INDOOR NAVIGATION SYSTEMS FOR UNMANNED AERIAL VEHICLES

WANG FEI

(B. Eng.(Hons.), NUS)

A THESIS SUBMITTED

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

NUS GRADUATE SCHOOL FOR INTEGRATIVE

SCIENCES AND 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]

WANG FEI

2 January 2014

Acknowledgments

Foremost, I would like to express my sincere gratitude to my supervisor, Prof. Ben M. Chen, for his continuous motivation and guidance during my Ph.D. study. His broad knowledge, systematic way of thinking have been the greatest assets to me not only inspires my research innovations but also enlightens my daily life.

I also wish to express my sincere thanks to the rest of my thesis committee, Prof. T. H. Lee, Prof. Lawrence Wong and Dr. Chang Chen, for their ideas, encouragements and insightful comments in meetings and discussions with me.

Special thanks also go to the NUS Unmanned Aircraft Systems Group. I will never forget the time when working with my teammates days and nights in meeting project deadlines and participating in various UAV competitions. Particularly, I would like to thank my seniors, Dr. Guowei Cai (assistant professor of Aerospace Engineering and Robotics Institute at Khalifa University), Dr. Feng Lin (research scientist at Temasek Laboratories, NUS) and Dr. Xiangxu Dong (research scientist at Temasek Laboratories, NUS), for sharing their valuable experiences in hardware design and software skills. Also, I really appreciate the advices from Dr. Kemao Peng (senior research scientist at Temasek Laboratories, NUS), Prof. Biao Wang (associate professor at Nanjing University of Aeronautics and Astronautics, China) and Prof. Delin Luo (associate professor at Xiamen University, China) who had been working in our group as senior scientists and research fellows. I am also thankful for the generous help from all other fellow team members and friends including Shiyu Zhao, Kevin Ang, Jinqiang Cui, Swee King Phang, Kun Li, Shupeng Lai, Peidong Liu, Tao Pang, Kangli Wang, Yijie Ke, Di Deng and Jing Lin.

Moreover, I am very grateful to my wife, Jing Han, who has consistently provided me supports and encouragements from my undergraduate study till now.

Last but not the least, I would like to thank my parents, for their everlasting love and care, as well as their supports for my education and research journey in Singapore.

Contents

Summary

This thesis aims to develop an advanced indoor navigation system for unmanned aerial vehicles. Two different UAV platforms have been developed as test beds for the study, namely a coaxial helicopter with a compact footprint and a quadrotor helicopter with larger payload. Modeling and design of flight control laws have been done successfully for both platforms. With the help of the onboard camera and laser scanner sensors, both visual and laser-based odometry methods have been implemented to solve the GPS-denied condition in an indoor environment. To get a better drift-free position estimation and to reconstruct a map along the UAV path, a simultaneous localization and mapping technique is explored in breadth and depth. An innovative FastSLAM algorithm in cooperating both corner and line features have been proposed and tested with great success. It is found that when indoor environment is partially known, a much more robust and efficient localization method can be implemented onboard of the UAV with a few reasonable assumptions. The developed UAV indoor navigation system has been verified in numerous flight tests and helped the Unmanned Aircraft Systems Group from the National University of Singapore win the overall championship in the 2013 Singapore Amazing Flying Machine Competition.

List of Tables

List of Figures

List of Symbols

Latin variables

$a_{\mathrm{dw}}, b_{\mathrm{dw}}$	Longitudinal and lateral flapping angles of lower rotor
$a_{\mathrm{up}}, b_{\mathrm{up}}$	Longitudinal and lateral flapping angles of upper rotor
$\boldsymbol{a}$	Acceleration measurement vector
$\boldsymbol{a}_{\mathrm{c}}$	Acceleration command vector
$\boldsymbol{a}_{\mathrm{r}}$	Acceleration reference vector
A	Rotor disk area
$A_{\mathrm{a,dw}}$	Lower rotor longitudinal on-axis ratio
$A_{\mathrm{a,up}}$	Upper rotor longitudinal on-axis ratio
$A_{\mathrm{b,dw}}$	Lower rotor longitudinal coupling ratio
$A_{\mathrm{b,up}}$	Upper rotor longitudinal coupling ratio
A_q, B_p	Rotor damping coefficients
$\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}$	System matrices of a time-invariant linear system
$B_{\mathrm{a,dw}}$	Lower rotor lateral coupling ratio
$B_{\mathrm{a,up}}$	Upper rotor lateral coupling ratio
$B_{\mathrm{b,dw}}$	Lower rotor lateral on-axis ratio
$B_{\mathrm{b,up}}$	Upper rotor lateral on-axis ratio
E	Pixel density
E_x	Partial derivative of pixel density in the image-frame x-axis
E_y	Partial derivative of pixel density in the image-frame y-axis
E_t	Partial derivative of pixel density w.r.t time
f	Focal length of a camera
$\mathbf{F}$	Resultant force vector in body frame
g	Gravitational constant

$\mathbf{G}_{\mathrm{c}}$	Inner-loop command generation matrix
$\mathbf{H}$	Homography matrix
$\mathbf{J}$	Moment of inertia of UAV
J_{xx}, J_{yy}, Jzz	Rolling, pitching and yawing moment of inertia
J_{dw}	Moment of inertia of the lower rotor
J_{up}	Moment of inertia of the upper rotor
$k_{T,\mathrm{up}}, C_{T,\mathrm{dw}}$	Lift coefficients of the upper and lower rotors
$k_{Q,\mathrm{up}}, C_{T,\mathrm{dw}}$	Drag coefficients of the upper and lower rotors
K_{a}	Scaling factor of the headlock gyro
K_{I}	Integral gain of the headlock gyro
K_{P}	Proportional gain of the headlock gyro
$\boldsymbol{l}_{\mathrm{dw}}$	Length vector from UAV CG to the lower rotor hub
$\boldsymbol{l}_{\mathrm{up}}$	Length vector from UAV CG to the upper rotor hub
m	Mass of UAV
$m_{\mathrm{up}}, m_{\mathrm{dw}}$	Upper, lower rotor speed to input ratio
$\mathbf{M}$	Resultant torque vector in body frame
n_t	Current robot landmark associations in SLAM formulation
n^t	History of robot landmark associations in SLAM formulation
N	Normal vector of the ground
$\mathcal{N}(\mu, \Sigma)$	Multivariate Gaussian distribution with mean μ and covariance Σ
p, q, r	Angular velocities in body frame
$\boldsymbol{p}$	Position vector
$\boldsymbol{p}_{\mathrm{r}}$	Position reference vector
P	Target point cloud
Q	Reference point cloud
$\mathbf{Q}$	Covariance matrix of input noise
$\boldsymbol{Q}_{\mathrm{d,dw}}$	Drag torque generated by the lower rotor
$\boldsymbol{Q}_{\mathrm{d,up}}$	Drag torque generated by the upper rotor
$\boldsymbol{Q}_{\mathrm{d,dw}}$	Reaction torque generated by the lower rotor
$\boldsymbol{Q}_{\mathrm{r,up}}$	Reaction torque generated by the upper rotor
(r_k, θ_k)	The k-th laser scanner measurement in polar coordinates
$\mathbf{R}$	Covariance matrix of measurement noise

$\mathbf{R}_{\mathrm{b/g}}$	Rotation matrix from NED frame to body frame
$\mathbf{R}_{\mathrm{b/w}}$	Rotation matrix from local-wall frame to body frame
$\mathbf{R}_{\mathrm{g/b}}$	Rotation matrix from body frame to NED frame
s_t	Current robot pose in SLAM formulation
s^t	History of robot poses in SLAM formulation
S_x, S_y, S_z	Effective drag area in the body-frame x-, y-, z-axis
t	Time
$\boldsymbol{T}_{\mathrm{up}}$	Trust generated by the upper rotor
$\boldsymbol{T}_{\mathrm{dw}}$	Trust generated by the lower rotor
u_t	Current robot control in SLAM formulation
u^t	History of robot controls in SLAM formulation
u, v, w	Linear velocities in body frame
$u_{\mathrm{g}}, v_{\mathrm{g}}, w_{\mathrm{g}}$	Linear velocities in NED frame
$\boldsymbol{v}$	Velocity vector
$\boldsymbol{v}_{\mathrm{r}}$	Velocity reference vector
x, y, z	Position coordinates in NED frame
(x_k, y_k)	The k-th laser scanner measurement in Cartesian coordinates
$\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}, \boldsymbol{u}$	State, measured output, controlled output, control input of a system
$X_{\mathrm{fus}}, Y_{\mathrm{fus}}, Z_{\mathrm{fus}}$	Fuselage drag force in the body-frame x-, y-, z-axis
z_t	Current robot measurement in SLAM formulation
z^t	History of robot measurements in SLAM formulation

Greek variables

δ_{ail}	Aileron input
δ_{ele}	Elevator input
δ_{rud}	Rudder input
δ_{thr}	Throttle input
ϕ, θ, ψ	Euler angles
$\phi_{\mathrm{c}}, \theta_{\mathrm{c}}, \psi_{\mathrm{c}}$	Euler angle commands (references)
$\phi_{\mathrm{sb}}, \theta_{\mathrm{sb}}$	Roll, pitch angles of the stabilizer bar
Θ	Map in SLAM formulation
τ_{sb}	Time constant of the stabilizer bar flapping
τ_{mt}	Time constant of the motor dynamics

ρ	Density of air
ω_{n}	Natural frequency of a second-order system
$\Omega_{\mathrm{up}}, \Omega_{\mathrm{dw}}$	Rotational speeds of the upper and lower rotors
ζ	Damping ratio of a second-order system

Acronyms

2-D	Two-Dimensional
3-D	Three-Dimensional
ABS	Acrylonitrile Butadiene Styrene
AHRS	Attitude and Heading Reference System
ARM	Advanced RISC Machine
ATEA	Asymptotic Time-scale and Eigenstructure Assignment
BCE	Brightness Constancy Equation
CG	Center of Gravity
CIFER	Comprehensive Identification from FrEquency Responses
CMOS	Complementary Metal-Oxide-Semiconductor
COTS	Commercial Off-the-Shelf
CPU	Central Processing Unit
DC	Direct Current
DCM	Directional Cosine Matrix
DOF	Degrees-of-Freedom
DSP	Digital Signal Processor
EKF	Extended Kalman Filter
ESC	Electronic Speed Control
FPGA	Field-Programmable Gate Array
FPS	Frames Per Second
GCS	Ground Control System
GPS	Global Positioning System
GPS/INS	GPS-aided Inertial Navigation System
I^2C	Inter-Integrated Circuit
ICP	Iterative Closest Point
IMU	Inertial Measurement Unit
INS	Inertial Navigation System

KF	Kalman Filter
LHS	Left Hand Side
Li-Po	Lithium-Polymer
LTI	Linear Time Invariant
MAV	Micro Aerial Vehicle
MEMS	Micro-Electro-Mechanical Systems
MIMO	Multi-Input Multi-Output
NED	North-East-Down
NUS	National University of Singapore
OpenCV	Open Source Computer Vision
OpenGL	Open Graphics Library
PC	Personal Computer
PCB	Printed Circuit Board
PPM	Pulse Position Modulation
PWM	Pulse Width Modulation
RC	Radio-Controlled
RHS	Right Hand Side
RISC	Reduced Instruction Set Computing
RMS	Root-Mean-Square
RPM	Revolutions Per Minute
RPT	Robust and Perfect Tracking
RS232	Recommended Standard 232
SCB	Special Coordinate Basis
SD	Secure Digital
SFM	Structure from Motion
SISO	Single-Input Single-Output
SLAM	Simultaneous Localization and Mapping
SVD	Singular Value Decomposition
TPP	Tip-Path-Plane
UAS	Unmanned Aircraft System
UAV	Unmanned Aerial Vehicle
UGV	Unmanned Ground Vehicle

UKF	Unscented Kalman Filter
USB	Universal Serial Bus
VTOL	Vertical Take Off and Landing
WiFi	Wireless Fidelity

Chapter 1

Introduction

1.1 Motivation

In recent years, the research on advanced indoor navigation systems for miniature unmanned aerial vehicles (UAVs) has aroused worldwide interests because of its great potential in military and civil applications [58, 67]. Indoor navigation technologies enable small-size UAVs to fly fully autonomously in known or unknown indoor environments with localization and map generation capabilities. If a pragmatic UAV indoor navigation system is realized, it can be used for applications like surveillance and patrolling, exploration and mapping, search and rescue and other indoor missions which were tedious and dangerous to human operators in the past. However, this UAV indoor navigation system has to be developed intelligent and robust enough to face challenges caused by the complicated indoor environments, such as denied reception of GPS signals and scattered obstacles, as well as physical constraints of the UAV platform, such as payload limitation. Furthermore, existing works on the topic of indoor navigation usually focus on 2-D environments and the majority of them are implemented on ground robots. The extension of an autonomous navigation system from the 2-D ground robot case to the 3-D UAV case is non-trivial and its development is still at a preliminary stage.

While the general aim of this thesis is to develop a comprehensive UAV indoor navigation system, special attention has been paid to realizing the navigation algorithms onboard of the UAV platform in real time. It is believed that a UAV system is much more valuable if its core navigation algorithms can be executed without relying on external sensory information or external computational power. In this way, it can be used for more general conditions and is more robust against environmental disturbances such as wireless communication loss. It should also be highlighted that most of the proposed navigation methods in this thesis utilize multiple onboard sensors, which include the inertial measurement unit (IMU), the scanning laser range finder and the camera. To realize a robust and efficient navigation system, different sensors need to used in a coherent and complementary way.

1.2 Challenges of UAV Indoor Navigation

1.2.1 Platform Constraints

Unlike unmanned ground vehicles (UGVs) or large-size outdoor UAVs, indoor UAVs have to be designed in small footprints so that they are able to maneuver in crowded indoor spaces. However, small footprints usually mean limited thrust and unconventional aerial dynamic designs. In consequence, only low quality sensors such as short-range laser scanner, low-resolution micro cameras and low-accuracy MEMS-based IMUs can be used onboard. In addition, the onboard processor will also be limited in computational power, which makes the sophisticated navigation algorithms difficult to be implemented in real time. Naive transfer of navigation algorithms from ground vehicles or large-size outdoor UAVs to indoor UAV systems will most likely fail.

Furthermore, the unconventional aerial dynamic design of the indoor UAV platforms also poses challenging problems to the whole system development. While modeling and control of conventional airplane or helicopter types of UAVs have been documented extensively in literature [13, 15, 66, 74], there is much less information of modeling and control of miniature coaxial helicopters or quadrotor helicopters, which are two commonly chosen platforms for UAV indoor applications. In consequence, large amount of time and efforts have been put into them at the starting phase of this work. The nonlinear coaxial helicopter model and its control method discussed in this thesis, although being just a byproduct of this research study, is actually a valuable contribution to the UAV modeling and control community.

1.2.2 GPS-denied Navigation

Unlike the conventional GPS/INS based navigation in which the UAV global position and velocity can be easily obtained, an indoor UAV system needs to get these information by developing complicated algorithms based on relative environmental sensing. Even if the GPS signal is available, its position measurement may not be accurate enough for UAVs to navigate in a confined indoor space. Hence, environmental sensing technologies and GPS-less UAV state estimation technologies play important roles in this research work.

Recent miniature-size UAV platforms developed by various research labs are equipped with two main sensory sources, namely the scanning laser range sensor and the vision sensor. The laser sensor can provide 2-D range information about the surrounding objects. Thus, relative 2-D positions of indoor walls and scattered obstacles with respect to the UAV body can be obtained. Another important function of laser sensor is to obtain the UAV rigid body motion, i.e. 2-D translational motion and 2-D rotational motion, by point cloud matching between consecutive scans.

For the visual sensor, a single camera can be used to estimate inter-frame motion of the UAV by searching for feature correspondences among consecutive image frames. If there are more than enough feature correspondences, the fundamental matrix describing the motion of the camera can be computed as an optimization problem. Then the rotational and translational motion matrices can be extracted explicitly. While the rotational matrix can be computed uniquely, the translational matrix is only up to a scale factor. Two solutions to eliminate this scale factor will be discussed in this thesis.

Laser odometry and visual odometry have their respective advantages and disadvantages. Laser odometry is in general more accurate and convenient to be used than that of vision and it does not have scale ambiguity. However, visual odometry can provide 3-D information which can be used to control the UAV vertical axis motion also. Since they have their respective advantages, it is better to combine them together through data filtering and fusion. By also bringing in the information from the inertial measurement sensor, Kalman filter or the Extended Kalman filter (EKF) can be used to estimate the UAV position, velocity, attitude angles and angular rates by considering the dynamic model of the controlled platform. This concept of multisensory data fusion has been studied in a long history of robotics [50], but only recently applied to UAV applications with success [5, 70].

1.2.3 Simultaneous Localization and Mapping

A key topic of this thesis is about the indoor simultaneous localization and mapping (SLAM) problem. SLAM is the method to build up the map for an unmanned vehicle within an unknown environment, and at the same time, to determine the vehicle's location within the map. In fact, for a long historical time, the localization problem and the mapping problem were considered as two separate issues and solved using different techniques. The objective of map generation is to integrate the information from different sensors to build a consistent model of the environment, such as the local obstacle map and the depth map [72]. On the other hand, localization is considered as a problem of estimating the position and attitude of the robot or vehicle in the map. In localization, data matching and association plays a critical role in obtaining correspondence between geometric or visual features.

It is only after the 90's when robots and unmanned vehicles started to have the capability of building up a map and keeping tack of their own positions simultaneously. It was found that even when the mapping and localization problems are combined together, the whole estimation problem is proven to be convergent [24]. The principle idea of probabilistic SLAM is to achieve monotonic decrease of estimation noise for vehicle pose and landmark positions and to achieve monotonic increase of correlations between landmark estimates when more and more observations are made [23]. To solve the probabilistic SLAM problem, it is necessary to find an appropriate representation of the observation model and motion model. If the motion model is represented in a state-space form, then the EKF is widely used. On the contrary, if motion model is given in a set of samples of the general non-Gaussian probability distribution, it leads to the use of the Rao-Blackwellised particle filter, or called the FastSLAM algorithm [48].

Although researchers after the 90's have successfully implemented SLAM in different robotics applications [24, 6], topics on robust data association, effective landmark representation, SLAM for large environments and SLAM for large number of landmark features still have unsolved problems. One well-known problem is about the wrong data association caused by non-distinctive geometric landmarks. In the standard SLAM formulation, the estimated states include the vehicle pose and a list of observed landmark. However, discrete identifiable landmarks are not easily discerned and direct alignment of sensed data is simpler or more reliable. Alternative formulation of the SLAM problem is consequently proposed, for example the trajectory-oriented SLAM [54]. In such solutions, 3-D point registration approaches are used to realize a reliable map reconstruction result. Nevertheless, even if robust and large-scale SLAM problem can be be solved theoretically, real-time implementation of these computationally intensive algorithms to the onboard system of a payload-limited UAV is still a question mark. Innovative assumptions about the navigation environment need to be made so that the SLAM algorithms can be simplified to a large extent, while still work reasonably well for practical scenarios.

1.2.4 Path Planning with Collision Avoidance

Path planning (or called motion planning) is also an essential module in advanced UAV indoor navigation systems, without which the controlled UAV cannot fly with meaningful purposes and it may even crash into obstacles. The usual way of path planning in literature is based on the assumptions of a known map and known UAV poses. That means the aforementioned SLAM problem needs to be solved first if UAV global position information and the environment is unknown. If the path planning algorithm is dependent on the result from SLAM, then onboard implementation is again questionable. As such, path planning strategies which only rely on raw sensor measurements or local map information will be considered in this thesis.

One approach is to utilize the potential field concept [8, 84]. This method of path planning can be used for both the globally known map and the locally known map cases. It employs repulsive fields around obstacles and an attractive field around the goal. The gradient of the resultant potentials will guide the controlled robot or UAV to move towards the goal while avoiding obstacles in a smooth way. One major drawback of these potential field methods is that there usually exists local minimums to the resultant potential fields which may trap the robot at that point infinitely. However, by manipulating the ‘goal’ or doing special case decisions, the local minimum problem can be largely avoided. Nevertheless, the potential field methods normally require less computational power as compared to the other searching-based methods, thus can be implemented onboard easily.

Another innovative approach for obstacle detection is to use the *time-to-collision* concept to realize visual collision detection, where an image sequence from a forward looking camera is employed to compute the time to collision for surfaces in the scene [87]. Although it cannot find the absolute depth information, optical flow can tell the time-to-collision, which is also useful information to avoid obstacles. Other approaches to achieve computationally efficient path planning are also studied recently in [20, 35, 36, 62]. They are especially popular nowadays because more and more research projects based on small-size UAVs have been launched worldwide.

1.3 Thesis Outline

The structure of this thesis is organized as follows. Chapter 2 reviews the state-of-the-art indoor UAV platforms and their capabilities. By comparing the pros and cons of different types of aerial platforms, two suitable types are chosen for this research work. Chapter 3 thoroughly lists onboard avionics that can be used for UAV indoor navigation purposes and chooses the optimum set for both selected UAV platforms. In Chapters 4 and 5, model formulation and identification of the chosen platforms are explained in detail. With the obtained model, inner-loop and outer-loop flight control laws are designed and implemented with actual flight tests. Visual odometry, laser odometry and sensor fusion methods are proposed and explained in Chapter 6, which tries to solve the navigation problem in GPS-denied conditions. Chapter 7 discusses about UAV indoor path planning and proposes a wall-following strategy that only relies on local laser range information. Next, the SLAM problem is thoroughly discussed in Chapter 8 and a customized FastSLAM algorithm based on corner and line features extracted from laser scanner data has been proposed and tested. It is argued in Chapter 9 that quite a few indoor UAV applications can be done in a partially known map condition. By making reasonable assumptions about a modern indoor environment, an efficient and robust localization method is developed. Based on the localization result, 3-D map reconstruction can be done by installing a second laser scanner orthogonally to the first. In Chapter 10, concluding remarks are made and future works are discussed.

Chapter 2

Platform Review and Selection

Since actual implementation and flight tests are the most solid proof of UAV-related theoretical studies, the first task of this research work is to develop a physical aerial platform suitable for navigation in confined indoor environments. Indeed, the choice of the bare aerial platform is one of the most important hardware factors which will affect the ultimate successfulness of any work involving algorithm implementation on real UAV platforms. Moreover, navigation in different environments require different platforms to be chosen so that the overall solution can be optimized in the hardware level, which effectively relieves burden for the later software algorithm development. This chapter will therefore present a comprehensive review of all types of UAV platforms and choose the most promising candidates as test beds with justifications. A few successful examples of indoor UAV platforms and their respective capabilities and applications will also be listed for reference.

2.1 Platform Choices

There are generally four types of UAV platforms, namely the fixed wing UAV (Fig. 2.1), the airship UAV (Fig. 2.2), the VTOL UAV (Fig. 2.3), and the unconventional UAV (Fig. 2.4). Note that these types of platforms can be used for both indoor and outdoor applications. However, they have different characteristics in shape, size, payload, stability and cruising speed, thus resulting in different levels of compatibility with indoor flight and different challenges in designing control and navigation algorithms.

A pros-and-cons comparison between the three conventional types of UAV platforms is shown in Table 2.1. It can be seen that the fixed wing airplanes are too fast to fly in a confined

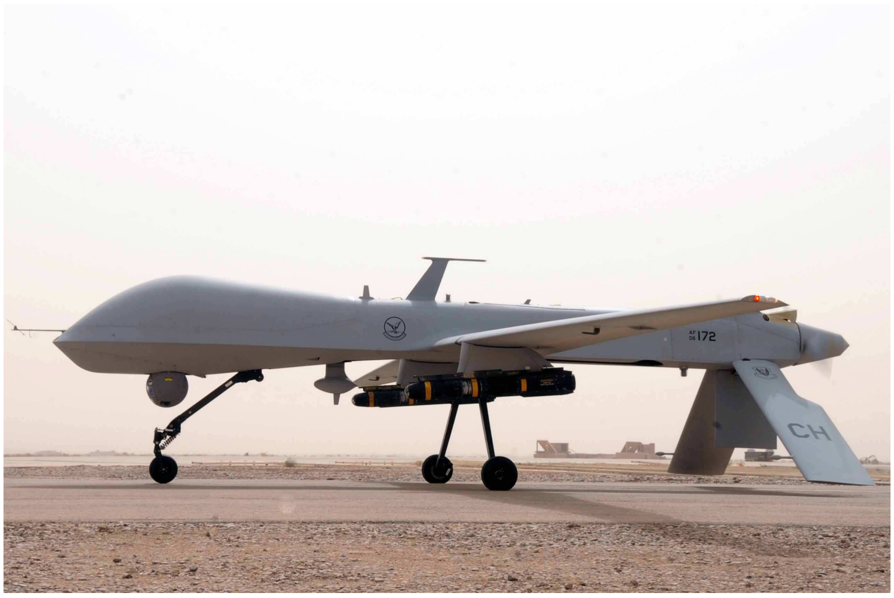


Figure 2.1: Fixed wing UAV: the Predator from General Atomics

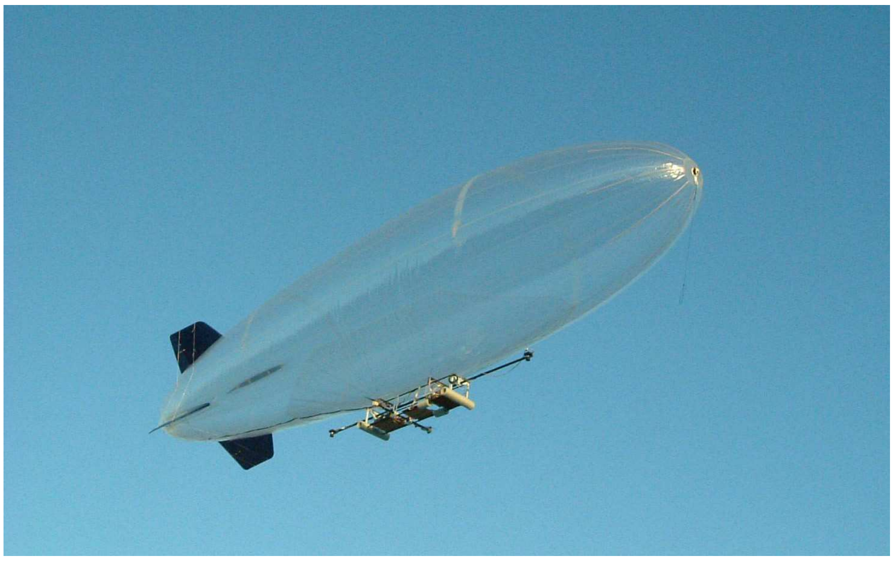


Figure 2.2: Airship UAV: Karma at LAAS-CNRS, in COMETS project

Table 2.1: Comparison between different types of UAVs

Types	Advantages	Disadvantages
Fixed Wing	Fast speed, long endurance, easy to be controlled	Unable to hover, unable to fly with low speed
VTOL	Great maneuverability, capability of hover	Difficult to be controlled, short endurance
Airship	Stable, energy saving, best for taking images	Large size, slow speed, hard to be controlled with position precision

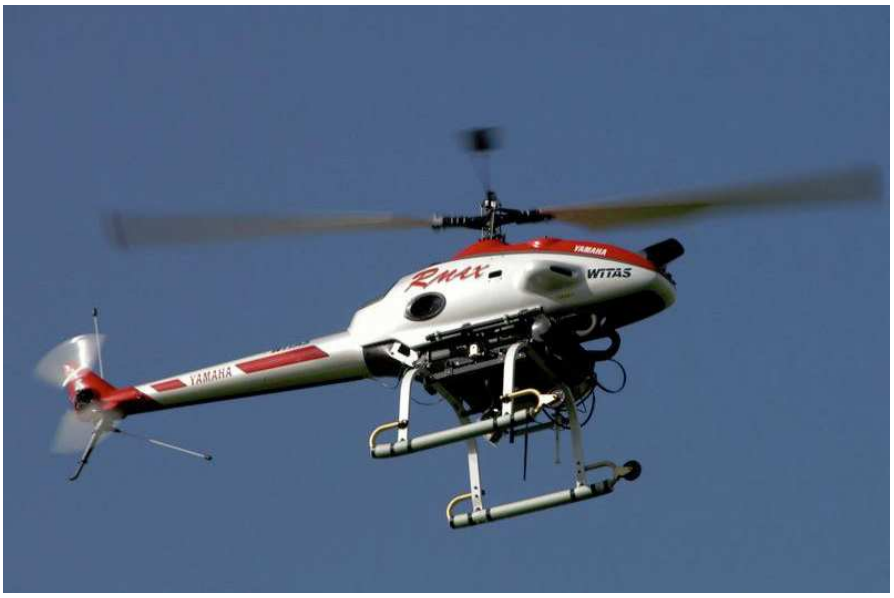
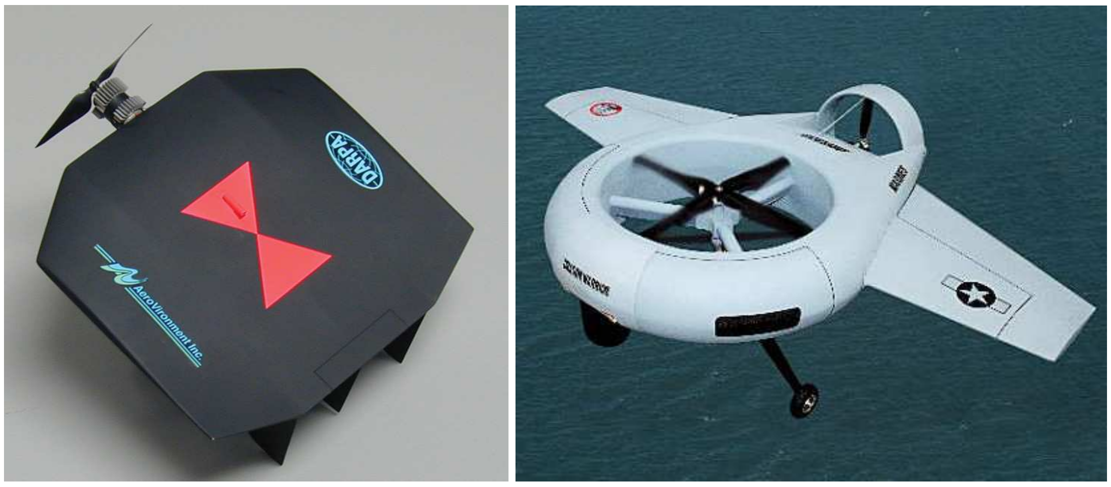


Figure 2.3: Helicopter UAV: Yamaha Rmax in the WITAS project



(a) Black Widow from DARPA

(b) Dragon Warrior from Sikorsky Aircraft

Figure 2.4: Unconventional UAVs

indoor space and they lack the hovering capability which is essential for most indoor tasks. On the other hand, the airship type of UAV platforms are too large in size to enter small rooms or corridors. The remaining two are the VTOL type and the unconventional type. By further surveying about common indoor UAV platforms and applications, it is found that the coaxial helicopter, which belongs to the VTOL type, and the quadrotor helicopter, which belongs to the unconventional type, are the most popular candidates. Note that the quadrotor helicopter was still unconventional when this Ph.D. study began, but it became gradually conventional after being extensively used by research groups and industries over the recent few years. Comparing with all other VTOL or unconventional aerial platforms, these two types of platforms have very impressive payload-to-size ratio. In an indoor environment, the UAV maximum horizontal dimension should not exceed the width of a door or a window which is most likely 1 to 1.5 meters. On the other hand, indoor navigation algorithms and control law implementations, if executed onboard, require large amount of computational power and measurement accuracy. These rely on high performance onboard processors and sensors, which burden a lot to the UAV payload. Hence, the coaxial and quadrotor helicopter platform are the more suitable candidates for this study. The coaxial configuration provides several advantages over the other types of platforms, summarized as follows:

1. It is relatively stable due to the damping effect introduced by a stabilizer bar [51];
2. It is proven to be more power efficient as compared to the single-rotor or quad-rotor configurations [21];
3. It has higher maximum forward speed than a single-rotor helicopter since it always has a pair of advancing and retreating blades, creating a symmetric lift in forward flight [19];
4. It has higher payload to dimension ratio than all the other configurations.

On the other hand, the quadrotor is mechanically simple and robust, with minimal number of moving parts, and it has a better shape for onboard avionics mounting. In the later part of this chapter, several existing coaxial and quadrotor UAV platforms from various universities and their corresponding applications will be reviewed. They serve as valuable references for the platform selection and design in this research work.

In order to control and utilize the coaxial and quadrotor platforms well, we need to first understand their basic working principles and characteristics. Both being lifted by rotors, their throttle and rudder control principles are quite similar. However, the mechanism of their aileron

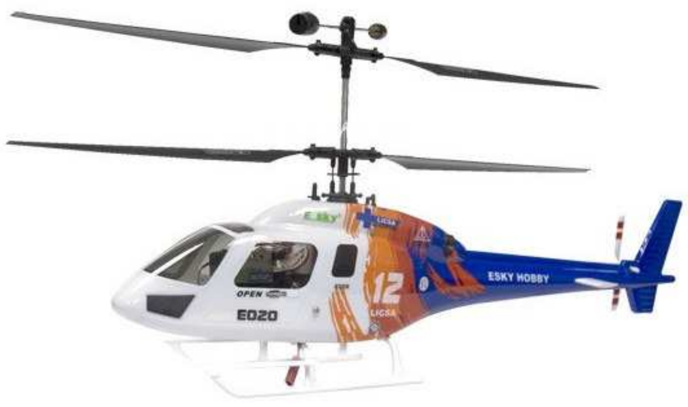


Figure 2.5: Esky Big Lama coaxial helicopter

and elevator control are very different.

1. **Coaxial Helicopter:**

   Unlike the conventional single-rotor helicopter, the coaxial helicopter (see Esky Big Lama in Fig. 2.5 as an example) has no tail rotor. It has two contra-rotating main rotors which are revolutions per minute (RPM) controlled. In general, the throttle signal controls the sum of the rotor speeds so that the platform can fly up and down, while the rudder signal controls the difference of the rotor speeds so that the heading of the platform can turn. Usually, a hardware headlock gyro is used as the most inner-loop stabilization to control the RPM difference of the two rotors. For the aileron and elevator control, the lower rotor is connected to a swashplate controlled by two servo motors so that its cyclic pitch can be changed to various directions and magnitude, thus resulting in the forward-backward or left-right tilting of the helicopter body. The upper rotor is passively balanced by a stabilizer bar which largely damps the rolling and pitching motion and makes the helicopter dynamics inherently stable. Hence, the coaxial helicopter has basically four controlling channels, namely aileron, elevator, throttle, rudder, and its manual flight performance is relatively stable comparing with other rotor-based aerial platforms.

2. **Quadrotor Helicopter:**

   For the quadrotor helicopter, the Parrot ARDrone can be used as an example (see Figure 2.6). From the name quadrotor, one can easily guess that there are four rotors on the aerial platform. All of the four rotors are RPM controlled and they are all on the same level plane. In fact, two of them rotate clockwise and the other two rotate anticlockwise. In this way, the resulting net torque around the platform vertical axis can cancel and the

Figure 2.6: Parrot ARDrone quadrotor helicopter

vehicle heading can be stabilized. Any imbalance of torque generated in this axis will result in yaw angle acceleration. To create a rolling motion, the left and right rotors have to be at difference RPM values so that the difference in the left and right thrust can tilt the platform sideward. Same principle applies to the control of pitching motion; the front and back rotors have to be at different RPM values. Last but not least, the heave motion is controlled by changing the average RPM of the four rotors. Unlike a coaxial helicopter with the swash plate, there are no moving servo motors on the quadrotor helicopter. This makes it mechanically simple and robust. The flight dynamics model of a quadrotor is standard to be formulated and its motion in four different channels can be largely decoupled, making it easier to be automatically controlled. In addition, its almost empty center space favors avionics mounting which is very needed for development of UAV autonomous navigation. However, quadrotor platform usually ends up with larger dimensions than the coaxial counterpart if the same amount of payload is required.

2.2 Review of State-of-the-Art Indoor UAV Platforms

In recent years, various indoor UAV platforms have been developed by research groups worldwide. In what follows will be a list of outstanding coaxial and quadrotor platforms with their corresponding indoor applications that have appeared in publications.

