at 120 knots for up to eight hours. Some research groups take the quad-rotor design as the VTOL scheme, distributing four rotors on the wing and fuselage. Certain rotors are able to tilt forward to provide propulsion for fixed-wing flight. However, limited by thrust provided and weight budget, the multi-rotor option, subjected to low efficiency and large weight, is not an ideal choice. With these considerations, the vector thrust with a gimbal mechanism outperforms other approaches with regards to weight, design complexity and energy efficiency.

The mission of long-range flight and high cruising speed brings about several problems. Wing shape, configuration and its layout are main factors that affect the efficiency, which are relative to the endurance, maximum velocity, as well as the maneuverability.

The wing shape is typically defined by aspect ratio. The aspect ratio is a measurement of how long and slender the wing appears when seen from above. Generally, wings are categorized into low, moderate and high aspect ratio according to different length-to-breadth ratio. Airplanes with low aspect ratio wings are more structurally efficient and have higher instantaneous roll rate. Low aspect ratio planes allow for high and ultra high speed, and more aggressive maneuverability. The moderate aspect ratio wings are made for most of the general purpose aircrafts with requirements on moderate velocity and endurance. For the high aspect ratio wings, with a long and slender appearance, they are applied to the aircrafts capable of long range and extremely stable cruise flight due to aerodynamical efficiency of the wings and less induced drag. For this unconventional UAV, the moderate aspect ratio (between 2 to 7) are selected due to the tradeoff between high speed and long endurance.

Wing sweep angle is one of the main factors that affect the aerodynamic characteristics and efficiency of wings. The straight wings are the most structurally efficient ones that are adopted by the majority of low-speed aircraft designs. Some wings sweep forward from the root to the tip to avoid tip stall problems and reduce tip losses, while forward swept wings are subject to aeroelastic flutter. The example of forward swept wings can be found on the design of Su-45 Berkut. However, swept wings aircraft are often developed with the wings sweeping rearwards from the root to the tip. At high speed range, the swept wings have lower drag so that the aircraft is provided with high maneuverability. Thus, in order to achieve both long endurance and high speed within one platform, the variable sweep wing is chosen to make the wing positions configurable at different occasions to suit different operational needs. A four-bar mechanism was developed to meet the requirement of wing repositioning in our design and will be describe in Section 2.6.2.

2.6.2 Design of U–Lion

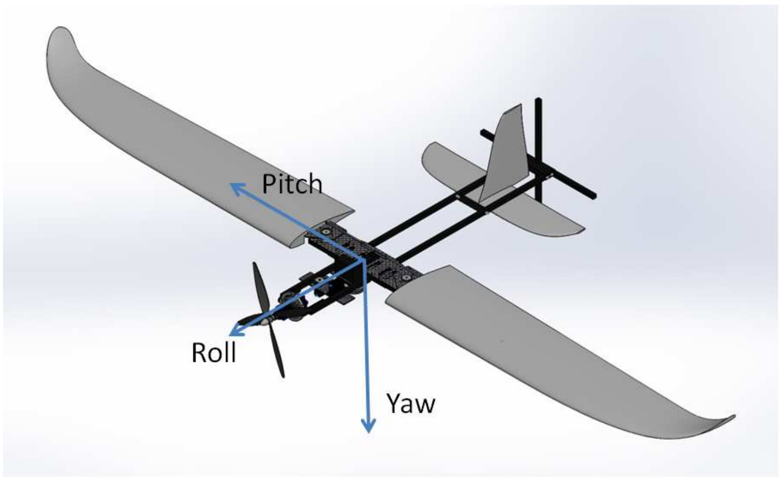


Figure 2.33: CAD drawing of U–Lion in cruise mode.

Wing Design

Wing design is the most crucial part for an aircraft design, as the lift created by the wings needs to exceed the gravitational force by around 50% to provide enough control margin. To describe the performance of wings, several parameters are generally used including the lift coefficient C_L, drag coefficient C_D and wing loading. These characteristics are affected by aircraft forward velocity, angle of attack (AOA) and wing airfoil shape [52]. The following formulae represent

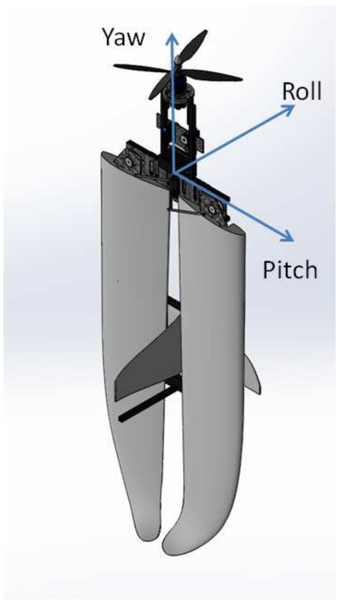


Figure 2.34: CAD drawing of U–Lion in hovering mode.

the lift coefficient and drag coefficient,

$$C_L = \frac{2L}{\rho v^2 A}, \tag{2.23}$$

$$C_D = \frac{2F_d}{\rho v^2 A}, \tag{2.24}$$

where L is the lift force, F_d is the drag force, ρ is the mass density of the fluid, v is the speed of the object relative to the fluid, which is actually flight speed, and A is the reference area, which is the wing area.

Aerodynamics analysis

The aerodynamics analysis is based on empirical methods for the airfoil. The wing parameters are shown in Table 2.7. To achieve high flight efficiency in cruise flight, we adopt the notion from eagles with high aspect ratio wings. "Clark Y" airfoil is selected as our choice because of its great performance in the low-speed range.

Table 2.7: Wing parameters for fixed-wing configuration.

Parameter	Value
Reference wing area	0.36 m^2
Wing span	1.8 m
Wing chord length	0.2 m

In the condition of $8\ \mathrm{m/s}$ flow velocity and $10°$ AOA, the Reynolds number is:

$$Re = \rho V l/\mu = 1.1 \times 10^5, \tag{2.25}$$

then the lift coefficient of Clark Y airfoil is estimated to be 1.3 using lift and drag coefficient curves generated by the Profili software, which is shown in Fig. 2.35. Hence the lift force can be calculated as:

$$L = \frac{1}{2} C_l \rho V^2 S = 19.3\ \mathrm{N}, \tag{2.26}$$

Assisted by the vectoring thrust provided by the coaxial propeller and lift force from the fuselage, the lift force from the wings is sufficient to support the 2 kg weight of U–Lion in cruise flight.

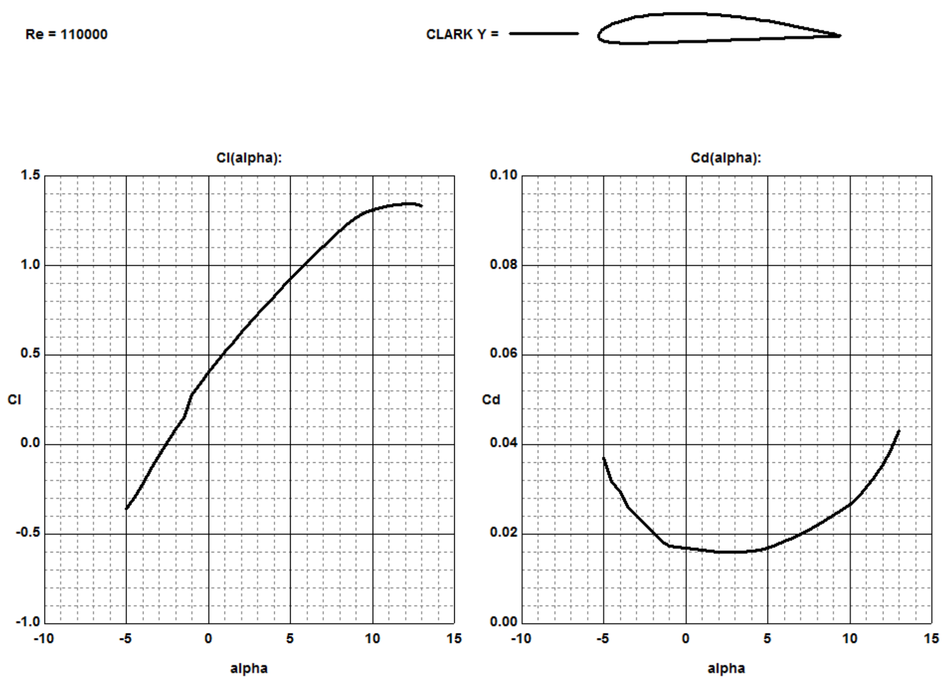


Figure 2.35: Lift and drag coefficient curves of Clark Y airfoil when $Re = 1.1 \times 10^5$.

Vector Thrust Design

Since U–Lion was designed to achieve VTOL, hovering, and cruise flight capabilities, we require the thrust mechanism to be designed with the following features:

1. For VTOL mode, the thrust is enough to lift the gross take-off weight of the aircraft.

2. For full-envelope VTOL mode control, 3-axis attitude control should be realizable including pitching, rolling and yawing motions.

3. For cruise mode, the mechanism is able to assist cruise flight in fixed-wing mode.

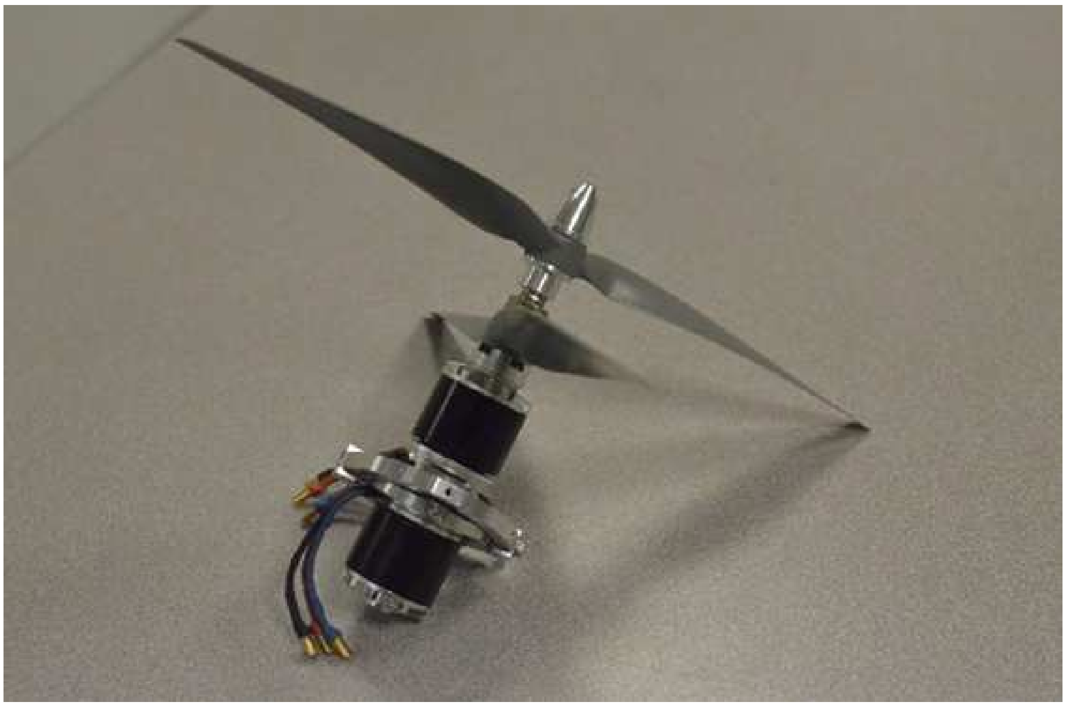


Figure 2.36: The self-customized contra-rotating motor.

The overall take-off weight inclusive of the battery was estimated to be $2\ \mathrm{kg}$. Thus, the co-axial contra-rotating rotor must be able to provide at least $3.5\ \mathrm{kg}$ thrust for the purpose of attitude control and maneuverability. After surveying commercial-off-the-shelf (COTS) brushless motors, a brushless motor with $1100\ \mathrm{Kv}$ was selected to build the contra-rotating rotor. It is shown in Fig. 2.36 that the shaft in the upper motor was removed and an extended lower motor shaft was used in its stead. A bearing was inserted between the upper motor and the lower motor shaft to realize the free movement between the two rotors. A 11×7 pusher propeller was fixed onto the shaft and a smaller counter-clockwise 10×5.5 propeller was screwed on the upper motor. In the hovering condition, the torque of each rotor was counterbalanced with the same revolutions per minute (RPM) and yawing motion was created with the difference between the upper and lower rotor. As shown in Fig. 2.37, for pitching and rolling motions, a gimbal device was installed to realize the vector thrust pointing to any desired direction. An aluminum inner ring was fastened with the outer ring by two bearings between them and the outer ring was assembled on the top of the aircraft. Both of the gimbal rings were attached with a linkage

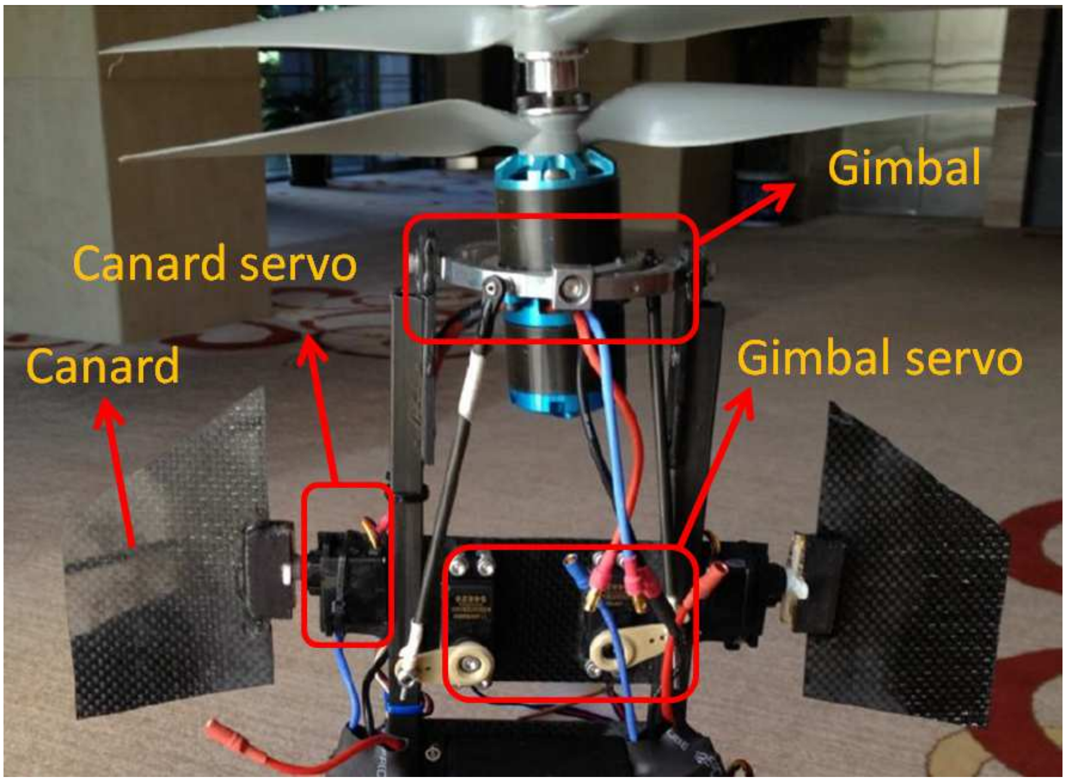


Figure 2.37: U–Lion propulsion system.

controlled by a servo. The angle of the servo horn could be adjusted to realize the required angle of the gimbal ring which results in a full-envelope control of the UAV. With this vector-thrust contra-rotating motor, x-axis and y-axis rotating motions are realized. To create a greater yaw motion, two rotating canard wings were designed in the down-flow areas of propellers to assist yaw control in VTOL mode.

Reconfigurable Wing Design

The tail-sitter is capable of two flight modes: cruise flight mode and VTOL mode. For cruise flight mode, a larger wing span is preferable to generate more lift at a lower speed. For VTOL mode, the aircraft has to maintain a small footprint to reduce the effect of wind disturbance. The fully extended wing in VTOL mode also makes it harder for the vehicle to maneuver in its heading direction. Keeping this trade-off in mind, we came out with a reconfigurable wing design which extends the wing in cruise mode and retracts the wing in VTOL mode. To keep the system structure simple and minimize the platform weight, the reconfiguration is implemented using a four bar linkage design [53]. The design procedure and implementation details will be covered in this section.

Four-bar Linkage Design

The design of a four-bar system involves the determination of each of the four bars' length subject to a set of constraints. In this design, we have the following constraints:

1. **Single servo driving**: To simplify the system structure and reduce the weight of the system, we design two sets of four-bar linkage system which use only one driving servo. As shown in Fig. 2.38, there are two sets of four bar linkage mechanisms, $\{OABE\}$ and $\{OCDF\}$. The three revolute joints (O, E, F) are fixed to the air frame. Joint B and D are attached to the two sets of wings. While the servo drives the bar CA to sweep a certain angle around joint O, both EB and FD will change accordingly, adjusting the wings' spanning angle.

2. **Minimum load on servo**: In order to exert minimum load on the servo bar OA, especially when the wings are fully extended, one extreme position is set to the position where the rocker OA and the coupler AB are co-linear with each other seen in Fig. 2.39. In this condition, the load induced by the right wing are compensated by the one from the left wing. This corresponds to the case when the wings are fully extended. In the other aspect, in order to effectively extend the wings from vertical mode to horizontal mode, the transmission angle between AB and EB must be defined specifically. We design the transmission angle to be in the range of $[45°,\ 90°]$.

3. **Wing sweeping angle**: To fully extend and retract the wings between horizontal mode and vertical mode, the sweep angle of the wing are designed to be 90°, denoted by 2β in Fig. 2.39. What's more, the sweeping angle of the rocker OA is defined as ϕ and set to be 90°. The two extreme positions of the reconfigurable wing are labeled as $OABE$ and $OA'B'E$ in Fig. 2.39.

In practice, referring to Fig. 2.38, the distance between revolute joint E and F is determined by the overall vehicle frame size. The length of the rocker arms OA and OC are defined by the available servo horn length. With all the above constraints, we could derive the length of linkage bar system as:

$$EB = \frac{0.5\,EF \cdot OA}{(\,EF - OA\,)\,\sin\beta} \tag{2.27}$$

$$AB = 0.5\,EF - OA + EB\,\sin\beta \tag{2.28}$$

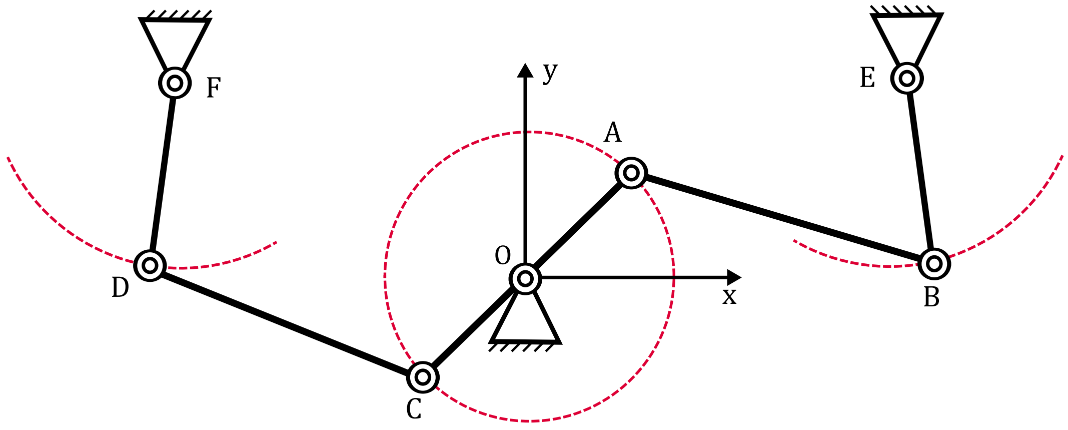


Figure 2.38: Symmetric four-bar linkage system.

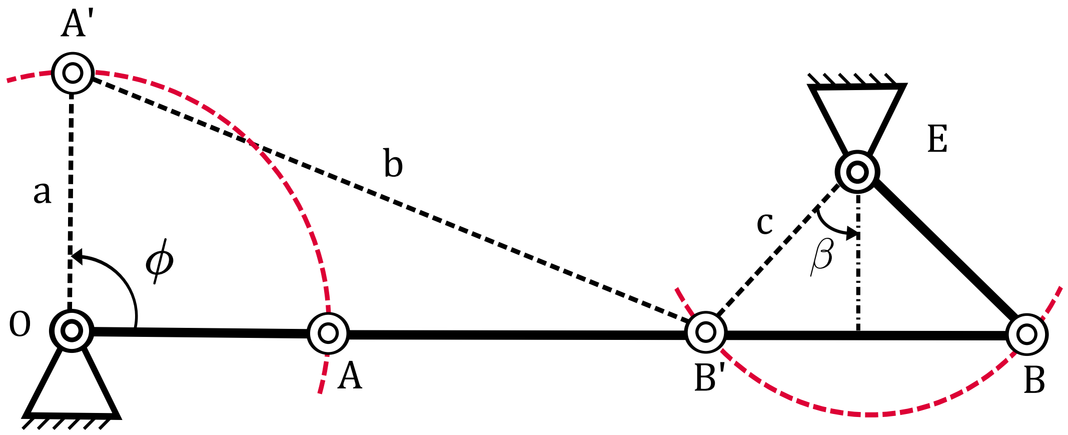


Figure 2.39: Parameter determination of four-bar linkage.

With the specified constraints and the Eq. 2.27 and Eq. 2.28, we could derive the parameters of four bar linkage system as shown in Table 2.8.

Table 2.8: Parameters of four bar linkage.

Parameter	Value
OA/OC	38 mm
AB/CD	109.4 mm
EB/FD	31.68 mm
EF	250 mm
Rocker Sweeping Angle	90 deg

Four-bar Linkage Implementation

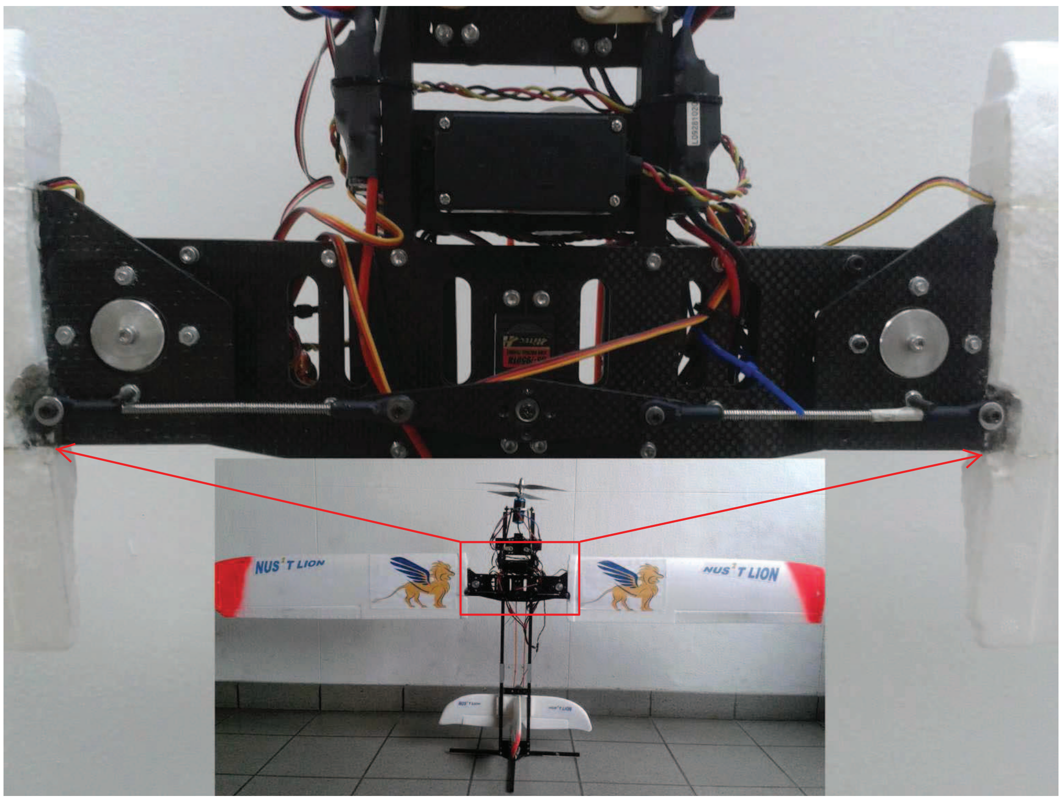


Figure 2.40: Extended wing configuration.

With the designed four-bar linkage parameters determined, we build the reconfigurable wing structure as shown in Fig. 2.40 and 2.41. Fig. 2.40 illustrates the case when the wings are fully extended for cruise flight and Fig. 2.41 demonstrates the retracted wing configuration for VTOL mode.

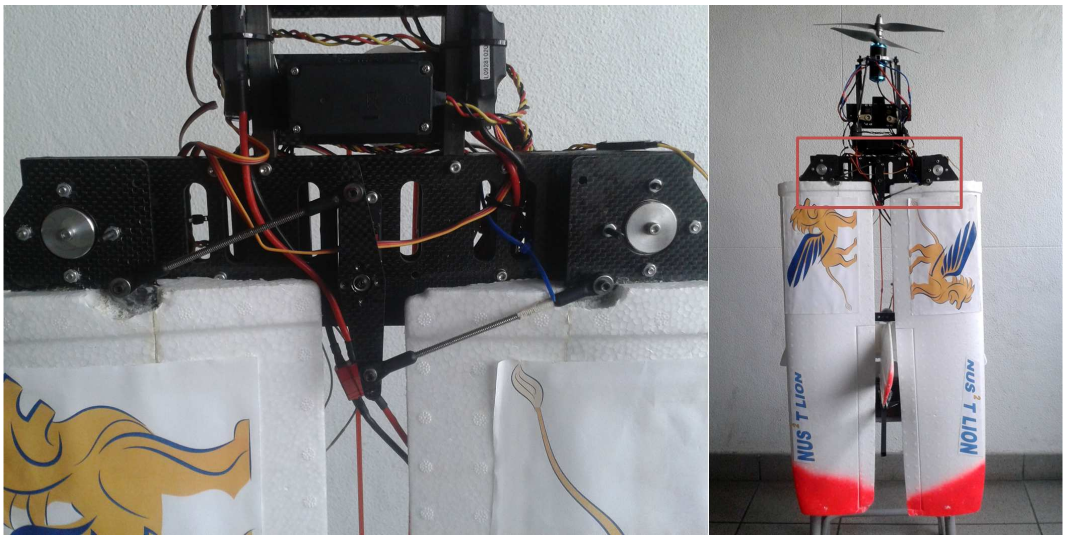


Figure 2.41: Retracted wing configuration.

2.6.3 Adaptive Center of Gravity

The CG of an aircraft should be located around the center of lift of the wings during cruise flight. On the other hand, during VTOL and hovering, the CG should be located as far away from the propellers as possible, based on our experiences from flight tests. The adaptive CG mechanism is able to reposition the CG of the aircraft to a certain extent, so that the CG requirements for both cruise and VTOL/hovering could be met.

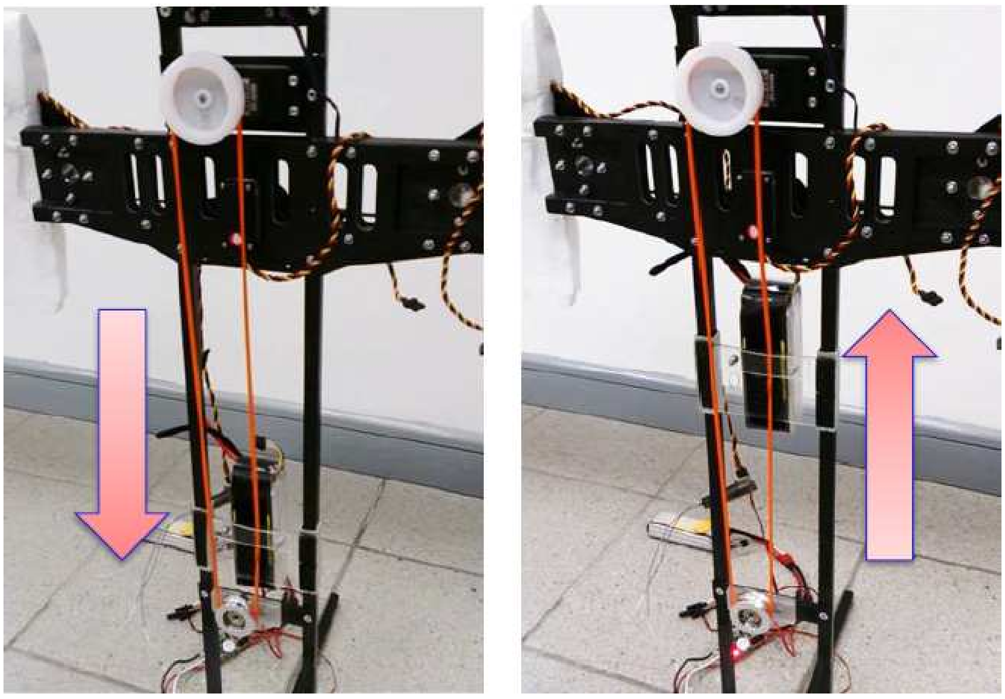


Figure 2.42: Adaptive CG mechanism.

CG position is adjusted by moving the heaviest component on the aircraft, i.e. the battery. It weighs $300\ \mathrm{g}$, about one seventh of the total mass of the aircraft. During cruise flight, the battery is located immediately after the central plates. The separation between the central plates and the tail, i.e. the maximum distance that the battery can travel, is $40\ \mathrm{cm}$. By pushing the battery all the way from the center to the tail, the CG of the aircraft can be pushed towards the tail by $5-6\ \mathrm{cm}$. The mechanism to move the battery consists of a pulley near the tail, a servo motor near the propeller, a belt and a slider on the fuselage. The battery is strapped to the slider which is attached to the belt. The servo turns the belt, and the battery moves accordingly.

Although it is not the definitive solution to the CG problem between transitions, the adaptive CG mechanism has proved to be helpful. During our flight tests, significant improvement in hovering stability is observed after installing the mechanism. Stability in VTOL/hovering modes turns out to be affected by more than CG position alone. Large surface area, inadequate power from propeller and other unknown factors could all have contributed to the instability, which is why the adaptive CG mechanism is only a complementary solution to the stability problem. The current version of adaptive CG mechanism is a promising solution to the stability problem between transitions. Further improvement in the mechanism, as well as a more thorough understanding of the causes of instability, are necessary to completely address the stability issue.

2.6.4 Material Stress Analysis

Based on flight tests, the most vulnerable part of the fuselage is determined to be the central plate. A preliminary structural analysis of the central plate is thus conducted.

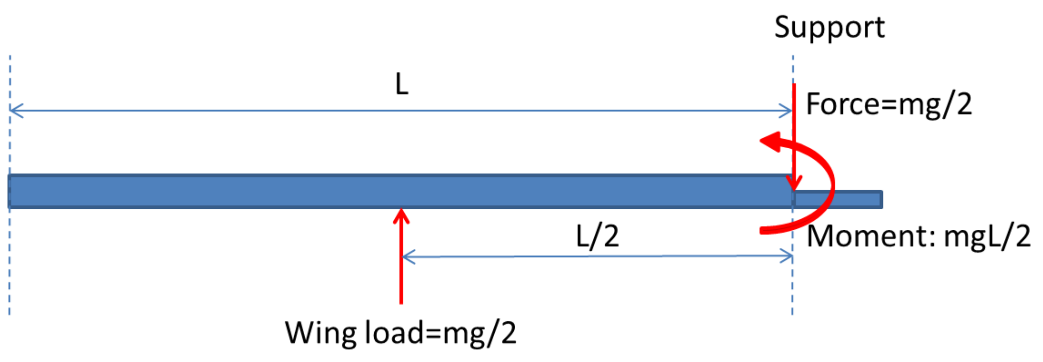


Figure 2.43: Free body diagram of the wing.

As shown in Fig. 2.43, the reactions at the wing-body joint consist of one force and one mo-

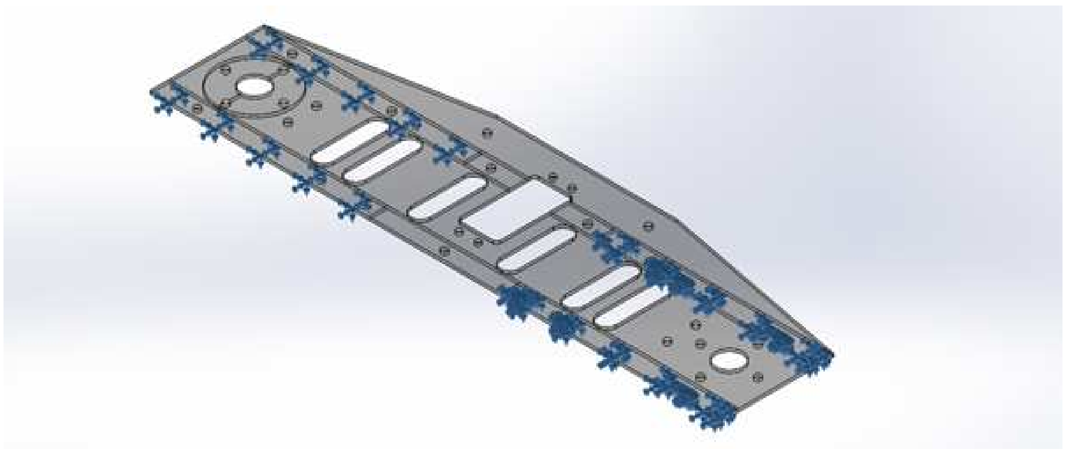


Figure 2.44: Fixtures in FEM Simulation.