Quadrotor UAV from TUM and MIT

A quadrotor UAV (see Figure 2.7) was presented by Technische Universitat Munchen, Germany and MIT, USA. In cooperation with Ascending Technologies, Germany, the researchers in TUM

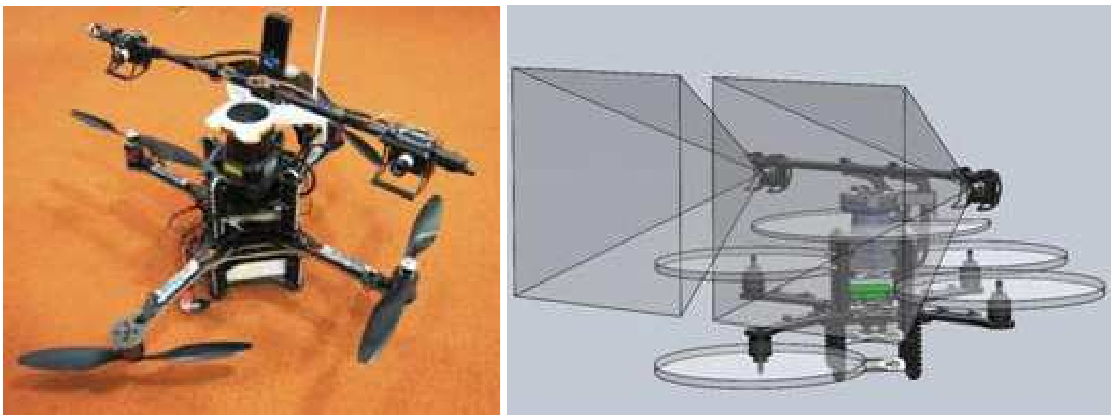


Figure 2.7: Quadrotor UAV from TUM and MIT

and MIT had designed this quadrotor helicopter capable of carrying additional 500 grams of hardware components, excluding the bare vehicle and battery, and continuously flying for about 10 minutes. Comparing with Ascending Technologies' old Hummingbird platform, this vehicle uses larger rotors (10 inches in diameter) as well as more powerful brushless motors. There is an interlocking rack at the top of the quadrotor, which can be used to mount two cameras for stereo vision. More creatively, the front rotor was placed below the arm to avoid camera obstruction while keeping the center of gravity low. There is a Hokuyo laser scanner mounted at the middle of the platform which is in charge of sensing its surrounding objects and obstacles. It is also the main sensor source for the UAV's map building function.

This UAV can perform fully autonomous navigation and exploration in GPS-denied indoor environments. It can accomplish missions like fully autonomous take-off, flying through windows, exploration and mapping, searching for objects of interest. In March 2008, this platform participated in the 1st US-Asian Demonstration and Assessment of Micro-Aerial and Unmanned Ground Vehicle Technology in Agra, India. Competing in a hostage-rescue mission scenario, it won the “Best Mission Performance Award”, the “Best Rotary Wing Aircraft Award”, and the “AMRDEC Award”. In 2009, it accomplished all the missions in the AUVSI indoor flight competition. The whole system has been proven robustly stable and practically capable [1].

Quadrotor UAV from Virginia Tech

The Virginia Tech research team have designed a quadrotor UAV (see Figure 2.8) equipped with a Microstrain 3DM-GX2 IMU, a Maxbotix LV-Maxsonar-EZ4 ultrasonic range sensor and a Black Widow AV KX-141 micro video camera. In order to protect the platform, the quadrotor UAV has aluminum bumpers installed when performing flight tests. The flight controller uses

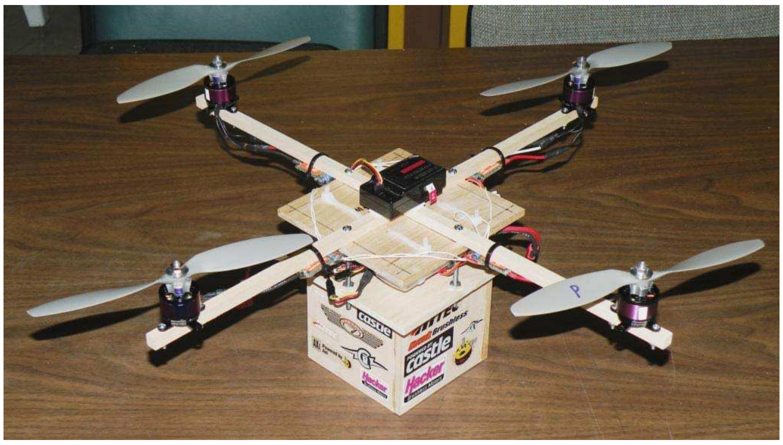


Figure 2.8: Quadrotor UAV from Virginia Tech

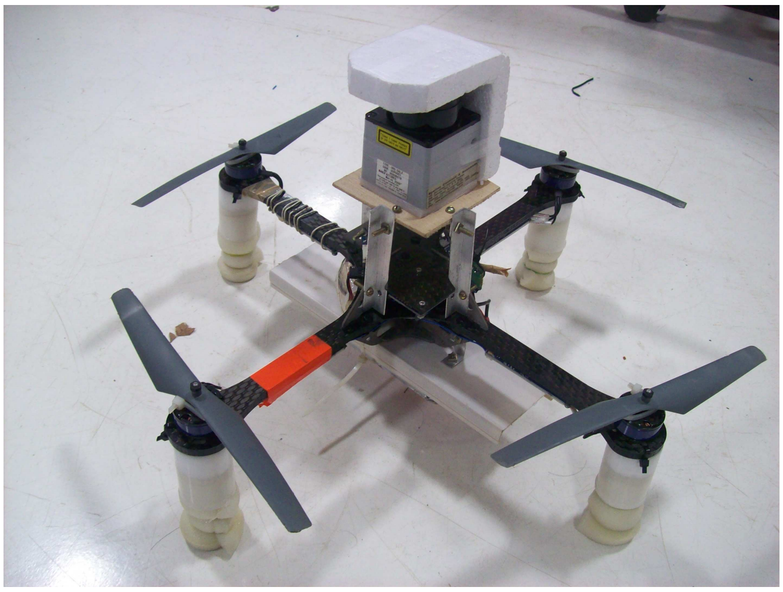


Figure 2.9: Quadrotor UAV from IIT Madras

estimated velocity together with IMU data to maintain flight stability. Simple-scenario obstacle avoidance was realized via analyzing ultrasonic range data. The high order commands are autonomously sent by the ground control station (GCS) which is in charge of vision processing [10].

Quadrotor UAV from IIT Madras

Figure 2.9 shows the photo of a quadrotor UAV from the Indian Institute of Technology Madras. The quadrotor frame is made of a combination of balsa wood and carbon fiber plates. The central frame is made of aluminum and it encapsulates the electronic circuits. A casing for the battery is made and placed at the bottom. A stand and a shelter is constructed to accommodate the laser scanner. Its control system has been partitioned into three layers. The lowest layer is in charge

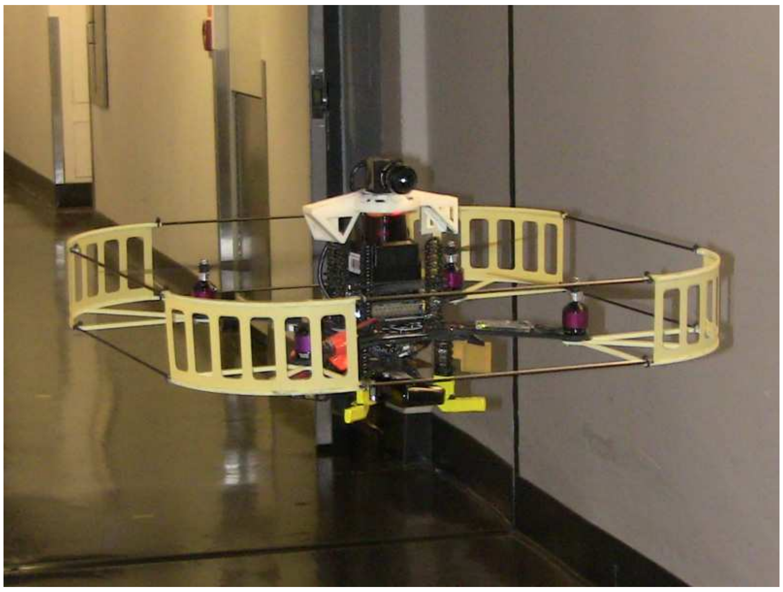


Figure 2.10: Quadrotor UAV from University of Pennsylvania

of the platform stability. It takes inputs from the IMU and control the lift provided by the four propellers so as to maintain a horizontal pose. The middle layer involves the velocity control system and the obstacle avoidance function. It takes in 2-dimensional obstacle profile from the laser scanner and by comparing consecutive scans, estimates the 2-dimensional velocity of the UAV itself. It then computes the control inputs needed to move the vehicle at the required velocity and send the signal to the motor driver. If there is obstacles detected nearby, the 3rd layer, called path planner, will draw a trajectory which can avoid the obstacles completely [64].

Quadrotor UAV from Upenn

Research team lead by Professor Vijay Kumar from the University of Pennsylvania have done impressive work in UAV indoor navigation. Their quadrotor platform bought from Ascending Technologies (see Fig. 2.10) is equipped with an IMU sensor, a Hokuyo UTM-30LX scanning laser range finder, a uEype 1220SE camera and a powerful 1.6 GHz Atom processor. The indoor navigation algorithm is developed using the Robot Operating System (ROS) which incorporates useful libraries and tools for robotic applications. With a navigation structure shown in Fig. 2.11, this UAV system is able to navigate in a multi-floor indoor environment with all necessary computation done onboard. While the UAV flies through the environment, a fairly detailed 3-D map can be generated. This is so far one of the most successful implementation of UAV indoor navigation [70].

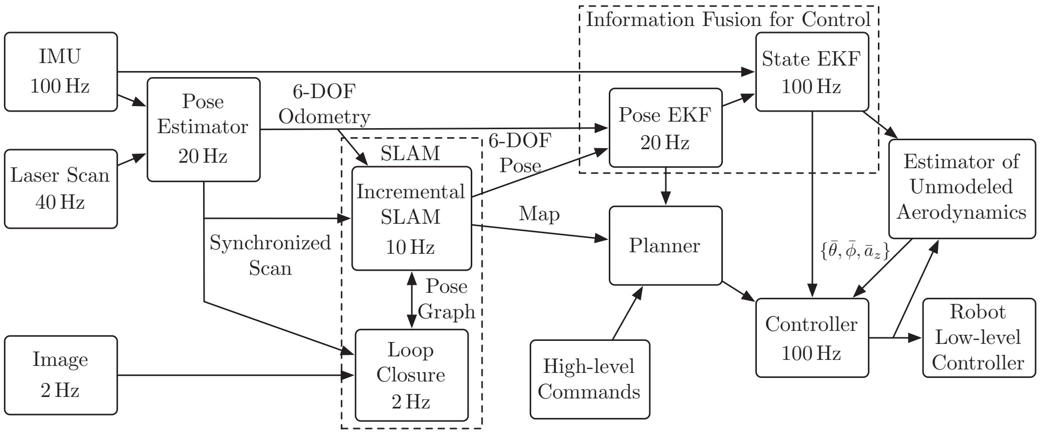


Figure 2.11: Navigation structure of the quadrotor UAV system from University of Pennsylvania

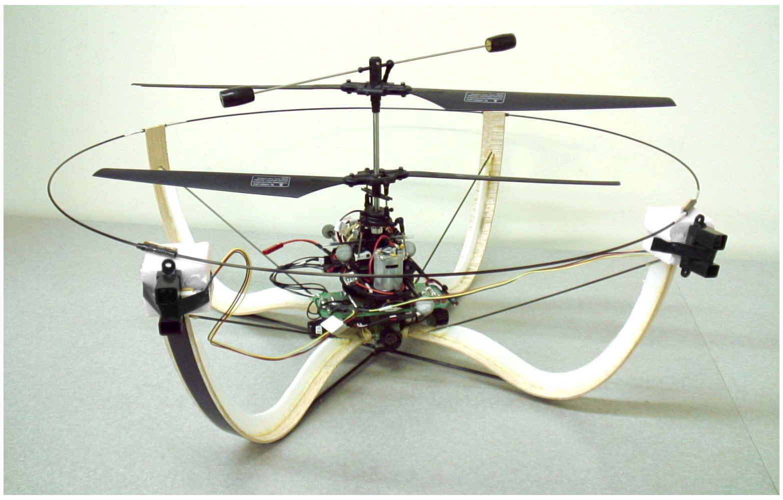


Figure 2.12: Coaxial UAV from Georgia Institute of Technology

Coaxial UAV from Georgia Institute of Technology

The Georgia Tech Aerial Robotics (GTAR) team have designed and built a vehicle (see Fig. 2.12) based on a commercially available stable platform - the Esky Big Lama. To keep the vehicle small and light, inexpensive infrared and ultrasound sensors were used to detect obstacles and walls. The UAV is controlled to follow the walls while avoiding frontal obstacles. A simple microcontroller is used onboard to handle guidance and navigation logics, as well as obstacle avoidance. An altitude-hold control loop maintains a constant altitude, simplifying the navigation problem. A video camera onboard captures real-time images and wirelessly transmits the data to the GCS, which processes the image streams and identifies potential targets. The GCS also displays vehicle health, status, and location information, and shows notifications when the target has been successfully identified.

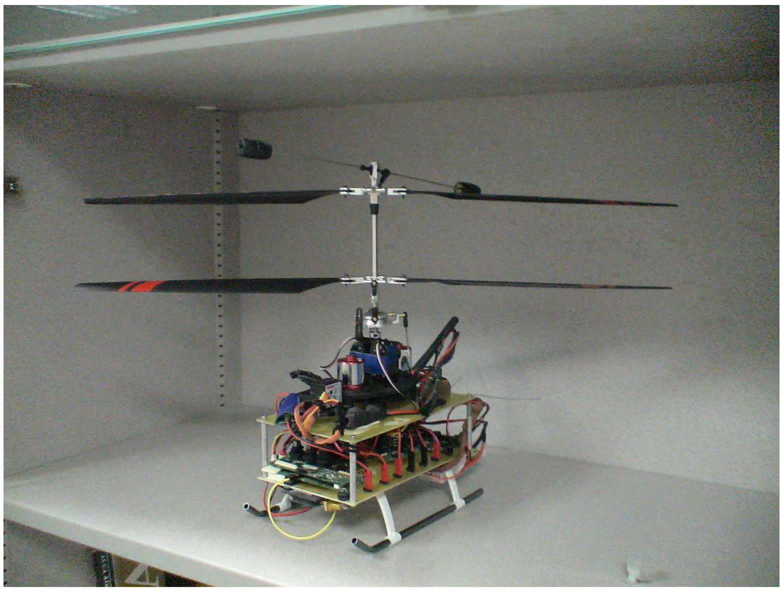


Figure 2.13: KingLion coaxial UAV from NUS

Coaxial UAV from the National University of Singapore

In indoor UAV systems developed in literature, the image processing and machine vision algorithms were usually executed on the GCS because of limited computational power onboard. Such transmission-decision-transmission manner will caused many problems in the vision-aided indoor navigation solution, such as extra image noises and transmission latency. This structure will also greatly limit the operating range of UAVs, and the responsiveness of UAVs in highly dynamic environments.

To increase the flexibility of UAV applications, the onboard vision processing mode has attracted much attention recently. The Unmanned Aircraft System (UAS) Group from the National University of Singapore have achieved great progress in onboard vision processing for its indoor miniature-size UAVs. Its KingLion (see Fig. 2.13), an indoor coaxial helicopter, has participated in the Category D section of Singapore Amazing Flying Machine Competition (SAFMC) 2009 and won the “Best Theory of Flight ” award and the “Best Performance” award. The main sensors on the avionic system are a CMOS camera and an ultrasonic sensor, both pointing downwards. The overall structure of this indoor UAV system is simple and elegant. It can fly indoor in a fully autonomous manner provided that there is a colored track on the ground for guiding, which is one of the main requirements in SAFMC 2009. The control algorithm execution and image processing are both done onboard, which means the vehicle can fly without the GCS once it takes off. While flying, the onboard system sends the real-time image streams to the GCS only for inspection purpose. Another amazing highlight of this system is that it uses two Gumstix embedded computers installed with the free Linux system. All codes are modified from open source code packages. This means that the overall system is not only cheap, but also expandable and reproducible [60]. In addition, the configuration of using two separated embedded computers in an onboard system, one for low level flight control and the other for high level navigation and decision making, is recommenced due to the following reasons:

1. The computation consumption of flight control and vision-based navigation algorithms are both heavy, which can hardly be carried out together in a single embedded computer;
2. The sampling rate of the flight control algorithm is much faster than that of vision processing. It is inefficient to implement both algorithms in a single executable program;
3. The two-computer structure reduces the negative effect of data blocking caused by the navigation program to the flight control system, and thus make the overall system more reliable.
4. If more suitable embedded computer products are released, the two-computer structure makes it possible to upgrade individual one easily.

2.3 Platform Decision

Although the platform selection has been boiled down to only two choices, namely the coaxial platform and the quadrotor platform, it remains a hard decision. With trade-offs between the compact physical form from the coaxial platform and the rigidity and reliability from the quadrotor platform, the ultimate decision goes to both. Therefore, two different platforms have been built and served as the test beds for this research work. One is a 450 grams (bare frame and battery) coaxial helicopter with 500 grams of extra payload, and the other is a 1300 grams quadrotor helicopter with 1600 grams of extra payload. The quadrotor is purposely built larger because we want to mount more powerful sensors and embedded computers on it, while the coaxial helicopter is equipped with cheaper and lighter sensors to further highlight its minimum form factor. The next chapter will list down two different sets of avionics components mounted on these two platforms. It will be seen that the sensors and onboard computers mounted on the quadrotor UAV are much more powerful, while the coaxial UAV's form factor is more attractive. The detailed specifications of the two selected platforms are discussed below, with the

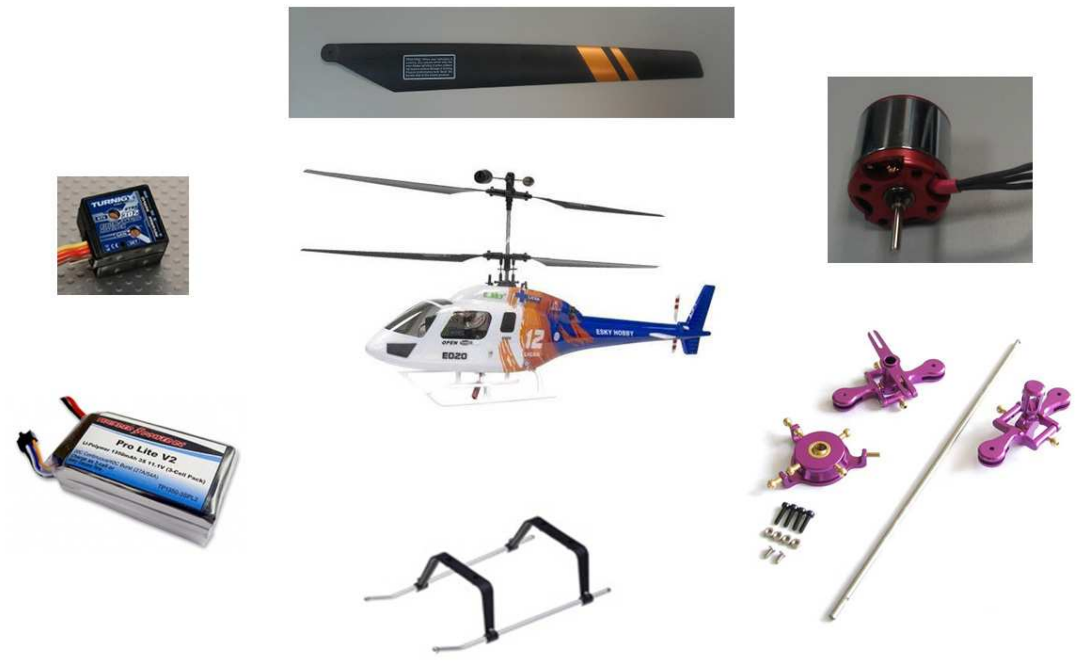


Figure 2.14: Esky Big Lama upgrades

coaxial platform upgraded from an commercial off-the-shelf (COTS) product and the quadrotor platform fully custom-made.

2.3.1 Coaxial Platform and Specifications

At the beginning of this indoor navigation study, the Esky Big Lama was one of the most well made coaxial RC toy helicopters with a miniature size. Unlike other RC toy helicopters, it has full 4-channel control and is capable of performing stable take-off, hovering, forward-backward flying, left-right sliding, yawing and landing. However, the original platform's take-off weight, as expected from most RC toy helicopters, is already marginal. Hence, a few hardware upgrades have been done to increase its payload so that additional avionics can be carried onboard to realized autonomous control. Fig. 2.14 has shown the individual upgraded components around the original Esky Big Lama platform, while Table 2.2 has highlighted the specifications before and after the upgrading.

2.3.2 Quadrotor Platform and Specifications

Instead of buying a COTS product, the quadrotor platform is fully custom-made because it is mechanically more manageable. The constructed quadrotor frame is composed of carbon fiber plates and rods with a durable Acrylonitrile Butadiene Styrene (ABS) landing gear (see

Table 2.2: Esky Big Lama before and after hardware upgrading

	Before	After
Frame and shaft	Mostly plastic	Metallic
Rotors	215 mm in length and soft	225 mm in length and stiff
Motors	3700 RPM V brushed motors	3800 RPM V brushless motors
Battery	3-cell 800 mAh Li-Po	3-cell 1400 mAh Li-Po
Take-off weight	410 g	950 g
Gyro and mixer	3-in-1 motor controller	Stand alone gyro, mixer and ESC

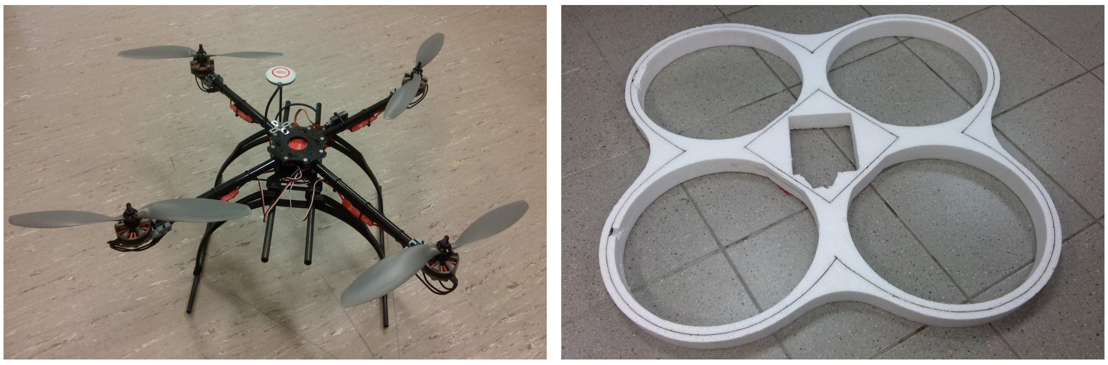

(a) The quadrotor platform (b) The quadrotor protection

Figure 2.15: The custom-made quadrotor platform and its foam protection

Fig. 2.15(a)). Its dimensions are 35 cm in height with a 86 cm tip-to-tip diameter. It is also built with reinforced aluminum motor mounts and platform mounts to strengthen the overall structure. This custom-made quadrotor has a total take-off weight of 2.9 kg and can fly up to 8 m/s. It hovers for about 10 to 15 mins, depending on sensor configuration and environmental factors. Since the quadrotor's main body only weighs about 1.3 kg, it can carry extra payload of 1.6 kg for onboard avionics and battery. Current battery used is a 4-cell 4300 mAh lithium polymer battery. The platform is also fully customizable in terms of sensor arrangement and is scalable such that additional computational boards could be mounted with a stack-based design. The motors used for the platform are 740 KV T-Motors with Turnigy Plush - 25A Bulletproof ESCs. The propellers used are APC 12X3.8 clockwise and anti-clockwise fixed pitched propellers. Each motor and propeller setup can generate 15 kN static thrust. Styrofoam protection (see Fig. 2.15(b)) reinforced with carbon fiber strips has been designed and installed to make the platform immune to collisions. This is particularly useful for a research-oriented platform as testing new algorithms will inadvertently result in the risk of flight crashes.

In conclusion, this chapter has presented a review on existing indoor UAV platforms. Several guidelines for choosing the most suitable platform have been proposed with justifications. In the end, the coaxial and quadrotor helicopters are chosen to be the testing platforms for this research work. The coaxial platform is upgraded from the COTS Esky Big Lama RC toy helicopter which has an extremely small form factor, while the quadrotor is totally custom-made with much better payload capacity.

Chapter 3

Onboard Avionics Systems

The previous chapter has discussed about the selection of aerial platform for this research work. However, to make an indoor UAV fly fully autonomous, onboard avionics plays an equally important role. In this chapter, various avionics components will be listed and compared, while the most suitable ones will be chosen with justifications.

First of all, the onboard avionics system of a typical indoor UAV consists of the following hardware devices [16] and its overall structure usually follows Fig. 3.1.

1. Inertia Measurement Unit
2. Range sensors
3. Vision sensors
4. Embedded computers
5. Servo driving and fail-safe electronic boards

In what follows will be a detailed functional explanation of these components with their corresponding state-of-the-art product examples.

3.1 Inertial Measurement Units

IMU is the core of the sensory system. It measures the UAV body-frame accelerations, angular rates, and usually estimates the UAV roll, pitch, yaw attitude angles via built-in digital filters. It contains fundamental measurements required by the inner-loop stability control of the UAV system. The flight performance of the UAV is highly dependent on the quality of these signals. Fig. 3.2 to 3.6 show a series of state-of-the-art IMU sensors from different companies or hobby

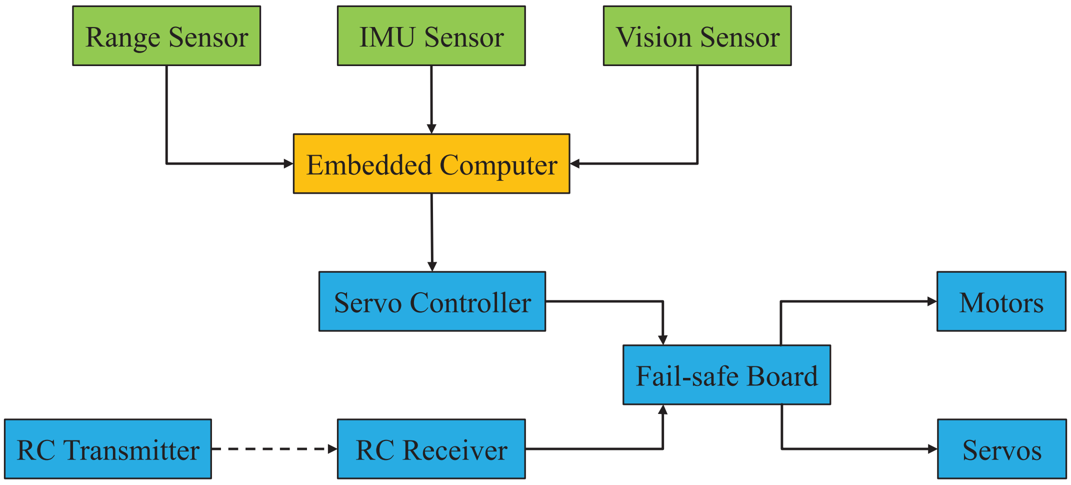


Figure 3.1: Common structure of an indoor UAV onboard avionics

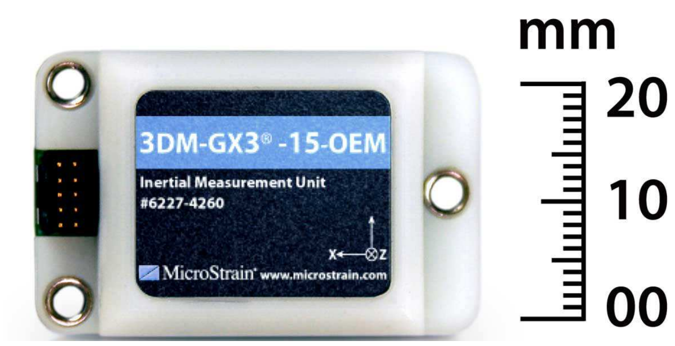


Figure 3.2: 3DM-GX3 -15-OEM from MicroStrain

developers. They have common characteristics such as small size, light weight and good accuracy, which suit for indoor UAV applications to a large extent. Table 3.1 shows a comprehensive comparison of key specifications of these IMU products.

3.2 Range Sensors

Range sensors have many varieties. From the low-end infra-red range sensors to the high-end scanning laser range finders, they can measure relative distance of the detected objects. Different types of range sensors utilize different types of waves, namely the infra-red wave, the ultrasonic wave, and the laser (light) wave. An object is said to be detectable with respect to a particular kind of wave means the object surface can reflect that kind of wave effectively. The distance from the sensor to the interested object can be calculated by multiplying the wave speed and the return time (from emitting to reflecting to receiving) and divide by two. Fig. 3.7-3.9 show these three main types of range sensors.

Figure 3.3: Colibri from Trivisio

Figure 3.4: IG-500N from SBG Systems

Figure 3.5: MTi from Xsens

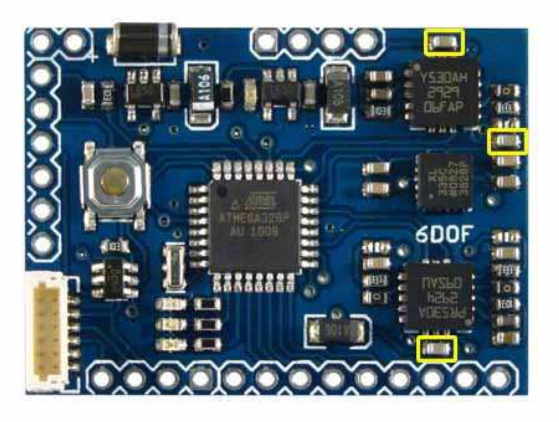


Figure 3.6: ArduIMU V2 (Flat) from DIY Drones

Figure 3.7: GP2D12 IR Sensor from Sharp

Figure 3.8: LV-MaxSonar-EZ ultrasonic sensor from MaxBotix

Table 3.1: Comparison between miniature IMU products

Model	**3DM-GX3**	**Colibri**	**IG-500N**	**Mti**	**ArduIMU**
Accelerometer range	5 g	16 g	5 g	5 g	3 g
Gyroscope range	300/s	1500/s	300/s	300/s	300/s
Static accuracy	0.5	0.5	0.5	0.5	Not provided
Dynamic accuracy	2.0	2.0	1.0	2.0	Not provided
Update rate	1000 Hz	100 Hz	100 Hz	256 Hz	50 Hz
Interface options	USB 2.0 / TTL	USB	RS232 / TTL / USB	RS232 / RS485 / USB	TTL
Supply voltage	3.1 - 5.5 V	5 V	3.3 - 30 V	4.5 - 30 V	5 V
Power consumption	400 mW	200 mW	800 mW	350 mW	200 mW
Weight	11.5 g	22 g	48 g	50 g	6 g
Size (mm)	$40 \times 20 \times 9$	$30 \times 30 \times 13$	$49 \times 36 \times 25$	$58 \times 58 \times 22$	$39 \times 29 \times 3$

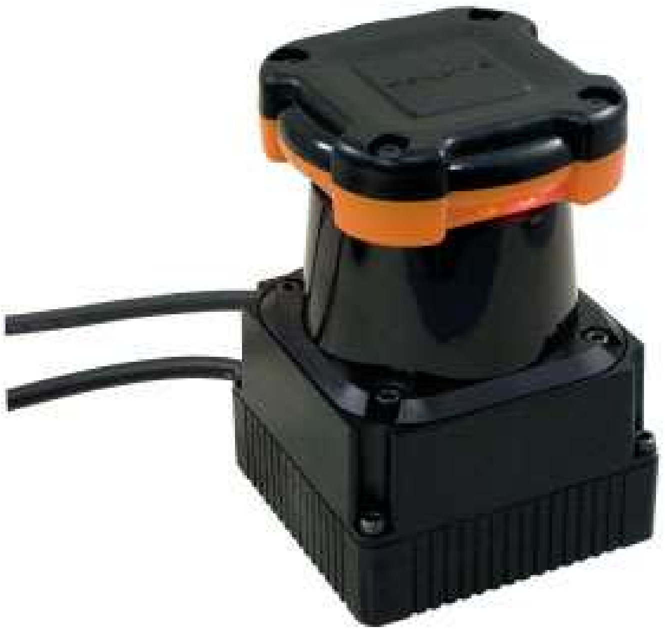


Figure 3.9: UTM-30LX Laser Scanner from Hokuyo

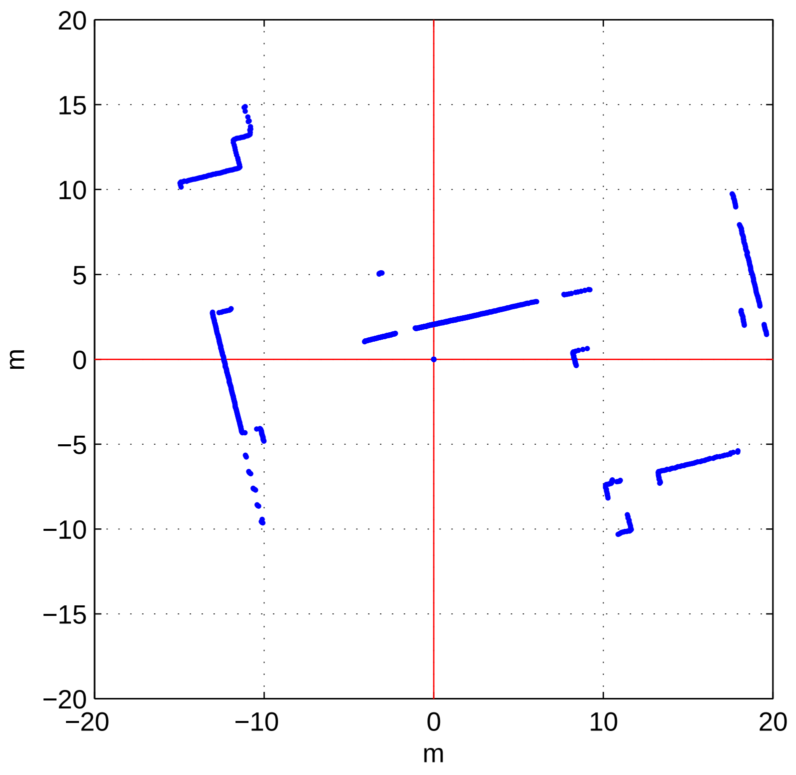


Figure 3.10: Measurement from a scanning laser range finder

Figure 3.11: 2.4GHz wireless CMOS camera

Among all these range sensors, the scanning laser range finder gains the greatest amount of interests from researchers. It has three main attractive features, including the superior accuracy and resolution, the almost omni-directional measurements and its compact physical form. The working principle of a scanning laser range finder is simple. It keeps emitting a narrow beam of laser wave, while a mirror inside continues rotating so that the laser beam can be reflected and sent to all directions. For one round of rotating (scanning), it records down a set of distance values in all directions from the source to the nearest object in that direction. Thus, objects in all directions can be detected (see Fig. 3.10) and more importantly, by analyzing the differences among consecutive scans, algorithms can be implemented to estimate the rigid body motion (translational and rotational) of the UAV body while flying.