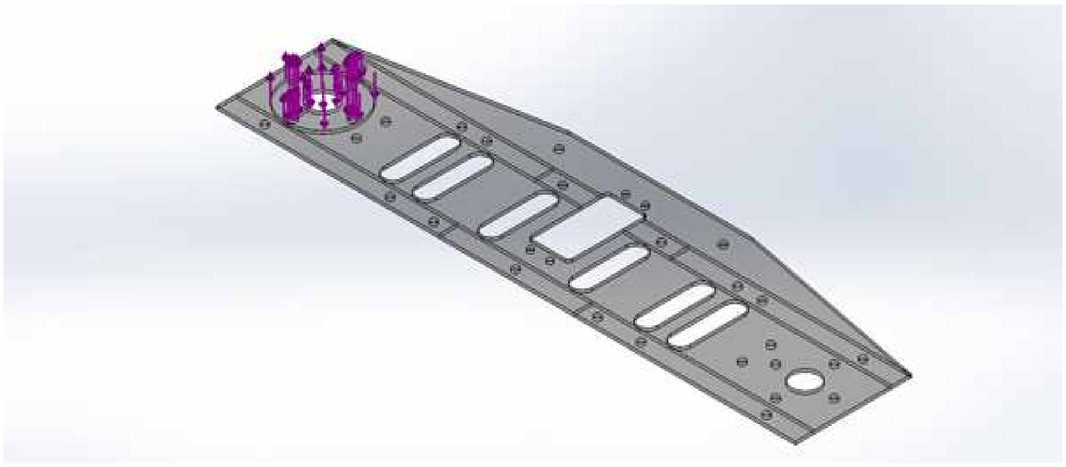


Figure 2.45: Load in FEM Simulation.

ment. Assuming that sections of the plate glued to carbon fiber square tubes are fixed (Fig. 2.44), and that load is applied to the plate at where the bearing holder and the plate contact (Fig. 2.45). The load on the plate is equivalent to 10 N of lift at the center of each wing ($mg/2 = 10$ N in Fig. 2.43) with a safety factor of 1.5. With this set-up, a Finite Element Method (FEM) simulation is carried out using SolidWorks Simulation. As shown in the Fig. 2.46 and Fig. 2.47, the maximum stress is well below the yield strength of carbon fiber, and the displacements are negligible.

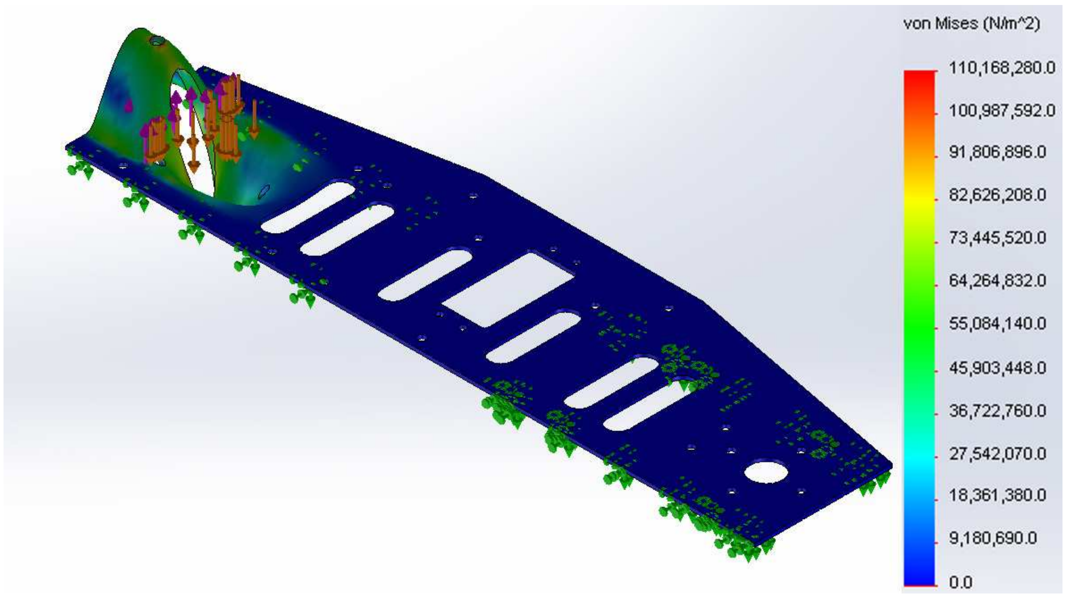


Figure 2.46: Stress simulation conducted on central plate.

A complete FEM simulation, however, is necessary to get a thorough understanding of the behavior of the fuselage when loaded. That analysis is beyond the capability of Solidworks Simulation, and thus requires more sophisticated FEM software and techniques [54].

2.6.5 Electronic Configuration

The electrical components are listed in Table 2.9. The motor is a self-customized contra-rotating motor with maximum lifting force generated at around 35 N. The two electrical speed controller (ESC) are chosen to be DualSky 40XC4018BA for the accuracy of the controlling the speed of the motors. The weight of the ESC is 30 g and the maximum current allowed is 40 A which is suitable for the control of the rotating motor.

In this U–Lion design, there are quite a few actuators with different requirements. Based on the requirements, different servos are chosen for minimum weight and larger safety margin. The

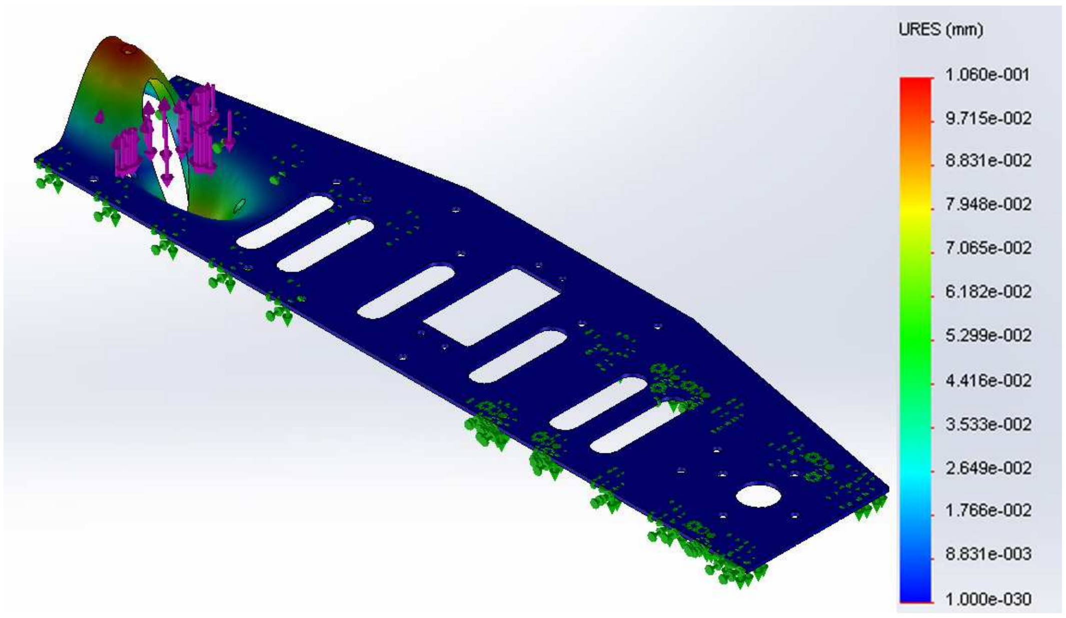


Figure 2.47: Displacement simulation conducted on central plate.

Table 2.9: List of electrical components.

Component	**Part Number**
Motor	Himax CR3516
Propellers	APC 11×5.5P & 10×7
ESC	Dualsky 40 XC4018BA
Gimble servo	Airtronics 94820 Servo
Four bar mechanism servo	HS7950TH
Wing control surface servo	HS-65MG
Tail control surface servo	HS-85MG
Canard servo	HS-65MG
Gyro	Futaba GY240
Receiver	Futaba R6014HS
Transmitter	Futaba 14FG
Battery	4S 2200mAh

specifications of the selected servos are listed in Table 2.10. The servo used for supporting the gimbal mechanism is selected to be the Airtronics 94141Z. The gimbal support servo must have a fast response for good controllability and the torque has be large enough for supporting the gimbal movement. Since the maximum tilting angle of the gimbal is 30° and the radius of the gimbal is 3 cm the maximum torque generated to the gimbal is approximately $2 \times \sin 30^\circ \times 3 = 3\ \mathrm{kgcm}$. The selected Airtronics servo fulfils the requirement with maximum torque of $4\ \mathrm{kgcm}$ with a response speed of 0.16 sec/60°. The weight of the Airtronics servo is 32.9 g which is light for U–Lion.

Table 2.10: Specifications of servos.

Component	**Part No.**	**Torque** (kgcm)	**Speed** (sec/60°)	**Weight** (g)
Gimble servo	Airtronics 94820	4	0.16	32.9
Four bar servo	HS7950TH	29	0.15	68
Wing control servo	HS-65MG	1.5	0.11	10
Tail control servo	HS-85MG	3.5	0.14	21.83
Canard servo	HS-65MG	1.5	0.11	10

The wing control surface servos and canard servos are selected to be HS-65MG servo. The main reason to select this servo is because of its light weight. The HS-65MG servo only weighs 10 g while providing a relatively high torque of 1.5 kgcm and a fast response speed of 0.11 sec/$60°$. For the tail control surface servo, since the tail control surface requires larger torque and more stable response, the servo is selected to be HS-85MG. The HS-85MG servo has a higher torque of 3.5 kgcm and the weight is 21.83 g.

The servo for the four bar mechanism is selected to be HS7950TH which has a high torque and low weight. Based on the four bar mechanism calculation and the weight of the wings, in order to support the foldable wings, the servo has to provide a torque of at least 15 kgcm. The HS7950 servo could provide a torque of 29 kgcm which fulfils the requirement with a safety factor of around 2. The weight of the servo is 68 g which is lower than other servos that have similar torque. The reaction time for the servo is 0.15 sec/60° which provides a prompt response for the wing folding. The main controller for the VTOL mode is the gyro controller, which serves as a proportional controller for the attitude control. The gyro is selected to be Futaba GY240. The gyro has a very accurate gyro sensor and fast controlling rate (70 Hz) which is suitable for U–Lion.

2.6.6 Control Modes

The control methods for VTOL mode and cruise flight mode are quite different for a UAV. For conventional VTOL UAV, such as helicopter or co-axial helicopters, the VTOL six degree control is achieved by varying the angle of wing trust. However, for the fixed wing plane, the motion control is achieved by the control surface on the wings or the tails. The difficulty of unconventional UAV with VTOL and cruise flight ability lies greatly in how to combine the two control modes as well as the transition between the two modes. In our UAV design, we adopt

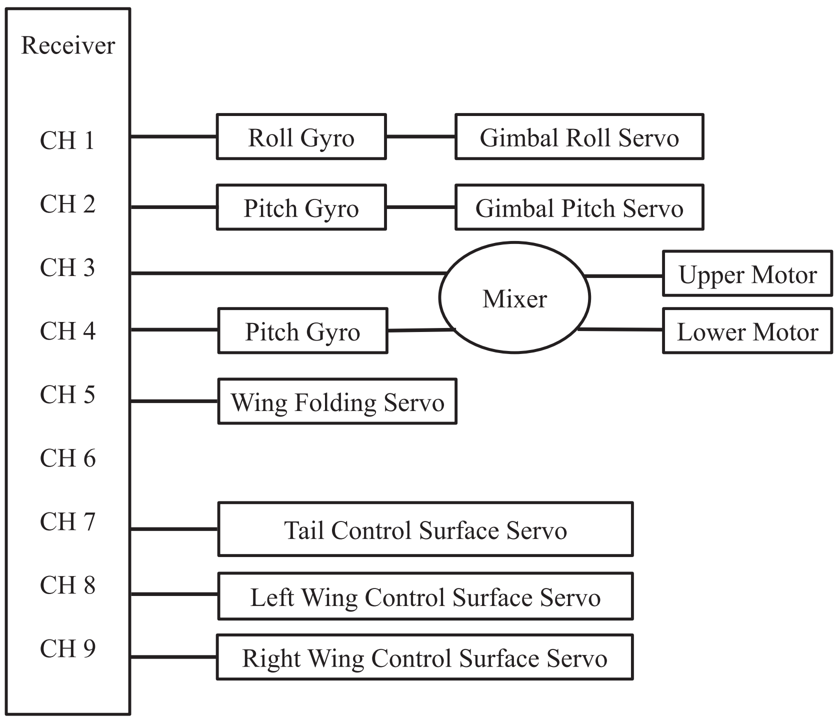


Figure 2.48: The control circuit of U–Lion.

the vector trust for VTOL control and control surface for the cruise flight control.

The control involves three modes of control, i.e. VTOL mode, fixed wing mode and the mixed control mode. The U–Lion is controlled manually at the current stage. The onboard electronic circuit mainly consists of a receiver and extending components, as shown in Fig. 2.48. The control logic is programmed in the transmitter as shown in Fig. 2.49. Defining the global coordinate as the standard North-East-Down (NED) XYZ frame with X axis pointing north, Y axis pointing east and Z axis pointing down. VTOL flight requires the motor to be pointing in the negative Z direction and cruise flight requires the motor pointing in XY plane. Defining the local frame of U–Lion to be ($x_{\text{local}}y_{\text{local}}z_{\text{local}}$) with z_{local} axis pointing down, y_{local} axis pointing to the right wing and parallel to the XY plane, x_{local} axis perpendicular to the y axis and z axis following the right hand rule. The coordinate system is shown in Fig. 2.50. Due to the special tail sitter design, we can see that the pitch, roll and yaw movement agrees in both VTOL mode and cruise mode. Thus it allows a smooth transition in control signals between VTOL and cruise mode. This results in a clear control logic for manual control and paves the way for future automated control.

For the VTOL control mode, the main controller for the VTOL mode is the gyro controller,

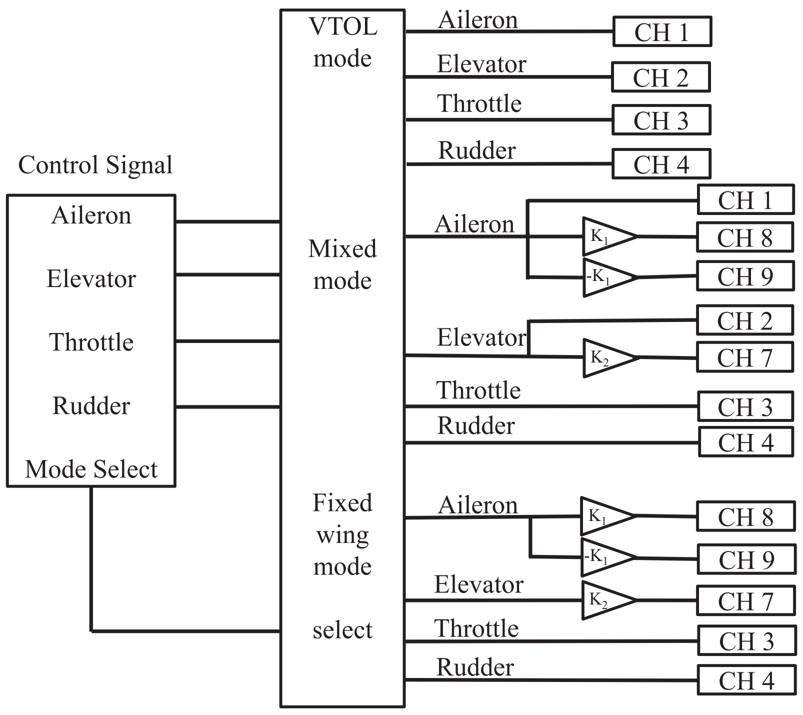


Figure 2.49: The control logic of U–Lion.

which serves as a proportional controller for the attitude control. Three gyros are used in U–Lion, two of them are sensing the pitch and roll movement in VTOL mode. They are connected to the servos supporting the gimbals as shown in Fig. 2.36. The gyro will sense the rate of the attitude change in each of the channel and feedback to the servo to change the direction of the trust which stabilizes the UAV and control the attitude for desired motion. The third gyro is placed to sense the yaw movement of the UAV. The raw control signal generated by the yaw gyro is then mixed with the throttle input signal and sent to the contra-rotating motors to control the yaw and heave motion in the VTOL mode.

In the fixed wing mode, the wings are reconfigured into the horizontal position. The vector trust is fixed in vertical position and the control is achieved by changing the control surface on the wings and the tails. In this way, after transiting into fixed wing mode, the plane can just fly similar to a normal fixed wing plane.

The mixed control mode is the one that combines the above mentioned two methods. The vector trust will change its direction and the control surface can vary to assist the control in both VTOL and fixed wing mode. The canard will assist the yaw control in the VTOL mode; the variation of the tail control surface will assist the pitch control so as to achieve a more stable flight performance and better control performance. In the fixed wing flight, the variation of the vector thrust system will allow for a more agile motion control. The plane could have a more aggressive control performance.

The transition between VTOL mode and cruise mode requires the U–Lion to transform from vertical flight into horizontal flight. The mixed control mode serves as an important bridge between VTOL mode and cruise mode. The change of pitch angle of the vector trust will push the head down and the elevator on the tail also assists this transition process. The control surface on the wings will help to stabilize the UAV during transition. After transition, the wing and tail control surfaces can effectively control the UAV and the control mode is switched to cruise mode.

2.6.7 The 2^{nd} AVIC Cup - International UAV Innovation Grand Prix

The 2nd AVIC Cup International UAV Innovation Grand Prix (abbr. UAVGP) is a large-scale biennial aviation event authorized by the Ministry of Science and Technology, and co-organized by the Aviation Industry Corporation of China (AVIC) and Chinese Society of Aeronautics and Astronautics (CSAA), hosted by AVIC Culture Co. Ltd (ACUL). As an aviation mega festival, UAVGP provides an opportunity to show innovative ideas and new-technological works, as well as act as a platform of international communication for individuals and groups all over the world.

The UAVGP comprises several different competition categories such as the Athletic Grand Prix, the Creativity Grand Prix and the Air Show competition. U–Lion was registered into the Creativity Grand Prix along with 76 other teams. In the Creativity Grand Prix, competitors had to design a new type of aircraft that has unique characteristics and prove that it was capable of stable flight. During the competition, the different teams all exhibited their platforms and while there were a few really novel platforms, a lot of platforms could not perform due to malfunction or crashed during flight. U–Lion performed flawlessly and demonstrated its flight capabilities of VTOL then transiting to cruise mode before hovering and finally landing. The NUS team achieved 11^{th} position out of 76 teams and garnered the New Innovation Star Award for U–Lion's unique features.

The flight on the competition day at Beijing is shown in Fig. 2.51. The subfigures in the figure are cropped from the video taken on the day with 0.25 seconds time difference between each frame. From the figures we can see a very smooth transition between VTOL mode and cruise mode. The flight shows very good dynamic performance of the U–Lion.

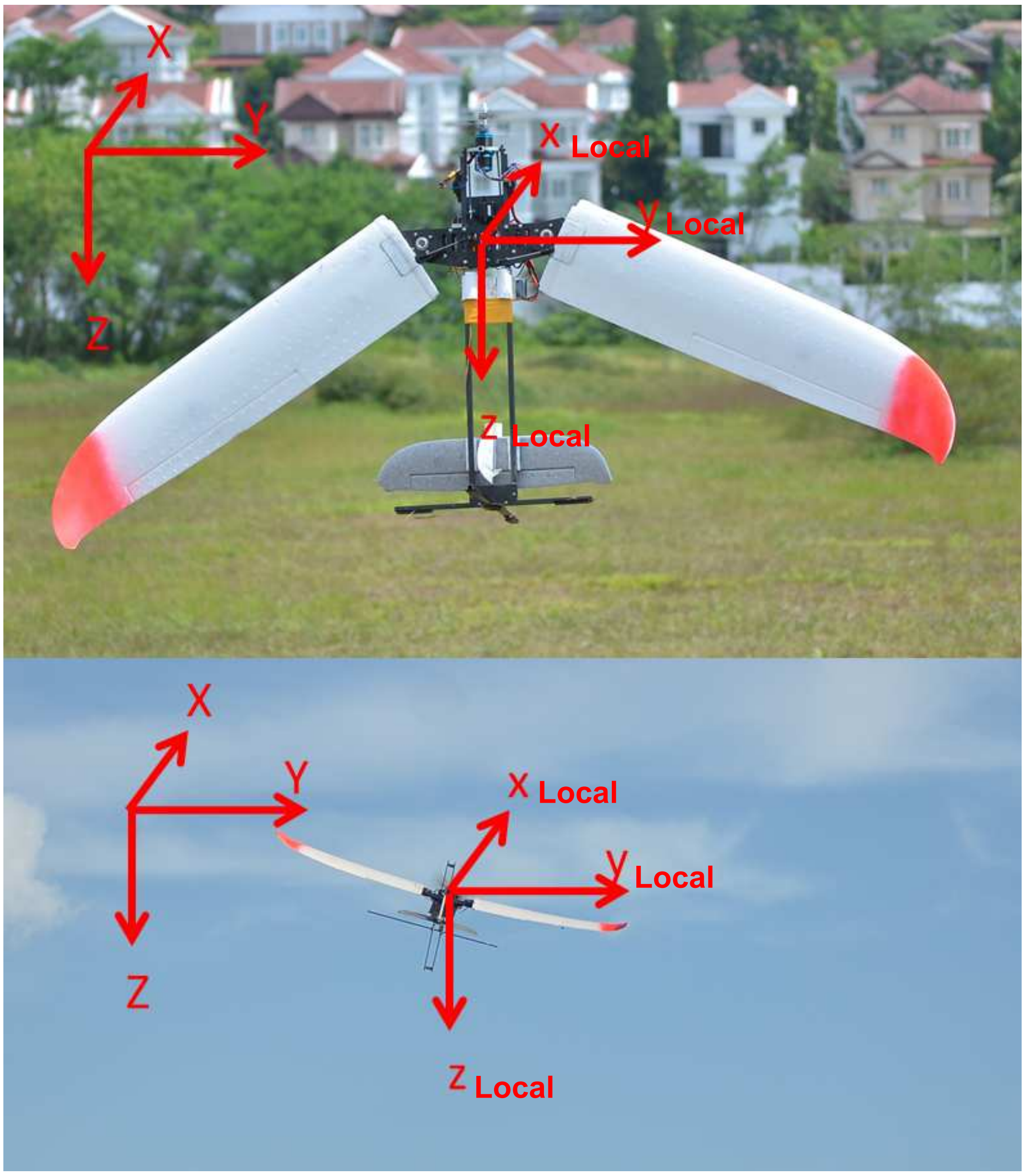


Figure 2.50: The coordinate definition of U–Lion.

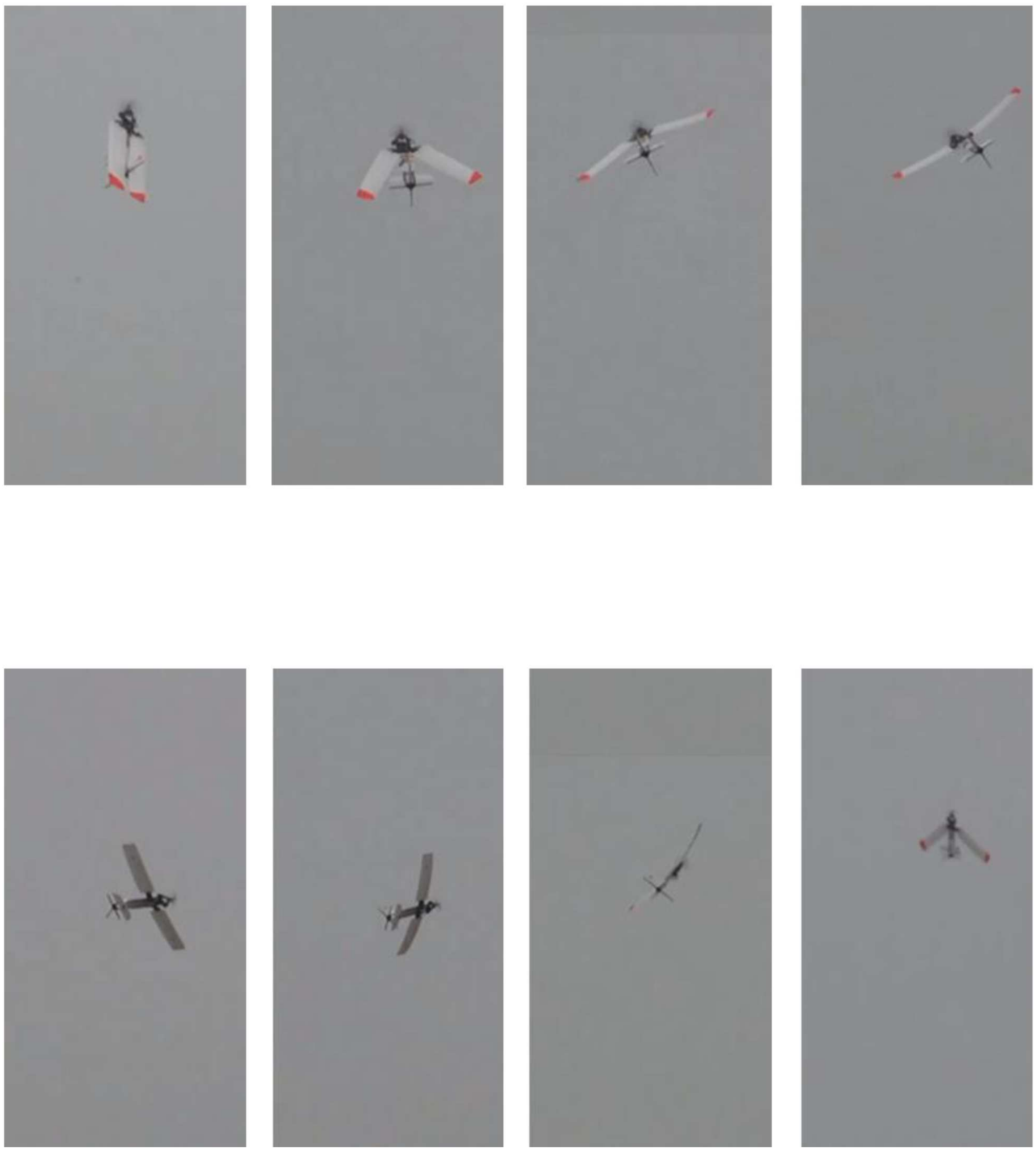


Figure 2.51: U–Lion displaying wing reconfiguration on the competition day.

Table 2.11: U–Lion Mark III Specifications.

Specification	**Value / Unit**
Weight	$2.2\ \mathrm{kg}$
Max Thrust	$40\ \mathrm{N}$
Max Cruise Speed	$10\ \mathrm{m/s}$
Flight Endurance	$10\ \mathrm{mins}$
Dimensions	$2.2\ \mathrm{m} \times 0.9\ \mathrm{m} \times 0.3\ \mathrm{m}$

In this section, we presented the development and implementation of the U–Lion. U–Lion has been designed with a reconfigurable wing and a tail-sitter structure, which combines the advantages of a fixed-wing plane and a rotor helicopter effectively. During the competition, U–Lion has demonstrated that it could transit from vertical takeoff to a hovering stage before flying in cruise mode to realize efficient long duration flight. U–Lion also performed well in the creative category with the special internal designs that empowered its capabilities. These are namely, the reconfigurable wings, the adaptive center of gravity and the unique contra-rotating thrust-vectored propulsion system.

Further experiments are in progress to obtain a reliable and accurate dynamic model of the U–Lion such that we can fulfill fully-autonomous flight in the future. In addition, we intend to implement intelligent algorithms such as vision-based techniques in obstacle avoidance and target tracking to allow the U–Lion to be used in multi-task autonomous missions.

2.7 Conclusion

In this chapter, we described in detail the methodology used in the development of our platform which covers the design and implementation of the hardware, avionics and sensor systems. The chapter then covered in detail the modeling of the quadrotor platform which derives the mathematical model of the quadrotor through combining aerodynamic forces and blade theory. Many of the methods depicted are essential to the successful modeling and system identification of the quadrotor and this was proven with the model validation experiments conducted. The last part of the chapter goes into details on the development of an unconventional UAV which was later verified in the 2013 Beijing UAVGP competition. Through this chapter, the reader is able to fully understand the theory and experimental requirements to the successful development of UAV platforms.

Chapter 3

Vision-based Obstacle Detection

3.1 Introduction

UAVs that fly with a low altitude will almost always encounter obstacles in its flight path. The obstacles present in the UAV's operating environment threatens the UAV's capability to complete its mission. Therefore, it is imperative that UAVs have a form of obstacle detection and avoidance scheme to ensure its continued survival in these environments. Many current UAVs lack this obstacle detection and avoidance capability and mostly rely on the operators to plan a path that is free of obstacles prior to the start of its mission. However, obstacles exists in every environment and it is almost impossible for operators to have any precedent knowledge on new obstacles that may surface during the mission. As such, many long ranged UAV missions still require operators to be ever vigilant and reactively control the UAVs to avoid any possible obstacles. This chapter proposes the use of vision sensors as the primary sensor of choice for the detection of either generic or specific obstacles.

Figure 3.1: PointGrey Bumblebee2 stereo camera.

From literature, one of the most robust vision system is a stereo vision system [55] where after calibration is able to output obstacle's position with respect to the camera frame. One such camera system is the PointGrey Bumblebee2 stereo camera as show in Fig. 3.1. We have also built our own customized stereo system which give us the liberty to adjust its baseline based on different uses. Fig. 3.2 shows our stereo vision system and with it mounted on our quadrotor UAV. Stereo vision algorithms also do not have the distance scaling problem as compared to monocular algorithms [56] which require complex filtering and other sensors to correct its scaling problem. However, as with most vision-based methods, the accuracy of the vision sensors is not comparable to scanning lasers and its error will have to be addressed in Section 3.2 before going into some of the obstacle detection algorithms developed. Apart from stereo vision based algorithms, we will also explore the use of monocular vision as a way to detect depth of our obstacles in Section 3.4.

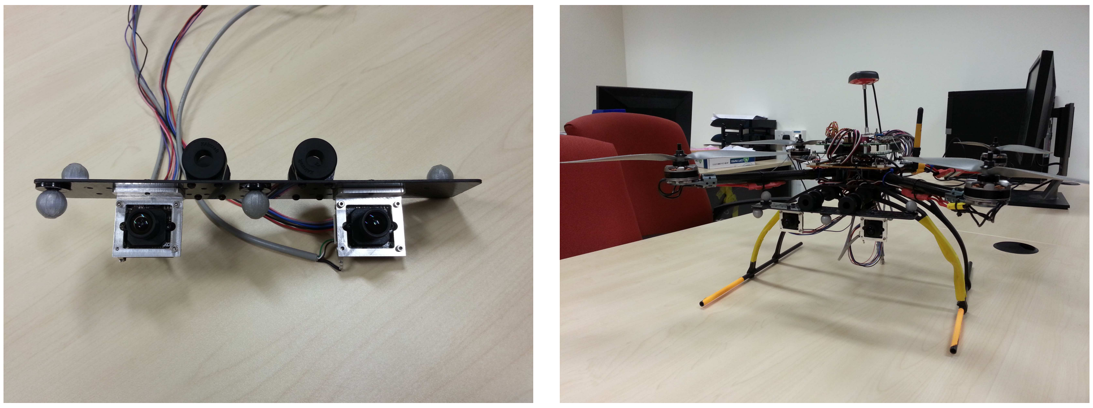


Figure 3.2: Left: Customized stereo camera, Right: Stereo camera mounted on Quadrotor.

3.2 Stereo Triangulation Error Model

Stereo vision techniques arose from the similarity that drives the stereo imaging capability that our eyes give us. It aims to emulate this capability through the use of computational systems and cameras to mimic that which nature already has. Stereo vision allows the use of rectified images to calculate a disparity map by applying the Sum of Absolute Difference (SAD) windows. The disparity map could then be projected into 3D space such that almost all image features will have an accompanying 3D coordinate. The flow of stereo vision calculation is observed below:

Stereo Vision Algorithm Flow

1. **Calibration:** Using camera calibration techniques, we could obtain the camera intrinsic

parameters. These parameters consist of radial and tangential distortion values as well as camera intrinsics such as focal length and image size.

2. **Rectification:** Warps the left and right images of a stereo pair such that their epipolar lines are row aligned. These rectified images signify the images you will get if a perfectly set up pair of cameras is achieved in a stereo rig. The images have their lens distortion removed and results in a perfect image of the surrounding.

3. **Correlation:** A disparity map is constructed by locating matching pixels in the left and right images. This is usually achieve by using the sum-of-absolute differences window to search along the epipolar line until the matching pixel is found. The difference in pixel coordinate is then recorded as the disparity of the image. The calculated disparity at each pixel is proportional to the inverse of the range of the scene.

4. **Triangulation:** With the known geometric arrangement of the cameras as found in the calibration portion, we can then turn the calculated disparity into distances relative to the camera. This is through the process of triangulation.

In a stereo system, errors in measurement can come from many sources, including quantization errors of the image, photometric and geometric distortion of the camera and numerical errors in the matching algorithm. These errors in turn cause true locations of perceived feature points to be inferred with some error. In a stereo camera setup, two cameras are usually placed at offsets of baseline distance, B, from a coordinate system centered with reference to the cameras. Assuming that the stereo camera setup is calibrated and that the images captured are rectified accurately, the projections of the same 3D point in the left and right images lie on corresponding epipolar lines. If the stereo camera setup is in the normal (left-right) camera configuration, the epipolar lines will coincide with the horizontal (v) coordinate lines. Therefore, for a given pixel at position (u, v) in the left image where u represents the $x-axis$ and v represents the $y-axis$ in the image plane, the search for the corresponding pixel in the right image will be along the $1D$ direction of (i, v) where only i varies accordingly to the number of columns present in the image.

A 3D point P, in the world coordinate is captured by the stereo camera and projects onto the left image at x_l and also projects onto the right image at x_r. Single pixel values are unstable as matching primitives, so small windows around the pixels are usually selected to act as a descriptor for the feature point of interest. For the fixed pixel on the left image at x_l, the search is established on the right image along the horizontal epipolar line until a neighborhood intensity pattern is found to be similar to the window descriptor of the left image pixel. Due to the above mentioned errors, the stereo camera system will determine x_l and x_r with some errors.

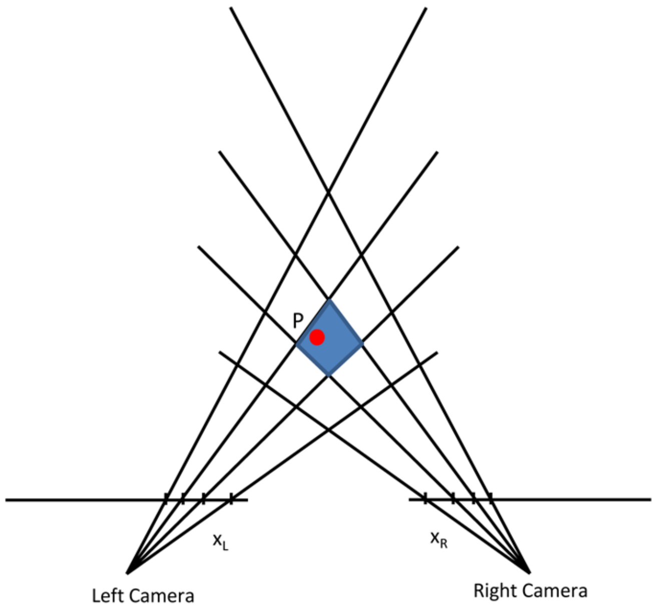


Figure 3.3: Triangulation error uncertainty.

Fig. 3.3 illustrates the errors caused by image quantization where because of the resolution limits, the estimated location of point P can lie anywhere in the shaded region surrounding the true location. Random effects can cause this region to have boundaries that are less sharp but the general shape will be similar.