3.3 Vision Sensors

Small, light and low-power CMOS cameras are usually used for vision sensing on indoor UAVs. They normally provide images with pixel resolution of 640×480 or 320×240 in a real-time (30-60 Hz) frame rate. After the images have been captured, two types of vision processing approaches can be adopted. One is to use wireless communication to send the source images to the GCS for vision processing, and then send back the computed results to the onboard computer for control and navigation purposes (see Fig. 3.11). This approach is broadly used because vision processing algorithms usually needs intensive computation that normal embedded computers may not be able to handle in real time. Since the GCS does not have such weight limitation, powerful computers can be used. However, this approach is not that useful from a pragmatistic point of view. The wireless communication between the UAV and the GCS not only generates delays, but also makes the UAV too dependent on the GCS. Ideal communication needs

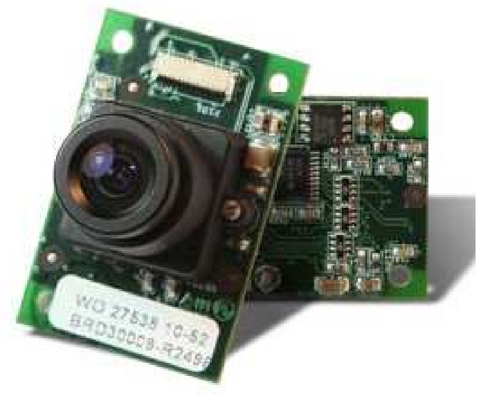


Figure 3.12: Gumstix Caspa$^{\text{TM}}$ VL camera

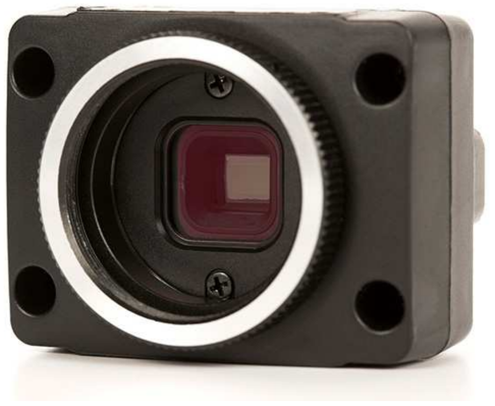


Figure 3.13: PointGrey FireFly® USB 2.0 Camera

to be maintained so that the whole system does not malfunction. For complicated missions in unknown environments, maintaining high-quality wireless communication is almost impossible.

To achieve high robustness and high application usefulness, vision processing needs to be done onboard. If this second approach is adopted, powerful embedded computers as well as efficient vision processing algorithms are needed. For this approach, we can use cameras that can directly communicate with the embedded computers. For example, the Gumstix Caspa$^{\text{TM}}$ VL camera in Fig. 3.12 and the PointGrey FireFly® USB 2.0 Camera in Fig. 3.13.

Some robotics research groups have also managed to use the so-called omni-directional camera which can capture a 360° image (see Fig. 3.14). If installed, the indoor UAV can have full vision information around it which is very beneficial for obstacle detection and map building. However, the drawback is its unbearable weight for an indoor UAV with limited payload.

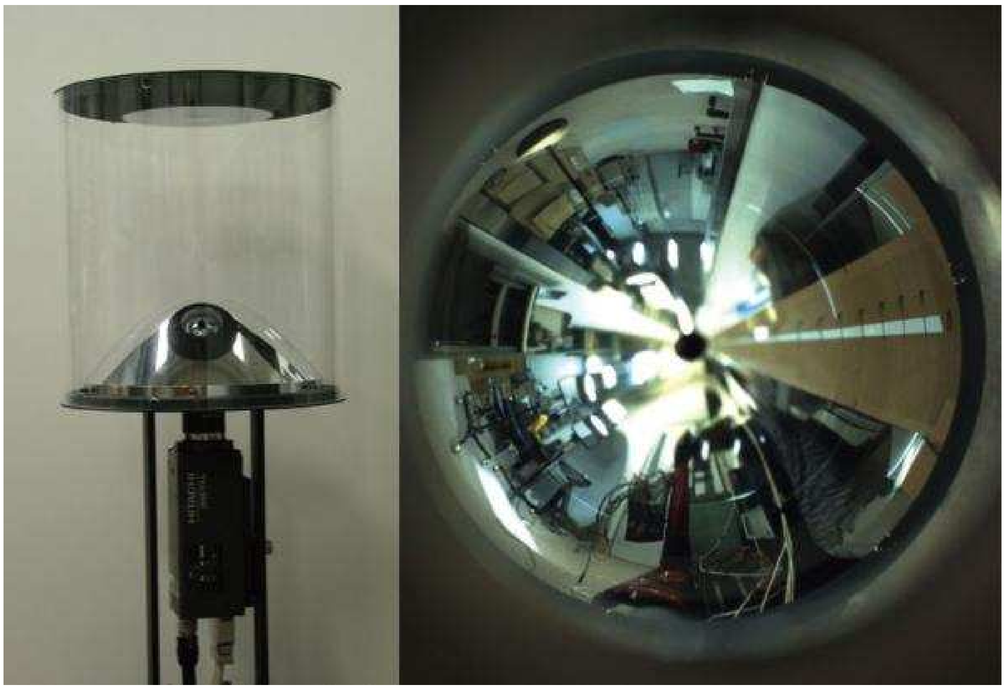


Figure 3.14: Omni-directional camera

Figure 3.15: Gumstix Verdex Pro working with Console-vx expansion board

3.4 Embedded Computers

To implement control and navigation algorithms, indoor UAVs are usually equipped with small-size embedded computers. Acting as the brain of the whole navigation system, the embedded computer reads measurements from sensors, applies data fusion, executes control laws, and outputs control signals. Sometimes, it is in charge of data logging and communication with the GCS too. The authors in [60] presented a vision system develop based on the Gumstix Overo Fire and a webcam to realize vision-aided indoor navigation. In addition, there are several commercialized state-of-the-art embedded computers available as of writing. They are shown in Figs. 3.15–3.18.

Another alternative is to design a customized embedded computer which optimally suits

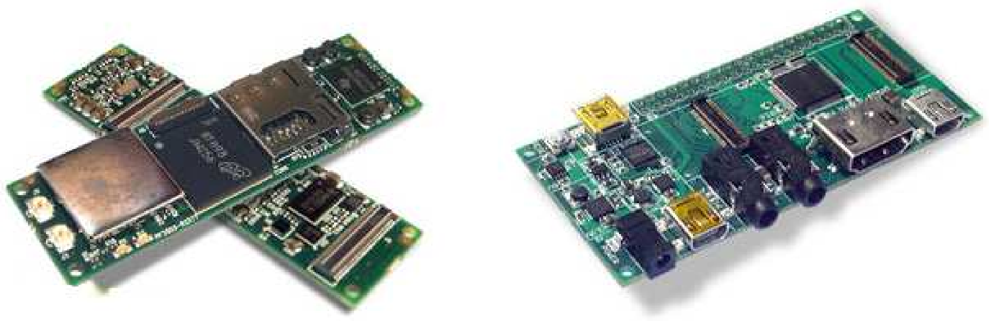


Figure 3.16: Gumstix Overo Fire working with Summit expansion board

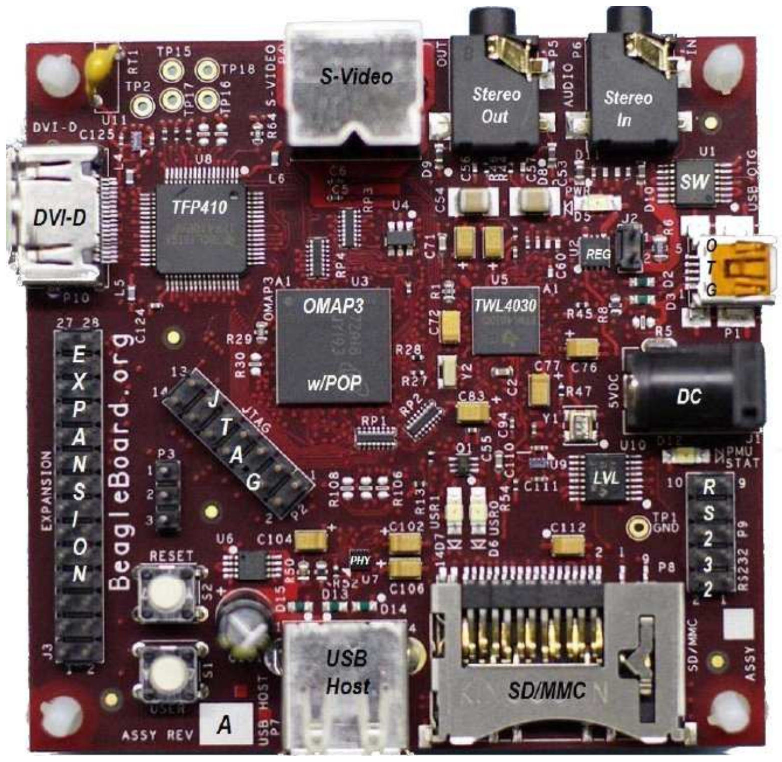


Figure 3.17: The Beagleboard

Figure 3.18: fit-PC2 from CompuLab

for the UAV indoor navigation project. For instance, a FPGA based vision system is proposed in [27] to realize the drift-free control for a Micro-UAV in an indoor environment. This vision system was called Helios, composed of SDRAM, SRAM, a Virtex-4 FPGA, and USB connectivity. Harris corner detection and template matching algorithms are implemented in the custom-made vision system, which are used to detect the drift of the helicopter in the x- and y-axis. However, the custom-made systems need expertise and extra manpower and time. The decision depends on how stringent is the weight budget. If the commercial embedded computers can satisfy all the requirements without major problems, it is more productive to directly build high level software algorithms on it without worrying about the low level computer hardware design.

3.5 Servo Driving and Fail-Safe Electronic Boards

Usually, the embedded computer does not directly output PWM signals to drive the motors or servos (it can be done, but very inefficient as it consumes a lot of extra computational power), but sends control law outputs to the servo controller (or servo driving board), which can generate the corresponding PWM signals to drive the actuators. Servo controller usually takes in serial or I^2C format inputs and convert them to multiple channels of PWM signals that can be recognized by motor ESCs and servos. Fig. 3.19 shows a small-size standard servo controller that can be

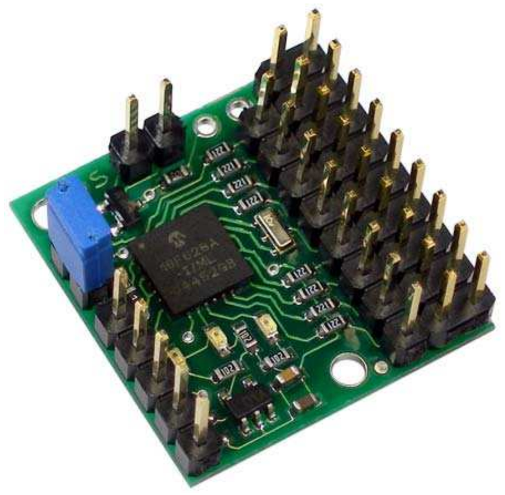


Figure 3.19: Micro Serial Servo Controller from Pololu

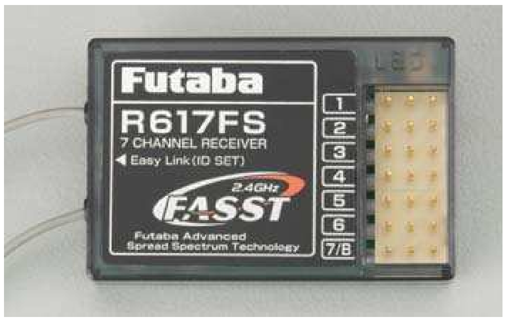


Figure 3.20: Futaba R617FS 7-Channel 2.4GHz FASST Receiver

chosen.

Although the indoor UAV will be eventually fully autonomous, its manual control capability still needs to be retained. Manual control is necessary for in-flight data collection for model identification and fail-safe protection if things go wrong. Hence, an RC receiver and a fail-safe multiplexer can be usually found on a UAV platform. With their presence, the controlled UAV can be easily switched between automatic mode and manual mode via an auxiliary channel from the RC receiver which connects to the ‘select’ ping of the fail-safe board. Fig. 3.20 and Fig. 3.21 show the RC receiver and the fail-safe board respectively.

3.6 Two Avionic Configurations of the Indoor UAV Platforms

As explained in Chapter 2, both the coaxial helicopter and the quadrotor helicopter are chosen to be used as the testing platforms for this research work. However, they have different onboard avionics configurations. Table 3.2 has listed the different avionics components selected for the coaxial and quadrotor platforms. It can be seen that the coaxial platform, being restricted by

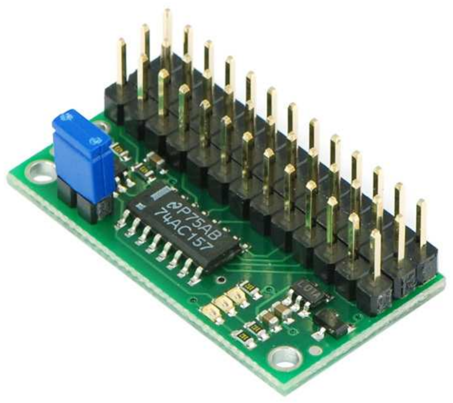


Figure 3.21: Fail-safe multiplexer

Table 3.2: Dual onboard configurations of the indoor UAV platforms

	Coaxial Platform	**Quadrotor Platform**
IMU	ArduIMU (DIY Drones)	IG-500N (SBG Systems)
Laser scanner	URG-04LX (HOKUYO)	UTM-30LX (HOKUYO)
Camera	Caspa$^{\mathrm{TM}}$ VL (Gumstix)	FireFly® (PointGrey)
Control computer	Overo® Fire COM (Gumstix)	Overo® Fire COM (Gumstix)
Vision computer	Overo® Fire COM (Gumstix)	fit-PC2 (CompuLab)
Servo controller	Micro Maestro (Pololu)	UAV100 (Pontech)

payload capacity, is equipped with lighter and low-performance sensors and embedded computers. In contrast, the quadrotor has more powerful avionic components due to its larger payload capacity.

For the IMU sensor, IG-500N GPS/INS (GPS-aided inertial navigation system) unit mounted on the quadrotor platform is a complete attitude and heading reference system (AHRS) with high-quality and well-calibrated sensor chips. It can reliably outputs UAV position and velocity if used outdoor with GPS. However in an indoor setting, only the UAV's attitude angles can be obtained. Nevertheless, it can still provide precise and drift-free 3-D orientation even during aggressive maneuvers, updated at 100 Hz. As compared to the other available miniature IMU sensors, IG-500N's dynamic performance is superior. In contrast, ArduIMU used on the coaxial platform, is only hobby standard. Its DCM-based algorithm is inherently flawed in non-zero acceleration flight conditions. However, its under 10 g weight and flat shape suits very well to be mounted on the much smaller coaxial platform.

For the scanning laser range finder, URG-30LX has a maximum range of 30 m and can scan its frontal 270° fan-shaped area with an extraordinary fine resolution of 0.25°. On the other hand, URG-04LX's scanning area is only 4 m and 240° with angle resolution about 0.36°. Both of them have range resolution of 1 mm and accuracy about 1% of the measured range. In the later implementation stage, it will be found that URG-04LX is enough for normal flight tests with obstacle avoidance functions. However, to do a complete localization and mapping with such short measurement range is rather difficult because its scanned result at one instant will easily form singular cases, such as nothing in 4 m's range or there is only one straight wall in range.

The cameras used on both platforms are not much different in terms of performance. Instead, the choice depends more on the supporting driver of the vision processing computers. Caspa™ VL is specially designed for Gumstix Overo® COM series. It connects to Gumstix through a parallel port which makes the retrieving of image information lightening fast. FireFly® Camera from PointGrey, although having a better resolution and maximum frame rate, can only provide USB 2.0 signals which makes the image capturing step relatively slow. As the onboard real-time image processing algorithm is only expected to be run upon a 320×240 single channel grey image at about 10 Hz, both camera's resolution and frame rate specifications are more than enough for this project. The bottleneck is indeed at the vision processing computer.

For the vision processing computer, Gumstix Overo® Fire COM is used on the coaxial platform to save weight. Being small though, it still has a 720 MHz main clock and a DSP co-processor. It is verified that an indoor colored-road-tracking algorithm can be successfully run onboard in real time [60]. However, quite a few assumptions and algorithm simplifications need to be done in the expense of tracking robustness and accuracy. In order to implement a more general and practical vision algorithm for UAV indoor navigation, more powerful onboard computers are recommended. Hence, we choose the fit-PC2 from CompuLab for the quadrotor platform. As of writing, the fit-PC2 is the smallest, most energy-efficient fanless PC on the market. It has low-end desktop PC performance with CPU clock at 1.6 GHz, 1 GB of ram and 16 GB of SSD storage. Besides, it provides 2 ethernet ports, 2 USB ports and 1 RS232 port for peripheral devices. In this case, one USB will be used to communicate with the camera and the RS232 port will be used to communicate with the control computer. It has a weight of 270 g, which is acceptable for the quadrotor platform but too heavy for the coaxial platform.

For the control computer, Gumstix Overo® Fire COM is more than enough to carry out sensor data retrieving and fusion, control law implementation, communication with GCS, as well as data logging. The IMU reading and control loop runs at 50 Hz, while other peripheral threads runs at lower frequencies. Working with the Pinto-TH expansion board, the whole

Gumstix computer is only 30 g in weight and it includes an built-in WiFi module, perfect for remote debugging and communication purposes.

For the servo driving electronics, the Micro Serial Servo Controller, working together with the 4 Channel fail-safe multiplexer, both from Pololu, are used on the coaxial platform. Again, this choice favors weight reduction but the drawback lies in its limited functions. Unlike the UAV100 servo controller used on the quadrotor platform, the Pololu sets are not able to feedback the channel control values to the control computer. This is a huge problem for the later model-based parameter identification process. When the pilot performs manual perturbations to the flying vehicle, we need to record down the four channel control inputs so that model identification can be done. To solve this problem, the RC receiver's PPM signal (contains the combined information from all channel PWM signals) is hacked and fed to one spare pin of ArduIMU. As ArduIMU is open source, additional code is added to its bare IMU functionality to decode the PPM signal and output all channel control values together with the original IMU measurements. On the other hand, UAV100 communicates with the control computer in a 2-way fashion. It not only listens to control computer's commands and output the corresponding PWM signals, but also feedback the servo control signals (both autonomous and manual) to the control computers for logging purposes.

After all, the full onboard avionics configurations for the two different platforms are summarized in Fig. 3.22 and 3.23.

3.7 Computer-aided Layout Design

After selecting and configuring the individual avionic components, all of them need to be assembled together on to the bare aerial platform to form a complete UAV. To accomplish this task, special attention needs to be paid to the layout design of the overall onboard system. Despite the connection and signal flow among the hardware devices, their physical position, orientation and mounting method need to be precisely designed so that the platform CG, rigidity and aerodynamic characteristics are not adversely affected.

The next thing needs to be ensured is to place the IMU sensor as close as possible to the CG of the whole UAV platform to minimize the so-called lever effect, which causes bias to acceleration measurement when the UAV platform performs rotational motion. Usually there is no difficulty to align the IMU with the UAV CG in the x- and y-axis. However, the z-axis

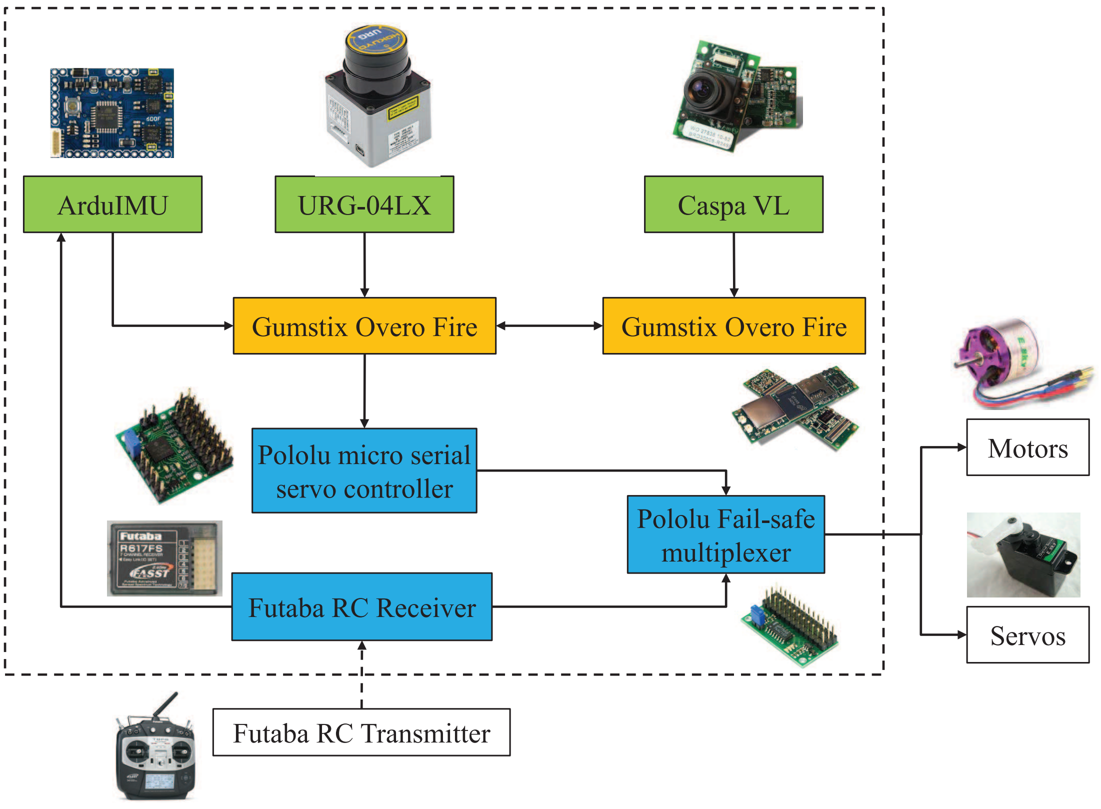


Figure 3.22: Onboard avionics configuration of the coaxial platform

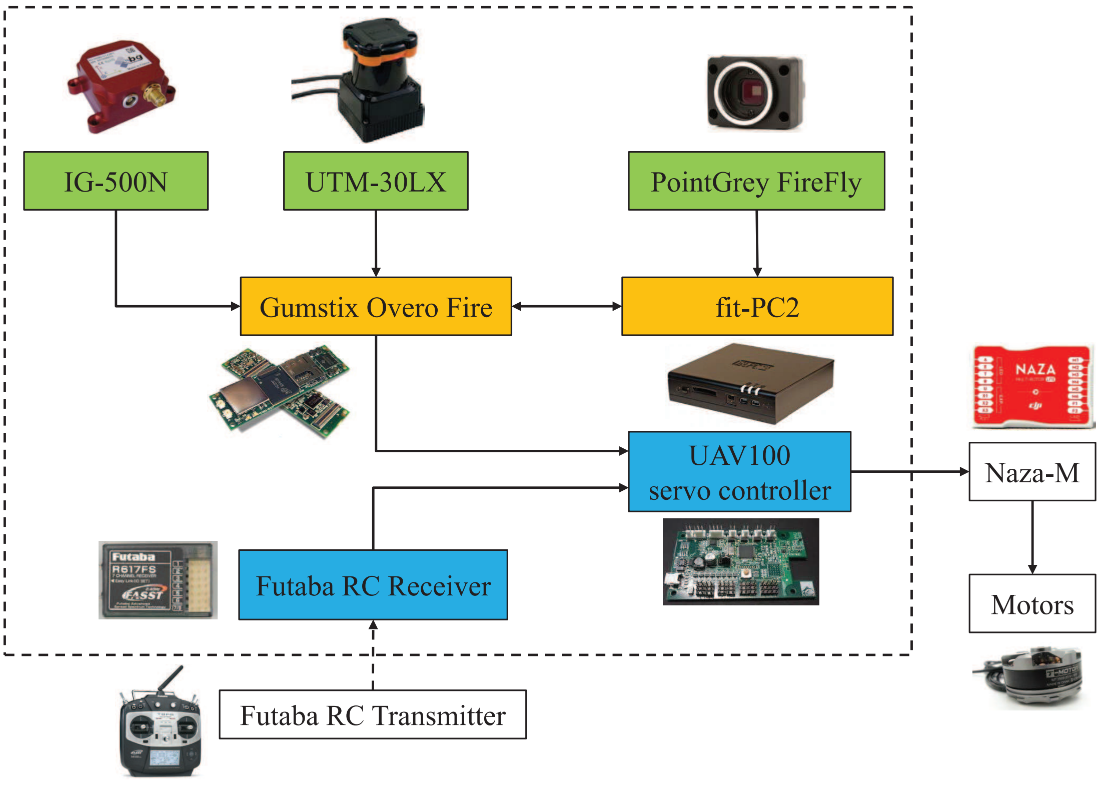


Figure 3.23: Onboard avionics configuration of the quadrotor platform

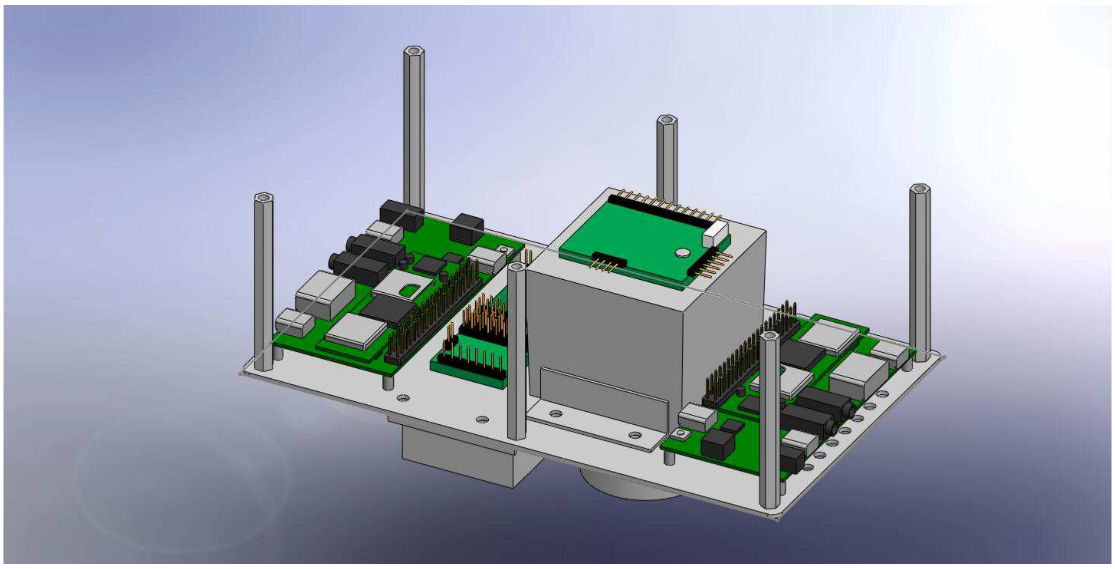


Figure 3.24: SolidWorks design for the coaxial avionics

alignment is difficult to be designed perfectly due to practical issues. If it happens, software compensation has to be implemented to minimize the measurement error caused by this vertical offset.

Then is to design the placement of laser scanner and camera. In theory, all sensors are preferred to be placed near to the CG of the UAV so that rotation of the UAV will not induce large translation of the sensors. However, it is impossible to position all sensors at the same place, and laser scanner and vision sensor will have occlusion problems if they are put very inside of the UAV body. In view of this, the laser scanner for the quadrotor platform is positioned at a top-center position while the laser scanner for the coaxial platform is up-side-down mounted at a bottom-center position. The cameras for both case are installed at the frontal part of the UAV platforms.

To facilitate the design process, a virtual 3D drawing software, SolidWorks, is used before carrying out the actual platform assembly. Such a software aided design method avoids the problem of unnecessary redesigning because of careless mistakes, thus saves a lot of time. Figs. 3.24–3.26 illustrate the virtual assembly of the avionic components of the coaxial platform and the quadrotor platform via SolidWorks respectively.

3.8 Hardware Assembly Results

In this chapter, a comprehensive survey of avionics components for indoor UAVs has been given. Based on the specifications of two different platforms, two avionics configurations have

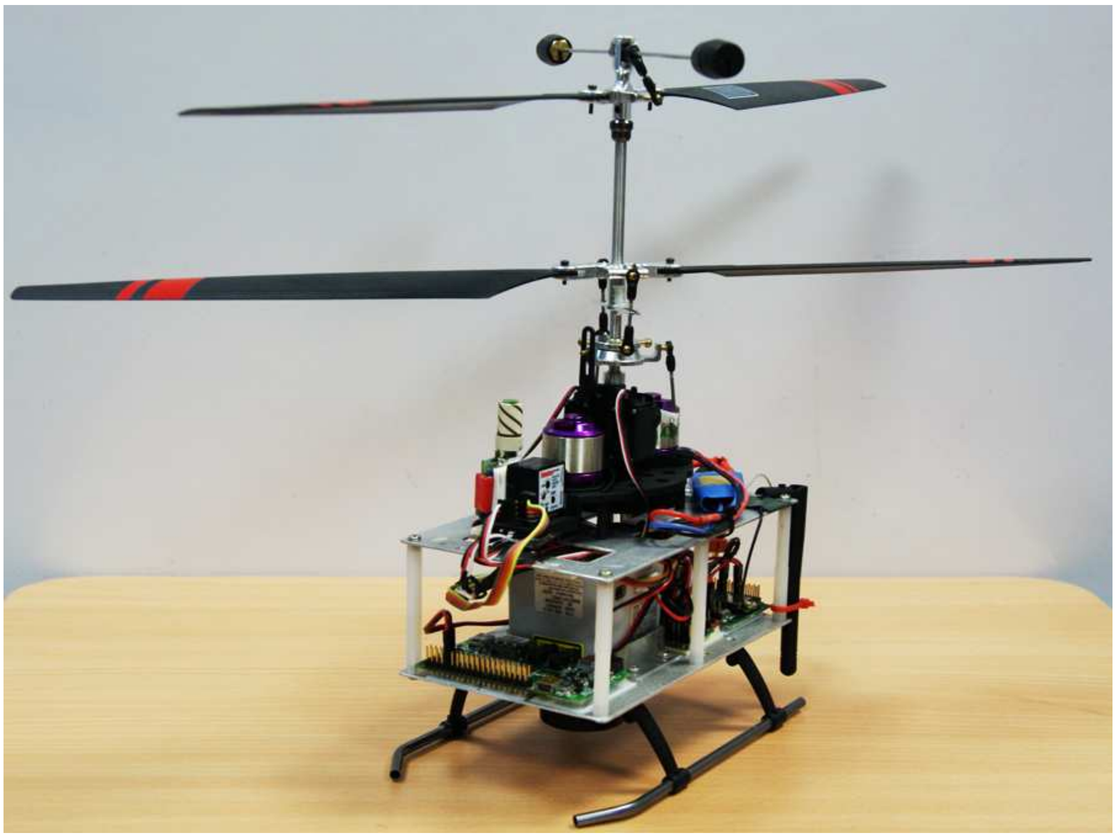


Figure 3.25: Physical view of the fully assembled coaxial platform

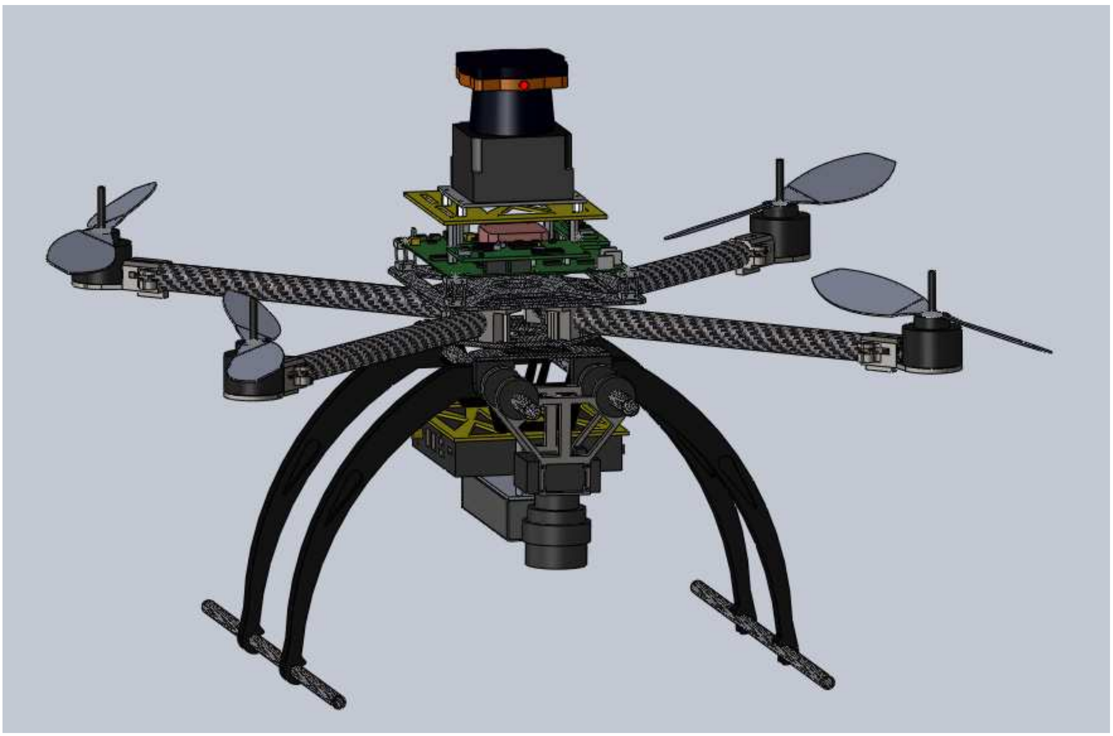


Figure 3.26: SolidWorks design for the whole quadrotor platform

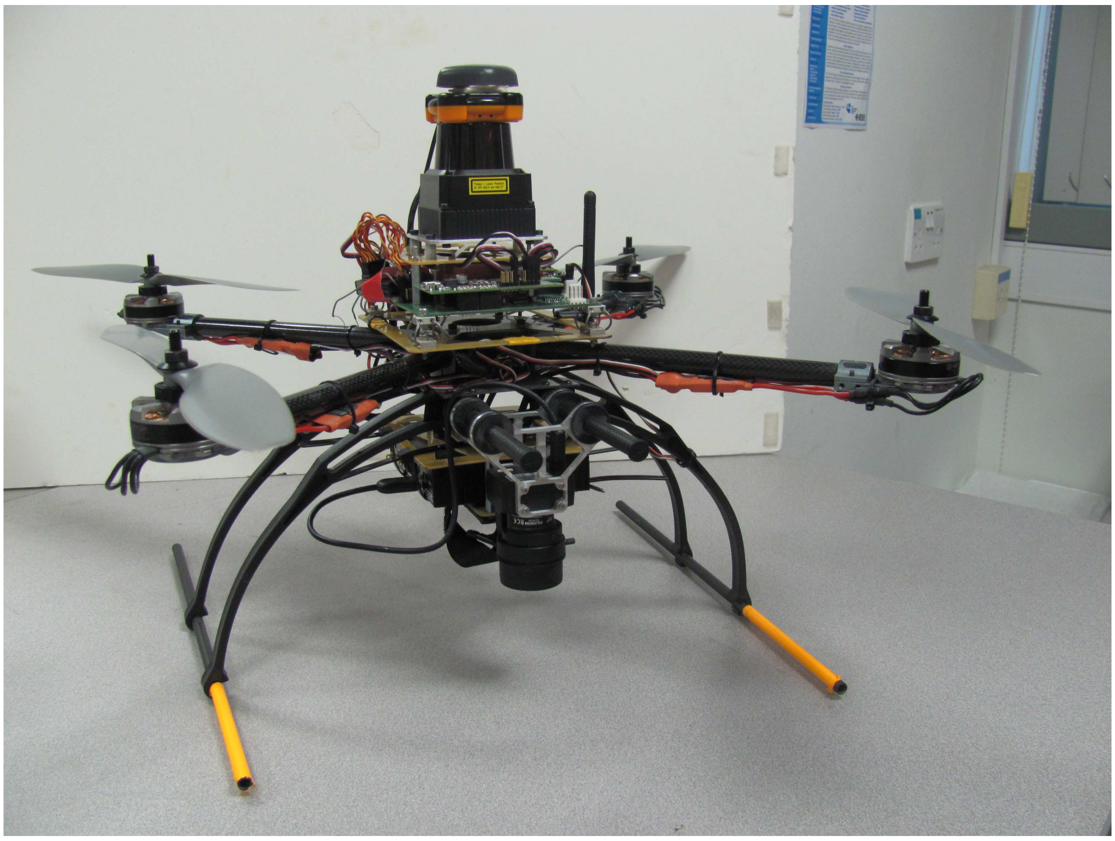


Figure 3.27: Physical view of the fully assembled quadrotor platform

been determined. The coaxial platform carries lighter and simpler avionics, while the quadrotor platform carries a much more powerful set. A 3-D drawing software SolidWorks has been utilized to design the placement and mounting of the avionics system. Till now, two fully functional indoor UAV platforms with their respective onboard sensors and computers have been assembled and tested. Figs. 3.25–3.27 show the physical assembly of the hardware platforms. Both platforms can hover at about ‘half throttle’ with all avionics mounted, and all sensor data can be decoded and logged by the onboard computers.