There are generally two kinds of errors involved in stereo triangulation. They are namely matching correlation errors and estimation errors. In the section below, we will go about looking in detail how these errors are described and modeled.

3.2.1 Stereo Correlation Errors

Correlation errors arise from where the matches between features are inaccurate thus resulting in a wrong correlation match and wrong disparity value obtained. As mentioned in Section 3.2, the correlation of features are usually obtained using search window masks. Larger stereo masks will provide better accuracy but more smoothing of the 3D surface will be observed. From experimentation, mask size 11 is a good compromise mask size which gives accurate readings as well as limited smoothing of the 3D surface to still produce pronounced surface details.

We have conducted an experiment to test the magnitude of stereo estimation errors. The results are summarized in Table 3.1 and Table 3.2 below. The results indicate the standard deviation of the disparity matching for estimation errors.

Table 3.1: Correlation accuracy results.

Resolution	**Correlation Accuracy**
160×120	0.10 pixels
320×240	0.11 pixels
640×480	0.10 pixels

Table 3.2: Correlation accuracy results for 320×240 resolutions.

Stereo Mask	**Correlation Accuracy**
5	0.18 pixels
7	0.18 pixels
9	0.14 pixels
11	0.11 pixels
13	0.10 pixels
15	0.10 pixels

From the tables, it is apparent that the correlation accuracy improves with bigger stereo matching masks. Changing the resolution will not affect the correlation accuracy but will indirectly affect the depth calculation as the camera intrinsics will be modified.

3.2.2 Stereo Estimation Errors

Estimation errors occur where the correlation match was correct but there are some errors in estimating the position of the features thus resulting in an erroneous subpixel disparity value. The accuracy of disparity calculations are very dependent on what the camera is being pointed at, but it is generally $1 - 2$ mm at up to approximately 2 m.

In terms of testing and verifying the quality of the disparity calculations, it is very difficult to do any absolute measurements - the reason for this is that the origin (0, 0, 0) of the camera is unknown and can vary from camera to camera. The quality of disparity results will also vary from environment to environment due to shadows and lighting conditions. However, determining the quality of disparity calculations in your environment could be done relatively in a couple of different ways:

Estimation Methods

1. Point Feature

    (a) Point the camera at a feature.

    (b) Determine the position of the feature relative to the camera.

    (c) Move the feature a known amount and re-determine its location.

    (d) Distance calculation and absolute error comparison.

2. Planar Surface

    (a) Point the camera at a highly textured planar surface.

    (b) Capture all of the points on the surface.

    (c) Fit a plane to the points and then determine how far off each pixel is from the plane.

    (d) Distance calculation and absolute error comparison.

The basic equations determining X, Y, Z position of a feature point, P, in world coordinates are:

$$\frac{u}{f} = \frac{X}{Z} \tag{3.1}$$

$$\frac{v}{f} = \frac{Y}{Z} \tag{3.2}$$

$$\frac{z}{f} = \frac{B}{d} \tag{3.3}$$

where d is the disparity value, (u, v) is the pixel position in the image relative to the image centre, B is the stereo baseline and f is the focal length.

The tolerance in X, Y are determined by the calibration error p shown below:

$$\Delta X = \frac{pZ}{f_x} \tag{3.4}$$

$$\Delta Y = \frac{pZ}{f_y} \tag{3.5}$$

The tolerance of Z is obtained as shown below:

$$\frac{\delta Z}{\delta d} = -\frac{fB}{d^2} \tag{3.6}$$

Substituting Eqn. 3.3 for d,

$$\delta Z = -\frac{Z^2}{fB}\delta d \tag{3.7}$$

where δd is the uncertainty in disparity. Therefore, for a given expected error for calibration error, p and disparity error, δd, we can calculate the errors in position of the 3D point.

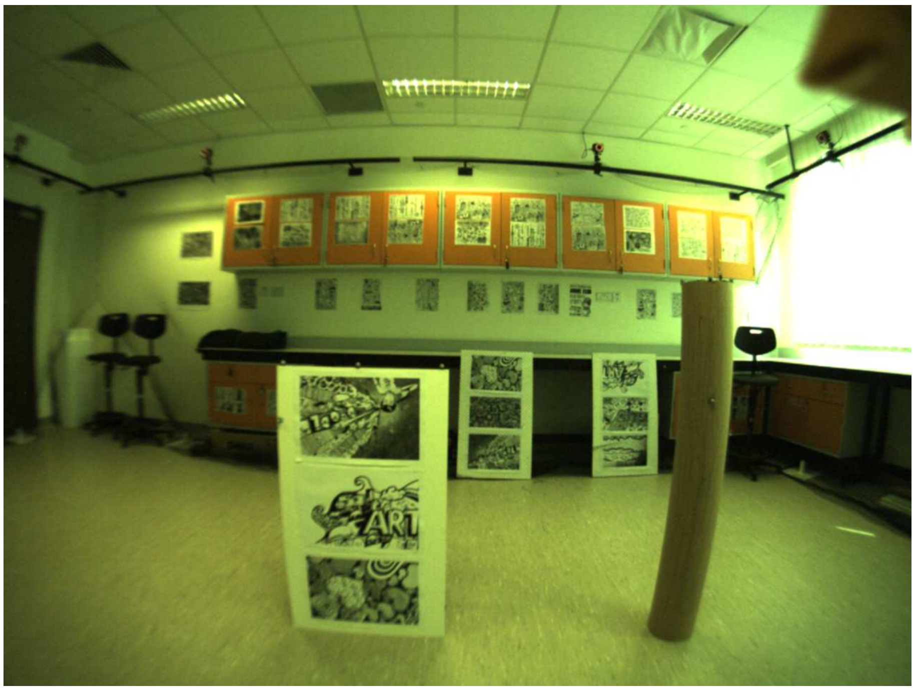


Figure 3.4: Image used for disparity error calculation.

We performed the above mentioned method of using a planar surface and a tree-trunk surface for error verification with the experimental setup as shown in Fig. 3.4 and Fig. 3.5. Both of the obstacles had highly textured surfaces which allowed a dense stereo correspondence to be calculated. The 3D coordinates are calculated for each of the disparity obtained from the planar surface and their values are then averaged to obtain a depth reading. The obtained depth readings are then compared with that of the VICON Motion Systems data which represents the true position of the test objects to obtain the absolute error for stereo triangulation. Fig. 3.6 and Fig. 3.7 shows the comparison and absolute error data.

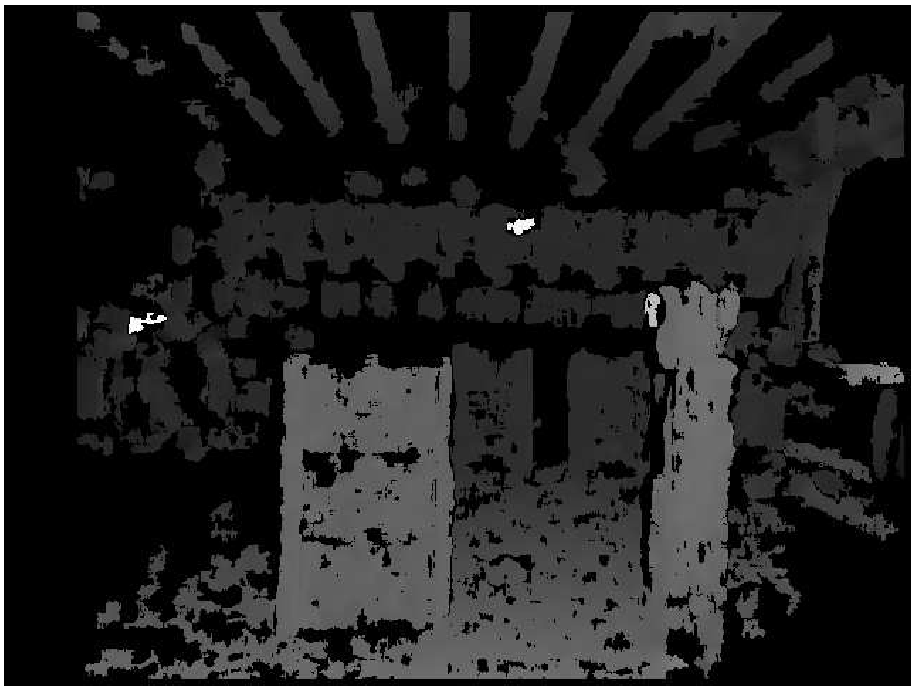


Figure 3.5: Disparity map used for error verification.

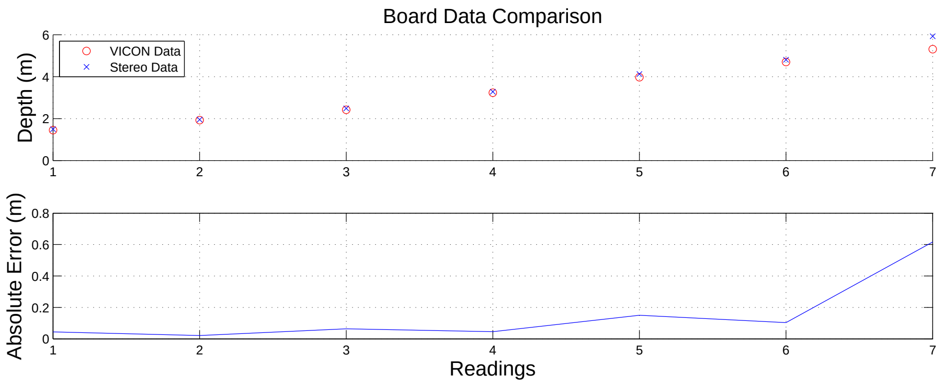


Figure 3.6: Depth data of “Board” obstacle and absolute error.

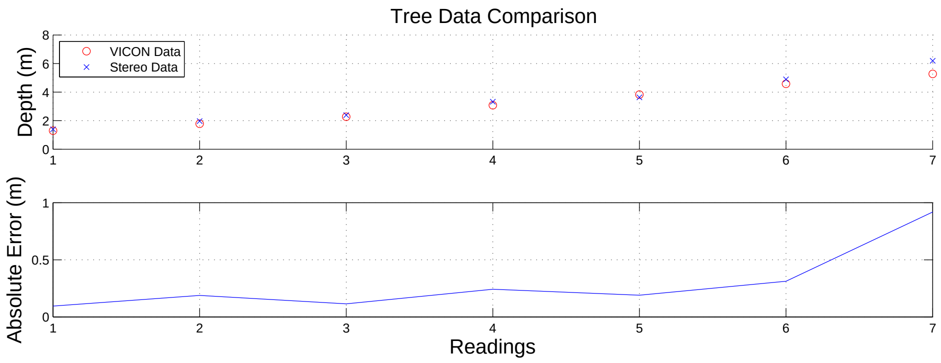


Figure 3.7: Depth data of “Tree” obstacle and absolute error.

It could be seen from the data that the error increases exponentially with distance. With a stereo camera that has a baseline of $12\ \mathrm{cm}$, it could be seen that the efficient working range of the stereo cameras is only within $5\ \mathrm{m}$. Within the working range, the absolute error for distances calculated still ranges between $10\ \mathrm{cm}$ to $20\ \mathrm{cm}$. One important observation to note is that for any features detected out of the $5\ \mathrm{m}$ working range, there will be an error of more than $0.5\ \mathrm{m}$. These data could only be used as an approximation and is not recommended for use in more intricate algorithms.

3.3 Stereo Vision Depth Map Estimation

Stereo vision allows the use of rectified images to calculate a disparity map by applying the Sum of Absolute Difference (SAD) [57] window to search for matches throughout. This generates a dense disparity map which we could use to obtain a dense depth map. A depth map is useful in many applications such as 3D shape matching [58], 3D model learning [59] and also in our case, obstacle detection. For most cases of obstacle detection, it will be advisable to have a generic 3D model of the obstacle such that the obstacle could be described clearly in 3D space.

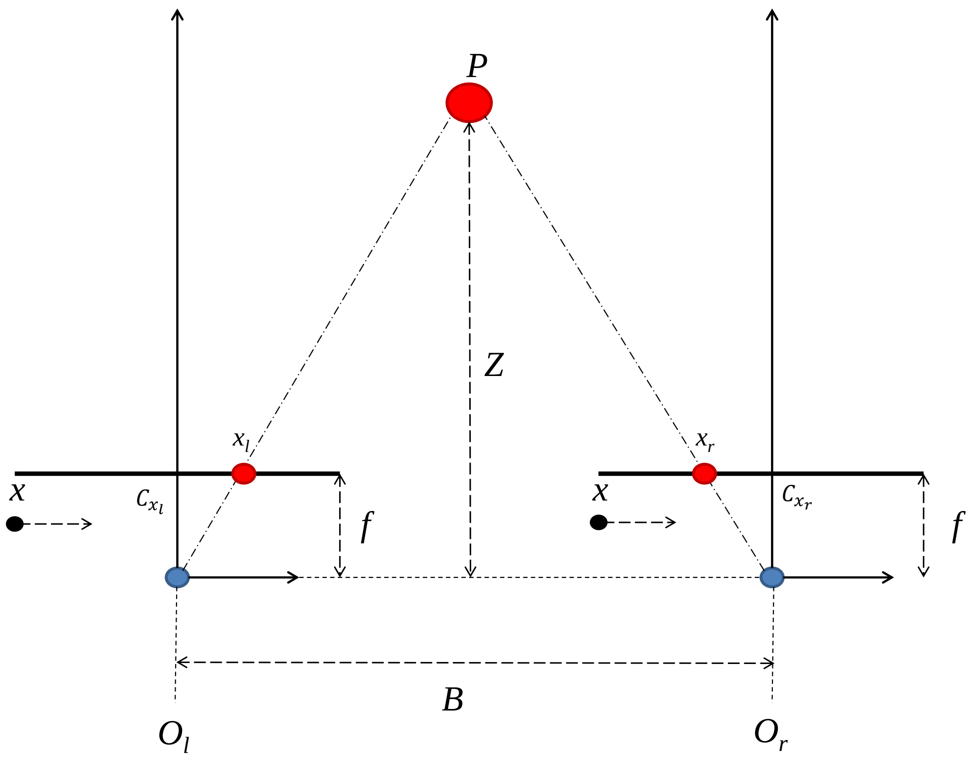


Figure 3.8: Calibrated and rectified stereo vision setup.

Eqn. 3.9 describes the basic relationship between distance of a feature with respect to its disparity. It is the basis of the generation of a 3D reconstruction or depth map. Assuming that through accurate calibration of the cameras and the rectification of the images depicted in Section 3.2, we will arrive at the setup shown in Fig. 3.8. The figure shows a point, P observed by both the left and right cameras and have their corresponding pixel values of x_l and x_r respectively. With a rectified image pair, the point will be observed in the same horizontal axis in both images. The difference between both pixel values will be the disparity value, $d = x_l - x_r$.

The depth Z could be found by similar triangles

$$\frac{B - (x_l - x_r)}{Z - f} = \frac{B}{Z} \tag{3.8}$$

$$Z = \frac{fB}{x_l - x_r} \tag{3.9}$$

where f is the focal length of the camera, B is the baseline distance of the stereo rig and $(x_l - x_r)$ is the disparity calculated.

Stereo vision also allows us to be able to calculate points in three dimension. This is basically done by using the points' screen coordinates and the camera intrinsic matrix to reproject the points into 3D space. The reprojection matrix is:

$$Q = \begin{pmatrix} 1 & 0 & 0 & -c_{x_l} \\ 0 & 1 & 0 & -c_{y_l} \\ 0 & 0 & 0 & f \\ 0 & 0 & -1/B & (c_{x_l} - c_{x_r})/B \end{pmatrix} \tag{3.10}$$

Given a 2D homogeneous point and its associated disparity, d, we can project the point into three dimensions using Eqn. 3.11.

$$Q \cdot \begin{pmatrix} x \\ y \\ d \\ 1 \end{pmatrix} = \begin{pmatrix} X \\ Y \\ Z \\ W \end{pmatrix} \tag{3.11}$$

The 3D coordinates are then calculated by dividing them by the weight, $W : (X/W, Y/W, Z/W)$.

With these set of 3D coordinates, we can generate a point cloud for every pair of stereo calibrated images.

3.4 Monocular Vision Depth Estimation

In order to detect and avoid obstacles during flight, a depth-based obstacle detection algorithm could also be used. Conventionally, computer vision techniques such as pattern recognition are used for detection of objects and obstacles as mentioned in Section 3.7. However, in many scenarios, the environment consist of obstacles which are unknown. It is impossible to "pre-train" the UAV to "recognize" all obstacles. Therefore, a depth-based 3D vision obstacle detection method is used to compute the depth of the scene in front of the UAV. If the depth of a certain part of the scene is less than a prescribed threshold, that portion is classified as an obstacle and obstacle avoidance procedure is activated.

The main steps in the algorithm are feature extraction, feature matching, and depth estimation. The reason to perform feature extraction and matching is to obtain the feature position on the image, and to calculate the feature velocity on the image. The details of the depth estimation will be given in the next section. The idea of depth estimation is structure from motion [60]. More precisely, if the state of the UAV can be measured and the image of a 3D point can be matched between two images, the 3D position of the point can be determined. In obstacle avoidance, we are more interested in the distance between the obstacle and the UAV, we focus on how to estimate the depth of the 3D point to achieve this. The state (i.e., position, velocity and attitude) of the UAV can be measured by GPS and inertial sensors.

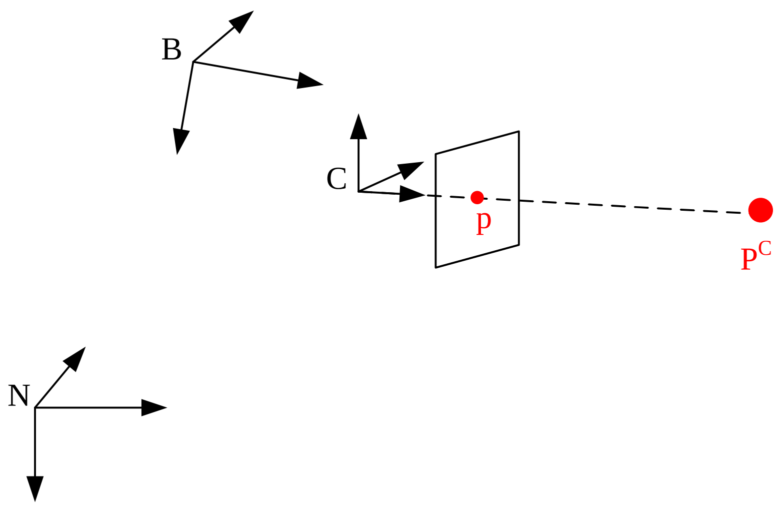


Figure 3.9: Reference frames involved in depth estimation. N: NED frame, B: body frame, C: camera frame.

3.4.1 Monocular Depth Estimation Algorithm Design

The coordinate frames involved in depth estimation algorithm are given in Fig. 3.9. The notation used are also given below.

- $P^C = [X, Y, Z]^T$: the coordinates of a 3D point P expressed in the camera frame. This is the unknown variable we are going to estimate.
- $\omega^C = [\omega_x^C, \omega_y^C, \omega_z^C]^T$: the angular velocity of the camera frame relative to the NED frame, with coordinates expressed in the camera frame.
- $v^C = [v_x^C, v_y^C, v_z^C]^T$: the linear velocity of the camera frame relative to the NED frame, with coordinates expressed in the camera frame.
- $p = [x, y]^T$: the image feature point of the 3D point on the ideal image plane ($f = 1$).

The angular rate ω^B and velocity v^B of the UAV resolved in the body frame can be obtained from GPS and inertial sensors. Then $\omega^C = R_B^C \omega^B$, $v^C = R_B^C v^B$, where R_B^C is the rotation transformation from the body frame to the camera frame. The feature point position p can be obtained from feature detection algorithms. The feature velocity on the image, $\dot{p}$, can be computed based on optical flow.

The following depth estimation algorithm is valid only for static scenes without any dynamic objects, i.e., $\dot{P}^N = 0$.

From

$$P^C = R_N^C(P^N - P_{C_0}^N) \tag{3.12}$$

and

$$\dot{R}_N^C = -[\omega^C]_\times R_N^C, \tag{3.13}$$

we have

$$\begin{aligned}\dot{P}^C &= \dot{R}_N^C P^N - \dot{R}_N^C P_{C_0}^N - R_N^C \dot{P}_{C_0}^N \\ &= -[\omega^C]_\times R_N^C P^N + [\omega^C]_\times R_N^C P_{C_0}^N - v^C \\ &= -[\omega^C]_\times P^C - v^C. \end{aligned} \tag{3.14}$$

The operator $[*]_\times$ converts a 3×1 vector to the associated 3×3 skew-symmetric matrix. Expanding the above equation into components gives

$$\begin{aligned}\dot{X} &= -v_x^C - \omega_y^C Z + \omega_z^C Y \\ \dot{Y} &= -v_y^C - \omega_z^C X + \omega_x^C Z \\ \dot{Z} &= -v_z^C - \omega_x^C Y + \omega_y^C X. \end{aligned} \tag{3.15}$$

Substituting

$$x = X/Z \tag{3.16}$$

$$y = Y/Z \tag{3.17}$$

into Eqn. (3.15) yields

$$\dot{Z} = -v_z^C - \omega_x^C yZ + \omega_y^C xZ, \tag{3.18}$$

which is equivalent to

$$\dot{Z} = (\omega_y^C x - \omega_x^C y)Z - v_z^C \tag{3.19}$$

On the other hand, we have

$$\dot{x} = (X/Z)' \tag{3.20}$$

$$\dot{y} = (Y/Z)' \tag{3.21}$$

Substituting Eqn. (3.15) into the above equation gives

$$\begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = \frac{1}{Z} \begin{pmatrix} -1 & 0 & x \\ 0 & -1 & y \end{pmatrix} \begin{pmatrix} v_x^C \\ v_y^C \\ v_z^C \end{pmatrix} + \begin{pmatrix} xy & -(1+x^2) & y \\ 1+y^2 & -xy & -x \end{pmatrix} \begin{pmatrix} \omega_x^C \\ \omega_y^C \\ \omega_z^C \end{pmatrix} \tag{3.22}$$

The above equation can be rewritten as

$$\underbrace{\begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} - \begin{pmatrix} xy & -(1+x^2) & y \\ 1+y^2 & -xy & -x \end{pmatrix} \begin{pmatrix} \omega_x^C \\ \omega_y^C \\ \omega_z^C \end{pmatrix}}_{b} = \underbrace{\begin{pmatrix} -1 & 0 & x \\ 0 & -1 & y \end{pmatrix} \begin{pmatrix} v_x^C \\ v_y^C \\ v_z^C \end{pmatrix}}_{A} \frac{1}{Z} \tag{3.23}$$

which is in the form:

$$b = A\frac{1}{Z}. \tag{3.24}$$

So the depth Z can be computed as

$$Z_{\text{measure}} = \frac{1}{(A^T A)^{-1} A^T b}. \tag{3.25}$$

To summarize, we have

$$\dot{Z} = (\omega_y^C x - \omega_x^C y)Z - v_z^C + \varepsilon_1 \tag{3.26}$$

$$Z_{\text{measure}} = Z + \varepsilon_2. \tag{3.27}$$

The Eqn. 3.26 is the process model of the depth Z. ω^C and v^C can be measured by GPS and inertial sensors; x and y can be obtained from feature extraction. Eqn. 3.27 is the measurement model of the depth Z. This measurement actually is a closed-form estimation of the depth. However, since there are measurement noises in x, y ,$\dot{x}$, $\dot{y}$, ω^C and v^C, the closed-from estimation is corrupted by noises. So an optimal approach is to combine the process model and measurement model together then apply the extended Kalman filter (EKF) [61] .

3.4.2 Monocular Depth Estimation Simulation Results

Simulations were carried out with aerial photos taken from Google Maps and they were arranged into different scenarios to verify the proposed monocular depth estimation approach. In our simulations, the depth of the scene is displayed as yellow numbers. Should the depth of the scene be very far from the UAV and falls out of our obstacle threshold, the value of 10000 will appear. This is an arbitrary value and is present to display that the particular location of the scene is free of obstacles.

A few scenarios are depicted as follows:

1. **Scenario 1** Assume that the UAV flies along a level straight line. The attitude including heading angle and velocity are constant. The altitude of the UAV is fixed to $20\ \mathrm{m}$.

2. **Scenario 2** The UAV is performing a landing task. The attitude and velocity are constant. But the altitude decreases continuously.

The depth estimation for Scenario 1 is given in Fig. 3.10. It can be seen that the depth estimation is very accurate. The plan view images are purely simulated according to the scenarios shown above and the algorithm is able to obtain accurate results. The errors obtained were less than $2\ \mathrm{m}$ which relates to a percentage error of less than 10%. In order to make the depth estimation more robust, we divide the image into 4 by 3 grids. The average depth of each grid is given by yellow numbers. The red number is the depth estimation for each feature point. The red arrow indicates the optical flow of each feature. Although the depth estimation is more robust across the whole image, there are times when a grid has very few feature points which result in erroneous feature matching and thus causes spikes in error for up to 20%.

Fig. 3.11 illustrates how to use depth estimation to detect obstacles. In the lower two rows of grids in Fig. 3.11, the depth are $20\mathrm{m}$. But the depth of the upper two rows of grids are very large. Then we can claim that obstacles exist in the lower two rows of grids.

In the second scenario, the UAV was performing a gradual landing task. From Fig. 3.12 to Fig. 3.13, we can see the height of the UAV reducing as it gradually descends with time. Overall, the depth can be estimated relatively accurately.

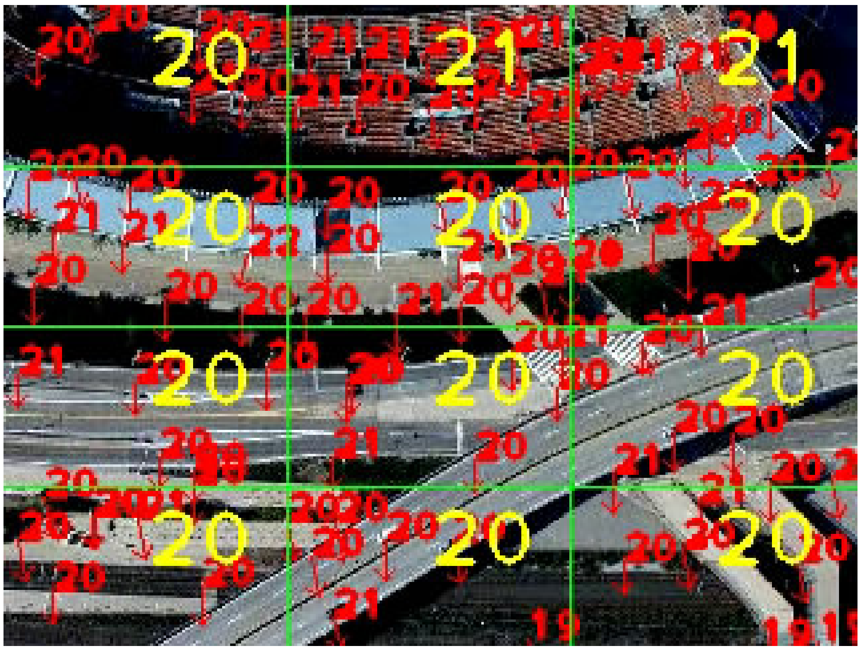


Figure 3.10: Depth estimation in scenario 1.

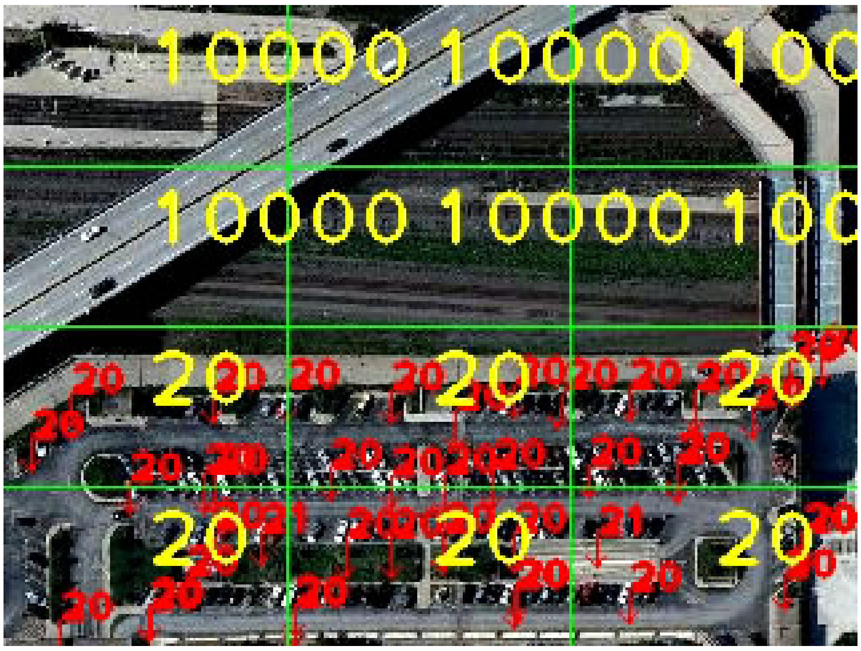


Figure 3.11: Illustration on how to classify obstacles.

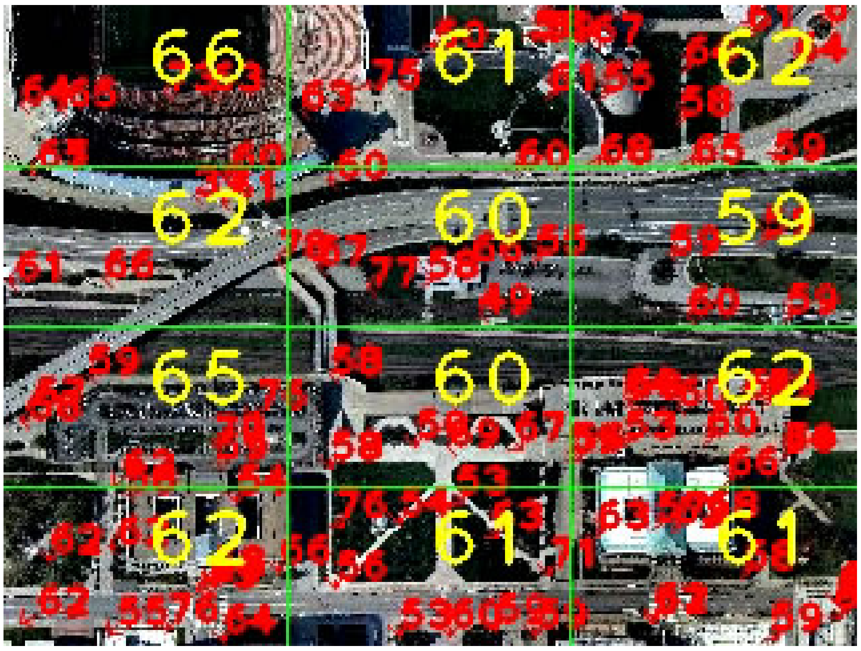


Figure 3.12: Early frame captures of depth estimation in Scenario 2.

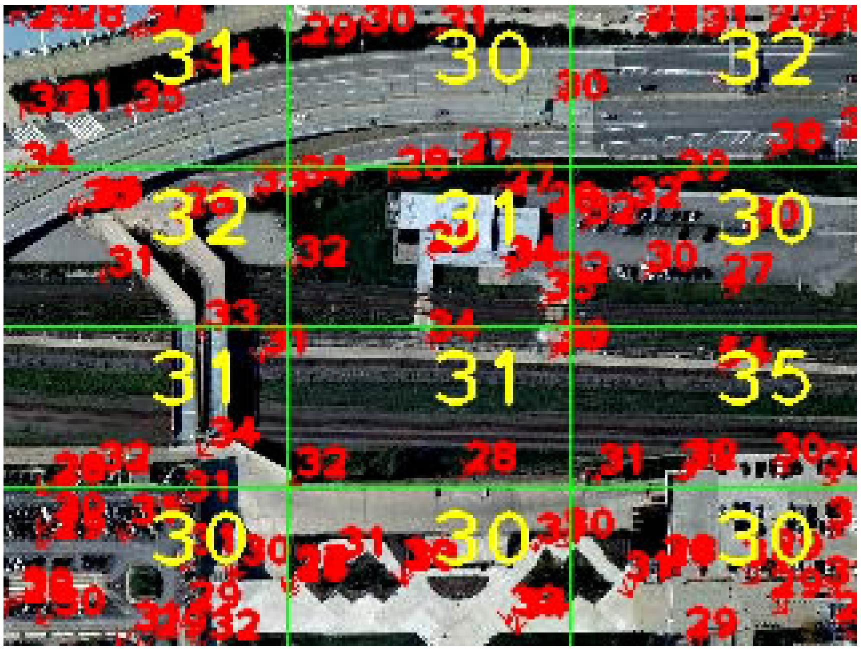


Figure 3.13: Near landing frame captures of depth estimation in Scenario 2.

This method assumes the state of the UAV can be measured by GPS and inertial sensors which we assume to be accurate here. The scene is also assumed to be static. Then an EKF based depth estimation method is proposed to obtain a stable depth estimate. Simulations show that the method can give accurate depth estimation. It is notable that the depth estimation inherently is a structure from motion technique.

3.5 Stereo Vision General Obstacle Detection

With the implementation of Section 3.3, we are able to obtain 3D point clouds from every image pair captured by the stereo camera. There are many uses for a 3D point cloud. One such use is to obtain the motion estimation of the camera using algorithms such as Absolute Orientation Calculation [62] where frame to frame motion is estimated by matching the features between frames and obtaining the rotation and translation vectors of the camera which induce such a change in relative position. This method for navigation will be covered in Chapter 5. Another important use of a 3D point cloud is to detect obstacles which we will describe here.

In a cluttered indoor environment, obstacles exists in the form of tables, chairs and even human beings. There is no fixed form for an obstacle, therefore, the algorithm we created for obstacle detection is a generic one where any form of obstacles which are near to the camera will be detected and segmented. This technique is robust to inaccurate UAV state readings and is also able to detect obstacles that are dynamic which is very different from the algorithm discussed in Section 3.4. The basic steps to our algorithm are shown below:

General Obstacle Detection Algorithm

1. **Stereo Disparity Map:** Using the disparity map generated from Section 3.3, a 3D Point cloud could be obtained by using the disparity calculated and reprojecting it into world coordinate using Eqn. 3.11. Fig. 3.14 shows a typical cluttered indoor environment and its corresponding disparity map.

2. **Voxel-Grid Filter:** Voxel-Grid filtering is used to down-sample the 3D point cloud into manageable data sets which are more computational efficient. The Voxel-Grid Filter sets a specific minimum resolution dimension in the created 3 dimension space and collapses all the 3D world points into that voxel space. Thereby reducing the resolution of the 3D point cloud and reducing the computational load required for real-time computation.

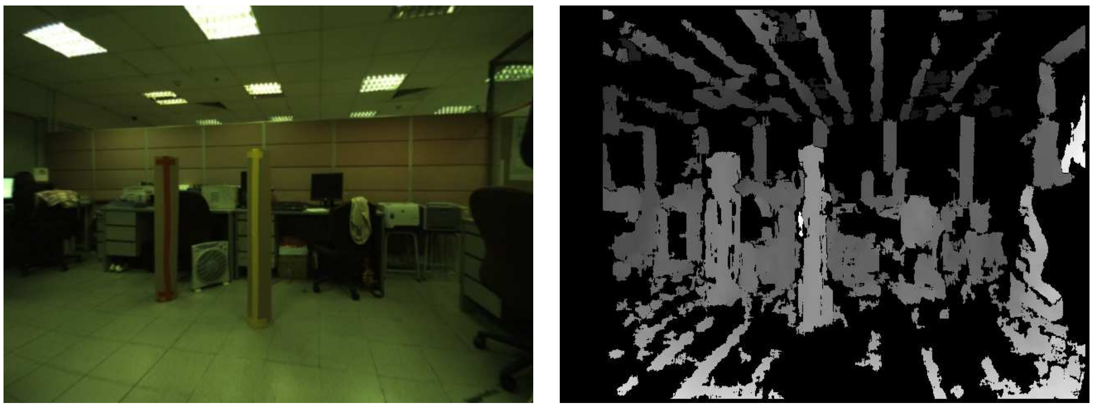


Figure 3.14: Left: Cluttered indoor environment, Right: Disparity map generated for indoor environment.

3. **Distance Thresholding:** In stereo vision, the error in 3D estimation of world point, P, increases with distance relative to the camera center as described in Section 3.2. In obstacle detection, the objects of interest are also relatively near to the camera. Therefore, we used a simple Z-direction filter to isolate the 3D points of interests that are within a fixed depth from the camera. This gives us more accurate readings and allow us to track obstacles nearer to the camera better.

4. **Contour Generation:** The 3D points are further clustered into groups based on the K-th Nearest Neighbour (KNN) [63] clustering algorithm and contours are created and drawn in the image for better visualization for the user. The use of contours helps to filter out noise which are created during the disparity map generation process. For general obstacle detection, one is usually only interested in obstacles that of critical size. Contour generation aims to help categorize the obstacles observed in a scene into different sizes.

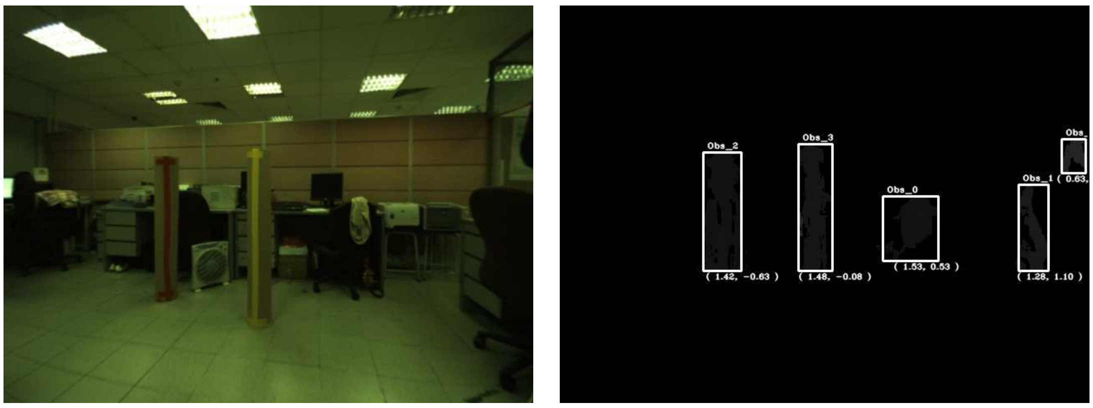


Figure 3.15: Left: Cluttered indoor environment, Right: Obstacle classification with rectangular bounding box.

5. **Obstacle Classification:** Obstacles are classified by creating a bounding box around the contour detected as mentioned above. In our classification, we create a 3D cylindrical block around the detected obstacle and output the center position of the obstacle with an accompanying radius and height which corresponds to the cylindrical shaped obstacle. The UAV can then plan its path around this cylindrical obstacle as a safety distance could be implemented into the classification of the obstacle. Fig. 3.15 shows the successful implementation of this obstacle classification scheme.

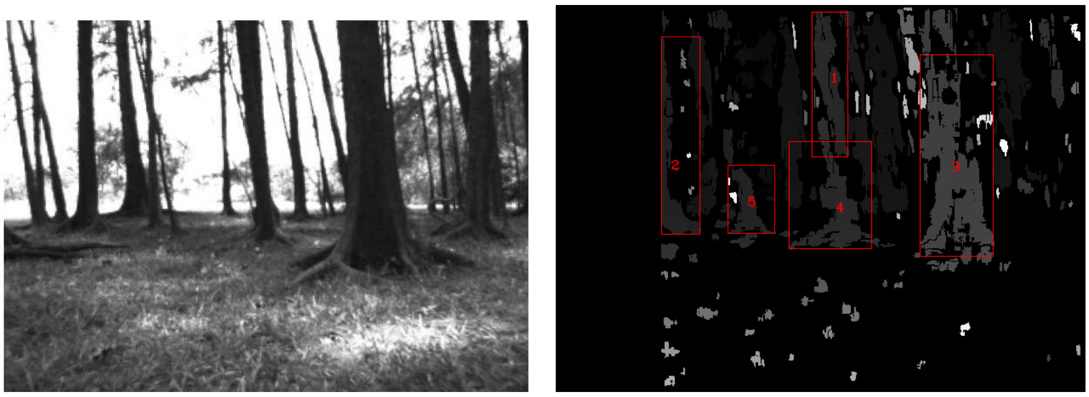


Figure 3.16: Left: Outdoor forested environment, Right: Tree obstacles detected in forested environment.

We have also implemented this obstacle detection scheme in a real-life outdoor environment such as that of a forested environment as shown in Fig. 3.16. We are able to successfully segment the obstacles in such an environment and classify the obstacles in our above mentioned classification scheme. The obstacle detection capabilities of our algorithm in a real-life outdoor environment are observed to be robust and accurate.

3.5.1 Stereo Depth Map Generation

In OpenCV [64], there are two main algorithms used for the calculation of dense disparity maps in stereo vision applications: Block Matching (BM) [65] and Semi-Global Block Matching (S-GBM) [66] methods. They basically try to find stereo correspondences between two rectified images using Sum of Absolute Difference (SAD) windows. The Block Matching method is a very fast algorithm but it will miss a lot of details if the image has a lot of areas with homogeneous texture. The Semi-Global Block Matching method is relatively slower than Block Matching as it performs a global smoothing to the image and extrapolates correspondences in areas with features are not very pronounce.

Fig. 3.17 to Fig. 3.19 shows the raw image taken from the left camera and the stereo disparity maps generated using both algorithms.

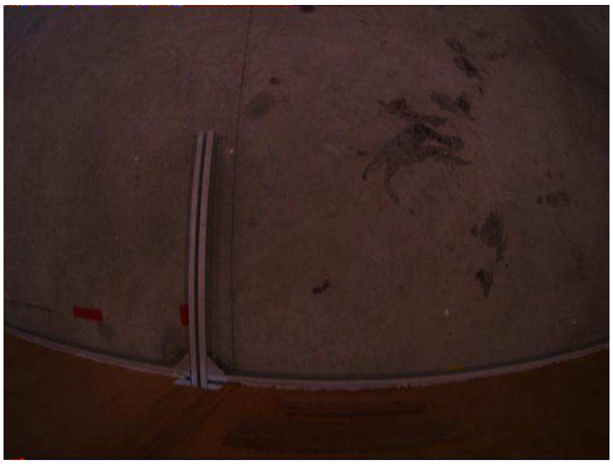


Figure 3.17: Unrectified image take with left camera.

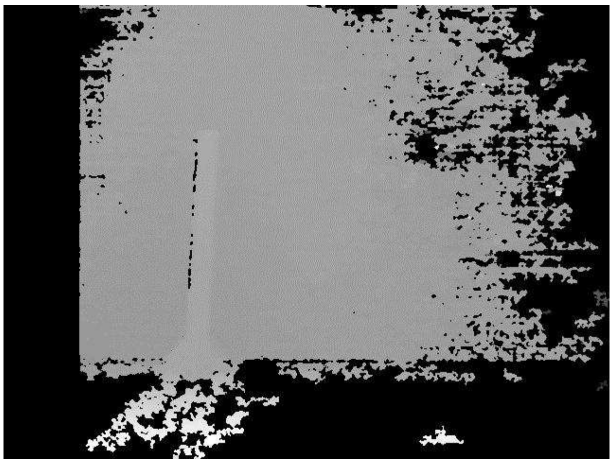


Figure 3.18: Stereo disparity map obtained using SGBM.

After the calculation of disparity maps by the above mentioned methods, the disparity values could be used to reproject the pixel coordinates of the features and their corresponding disparity values into the 3D world coordinate with the Eqn. 3.11.

Using another library which could process large point clouds called Point Cloud Library (PCL) [67], we could generate the point cloud visually and display it for users to manipulate. Fig. 3.20 and Fig. 3.21 shows the stereo image and point cloud generated from it.

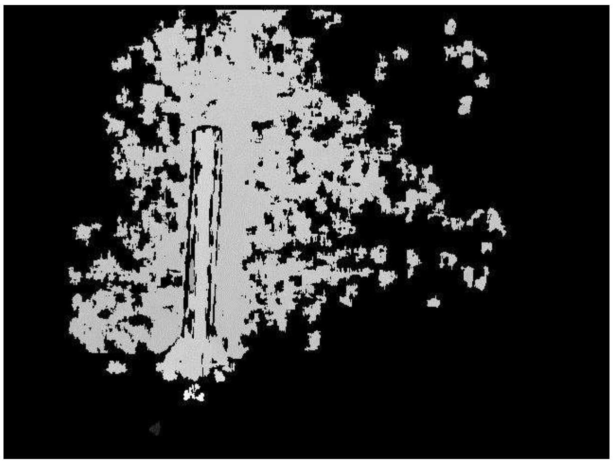


Figure 3.19: Stereo disparity map obtained using BM.

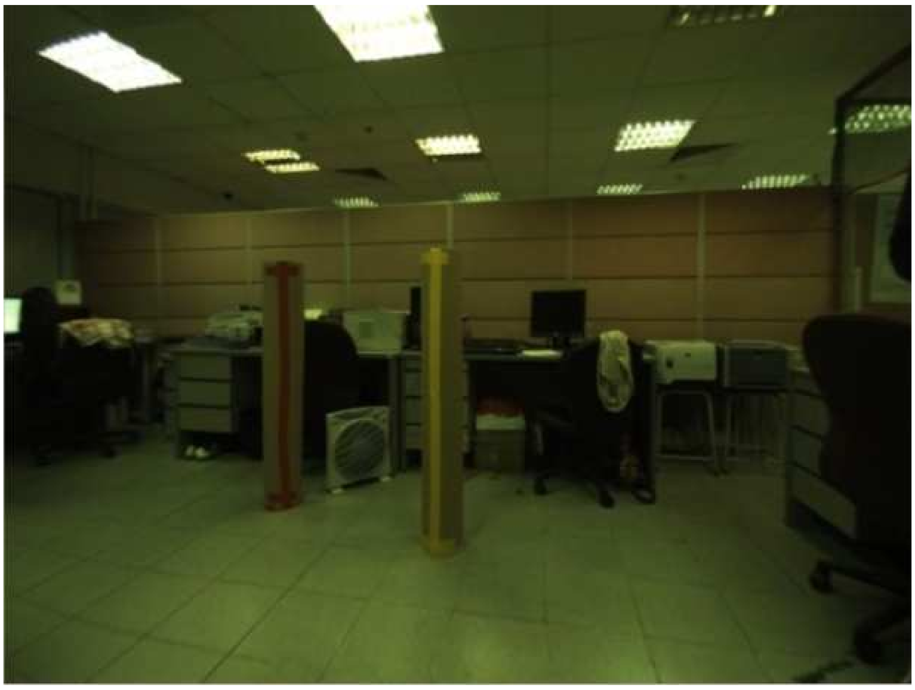


Figure 3.20: Left image of stereo pair used to generate point cloud.

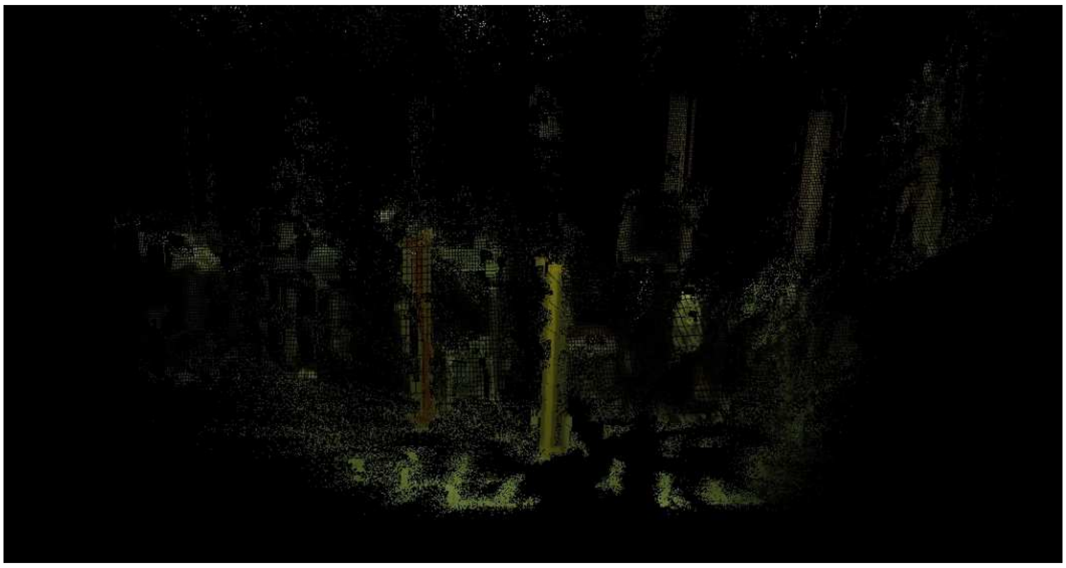


Figure 3.21: Point cloud calculated and displayed in PCL.

3.6 Power Line Detection

In urban environments there exist obstacles that are potentially hazardous for a low flying UAV. “Wire strikes” are the number one cause of fatal helicopter accidents. The operational safety of UAVs in their environment is frequently overlooked but this is essential for mission completion. Detection of power lines will be primarily achieved by using vision processing. Through the understanding of how power lines appear in images, we designed an algorithm which could create detection specifically for power lines. It is the implementation of a series of gradient mask operators, image inverse thresholding, line-thickness filtering and segmentation that will result in the detection of power lines in images.

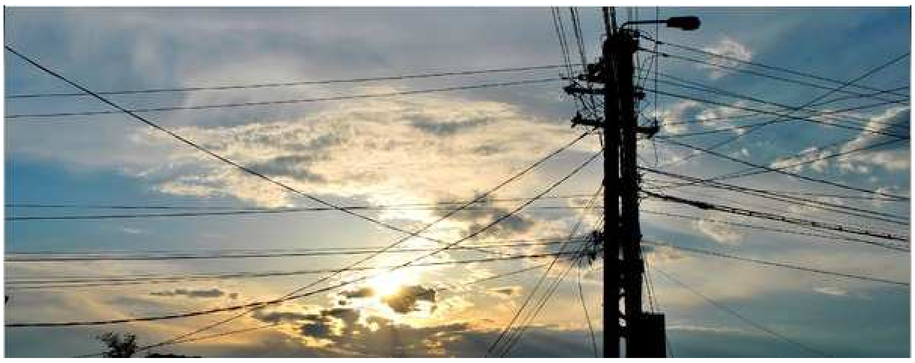


Figure 3.22: Powerline image with noisy background.

During the research into power line detection, we noticed that powerlines vary in appearance based on the different view perspectives. From a bird’s eye point of view, powerlines will appear as near parallel straight lines. Tradition methods include Hough transform [68] which could correct discontinuities in the power lines detected to present a clearer image of the powerlines and also form as a prediction on how far the powerlines might extend. However, this point of view is not the view perspective a UAV will observe should it be flying into the powerlines.

What we conclude was that powerlines will be observed from a frontal view much like that as shown in Fig. 3.22. We then investigated into the techniques used to segment the powerlines from the image which will be covered in the rest of the section.

3.6.1 Convolution Filtering

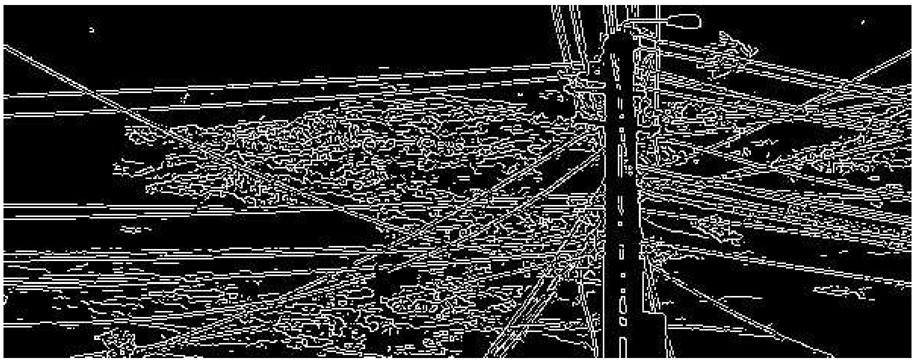


Figure 3.23: Canny edge detector used on power line detection.

One of the first steps to the detection of powerlines will be to carry out the detection of lines in an image. There are many forms of edge detectors with the Canny Edge Detector [69] being one of the most well known. We performed the canny edge detection algorithm on a sample powerline image as shown in Fig. 3.23 and found that if the background was very noisy, the edge detector will generate much more false detections and will not aid our algorithm in any way. Therefore, we decided to perform convolution filtering to customize a form of edge detection that was suitable to our needs.

Convolution filtering is a neighbourhood-based operation in which the output image is obtained by convolving the input with a kernel with a window size of $m \times n$. This kernel is centered at each pixel under consideration and shifts from the top left pixel to that of the bottom right pixel of the image through iteration. The convolution of each pixel is given by:

$$g(x, y) = \sum_{u=-m}^{m} \sum_{v=-n}^{n} f(x + u, y + v)w(u, v) \tag{3.28}$$

In our application, we will like to perform specific edge detection that was robust to noise.

This meant that the kernel should be able to be configured for detection of edges either in the x or y direction in images. The kernel should also be able to have a smoothing effect for noise and be tunable for powerlines of different thickness. One very suitable kernel for this application was the Sobel gradient operator [70]. The Sobel gradient operator is a well-known, non-linear edge enhancement operator that could perform gradient detection in images which enhance edges as well as perform smoothing operations in the process. The Sobel operator computes image gradients over a 3×3 area and has the advantage of increasing the averaging of the sampled image area. This creates the smoothing effect that is less sensitive to noise. Eqn. 3.29 and Eqn. 3.30 describe the Sobel kernel used to detect horizontal and vertical edges respectively.

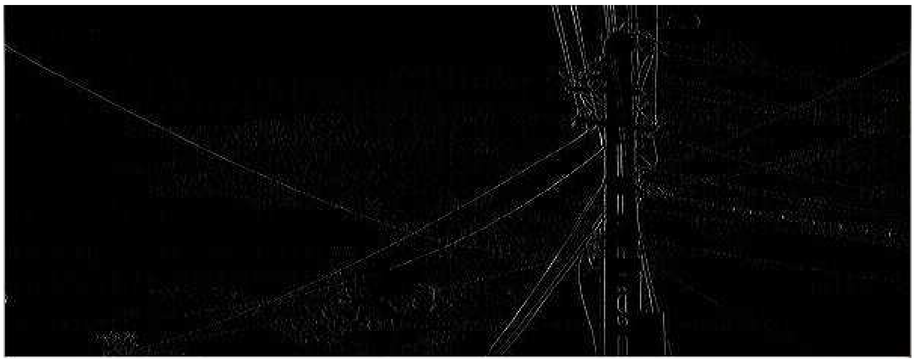


Figure 3.24: Sobel kernel filtering in δy.

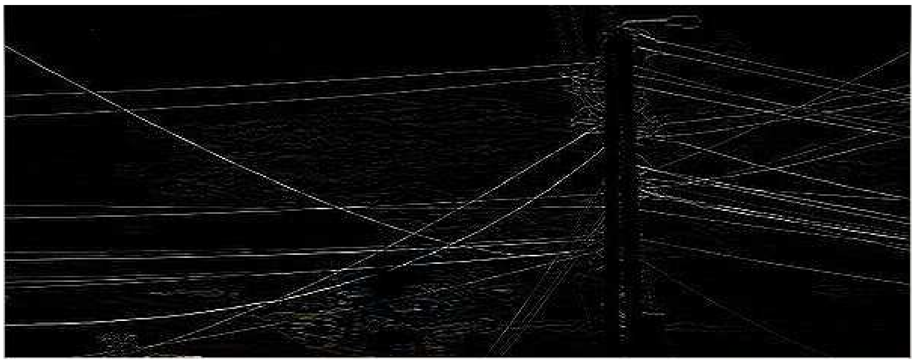


Figure 3.25: Sobel kernel filtering in δx.

$$\frac{\delta f}{\delta x} = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \tag{3.29}$$

$$\frac{\delta f}{\delta y} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \tag{3.30}$$

In Fig. 3.24 and Fig. 3.25, we have applied the Sobel gradient kernel to great success when compared with the Canny edge detector. One can observe that the filtering has removed most of the noise created by the background, leaving behind only the powerlines of interest. The result from Fig. 3.25 is of particular interest in this topic of powerline detection and we use it to progress further in our algorithm.

3.6.2 Line Segment Detector

Upon obtaining a filtered edge map from the above section, we will like to obtain the coordinates of these edges and also to determine if these edges form lines that could possibly represent powerlines. One particular parameter for powerline detection is the connectivity of the lines as well as the curvature in which the lines represent. From inspection of powerline images, we can see that powerlines usually form quadratic curves of varying curvatures. We will want to obtain these points that describe the quadratic curves prior to applying curve fitting algorithms covered in Section 3.6.3.

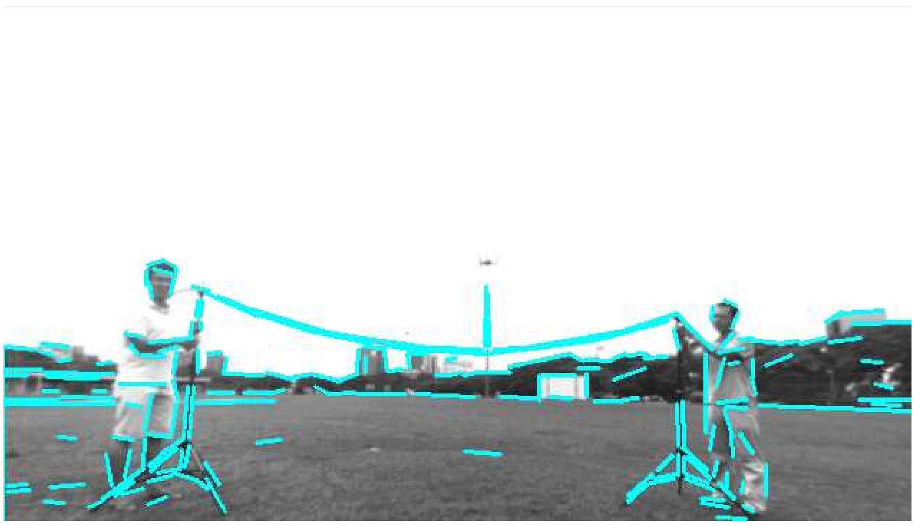


Figure 3.26: Line segment detector applied to powerline image.

Since a quadratic curve can be seen as many segments of straight lines, we apply a line segment detector that could find these straight lines. LSD is a linear-time Line Segment Detector [71] which gives subpixel accurate results without any prior tuning that could be applied to any form of image. It controls the number of false alarms to an average of only one per image and is based on the Burns, Hanson, and Risemans method [72]. LSD is aimed at detecting locally straight contours on images which are then defined as line segments.

We obtained real images of powerlines in an outdoor environment and applied both the convolution filtering as describe in Section 3.6.1 and LSD to the image. From Fig. 3.26, it can be seen that the line segments were detected clearly and the powerline was detected as well. Each line segment obtained includes the image coordinates of the start and end of the segment. We then used this information to further filter out lines that were either too short or that they were in a near vertical position. Thereby allowing line segments of only a specific gradient to be obtained from our algorithm.

3.6.3 Least Squares Quadratic Curve Fitting

From the above sections, we observe that powerlines exhibit a straight line to a curve characteristic usually in a noisy background. The curvature is determined by how much slack or tension the powerlines are experiencing. Therefore, we performed a least squares quadratic curve fitting to obtain the powerline curve accurately. A quadratic curve equation is shown below:

$$y = b_0 + b_1 x + b_2 x^2 \tag{3.31}$$

To achieve the best fit of the quadratic curve with unknown parameters b_0, b_1 and b_2 shown in Eqn. 3.31, we will have to reduce the measure of the error between the data and the fit curve through the simple generalization of the linear error function, Eqn. 3.32. This minimum error is obtained when all the partial derivatives of the error function with respect to the unknown parameters are all zero.

$$Error = \epsilon(b_0, b_1, b_2) = \sum_{i=1}^{n} \left(b_0 + b_1 x + b_1 x^2 - y\right)^2 \tag{3.32}$$

The equation that we arrive at from evaluating the partial derivative with respect to b_0 is,

$$\frac{\partial \epsilon}{\partial b_0} = \sum_{i=1}^{n} 2\left(b_0 + b_1 x_i + b_2 x_i^2 - y_i\right) \tag{3.33}$$

$$\frac{\partial \epsilon}{\partial b_0} = 2\left[n b_0 + b_1 \sum_{i=1}^{n} x_i + b_2 \sum_{i=1}^{n} x_i^2 - \sum_{i=1}^{n} y_i\right] = 0 \tag{3.34}$$

Dividing both sides and rearranging the equation results in the form,

$$n b_0 + \left[\sum_{i=1}^{n} x_i\right] b_1 + \left[\sum_{i=1}^{n} x_i^2\right] b_2 = \sum_{i=1}^{n} y_i \tag{3.35}$$

Similarly, the partial derivatives with respect to b_1 and b_2 and equating to zero are shown respectively.

$$\left[\sum_{i=1}^{n} x_i\right] b_0 + \left[\sum_{i=1}^{n} x_i^2\right] b_1 + \left[\sum_{i=1}^{n} x_i^3\right] b_2 = \sum_{i=1}^{n} y_i x_i \tag{3.36}$$

$$\left[\sum_{i=1}^{n} x_i^2\right] b_0 + \left[\sum_{i=1}^{n} x_i^3\right] b_1 + \left[\sum_{i=1}^{n} x_i^4\right] b_2 = \sum_{i=1}^{n} y_i x_i^2 \tag{3.37}$$

Therefore, by using the linear algebra notation,

$$\begin{bmatrix} n & \sum_{i=1}^{n} x_i & \sum_{i=1}^{n} x_i^2 \\ \sum_{i=1}^{n} x_i & \sum_{i=1}^{n} x_i^2 & \sum_{i=1}^{n} x_i^3 \\ \sum_{i=1}^{n} x_i^2 & \sum_{i=1}^{n} x_i^3 & \sum_{i=1}^{n} x_i^4 \end{bmatrix} \cdot \begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} y_i \\ \sum_{i=1}^{n} y_i x_i \\ \sum_{i=1}^{n} y_i x_i^2 \end{bmatrix} \tag{3.38}$$

Simplifying the notations, we obtain:

$$X \cdot \vec{b} = y \tag{3.39}$$

By calculating the transpose of the X matrix and moving the terms to the righthand side, I can get:

$$\vec{b} = \left(X^T X\right)^{-1} X^T y \tag{3.40}$$

where $\vec{b} = [b_0, b_1, b_2]^T$.

3.6.4 Powerline Detection Results

We conducted experiments with images of powerlines obtained in outdoor environments and captured the images using our self-built stereo camera featured in Section 3.1. In both images, we applied the above mentioned algorithms to isolate and detect powerlines prior to calculating the powerline's 3D coordinates.

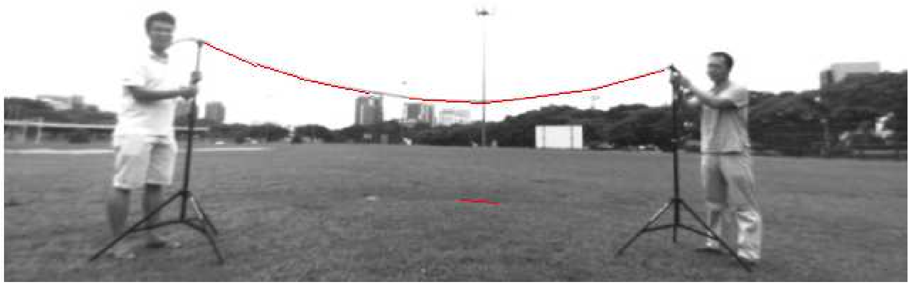


Figure 3.27: Powerline segmented from image.

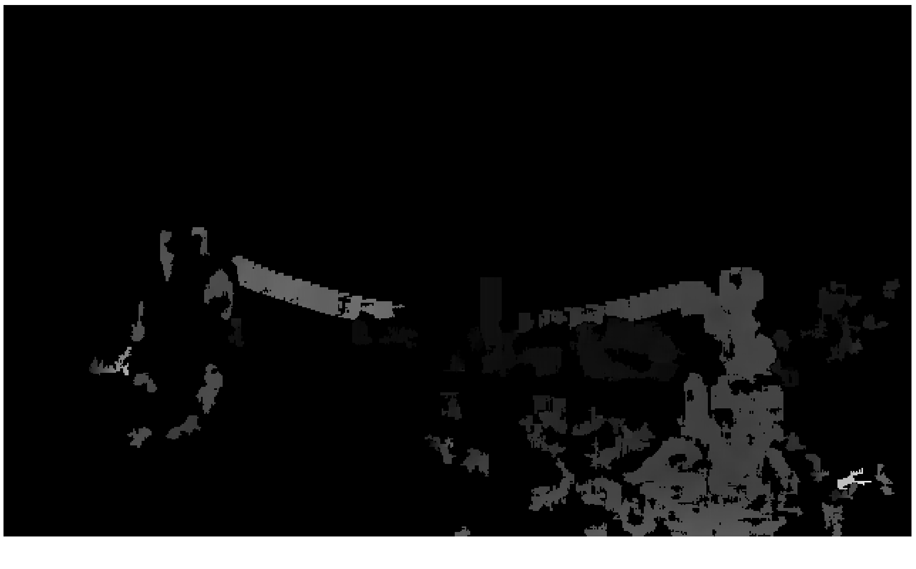


Figure 3.28: Disparity found in powerline image.

From Fig. 3.27 and Fig. 3.28 you can see that we have successfully detected the powerline in our image. Not only did we detect the powerline in the image, we could also obtain a disparity image of detected powerline to gauge its exact 3D coordinates which could then be used for performing obstacle avoidance.

However there are still cases where the algorithm could not perform. For instance, if the powerline exist in a background that is very complicated or filled with objects such as that shown in Fig. 3.29, we will not be able to detect the powerline specifically. However, we can still employ the general obstacle detection as explained in Section 3.5.

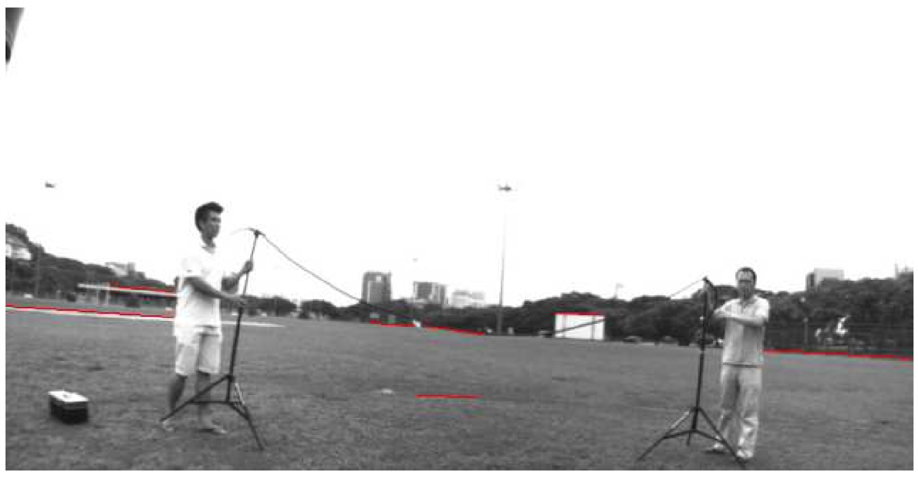


Figure 3.29: Complicated background restricts the detection of powerlines.

3.7 Vision-based Obstacle Tracking

In an urban environment, there are obstacles that are known beforehand and frequently appear such as trees, lamp-posts and cars. A strategy to apply this knowledge into a more narrow-form of obstacle avoidance with the implementation of monocular camera obstacle avoidance could be done. In the application of this type of obstacle avoidance, the user can mark the targets that are required to be avoided and train the onboard computer to avoid these types of obstacles should it appear again. This is done so by using the image captured by the onboard camera and drawing a rectangular target box around the required target. Next, the selected target box is extracted from the image and training of the target is implemented using the Continuously Adaptive Mean-SHIFT, CAMSHIFT [73]. Once trained, the UAV will be able to identify and avoid similar obstacles. This algorithm could also be applied automatically after the detection of obstacles highlighted in Section 3.5 and is efficient enough to run on very low powered processors and still be achievable in real-time.

3.7.1 CamShift Algorithm

The Mean Shift Algorithm [74] is a non-parametric feature-space analysis technique used frequently for clustering in computer vision applications. The Mean Shift algorithm is a procedure used for locating the maxima of a density function. It is an iterative method that is useful for detecting the modes of the density. In our application, it is used to detect the centroid of our selected target's color. It is the precursor to the algorithm which we will use and therefore is important for us to understand the strategy it uses to track targets or in our case, obstacles.

Mean Shift Algorithm Steps:

1. Choose a search window size.
2. Choose the initial location of the search window.
3. Compute the mean weighted center of mass location in the search window.
4. Center the search window at the center of mass location computed in Step 3.
5. Repeat Steps 3 and 4 until the search window center converges. i.e. until it has moved a distance less than the pre-set threshold.