Chapter 4

Modeling and Control of a Coaxial Helicopter

To enable a UAV to navigate in indoor environments fully autonomous, the first thing needs to be ensured is the platform's attitude stability and its capability of way point tracking. This requires the design of high-performance robust flight control laws so that sufficient position control precision can be guaranteed in a confined indoor space. Else, the high-level navigation algorithms will have no foundation to build upon.

As most modern control methods are model based, a precise dynamic model of the controlled object is needed. In this chapter, the model formulation and parameter identification of a coaxial helicopter will be presented. It not only serves as the basis for the design of flight control laws for indoor navigation purposes, but also complements the existing works of nonlinear modeling of UAVs, as the study of coaxial helicopter modeling is much less substantial in literature as compared with other types of aerial platforms, such as the conventional single-rotor helicopter. In a few recent works, although fairly complete linear or nonlinear models for coaxial helicopters are obtained [19, 21], their publications lack intuitive explanation of the model formulation and their methods of parameter identification are not detail enough. For example in [52], the helicopter dynamics were treated as a black box, while the whole system is vaguely identified using an existing model fitting toolkit. Moreover, a lot of works only concentrate on a few parts of the coaxial helicopter dynamic model without combining them together for the ultimate control purpose [63, 37]. Very complete modeling work for a miniature coaxial helicopter can only be found in [69]. To complement the research work in this area, this chapter presents a detailed derivation of the nonlinear model for a fixed-pitch coaxial helicopter, together with

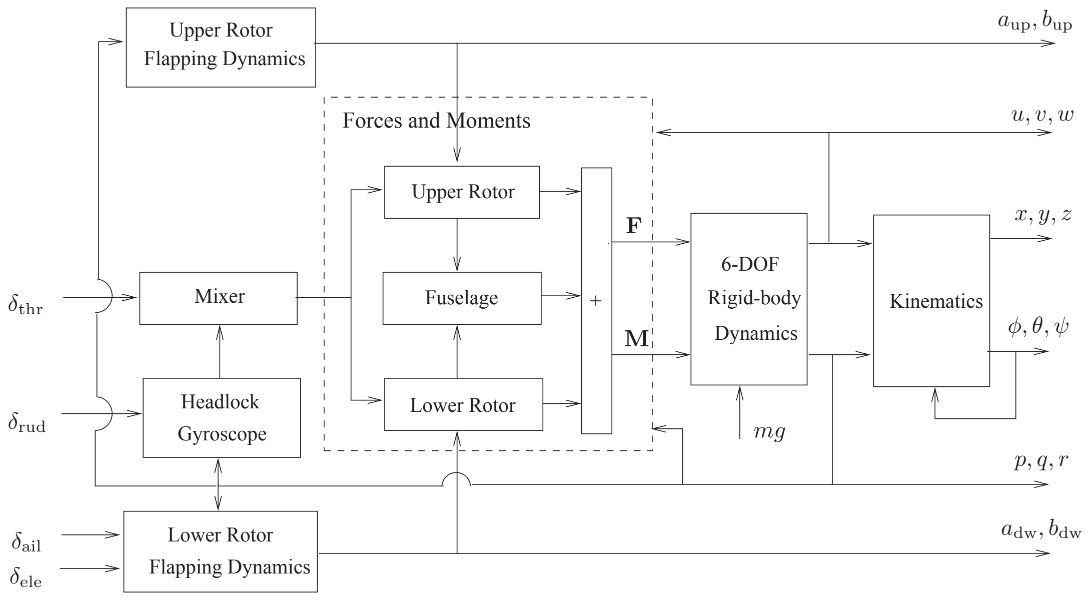


Figure 4.1: Overview of the coaxial helicopter model structure

experimental methods proposed to identify the key parameters in the model.

4.1 Basic Working Principle and Model Overview

For a fixed-pitch coaxial helicopter, the collective pitch of the rotor blades cannot be changed. Heave and yaw motion of the helicopter can only be achieved by varying the rotational speed of the rotors, which are controlled by two separate motors. Generally, the summation of the motor speeds determines the helicopter vertical motion, while the difference of the two determines the yaw motion. Rolling and pitching are accomplished by introducing a slanted orientation of the swashplate, which is controlled by the aileron and elevator servos. In this way, a tilted flapping of the rotor blades can be induced, and the resulting thrust generated becomes non-vertical.

An overview of the model structure is shown in Fig. 4.1 in which $\delta_{\rm ail}$, $\delta_{\rm ele}$, $\delta_{\rm thr}$ and $\delta_{\rm rud}$ are the aileron, elevator, throttle and rudder inputs to the dynamic system respectively. State variables can be found at the right side of the model structure. From the inputs to the state variables, there are numerous blocks representing individual sub-systems. In the next section of this chapter, model formulation in all these blocks will be explained in detail.

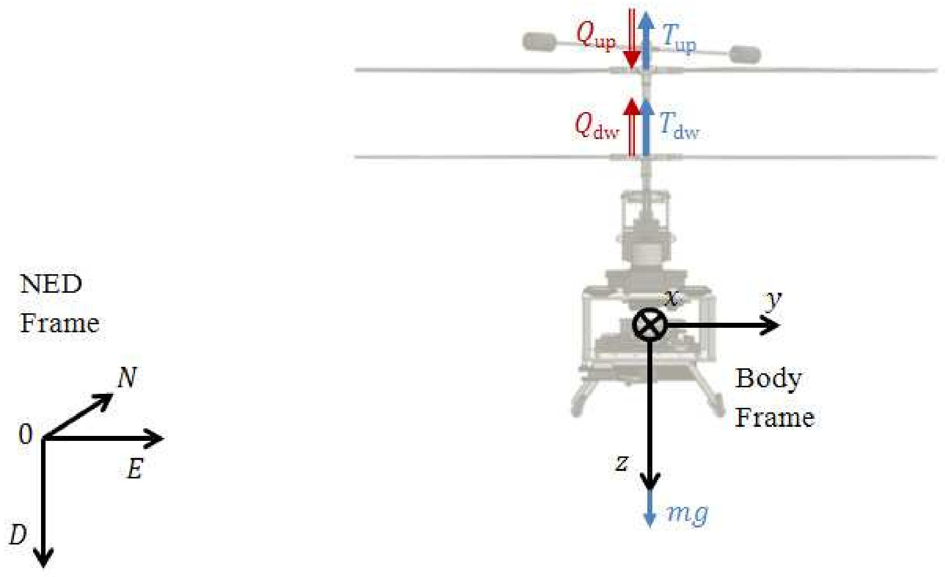


Figure 4.2: The NED and body coordinate frame systems

4.2 Model Formulation and Parameter Identification

Coordinate Systems and Rigid-Body Dynamics

As a common practice of aeronautic analysis, two main coordinate frames will be used here. One is the North-East-Down (NED) frame and the other is the helicopter body frame. While the NED frame is stationary with respect to a static observer on the ground, the body frame is placed at the Center of Gravity (CG) of the coaxial helicopter, where its origin and orientation move together with the helicopter fuselage (see Fig. 4.2). The following navigation equation shows the relationship between the NED-frame position and the body-frame velocity:

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{pmatrix} = \begin{bmatrix} c_\psi c_\theta & c_\psi s_\theta s_\phi - s_\psi c_\phi & c_\psi s_\theta c_\phi + s_\psi s_\phi \\ s_\psi c_\theta & s_\psi s_\theta s_\phi + c_\psi c_\phi & s_\psi s_\theta c_\phi - c_\psi s_\phi \\ -s_\theta & c_\theta s_\phi & c_\theta c_\phi \end{bmatrix} \begin{pmatrix} u \\ v \\ w \end{pmatrix}, \tag{4.1}$$

where x, y, z are the NED-frame position components of the helicopter, u, v, w are the body-frame velocity components, ϕ, θ, ψ are the roll, pitch, yaw angles of the helicopter fuselage and s_*, c_* denote $\sin(*)$, $\cos(*)$ respectively. It is also critical to point out that the Euler angle derivatives, $\dot{\phi}$, $\dot{\theta}$, $\dot{\psi}$, are not orthogonal to each other. They are related to the body frame angular rates, p, q, r, by the following kinematic equation:

$$\begin{pmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{pmatrix} = \begin{bmatrix} 1 & s_\phi s_\theta / c_\theta & c_\phi s_\theta / c_\theta \\ 0 & c_\phi & -s_\phi \\ 0 & s_\phi / c_\theta & c_\phi / c_\theta \end{bmatrix} \begin{pmatrix} p \\ q \\ r \end{pmatrix}. \tag{4.2}$$

Note that the above equation has singularity at $\theta = 90°$. If full-envelope flight is required, a quaternion representation is recommended. However, since the coaxial helicopter will be flying at near-hover condition for the task of indoor navigation, it is still adequate to use (4.2) in this work.

By treating the whole coaxial platform as a rigid mass, the 6 degrees-of-freedom (DOF) motion can be described by the following Newton-Euler equations:

$$\begin{pmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{pmatrix} = \frac{1}{m} \begin{pmatrix} F_x \\ F_y \\ F_z \end{pmatrix} - \begin{pmatrix} p \\ q \\ r \end{pmatrix} \times \begin{pmatrix} u \\ v \\ w \end{pmatrix}, \tag{4.3}$$

$$\begin{pmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{pmatrix} = \mathbf{J}^{-1} \left\{ \begin{pmatrix} M_x \\ M_y \\ M_z \end{pmatrix} - \begin{pmatrix} p \\ q \\ r \end{pmatrix} \times \mathbf{J} \begin{pmatrix} p \\ q \\ r \end{pmatrix} \right\}, \tag{4.4}$$

where F_x, F_y, F_z are projections of the net force, $\mathbf{F}$, onto the body-frame x-, y-, z-axis, and M_x, M_y, M_z are projections of the net torque, $\mathbf{M}$, onto the body-frame x-, y-, z-axis. The compositions of $\mathbf{F}$ and $\mathbf{M}$ come from various parts of the coaxial helicopter and will be explained in detail later. The center of gravity (CG) of the helicopter can be determined by hanging the platform in two different directions (see Fig.4.3) and examine the intersection of suspension lines. The total mass of the platform, m, can be easily measured, while $\mathbf{J}$ is the moment of inertia of the platform, which is in the form of

$$\mathbf{J} = \begin{bmatrix} J_{xx} & -J_{xy} & -J_{xz} \\ -J_{xy} & J_{yy} & -J_{yz} \\ -J_{xz} & -J_{yz} & J_{zz} \end{bmatrix}.$$

Since the coaxial helicopter is almost symmetric in both longitudinal and lateral directions, J_{xy}, J_{xz}, J_{yz} are extremely small and can be ignored. J_{xx}, J_{yy}, J_{zz} can be measured by the

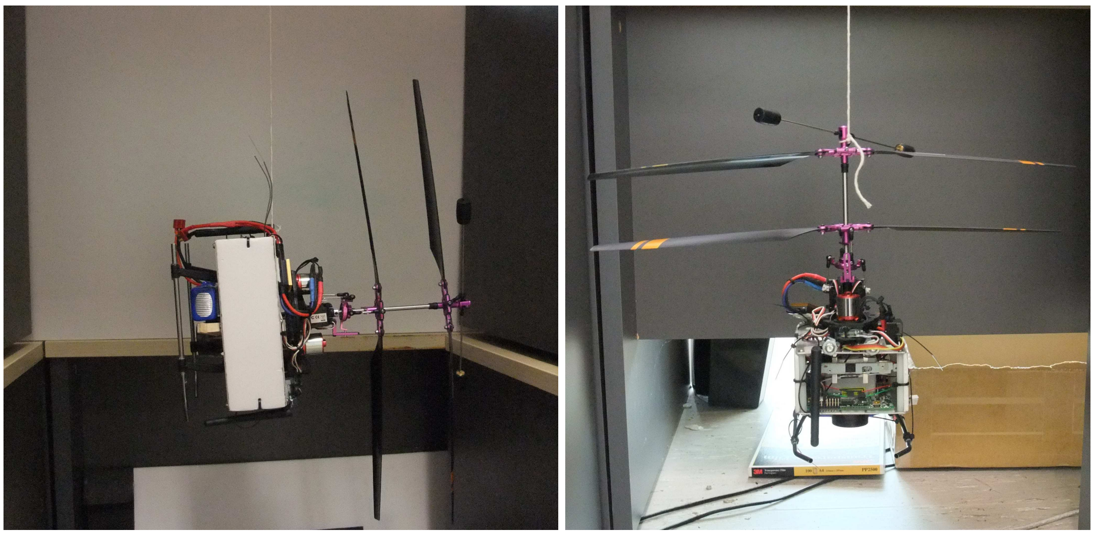


Figure 4.3: Hanging the platform to determine its CG

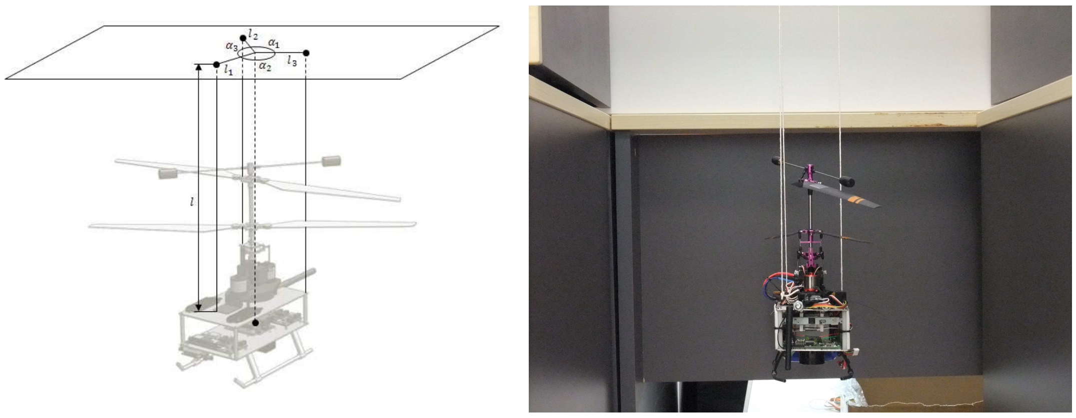


Figure 4.4: The trifilar pendulum method in helicopter z-axis

trifilar pendulum method proposed in [31]. The experimental setup is shown in Fig. 4.4. In this experiment, the coaxial platform is suspended by three flexible strings with equal length l. The horizontal distances between the attached points and the CG are l_1, l_2 and l_3 respectively. The platform can be slightly twisted and released around the vertical axis and then record its oscillation period t_l. The moment of inertia in this axis can be calculated as:

$$J_{zz} = \frac{mgl_1l_2l_3t_l^2}{4\pi^2 l} \cdot \frac{l_1 \sin\alpha_1 + l_2 \sin\alpha_2 + l_3 \sin\alpha_3}{l_2l_3 \sin\alpha_1 + l_1l_3 \sin\alpha_2 + l_1l_2 \sin\alpha_3}, \tag{4.5}$$

where α_1, α_2 and α_3 are the angles denoted in Fig. 4.4. Similar experiments can be done to obtain the moment of inertia around the other two axes (see Fig. 4.5).

Force and Torque Composition

As mentioned in the previous sub-section, force and torque acting on the coaxial helicopter come from various mechanical parts. First of all, the helicopter weight exerts a force of mg in

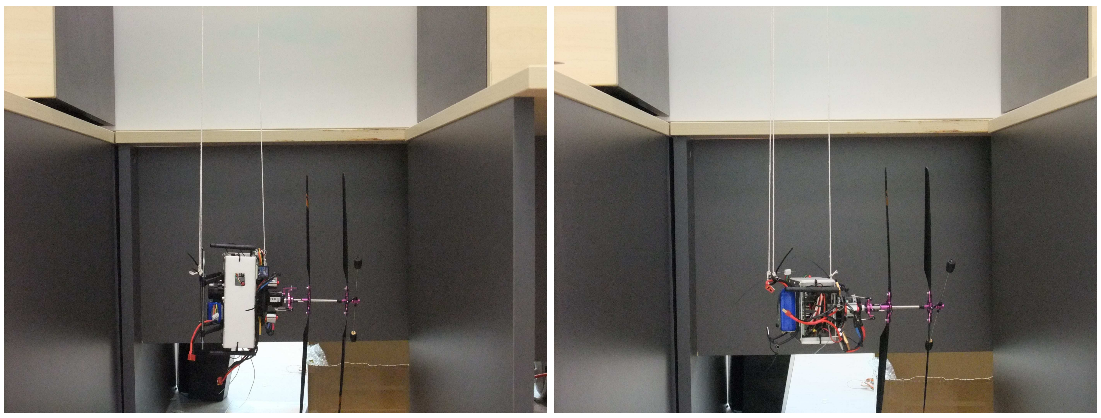


Figure 4.5: The trifilar pendulum method in helicopter x- and y-axis

the NED-frame z-axis. After converting it to the body frame, the vector is shown as the second term on the right hand side of (4.6).

Next, when the rotor blades spin, they generate thrusts, $\boldsymbol{T}_i$ (i = up, dw) in the direction perpendicular to their respective tip-path-plane (TPP). When the upper and lower TPPs deviate from their default orientation, the thrust vectors no longer pass through the CG of the helicopter, thus creating rotational torque. The torque vectors caused by the rotor thrusts can be calculated by $\boldsymbol{l}_{\text{up}} \times \boldsymbol{T}_{\text{up}}$ and $\boldsymbol{l}_{\text{dw}} \times \boldsymbol{T}_{\text{dw}}$, where $\boldsymbol{l}_{\text{up}}$ and $\boldsymbol{l}_{\text{dw}}$ are the displacement vectors from helicopter CG to the upper rotor hub and the lower rotor hub respectively. The deviation of the TPP can be described by the longitudinal flapping angle a_i and the lateral flapping angle b_i. The thrust decomposition to the body-frame axes can be approximated by the second equation in (4.8). Non-zero a_i and b_i also directly result in flapping torque on the rotor hub. This torque can be simplified as the second term on the right hand side of (4.7), where K_β is the effective spring constant and it has the same value for both the upper and lower rotors.

At the same time, the rotation of the rotors also creates the drag torque, $\boldsymbol{Q}_{\text{d,up}}$ and $\boldsymbol{Q}_{\text{d,dw}}$, around the body-frame z-axis. When the coaxial helicopter hovers without yaw motion, the two torques have the same magnitude, thus canceling each other. Else, if the net drag torque is non-zero, yaw acceleration is generated. In addition, the change of rotational speeds of the rotors also generate the reaction torques on the helicopter body (denoted by $\boldsymbol{Q}_{\text{r,up}}$ and $\boldsymbol{Q}_{\text{r,dw}}$). They are described in (4.10), where J_{up} and J_{dw} are the moment of inertia of the upper rotor (with stabilizer bar) and the lower rotor with respect to the axis of rotor shaft. They can be calculated by measuring the mass and dimension of the rotor blades and stabilizer bar and assuming a regular geometric shape.

Last but not least, when the helicopter moves in air, its fuselage experiences drag forces,

X_{fus}, Y_{fus}, Z_{fus}, due to air resistance. Equation (4.6) and (4.7) have summarized all the forces and torques mentioned above, with (4.8)-(4.10) explaining how to evaluate the individual terms:

$$\begin{pmatrix} F_x \\ F_y \\ F_z \end{pmatrix} = \sum \boldsymbol{T}_i + mg \begin{pmatrix} -s_\theta \\ s_\phi c_\theta \\ c_\phi c_\theta \end{pmatrix} + \begin{pmatrix} X_{\text{fus}} \\ Y_{\text{fus}} \\ Z_{\text{fus}} \end{pmatrix}, \tag{4.6}$$

$$\begin{pmatrix} M_x \\ M_y \\ M_z \end{pmatrix} = \sum \boldsymbol{l}_i \times \boldsymbol{T}_i + \sum K_\beta \begin{pmatrix} b_i \\ a_i \\ 0 \end{pmatrix} + \sum \boldsymbol{Q}_{\text{d},i} + \sum \boldsymbol{Q}_{\text{r},i}\,, \tag{4.7}$$

$$\boldsymbol{l}_i = |\boldsymbol{l}_i| \begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix}, \quad \boldsymbol{T}_i = |\boldsymbol{T}_i| \begin{pmatrix} -\sin a_i \\ \sin b_i \\ -\cos a_i \cos b_i \end{pmatrix}, \tag{4.8}$$

$$\boldsymbol{Q}_{\text{d,up}} = |\boldsymbol{Q}_{\text{d,up}}| \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, \quad \boldsymbol{Q}_{\text{d,dw}} = |\boldsymbol{Q}_{\text{d,dw}}| \begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix}, \tag{4.9}$$

$$\boldsymbol{Q}_{\text{r,up}} = J_{\text{up}}\dot{\Omega}_{\text{up}} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, \quad \boldsymbol{Q}_{\text{r,dw}} = J_{\text{dw}}\dot{\Omega}_{\text{dw}} \begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix}. \tag{4.10}$$

Thrust and Torque from Rotors

In this sub-section, the magnitude of the rotor thrust and drag torque, $|\boldsymbol{T}_i|$ and $|\boldsymbol{Q}_{\text{d},i}|$, will be investigated. According to the aerodynamic actuator disk theory [14], the magnitude of thrust generated by the rotors can be formulated as follows:

$$|\boldsymbol{T}_i| = \rho C_{T,i} A(\Omega_i R)^2, \tag{4.11}$$

where ρ is the density of air, $C_{T,i}$ is the lift coefficient, A is the rotor disk area, Ω_i is the rotational speed of the rotor and R is the rotor blade length. Since this is a fixed-pitch coaxial

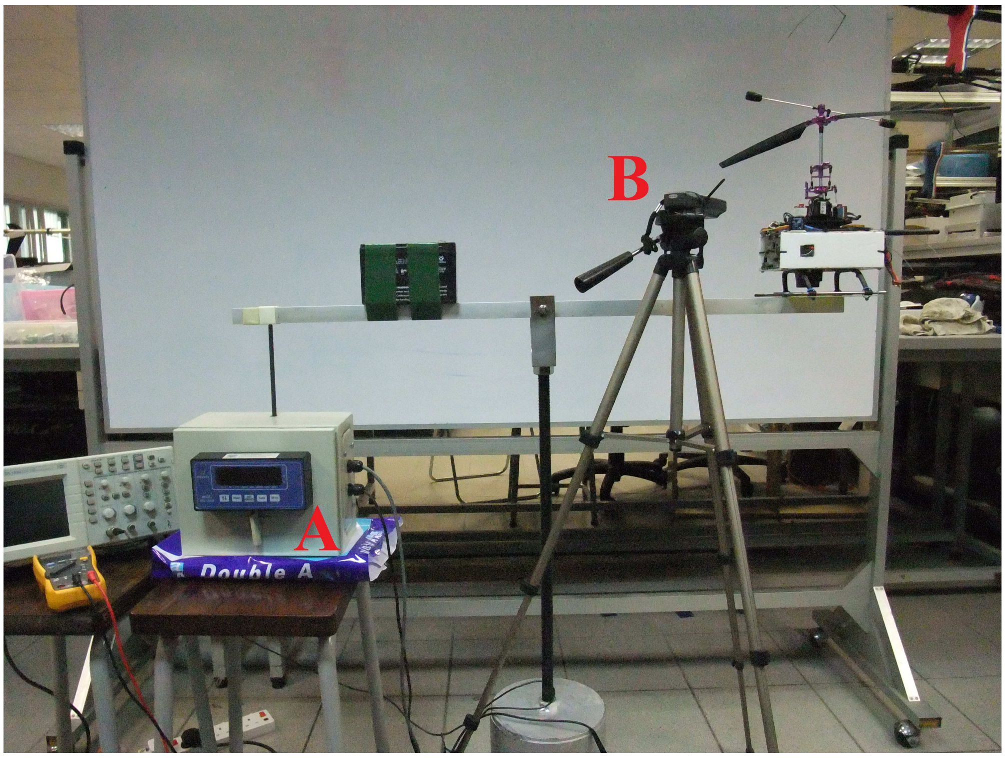


Figure 4.6: Setup to investigate relation between thrust and rotor speed

helicopter, $C_{T,i}$, like the other parameters in (4.11), is constant. The only variable is Ω_i. Hence, the equation can be simplified to:

$$|\boldsymbol{T}_i| = k_{T,i}\Omega_i^2, \tag{4.12}$$

where $k_{T,i}$ is a lumped thrust coefficient that needs to be identified. Similar assumptions and formulation can be applied to the relationship between the drag torque and the rotational speed of the rotors:

$$|\boldsymbol{Q}_{\mathrm{d},i}| = k_{Q,i}\Omega_i^2. \tag{4.13}$$

To identify $k_{T,i}$ and $k_{Q,i}$, two test bench experiments were carried out (see Fig. 4.6 and Fig. 4.7). The main measurement sensors include a force meter (A) and a tachometer (B). For the thrust experiment, results are summarized in Fig. 4.8. There are four lines in the plot, in which two of them (solid lines) perfectly match. They represent the cases when only one rotor (upper rotor or lower rotor) is rotating. The dashed line on the top is a numerical combination of the two solid lines, while the dash-dot line comes from actual tests with both rotors spinning at the same speed. The gap between the two lines shows a drop in thrust efficiency caused by

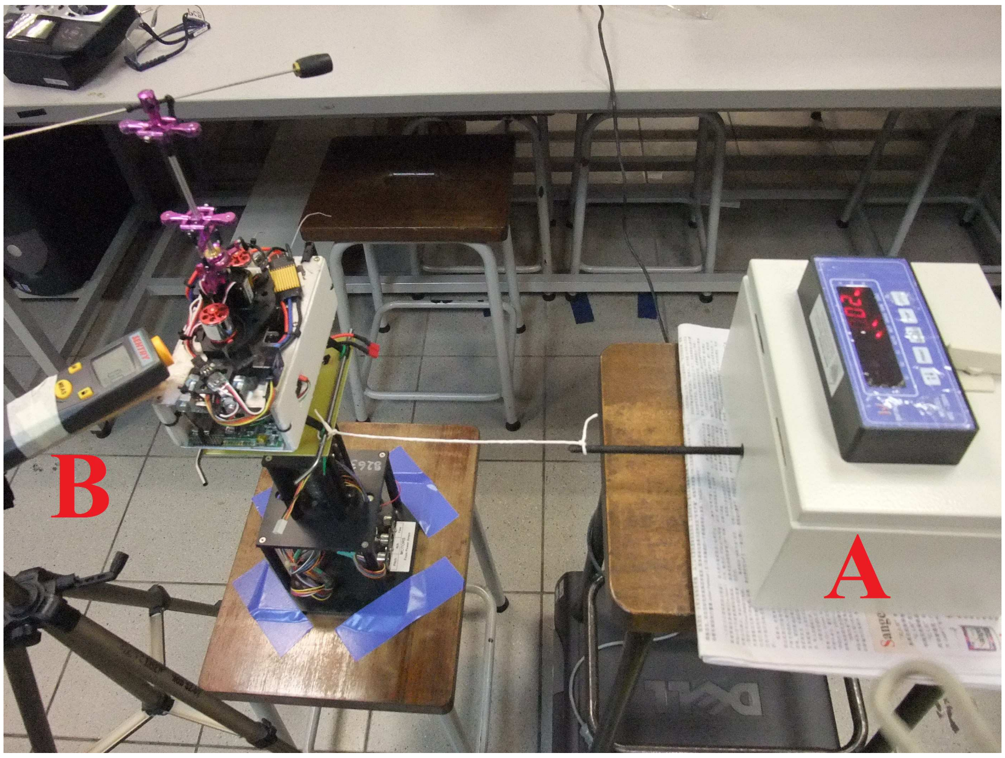


Figure 4.7: Setup to investigate relation between torque and rotor speed

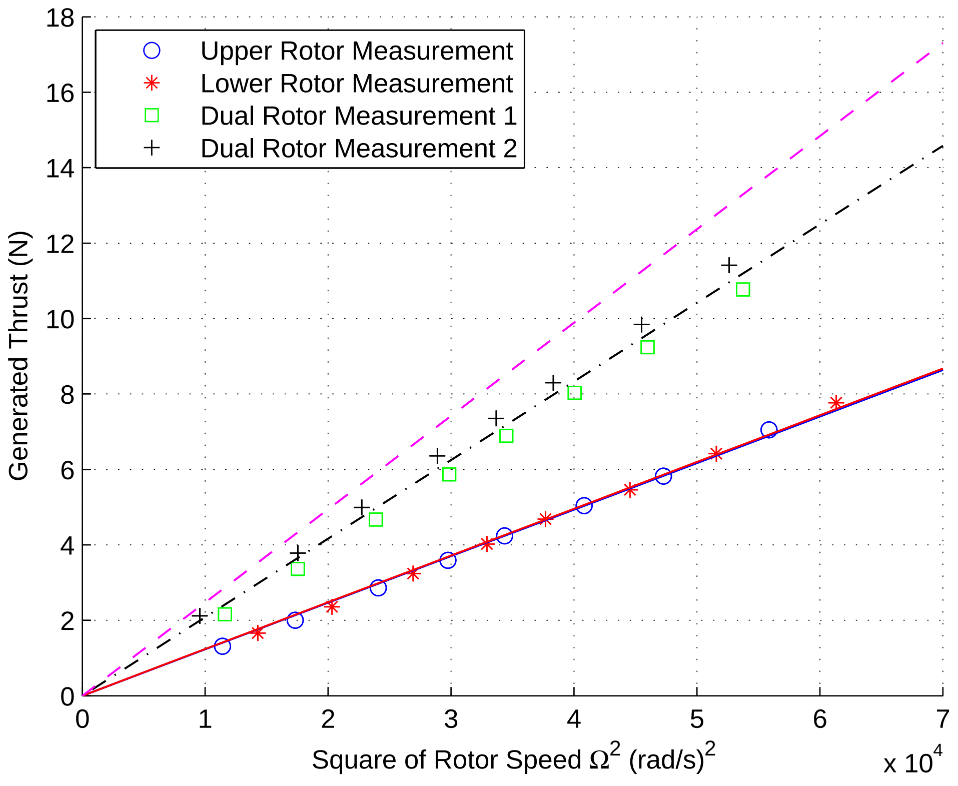


Figure 4.8: Data plot of thrust against square of rotor speed

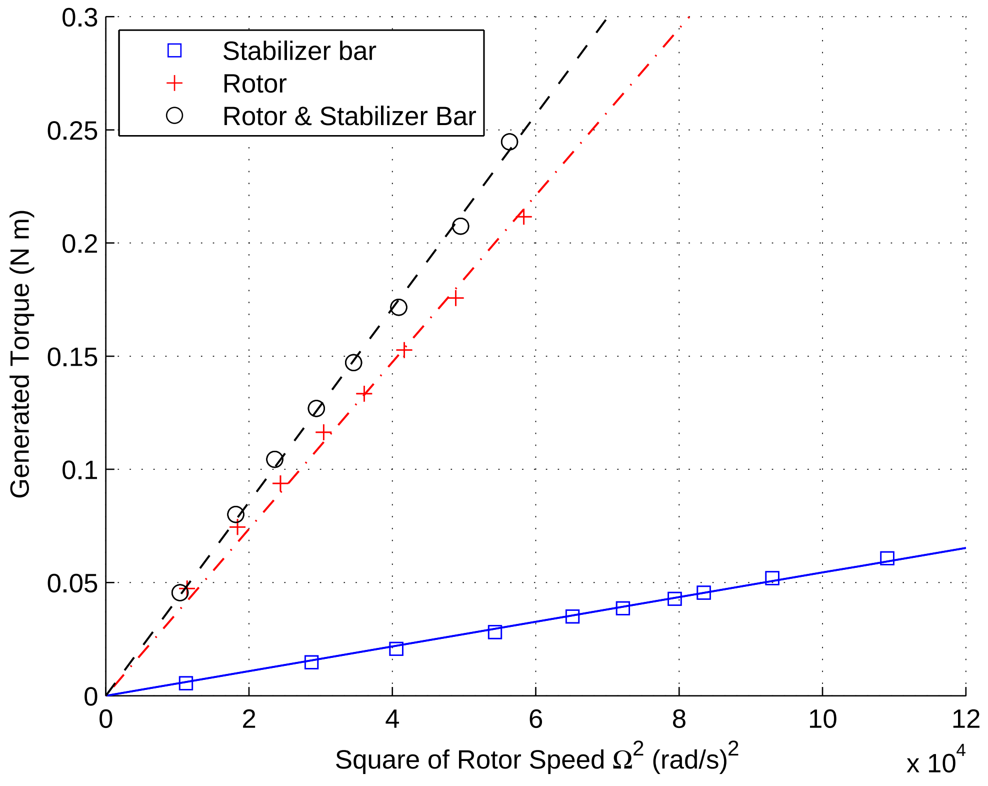


Figure 4.9: Data plot of torque against square of rotor speed

aerodynamic interactions between the two rotors. According to [22], for a coaxial helicopter operating in near-hover condition, the induced-velocity effect of the upper rotor to the lower rotor is significantly larger than that of the lower rotor to the upper rotor. Thus, the loss of thrust efficiency can be fully accounted on the lower rotor thrust coefficient. Hence, $k_{T,\mathrm{up}}$ is the gradient of the solid line and $k_{T,\mathrm{dw}}$ is the gradient difference between the dash-dot line and the solid line.

For the torque experiment, results are summarized in Fig. 4.9. The solid line represents the case when only the stabilizer bar is rotating, while the dash-dot line is for a single rotating rotor. The dashed line is generated with the upper rotor and the stabilizer bar spinning together. Unsurprisingly, it matches the numerical combination of the lower two lines. Thus, the gradient of the dashed line is $k_{Q,\mathrm{up}}$, and the gradient of the dash-dot line is $k_{Q,\mathrm{dw}}$.

Rotor Tip-Path-Plane Motion

For this type of coaxial helicopter, the rotor collective pitch is fixed, while its cyclic pitch can be changed. For the lower rotor, the rotor hub is connected to the aileron and the elevator

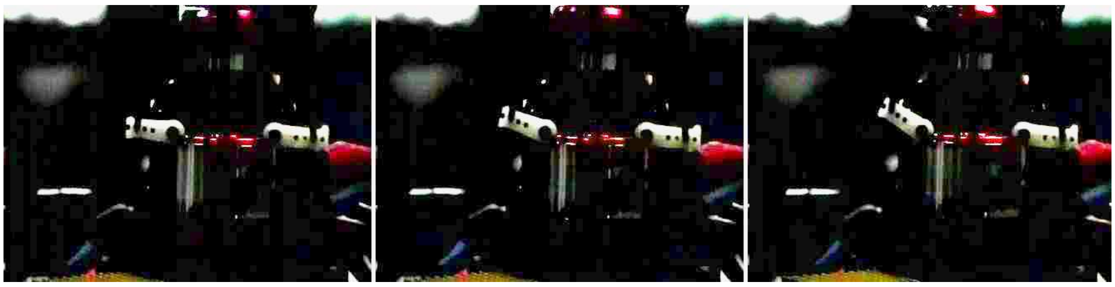


Figure 4.10: Step response of servo motion (Left: $t = 0$; Middle: $t = 0.0375$ s; Right: $t = \infty$)

servos via a swashplate. When the swashplate tilts, it teeters the rotor hub and creates a cyclic pitch on the rotor. For every cycle of rotation, the rotor blade will reach the maximum angle of attack at the same phase angle when the lift on the blade is largest. This results in the flapping of the rotor disk. The whole mechanism is a combination of gyroscopic precession and aerodynamic precession. For the case of the Esky Big Lama, if one observes the rotor blade in a slow motion, the maximum rotor flapping occurs roughly at $45°$ lag with respect to the occurrence of maximum angle of attack. This explains why the aileron and elevator servos of the off-the-shelf coaxial platform are connected to the swashplate $45°$ off the body-frame x-, y-axis. In this way, the aileron servo mainly controls the lateral flapping of the lower rotor, and the elevator servo mainly controls the longitudinal flapping. However, the flapping phase lag is not exactly equal to $45°$ (slightly larger than $45°$ from test bench observations) due to mechanical modifications to the original RC platform (original rotor blades have been replaced by stiffer ones for larger payload). This results in non-negligible coupling between the servo inputs and the lower rotor longitudinal and lateral flapping angles. As the lower rotor does not have any additional damping mechanism attached, its flapping process is almost instantaneous. By assuming a first order dynamics, the time constant can be observed via a high-speed camera. The result turns out to be 0.0375 second (see Fig. 4.10), which is very small as compared to dynamics happening in other parts of the coaxial helicopter, thus can be ignored. Hence, the relationship between servo inputs and lower rotor flapping angles can be formulated in a non-dynamic way:

$$a_{\text{dw}} = A_{a,\text{dw}}\,\delta_{\text{ele}} + A_{b,\text{dw}}\,\delta_{\text{ail}} - A_q\,q, \tag{4.14}$$

$$b_{\text{dw}} = B_{b,\text{dw}}\,\delta_{\text{ail}} + B_{a,\text{dw}}\,\delta_{\text{ele}} - B_p\,p, \tag{4.15}$$

where δ_{ail}, δ_{ele} are the servo inputs normalized to [-1, 1], $A_{a,\text{dw}}$ and $B_{b,\text{dw}}$ are the on-axis steady-state ratio from servo inputs to flapping angles, and $A_{b,\text{dw}}$ and $B_{a,\text{dw}}$ are the off-axis (coupling) values. The terms involving angular rates, p and q, come from an effect called rotor damping, which was also considered in [26].