CamShift stands for the "Continuously Adaptive Mean-SHIFT" algorithm. A flow chart is shown in Fig. 3.30 that summarizes this algorithm. For each video frame, the raw image is converted to a color probability distribution image via a color histogram model of the color being tracked, e.g., skin color in the case of face tracking. The center and size of the color object are found via the CamShift algorithm operating on the color probability image. The current size and location of the tracked object are reported and used to set the size and location of the search window in the next video image. The process is then repeated for continuous tracking. The algorithm is a generalization of the Mean Shift algorithm, highlighted in the flow chart.

CamShift operates on a 2D color probability distribution image produced from the histogram back projection. The core part of the CamShift algorithm is the Mean Shift algorithm.

CamShift Algorithm Steps:

1. Set the calculation region of the probability distribution from the whole image.
2. Choose the initial location of the 2D mean shift search window.

3. Calculate the color probability distribution in the 2D region centered at the search window location in a Region of Interest (ROI) slightly larger than the Mean Shift window size.

4. Run Mean Shift algorithm to find the search window center. Store the moment (area or size) and center location.

5. For the next image frame, center the search window at the mean location stored in Step 4 and set the window size to a function of the moment found there. Go to Step 3.

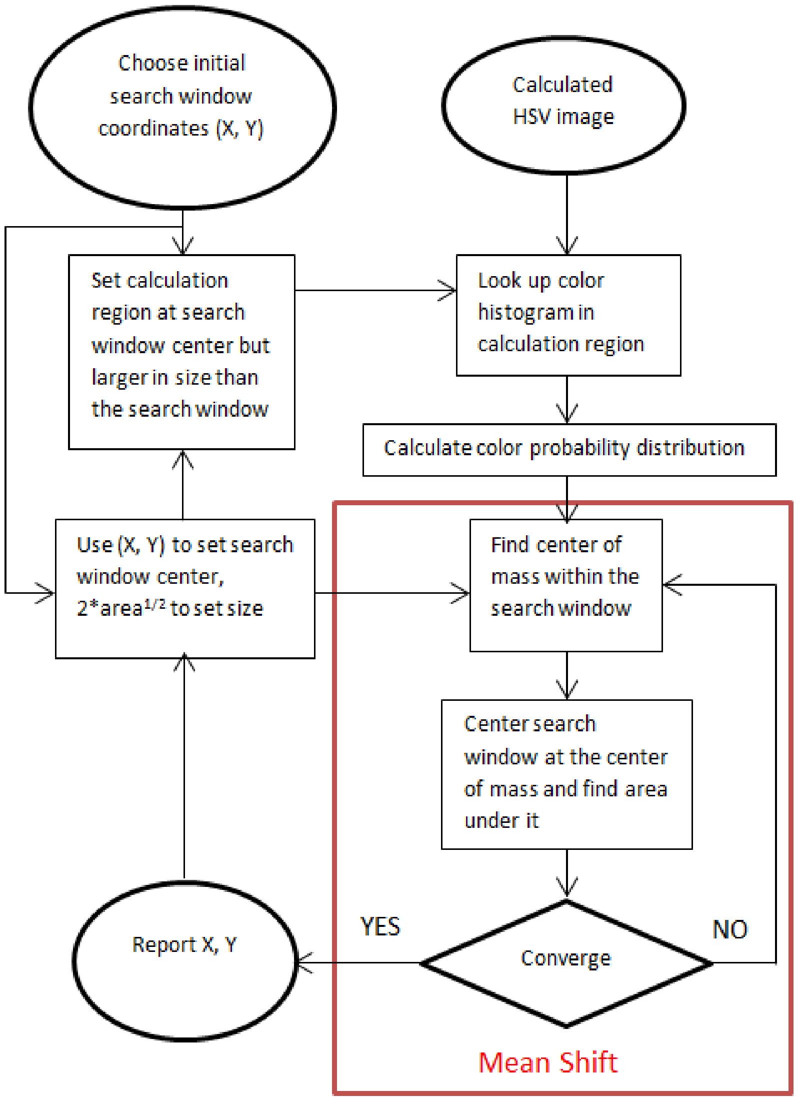

Choose initial search window coordinates (X, Y)
Calculated HSV image
Set calculation region at search window center but larger in size than the search window
Look up color histogram in calculation region
Calculate color probability distribution
Use (X, Y) to set search window center, 2*area$^{1/2}$ to set size
Find center of mass within the search window
Center search window at the center of mass and find area under it
YES
NO
Converge
Report X, Y
Mean Shift

Figure 3.30: CamShift algorithm flowchart.

Unlike the Mean Shift algorithm, which is designed for static distributions, CamShift is designed for dynamically changing distributions. In other words, CamShift is the use of Mean Shift calculation with an adaptive window size that finds the optimal rotation of the selected target. This occurs when objects in video sequences are being tracked and the object moves so that the size and location of the probability distribution changes in time. The CamShift algorithm adjusts the search window size in the course of its operation. Initial window size can be set at any reasonable value. Instead of a set, or externally adapted window size, CamShift relies on the moment information, extracted as part of the internal workings of the algorithm, to continuously adapt its window size within or over each image frame.

3.7.2 CamShift Formulation

In the CamShift algorithm, it relies on the tracking of the centroid of the intended target. Therefore, for the calculation of the centroid of the target in the search window, we will have to obtain the image's mean location for the probability distribution. For discrete 2D image probability distributions, the mean location (the centroid) within the search window is found as follows:

Finding the zero moment and the first moments for x and y,

$$M_{00} = \sum_x \sum_y I\left(x, y\right) \tag{3.41}$$

$$M_{10} = \sum_x \sum_y xI\left(x, y\right) \tag{3.42}$$

$$M_{01} = \sum_x \sum_y yI\left(x, y\right) \tag{3.43}$$

Search window location (the centroid) is then found as,

$$x_c = \frac{M_{10}}{M_{00}} \tag{3.44}$$

$$y_c = \frac{M_{01}}{M_{00}} \tag{3.45}$$

Where I is the pixel (probability) value in the position (x,y) in the image and x and y range over the search window.

The 2D orientation of the probability distribution is also easy to obtain by using the second moments in the course of CamShift operation where the point (x, y) ranges over the search window, and I is the pixel (probability) value at the point (x, y).

Second moments calculations are,

$$M_{20} = \sum_{x}\sum_{y} x^2 I(x,y) \tag{3.46}$$

$$M_{02} = \sum_{x}\sum_{y} y^2 I(x,y) \tag{3.47}$$

Then the object orientation or the direction of the major axis is,

$$\Theta = \frac{\arctan\left[\frac{2\left(\frac{M_{11}}{M_{00}} - x_c y_c\right)}{\left(\frac{M_{20}}{M_{00}} - x_c^2\right) - \left(\frac{M_{02}}{M_{00}} - y_c^2\right)}\right]}{2} \tag{3.48}$$

The first two eigenvalues which is length and width of the probability distribution of the obstacle found by CamShift can be calculated in a closed form as shown below,

By letting,

$$a = \frac{M_{20}}{M_{00}} - x_c^2 \tag{3.49}$$

$$b = 2\left(\frac{M_{11}}{M_{00}} - x_c y_c\right) \tag{3.50}$$

$$c = \frac{M_{02}}{M_{00}} - y_c^2 \tag{3.51}$$

The length l and width w from the distribution centroid can be found,

$$l = \sqrt{\frac{(a+c) + \sqrt{b^2 + (a-c)^2}}{2}} \tag{3.52}$$

$$w = \sqrt{\frac{(a+c) - \sqrt{b^2 + (a-c)^2}}{2}} \tag{3.53}$$

With these values of orientation, length and width calculated, the CamShift algorithm is then adapted to adjust the search window and track window based on these values.

3.7.3 Simulation Results

The Camshift code was edited to either have an adaptive searching window or to have a static search window. During the indoor trials, the simulation was done by choosing the head of one of our researchers as a tracking target. Therefore, in this simulation result, it will be demonstrating a face tracking result.

The simulation is carried out on our own developed Ground Control Station (GCS) with the following steps:

1. From the stream of images sent from the on-board video, a frame is freezed for easy selection of the target.

2. Selection of the target is done by clicking the four corners that bound the intended target. The coordinates of the selection are then sent to the on-board computer for real-time tracking.

3. A display then appears that shows the tracked target and the search area. The search area can be configured to be adaptive or static.

4. The target is then tracked in real-time within the captured frame.

From the simulation, it can be seen that the target tracking is successful for both horizontal and vertical movement as shown in Fig. 3.33 and Fig. 3.34 respectively. At higher speeds, the algorithm is also able to track the target successfully. However, the tracking could be done better if the search area is set larger.

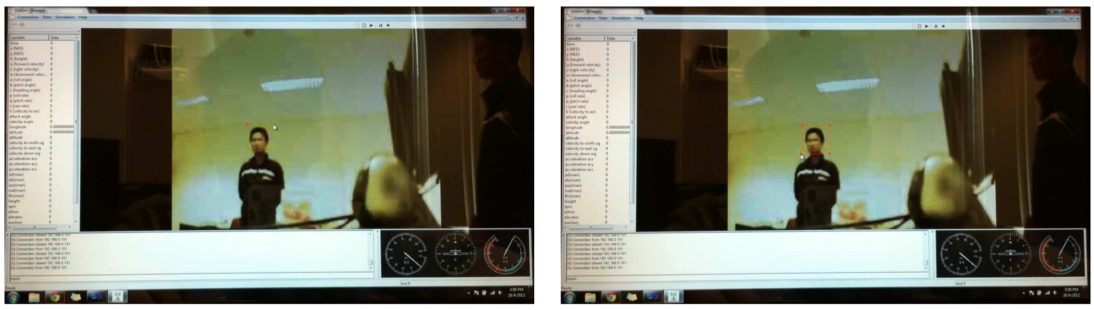
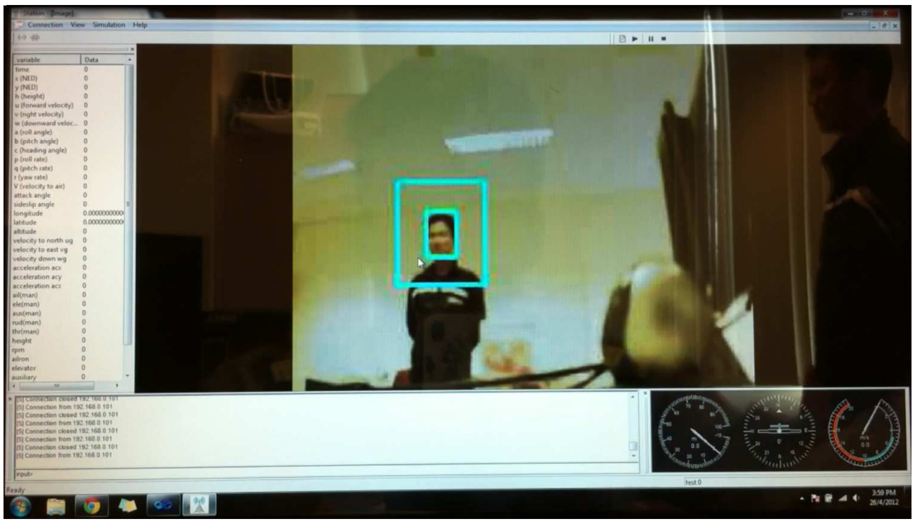


Figure 3.31: Left: Freezing a frame, Right: Selecting target within frame.

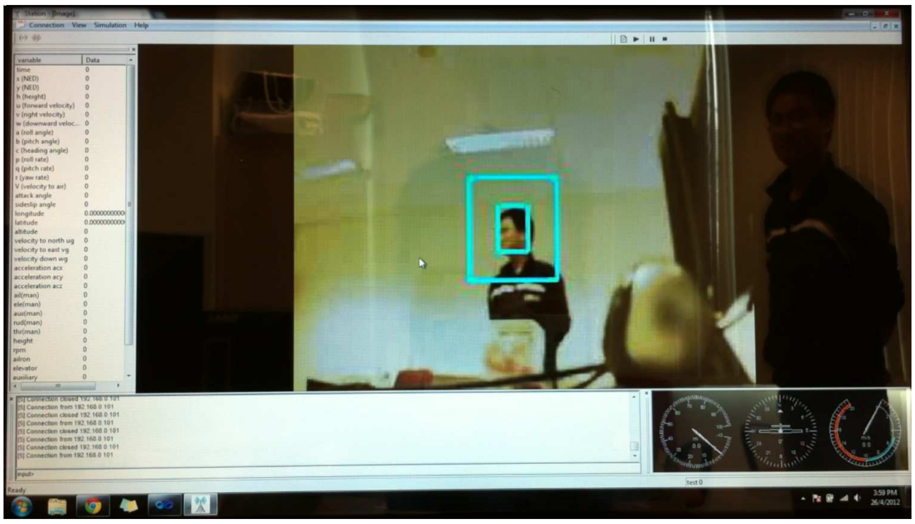
From the results, CamShift algorithm was successfully used to track the selected target by calculating the probability of the selected color and having search windows set around the calculated center. The method assumes that the background is of significantly different color as compared to the selected target for the algorithm to be effective. The error for the target center lies in the range of less than 2 pixel error if the motion is not large and that there is no significant change in color intensity of the target. However, when movement is large and there is a change of perspective of the target, the target center could drift up to 10 pixels. But the algorithm is able to quickly converge back to the target center when movement and intensity stabilizes.



variable	Data
time	0
x (NED)	0
y (NED)	0
h (height)	0
u (forward velocity)	0
v (right velocity)	0
w (downward veloc...	0
a (roll angle)	0
b (pitch angle)	0
c (heading angle)	0
p (roll rate)	0
q (pitch rate)	0
r (yaw rate)	0
V (velocity to air)	0
attack angle	0
sideslip angle	0
longitude	0.0000000000
latitude	0.0000000000
altitude	0
velocity to north ug	0
velocity to east vg	0
velocity down wg	0
acceleration acx	0
acceleration acy	0
acceleration acz	0
ail(man)	0
ele(man)	0
aux(man)	0
rud(man)	0
thr(man)	0
height	0
rpm	0
ailron	0
elevator	0
auxiliary	0

```
[5] Connection closed 192.168.0.101
[5] Connection from 192.168.0.101
[5] Connection closed 192.168.0.101
[5] Connection from 192.168.0.101
[5] Connection closed 192.168.0.101
[5] Connection from 192.168.0.101
[5] Connection closed 192.168.0.101
[5] Connection from 192.168.0.101
input>
```

Figure 3.32: Searching area & detected target.

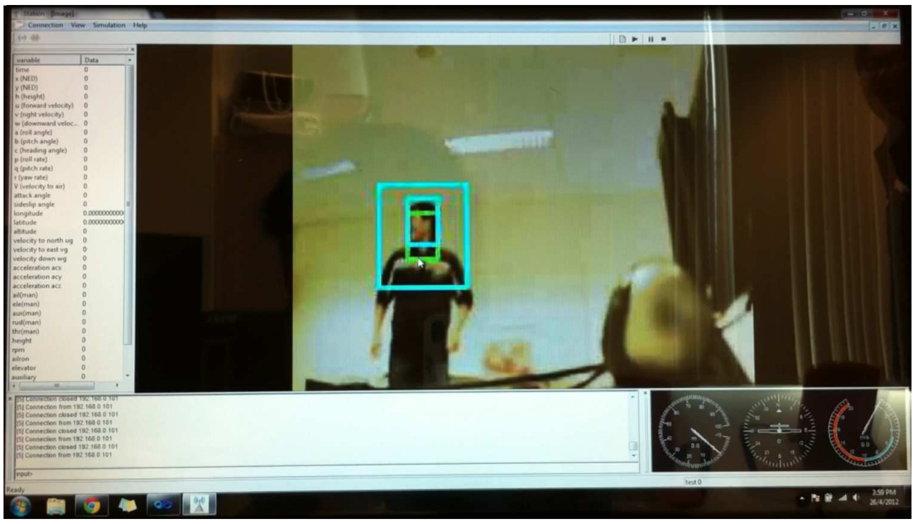


variable	Data
time	0
x (NED)	0
y (NED)	0
h (height)	0
u (forward velocity)	0
v (right velocity)	0
w (downward veloc...	0
a (roll angle)	0
b (pitch angle)	0
c (heading angle)	0
p (roll rate)	0
q (pitch rate)	0
r (yaw rate)	0
V (velocity to air)	0
attack angle	0
sideslip angle	0
longitude	0.0000000000
latitude	0.0000000000
altitude	0
velocity to north ug	0
velocity to east vg	0
velocity down wg	0
acceleration acx	0
acceleration acy	0
acceleration acz	0
ail(man)	0
ele(man)	0
aux(man)	0
rud(man)	0
thr(man)	0
height	0
rpm	0
ailron	0
elevator	0
auxiliary	0

```
[5] Connection closed 192.168.0.101
[5] Connection from 192.168.0.101
[5] Connection closed 192.168.0.101
[5] Connection from 192.168.0.101
[5] Connection closed 192.168.0.101
[5] Connection from 192.168.0.101
[5] Connection closed 192.168.0.101
[5] Connection from 192.168.0.101
input>
```

Figure 3.33: Tracking of target in horizontal movement.

Simulation do show that the method can track the selected target accurately but it can be improved further by the use of target structure and color together. In the application for obstacle avoidance, the search window could be set to be the whole image frame after the algorithm is trained to recognize intended obstacles.



variable	Data
time	0
x (NED)	0
y (NED)	0
h (height)	0
u (forward velocity)	0
v (right velocity)	0
w (downward veloc...	0
a (roll angle)	0
b (pitch angle)	0
c (heading angle)	0
p (roll rate)	0
q (pitch rate)	0
r (yaw rate)	0
V (velocity to air)	0
attack angle	0
sideslip angle	0
longitude	0.0000000000
latitude	0.0000000000
altitude	0
velocity to north ug	0
velocity to east vg	0
velocity down wg	0
acceleration acx	0
acceleration acy	0
acceleration acz	0
ail(man)	0
ele(man)	0
aux(man)	0
rud(man)	0
thr(man)	0
height	0
rpm	0
ailron	0
elevator	0
auxliary	0

```
[5] Connection closed 192.168.0.101
[5] Connection from 192.168.0.101
[5] Connection closed 192.168.0.101
[5] Connection from 192.168.0.101
[5] Connection closed 192.168.0.101
[5] Connection from 192.168.0.101
[5] Connection closed 192.168.0.101
[5] Connection from 192.168.0.101
input>
```

Figure 3.34: Tracking of target in vertical movement.

3.8 Conclusion

Vision-based obstacle detection was implemented in real-time and focus on the detection of obstacles through the use of stereo cameras or a monocular camera. The obstacles were classified into general obstacles and specific obstacles such as powerlines. In current research, there are very limited work done on the detection of power lines. They exist mostly from a bird's eye view and detects powerlines underneath UAVs. The proposed work for powerline detection is unique and could be applied robustly in real-time onboard a UAV. This work could see a lot of applications in low flying UAVs as the ability to sense and avoid is imperative for a UAV to operate in a low flying altitude as this air zone is where a lot of obstacles exists.

Chapter 4

Active Stereo Vision in Navigation

4.1 Introduction

In this chapter, we propose an active stereo vision system that allows image processing of surfaces that lack features or are smooth. Active stereo ranging could be achieved by the projection of structured light on the required surfaces. The depth of the surfaces can then be derived by the matching of the created features on the surface to the features extracted from the acquired image through triangulation. This method requires the illumination of the working environment to be well controlled such that binary images of the working environment could be generated. Following which stereo correspondence between the image pair could be found based on the projected pattern of the light source.

In our application, we do not used a structured pattern light source but instead use multiple visible laser beams. In this way, the stereo correspondence problem does not exist as only the brightest spots in the images generated by the light source needs to be detected and matched. This means that instead of creating a dense stereo correspondence map, we instead go for a sparse map which is efficient in calculation and is less susceptible to noise. The stereo correspondence map is then used to create a sparse 3D point cloud which could be used for navigation or obstacle detection.

We created a detection and matching algorithm which was able to detect the created feature points and matched them accordingly. As this is a stereo system based technique, the features were found along similar epipolar lines that coincide with the horizontal pixel coordinate which is similar to that described in Chapter 3.

What is unique about our application is that whereas most vision algorithms require environments to be rich in features, our method allows us to create a sparse point cloud map even in environments which are devoid of image features. This method is also able to run in real-time at 20 Hz on our Intel $i7$ onboard computer. Our algorithm was also used in the Singapore Amazing Flying Machine Competition (SAFMC) 2014 to create features in an environment which is homogeneous. Our team was commended with the Theory of Flight Award for this implementation.

4.2 Active Stereo Vision Hardware Setup

Active stereo vision setup consists of two major components. Firstly, the backbone of any stereo system are its cameras. As this is a customized stereo system application, we needed to build our own active stereo camera. Matrix Vision mvBlueFOX USB2.0 compact industrial CMOS cameras were chosen to be built upon because of its superior image quality and customization capabilities such as adjusting its white balance, exposure and frame rate. The cameras could also be connected to become hardware synchronized and capture images that are perfectly synchronized. This was essential in any stereo vision system as synchronization was mandatory to obtain correct stereo images for correspondence calculation. The set of cameras were built with a baseline of 10 cm apart and were calibrated to obtain their individual intrinsic parameters as well as the rotation and translational vectors between both cameras.

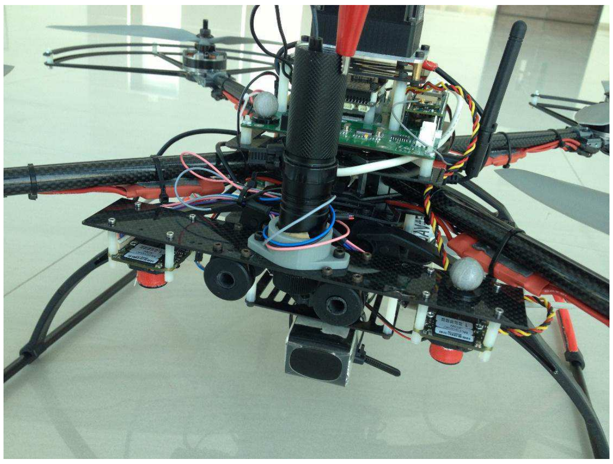


Figure 4.1: Active stereo vision system with laser emitter.

The unique feature about the stereo active vision setup is that it includes a green laser emitter and diffraction grating mounted in the middle of the stereo camera. The green laser emitter emits a laser beam that is parallel to the cameras' principle axis but passes through a diffraction grating filter which splits the beam into multiple beams. These beams projects feature points onto the surfaces the active stereo system is facing. Fig. 4.1 shows the setup of the active stereo system mounted onto our quadrotor UAV. The cameras are connected to the onboard computer and are both mechanically and digital synchronized. Fig. 4.2 depicts the laser rays projected from the laser emitter. Through the diffraction grating, we are able to obtain more than 200 individual laser points to which we could obtain their 3D coordinates.

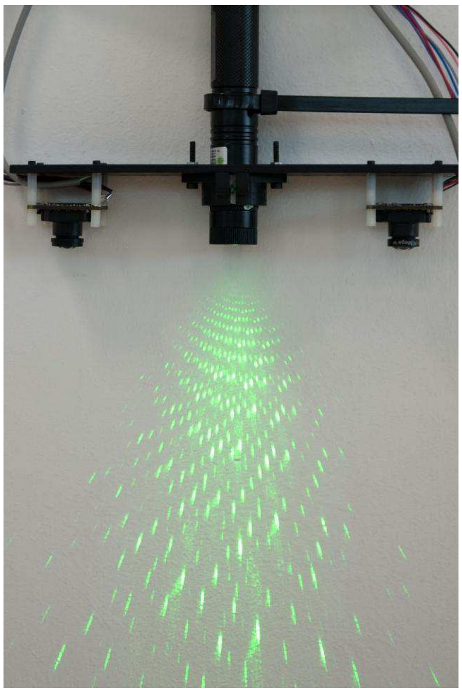


Figure 4.2: Laser rays from laser emitter

In the following section, we will describe the active stereo vision algorithm in detail and cover the essential components of the algorithm such as feature extraction, feature matching and also the method of stereo triangulation used.

4.3 Active Stereo Vision Algorithm

4.3.1 Feature Extraction

The image captured by the active stereo system consists of the laser projected features on the surface of interest. Our algorithm works regardless of whether the surface of interest is textured or homogenous. In Fig. 4.3, you could see clearly where the sparse laser features are projected. The laser features from the laser emitter are much brighter than the environmental lighting which makes it simpler to detect. Therefore, to extract the laser features, our algorithm creates

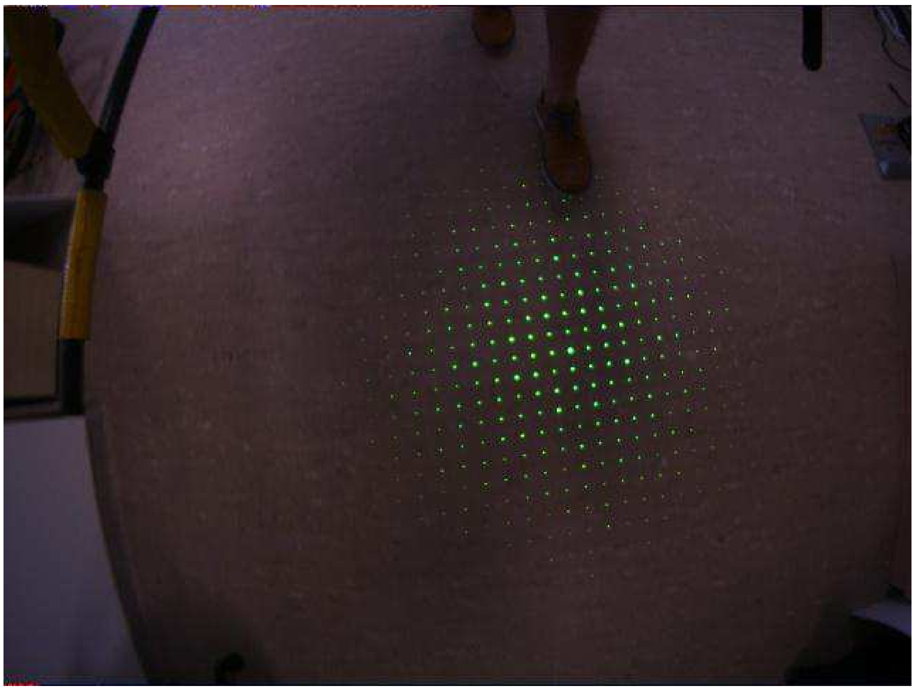


Figure 4.3: Sparse laser features from the left camera view.

a threshold image based on the intensity of the laser features with respect to the environment to give us a binary image as shown in Fig. 4.4. From the image, you may observe that most of the features have been retained and that the background noise has been removed.

Using this binary image, we first obtain contours that bound the binary features and fit circles onto these contours. With the fitted circles, we will be able to obtain the circle center position in pixel coordinates as well as the radius of the circle. Using the radius of the circle, we calculate the pixel area of each contoured binary features. The binary spot features are then filtered based on their pixel area calculated to remove any form of noise and to retain the stronger features of interest. The features found are defined with image pixel coordinates that represent the center of this circle contours.

After extracting the laser features, since we have a calibrated stereo vision system, for the same point in 3D space, it will have 2 projections in both the left and the right images. Thus, we can try to match these 2 projection points first using our feature matching process and project it back to 3D space to give us a sparse 3D point cloud.

4.3.2 Feature Matching

For the matching of the features found in Section 4.3.1, we designed a search pattern algorithm where we can match the features detected in both images of the stereo system. We first rectify the

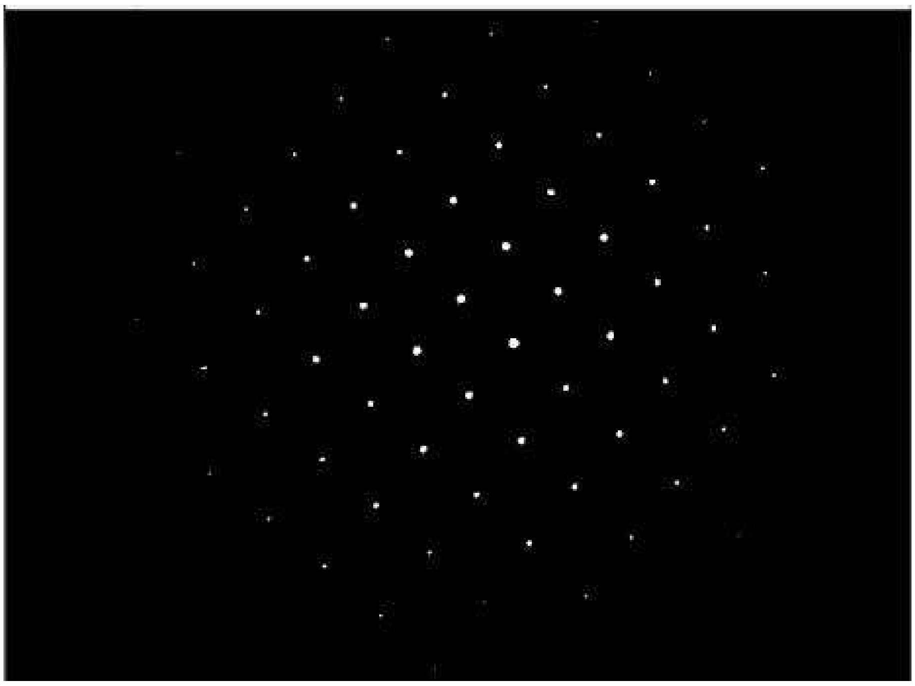
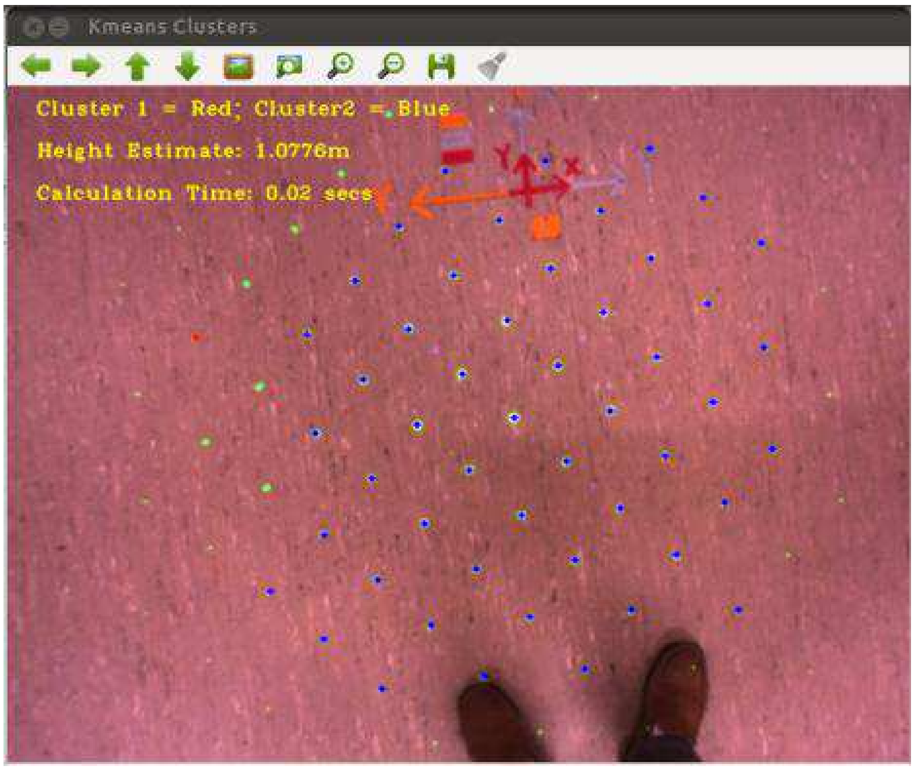


Figure 4.4: Binary image of laser features.



Cluster 1 = Red; Cluster2 = Blue
Height Estimate: 1.0776m
Calculation Time: 0.02 secs

Figure 4.5: Feature point clustering and height estimation.

images based on the calibration parameters we obtain from stereo calibration. After rectification, for a feature found on the left camera, we will be able to search for the same feature along the same epipolar line which is alone the horizontal axis.

In a stereo vision system, when a feature point is found in the left image with image coordinates, (x_i, y_i), its corresponding feature will usually be found in a range left of this image coordinate. Therefore, we set a rectangular search range to the left of the detected feature in the left image and search for the corresponding feature in the right image. We define a search range that we could tune to fix our required working range accordingly. The search range we used is: in x-direction, $20 \leq x_i \leq 150$ pixels; in y-direction, $-10 \leq y_i \leq 10$ pixels. If this search is successful, the two matched feature points will be used for stereo triangulation method. During the matching, there are times where we could have some wrong matches or multiple matches. These are solved by rejecting multiple matches and treating these matches as erroneous matches.

4.3.3 Linear Stereo Triangulation

After matching the feature points, we could then calculate the respective 3D world coordinate of the feature using the reprojection matrix found for each camera. Simple triangulation by back-projecting rays from the two matched points will fail because the rays do not necessarily intersect due to errors in measured image coordinates.

For a 3D world point, X, which satisfies the equations, $x = PX, x' = P'X$. Where P and P' are projection matrices for the left and right camera. It will be possible to estimate a best solution for this 3D point. Through camera calibration, we have already obtained the camera intrinsic matrices with great accuracy. The core idea of stereo triangulation is to estimate a 3D point, $\hat{X}$, which exactly satisfies:

$$\hat{x} = P\hat{X} \tag{4.1}$$

$$\hat{x'} = P'\hat{X} \tag{4.2}$$

The triangulation method used in our algorithm is the linear triangulation method. Assuming that x from Eqn. 4.1 is written in homogeneous coordinates, $\mathbf{x} = w(u, v, 1)^T$, where (u, v) are the observed point coordinates and w is an unknown scale factor. Now, denoted by p_i^T which represent the i-th row of the matrix, P, then Eqn. 4.1 could be written as:

$$wu = p_1^T X \tag{4.3}$$

$$wv = p_2^T X \tag{4.4}$$

$$w = p_3^T X \tag{4.5}$$

Substituting the unknown scale factor, w, back into the equations, we get,

$$up_3^T X = p_1^T X \tag{4.6}$$

$$vp_3^T X = p_2^T X \tag{4.7}$$

The above two equations could be written in the form $\mathbf{AX} = \mathbf{0}$ which represents homogeneous equations. By setting $X = (x, y, z, 1)^T$, one can reduce the set of homogenous equations to a set of 4 non-homogeneous equations with 3 unknowns in the form:

$$AX + b = 0 \tag{4.8}$$

where,

$$A = \begin{bmatrix} u_1 \times P_{(3,1)} - P_{(1,1)} & u_1 \times P_{(3,2)} - P_{(1,2)} & u_1 \times P_{(3,3)} - P_{(1,3)} \\ v_1 \times P_{(3,1)} - P_{(2,1)} & v_1 \times P_{(3,2)} - P_{(2,2)} & v_1 \times P_{(3,3)} - P_{(2,3)} \\ u_2 \times P'_{(3,1)} - P'_{(1,1)} & u_2 \times P'_{(3,2)} - P'_{(1,2)} & u_2 \times P'_{(3,3)} - P'_{(1,3)} \\ v_2 \times P'_{(3,1)} - P'_{(2,1)} & v_2 \times P'_{(3,2)} - P'_{(2,2)} & v_2 \times P'_{(3,3)} - P'_{(2,3)} \end{bmatrix} \tag{4.9}$$

$$X = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{4.10}$$

$$b = \begin{bmatrix} u_1 \times P_{(3,4)} - P_{(1,4)} \\ v_1 \times P_{(3,4)} - P_{(2,4)} \\ u_2 \times P^{'}_{(3,4)} - P^{'}_{(1,4)} \\ v_2 \times P^{'}_{(3,4)} - P^{'}_{(2,4)} \end{bmatrix} \tag{4.11}$$

Eqn. 4.8 could then be solved in a least squares manner using Singular Value Decomposition (SVD). Solving the SVD, we arrive at the 3D coordinates of the feature point with respect to the camera frame.

4.3.4 K-means Clustering

In every active stereo pair of images, we are able to obtain at least 50 3D coordinates and their corresponding image feature points. After obtaining this sparse point cloud generated by the active stereo system, we then go about performing a K-means clustering [75] of the 3D world coordinates as shown in Fig. 4.5. Performing clustering allows us to create an obstacle based map where 3D points are clustered into groups and could represent groups of obstacles. Other than creating a generic obstacle map, K-means clustering allows us to reduce any possible residual errors obtain from stereo triangulation. The points are clustered and the center of each cluster with the point deviation can be obtained and further analysis could be performed either for obstacle detection or navigation purposes.

4.4 Urban Environment Scenario

There are scenarios in certain missions where UAVs are required to fly into buildings for reconnaissance and surveillance missions. The UAVs help to participate in urban search and rescue missions and are required to fly and navigate without aid from GPS information. Therefore, there is a need to develop a system which allows a UAV to navigate in a GPS-less environment. In urban indoor environments, there are many cases where walls, flooring and ceilings are of a homogenous color or pattern such as that shown in Fig. 4.6 and these are extremely difficult environments for image processing algorithms to work.

In such an environment, scanning laser rangefinders are frequently used for navigation but they draw a lot of power, weigh around 450 g and only provide planar information for navi-

Figure 4.6: Urban indoor environment.

gation. Laser rangefinders frequently do not provide a rich enough point cloud to aid in the capability in detecting obstacles that are either above or below their scanning plane. An active stereo system could be used in this situation to provide a 3D point cloud that would allow the UAV system to navigate in such an environment.

4.5 Conclusion

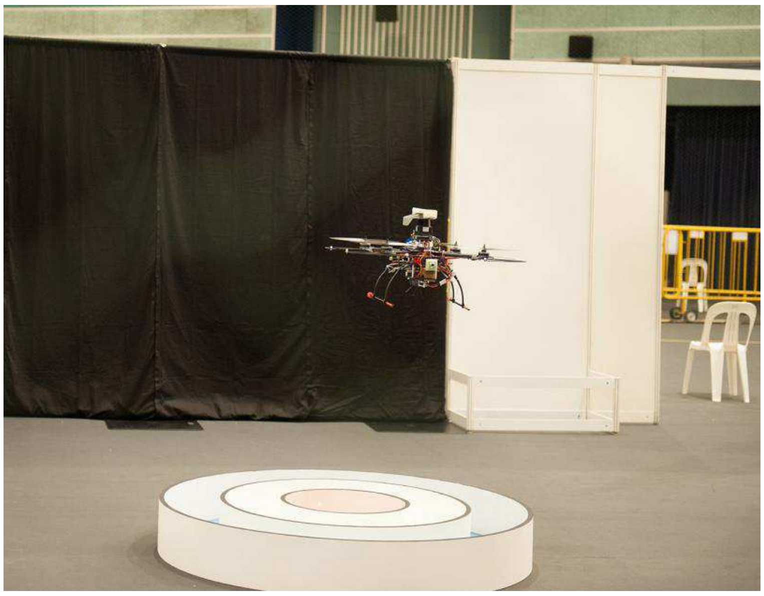


Figure 4.7: Vision guidance flight.

We tested the above mentioned active stereo system during the SAFMC 2014 Competition as shown in Fig. 4.7. The active stereo system was placed near the front of the quadrotor UAV facing downwards. This allowed us to create a sparse point cloud below the UAV which could detect obstacles like tables and chairs as it was flying in an urban indoor environment.

There were tasks depicted in the competition which required the UAV to fly through window-like openings. Therefore, we used this active stereo system to calculate and maintain the height of the UAV as it flies through windows. This scenario might seem simple but certain sensors such as a single laser rangefinder will detect a spike in readings as it passes a window. By using the K-means clustering, we could prevent this spike in readings and allow us to have a smooth height reading that could be used for flight control.

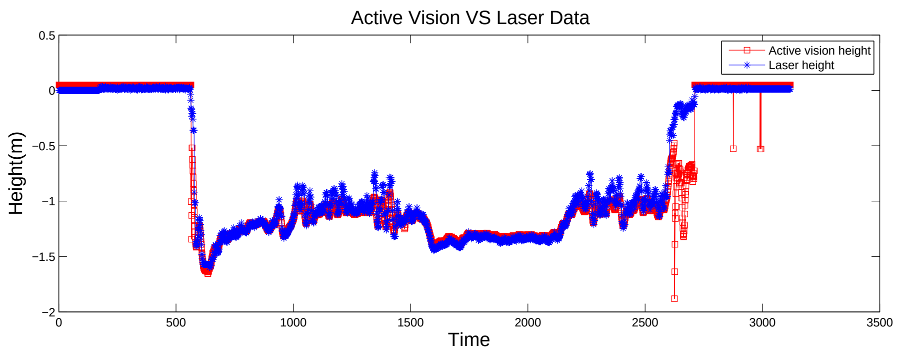


Figure 4.8: Height comparison between laser readings and active stereo system.

Fig. 4.8 shows the height data comparison between the active stereo system and raw laser rangefinder readings. From the figure, you can see that the height readings obtained from active stereo vision is smoother as compared to the laser reading. The noisy readings of the active stereo system happen near the end of the data set due to the fact that the UAV is performing landing and that the height of the UAV is not within the working range of the algorithm.

Comparing the results obtain from the active vision height algorithm and the raw data from the laser rangefinder is not conclusion as both measurements have their pros and cons. The laser rangefinder data is very accurate during movement on flat ground and could be used as a ground truth. During this situation, the active stereo system is able to have errors less than $5\ \mathrm{cm}$ when compared with the laser readings. However when there is movement over a wall, the active stereo system is more robust and could provide very smooth results while the laser rangefinder data has spikes and erroneous readings.

Chapter 5

Stereo-based Visual Navigation

5.1 Introduction

Stereo vision has been used primarily as the main sensor in many UAV system navigation algorithms. The stereo vision navigation system has to operate in real-time with low delay and the motion estimation obtained can be used to perform path planning and obstacle avoidance. The forefront of the algorithms are the feature detection and matching between subsequent frames obtained by the stereo cameras. Each stereo image pair is also used to generate a sparse disparity map which allows us to calculate the 3D world coordinate of each point detected by the feature detector. Robust estimates of the camera's egomotion can then be obtained through both linear and non-linear optimization before translating it to that of UAV motion. Random Sample Consensus [76] (RANSAC) outlier rejection is frequently applied to reject erroneous points that have been included into the calculation. The RANSAC-based rejection scheme can be used to reject dynamic points that are frequently present in such scenes. Once the features are detected and robustly tracked, they are used to match to their corresponding 3D coordinate found in the stereo depth map. The camera's pose estimation could then be calculated using this set of 3D-to-2D feature pairs. This function finds the pose that minimizes reprojection error, i.e. the sum of squared distances between the observed projections image points and the projected object points.

In the following sections, we will describe a detailed framework that is used to achieve stereo-based visual navigation. The navigation work done in this thesis will encompass only odometry based techniques. Odometry is the technique where it uses data from motion sensors to estimate change in position over time. It basically integrates the velocity and position estimates to estimate the UAVs' relative position against its starting position.

Stereo odometry is particularly useful for UAV motion estimation as UAVs generally travel long distances and without repeating their paths. The proposed stereo odometry algorithm solves this motion estimation problem efficiently and robustly. Although there are other techniques such as visual Simultaneous Localization and Mapping (SLAM) [77] , they usually are less computationally efficient and UAV applications in urban environments seldom require the close-loop capability of visual SLAM. As mentioned in Chapter 3, since our obstacle sensing algorithm runs mainly based on a stereo system, we decided to extend this stereo system to encompass navigational capabilities. The combination of stereo ranging and stereo visual odometry is generally faster and more reliable than stereo ranging followed by monocular visual odometry and we aim to use this to provide reliable motion estimation for UAVs.

We will cover the use of visual odometry obtained from 3D-to-3D absolute orientation calculation as well as 3D-to-2D calculations using "Perspective-n-Points".

5.2 Feature Matching & Calculating 3D Points

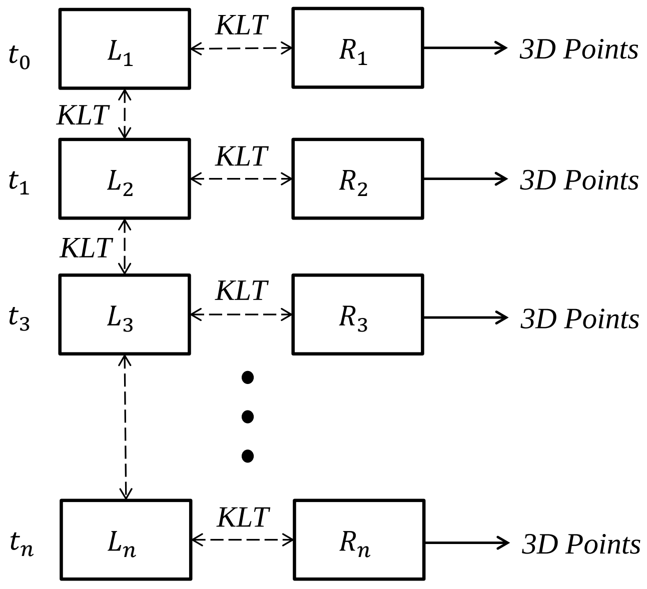


Figure 5.1: Feature detection & 3D coordinate generation.

Given sequential images obtained from a stereo camera, we can track the features using the

KLT optical flow tracker. KLT provides fast and reliable feature tracking while maintaining low computational loads. During feature detection we perform two sets of checks to ensure robust image features. Firstly, we set a mask in the center of the image such that features found at the edges of the image are discarded as these features will usually have optical flow tracks that extend out of the image. Secondly, we set a threshold which will allow us to detect image features that are at least a set number of pixels apart. In the case visual odometry, we set this threshold to be at least 15 pixels to ensure we have a well spread our feature set as well as reducing the total number of features to have a more efficient detection phase. During the feature tracking phase, we perform the KLT optical flow tracker iteratively with gaussian pyramids. This ensures robust feature tracking that could effectively reject noise.

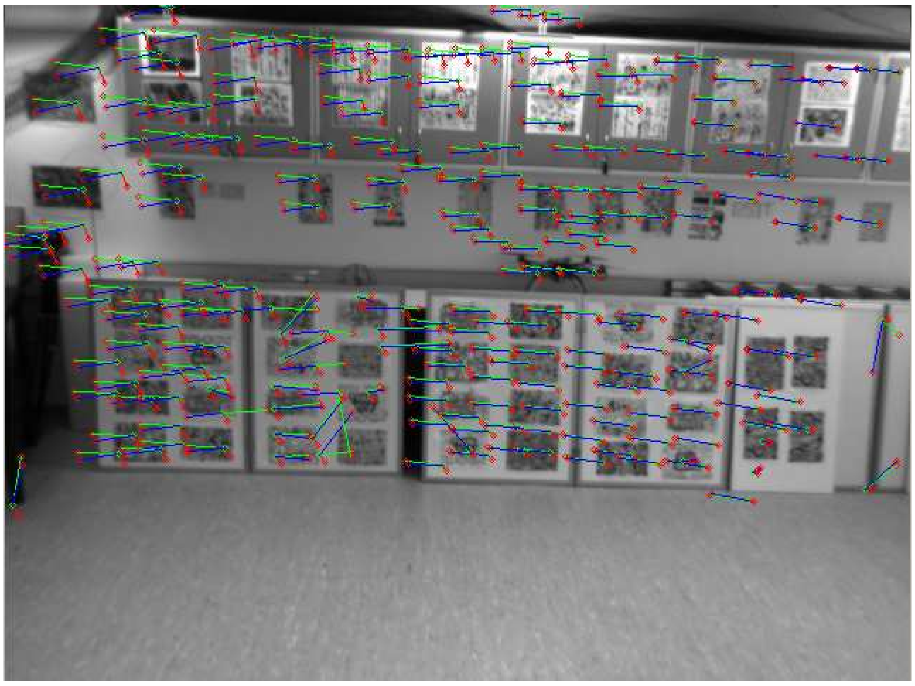


Figure 5.2: KLT feature tracks.

Seen in Fig. 5.1, KLT optical flow tracker is applied between every stereo image pair to obtain stereo correspondence matches. We performed a sparse matching as motion estimation does not require a large number of features to be solved. KLT is further applied to each subsequent left image frame to ensure tracking of features between different time frames. In Fig. 5.2, you can see the feature tracks which represent different correspondences. Blue and green tracks represent the matches found in between stereo pairs. The red track represents the tracks between each time frame.

The features are detected and matched across both left and right images to ensure stereo correspondence. We search for the features along the horizontal epipolar lines with a fixed threshold error to allow matches that maybe slightly displaced from the epipolar lines. The stereo correspondence matches are then used to triangulate the 3D coordinate of the feature point. We perform linear stereo triangulation as mentioned in Section 4.3.3 to obtain the 3D world coordinates of each image feature.

To ensure continuous motion estimate, we must ensure that the features found in the left image frame are matched. The features detected in the left frame are matched such that we could use this relative motion to obtain the optimal rotation and translation as described in Section 5.3.

5.3 Stereo Odometry Motion Estimation

Stereo odometry used for autonomous ground vehicles have been prevalent in the past decade as seen in many works [78], [79]. It usually is performed using a combination of other sensors such as wheel encoders, and inertial sensors. The literature uses stereo odometry to estimate vehicular motion from a sequence of images captured from a camera and fuses this with other sensors for a smoother and more reliable estimate. Its has also seen uses in the Mars Exploratory Rover [80]. In this section, we will investigate motion estimation using stereo odometry on UAVs.

There are many ways to achieve motion estimation using features found in images. There are 2D-to-2D methods which mainly delve in monocular camera applications. Then there are 3D-to-3D methods and 3D-to-2D which are applicable to stereo vision systems. For 3D-to-3D motion estimation, we intend to use the Absolute Orientation method. For the 3D-to-2D motion estimation, we intend to use "Perspective-n-Points".

In this section, we will describe in detail both methods respectively and investigate their results in Section. 5.9.

5.3.1 Rigid Motion Computation

With two sets of 3D points, point set P and point set Q, that correspond with each other between two subsequent frames, we can find a rigid transformation that will optimally align the two sets in the least squares sense. Point set P being the 3D points obtained at time t_1 and Point set Q being the 3D points obtained at time t_2 where $t_2 > t_1$. This is the well known Absolute Orientation Problem [62]. Given that point set $\boldsymbol{P} = [p_1, p_2, ..., p_n]$ and point set $\boldsymbol{Q} = [q_1, q_2, ..., q_n]$, for obtaining the optimal translation ***t*** and rotation ***R***, we will need to minimize:

$$(R, t) = \underset{R,t}{argmin} \sum_{i=1}^{n} w_i \left\|(Rp_i + t) - q_i\right\|^2 \tag{5.1}$$

where $w_i > 0$ are the weights of each matching point pair.

We start by computing the weighted centroids of both the point sets:

$$\bar{p} = \frac{\sum_{i=1}^{n} w_i p_i}{\sum_{i=1}^{n} w_i} \tag{5.2}$$

$$\bar{q} = \frac{\sum_{i=1}^{n} w_i q_i}{\sum_{i=1}^{n} w_i} \tag{5.3}$$

For our case, we currently weigh all of our points with equal weightage.

Next, using the calculated centroids, we can then compute the centered vectors for all $i = (1, 2, ..., n)$:

$$a_i = p_i - \bar{p} \tag{5.4}$$

$$b_i = q_i - \bar{q} \tag{5.5}$$

Upon obtaining the centered vectors, we form A and B which are $d \times n$ matrices where d is the dimensionality of the points. These matrices have a_i and b_i as their respective columns. We then compute the $d \times d$ covariance matrix, S:

$$S = AWB^T \tag{5.6}$$

where $W = diag(w_1, w_2, ..., w_n)$.

The Singular Value Decomposition (SVD) of the covariance matrix is obtained, $S = U\Sigma V^T$. With this, we can finally obtain both the optimal rotation and translation.

$$R = V \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & det(VU^T) \end{pmatrix} U^T \tag{5.7}$$

$$t = \bar{q} - R\bar{p} \tag{5.8}$$

During the calculation of the optimal rotation and translation, there will arise a unique case where the reflection is obtained instead. This happens when $det(VU^T) = -1$ and will return a reflection solution which is not desirable. When this case happens, the desired rotation R can be obtained by $R = V'U^T$ where V' is obtained from V by changing the sign of the 3rd column. However, from Eqn. 5.7, with a simple implementation of the $det(VU^T)$, we can resolve this problem and obtain only the true optimal rotation and translation we require.

5.3.2 Perspective-n-Points Motion Estimation

In computer vision literature, to estimate the camera pose from n 3D-to-2D point correspondences is a very fundamental problem. Most problems requires the estimation of six degrees of freedom of the pose using five camera calibration parameters. They are namely focal length, principal point, aspect ratio and skew obtained during camera calibration. It could in fact be established with a minimum of 6 correspondences using the well known Direct Linear Transform [81] (DLT) algorithm. We will explore the use of the Perspective-n-Points (PNP) problem which is an improved version of the DLT algorithm. The Perspective-n-Points problem assumes known camera calibration parameters which we obtain during our camera calibration procedure and provides accurate results.

Given a set of correspondences between 3D points, P_i, expressed in world coordinates, and their respective 2D image projections, u_i, we want to obtain and retrieve the pose of the camera body with respect to the world with the focal length, f.

From Fig. 5.3, assuming the camera, O, observed two points, p_i and p_j, $d_{ij} = \|p_i - p_j\|$. The pair of points also defines an angle, θ, between their line of sight rays.

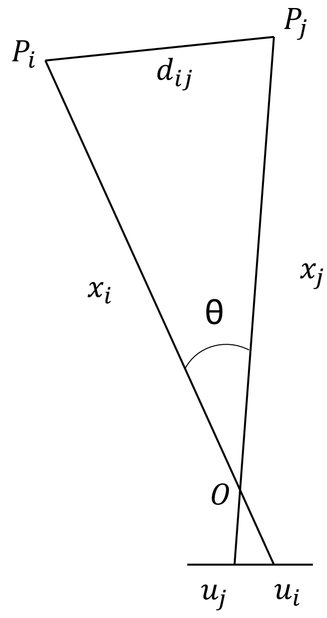


Figure 5.3: Perpective-n-Points formulation.

$$x_i = \|p_i - O\| \tag{5.9}$$

$$x_j = \|p_j - O\| \tag{5.10}$$

$$d_{ij}^2 = x_i^2 + x_j^2 - 2x_i x_j cos\theta_{ij} \tag{5.11}$$

$$f_{ij}(x_i, x_j) = x_i^2 + x_j^2 - 2x_i x_j \cos\theta_{ij} - d_{ij}^2 = 0 \tag{5.12}$$

where from camera calibration, we obtain,

$$\cos\theta_{ij} = \frac{u_i^T O u_j}{\sqrt{u_i^T O u_i}\sqrt{u_j^T O u_j}} \tag{5.13}$$

Assuming if there are 3 points, P3P, we arrive at an underdetermined system where three functions similar to Eqn. 5.12 could be obtained as shown below:

$$f_{12}(x_1, x_2) = 0 \tag{5.14}$$

$$f_{13}(x_1, x_3) = 0 \tag{5.15}$$

$$f_{23}(x_2, x_3) = 0 \tag{5.16}$$

With 4 points, P4P, we then arrive at an overdetermined system where we could form 4 P3P groups. Namely using a combination of the 4 points, we arrive at groups (P_1, P_2, P_3), (P_1, P_2, P_4), (P_1, P_3, P_4) and (P_2, P_3, P_4). We could then solve for the x_i associated with each point.

From the pinhole camera model, we have,

$$w \cdot x = K\,[R|t]\,X \tag{5.17}$$

We then project the image points into the scene,

$$p_i' = x_i K^{-1} u_i \tag{5.18}$$

Then solve for the optimal rotation and translation parameters using a set of linear equations represented by Eqn. 5.19.

$$w \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \cdot \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{5.19}$$

Our algorithm iteratively solves the above equations to obtain an optimal solution. It returns a rotation and translation vector referenced from a set of 3D points from time t_1 matched to the features at time t_2 where $t_2 > t_1$.

5.4 RANSAC-based Outlier Rejection

RANdom SAmple Consensus (RANSAC) is a very successful robust estimator that is able to cope with a large proportion of outliers. In this algorithm, RANSAC is used to reject outliers before calculating the optimal rotation and translation. The form of outliers come in two distinct types. In most navigation problems, dynamic objects are almost always present in the scene of interest and will contribute to gross outliers should they not be removed. Additionally, during the calculation of stereo disparity, its result includes a lot of noise as calculation of areas with constant textures will usually lead to error. Both these outliers must be robustly removed and we base it on the RANSAC-based Outlier Rejection scheme. Shown below is the RANSAC robust estimation algorithm.

RANSAC Algorithm

1. Given set of S matched 3D points, P and Q, randomly select a subset of 3 pairs of matched 3D points and apply the Rigid Motion Computation from Section 5.3.1 to obtain the optimal rotation R_s and translation T_s for the subset.

2. Apply the rotation and translation calculated to obtain $Q' = R_sP + T_s$. Calculate the Euclidean distance between each 3D point between point set Q and Q'.

3. Determine the set of data points S_i which are within a distance threshold t. The set S_i is the consensus set of the sample and defines the inliers of S.

4. If the size of S_i is greater than some threshold T, re-estimate the optimal rotation and translation from Section 5.3.1 using the inlier set S_i and terminate.

5. If the size of S_i is less than T, select a new subset and repeat the above.

6. After N (determined from Table 5.1) trials, the largest consensus set S_i is selected and the optimal rotation and translation are re-estimated.

Table 5.1: Number of iterations N for sample size s against proportion of outliers ϵ

Sample size	**Proportion of outliers, ϵ**						
s	5%	10%	20%	25%	30%	40%	50%
3	3	4	7	9	11	19	35

$$N = \frac{log(1-p)}{log(1-(1-\epsilon)^s)} \tag{5.20}$$

Table 5.1 is calculated from Eqn. 5.20 where p is the probability that at least one of the random samples of s points is free from outliers. p is chosen at 0.99. s is the sample size which is set as 3 for a rigid motion computation problem.

5.5 Non-linear Optimization

Upon obtaining the rotation and translation through the use of rigid motion calculation and RANSAC outlier rejection scheme, we use this as the initialization of the Levenberg-Marquardt algorithm for nonlinear least squares minimization [82]. The Jacobian required for this minimization is approximated by forward differencing. Since the 3×3 rotation matrix has only three degrees of freedom, we work with the euler angles instead. The variable for this minimization are the three euler angles and the three translation parameters as described in (5.30) to (5.31).

The error function to be minimized is given by:

$$\min \sum_{i=1}^{N} \left\| {\omega_i}' - {\omega_i}'' \right\|^2 \tag{5.21}$$

5.6 Pose Estimation

Finally, after obtaining the optimal rotation and translation of the camera, we can integrate the camera's rotation and translation at every time step to obtain its pose relative to its first coordinate frame.

Given P_0 as the position of the camera in frame 0 and P_1 as the position of the camera in frame 1,

$$P_1 = R_1 P_0 + t_1 \tag{5.22}$$

$$\begin{bmatrix} P_1 \\ 1 \end{bmatrix} = \begin{bmatrix} R_1 & t_1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} P_0 \\ 1 \end{bmatrix} \tag{5.23}$$

Also, for the next subsequent frame,

$$\begin{bmatrix} P_2 \\ 1 \end{bmatrix} = \begin{bmatrix} R_2 & t_2 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} P_1 \\ 1 \end{bmatrix} = \begin{bmatrix} R_2 & t_2 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} R_1 & t_1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} P_0 \\ 1 \end{bmatrix} \tag{5.24}$$

Generalizing, we have:

$$\begin{bmatrix} P_n \\ 1 \end{bmatrix} = \begin{bmatrix} R_n & t_n \\ 0 & 1 \end{bmatrix} \begin{bmatrix} R_{n-1} & t_{n-1} \\ 0 & 1 \end{bmatrix} \cdots \begin{bmatrix} R_1 & t_1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} P_0 \\ 1 \end{bmatrix} \tag{5.25}$$

$$\begin{bmatrix} P_n \\ 1 \end{bmatrix} = C \begin{bmatrix} P_0 \\ 1 \end{bmatrix} \tag{5.26}$$

$$\begin{bmatrix} P_0 \\ 1 \end{bmatrix} = C^{-1} \begin{bmatrix} P_n \\ 1 \end{bmatrix} \tag{5.27}$$

Where C^{-1} is the relative pose of the camera from frame n with respect to frame 0 coordinate system. C^{-1} is described as:

$$C^{-1} = \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{5.28}$$

The elements of C^{-1} can then be decomposed into the 3 Euler angles ϕ, θ, ψ that describe the roll, pitch and yaw and also the translation in x, y, z axis used for navigation.

$$\phi = atan2(r_{32}, r_{33}) \tag{5.29}$$

$$\theta = atan2(-r_{31}, \sqrt{r_{32}^2 + r_{33}^2}) \tag{5.30}$$

$$\psi = atan2(r_{21}, r_{11}) \tag{5.31}$$

5.7 Kalman Filter Design

The image calculations for position estimates can be achieved at a frequency of 10 Hz. Whilst the IMU data is obtained at 50 Hz. There is a need to fuse these data together to ensure a smoother UAV pose estimate output from our algorithm. Therefore, we decided to implement a 9-state Kalman Filter as our filter of choice.

Considering a Linear Time Invariant (LTI) system in the form:

$$\dot{\mathrm{x}} = \mathrm{Ax} + \mathrm{Bu} + \mathrm{v}(t) \tag{5.32}$$

$$\mathrm{y} = \mathrm{Cx} + \mathrm{w}(t) \tag{5.33}$$

where x, u and y are the states, inputs and measurement vectors. A, B, C are system matrices of dimensions, $[9 \times 9]$, $[9 \times 3]$ and $[3 \times 9]$ respectively. w and v are input and measurement noises which can be assumed to be Gaussian with zero means and covariance matrices Q and R respectively shown below:

$$E[v(t)] = 0 \tag{5.34}$$

$$E[w(t)] = 0 \tag{5.35}$$

Also,

$$E[v(t)v^T(\tau)] = Q\delta(t-\tau), \tag{5.36}$$

$$Q = Q^T \geq 0, \tag{5.37}$$

$$E[w(t)w^T(\tau)] = R\delta(t-\tau), \tag{5.38}$$

$$R = R^T > 0. \tag{5.39}$$

For the UAV pose estimation problem, the motion model and the measurement model which consists of the state vector x, input vector, measurement vector y and the system matrices need to be defined as Eqn. 5.40 to Eqn. 5.45.

$$\mathrm{x} = [x, y, z, u, v, w, \beta_x, \beta_y, \beta_z]^T \tag{5.40}$$

$$\mathrm{u} = [a_x, a_y, a_z]^T \tag{5.41}$$

$$\mathrm{y} = [x, y, z]^T \tag{5.42}$$

where x, y, z represent the NED position coordinates, u, v, w represent the NED velocities, a_x, a_y, a_z represent the NED accelerations and $\beta_x, \beta_y, \beta_z$ represent the IMU accelerometer bias.

The system matrices are defined as such,

$$\mathrm{A} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{5.43}$$

$$\mathrm{B} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{5.44}$$

$$\mathrm{C} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{5.45}$$

Apart from the above defined motion model and the measurement model, we have to define the input noise matrix Q and the measurement noise matrix R which are both positive definite and diagonal and could be obtained by logging flight test data. The diagonal elements in Q represent the acceleration measurement noises and the diagonal elements in R represent the noises obtained in position estimate from visual odometry. With the defined Kalman filter models, we can then build a discrete model to be implemented in our algorithm.

5.8 Transformation of Points in 3D Space

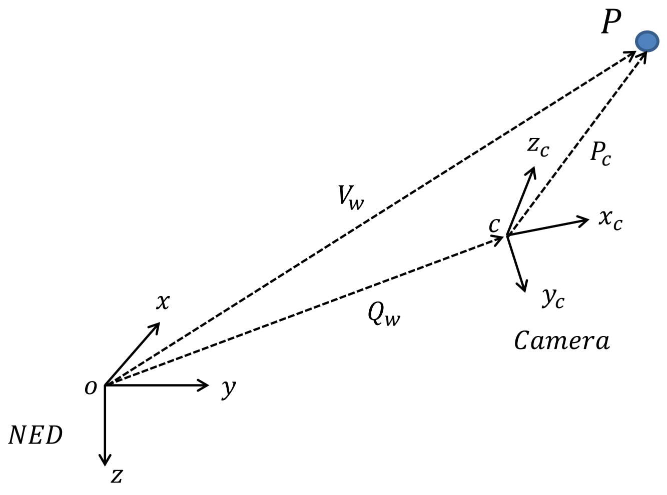


Figure 5.4: Camera coordinate system against NED coordinate system.

The UAV body frame is defined in the North-East-Down (NED) coordinate system. Whereas the camera coordinate frame is defined differently as shown in Fig. 5.4. From the defined coordinate system, you can see that the body frame's x-axis, y-axis and z-axis correspond to the camera frame's z_c-axis, x_c-axis and y_c-axis respectively. Therefore, there is a need to find the rotation matrix that is needed to transform the coordinates from the camera frame to that of the NED frame in Eqn. 5.48.

$$P_b = R_{b/c} P_c \tag{5.46}$$

$$\begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} = R_{b/c} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.47}$$

$$R_{b/c} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \tag{5.48}$$

With this rotation matrix calculated, all 3D coordinates of feature points found could be transformed into that of the body NED frame for reference to navigation and map building. The equation for creating 3D maps with the objects detected from the stereo vision algorithm is shown in Eqn. 5.49.

$$V_w = R_{w/c} P_c + Q_w \tag{5.49}$$

where V_w is the vector of the object in world coordinate. $R_{w/c}$ is the rotation matrix that transform camera coordinates to the world coordinate frame. Q_w is the position vector of the UAV in world coordinate.

$$R_{w/c} = R_{n/b} R_{b/c} \tag{5.50}$$

$R_{n/b}$ is obtained from UAV onboard IMU readings and calculated below. The rotation angles are the Euler angles obtained from the UAV. To rotate the navigation frame to coincide with the body frame, the sequence of rotation to be carried out is to rotate about the z, y and x axis respectively.

$$
\begin{aligned}
R_{n/b} &= R_z(\psi)R_y(\theta)R_x(\phi) &(5.51)\\
&= \begin{bmatrix} cos\psi & -sin\psi & 0 \\ sin\psi & cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} cos\theta & 0 & sin\theta \\ 0 & 1 & 0 \\ -sin\theta & 0 & cos\theta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\phi & -sin\phi \\ 0 & sin\phi & cos\phi \end{bmatrix} &(5.52)\\
&= \begin{bmatrix} cos\theta cos\psi & sin\phi sin\theta cos\psi - cos\phi sin\psi & cos\phi sin\theta cos\psi + sin\phi sin\psi \\ cos\theta sin\psi & sin\phi sin\theta sin\psi + cos\phi cos\psi & cos\phi sin\theta sin\psi - sin\phi cos\psi \\ -sin\theta & sin\phi cos\theta & cos\phi cos\theta \end{bmatrix} &(5.53)
\end{aligned}
$$

5.9 Conclusion

We set up our experiment in a VICON enabled workspace which allows us to obtain the true motion of our UAV system. The VICON system could measure the UAV's absolute position and attitudes relative to the defined VICON coordinate system. Our stereo system was mounted in front of the UAV facing a feature-rich environment. During our experiment, we positioned our UAV over the origin of the VICON system before moving in a rectangular fashion around the workspace. The distance we covered was a perimeter of a 3 m by 2 m rectangle.

Fig. 5.5 shows the 3D-to-3D Rigid motion stereo odometry estimation results plotted against the VICON system data. The plot is a 2D plan view of the experiment work space and depicts the motion that our camera system has undergone. It could be seen that the 3D-to-3D motion estimation is not very smooth and there were alot of erroneous movement estimated. The errors involved range from error spikes of up to $0.75\ \mathrm{m}$ against the VICON logged movement. On average, the error ranges between $20\ \mathrm{cm}$ to $30\ \mathrm{cm}$. However, one positive note was that the result at least had a closed loop which was in-line with our experiments.

We found the results unsatisfactory and sort to use the 3D-to-2D Perspective-n-Points motion estimation method. We believed that by using this method, we will reduce the error propagation we observed in stereo triangulation modeling.

Fig. 5.6 shows the implementation of Iterative PNP against the VICON data set. In our motion estimation comparison, it could be seen that the 3D-to-2D method exhibits less drift than the 3D-to-3D method and conforms more to the ground truth obtained from the VICON system. The red curve was our first implementation of our algorithm and we found that the results were skewed and showed a significant drift. We then enhanced our RANSAC outlier

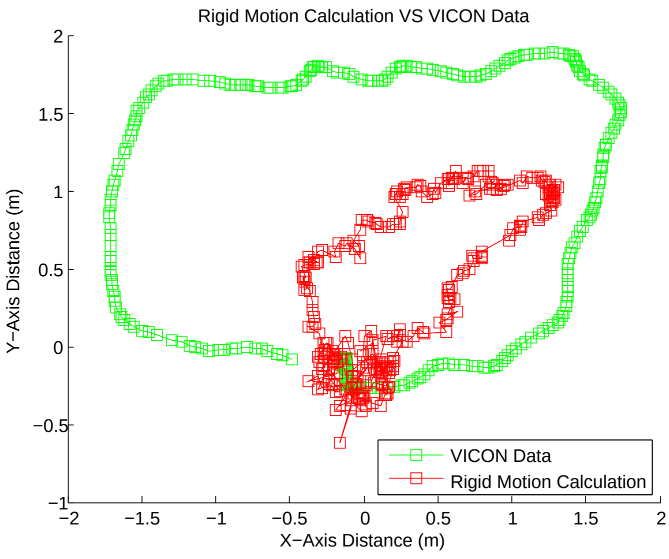


Figure 5.5: Rigid motion calculation vs VICON data.