For the upper rotor system, a stabilizer bar is attached to the rotor hub, so that they teeter together. As the stabilizer bar has large moment of inertia, it tends to remain at its original rotating plane. Hence, at the moment when the helicopter body tilts, the stabilizer bar TPP will remain at the level plane, thus creating a cyclic pitch on the upper rotor which leads to blade flapping. The torque generated by this flapping redresses the rotational motion of the helicopter and significantly stabilizes the whole platform attitude. Similar to the lower rotor system, the stabilizer bar is installed at $45°$ phase lead to the rotor blade. In this way, the maximum flapping happens at the direction that roughly counters the rotational motion of the helicopter. Again, there is coupling between the longitudinal and lateral channels because the flapping phase lag is not exactly $45°$. The following equations describe the aforementioned dynamics:

$$\begin{aligned}
\dot{\phi}_{\text{sb}} &= \frac{1}{\tau_{\text{sb}}}\,(\phi - \phi_{\text{sb}}), && (4.16)\\
\dot{\theta}_{\text{sb}} &= \frac{1}{\tau_{\text{sb}}}\,(\theta - \theta_{\text{sb}}), && (4.17)\\
a_{\text{up}} &= A_{a,\text{up}}\,(\theta_{\text{sb}} - \theta) + A_{b,\text{up}}\,(\phi_{\text{sb}} - \phi) - A_q\,q, && (4.18)\\
b_{\text{up}} &= B_{b,\text{up}}\,(\phi_{\text{sb}} - \phi) + B_{a,\text{up}}\,(\theta_{\text{sb}} - \theta) - B_p\,p, && (4.19)
\end{aligned}$$

where ϕ_{sb} and θ_{sb} are the roll and pitch angles of the stabilizer bar TPP, $A_{a,\text{up}}$ and $B_{b,\text{up}}$ are the on-axis steady-state ratio from the stabilizer bar teetering angles to the upper rotor flapping angles, and $A_{b,\text{up}}$ and $B_{a,\text{up}}$ are the off-axis (coupling) values. Again, the same rotor damping effects (terms depending on p and q) are considered for the upper rotor flapping dynamics.

For the identification of τ_{sb}, one can observe the transient step response of the stabilizer bar TPP (see Fig. 4.11) by a high-speed camera and record the time when the response reaches 63.1% of the overall amplitude. On-axis parameters $A_{a,\text{up}}$, $B_{b,\text{up}}$, $A_{a,\text{dw}}$ and $B_{b,\text{dw}}$ can be identified by measuring various angles (see Fig. 4.12 and Fig. 4.13) and assuming a linear relationship between each pair of them. For the other coupling values and K_β, they can be identified by analyzing flight test data with aileron and elevator channel perturbations (see Fig. 4.14 and Fig. 4.15). The software used for numerical analysis is called the Comprehensive Identification from FrEqency Responses (CIFER). It is a MATLAB-based software developed by NASA

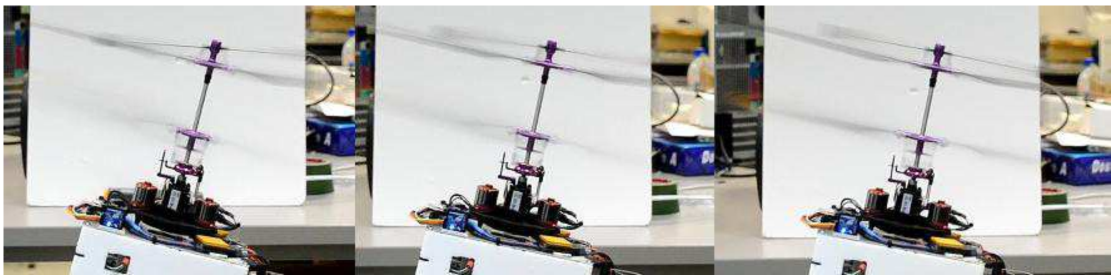


Figure 4.11: Step response of stabilizer bar (Left: $t = 0$; Middle: $t = 0.2$ s; Right: $t = \infty$)

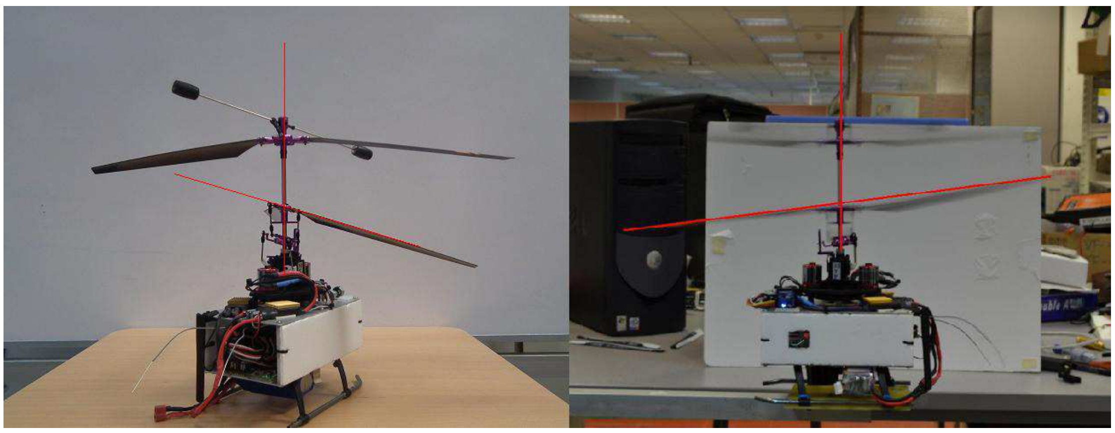


Figure 4.12: Left: Maximum teetering angle of the lower rotor hub; Right: Maximum flapping angle of the lower rotor

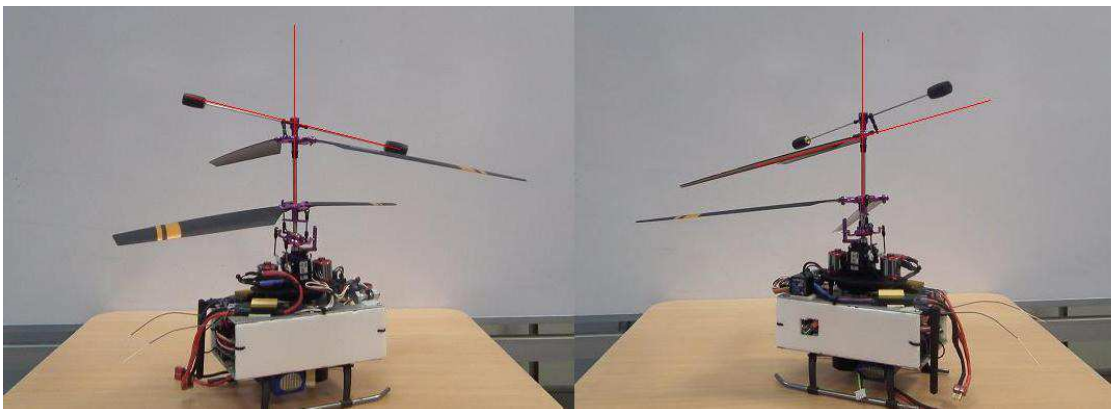


Figure 4.13: Left: Maximum teetering angle of the stabilizer bar; Right: Maximum teetering angle of the upper rotor hub

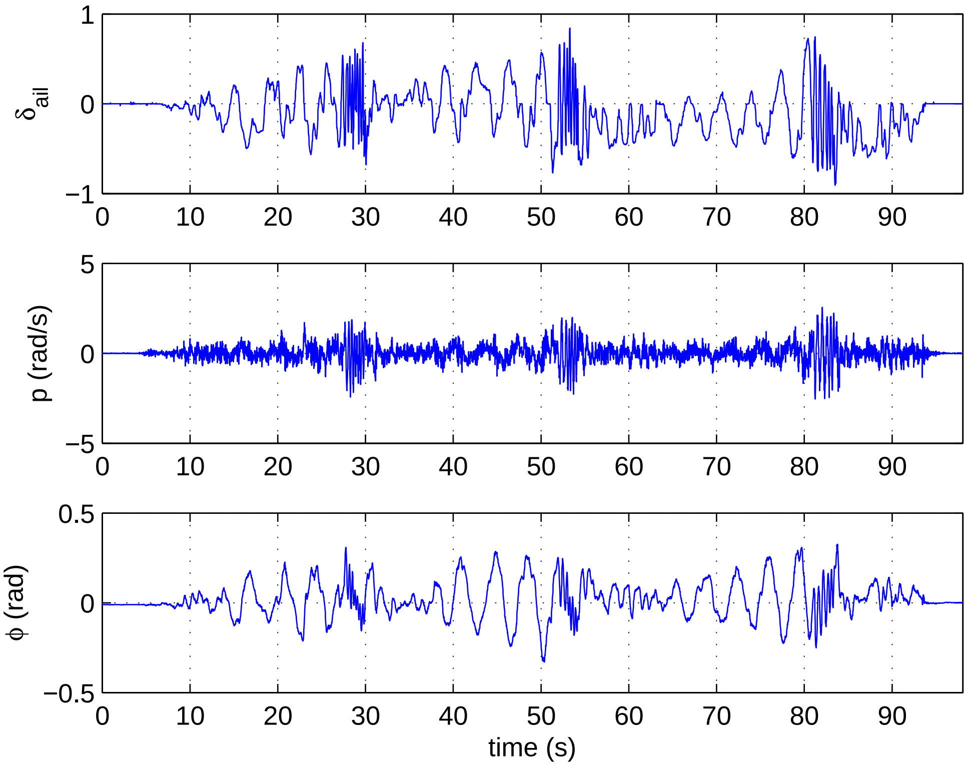


Figure 4.14: Manual flight data for aileron channel perturbation

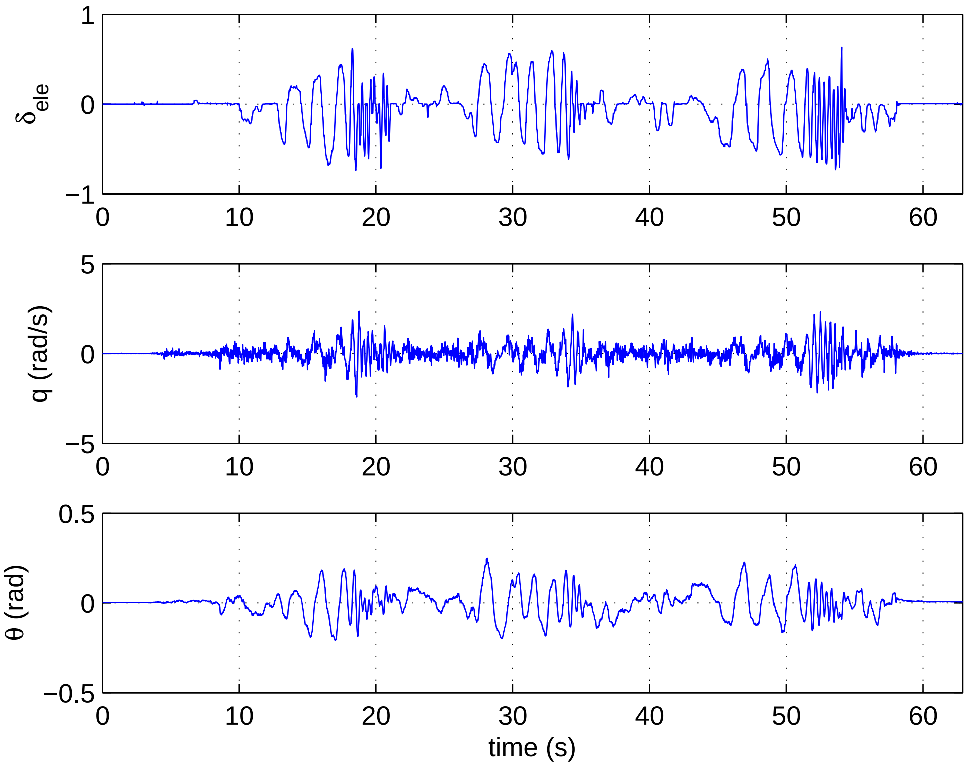


Figure 4.15: Manual flight data for elevator channel perturbation

Ames Research Center for military based rotorcraft system identifications. By combining and linearizing all the aforementioned equations related to angular rate dynamics and upper rotor flapping dynamics, the following linear state-space approximation can be obtained:

$$\dot{\begin{pmatrix} p \\ q \\ a_{\text{up}} \\ b_{\text{up}} \end{pmatrix}} = \begin{bmatrix} \frac{-X_{\text{dw}}B_{p,\text{dw}}}{J_{xx}} & 0 & 0 & \frac{X_{\text{up}}}{J_{xx}} \\ 0 & \frac{-X_{\text{dw}}A_{q,\text{dw}}}{J_{yy}} & \frac{X_{\text{up}}}{J_{yy}} & 0 \\ -A_{b,\text{up}} & -A_{a,\text{up}} & -\frac{1}{\tau_{\text{sb}}} & 0 \\ -B_{b,\text{up}} & -B_{a,\text{up}} & 0 & -\frac{1}{\tau_{\text{sb}}} \end{bmatrix} \begin{pmatrix} p \\ q \\ a_{\text{up}} \\ b_{\text{up}} \end{pmatrix} + \begin{bmatrix} \frac{X_{\text{dw}}B_{b,\text{dw}}}{J_{xx}} & \frac{X_{\text{dw}}B_{a,\text{dw}}}{J_{xx}} \\ \frac{X_{\text{dw}}A_{b,\text{dw}}}{J_{yy}} & \frac{X_{\text{dw}}A_{a,\text{dw}}}{J_{yy}} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{pmatrix} \delta_{\text{ail}} \\ \delta_{\text{ele}} \end{pmatrix}, \tag{4.20}$$

where $X_{\text{up}} = T_{\text{up}}l_{\text{up}} + K_\beta$ and $X_{\text{dw}} = T_{\text{dw}}l_{\text{dw}} + K_\beta$. By treating δ_{ail}, δ_{ele} as the inputs and p, q as the outputs (all can be logged during flight tests) and giving known constraints and reasonable initial values, CIFER helps to search for optimal numerical solution based on frequency response matching. A stable result with good matching is obtained as follows:

$$\dot{\begin{bmatrix} p \\ q \\ a_{\text{up}} \\ b_{\text{up}} \end{bmatrix}} = \begin{bmatrix} -17.19 & 0 & 0 & 934.1 \\ 0 & -5.360 & 291.3 & 0 \\ 0.2745 & -0.49 & -5 & 0 \\ -0.49 & -0.2745 & 0 & -5 \end{bmatrix} \begin{bmatrix} p \\ q \\ a_{\text{up}} \\ b_{\text{up}} \end{bmatrix} + \begin{bmatrix} -102.48 & -38.08 \\ -11.73 & 31.95 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \delta_{\text{ail}} \\ \delta_{\text{ele}} \end{bmatrix}, \tag{4.21}$$

With this set of numerical result, Fig. 4.16 to 4.19 show the corresponding comparison of frequency response between the data collected via actual flight tests and the CIFER derived model fit. For both the on-axis and off-axis responses, the matching is good, indicating a high-quality identification result. Next, By comparing (4.20) and (4.21), all the remaining parameters involved in angular rate and rotor flapping dynamics can be identified.

Fuselage Drag

When the helicopter fuselage moves in air, it experiences drag force acting on the opposite direction of the motion. For the body-frame horizontal directions, the rotor downwash is deflected by u and v. In the situation when u (or v) is less than v_i (the induced velocity of air at the lower rotor), the downwash effect needs to be taken into account. Otherwise, the downwash effect is

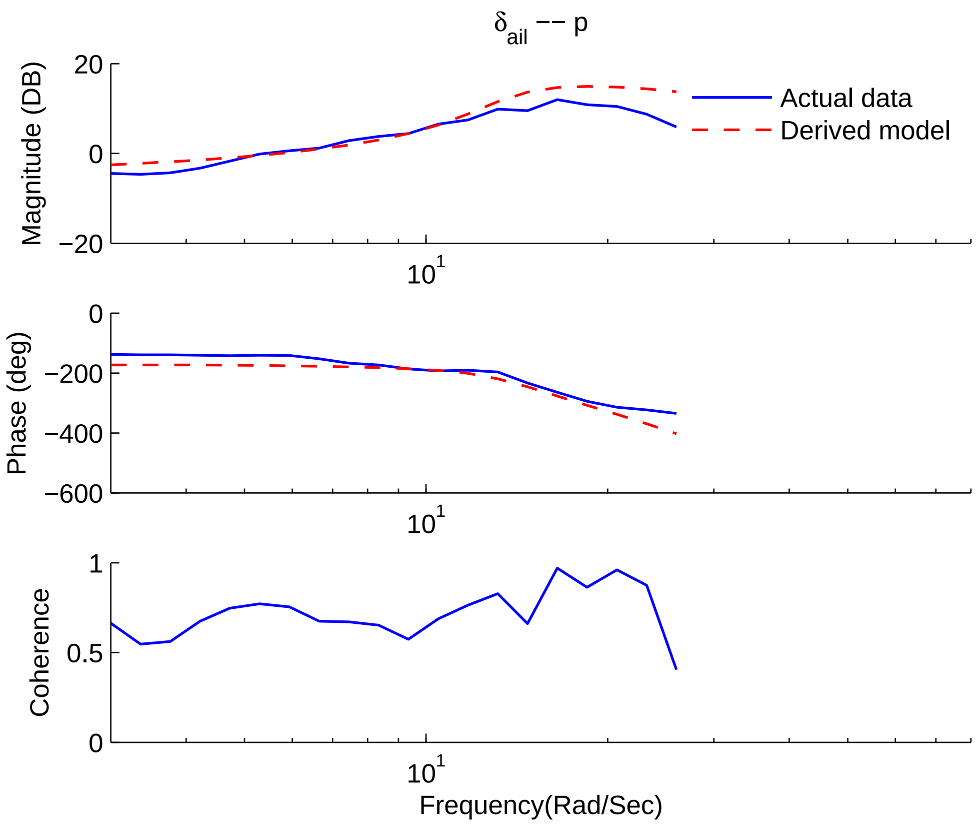


Figure 4.16: Response comparison using frequency-sweep input ($\delta_{\text{ail}} - p$)

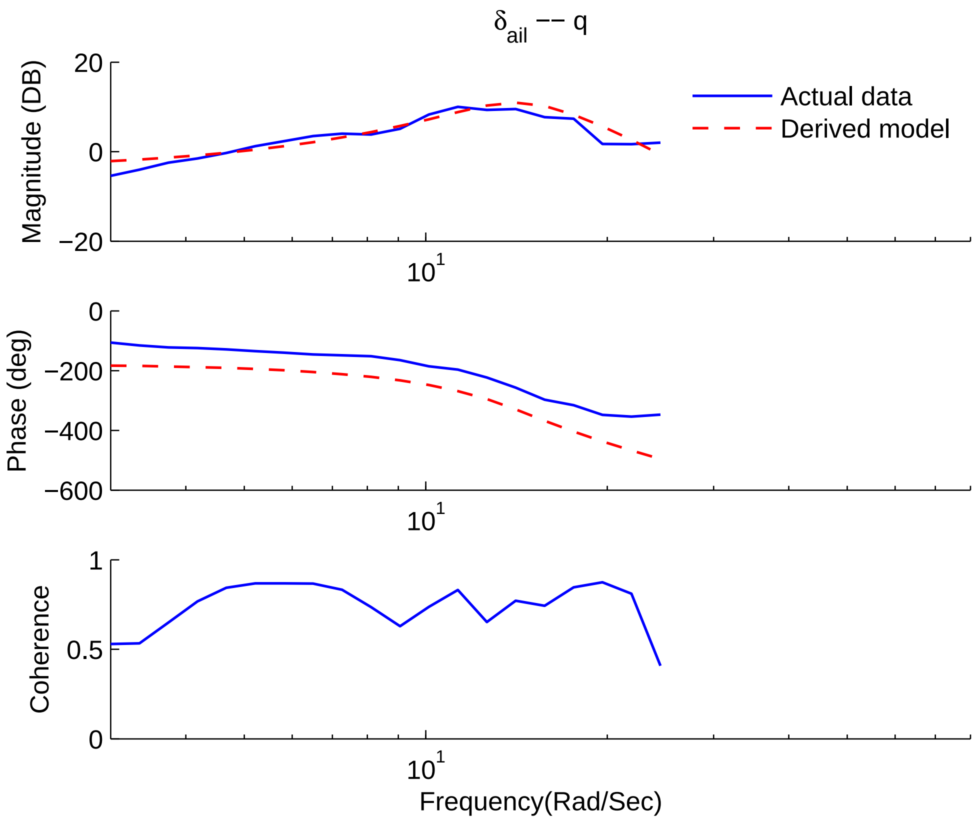


Figure 4.17: Response comparison using frequency-sweep input ($\delta_{\text{ail}} - q$)

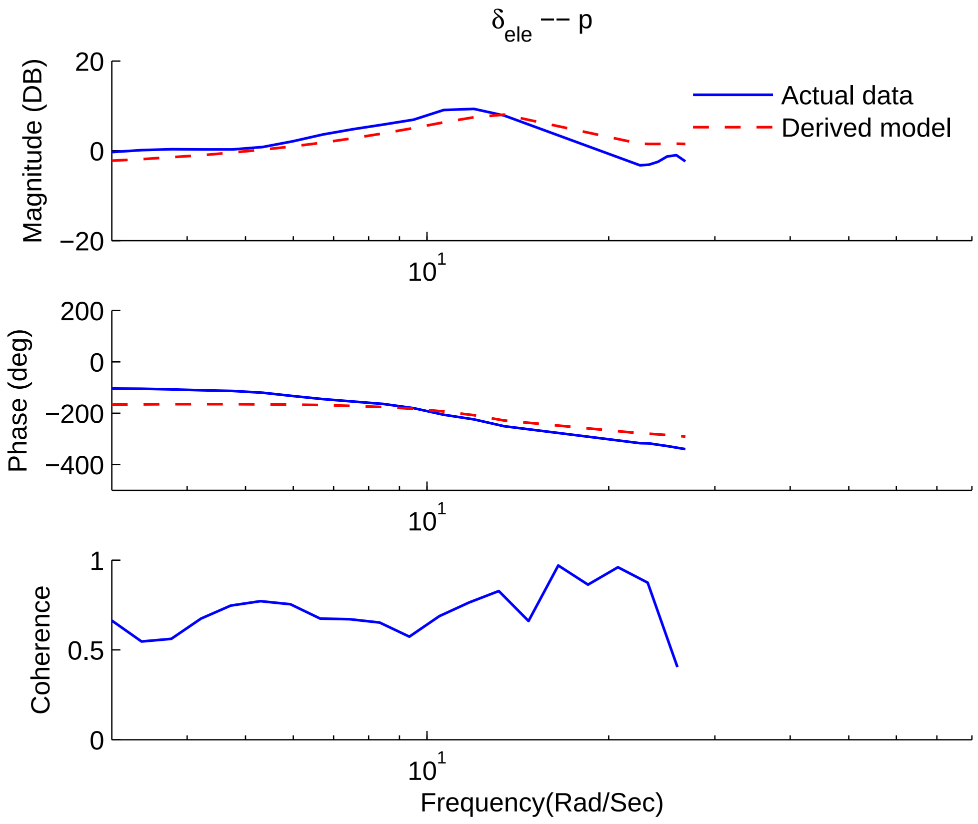


Figure 4.18: Response comparison using frequency-sweep input ($\delta_{\text{ele}} - q$)

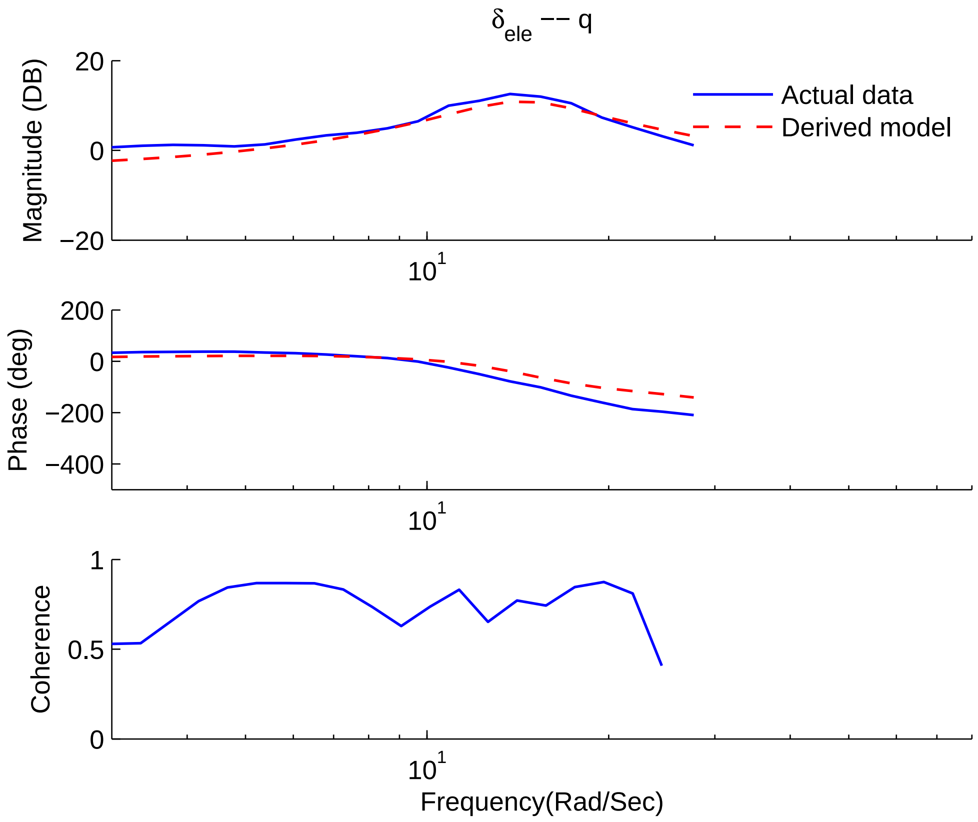


Figure 4.19: Response comparison using frequency-sweep input ($\delta_{\text{ele}} - p$)

relatively weak and can be ignored. The fuselage in all three directions are considered as a flat plate perpendicular to the helicopter motion, thus the drag coefficient is approximately unity. As such, the horizontal fuselage drag forces are formulated in a quadratic form:

$$\begin{aligned} X_{\text{fus}} &= -\frac{\rho}{2} S_x u \cdot \max(v_i, |u|), && (4.22) \\ Y_{\text{fus}} &= -\frac{\rho}{2} S_y v \cdot \max(v_i, |v|), && (4.23) \\ v_i &= \sqrt{\frac{|T_{dw}|}{2\rho \pi R^2}}, && (4.24) \end{aligned}$$

where S_x and S_y are the effective drag area along the body-frame x- and y-axis respectively.

For the vertical direction, since the fuselage is constantly exposed to the lower rotor downwash, it is commonly formulated in the following form:

$$Z_{\text{fus}} = -\frac{\rho}{2} S_z (w - v_i)|w - v_i|. \tag{4.25}$$

However, as the lift coefficient test for identifying $k_{T,i}$ in (4.12) was done with the presence of the fuselage (so the term $\frac{\rho}{2} S_z v_i^2$ has already been taken into account), the above equation can be simplified as:

$$Z_{\text{fus}} = -\frac{\rho}{2} S_z w \cdot \max(v_i, |w|), \tag{4.26}$$

where S_z is the effective drag area along the body-frame z-axis.

In this sub-section, parameters to be identified are ρ, R, S_x, S_y and S_z. All of them can be easily obtained by direct measurement.

Motor Speed Dynamics

Two brushless DC motors are used on the coaxial platform. Their rotational speed dynamics follows the differential equation of electro motors:

$$J_{\text{mot}} \dot{\omega} = \frac{k_{\text{m}} U - k_{\text{m}} k_{\text{e}} \omega}{R_{\text{mot}}} - d\omega - M_{\text{L}}, \tag{4.27}$$

where J_{mot} is the motor moment of inertia, k_{m} and k_{e} are the mechanical and electrical motor constants, U is the input voltage, R_{mot} is the resistance of the circuit, d is the friction coefficient, and M_{L} is the external torque acting on the motor shaft. Here, M_{L} is equal to the rotor drag

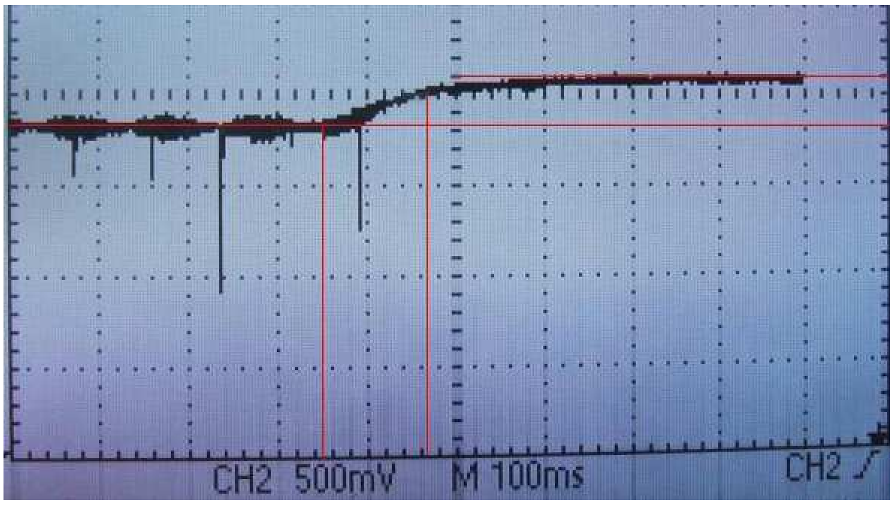


Figure 4.20: Estimation of time constant of motor dynamics

torque $Q_{\mathrm{d},i}$ appeared in (4.13). If the helicopter operates at a near-hover condition, everything can be approximated as a linear process. M_{L} can be assumed to be a combination of a constant trimming value, M_{L}^*, and another term proportional to extra rotational speed as compared to the trimming speed, Ω^*:

$$M_{\mathrm{L}} = M_{\mathrm{L}}^* + k_{\mathrm{L}}(\Omega - \Omega^*). \tag{4.28}$$

Further considering that the rotational speed of rotor, Ω, and the rotational speed of the motor, ω, are perfectly proportional by the gear ratio, the rotor speed dynamics can be simplified to the following first-order equations:

$$\dot{\Omega}_{\mathrm{up}} = \frac{1}{\tau_{\mathrm{mt}}}(m_{\mathrm{up}}\delta_{\mathrm{up}} + \Omega_{\mathrm{up}}^* - \Omega_{\mathrm{up}}), \tag{4.29}$$

$$\dot{\Omega}_{\mathrm{dw}} = \frac{1}{\tau_{\mathrm{mt}}}(m_{\mathrm{dw}}\delta_{\mathrm{dw}} + \Omega_{\mathrm{dw}}^* - \Omega_{\mathrm{dw}}), \tag{4.30}$$

where Ω_{up}^* and Ω_{dw}^* are the trimming values of the rotor rotational speed at hovering, τ_{mt} is the time constant of the motor speed dynamics, and m_{up}, m_{dw} are the steady-state ratio between the change of rotor speeds and the change of motor inputs.

The identification method of τ_{mt} is indirect here. Instead of examining the transient response of the rotor speed with motor step input, which is very difficult to be carried out, the transient response of the input voltage subject to the changes of the motor Back-EMF (voltage generated by the spinning motor) is recorded using an oscilloscope (see Fig. 4.20). The time constant of the the two transient response should be the same. m_{up} and m_{dw} can be identified by plotting

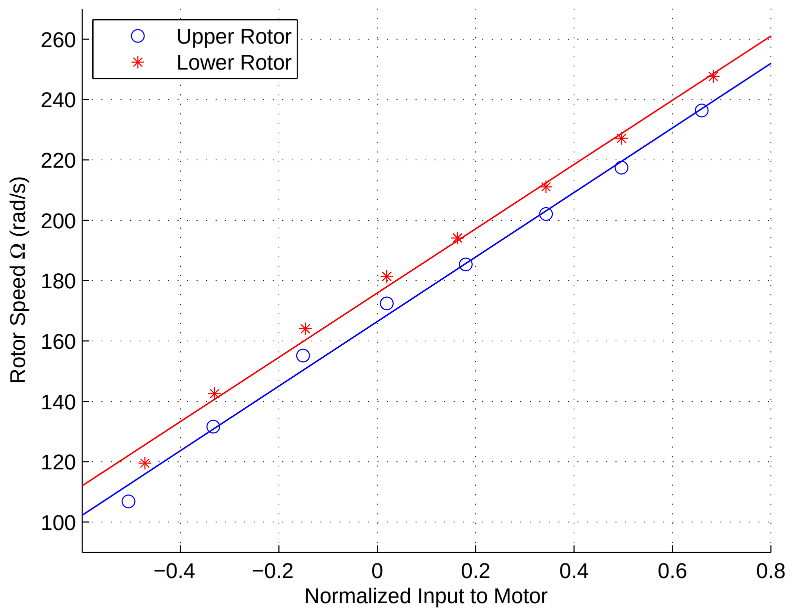


Figure 4.21: Data plot of rotor speed against motor input

the steady-state relationship between the rotor speed and the motor input (see Fig. 4.21). m_{up} and m_{dw} are the gradients of the two fitted lines in the figure.

Mixer and Headlock Gyro Dynamics

In order to decouple the throttle-heave and the rudder-yaw dynamics, the throttle and rudder signals are passed into a hardware mixer and transformed to dual motor control signals:

$$\delta_{up} = \delta_{\mathrm{thr}} + \bar{\delta}_{\mathrm{rud}}, \tag{4.31}$$

$$\delta_{dw} = \delta_{\mathrm{thr}} - \bar{\delta}_{\mathrm{rud}}. \tag{4.32}$$

It can be clearly seen that when the throttle signal δ_{thr} increases, inputs to both motors increase; when the rudder signal $\bar{\delta}_{\mathrm{rud}}$ increases, the input to the motor connected to the upper rotor increases while the input to the motor connected to the lower rotor decreases.

Note that the rudder signal in the above mixer equation is not the original signal δ_{rud}. From δ_{rud} to $\bar{\delta}_{\mathrm{rud}}$, there is a hardware headlock gyro which helps refine the rudder signal and acts as a most inner-loop yaw motion stabilizer. Usually, there is a P-I controller embedded inside the

Table 4.1: Yaw rate against rudder input: hovering turn

r (rad/s)	-1.50	-2.50	-2.60	-3.50
δ_{rud} (-1, 1)	0.25	0.35	0.40	0.55

headlock gyro and it can be formulated as follows:

$$\dot{r}_{\text{fb}} = K_{\text{a}}\delta_{\text{rud}} - r, \tag{4.33}$$

$$\bar{\delta}_{\text{rud}} = K_{\text{P}}(K_{\text{a}}\delta_{\text{rud}} - r) + K_{\text{I}}r_{\text{fb}}, \tag{4.34}$$

where r_{fb} is the augmented state needed by the integral control. K_{a} can be identified by performing manual hovering turn of the helicopter with rudder input at different values. The recorded data is shown in Table 4.1 (steady-state values). The linear gradient of yaw rate against rudder input equals to the value of K_{a}. Next, by placing the helicopter stationary on a test bench, K_{P} and K_{I} can be identified by observing the headlock gyro output signal (in Pulse Width Modulation form) caused by a small known step inputs. The initial ratio between the output and the input is $K_{\text{P}}/K_{\text{a}}$, while the climbing rate of the step response is $K_{\text{I}}/K_{\text{a}}$. At this point, the full dynamics of a coaxial helicopter have been mathematically formulated and all important model parameters have been identified. Table 4.2 has listed all the identified parameters for the coaxial helicopter.

4.3 Model Verification

In this section, a comprehensive evaluation on the fidelity of the obtained nonlinear model is shown. Four manual flight tests were carried out, which include:

1. Aileron channel perturbation with the coaxial helicopter rolling left and right,
2. Elevator channel perturbation with the coaxial helicopter pitching forward and backward,
3. Throttle channel perturbation with the coaxial helicopter flying up and down,
4. Rudder channel perturbation with the coaxial helicopter yawing clockwise and anticlockwise.

In these four flight tests, the human pilot was asked to try his best to agitate only one of the four input channels. However, to make sure the helicopter position does not drift too much

Table 4.2: Identified model parameters for the coaxial UAV

Parameters	Physical meaning
$m = 0.977\,\mathrm{kg}$	Total mass of platform
$g = 9.781\,\mathrm{ms}^{-2}$	Earth gravitational constant
$J_{xx} = 0.0059\,\mathrm{kgm}^2$	Rolling moment of inertia
$J_{yy} = 0.0187\,\mathrm{kgm}^2$	Pitching moment of inertia
$J_{zz} = 0.0030\,\mathrm{kgm}^2$	Yawing moment of inertia
$J_{\mathrm{up}} = 6.8613 \cdot 10^{-4}\,\mathrm{kgm}^2$	Upper rotor moment of inertia
$J_{\mathrm{dw}} = 3.2906 \cdot 10^{-4}\,\mathrm{kgm}^2$	Lower rotor moment of inertia
$\lvert\vec{l}_{\mathrm{up}}\rvert = 0.195\,\mathrm{m}$	Length from upper rotor hub to CG
$\lvert\vec{l}_{\mathrm{dw}}\rvert = 0.120\,\mathrm{m}$	Length from lower rotor hub to CG
$\rho = 1.204\,\mathrm{kg\,m}^{-3}$	Density of air at 1 atmosphere and 20° C
$R = 0.250\,\mathrm{m}$	Rotor radius
$S_{fx} = 0.00835\,\mathrm{m}^2$	Fuselage equivalent area in x-axis
$S_{fy} = 0.01310\,\mathrm{m}^2$	Fuselage equivalent area in y-axis
$S_{fz} = 0.01700\,\mathrm{m}^2$	Fuselage equivalent area in z-axis
$k_{T,\mathrm{up}} = 1.23 \times 10^{-4}\,\mathrm{Ns}^2$	Thrust coefficient of the upper rotor
$k_{T,\mathrm{dw}} = 8.50 \times 10^{-5}\,\mathrm{Ns}^2$	Thrust coefficient of the lower rotor
$k_{Q,\mathrm{up}} = 4.23 \times 10^{-6}\,\mathrm{Nms}^2$	Torque coefficient of the upper rotor
$k_{Q,\mathrm{dw}} = 3.68 \times 10^{-6}\,\mathrm{Nms}^2$	Torque coefficient of the lower rotor
$m_{\mathrm{up}} = 106.9002$	Upper rotor speed to input ratio
$m_{\mathrm{dw}} = 106.4461$	Lower rotor speed to input ratio
$\Omega^*_{\mathrm{up}} = 203.3769\,\mathrm{rad/s}$	Upper rotor trimming rotational speed
$\Omega^*_{\mathrm{dw}} = 217.8807\,\mathrm{rad/s}$	Lower rotor trimming rotational speed
$\tau_{\mathrm{sb}} = 0.2\,\mathrm{s}$	Time constant of upper rotor flapping
$\tau_{\mathrm{mt}} = 0.12\,\mathrm{s}$	Time constant of motor dynamics
$A_q = 0.0204$	Longitudinal rotor damping constant
$B_p = 0.0204$	Lateral rotor damping constant
$A_{a,\mathrm{up}} = 0.49$	Upper rotor on-axis longitudinal flapping ratio
$A_{b,\mathrm{up}} = -0.2745$	Upper rotor off-axis longitudinal flapping ratio
$B_{a,\mathrm{up}} = 0.2745$	Upper rotor off-axis lateral flapping ratio
$B_{b,\mathrm{up}} = 0.49$	Upper rotor on-axis lateral flapping ratio
$A_{a,\mathrm{up}} = 0.1217$	Lower rotor on-axis longitudinal flapping ratio
$A_{b,\mathrm{up}} = -0.045$	Lower rotor off-axis longitudinal flapping ratio
$B_{a,\mathrm{up}} = -0.045$	Lower rotor off-axis lateral flapping ratio
$B_{b,\mathrm{up}} = -0.1217$	Lower rotor on-axis lateral flapping ratio
$K_\beta = 4.377$	Spring constant of the rotor TPP
$K_{\mathrm{a}} = 5.3819$	Scaling factor of the headlock gyro
$K_{\mathrm{P}} = 0.1239$	Proportional gain of the headlock gyro
$K_{\mathrm{I}} = 0.1325$	Integral gain of the headlock gyro

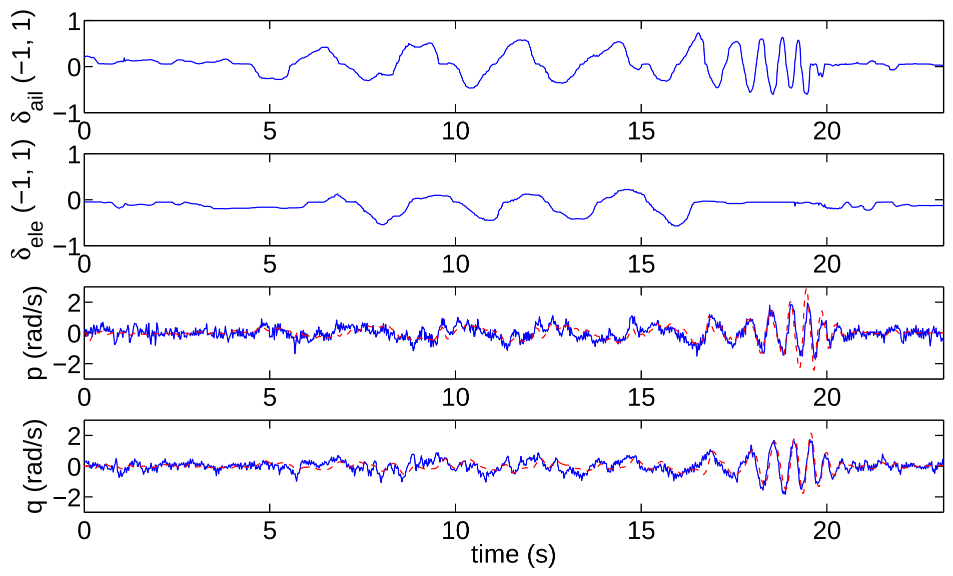


Figure 4.22: Responses from aileron input perturbation

(safety needs to be ensured), minor off-axis inputs were also issued to lightly counter the cross-couplings between the channels. The time-domain results are shown in Figs. 4.22–4.25. Based on the recorded inputs, the transient response of the UAV attitudes, angular rates and body-frame velocities are calculated by a MATLAB simulation program with the aforementioned nonlinear mathematical model (dashed lines in the figures). They are plotted together with the in-flight true data obtained by the onboard sensors (solid lines in the figures). The matching between the two is quite good. Note that for angular rate dynamics, both the on-axis response and off-axis response matches well. Some minor mismatches are caused by the ignorance of high frequency dynamics when formulating the model, especially for the motion of rotor flapping, which is highly complicated. Other discrepancies are believed to be from wind disturbances, ground effects and measurement noises present in actual flight tests. In general, this is an accurate nonlinear cross-coupled model for a fixed-pitch coaxial UAV with low maneuvering speed.

4.4 Control Structure Formulation

With the dynamic model of the coaxial UAV obtained, advanced control design methods can be applied to stabilize the motion of this UAV. Methods of controlling coaxial helicopters have been reviewed thoroughly at the starting phase of this research work [2, 26, 40, 51, 52, 68, 80, 82, 86]. With reference to these methods, the control strategy presented here is based on a dual-loop structure. The inner-layer dynamics of the UAV can be stabilized by an H-infinity control law

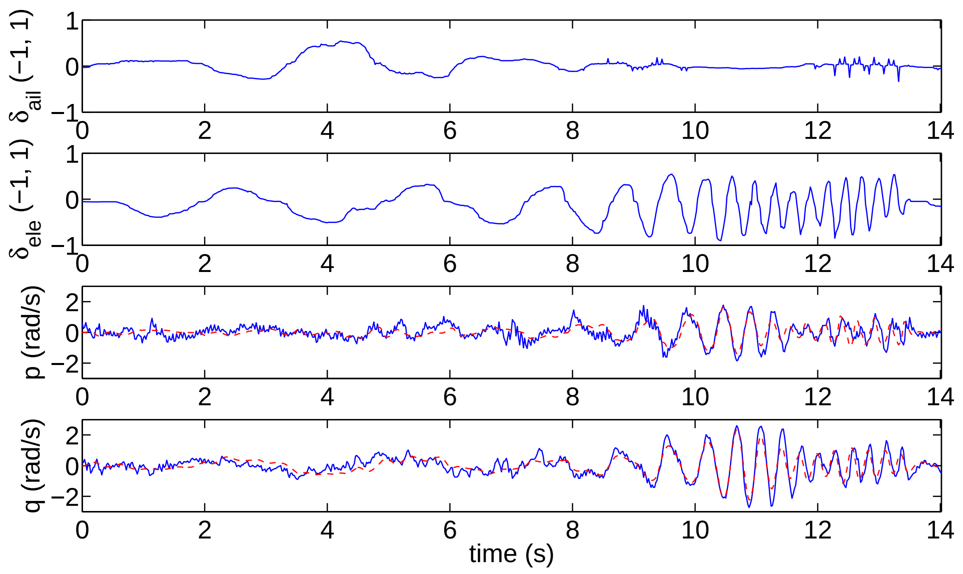


Figure 4.23: Responses from elevator input perturbation

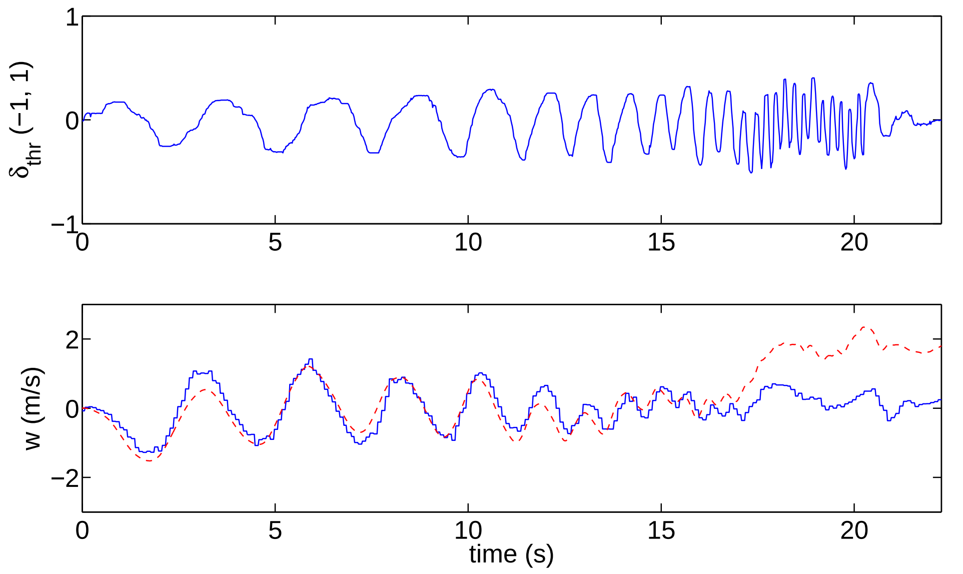


Figure 4.24: Responses from throttle input perturbation

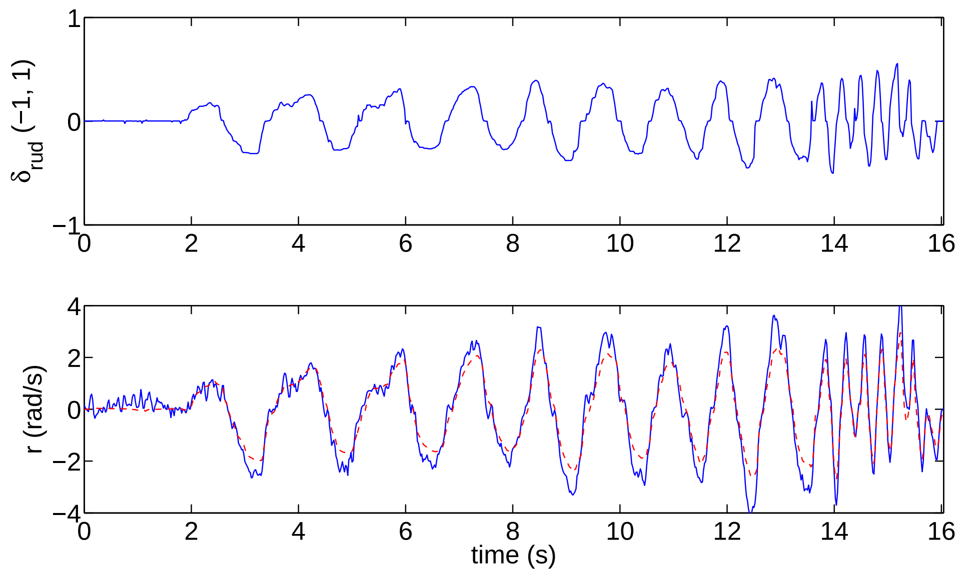


Figure 4.25: Responses from rudder input perturbation

which makes the UAV attitude stable, while a Robust and Perfect Tracking (RPT) controller is designed for the outer loop for position and velocity reference tracking. In addition, the so-called asymptotic time-scale and eigenstructure assignment (ATEA) approach has been adopted to tune the inner- and outer-loop control laws. It makes the whole design process systematic and effective.

In control engineering, the *divide-and-conquer* strategy is usually used when a relatively complex system needs to be handled. In flight control engineering, a natural stratification of the full-order dynamic model of a helicopter is based on motion types, i.e. rotational motion and translational motion. In general, the dynamics of rotational motion is much faster than that of the translational motion. Thus, the controlled object can be divided into two parts and the overall control system can be formulated in a dual-loop structure. In this way, inner-loop and outer-loop controllers can be designed separately. Moreover, it is found that the linearized model of the coaxial helicopter system is of non-minimum phase if the two motion dynamics are combined together. If not separated, they will highly complicate the control problem and degrade control performance.

For the inner loop, the controlled object covers the rotational motion of helicopter body, flapping motion of rotor blades and stabilizer bar, rotational motion of the motor-rotor driving system, as well as dynamics embedded within the head-lock gyro. The main task of the inner-loop controller is to stabilize the attitude and heading of the helicopter in all flight conditions.

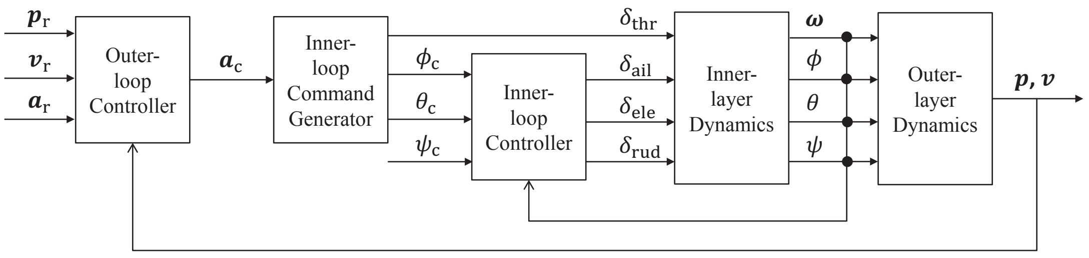


Figure 4.26: Dual-loop structure of the flight control system

H$_\infty$ technique is preferred for robust stability. For the outer loop, the controlled object covers only the translational motion. The main task is to steer the helicopter flying with reference to a series of given locations. A robust and perfect tracking (RPT) approach is implemented for the outer-loop since time factor is important. It is highlighted that both control laws are designed using the ATEA method, which is fully developed for MIMO LTI (multi-Input multi-Output linear time invariant) systems by Chen et al. [17]. It makes the design process very systematic and effective. To give an overall view, the dual-loop control structure is shown in Fig. 5.10.

4.5 Inner-loop Control Law Design

The inner-layer dynamics of the controlled coaxial helicopter is a 11th-order MIMO system with four control inputs, namely δ_thr, δ_ail, δ_ele, and δ_rud. To stabilize the attitude and heading angles ϕ, θ, ψ, only three of the four control inputs δ_ail, δ_ele, and δ_rud, are needed. The remaining one, δ_thr, is reserved for control of vertical motion and meanwhile needs be set at its trimming value. The system is then a 11th-order 3-input 3-output controlled object. For the measurement part, the IMU gives ϕ, θ, ψ, p, q, and r. The remaining state variables, i.e. the upper rotor flapping angles b_up, a_up, the two motor rotational speeds Ω_up, Ω_dw, and the intermediate state variable $\bar{\delta}_\mathrm{rud}$ of the head-lock gyro, have to be estimated by an observer. Therefore, the linearized inner-layer controlled object can be formulated as

$$\begin{cases} \dot{\boldsymbol{x}} = \mathbf{A}\boldsymbol{x} + \mathbf{B}\boldsymbol{u} + \mathbf{E}\boldsymbol{w} \\ \boldsymbol{y} = \mathbf{C}_1\boldsymbol{x} + \mathbf{D}_{11}\boldsymbol{u} + \mathbf{D}_1\boldsymbol{w} \\ \boldsymbol{z} = \mathbf{C}_2\boldsymbol{x} + \mathbf{D}_2\boldsymbol{u} + \mathbf{D}_{22}\boldsymbol{w} \end{cases} \tag{4.35}$$

with

$$
\begin{aligned}
\boldsymbol{x} &= (\phi \quad \theta \quad \psi \quad p \quad q \quad r \quad b_{\text{up}} \quad a_{\text{up}} \quad \Omega_{\text{up}} \quad \Omega_{\text{dw}} \quad r_{\text{fb}})^{\mathrm{T}}, \\
\boldsymbol{y} &= (\phi \quad \theta \quad \psi \quad p \quad q \quad r)^{\mathrm{T}}, \\
\boldsymbol{z} &= (\phi \quad \theta \quad \psi)^{\mathrm{T}}, \\
\boldsymbol{u} &= (\delta_{\text{ail}} \quad \delta_{\text{ele}} \quad \delta_{\text{rud}})^{\mathrm{T}},
\end{aligned}
$$

and

$$
\mathbf{A} = \begin{bmatrix}
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -35.62 & 0 & 0 & 449.95 & 252.06 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -11.24 & 0 & -79.53 & 141.96 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -31.25 & 0 & 0 & -1.32 & 0.367 & 33.39 \\
0 & 0 & 0 & -1 & 0 & 0 & -5 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 0 & 0 & -5 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -92.47 & 0 & 0 & -8.33 & 0 & 98.79 \\
0 & 0 & 0 & 0 & 0 & 92.08 & 0 & 0 & 0 & -8.33 & -98.37 \\
0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0
\end{bmatrix},
$$

$$
\mathbf{B} = \begin{bmatrix}
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0 \\
-100.73 & -37.25 & 0 \\
-11.75 & 31.78 & 0 \\
0 & 0 & 200.82 \\
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 594.27 \\
0 & 0 & -591.74 \\
0 & 0 & 6.43
\end{bmatrix}.
$$

where $\boldsymbol{y}$ is the measured output vector, $\boldsymbol{z}$ is the controlled output vector, and all variables are the deviations from their trimming values. Note that the direct feedthrough matrices $\mathbf{D}_{11}$ and $\mathbf{D}_2$ are both zero. No external disturbance is considered for this part of model at the current stage, so the disturbance input matrix $\mathbf{E}$ and the feedthrough matrices $\mathbf{D}_1$, $\mathbf{D}_{22}$ are all empty. They are reserved in the expression for integrity so that external disturbances such as wind gusts can be considered in future. The controlled subsystem characterized by the quadruple $(\mathbf{A}, \mathbf{B}, \mathbf{C}_2, \mathbf{D}_2)$ is both observable and controllable. By transforming the quadruple into the special coordinate basis (SCB) form [17], we find that the subsystem is invertible and of minimum phase with 5 stable invariant zeros and 3 infinite zeros of order 2. Hence, we can design an H_∞ controller via the ATEA method using state feedback to obtain robust stability. After that, an observer-based controller can be designed utilizing measurement feedback also via the same method. Usually, the control law designed via the ATEA method is parameterized by a number $\gamma > \gamma^*$, where γ^* is the infimum of the H_∞-norm of the closed-loop transfer matrix from disturbance ***w*** to the controlled output ***z***. It is found that the design result does not depend on the number γ because the controlled subsystem $(\mathbf{A}, \mathbf{B}, \mathbf{C}_2, \mathbf{D}_2)$ is right invertible and of minimum phase, and the measurable subsystem $(\mathbf{A}, \mathbf{E}, \mathbf{C}_1, \mathbf{D}_1)$ is left invertible and of minimum phase. This simplifies the design process significantly.

H_∞ State Feedback Control Design

It can be realized step-by-step via the ATEA method as follows. First, transform the matrix quadruple $(\mathbf{A}, \mathbf{B}, \mathbf{C}_2, \mathbf{D}_2)$ of the controlled subsystem into its SCB form by state, output, input transformations $\mathbf{T}_{\mathrm{s}1}$, $\mathbf{T}_{\mathrm{o}1}$, $\mathbf{T}_{\mathrm{i}1}$, and obtain

$$\tilde{\mathbf{A}} = \begin{bmatrix} \mathbf{A}_{\mathrm{aa}}^- & \mathbf{L}_{\mathrm{ad}}^-\mathbf{C}_{\mathrm{d}} \\ \mathbf{B}_{\mathrm{d}}\mathbf{E}_{\mathrm{da}}^- & \mathbf{A}_{\mathrm{dd}} \end{bmatrix}, \quad \tilde{\mathbf{B}} = \begin{bmatrix} \mathbf{0} \\ \mathbf{B}_{\mathrm{d}} \end{bmatrix}, \quad \tilde{\mathbf{C}} = \begin{bmatrix} \mathbf{0} & \mathbf{C}_{\mathrm{d}} \end{bmatrix}, \tag{4.36}$$

where

$$\mathbf{A}_{\mathrm{dd}} = \mathbf{A}_{\mathrm{dd}}^* + \mathbf{B}_{\mathrm{d}}\mathbf{E}_{\mathrm{dd}} + \mathbf{L}_{\mathrm{dd}}\mathbf{C}_{\mathrm{d}}.$$

Because the subsystem is invertible and of minimum phase, only the infinite eigenstructure needs to be assigned explicitly. Second, the subsystem related to the infinite zero structure has 3 infinite zeros of order 3, i.e. there are 3 chains of integrators from the 3 control inputs to the 3 controlled outputs and each chain is composed of 3 integrators in series. The desired characteristic functions of the 3 subspaces can be set as

$$p_1(s) = s^2 + a_{11}s + a_{12}, \tag{4.37}$$

$$p_2(s) = s^2 + a_{21}s + a_{22}, \tag{4.38}$$

$$p_3(s) = s^2 + a_{31}s + a_{32}. \tag{4.39}$$

For the case of the controlled coaxial helicopter, their eigenvalues are assigned respectively at $(-49,\ -3.8)$, $(-11.8,\ -4.9)$ and $(-37.8,\ -4.9)$, which correspond to roll, pitch, yaw control channels respectively. Thus, the time-scale assignment is as follows:

$$\mathbf{F}_{\mathrm{d}} = \begin{bmatrix} \frac{a_{12}}{\varepsilon^2} & \frac{a_{11}}{\varepsilon} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{a_{22}}{\varepsilon^2} & \frac{a_{21}}{\varepsilon} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{a_{32}}{\varepsilon^2} & \frac{a_{31}}{\varepsilon} \end{bmatrix}. \tag{4.40}$$

Finally, the composite state feedback gain by structural assignment is given as

$$\mathbf{F}_{\mathrm{s}} = -\mathbf{T}_{\mathrm{i1}} \left([0 \quad \mathbf{F}_{\mathrm{d}}] + [\mathbf{E}_{\mathrm{da}}^{-} \quad \mathbf{E}_{\mathrm{dd}}] \right) \mathbf{T}_{\mathrm{s1}}^{-1}. \tag{4.41}$$

$\mathbf{H}_\infty$ Measurement Output Feedback Control Design

The measurement feedback control design can be realized step-by-step via the ATEA method based on the above designed result of state feedback. First, define the auxiliary full state feedback system as follows,

$$\begin{cases} \dot{\boldsymbol{x}} = \mathbf{A}^{\mathrm{T}}\boldsymbol{x} + \mathbf{C}_1^{\mathrm{T}}\boldsymbol{u} + \mathbf{C}_2^{\mathrm{T}}\boldsymbol{w} \\ \boldsymbol{y} = \boldsymbol{x} \\ \boldsymbol{z} = \mathbf{E}^{\mathrm{T}}\boldsymbol{x} + \mathbf{D}_1^{\mathrm{T}}\boldsymbol{u} + \mathbf{D}_{22}^{\mathrm{T}}\boldsymbol{w} \end{cases} \tag{4.42}$$

Its controlled subsystem $(\mathbf{A}^{\mathrm{T}},\ \mathbf{C}_1^{\mathrm{T}},\ \mathbf{E}^{\mathrm{T}},\ \mathbf{D}_1^{\mathrm{T}})$ is just the dual of the measurable subsystem $(\mathbf{A},\ \mathbf{E},\ \mathbf{C}_1,\ \mathbf{D}_1)$ and can be decomposed into the SCB form by state, input, output transformations $\mathbf{T}_{\mathrm{s2}}$, $\mathbf{T}_{\mathrm{i2}}$, $\mathbf{T}_{\mathrm{o2}}$, and obtain

$$\tilde{\mathbf{A}} = \begin{bmatrix} \mathbf{A}_{\mathrm{aa}}^{-} & \mathbf{0} \\ \mathbf{B}_{\mathrm{c}}\mathbf{E}_{\mathrm{ca}}^{-} & \mathbf{A}_{\mathrm{cc}} \end{bmatrix}, \quad \tilde{\mathbf{B}} = \begin{bmatrix} \mathbf{0} \\ \mathbf{B}_{\mathrm{c}} \end{bmatrix}, \quad \tilde{\mathbf{C}} = \mathbf{0}. \tag{4.43}$$

There is only one stable invariant zero, so this dual measurable subsystem is right invertible and of minimum phase. We can then design a state feedback control law via the ATEA method. Different from the aforementioned state feedback design, the subsystem now is not left invertible and has no infinite zeros, so more assignment work has to be done. Second, let $\mathbf{K}_{\mathrm{c}}$ be an arbitrary matrix of dimensions compatible with the matrix pair $(\mathbf{A}_{\mathrm{cc}},\ \mathbf{B}_{\mathrm{c}})$ subject to the constraint that

$$\mathbf{A}_{\mathrm{cc}}^{\mathrm{c}} = \mathbf{A}_{\mathrm{cc}} - \mathbf{B}_{\mathrm{c}}\mathbf{K}_{\mathrm{c}} \tag{4.44}$$

is stable. This can be done because $(\mathbf{A}_{\mathrm{cc}}, \mathbf{B}_{\mathrm{c}})$ is controllable due to the property of SCB decomposition. The eigenvalues are placed at $(-6, -9, -15, -26; -6, -9, -15, -26; -2, -4, -6, -15)$, considering the stability and bandwidth of the observer being designed. The eigenvalues form into three groups: the first four and the second four are exactly the same which are assigned to the roll and pitch control respectively, the last four is assigned to yaw control. Third, the composite state feedback gain for the dual measurable subsystem is assigned as

$$\mathbf{K}_{\mathrm{m}} = -\mathbf{T}_{\mathrm{i2}} \left(\begin{bmatrix} \mathbf{0} & \mathbf{K}_{\mathrm{c}} \end{bmatrix} + \begin{bmatrix} \mathbf{E}_{\mathrm{ca}}^{-} & \mathbf{0} \end{bmatrix} \right) \mathbf{T}_{\mathrm{s2}}^{-1} \tag{4.45}$$

Finally, let

$$\mathbf{F}_{\mathrm{m}} = \mathbf{K}_{\mathrm{m}}^{\mathrm{T}} \tag{4.46}$$

We can construct the following full-order observer-based control law for the inner-loop

$$\begin{cases} \dot{\hat{\mathbf{x}}} = \mathbf{A}_{\mathrm{ci}}\hat{\mathbf{x}} + \mathbf{B}_{\mathrm{ci}}\mathbf{y} \\ \mathbf{u} = \mathbf{C}_{\mathrm{ci}}\hat{\mathbf{x}} \end{cases} \tag{4.47}$$

with

$$\begin{aligned} \mathbf{A}_{\mathrm{ci}} &= \mathbf{A} + \mathbf{B}\mathbf{F}_{\mathrm{s}} + \mathbf{F}_{\mathrm{m}}\mathbf{C}_1 \\ \mathbf{B}_{\mathrm{ci}} &= -\mathbf{F}_{\mathrm{m}} \\ \mathbf{C}_{\mathrm{ci}} &= \mathbf{F}_{\mathrm{s}} \end{aligned}$$

Therefore, the following closed-loop system is obtained:

$$\begin{cases} \begin{pmatrix} \dot{\boldsymbol{x}} \\ \dot{\hat{\boldsymbol{x}}} \end{pmatrix} = \begin{bmatrix} \mathbf{A} & \mathbf{B}\mathbf{C}_{\mathrm{ci}} \\ \mathbf{B}_{\mathrm{ci}}\mathbf{C}_1 & \mathbf{A}_{\mathrm{ci}} \end{bmatrix} \begin{pmatrix} \boldsymbol{x} \\ \hat{\boldsymbol{x}} \end{pmatrix} + \begin{bmatrix} \mathbf{B} \\ \mathbf{B} \end{bmatrix} \mathbf{G}_{\mathrm{i}}\boldsymbol{r} \\ \boldsymbol{z} = \begin{bmatrix} \mathbf{C}_2 & \mathbf{D}_2\mathbf{C}_{\mathrm{ci}} \end{bmatrix} \begin{pmatrix} \boldsymbol{x} \\ \hat{\boldsymbol{x}} \end{pmatrix} + \mathbf{D}_2\mathbf{G}_{\mathrm{i}}\boldsymbol{r} \end{cases} \tag{4.48}$$

where $\boldsymbol{r}$ constitutes the reference signals $(\phi_{\mathrm{r}}, \theta_{\mathrm{r}}, \psi_{\mathrm{r}})$ for roll, pitch, yaw tracking, and $\mathbf{G}_{\mathrm{i}}$ is the feedforward gain matrix to ensure a unitary gain of the closed-loop system. To facilitate the understanding of the control system architecture, the closed-loop system is shown in Fig. 4.27.

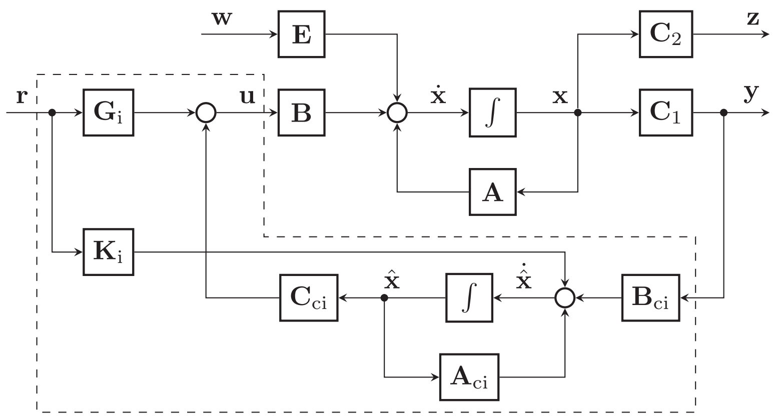


Figure 4.27: H_∞ design for attitude and heading control via ATEA method

In the figure, $\mathbf{K}_\mathrm{i} = \mathbf{B}\mathbf{G}_\mathrm{i}$. By this design, the closed-loop transfer matrix with state feedback has been recovered via output feedback.

4.6 Outer-loop Control Law Design

Hovering in the corner of two walls or flying along a wall are two basic tasks for indoor flight. This section tries to design the hovering and wall-following outer-loop controller for the indoor miniature coaxial helicopter. We have partitioned the whole dynamic model of our aircraft into two parts and have finished the inner-loop design which stabilizes attitude and heading. The outer-loop control can then be designed separately and based on the dynamic model of aircraft's translational motion only. Furthermore, the translational model can be divided into two parts: trajectory kinematics and trajectory dynamics. The kinematics part can be described in different coordinate systems, such as the NED coordinate system or the body-carried coordinate system. In our implementation, the positioning sensor is only a laser scanner, so global NED coordinates of aircraft are unknown. We can only obtain the local distance information relative to the walls (see Fig. 4.28). In the figure, the thick line represents the wall to be followed; $o_\mathrm{n}x_\mathrm{n}y_\mathrm{n}z_\mathrm{n}$ stands for an NED coordinate system which is usually fixed at the starting point of flight and approximates the inertial frame of reference; $o_\mathrm{b}x_\mathrm{b}y_\mathrm{b}z_\mathrm{b}$ stands for the body-axis coordinate system as the conventional definition in aeronautical engineering; $o_\mathrm{w}x_\mathrm{w}y_\mathrm{w}z_\mathrm{w}$ stands for the so-called local-wall coordinate system which is defined as follows. Its origin o_w is set at the intersection of the wall with the perpendicular line passing through o_b, x_w-axis is along the wall spanning an acute angle with x_b-axis, y_w-axis is along the perpendicular line pointing to o_b. In flight, the position and velocity of o_b in the NED coordinate system are unknown, but

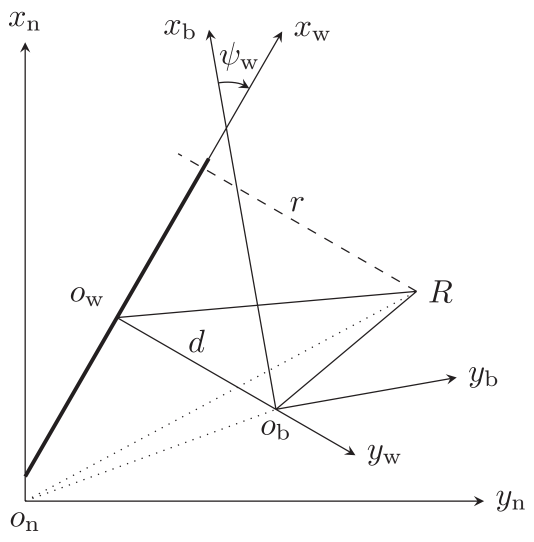


Figure 4.28: Outer-loop reference generation for flight along a wall

the UAV information with respect to the local-wall coordinate system is known. In this case, we have to describe and design the outer navigation loop in the body-axis coordinate system. Let $\boldsymbol{p} = (x, y, z)^\mathrm{T}$, $\boldsymbol{v} = (u, v, w)^\mathrm{T}$, $\boldsymbol{a} = (a_\mathrm{x}, a_\mathrm{y}, a_\mathrm{z})^\mathrm{T}$ be the position, velocity, acceleration of the helicopter with respect to the local-wall coordinate system. With reference to the kinematics of moving reference of frames, the trajectory kinematics of the helicopter can be described in the body-axis coordinate system as follows,

$$\dot{\boldsymbol{p}}_\mathrm{b} = \dot{\boldsymbol{p}}_\mathrm{n} - \boldsymbol{\omega} \times \boldsymbol{p} = \boldsymbol{v} - \boldsymbol{\omega} \times \boldsymbol{p} \tag{4.49}$$

$$\dot{\boldsymbol{v}}_\mathrm{b} = \dot{\boldsymbol{v}}_\mathrm{n} - \boldsymbol{\omega} \times \boldsymbol{v} = \boldsymbol{a} - \boldsymbol{\omega} \times \boldsymbol{v} \tag{4.50}$$

where the subscripts $_\mathrm{b}$ and $_\mathrm{n}$ means that the differentiation is performed in the body-axis coordinate system and in the NED coordinate system respectively, $\boldsymbol{\omega} = (p, q, r)^\mathrm{T}$ is the angular velocity of aircraft body described in the body-axis coordinate system. The above equations can be linearized as

$$\delta\dot{\boldsymbol{p}}_\mathrm{b} = \delta\boldsymbol{v} - [\boldsymbol{\omega}_0]_\times \delta\boldsymbol{p} + [\boldsymbol{p}_0]_\times \delta\boldsymbol{\omega} \tag{4.51}$$

$$\delta\dot{\boldsymbol{v}}_\mathrm{b} = \delta\boldsymbol{a} - [\boldsymbol{\omega}_0]_\times \delta\boldsymbol{v} + [\boldsymbol{v}_0]_\times \delta\boldsymbol{\omega} \tag{4.52}$$

where $\boldsymbol{p}_0$, $\boldsymbol{v}_0$, $\boldsymbol{\omega}_0$ are respectively the position, velocity, angular velocity at the operating points for linearization, $[\cdot]_\times$ is the anti-skew matrix spanned by the vector inside the brackets. Hereafter, for the economy of notations, the the prefix δ representing variable deviation and the subscript $_\mathrm{b}$ will be omitted if without ambiguity.