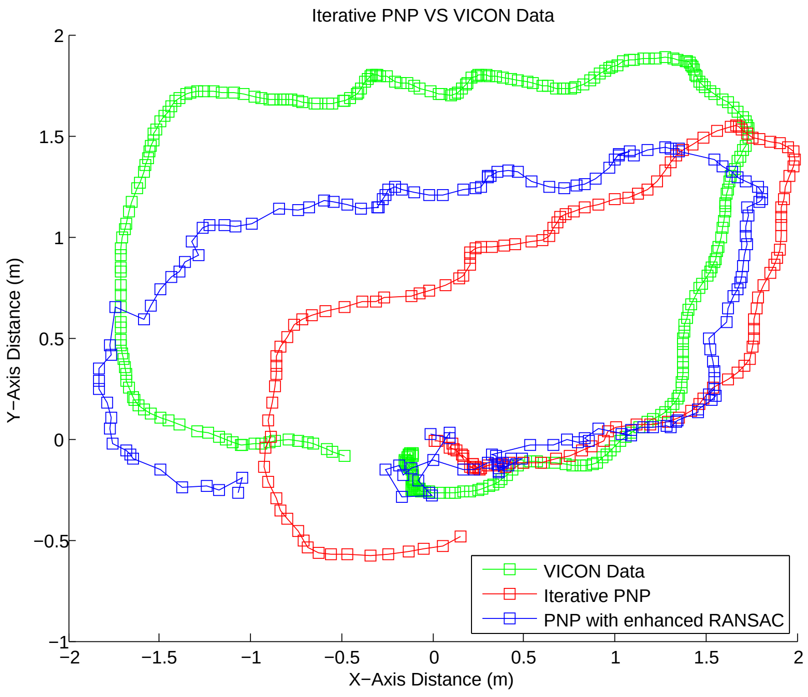


Figure 5.6: Iterative PNP vs VICON data.

rejection algorithm by increasing the threshold in which our algorithm accepts inliers, thereby increasing the strictness of our algorithm. The results were significantly better and they could be seen in the blue curve. From the results shown, I propose to obtain the camera pose estimation through 3D-to-2D reprojections instead of 3D-to-3D calculations.

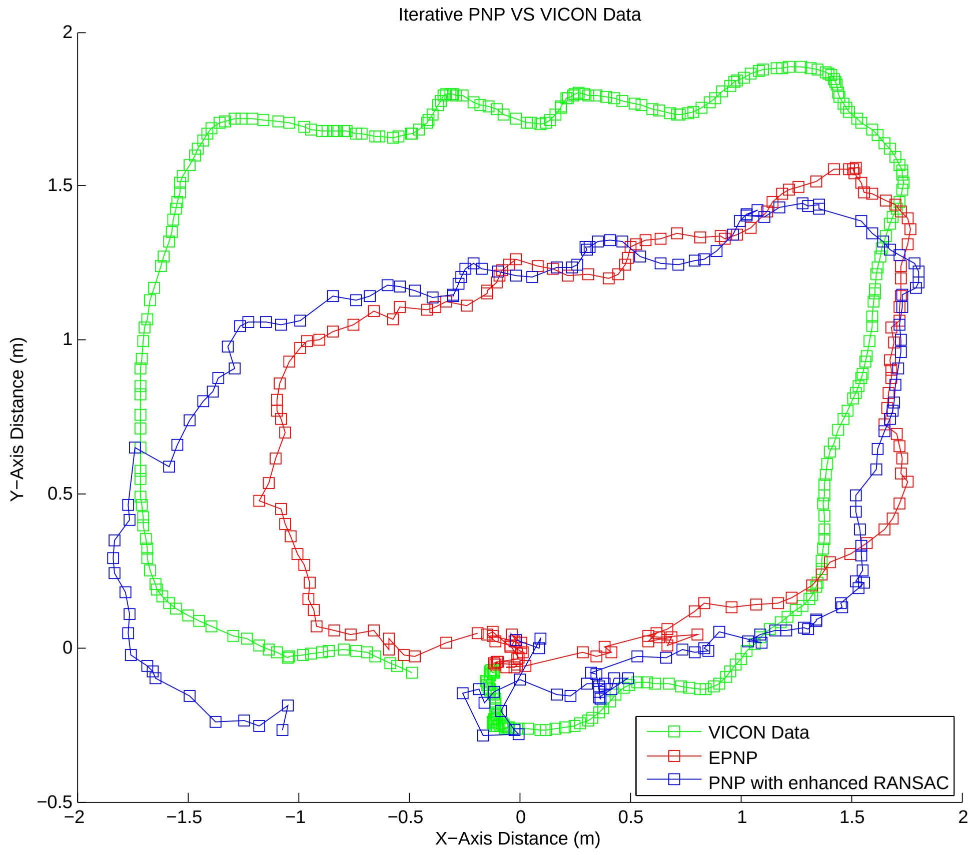


Figure 5.7: EPNP vs Iterative PNP.

We then implemented our algorithm with that using the EPNP [83] formulation which boast faster implementation and more accurate results. From Fig. 5.7, you may see the difference in performance of our algorithm with that of EPNP. Both algorithms perform well with our real-life test data. The improved algorithms now have possible spike errors of only about 30 cm and on average have errors less than 10 cm. However it was still not satisfactory against the VICON ground truth. Below, we create a list of possible factors that affects our test results as well as insights that we gained from our experiments.

Results Review

1. 3D-to-2D motion estimation has less drift when compared to 3D-to-3D motion estimation and posed less error from stereo triangulation due to only one use of triangulated 3D points during each frame of motion estimation.

2. Stereo triangulation possess significant error as mentioned in Section 3.2. Therefore, if the actual motion of the camera was too small, the error proportion might be too large and create erroneous estimates. This meant that our frequency for motion estimation was too high and we needed more motion between each image frame.

3. Our current camera baseline is only $0.1\ \mathrm{m}$. If we viewed image features that are too far, their 3D coordinate error will scale up exponentially. Therefore, we had to limit our feature usage to features that were calculated to be within our custom-built stereo camera's working range.

Figure 5.8: KITTI Vision Benchmark Suite urban images.

The KITTI Vision benchmark suite [84] is an open source set of data that provides real-world urban environment navigational data for benchmarking as shown in Fig. 5.8. The data suite consists of stereo images, GPS position, IMU data and a Velodyne laser scanner that are used to create a ground truth that researchers could use to compare their algorithm with. In this work, we made use of the data to verify our motion estimation algorithm to determine if the above mentioned problems might exist even with a different data set.

Fig. 5.9 shows our motion estimation algorithm using the KITTI data set and compared with the ground truth provided. This data set represents a motion of a vehicle traveling around $300\ \mathrm{m}$ in an urban environment. It can been seen that our motion estimation algorithm performed very well in this data set. The ground truth for this data set is the blue curve. We performed our algorithm twice on this data set and they are show as the red and green curve. Our data did have some wrong motion estimations and this could be see in as spikes in the curve. However, this could be remedied by using a stronger filter.

From the results shown, we can see that stereo vision has the advantage over monocular vision that both motion and structure are computed in the absolute scale which results in less drift if our parameters are tuned accurately. However, if the distance from the camera to the scene is much larger than the stereo camera's baseline, our case will degenerate from a stereo

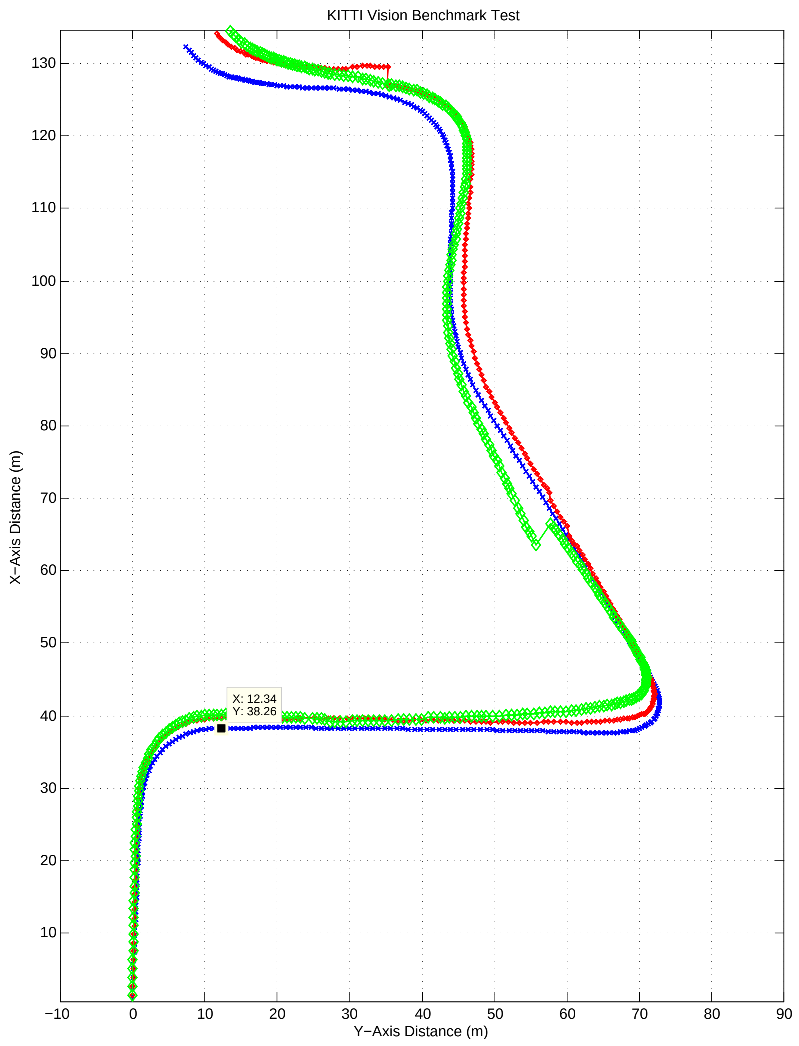


Figure 5.9: KITTI Vision Benchmark test.

visual odometry into a monocular visual odometry.

Chapter 6

2D Map Stitching

6.1 Introduction

In the recent International Micro Air Vehicle(IMAV) Competition 2014, we developed a methodology for real-time map stitching while the UAV is performing waypoint flight. Our methodology is very different from other map stitching methods as we emphasize on the real-time nature of performing map stitching in a disaster zone where rescuers will require a map of the area of operations fast. Current existing map stitching methods could not be achieved in real-time and will usually require desktop computers with high processing power. Our algorithm enables our UAV to capture the scene of the disaster area and create a stitched map while it is performing its flight around the area of interest.

During the competition, we were able to fly our UAV over a military disaster zone that was about $300\ \mathrm{m} \times 150\ \mathrm{m}$ and provide a stitched map of the area immediately after our UAV landed all in about $15\ \mathrm{mins}$. By achieving this requirement and other mission aspects, our team was then crowned the Champions of the IMAV Competition 2014 held in Delft, the Netherlands.

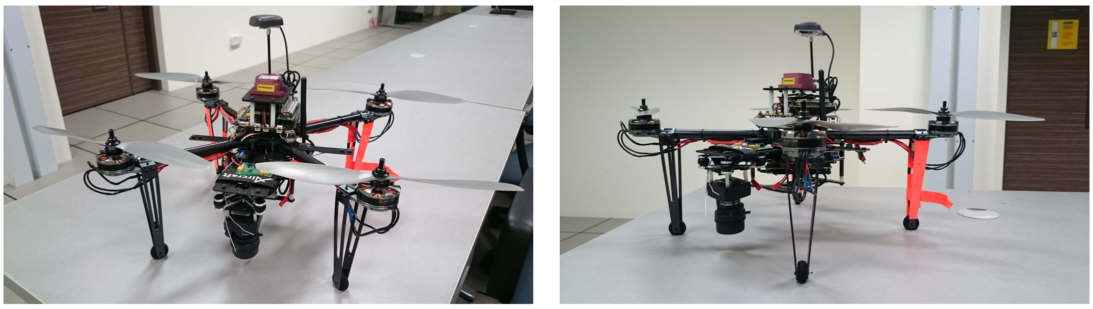


Figure 6.1: Quadrotor with downward facing camera.

The platform we used is a Quadrotor platform with a downward facing camera. The camera is hard-mounted to the UAV and is vibration damped using silicon vibration isolators. Our whole UAV system runs fully autonomously with our onboard system and it is shown in Fig. 6.1. The stitching algorithm runs on the onboard Intel $i7$ Quad-core Mastermind computer from Ascending Technologies Pte Ltd.

For our UAV to realize fully automatous flight and onboard image stitching, two processor layers are designed. One is used for automatic flight control which includes take-off, GPS waypoint tracking and landing. The other processor used for the vision algorithm captures and stores onboard images at 5 Hz. Given the village area to be covered in the competition, a sweep pattern path is generated online automatically from the GPS coordinates to maximize the image coverage. This coverage is calculated based on the pre-determined flight speed of the UAV and camera parameters while maintaining certain parameters such as image resolution of the village. Once the designated flight path has completed the last waypoint, the flight control processor will trigger the vision processor to start the image stitching process with the recorded onboard images. With such intelligence, the image stitching task can be realized near real time and is ready to be rendered to the ground control station once it returns home.

6.2 Homography-based Motion Model

Panoramic stitching relies on the concept behind projective transformation where transformation between two sets of points from two images are computed that represent the camera motion between the two images. This camera motion consists of rotation as well as translation and could be represented by the Homography [85] found based on features that correspond from one image to the next. We make use of this concept and apply it to map stitching of our target area.

In map stitching, we established a mathematical model that is able to map the pixel coordinates from one image onto another in 2D homogeneous coordinates $\mathbf{x_i}' = ({x_i}', {y_i}', 1)$ and $\mathbf{x_i} = (x_i, y_i, 1)$ such that,

$$ {x_i}' = H x_i \tag{6.1} $$

where Homography, H, is a 3×3 matrix of 9 elements.

Although the Homography matrix contains 9 entries, it represents transformation only up to scale and consist of 8 degrees of freedom in 2D transformation. A 2D point has two degrees of freedom corresponding to its x and y coordinates. Since each point-to-point correspondence accounts for two constraints, it can be seen that a total of four corresponding points are needed to solve and constraint H fully.

In an image set, it usually consists of many features points that could be detected and tracked across the different images. We have more feature points than needed to calculate the Homography matrix but much of these feature points are noisy and could represent bad matches. If we use all the features to calculate Homography, we might not get a solution which represents the best projective transformation for our features. As such, we implement RANSAC [76] with the large number of feature points and set a threshold value for a maximum re-projection error calculated in homogeneous coordinates to treat a point correspondence as an inlier as shown below,

$$\left\| {x_i}' - H \times x_i \right\| > Reprojection\ Error\ Threshold \tag{6.2}$$

In the RANSAC implementation, random sets of four corresponding points are chosen to estimate the homography matrix using a simple least-squares algorithm. Using the homography matrix estimate, we then compute the inlier ratio of the computed homography based on the re-projection error threshold shown above. The eventual best subset is then used to produce the initial estimate of the homography matrix with its set of inliers. Our aim is then to minimize the back-projection error,

$$\sum_i \left({x_i}' - \frac{h_{11}x_i + h_{12}y_i + h_{13}}{h_{31}x_i + h_{32}y_i + h_{33}} \right)^2 + \left({y_i}' - \frac{h_{21}x_i + h_{22}y_i + h_{23}}{h_{31}x_i + h_{32}y_i + h_{33}} \right)^2 \tag{6.3}$$

Finally, the computed homography is refined further with the Levenberg-Marquardt method [86] to further reduce the re-projection error.

One very important aspect to note is that in the calculation of homography matrix, the assumption is that the features found in the images all belong to the same plane. Therefore, in map stitching, it is of the upmost importance to include a robust calculation method which rejects outliers. This is due to the fact that in real UAV flight, the UAV flies over objects that project out from the ground plane such as trees and buildings as shown in Fig. 6.2.

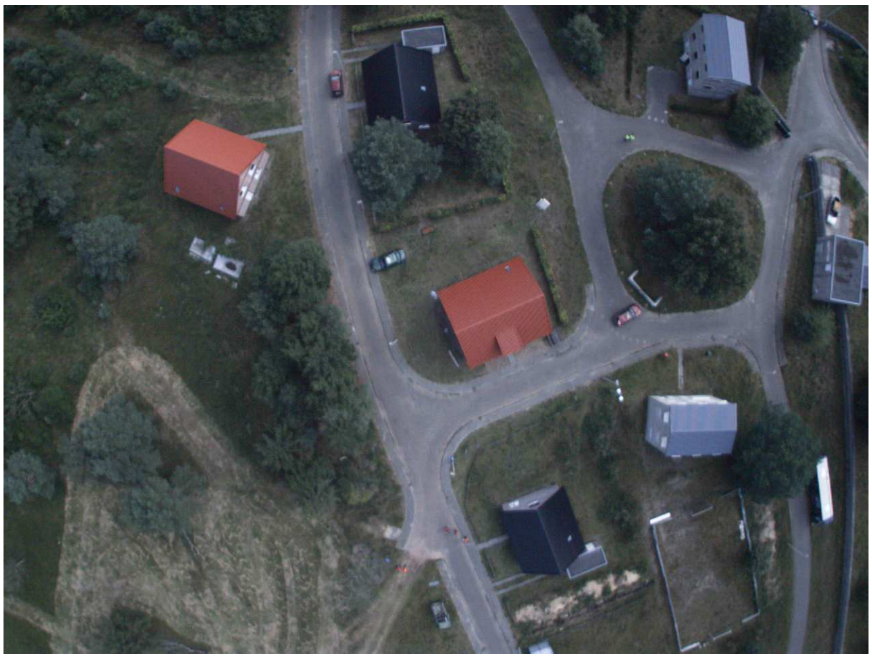


Figure 6.2: Military village in the Netherlands.

If a set of features belong in a fixed plane, they can be used to obtain a homography matrix which composes elements of rotation, translation as well as skew factor of an image with respect to the previous image. We then use the homography matrix to map each of the subsequent images onto the same main image to create a stitched map.

6.3 Onboard Stitching Implementation

Our methodology for solving the onboard stitching problem is to optimize and create a stitching algorithm that is robust and reliable while not including time consuming enhancements that aim to beautify the stitched map usually found in panoramic stitching algorithms. Some of the enhancement algorithms that we did not implement are gain compensation, multi-band blending and seam line detection where additional computational time will be imminent. Therefore, our stitched image may not be as "beautiful" as some other panorama stitching but it gives us an instant result suitable for use by disaster response teams.

6.3.1 First Image Initialization

Prior to starting the stitching process, there is a need to initialize the first image where the stitching algorithm takes reference from. This initial image is where the stitching will propagate from and is important in terms of orientation and location on the stitching canvas. During stitching, should you not know how the stitching movement will be, you will have to create a general canvas that can accommodate any form of movement for the images that are to come. This will usually be inefficient as the amount of computer memory used to create this canvas can be quite large and will generally slow down the whole computational process.

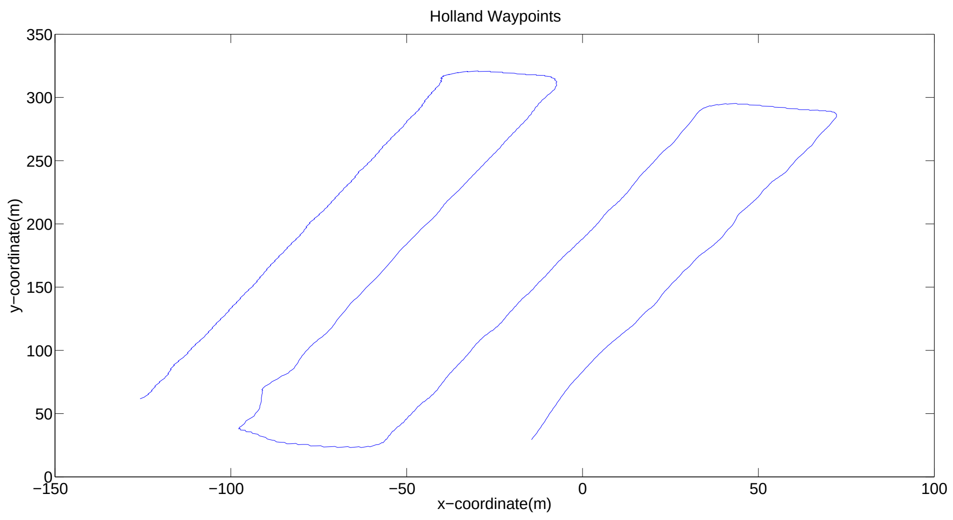


Figure 6.3: Waypoint generation for the UAV.

There are many map stitching methods which rely on different kinds of map initialization. For our implementation, we are able to automatically generate waypoints in our area of operations which creates a “zig-zig” path to efficiently cover the area of interest as shown in Fig. 6.3. We are then able to set the waypoint which we prefer as the initial start point for our stitching algorithm. Our stitching algorithm initializes the first image as the origin for 2D map building. To achieve fully autonomous flights with our UAV, we are able to know the pose information of the UAV through the readings from our IMU sensor. With these initial states, we can use it to correct the first image to be near parallel with the ground plane.

It is important to note that initialization is particularly important as the map is built upon the initialized image. Should the initial image be skewed, the rest of the map will be affected and propagate from this skew factor. Therefore, it is of great importance to initialize the first image well using our onboard IMU reading to bring the image as close to being parallel with the ground plane. After this initialization, we then carry out frame by frame feature detection and tracking.

6.3.2 Kanade Lucas Tomasi(KLT) Feature Detection and Tracking

In computer vision literature, there are many feature detection and matching algorithms such as that of SURF, SIFT and Harris corner detector. However, these feature detectors and matchers perform relatively slower as compared to the KLT Tracker. We have tested a lot of the feature detection algorithms and though they can be very robust and accurate, their calculation time was too much for our onboard system to handle. As such, we decided to perform our onboard stitching algorithm with the aid of the KLT tracker which shows good performance on top of being computationally efficient. The KLT tracker uses optical flow tracking that is calculated over different gaussian pyramids of the two images and it is proven to work well even in areas that seem homogeneous to the human eye such as that of grass patches and also foliage areas as shown in Fig. 6.4.

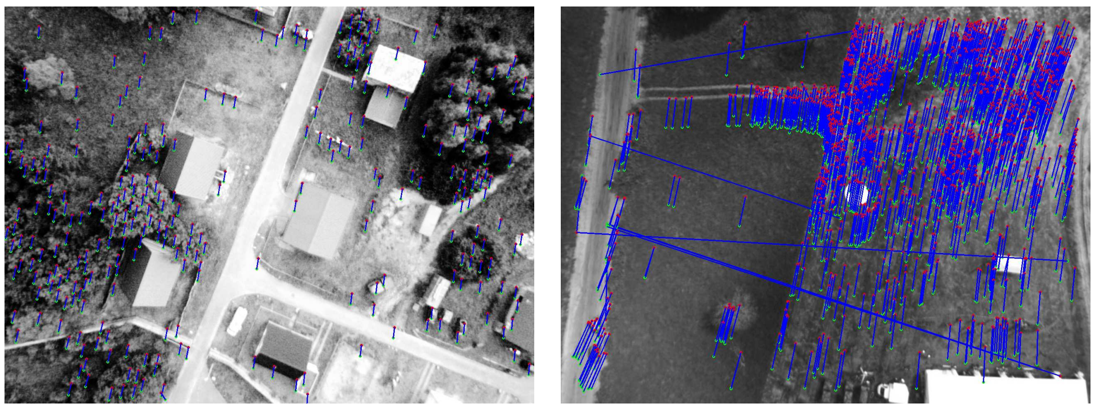


Figure 6.4: Left: KLT Optical flow, Right: FAST feature detector.

One important factor for using the KLT tracker effectively is that the frequency of images used should be relatively high such as that of $5Hz$ and above. During the running of the algorithm, should the image capturing frequency be too low, there are cases where the KLT tracker fails due to large movement. This can be seen in Fig. 6.5.

6.3.3 Homography Calculation, Updating and Failure Checks

After the detection and tracking of feature points in our image stream, we can then use them to calculate the homography matrix as shown in Sect. 6.2. The calculated homography matrix is updated through a simple matrix multiplication process and used for further stitching of subsequent images as show in Eqn. 6.4. However, before the update could be done, there are a set of checks that will have to be performed prior to the update process.

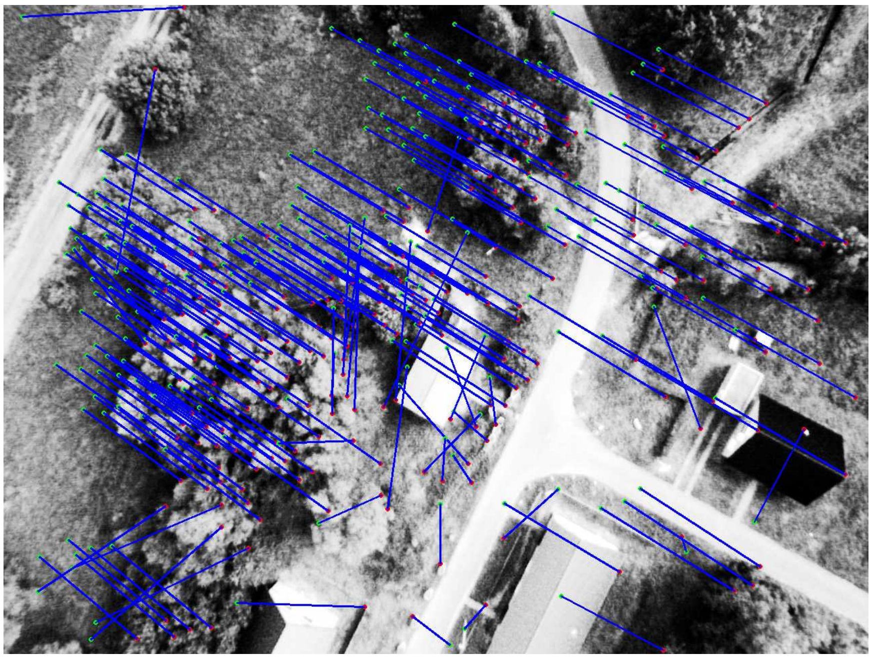


Figure 6.5: Erroneous optical flow tracking.

$$H_{updated} = H_{calculated}H_{previous} \tag{6.4}$$

During the calculations for the homography matrix, regardless of how robust the algorithm is, there are times when the calculation fails either due to a very noisy image or if the motion of the over system is too large for optical flow to manage. During these cases, we have developed a failsafe in the forms of homography matrix failure checks.

Firstly, we performed a check on the overall change in image size as compared to the original image. We allow an image size change of $\pm 20\%$ due to the skewing of the image and also any enlargement or shrinking that should occur due to slight height differences while our UAV system was flying.

Secondly, we performed a check on the overall translation of the image as compared to the previous image. This was done by calculating the centroid of the image that has been projective transformed by the calculated homography matrix. As we run our algorithm at $5Hz$ frequency, we expect the translation of the image to be very small. Therefore, we allow the translation to be less than half the diagonal distance of the original image.

The two failure checks are summarized in the pseudo code shown in Algorithm 1. If the checks are passed by the algorithm, the algorithm will continue to run as in the next section.

However, should any of the failure checks fail, an interim homography matrix will be calculated and used. This interim homography matrix is calculated from IMU states which encompasses the euler angles as well as the GPS coordinates.

Algorithm 1 Homography Failure Checks

```
1: procedure IMAGE SIZE CHECK
2:     $k = 1$ or 0 ← Check results
3:     Vector(points) ← Projective Transform from H
4:     Area ratio, A ← Area between Vector(points) and original
5:     if $|1 - A| \geq 0.2$ then
6:         $k \leftarrow 0$
7:     else
8:         if $|1 - A| \leq 0.2$ then
9:             $k \leftarrow 1$
10: procedure IMAGE TRANSLATION CHECK
11:     Centroid Difference, D ← Distance between centroid of Vector(points) and original
12:     if $|D| \geq 0.5 \times diag(img)$ then
13:         $k \leftarrow 0$
14:     else
15:         if $|D| \leq 0.5 \times diag(img)$ then
16:             $k \leftarrow 1$
17: procedure FAILURE CHECK RECTIFICATION
18:     $H_{IMU}$ ← IMU states input
19:     if $k = 1$ then
20:         $continue$;
21:     else
22:         if $k = 0$ then
23:             $IMU - based\ Homography\ Calculation, H_{IMU}$
```

6.3.4 Warping and Cropping of Images

After the successful calculation of the homography matrix, what is in line next is the warping of the current image using the homography matrix. The source image will have a perspective transform performed using the specified matrix to give a resulting destination image as shown in Eqn. 6.5. The perspective transform will map the current image onto a canvas which is a previously defined image of $(5 \times ImgWidth) \times (5 \times ImgHeight)$. Since we know the general movement of the images, we can reduce the canvas size significantly and improve computational time. During the warping of the new images onto the canvas, they are warped onto the top of the stitched map which means that the final map always has the newest view of the terrain.

$$dst(x, y) = src\left(\frac{M_{11}x + M_{12}y + M_{13}}{M_{31}x + M_{32}y + M_{33}}, \frac{M_{21}x + M_{22}y + M_{23}}{M_{31}x + M_{32}y + M_{33}}\right) \tag{6.5}$$

After all the images have been warped onto the canvas, you will usually see that there are big patches of the canvas that was not mapped with any images. If these areas are not removed, they will usually take up additional memory and result in a bigger than needed image. In order to remove this areas, we performed a threshold of the whole image to obtain the area of the final stitched map and create a bounding box to extract the minimal image size that could encompass the final stitched image. To obtain the bounding box, we converted the thresholded image into a contour where we could obtain the vertices of the final stitched image. With the vertices, we then created a bounding box that could cover the require area. In Fig. 6.6 , we see the image prior to performing cropping.

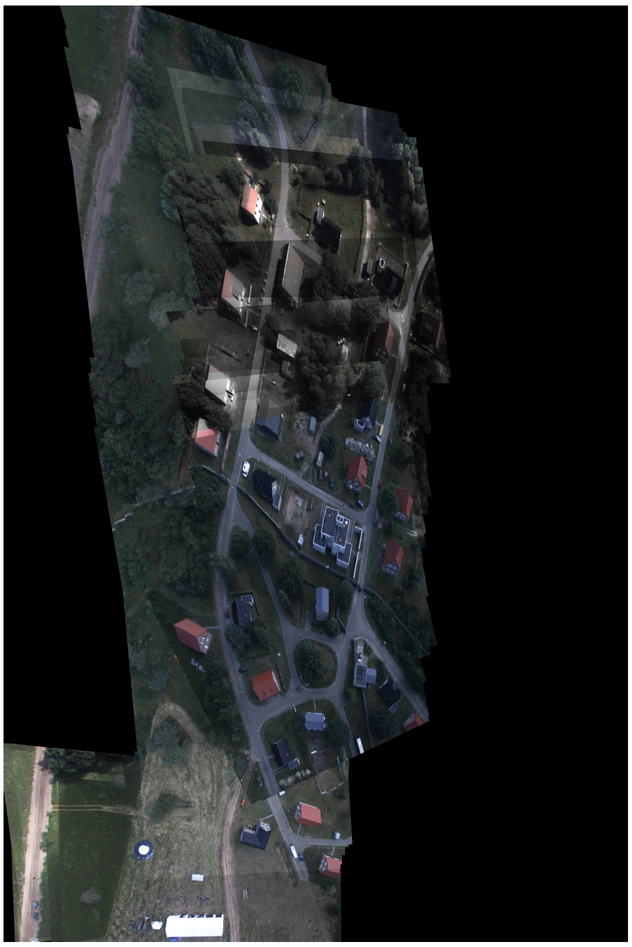


Figure 6.6: Map with uncropped boarder.

6.4 Stitching Performance

During the IMAV 2014 competition, our team was the only team that was able to perform on-board stitching from our UAV where we could submit the stitched map of our area of operations immediately after our UAV landed. Other teams still needed to download the images to a desktop computer which they setup and perform map stitching for up to 10 mins before they could submit their stitched map.

Our success lies in the strategy in which we chose to accomplish our task. We firstly programmed the UAV to perform waypoint flight over our area of interests and obtain initialization parameters for our determined first waypoint then proceed to capture images at a frequency of 5 Hz. When the UAV finishes its last waypoint over the area of interest, it starts to trigger the "return home" command. It is at this point in time that our program consolidates the images and UAV states and starts the stitching process. As the UAV takes about 2 mins to return to its home and perform the "landing" command, our program makes use of this time to stitch the images together. Even after landing, the program onboard still runs to complete our map stitching. Our full stitched map is usually ready after retrieving the UAV and connecting it to a monitor for display purposes.

Table 6.1: Computational time for stitching using different detectors & matcher sets.

Detector	**Descriptor**	**Matcher**	**Time**
FAST	BRIEF	Brute Force	0.118 s
GFTT	Optical Flow	Optical Flow	0.153 s
SURF	SURF	FLANN	0.292 s

In Table 6.1, it shows the different sets of detector, descriptor and matcher an their computational times. It could be seen that the FAST detector set has the fastest computational time but we have noticed that its performance is not as ideal as compared to the optical flow method. During our flight over our area of interest, we have taken over 1000 images and performed our stitching algorithm based on this number of images. The total time taken for stitching our map is 153 s which translate to only 2 mins 33 secs which is a very fast time.

6.5 Conclusion

We performed the above algorithm in a few real flight scenarios namely in different terrains in Singapore. Some of the terrain include a huge field of grass and sandy terrain with some urban features. We were unable to fly over real buildings due to safety concerns which means that the flight during the IMAV 2014 competition over an built-up urban village is the first time we have done so. From Fig. 6.7 you can see that the results we obtained are very good considering the quick computational time as well as the algorithm being run onboard a UAV. The areas covered are about $200\ \mathrm{m} \times 200\ \mathrm{m}$ each.