Notice that $\boldsymbol{a} = \boldsymbol{a}_\mathrm{b} - \boldsymbol{a}_\mathrm{w}$, where $\boldsymbol{a}_\mathrm{b}$ and $\boldsymbol{a}_\mathrm{w}$ are the acceleration of aircraft body and the wall coordinate system with respect to the NED coordinate system. The acceleration $\boldsymbol{a}_\mathrm{b}$ is one of the outputs from the inner-layer helicopter model whereas $\boldsymbol{a}_\mathrm{w}$ is unknown. Besides, we should know that instead of angular velocity $\boldsymbol{\omega}$, acceleration $\boldsymbol{a}_\mathrm{b}$ is the dominant input to the trajectory kinematics. Thus, let $\boldsymbol{p}$ and $\boldsymbol{v}$ be the states of the trajectory kinematics, $\boldsymbol{a}_\mathrm{b}$ be the control input, the following equation can be formed:

$$\begin{pmatrix} \dot{\boldsymbol{p}} \\ \dot{\boldsymbol{v}} \end{pmatrix} = \begin{bmatrix} -\left[\boldsymbol{\omega}^0\right]_\times & \mathbf{I} \\ \mathbf{0} & -\left[\boldsymbol{\omega}^0\right]_\times \end{bmatrix} \begin{pmatrix} \boldsymbol{p} \\ \boldsymbol{v} \end{pmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{I} \end{bmatrix} \boldsymbol{a}_\mathrm{b} + \begin{bmatrix} \left[\boldsymbol{p}^0\right]_\times & \boldsymbol{0} \\ \left[\mathbf{v}^0\right]_\times & -\mathbf{I} \end{bmatrix} \begin{pmatrix} \boldsymbol{\omega} \\ \boldsymbol{a}_\mathrm{w} \end{pmatrix} \tag{4.53}$$

where $\boldsymbol{\omega}$ and $\boldsymbol{a}_\mathrm{w}$ are the external disturbance. It can be represented compactly as

$$\dot{\boldsymbol{x}}_\mathrm{o} = \mathbf{A}_\mathrm{o}\boldsymbol{x}_\mathrm{o} + \mathbf{B}_\mathrm{o}\boldsymbol{u}_\mathrm{o} + \mathbf{E}_\mathrm{o}\boldsymbol{w}_\mathrm{o} \tag{4.54}$$

and is called the *outer-layer dynamics*. For the measurement, only position of the helicopter with respect to the walls can be acquired. By finite difference, the relative velocity can also be estimated. Although finite difference usually enlarges noises, it does work properly by simple filtering. Eventually, the measured and controlled output equations of the outer-layer model can be obtained as:

$$\boldsymbol{y}_\mathrm{o} = \mathbf{C}_1\boldsymbol{x}_\mathrm{o} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{pmatrix} \boldsymbol{p} \\ \boldsymbol{v} \end{pmatrix}, \tag{4.55}$$

$$\boldsymbol{z}_\mathrm{o} = \mathbf{C}_2\boldsymbol{x}_\mathrm{o} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \end{bmatrix} \begin{pmatrix} \boldsymbol{p} \\ \boldsymbol{v} \end{pmatrix}. \tag{4.56}$$

Equation (4.54), (4.55), (4.56) constitute the controlled object of the outer-loop design. Specifically,

$$\mathbf{A}_\mathrm{o} = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}, \tag{4.57}$$

because $\boldsymbol{\omega}_0$ is zero in hovering and forward flight conditions. Due to the decoupled characteristics of the outer-layer model, i.e. there are three control channels independent of each other. Outer-loop control laws based on the three decoupled SISO models can be designed next.

SISO Design of Outer-loop

The generalized linear model of each control channel is characterized by

$$\begin{cases} \dot{\boldsymbol{x}} = \mathbf{A}\boldsymbol{x} + \mathbf{B}u \\ \boldsymbol{y} = \mathbf{C}_1\boldsymbol{x} \\ z = \mathbf{C}_2\boldsymbol{x} \end{cases} \tag{4.58}$$

with

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad \mathbf{C}_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad \mathbf{C}_2 = \begin{bmatrix} 1 & 0 \end{bmatrix}.$$

To design the tracking system based on the RPT approach, an augmented system need be defined with reference input r and its derivatives $\dot{r}$, $\ddot{r}$, …, $r^{(\kappa-1)}$ as extra state variables, and the κ^{th}-order derivative $r^{(\kappa)}$ as external disturbance. Here, let $\kappa = 3$, so

$$\begin{cases} \begin{pmatrix} \dot{\boldsymbol{x}}_0 \\ \dot{\boldsymbol{x}} \end{pmatrix} = \begin{bmatrix} \mathbf{A}_0 & \mathbf{0} \\ \mathbf{0} & \mathbf{A} \end{bmatrix} \begin{pmatrix} \boldsymbol{x}_0 \\ \boldsymbol{x} \end{pmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{B} \end{bmatrix} u + \begin{bmatrix} \mathbf{E}_0 \\ \mathbf{0} \end{bmatrix} r^{(\kappa)} \\ e = \begin{bmatrix} -\mathbf{C}_0 & \mathbf{C}_2 \end{bmatrix} \begin{pmatrix} \boldsymbol{x}_0 \\ \boldsymbol{x} \end{pmatrix} \end{cases} \tag{4.59}$$

where

$$\mathbf{x}_0 = \begin{pmatrix} r & \dot{r} & \ddot{r} \end{pmatrix}^{\mathrm{T}}$$

and

$$\mathbf{A}_0 = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \quad \mathbf{E}_0 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \quad \mathbf{C}_0 = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$$

In this configuration, r, $\dot{r}$, and $\ddot{r}$ are the references of position, velocity and acceleration respectively. Transform the controlled subsystem from u to e of the augmented system into the SCB form by the state transformation $\mathbf{T}_{\mathrm{s3}}$, then

$$\tilde{\mathbf{A}} = \begin{bmatrix} \mathbf{A}_0 & \mathbf{0} \\ \mathbf{B}_{\mathrm{d}}\mathbf{E}_{\mathrm{da}}^0 & \mathbf{A}_{\mathrm{dd}} \end{bmatrix}, \quad \tilde{\mathrm{B}} = \begin{bmatrix} \mathbf{0} \\ \mathbf{B}_{\mathrm{d}} \end{bmatrix}, \quad \tilde{\mathbf{C}} = \begin{bmatrix} \mathbf{0} & \mathbf{C}_{\mathrm{d}} \end{bmatrix}. \tag{4.60}$$

It can be deduced that the augmented system has an infinite zero of order 2 and has 3 invariant zeros at origin. Next, assign the infinite eigenstructure as follows,

$$p(s) = s^2 + 2\zeta\omega_{\mathrm{n}} s + \omega_{\mathrm{n}}^2, \tag{4.61}$$

where ζ and ω_{n} are the expected damping ratio and natural frequency. This leads to the control law in full state feedback form,

$$\begin{bmatrix} \mathbf{G} & \mathbf{F} \end{bmatrix} = -\begin{bmatrix} \mathbf{E}_{\mathrm{da}}^0 & \mathbf{E}_{\mathrm{dd}} + \mathbf{F}_{\mathrm{d}} \end{bmatrix} \mathbf{T}_{\mathrm{s3}}^{-1}, \tag{4.62}$$

with

$$\mathbf{E}_{\mathrm{dd}} = \mathbf{B}_{\mathrm{d}}^{\mathrm{T}} \mathbf{A}_{\mathrm{dd}}, \qquad \mathbf{F}_{\mathrm{d}} = \begin{bmatrix} \omega_{\mathrm{n}}^2 & 2\zeta\omega_{\mathrm{n}} \end{bmatrix}.$$

In detail, the feedforward gain matrix and the feedback gain matrix are

$$\mathbf{G} = \begin{bmatrix} \omega_{\mathrm{n}}^2 & 2\zeta\omega_{\mathrm{n}} & 1 \end{bmatrix}, \qquad \mathbf{F} = -\begin{bmatrix} \omega_{\mathrm{n}}^2 & 2\zeta\omega_{\mathrm{n}} \end{bmatrix}. \tag{4.63}$$

Inner-loop Command Generator

We have designed the inner-loop and the outer-loop controllers separately to avoid the non-minimum phase problem and to relieve task complexity. To preserve the overall system stability, the closed outer loop should be slower enough than the closed inner loop. In this case, the closed inner loop can be seen as a static gain when combining with the outer loop. In physical meaning, the output of the outer-loop controller is the commanded acceleration described on the body-axis coordinate system, denoted as $\mathbf{a}_{\mathrm{c}}$ in Fig. 5.10. However, the inner-loop controller requires the attitude deflection commands (ϕ_{c}, θ_{c}, ψ_{c}) as control inputs. Obviously, a command conversion is needed. Furthermore, the body-axis acceleration $\boldsymbol{a}_{\mathrm{b}}$ does not interact with heading direction ψ, which is relatively independent of linear motion for helicopters. The throttle control input δ_{thr} is not manipulated by the inner-loop controller since it is not the direct dominator of attitude. It should also be transferred from the outer-loop controller because it dominates heave acceleration. Based on this idea, let $\mathbf{G}_{\mathrm{a}}$ be the steady-state gain matrix from the inner-loop inputs (δ_{thr}, ϕ_{c}, θ_{c}) to the acceleration output $\mathbf{a}_{\mathrm{b}}$. An approximated inner-loop command

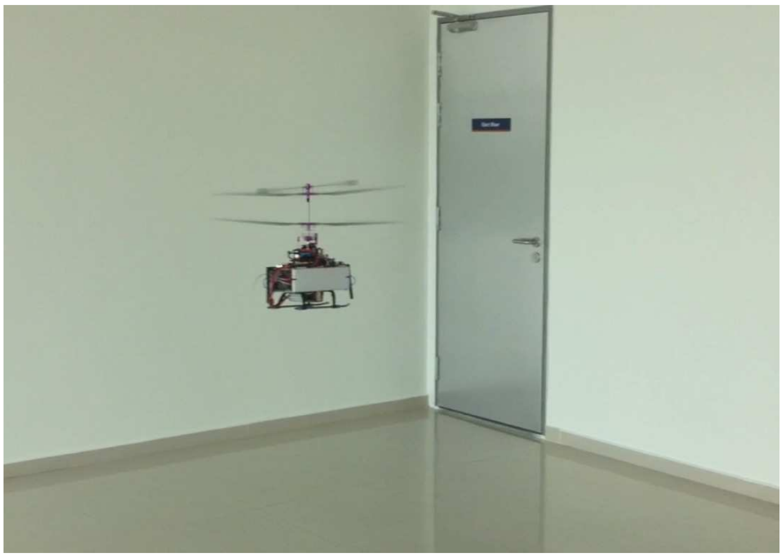


Figure 4.29: The indoor flight test environment

generator from the outer-loop controller output $\boldsymbol{a}_\mathrm{c}$ can be obtained as

$$\begin{pmatrix} \delta_\mathrm{thr} & \phi_\mathrm{c} & \theta_\mathrm{c} \end{pmatrix}^\mathrm{T} = \mathbf{G}_\mathrm{c}\boldsymbol{a}_\mathrm{c} = \mathbf{G}_\mathrm{a}^{-1}\boldsymbol{a}_\mathrm{c}. \tag{4.64}$$

Notice that $\mathbf{G}_\mathrm{a}$ must be non-singular. Otherwise, $\boldsymbol{a}_\mathrm{b}$ cannot be manipulated by the control inputs δ_thr, δ_ail or δ_ele. Flight tests show that this inner-loop command generator is feasible.

4.7 Flight Test Results

Several flight tests have been conducted to validate the proposed control scheme. Two representative scenarios are tried: hovering in the corner of two walls and forward flight along a wall. The testing environment is shown in Fig. 4.29. The reference generation for the whole control system must be consistent with the proposed RPT controller. That is, the position reference $\boldsymbol{p}_\mathrm{r}$, velocity reference $\boldsymbol{v}_\mathrm{r}$ and acceleration reference $\boldsymbol{a}_\mathrm{r}$ should all be calculated and properly assigned.

Outer-loop Reference Generation

In Fig. 4.28, the local-wall coordinate system has been defined to facilitate the implementation of flight test along a wall. Let $\boldsymbol{p} = (x \quad y \quad z)^\mathrm{T}$ be the actual position of helicopter with respect to the local-wall coordinate system. It can be easily deduced to be $(0 \quad d \quad 0)^\mathrm{T}$, where d is the measured perpendicular distance of the helicopter CG away from the wall. The reference point

R at next time step can be set at $\boldsymbol{p}_\mathrm{R} = (v_\mathrm{R}\Delta t \quad r \quad 0)^\mathrm{T}$, where v_R is the expected average speed along the wall during the time interval Δt and r is the distance reference of aircraft CG away from the wall. Hence, the desired position in the body-axis coordinate system can be derived as

$$\boldsymbol{p}_\mathrm{r} = \mathbf{R}_{\mathrm{b/w}}(\boldsymbol{p}_\mathrm{R} - \boldsymbol{p}) \tag{4.65}$$

where $\mathbf{R}_{\mathrm{b/w}}$ is the rotation matrix transforming vectors from the local-wall coordinate system to the body-axis coordinate system,

$$\mathbf{R}_{\mathrm{b/w}} = \begin{bmatrix} \cos\psi_\mathrm{w} & -\sin\psi_\mathrm{w} & 0 \\ \sin\psi_\mathrm{w} & \cos\psi_\mathrm{w} & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.66}$$

where ψ_w is the angle of wall with respect to helicopter heading. Similarly, the desired velocity described in the body-axis coordinate system is

$$\boldsymbol{v}_\mathrm{r} = \mathbf{R}_{\mathrm{b/w}} \begin{pmatrix} v_\mathrm{R} & 0 & 0 \end{pmatrix}^\mathrm{T} \tag{4.67}$$

and the desired acceleration $\boldsymbol{a}_\mathrm{r}$ should be $(0 \quad 0 \quad 0)^\mathrm{T}$ since we assume the helicopter flying at a constant speed. The position $\boldsymbol{p}$ of helicopter is forever $(0 \quad 0 \quad 0)^\mathrm{T}$ in the body-axis coordinate system.

For the hovering flight test in the corner of two walls, we can set

$$\begin{cases} \boldsymbol{p}_\mathrm{R} = (r_\mathrm{x} \quad r_\mathrm{y} \quad 0)^\mathrm{T} \\ \boldsymbol{v}_\mathrm{R} = (0 \quad 0 \quad 0)^\mathrm{T} \\ \boldsymbol{a}_\mathrm{R} = (0 \quad 0 \quad 0)^\mathrm{T} \end{cases} \tag{4.68}$$

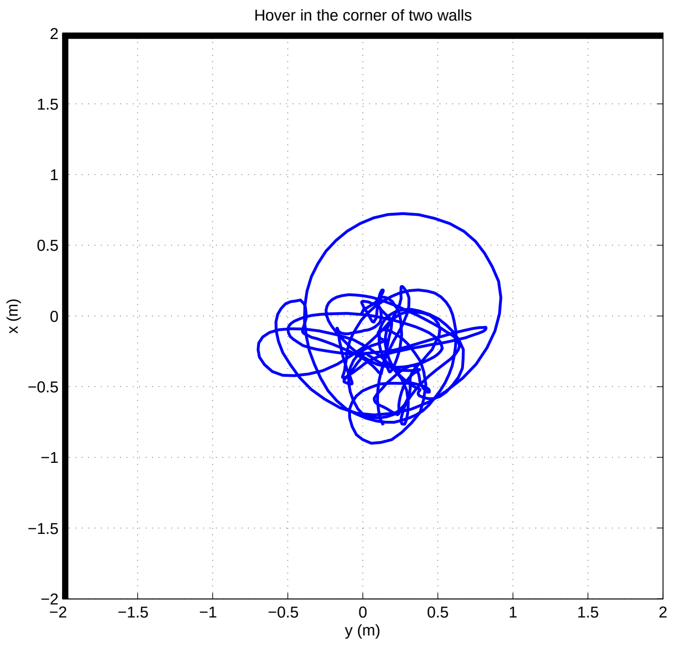
Then, transform them to the body-axis coordinate system

$$\begin{cases} \boldsymbol{p}_\mathrm{r} = \mathbf{R}_{\mathrm{b/w}}(r_\mathrm{x} \quad r_\mathrm{y} \quad 0)^\mathrm{T} \\ \boldsymbol{v}_\mathrm{r} = \mathbf{R}_{\mathrm{b/w}}(0 \quad 0 \quad 0)^\mathrm{T} \\ \boldsymbol{a}_\mathrm{r} = \mathbf{R}_{\mathrm{b/w}}(0 \quad 0 \quad 0)^\mathrm{T} \end{cases} \tag{4.69}$$

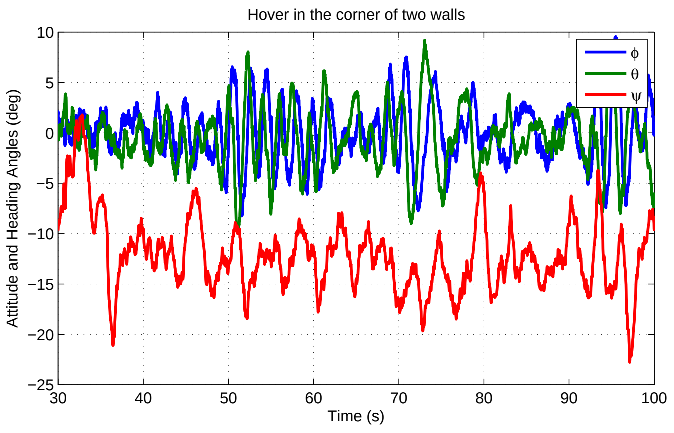
Test Results

The results of the two flight test scenarios are shown in Fig. 4.30 and Fig. 4.31. In Fig. 4.30(a), the x-y trajectory of the helicopter during flight in the corner of two walls is drawn. The two thick lines represent the walls from which the helicopter is commanded to hover 2 meters away. The x-y coordinates are drawn in a way that the helicopter hovers at the center of the figure. One can see that the hovering trajectory is more or less within a 0.5 m circle at the origin. The furthest trajectory is caused by a reflected wind gust from the wall. Fig. 4.30(b) shows the roll, pitch and heading angles of the helicopter. As shown, there is a 90° phase difference between the roll and pitch angles, which indicates a circle-like motion of the helicopter body, which is consistent with the circle-like trajectory. The heading angle tries to stick to a constant value because it was controlled to be parallel with the left wall which is fixed in the NED coordinate system. Fig. 4.30(c) gives the body-axis angular rates. It suggests that, for better performance, hardware noise reduction such as vibration isolation from IMU needs to be done.

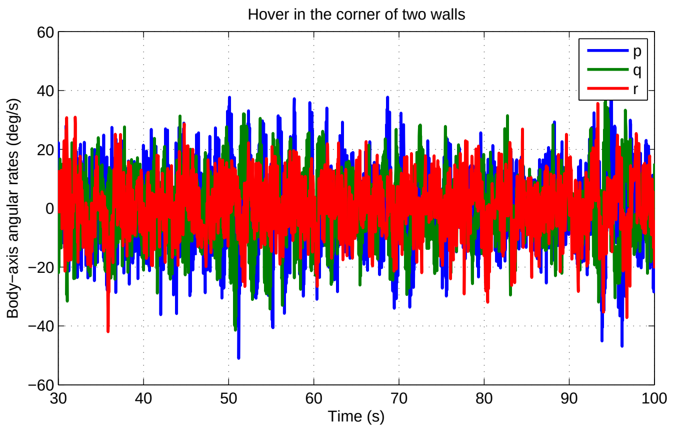
Fig. 4.31(a) shows the time history of the y-axis position of the helicopter during wall-following flight. It is described in the local-wall coordinate system. The helicopter is commanded to fly along its left wall, holding a constant distance of 1.4 meters away from the wall. It can be seen that the tracking accuracy is in the range about ± 0.5 meter. As in this case the laser range finder cannot capture the x-axis position, the x-y trajectory cannot be shown. Again, the attitude angles and the body-axis angular rates are shown in Fig. 4.31(b) and Fig. 4.31(c). They are similar to that of Fig. 4.30(b) and Fig. 4.30(c). Heading angles are different in the two cases because different walls were being referenced.



(a) Trajectory

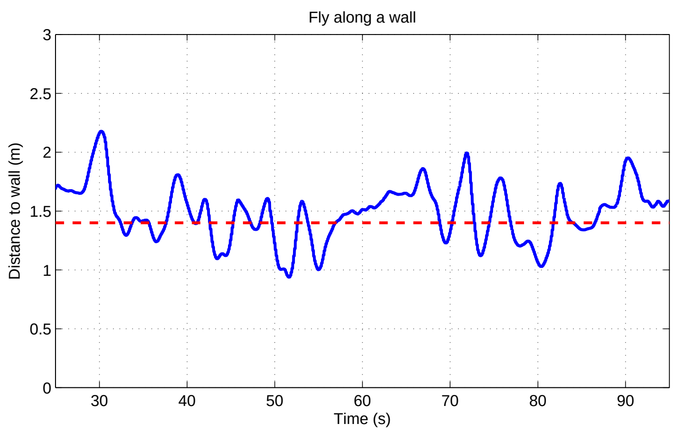


(b) Attitude and Heading

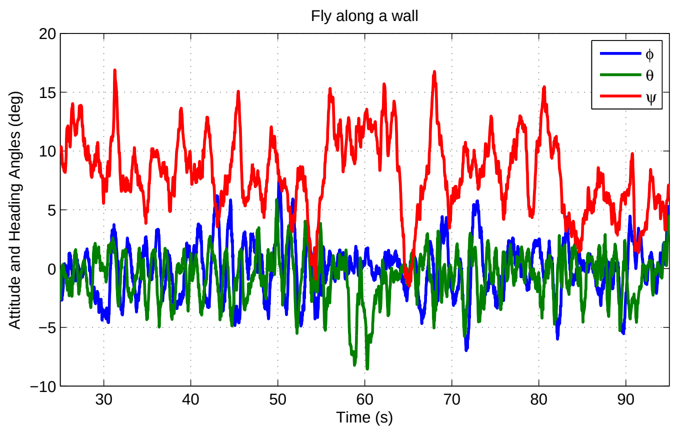


(c) Body-axis angular rates

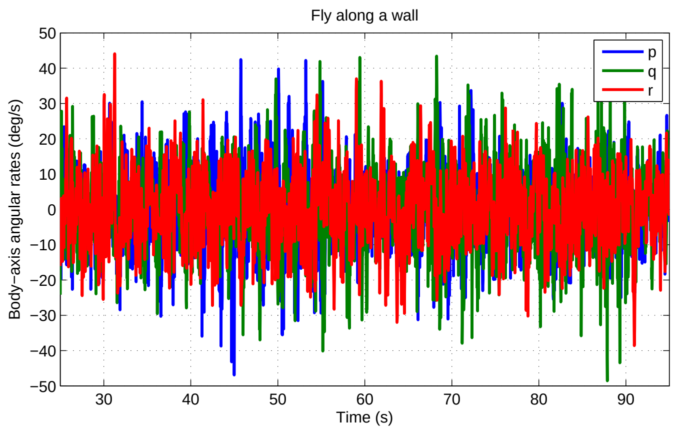
Figure 4.30: Hovering at the corner of two walls



(a) Distance to wall



(b) Attitude and Heading



(c) Body-axis angular rates

Figure 4.31: Flying along a wall

Chapter 5

Modeling and Control of a Quadrotor Helicopter

Being mechanically simple and robust, quadrotor helicopters, or simply called quadrotors, have been widely used as UAV platforms for research purposes these days [61, 75, 65]. This kind of aerial platforms were not popular in the past because they normally need an inner-loop stability controller with more than 100 Hz of control rate, which cannot be handled by microprocessor in the past. However, with the current technology in microprocessors and advanced control theories, the inner-loop stability of a quadrotor helicopter is no longer a problem. There are COTS multi-rotor control boards which can be installed into a quadrotor frame with convenient connections to the motor ESCs, and the resultant quadrotor platforms are usually stabilized well in the attitude level. By utilizing this existing technology, the second indoor UAV platform serving for the purpose of this thesis is a customized quadrotor platform with an inner-loop control board called Naza-M from the company DJI. The following content of this chapter will explain how to model and control the quadrotor helicopter. However, different from that of the coaxial helicopter, the inner-layer dynamics of the quadrotor mentioned here has already been stabilized and does not need to be touched anymore. We only need to know the inner-layer bandwidth and its steady state gain so that a outer-loop control law resulting in an appropriate closed-loop bandwidth can be designed and connected with the inner loop in a reasonable way.

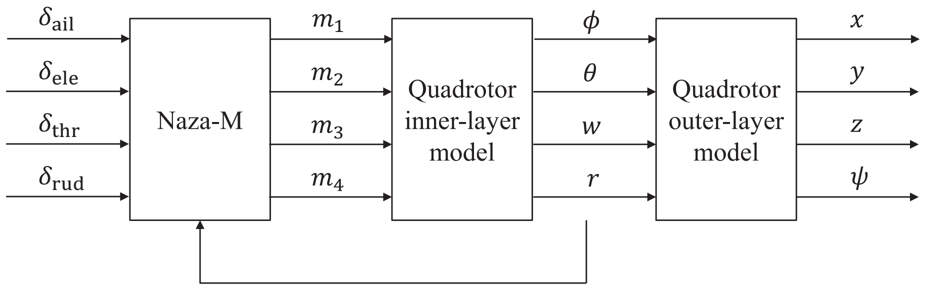


Figure 5.1: Overview of quadrotor model structure

5.1 Basic Working Principle and Model Overview

The model structure of the quadrotor platform constructed for this thesis is illustrated in Fig. 5.1. The normalized control inputs $(\delta_{\text{ail}}, \delta_{\text{ele}}, \delta_{\text{thr}}, \delta_{\text{rud}})$ are fed into the Naza-M controller, which is an all-in-one stability controller specially designed for multi-rotor multi-axis flying platforms. With a standard quadrotor frame construction, the default control gains built in Naza-M can already control the inner-loop stability very well. Naza-M controller outputs PWM signals (m_1, m_2, m_3, m_4) to drive the four rotors to generate the thrust forces, which not only lift the platform but also maintain its attitude stability. From the perspective of Naza-M, the four inputs correspond to the control references for the roll angle ϕ, pitch angle θ, yaw angular rate r, and the UAV body-frame vertical axis velocity w. In the outer-layer dynamics, the quadrotor heading ψ is simply the integration of r, and its vertical axis position z is the integration of w_{g} which is almost the same as w for near-hover flight. For the lateral and longitudinal motion, non-zero (ϕ, θ) angles will induce acceleration in the UAV body-frame x- and y-axis. If transformed to the NED frame, they integrates to the NED velocity (u_{g}, v_{g}) and integrates again to the NED position (x, y). All the notations used here are consistent with those used for the coaxial helicopter.

The platform operates in the so-called ‘X’ mode as shown in Fig. 5.2, where there are two frontal motors and two rear motors. The UAV body frame is defined as x-axis pointing forward, y-axis pointing rightward, and z-axis pointing downwards, following the right-hand rule. Since the structure configuration of the platform and the design of the onboard system are highly symmetric, it is reasonable to assume that the longitudinal and lateral dynamics of this platform are exactly the same, and the model is completely decoupled among all four channels. Hence, we can identify the dynamic models of the four channels independently. The overall system dynamics can be obtained by merging the four subsystem dynamics diagonally.

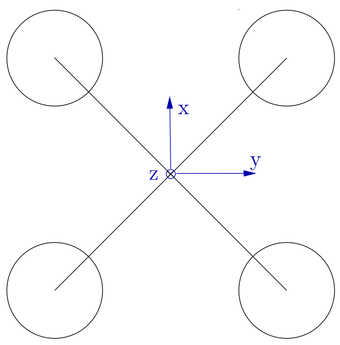


Figure 5.2: Quadrotor body frame definition

The model identification process is again done with the help from CIFER. It first converts the collected input-output data to frequency-domain responses. Then the frequency domain data are fed into NAVFIT, which is a low-order transfer function fitting tool. This is justified since the quadrotor model is decoupled and the subsystem dynamics are assumed to be low order linear invariant systems.

5.2 Roll Pitch Channel Model Identification

Due to the symmetric structure of of the quadrotor platform, the roll and pitch dynamics share the same model structure as well as parameters. When the platform is perturbed in the aileron or elevator channels, the onboard avionics system can record down the responses of roll angle ϕ (or pitch angle θ), the corresponding body-frame linear velocities v (or u), and the synchronized control inputs δ_{ail} (or δ_{ele}). The ultimate goal is to identify the dynamic model from control inputs to the body-frame velocities. However, we can divide this task into two sub-tasks, i.e., identify the model from control inputs to attitude angles and identify the model from angles to velocities. The former part contains information of inner-loop bandwidth and steady-state gain, while the latter part can be used to connect the outer-loop control outputs to the inner-loop control references. The details will be explained in the Section 5.5.

Model from Control Input to Attitude Angle

Using NAVFIT in CIFER, the transfer function from the aileron (or elevator) control input δ_{ail} (or δ_{ele}), to the roll ϕ (or pitch θ) angle can be well fitted by the following 4th order linear

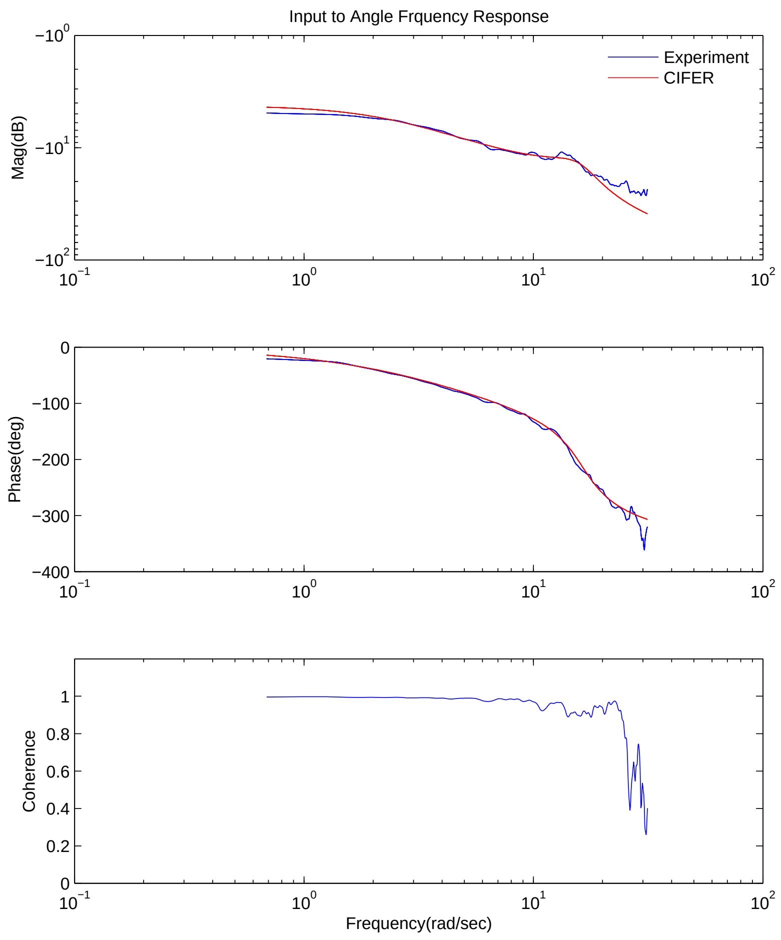


Figure 5.3: Response comparison using frequency-sweep input $\{\delta_{\text{ail}}\ \delta_{\text{ele}}\} - \{\phi, \theta\}$

process model:

$$\frac{\phi(s)}{\delta_{\text{ail}}(s)} = \frac{\theta(s)}{\delta_{\text{ele}}(s)} = \frac{9688}{s^4 + 27.68\, s^3 + 485.9\, s^2 + 5691\, s + 15750}. \tag{5.1}$$

It is a transfer function with bandwidth of 3.89 rad/s and steady-state gain of 0.6151. The frequency response comparison between the identified model and the real data is shown in Fig. 5.3. The third sub-plot in the figure shows the coherence value of the frequency domain matching. At frequencies below 20 rad/s, the coherence value remains above 0.8, indicating that the system can be well characterized by a linear process in this frequency range.