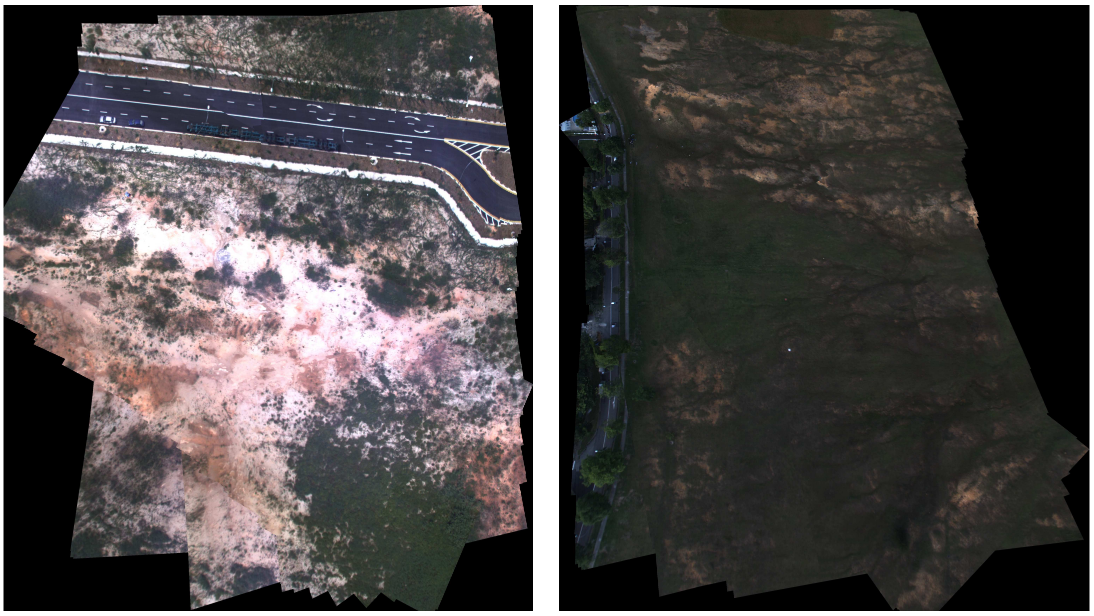


Figure 6.7: Left: Tuas stitched map, Right: Blackmore Drive stitched map.

During the IMAV 2014 competition, we are required to fly over the military village of Oostdorp, the Netherlands. This was quite an undertaking as we are required to fly over an area where there are buildings and trees as high as $15\ \mathrm{m}$ in height. As previously mentioned in Sect. 6.2, the homography calculation requires the observed points to be part of the same plane. However, with buildings and trees in the vicinity, this is obviously not the case. However, our algorithm, with its robust RANSAC-based homography calculation, was able to reject those features that were detected on the buildings and trees and produce a stitched map shown in Fig. **??**. We were able to produce the stitched map immediately after our UAV landed and thus enabled our team to obtain really high scores. Part of the requirement of the stitched map was to be able to allow the users to identify potential blockages or obstacles that might be a hindrance to rescue workers. In Fig. 6.9, you can observe that we were able to detect roadblocks and obstacles in the stitched map very clearly.

One of the main problems we encountered during our flight in the Netherlands was that due to high winds, the clouds move at a very high rate above our UAV. This creates ever changing exposure levels not in the whole image but in patches found in the image as shown in Fig. 6.10. This resulted in our stitched map being not very pleasing to the eyes as there are obvious changes

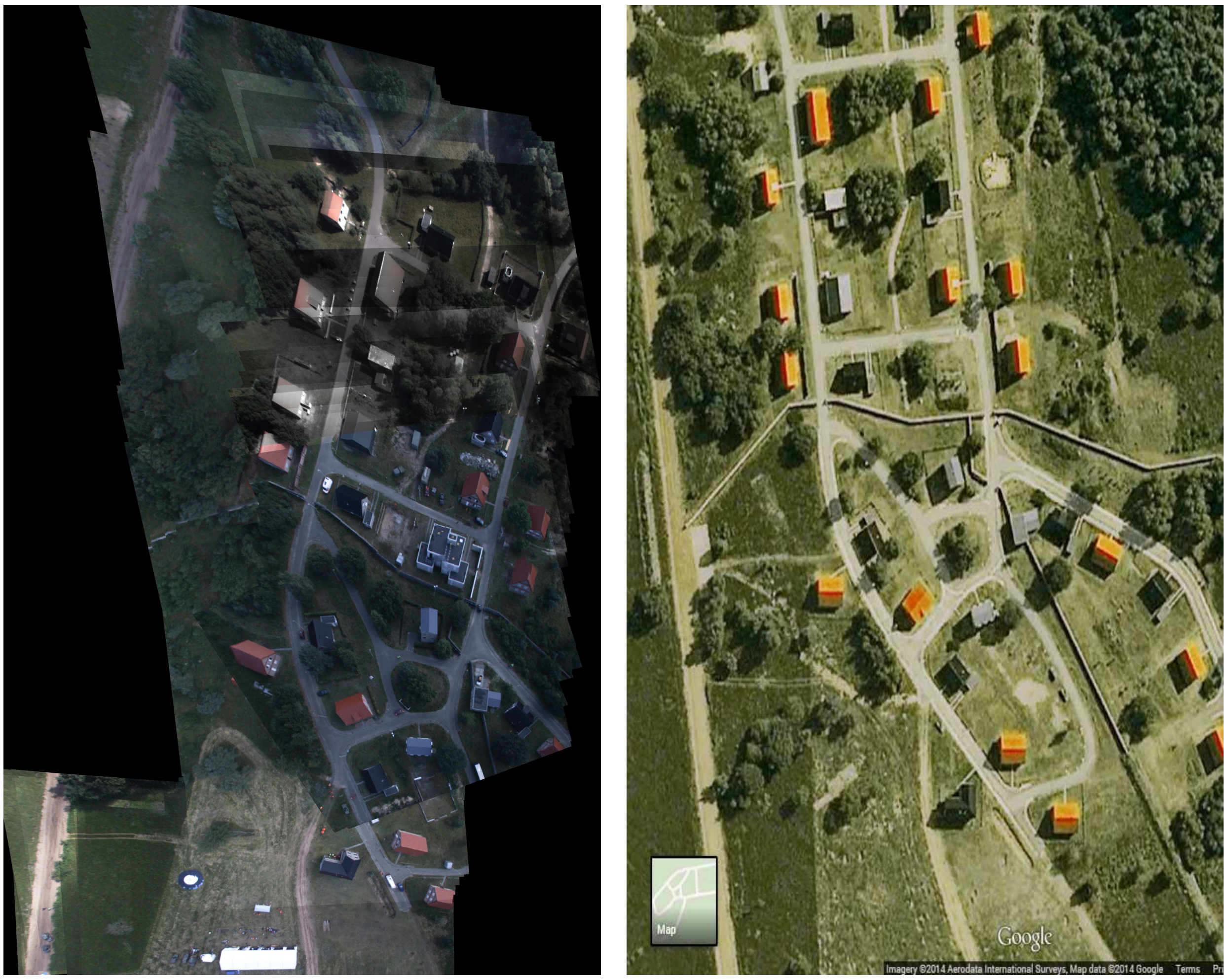

Map
Google
Imagery ©2014 Aerodata International Surveys, Map data ©2014 Google Terms

Figure 6.8: Left: Final stitched map, Right: Google map view.

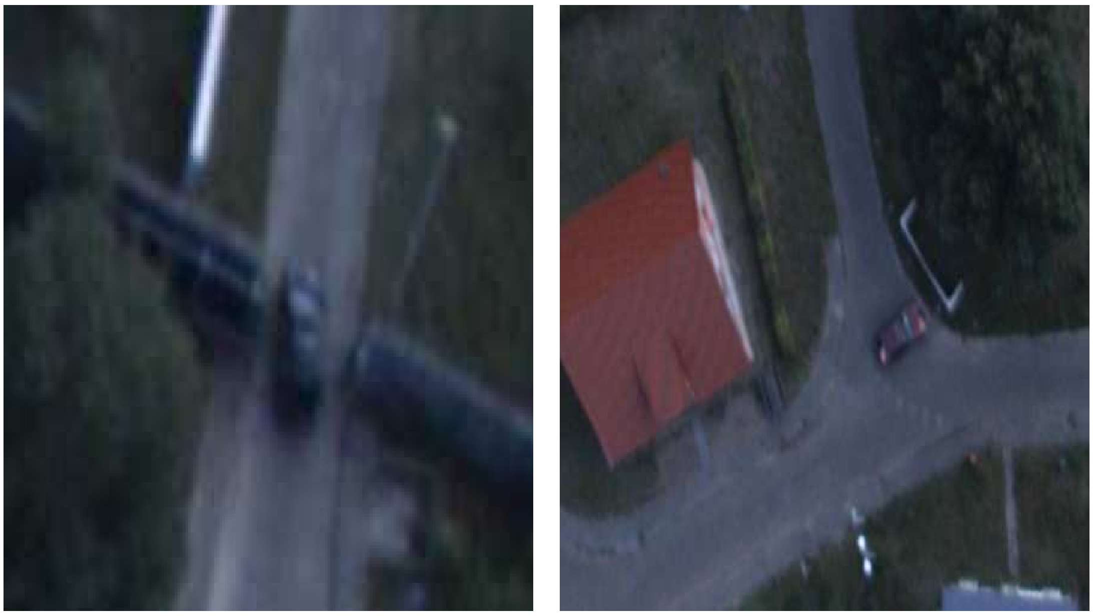


Figure 6.9: Detected obstacles in the stitched map.

in exposure during moments where the cloud cover cleared and resulted in the sun shining very brightly on the landscape. Our next step will be to solve this using adaptive exposure compensation to create a stitched map that does not exhibit such high exposure contrasts that sometimes might be distracting to the user.

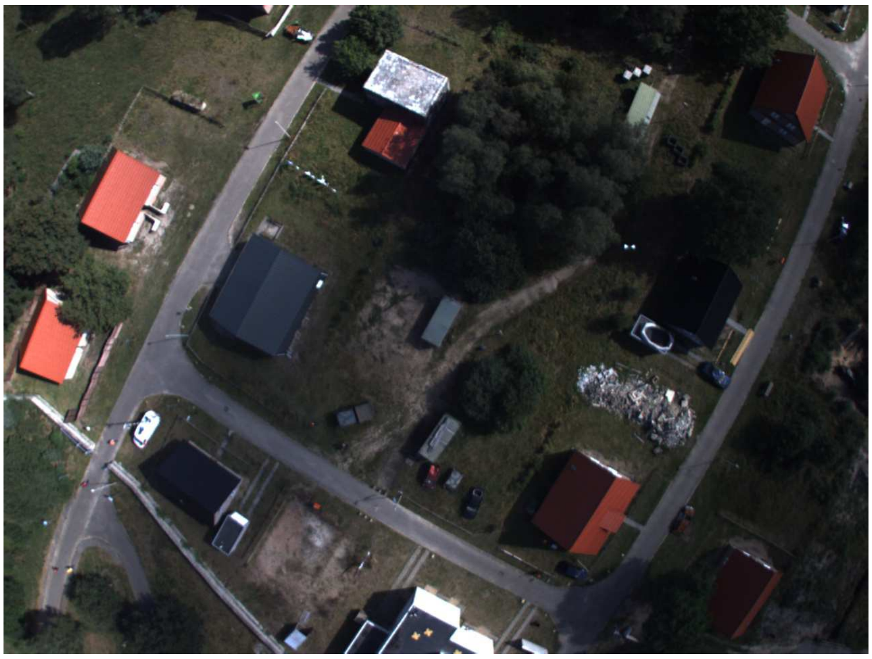


Figure 6.10: Uneven exposure due to clouds.

The exposure change also affected the optical flow feature detection. It violates the brightness constancy assumption that is specified in the Lucas-Kanade Optical flow method. However, due to the application of the gaussian pyramids, the high frequency of images used as well as the RANSAC outlier rejection scheme, our algorithm was able to reject the erroneous feature points.

Chapter 7

Conclusions & Future work

7.1 Conclusions

This thesis focused on the development of unmanned aerial vehicles with the capability for obstacle detection, motion estimation and map building using visual measurements. It presented the theoretical and practical aspects of these capabilities in urban environments through many scenarios.

The major contribution of this thesis is the description of a comprehensive UAV system that is capable of urban exploration. This thesis developed many algorithms that created various functionalities for UAVs keeping in mind that all algorithms must be able to run onboard the UAV and with real-time computation. In urban or outdoor UAV missions, it is believed that communication between the UAV and the ground control system might be severed due to range or line-of-sight issues. This brings out the need for the UAV to process all algorithms onboard without relying on off-board computational power.

In this thesis, we explored the development of the quadrotor UAV and an unconventional UAV, U-Lion, which provide the basis of UAV urban exploration. Both UAV platforms were self-constructed with their individual avionics systems built onboard. Their respective flight control laws and navigational capabilities were implemented and tested rigorously. Both UAVs have been used to take part in international UAV competitions such as that of the International Micro Air Vehicle Competition 2014 (IMAV) in the Netherlands and the UAV Grand Prix 2013 (UAVGP) in Beijing, China. They have also been used to participate in national competitions such as the Singapore Amazing Flying Machine Competition 2014 (SAFMC). Both UAV platforms achieved many awards in all the mentioned competitions.

Above the platform and its controls, lies the visual-based algorithms which provide a top level intelligence and mission completion capability. The visual-based algorithms developed in this thesis are all independent from the UAV platform itself. They can be applied to any UAV platform with the required sensor configuration and onboard processing power.

The visual-based algorithms developed assisted the UAV system to detect and classify obstacles while performing navigation based on the stereo camera system. Some novel visual-based algorithms that the thesis detailed are the fast onboard stitching algorithm and the active stereo vision algorithm that could be used in featureless environments. Both of which have been implemented during international and national competitions.

7.2 Future Work

Although this thesis depicts a comprehensive UAV system capable of urban navigation, it is in no way exhaustive. I truly believe that there are still plenty of room for improvement as some of the developed functions can still be built to be more robust and intelligent.

The following is a list of future work that could be performed to enable more functions for the UAV.

1. Unconventional UAV:

   The development of U-Lion is still at its infant stages. The structure and mechanisms of U-Lion have been designed and optimized but we have yet to perform autonomous flight using the unconventional UAV. There are still much more work we can perform on U-Lion to enable it to perform autonomous flight and autonomous transition. It is a one of a kind hybrid aircraft and there is still a lot of potential yet to be unleashed from it.

2. Obstacle Avoidance Algorithm:

   During the discussion of obstacle detection in Chapter 3, we did not cover in detail about the possible obstacle avoidance strategies and algorithms that could be developed. This topic is a huge topic and it has been a major deficient capability among all of the UAVs in recent development. There is definitely work that could be done to implement novel avoidance strategies into our UAVs increase the intelligence of our system.

3. GPS-Less Motion Estimation:

   GPS-less navigation has been of great interest in potential applications of UAV systems. However, although many have implemented GPS-less navigation in the form of visual-based or lidar-based navigation, not one system could achieve an all rounded solution to this ever present topic. There are still many attractive and unique solutions to the GPS-less navigation problem.

4. Path Planning Algorithms:

   The current development of UAVs in this thesis does not include path planning algorithms. The literature behind path planning is very vast. It includes search algorithms such as computational geometry-based approaches such as cell decomposition, road map and potential field methods. These then leads to the use of multi-tiered path planning where different algorithms are used for global, local and reactive search respectively. I will like to work on a basic path planning algorithm that could be used to allow the Quadrotor to navigate through an urban environment and use the obstacle avoidance technology effectively.

Bibliography

[1] A. Ollero and L. Merino, “Control and perception techniques for aerial robotics,” *Annual reviews in Control*, vol. 28, no. 2, pp. 167–178, 2004.

[2] Z. Shiyu, D. Xiangxu, C. Jinqiang, Z. Y. Ang, L. Feng, P. Kemao, B. Chen, and T. Lee, “Design and implementation of homography-based vision-aided inertial navigation of uavs,” in *Control Conference (CCC), 2013 32nd Chinese*, pp. 5101–5106, IEEE, 2013.

[3] R. M. Murray, “Recent research in cooperative control of multivehicle systems,” *Journal of Dynamic Systems, Measurement, and Control*, vol. 129, no. 5, pp. 571–583, 2007.

[4] F. Lin, K. Peng, X. Dong, S. Zhao, and B. M. Chen, “Vision-based formation for uavs,” in *Control & Automation (ICCA), 11th IEEE International Conference on*, pp. 1375–1380, IEEE, 2014.

[5] A. Tayebi and S. McGilvray, “Attitude stabilization of a vtol quadrotor aircraft,” *Control Systems Technology, IEEE Transactions on*, vol. 14, no. 3, pp. 562–571, 2006.

[6] R. Roberto Sabatini, M. Richardson, C. Bartel, T. Shaid, and S. Ramasamy, “A low-cost vision based navigation system for small size unmanned aerial vehicle applications,” *J Aeronaut Aerospace Eng*, vol. 2, no. 110, p. 2, 2013.

[7] G. Chowdhary, E. N. Johnson, D. Magree, A. Wu, and A. Shein, “Gps-denied indoor and outdoor monocular vision aided navigation and control of unmanned aircraft,” *Journal of Field Robotics*, vol. 30, no. 3, pp. 415–438, 2013.

[8] R. Sabatini, S. Ramasamy, A. Gardi, and L. R. Salazar, “Low-cost sensors data fusion for small size unmanned aerial vehicles navigation and guidance,” *International Journal of Unmanned Systems Engineering*, vol. 1, no. 3, pp. 16–47, 2013.

[9] W. Hu, T. Tan, L. Wang, and S. Maybank, "A survey on visual surveillance of object motion and behaviors," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 34, no. 3, pp. 334–352, 2004.

[10] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision.* Cambridge university press, 2003.

[11] E. D. Dickmanns and F.-R. Schell, "Autonomous landing of airplanes by dynamic machine vision," in *Applications of Computer Vision, Proceedings, 1992., IEEE Workshop on*, pp. 172–179, IEEE, 1992.

[12] S. Hrabar, G. Sukhatme, P. Corke, K. Usher, and J. Roberts, "Combined optic-flow and stereo-based navigation of urban canyons for a uav," in *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*, pp. 3309–3316, IEEE, 2005.

[13] E. Trucco and A. Verri, *Introductory techniques for 3-D computer vision*, vol. 201. Prentice Hall Englewood Cliffs, 1998.

[14] B. K. Horn and B. G. Schunck, "Determining optical flow," in *1981 Technical Symposium East*, pp. 319–331, International Society for Optics and Photonics, 1981.

[15] B. D. Lucas, T. Kanade, *et al.*, "An iterative image registration technique with an application to stereo vision.," in *IJCAI*, vol. 81, pp. 674–679, 1981.

[16] C. Harris and M. Stephens, "A combined corner and edge detector.," in *Alvey vision conference*, vol. 15, p. 50, Manchester, UK, 1988.

[17] T. Lindeberg, "Scale-space for discrete signals," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 12, no. 3, pp. 234–254, 1990.

[18] K. Mikolajczyk and C. Schmid, "An affine invariant interest point detector," in *Computer VisionECCV 2002*, pp. 128–142, Springer, 2002.

[19] K. Mikolajczyk and C. Schmid, "A performance evaluation of local descriptors," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 27, no. 10, pp. 1615–1630, 2005.

[20] D. G. Lowe, “Object recognition from local scale-invariant features,” in *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, vol. 2, pp. 1150–1157, Ieee, 1999.

[21] H. Bay, T. Tuytelaars, and L. Van Gool, “Surf: Speeded up robust features,” in *Computer Vision–ECCV 2006*, pp. 404–417, Springer, 2006.

[22] Y. Koren and J. Borenstein, “Potential field methods and their inherent limitations for mobile robot navigation,” in *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on*, pp. 1398–1404, IEEE, 1991.

[23] G. Hoffmann, D. G. Rajnarayan, S. L. Waslander, D. Dostal, J. S. Jang, and C. J. Tomlin, “The stanford testbed of autonomous rotorcraft for multi agent control (starmac),” in *Digital Avionics Systems Conference, 2004. DASC 04. The 23rd*, vol. 2, pp. 12–E, IEEE, 2004.

[24] H. Huang, G. M. Hoffmann, S. L. Waslander, and C. J. Tomlin, “Aerodynamics and control of autonomous quadrotor helicopters in aggressive maneuvering,” in *Robotics and Automation, 2009. ICRA’09. IEEE International Conference on*, pp. 3277–3282, IEEE, 2009.

[25] G. M. Hoffmann, H. Huang, S. L. Waslander, and C. J. Tomlin, “Quadrotor helicopter flight dynamics and control: Theory and experiment,” in *Proc. of the AIAA Guidance, Navigation, and Control Conference*, vol. 2, 2007.

[26] G. M. Hoffmann, H. Huang, S. L. Waslander, and C. J. Tomlin, “Precision flight control for a multi-vehicle quadrotor helicopter testbed,” *Control engineering practice*, vol. 19, no. 9, pp. 1023–1036, 2011.

[27] P. Pounds, R. Mahony, P. Hynes, and J. Roberts, “Design of a four-rotor aerial robot,” in *Australasian Conference on Robotics and Automation*, pp. 145–150, 2002.

[28] P. Pounds, R. Mahony, and P. Corke, “Modelling and control of a large quadrotor robot,” *Control Engineering Practice*, vol. 18, no. 7, pp. 691–699, 2010.

[29] K. Nonami, F. Kendoul, S. Suzuki, W. Wang, and D. Nakazawa, “Mathematical modeling and nonlinear control of vtol aerial vehicles,” in *Autonomous Flying Robots*, pp. 161–193, Springer, 2010.

[30] M. Valenti, B. Bethke, D. Dale, A. Frank, J. McGrew, S. Ahrens, J. P. How, and J. Vian, "The mit indoor multi-vehicle flight testbed," in *Robotics and Automation, 2007 IEEE International Conference on*, pp. 2758–2759, IEEE, 2007.

[31] J. Kim, M.-S. Kang, and S. Park, "Accurate modeling and robust hovering control for a quad-rotor vtol aircraft," in *Selected papers from the 2nd International Symposium on UAVs, Reno, Nevada, USA June 8–10, 2009*, pp. 9–26, Springer, 2010.

[32] C. Nicol, C. Macnab, and A. Ramirez-Serrano, "Robust adaptive control of a quadrotor helicopter," *Mechatronics*, vol. 21, no. 6, pp. 927–938, 2011.

[33] T. Madani and A. Benallegue, "Control of a quadrotor mini-helicopter via full state backstepping technique," in *Decision and Control, 2006 45th IEEE Conference on*, pp. 1515–1520, IEEE, 2006.

[34] F. Wang, T. Wang, B. M. Chen, and T. H. Lee, "An indoor unmanned coaxial rotorcraft system with vision positioning," in *Control and Automation (ICCA), 2010 8th IEEE International Conference on*, pp. 291–296, IEEE, 2010.

[35] X. Dong, M. Dong, B. Wang, B. M. Chen, and T. H. Lee, "A comprehensive software system architecture for unmanned aerial vehicles," in *Service Operations, Logistics, and Informatics (SOLI), 2011 IEEE International Conference on*, pp. 595–600, IEEE, 2011.

[36] X. Dong, B. M. Chen, G. Cai, H. Lin, and T. H. Lee, "Development of a comprehensive software system for implementing cooperative control of multiple unmanned aerial vehicles," in *Control and Automation, 2009. ICCA 2009. IEEE International Conference on*, pp. 1629–1634, IEEE, 2009.

[37] M. Dong, B. M. Chen, G. Cai, and K. Peng, "Development of a real-time onboard and ground station software system for a uav helicopter," *Journal of Aerospace Computing, Information, and Communication*, vol. 4, no. 8, pp. 933–955, 2007.

[38] C. GUOWEI, *Development of small-scale unmanned-aerial-vehicle helicopter systems*. PhD thesis, 2009.

[39] L. Besnard, Y. B. Shtessel, and B. Landrum, "Control of a quadrotor vehicle using sliding mode disturbance observer," in *American Control Conference, 2007. ACC'07*, pp. 5230–5235, IEEE, 2007.

[40] S. A. Raza, *Design and control of a quadrotor unmanned aerial vehicle*. University of Ottawa, 2010.

[41] R. W. Beard, "Quadrotor dynamics and control," *Brigham Young University*, 2008.

[42] G. Cai, B. M. Chen, and T. H. Lee, *Unmanned rotorcraft systems*. Springer, 2011.

[43] C. M. Harris, A. G. Piersol, and T. L. Paez, *Harris' shock and vibration handbook*, vol. 5. McGraw-Hill New York, 2002.

[44] M. B. Tischler and R. K. Remple, "Aircraft and rotorcraft system identification," *AIAA Education Series. American Institute of Aeronautics and Astronautics, New York*, 2006.

[45] B. Peeters and G. De Roeck, "Reference-based stochastic subspace identification for output-only modal analysis," *Mechanical systems and signal processing*, vol. 13, no. 6, pp. 855–878, 1999.

[46] J. A. Guerrero, R. Lozano, G. Romero, D. Lara-Alabazares, and K. Wong, "Robust control design based on sliding mode control for hover flight of a mini tail-sitter unmanned aerial vehicle," in *Industrial Electronics, 2009. IECON'09. 35th Annual Conference of IEEE*, pp. 2342–2347, IEEE, 2009.

[47] K. Jackson, J. Li, E. Timmons, and J. Wallace, "icaruslabs: An adventure in crowdsourcing," *Unmanned Systems*, vol. 1, no. 02, pp. 199–209, 2013.

[48] E. Çetinsoy, E. Sirimoglu, K. T. Oner, C. Hançer, M. Unel, M. F. Aksit, I. Kandemir, and K. Gulez, "Design and development of a tilt-wing uav," *Turk. J. Elec. Eng. & Comp. Sci*, vol. 19, no. 5, 2011.

[49] T. J. Mueller, *Fixed and flapping wing aerodynamics for micro air vehicle applications*, vol. 195. AIAA, 2001.

[50] W. Shyy, H. Aono, S. K. Chimakurthi, P. Trizila, C.-K. Kang, C. E. Cesnik, and H. Liu, "Recent progress in flapping wing aerodynamics and aeroelasticity," *Progress in Aerospace Sciences*, vol. 46, no. 7, pp. 284–327, 2010.

[51] S. Ho, H. Nassef, N. Pornsinsirirak, Y.-C. Tai, and C.-M. Ho, "Unsteady aerodynamics and flow control for flapping wing flyers," *Progress in Aerospace Sciences*, vol. 39, no. 8, pp. 635–681, 2003.

[52] J. D. Anderson Jr, *Fundamentals of aerodynamics*. Tata McGraw-Hill Education, 1985.

[53] R. R. Bulatovic and S. R. DJordjevic, “Optimal synthesis of a four-bar linkage by method of controlled deviation,” *Theoretical and applied mechanics*, vol. 31, no. 3-4, pp. 265–280, 2004.

[54] P. Kurowski, *Engineering Analysis with SolidWorks Simulation 2013*. SDC publications, 2013.

[55] M. Bertozzi and A. Broggi, “Gold: A parallel real-time stereo vision system for generic obstacle and lane detection,” *Image Processing, IEEE Transactions on*, vol. 7, no. 1, pp. 62–81, 1998.

[56] H. Strasdat, J. Montiel, and A. J. Davison, “Real-time monocular slam: Why filter?,” in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pp. 2657–2664, IEEE, 2010.

[57] H. Hirschmüller, P. R. Innocent, and J. Garibaldi, “Real-time correlation-based stereo vision with reduced border errors,” *International Journal of Computer Vision*, vol. 47, no. 1-3, pp. 229–246, 2002.

[58] M. Körtgen, G.-J. Park, M. Novotni, and R. Klein, “3d shape matching with 3d shape contexts,” in *The 7th central European seminar on computer graphics*, vol. 3, pp. 5–17, 2003.

[59] D.-Y. Chen, X.-P. Tian, Y.-T. Shen, and M. Ouhyoung, “On visual similarity based 3d model retrieval,” in *Computer graphics forum*, vol. 22, pp. 223–232, Wiley Online Library, 2003.

[60] R. C. Bolles, H. H. Baker, and D. H. Marimont, “Epipolar-plane image analysis: An approach to determining structure from motion,” *International Journal of Computer Vision*, vol. 1, no. 1, pp. 7–55, 1987.

[61] S. S. Haykin, S. S. Haykin, and S. S. Haykin, *Kalman filtering and neural networks*. Wiley Online Library, 2001.

[62] B. K. Horn, “Closed-form solution of absolute orientation using unit quaternions,” *JOSA A*, vol. 4, no. 4, pp. 629–642, 1987.

[63] M. A. Wong and T. Lane, “A kth nearest neighbour clustering procedure,” in *Computer Science and Statistics: Proceedings of the 13th Symposium on the Interface*, pp. 308–311, Springer, 1981.

[64] G. Bradski and A. Kaehler, *Learning OpenCV: Computer vision with the OpenCV library*. ” O’Reilly Media, Inc.”, 2008.

[65] S. Birchfield and C. Tomasi, “A pixel dissimilarity measure that is insensitive to image sampling,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 20, no. 4, pp. 401–406, 1998.

[66] H. Hirschmuller, “Stereo processing by semiglobal matching and mutual information,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 30, no. 2, pp. 328–341, 2008.

[67] R. B. Rusu and S. Cousins, “3d is here: Point cloud library (pcl),” in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 1–4, IEEE, 2011.

[68] D. H. Ballard, “Generalizing the hough transform to detect arbitrary shapes,” *Pattern recognition*, vol. 13, no. 2, pp. 111–122, 1981.

[69] C. Harris and M. Stephens, “A combined corner and edge detector.,” in *Alvey vision conference*, vol. 15, p. 50, Manchester, UK, 1988.

[70] N. Kanopoulos, N. Vasanthavada, and R. L. Baker, “Design of an image edge detection filter using the sobel operator,” *Solid-State Circuits, IEEE Journal of*, vol. 23, no. 2, p-p. 358–367, 1988.

[71] R. G. Von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall, “Lsd: A fast line segment detector with a false detection control,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 4, pp. 722–732, 2010.

[72] J. B. Burns, A. R. Hanson, and E. M. Riseman, “Extracting straight lines,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, no. 4, pp. 425–455, 1986.

[73] G. R. Bradski, “Computer vision face tracking for use in a perceptual user interface,” 1998.

[74] D. Comaniciu and P. Meer, “Mean shift: A robust approach toward feature space analysis,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, no. 5, pp. 603–619, 2002.

[75] J. A. Hartigan and M. A. Wong, “Algorithm as 136: A k-means clustering algorithm,” *Applied statistics*, pp. 100–108, 1979.

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

[77] A. A. B. Pritsker and J. J. O’Reilly, *Simulation with Visual SLAM and AWESIM*. John Wiley & Sons, 1999.

[78] D. Nistér, O. Naroditsky, and J. Bergen, “Visual odometry,” in *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, vol. 1, pp. I–652, IEEE, 2004.

[79] D. Nistér, O. Naroditsky, and J. Bergen, “Visual odometry for ground vehicle applications,” *Journal of Field Robotics*, vol. 23, no. 1, pp. 3–20, 2006.

[80] M. Maimone, Y. Cheng, and L. Matthies, “Two years of visual odometry on the mars exploration rovers,” *Journal of Field Robotics*, vol. 24, no. 3, pp. 169–186, 2007.

[81] Y. Abdel-Aziz, “Direct linear transformation from comparator coordinates in close-range photogrammetry,” in *ASP Symposium on Close-Range Photogrammetry in Illinois, 1971*, 1971.

[82] D. W. Marquardt, “An algorithm for least-squares estimation of nonlinear parameters,” *Journal of the Society for Industrial & Applied Mathematics*, vol. 11, no. 2, pp. 431–441, 1963.

[83] V. Lepetit, F. Moreno-Noguer, and P. Fua, “Epnp: An accurate o (n) solution to the pnp problem,” *International journal of computer vision*, vol. 81, no. 2, pp. 155–166, 2009.

[84] A. Geiger, P. Lenz, and R. Urtasun, “Are we ready for autonomous driving? the kitti vision benchmark suite,” in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[85] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[86] J. J. Moré, “The levenberg-marquardt algorithm: implementation and theory,” in *Numerical analysis*, pp. 105–116, Springer, 1978.

Chapter 8

List of Author's Publications

Journal Articles:

1. Kevin Z. Y. Ang, J. Cui, T. Pang, K. Li, K. Wang, Y. Ke, B. M. Chen, Design and Development of a Thrust-vectored Unmanned Tail-sitter with Reconfigurable Wings. Accepted for publication in the Unmanned Systems Journal 2015.

2. S. Zhao, Z. Hu, M. Yin, K. Z. Y. Ang, P. Liu, F. Wang, X. Dong, F. Lin, B. M. Chen, and T. H. Lee, A Robust Real-time Vision System for Autonomous Cargo Transfer by an Unmanned Helicopter. IEEE Transactions on Industrial Electronics, Vol. 62, Issue 2, pp 1210-1219, August 2014.

3. F. Lin, K. Z. Y. Ang, F. Wang, B. M. Chen, T. H. Lee, B. Yang, M. Dong, X. Dong, J. Cui, S. K. Phang, B. Wang, D. Luo, K. Peng, G. Cai, S. Zhao, M. Yin and K. Li, Development of an unmanned coaxial rotorcraft for the DARPA UAVForge Challenge, *Unmanned Systems*, Vol. 1, No. 2, pp. 211-245, October 2013.

Book Chapters:

1. F. Lin, Kevin Z. Y. Ang, F. Wang, B. M. Chen, et al., Development of an unconventional unmanned coaxial rotorcraft: GremLion, Lecture Notes on Computer Science: Design, User Experience, and Usability, (Edited by A. Marcus), Volume 8014, pp. 120-129, Springer-Verlag, Berlin, Germany, 2013 (ISBN: 978-3-642-39237-5).

Conference Papers:

1. Kai-Yew Lum, X. X. Dong, Kevin Z. Y. Ang and F. Lin, Simulation study of homography-based vision-aided inertial navigation for aerial vehicles, 11th IEEE International Conference on Control & Automation, pp. 1357-1362, Taichung, Taiwan, June 2014.

2. Kevin Z. Y. Ang, J. Cui, T. Pang, K. Li, K. Wang, Y. Ke, B. M. Chen, Development of an Unmanned Tail-sitter with Reconfigurable Wings: U-Lion, 11th IEEE International Conference on Control & Automation, pp. 750-755, Taichung, Taiwan, June 2014.

3. S. Zhao, Z. Hu, M. Yin, Kevin Z. Y. Ang, P. Liu, F. Wang, X. Dong, F. Lin, B. M. Chen, and T. H. Lee, A robust vision system for a UAV transporting cargoes between moving platforms, 2014 Chinese Control Conference (CCC), pp. 544-549, IEEE.

4. F. Lin, Kevin Z. Y. Ang, F. Wang, B. M. Chen, T. H. Lee, et al., Development of an unconventional unmanned coaxial rotorcraft: GremLion, 15th International Conference on Human-Computer Interaction, Las Vegas, USA, July 2013. In Design, User Experience, and Usability. User Experience in Novel Technological Environments (pp. 120-129). Springer Berlin Heidelberg.

5. J. Q. Cui, F. Wang, X. Dong, Kevin Z. Y. Ang, B. M. Chen and T. H. Lee, Landmark extraction and state estimation for UAV operation in forest, Proceedings of the 32nd Chinese Control Conference, Xi'an, China, pp. 5210-5215, July 2013.

6. Partovi, A. R., Kevin Z. Y. Ang, Lin H., Cai G. and Chen, B. Development of a cross style quadrotor. In AIAA Guidance, Navigation, and Control Conference 2012.