Time domain verification of the model using a different set of experimental data is performed also. The input signal from the verification data set is fed into the model and its predicted output is compared with the experimental output. Fig. 5.4 shows the model performance for a series of chirp signals, and Fig. 5.5 shows the error difference between the model output and the

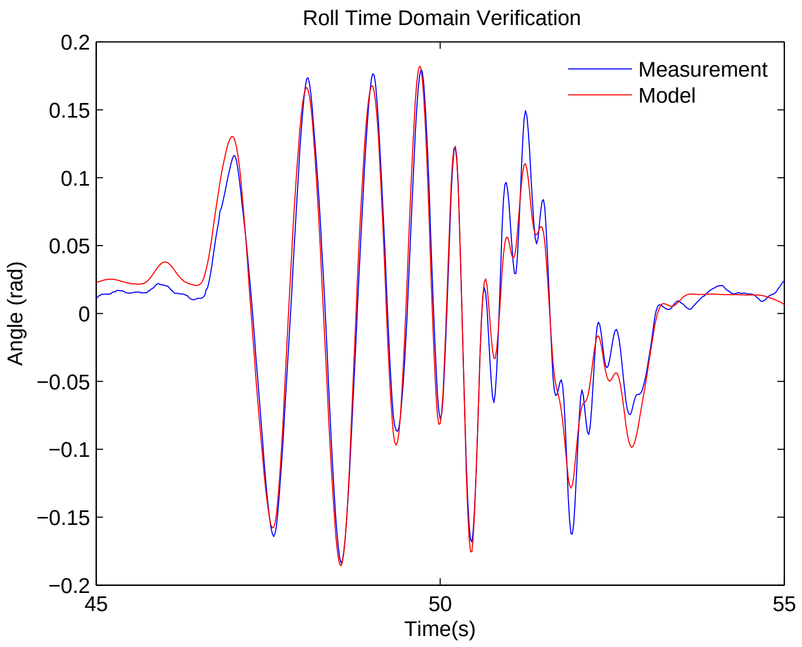


Figure 5.4: Time domain model verification

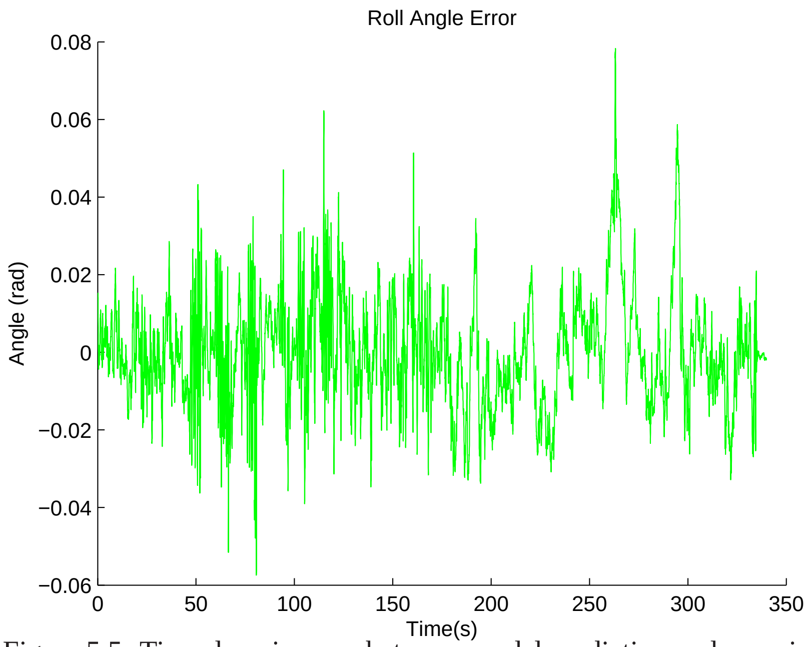


Figure 5.5: Time domain error between model prediction and experiment

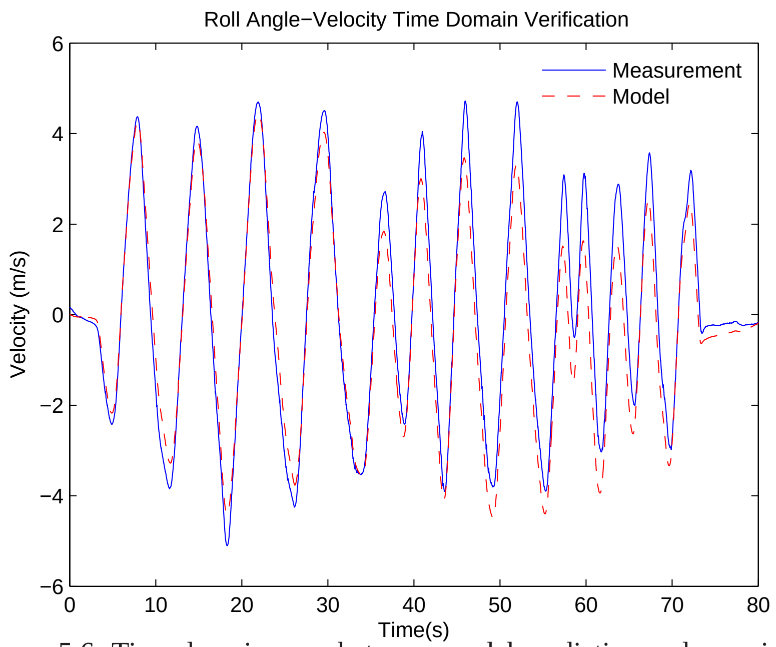


Figure 5.6: Time domain error between model prediction and experiment

experimental output. It can be seen that the error is very small, indicating that the obtained model is very reliable.

Model from Attitude Angle to Linear Velocity

Using the same approach, the transfer function from roll ϕ (or pitch θ) angle to the lateral (or longitudinal) velocity can be obtained by fitting a first order transfer function as below:

$$\frac{v(s)}{\phi(s)} = \frac{u(s)}{\theta(s)} = \frac{8.661}{s + 0.09508}. \tag{5.2}$$

This relationship will be used later in Section 5.5 to connect the inner-loop and outer-loop control layers. The time domain verification results are shown in Fig. 5.6.

5.3 Yaw Channel Model Identification

Since the inner-loop dynamics in the yaw channel is extremely fast, thanks to the superb performance from Naza-M, the relationship between the rudder input δ_{rud} and the yaw rate r can be treated as a static gain. If we consider the outer-layer dynamics in this channel also, then the transfer function from rudder input δ_{rud} to the yaw angle ψ is just an integration of a constant:

$$\frac{\psi(s)}{\delta_{\text{rud}}(s)} = \frac{3.372}{s}. \tag{5.3}$$

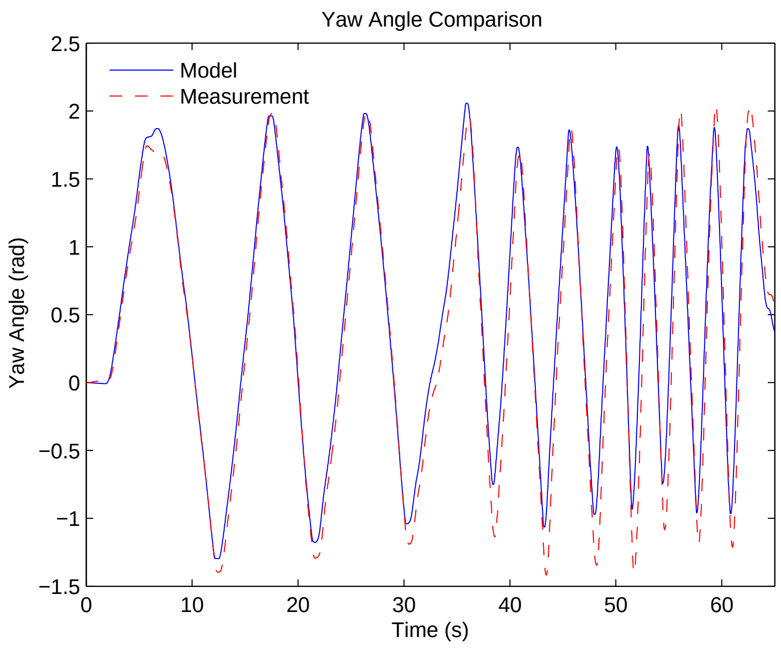


Figure 5.7: Time domain comparison of yaw angle between model prediction and experiment

Fig. 5.7 and Fig. 5.8 shows the time domain verification results for both the yaw angle and angular rate. In both figure, the experimental data agrees well with that predicted by the identified model.

5.4 Heave Channel Model Identification

The transfer function from the throttle input δ_{thr} to the body-frame z-axis velocity w is identified as:

$$\frac{w(s)}{\delta_{\text{thr}}(s)} = -\frac{13.35}{s+2.32}\,. \tag{5.4}$$

The negative sign is due to the opposite definition of positive direction for the input and output. When the throttle stick is pushed up, all four motors speed up. The generated force will lift the UAV platform upwards. However, this upward motion is actually seen as a negative velocity as defined in the z-axis of the UAV body frame. Fig. 5.9 shows the time domain verification results for the heave velocity.

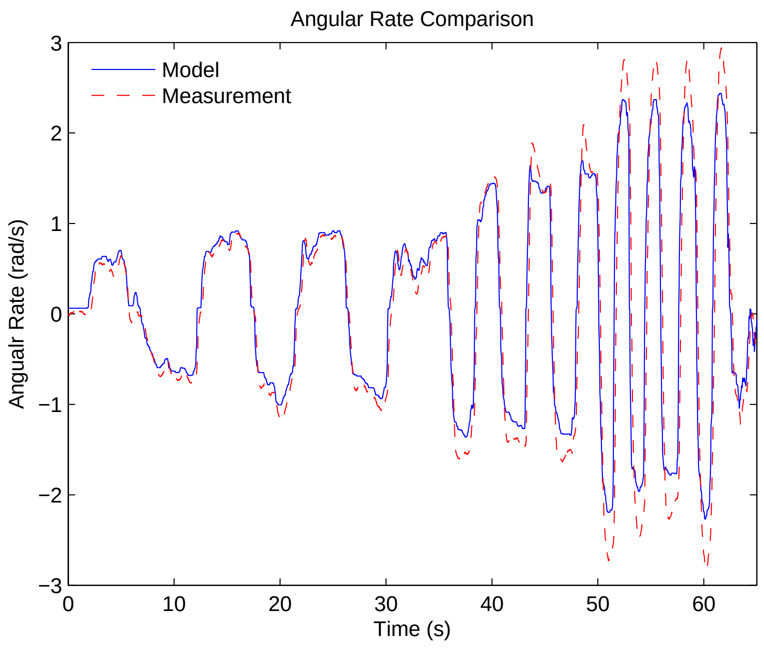


Figure 5.8: Time domain comparison of yaw angular rate between model prediction and experiment

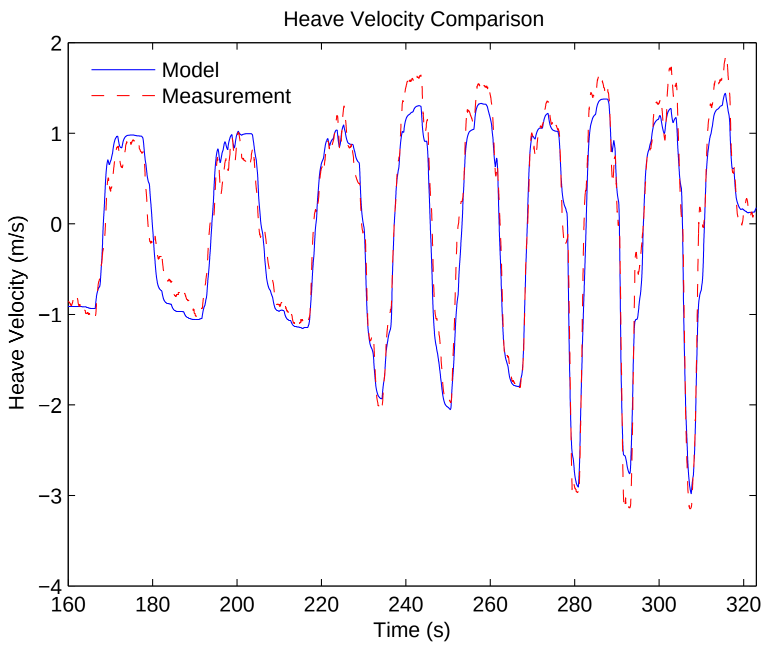


Figure 5.9: Time domain comparison of heave velocity between model prediction and experiment

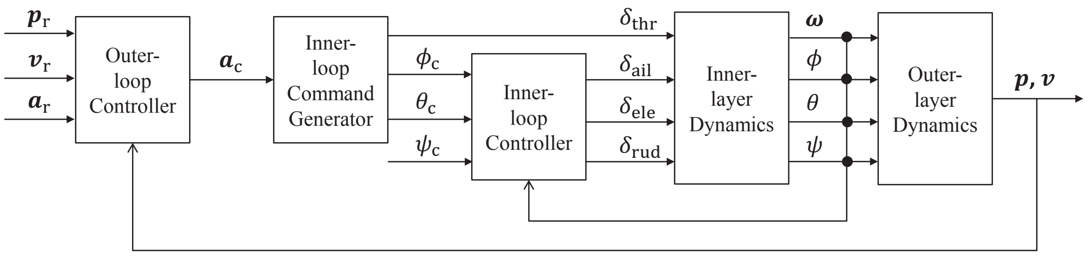


Figure 5.10: Control structure of the quadrotor UAV

5.5 Control Law Design

As the platform is already stabilized in the attitude dynamics by the Naza-M controller (see *Inner-loop controller* in Fig. 5.10), only an outer-loop controller (see *Outer-loop controller* in Fig. 5.10) for position tracking needs to be designed. Here, we adopt a RPT control concept from [18] and apply it to the outer-loop control of the quadrotor UAV. Theoretically, a system controlled by this method is able to track any given reference with arbitrarily fast settling time subjected to disturbances and initial conditions. The basic idea is as follows. For a linear time invariant system

$$\Sigma = \begin{cases} \dot{\boldsymbol{x}} &= A\boldsymbol{x} + B\boldsymbol{u} + E\boldsymbol{w} \\ \boldsymbol{y} &= C_1\boldsymbol{x} + D_1\boldsymbol{w} \\ \boldsymbol{h} &= C_2\boldsymbol{x} + D_2\boldsymbol{u} + D_{22}\boldsymbol{w} \end{cases}, \tag{5.5}$$

with $\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{w}, \boldsymbol{y}, \boldsymbol{h}$ being the state, control input, disturbance, measurement and controlled output respectively, the task of RPT controller is to formulate a dynamic measurement control law of the form

$$\dot{\boldsymbol{v}} = A_c(\varepsilon)\boldsymbol{v} + B_c(\varepsilon)\boldsymbol{y} + G_0(\varepsilon)r + ... + G_{\kappa-1}(\varepsilon)r^{\kappa-1},$$
$$\boldsymbol{u} = C_c(\varepsilon)\boldsymbol{v} + D_c(\varepsilon)\boldsymbol{y} + H_0(\varepsilon)r + ... + H_{\kappa-1}(\varepsilon)r^{\kappa-1},$$

so that when an proper $\varepsilon^* > 0$ is chosen,

1. The resulted closed-loop system is asymptotically stable subjected to zero reference.
2. If $e(t, \varepsilon)$ is the tracking error, then for any initial condition $\boldsymbol{x}_0$, there exists:

$$\|e\|_p = (\int_0^\infty |e(t)^p|dt)^{1/p} \to 0, \quad as \quad \varepsilon \to 0. \tag{5.6}$$

For non-zero references, their derivatives are used to generate additional control inputs. Thus, any reference of the form $r(t) = p_1 t^k + p_2 t^{k-1} + ... + p_{k+1}$ are covered in the RPT formulation. Furthermore, any references that have a Taylor series expansion at $t = 0$ can also be tracked using the RPT controller.

Similar to the case introduced in [45], the outer dynamics of the quadrotor is differentially flat. That means all its state variables and inputs can be expressed in terms of algebraic functions of flat outputs and their derivatives. A proper choice of flat outputs could be

$$\sigma = [x, y, z, \psi]^{\mathrm{T}}. \tag{5.7}$$

It can be observed that the first three outputs, x, y, z, are totally independent. In other words, we can consider the UAV as a mass point with constrained velocity, acceleration, jerk, and so forth, in the individual axis of the 3-D global frame when designing its outer-loop control law. Hence, a stand-alone RPT controller based on multiple-layer integrator model in each axis can be designed to track the position reference in that axis. For the x-axis or the y-axis, the nominal system can be written as

$$\begin{cases} \dot{\boldsymbol{x}}_{\mathrm{n}} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \boldsymbol{x}_{\mathrm{n}} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u_{\mathrm{n}} \\ \boldsymbol{y}_{\mathrm{n}} = \boldsymbol{x}_{\mathrm{n}} \end{cases}, \tag{5.8}$$

where $\boldsymbol{x}_{\mathrm{n}}$ contains the position and velocity state variables and u_{n} is the desired acceleration.

To achieve better tracking performance, it is common to include an error integral to ensure zero steady-state error subjected to step inputs. This requires an augmented system to be formulated as

$$\begin{cases} \dot{\boldsymbol{x}}_{xy} = \begin{bmatrix} 0 & -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \boldsymbol{x}_{xy} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} u_{xy} \\ \boldsymbol{y}_{xy} = \boldsymbol{x}_{xy} \\ h_{xy} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \boldsymbol{x}_{xy} \end{cases}, \tag{5.9}$$

where $\boldsymbol{x}_{xy} = \begin{bmatrix} \int(p_e) & p_r & v_r & a_r & p & v \end{bmatrix}^{\mathrm{T}}$ with p_r, v_r, a_r as the position, velocity and acceleration references in the controlled axis, p, v as the actual position and velocity and $p_e = r_p - p$ as the tracking error of position. In Fig. 5.10, $\boldsymbol{x}_x$ and $\boldsymbol{x}_y$ are the respective representation of $\boldsymbol{x}_{xy}$ in the x- and y-axis. By following the procedures in [17], an linear feed back control law of the form below can be acquired,

$$u_{xy} = F_{xy}\boldsymbol{x}_{xy}, \tag{5.10}$$

where

$$F_{xy} = \begin{bmatrix} \frac{k_i\omega_n^2}{\varepsilon^3} & \frac{\omega_n^2 + 2\zeta\omega_n k_i}{\varepsilon^2} & \frac{2\zeta\omega_n + k_i}{\varepsilon} & 1 & -\frac{\omega_n^2 + 2\zeta\omega_n k_i}{\varepsilon^2} & -\frac{2\zeta\omega_n + k_i}{\varepsilon} \end{bmatrix}.$$

Here, ε is a design parameter to adjust the settling time of the closed-loop system. ω_n, ζ, k_i are the parameters that determines the desired pole locations of the infinite zero structure of (5.9) through

$$p_i(s) = (s + k_i)(s^2 + 2\zeta\omega_n s + \omega_n^2) \tag{5.11}$$

The z-axis control is similar but in a lower-order form. As the inner-loop is directly looking for velocity reference in this axis, it is straight forward to model the outer loop as a single integrator from velocity to position, and it leads to the augmented system as

$$\begin{cases} \dot{\boldsymbol{x}}_z = \begin{bmatrix} 0 & -1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \boldsymbol{x}_z + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} u_z \\ \boldsymbol{y}_z = \boldsymbol{x}_z \\ h_z = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} \boldsymbol{x}_z \end{cases}. \tag{5.12}$$

where $\boldsymbol{x}_z = \begin{bmatrix} \int (p_e) & p_r & v_r & p \end{bmatrix}^{\mathrm{T}}$. This leads to a linear feedback control law of

$$u_z = F_z \boldsymbol{x}_z, \tag{5.13}$$

where

$$F_z = \begin{bmatrix} -\frac{\omega_n^2}{\varepsilon} & \frac{2\omega_n\zeta}{\varepsilon^2} & 1 & -\frac{2\omega_n\zeta}{\varepsilon^2} \end{bmatrix}.$$

Theoretically, when the design parameter ε is small enough, the RPT controller can give arbitrarily fast responses. However, in real life, due to the constraints of the UAV physical dynamics and its inner-loop bandwidth it is safer to limit the bandwidth of the outer loop to be one fifth to one third of the controlled inner-loop system. For the case of QuadLion, the following design parameters are used:

$$x, y \text{ axis}: \begin{cases} \varepsilon & = & 1 \\ \omega_n & = & 0.99 \\ \zeta & = & 0.707 \\ k_i & = & 0.25 \end{cases} \qquad z \text{ axis}: \begin{cases} \varepsilon & = & 1 \\ \omega_n & = & 0.559 \\ \zeta & = & 2 \end{cases}$$

There is still one problem unsolved. From Fig. 5.10, it can be seen that the output from the outer-loop controller in physical meaning is the desired accelerations in xy-axis and the desired velocity in z-axis, both in global frame. However, the inner-loop controller is looking for attitude references (ϕ_{r}, θ_{r}, ψ_{r}) and the body-frame z-axis velocity reference. A conversion is needed to link the two control layers together. This leads to another functional block called the *Inner-loop command generator*, in which a rotational conversion from the global frame to the body frame $\mathbf{R}_{\mathrm{b/g}}$ is needed and another matrix $\mathbf{G}_c$ is used to convert the desired acceleration references to the desired attitude angles. For all quadrotor UAVs,

$$\mathbf{G}_{\mathrm{c}} \approx \begin{bmatrix} 0 & 1/g \\ -1/g & 0 \end{bmatrix}. \tag{5.14}$$

where g is the gravitational constant.

5.6 Flight Test Results

The designed outer-loop control law is coded in C++ and executed by the onboard computer of the actual quadrotor platform. Both indoor and outdoor flight tests were carried out to verify the control performance of the closed-loop system. For the indoor flight test, a scenario similar to that of the coaxial indoor test was carried out. The quadrotor was commanded to hover in the corner of an indoor hall at 2 meters away from two perpendicular walls in front and on its left. Fig. 5.11 shows the x, y position logged in the flight test. The overall position error is obviously smaller than that of the coaxial helicopter and there is no circular motion observed during the flight, which further confirms that the lateral and longitudinal channels of the quadrotor dynamics are almost decoupled.

Outdoor flight tests were also carried out with full position and velocity measurement available. This test isolates the dependencies of control law design and state estimation. With full state measurable for the outer loop, the control law can be reliably tested without worrying about state estimation looping back in to the controller and vice versa. Fig. 5.12 and Fig. 5.13 show the logged position, velocity and heading during a way point flight test. The position reference is a rectangular path in the xy plane, while the z reference keeps constant after issuing the way point flight command (at about $t = 110$). It can be observed that the quadrotor position and velocity tracks their corresponding references quite well. The maximum error occurred for the whole flight was within the safety margin.

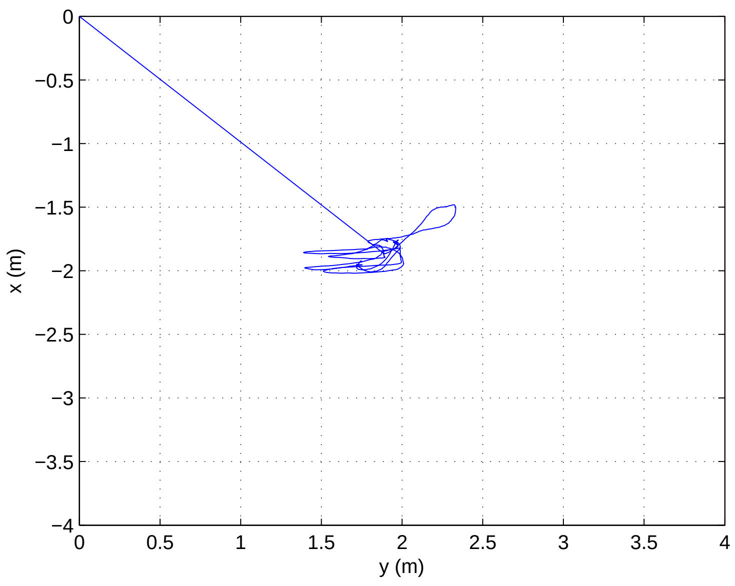


Figure 5.11: Indoor hover flight test for the quadrotor

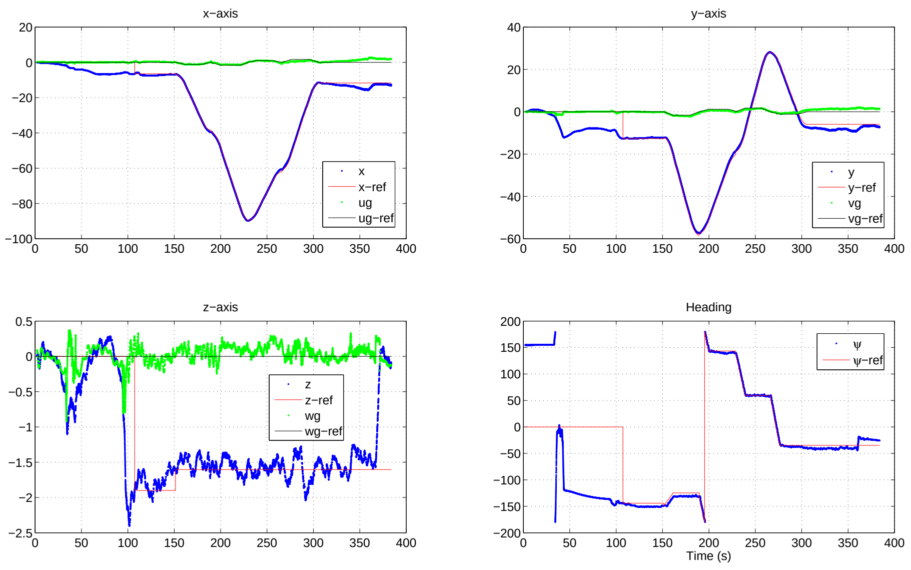


Figure 5.12: Waypoint flight test for the quadrotor - Result 1

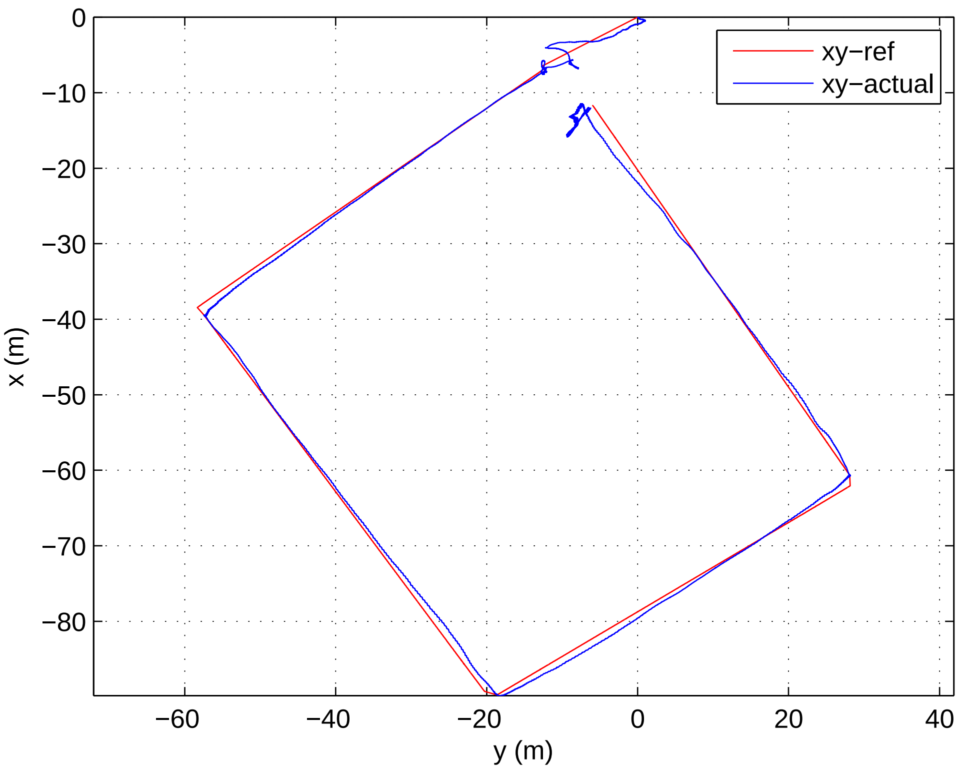


Figure 5.13: Waypoint flight test for the quadrotor - Result 2

Chapter 6

Vision and Laser Based Odometry for Unknown Indoor Environments

While angle and angular rate measurements required by the UAV inner-loop controller can be directly provided by the onboard IMU, the outer loop's position and velocity measurements need to be estimated indirectly. Since dead reckoning using IMU acceleration will face severe drifting problems, information from other sensors must be fused in to cure the divergence and at the same time, reduce noises. After thorough consideration and review, the best solution goes to the onboard camera and the laser scanner.

6.1 Visual Odometry

Odometry is the technique of estimating change of position over time by analyzing data from moving actuators or sensors. Traditional odometry uses information from the movement of actuators to estimate change in position through devices such as rotary encoders to measure wheel rotations. While useful for many wheeled or tracked vehicles, traditional odometry techniques are difficult to be applied to mobile robots with non-standard locomotion and impossible to be used on aerial robots. To solve this problem, a new type of odometry, which utilizes computer vision technology, has been proposed and developed [38, 41]. Due to its flexibility, visual odometry has been used in a wide variety of robotic applications, such as on the Mars Exploration Rovers. Although odometry will drift in the long run due to its integration nature, accurate data collection and careful equipment calibration can reduce this error to the minimum and it is a perfect choice if velocity is the ultimate information required for some special robotic applica-

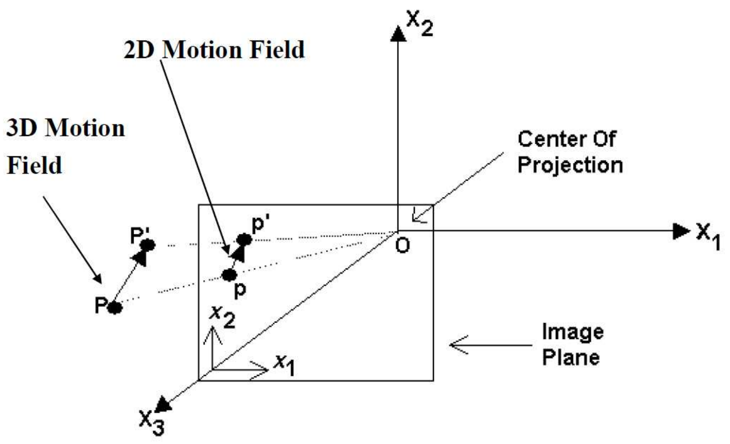


Figure 6.1: Relating 2-D motion and 3-D motion

tions.

In the robotics community, when a robot moves in an unknown environment, the SLAM technique aims to build the map of the environment and localize the robot in the map by analyzing information from inputs and sensor measurements. To solve the same problem in a computer vision sense, it is called the technique of structure from motion (SFM). In technical terms, SFM is defined as the method to recover the 3-D rigid transformation (rotation and translation) of a camera sensor and the 3-D structure of the imaged scene by extracting information from multiple views of the scene projected in the 2-D images. There are two fundamental problems in SFM:

1. Correspondence – 2-D motion in the image: Which elements in Frame 1 corresponds to which elements in the Frame 2.
2. Reconstruction – 3-D motion of the camera: Given a number of correspondences, and possibly the knowledge of camera's intrinsic parameters, how to recover the 3-D motion and structure of the observed world.

Fig. 6.1 illustrates the idea that when a 3-D feature point moves relative to a camera, its projection on the 2-D image will have a corresponding movement. By looking at this relationship in a reverse way, if the feature point is static in the 3-D environment, by observing its projected 2-D motion in the image, the 3-D motion of the camera with respect to this static feature point can be interpreted. It is important to highlight that the 3-D to 2-D mapping is unique while the 2-D to 3-D mapping is not, because there is loss of dimension in the process of camera projection. In consequence, the information of 2-D motion from a single feature point is not sufficient to recover 3-D motion of the camera. Intuitively, if the 2-D motion of more than one feature point can be obtained, then it may be possible to recover the camera 3-D motion. In the following content, solutions to the above two problems will be proposed and implemented by considering the case of UAV indoor navigation. Unlike the pure vision approaches used by researchers from the computer vision society, the proposed visual odometry methods in this thesis achieve accurate and real-time performance by fully utilizing resources from the UAV avionic system and reasonable assumptions about a structured indoor environment.

6.1.1 2-D Optical Flow Computation

There are two prevalent approaches in literature to compute the 2-D optical flow between consecutive images:

1. The gradient techniques – to calculate optical flow from spatial and temporal image intensity derivatives;
2. The feature matching techniques – to correspond distinctive feature points among consecutive frames.

The gradient techniques [43, 71, 78] are stable and fast in computation but they require assumptions like constant ambient illumination and small, continuous motion of the camera sensor. The feature matching techniques [42, 9] can handle uneven illumination and relatively larger motion but suffer from scattered outliers and are computationally intensive. Since the motion of the camera mounted on an indoor UAV should be continuous and slow, and the illumination in an indoor environment is usually homogeneous, the gradient techniques are very suitable here. By further considering the limited computational power on a miniature indoor UAV, the feature matching techniques will not be chosen for this thesis.

The problem of estimating optical flow requires the knowledge of spatial and temporal image intensity derivatives. Let the grey-level intensity of a pixel on the image be $E(x, y, t)$. It is a continuous and differentiable function of space and time. Suppose the brightness pattern is locally displaced by a distance dx, dy over time period dt, the intensity of the displaced pixel should be the same as the original pixel, i.e.

$$E(x, y, t) = E(x + dx, y + dy, t + dt). \tag{6.1}$$

In other words, the total derivative of E w.r.t. time is zero:

$$\frac{dE}{dt} = 0, \tag{6.2}$$

or

$$\frac{\delta E}{\delta x}\frac{dx}{dt} + \frac{\delta E}{\delta y}\frac{dy}{dt} + \frac{\delta E}{\delta t} = 0. \tag{6.3}$$

If we denote

$$E_x = \frac{\delta E}{\delta x}, E_y = \frac{\delta E}{\delta y}, E_t = \frac{\delta E}{\delta t}, v_x = \frac{dx}{dt}, v_y = \frac{dy}{dt},$$

then the equation can be rewritten as:

$$E_x v_x + E_y v_y + E_t = 0. \tag{6.4}$$

The above is known as the Brightness Constancy Equation (BCE). It is valid if the intensity changes on the image pixels are caused by camera motion only.

In BCE, E_x, E_y and E_t are measurable, while v_x and v_y are the unknown 2-D flows. One individual BCE is not enough to estimate the 2-D flows. Solutions to this normally involve considering a small window of adjacent pixels. Among all these solutions, the Lucas & Kanade algorithm [43] is the most classical and elegant one. It assumes that the motion field at a given time is constant over a block of pixels. For that particular block of n number of pixels, there are n BCEs. They form a linear over-determined equation set. Least square estimation can be used to obtain a reliable v_x, v_y over that region:

$$\begin{bmatrix} E_{1,x} & E_{1,y} \\ E_{2,x} & E_{2,y} \\ \vdots & \vdots \\ E_{n,x} & E_{n,y} \end{bmatrix} \begin{bmatrix} v_x \\ v_y \end{bmatrix} = \begin{bmatrix} -E_{1,t} \\ -E_{2,t} \\ \vdots \\ -E_{n,t} \end{bmatrix} \tag{6.5}$$

The above optical flow calculation can be carried out at any pixel locations (x, y) on the image. However, some locations are more stable and reliable for the calculation, while others are not. First of all, if an image region is almost homogeneous, then the values of E_x, E_y, E_t are

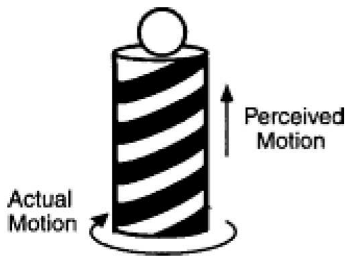


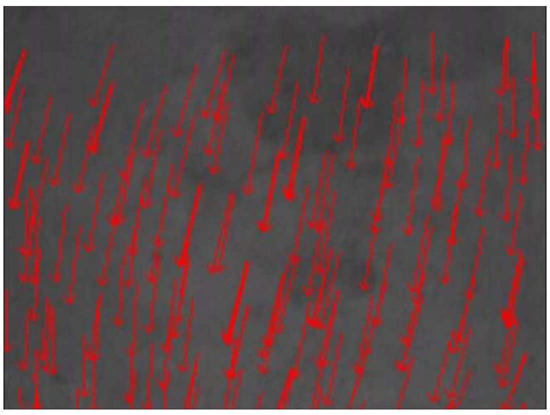
Figure 6.2: Aperture problem – the barber pole illusion

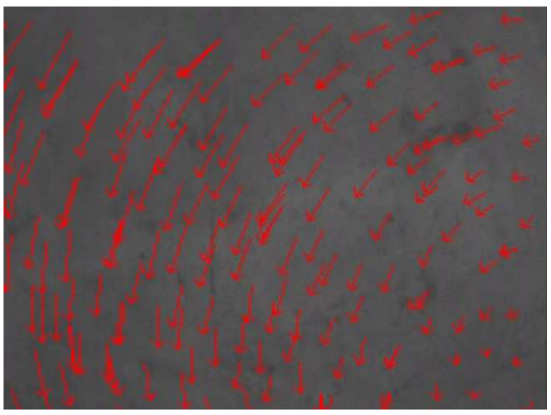
near zero. This will result in numerical problems or an unsolvable under-determined equation set. Second, even if the region is not homogeneous, there is the well-known aperture problem (see Fig. 6.2). Motion flow is ambiguous when only straight edges are being observed. In general, corner-like feature points are more suitable to be set as the center of optical flow calculation. Hence, detection of good feature points needs to be done before optical flow calculation is carried out at these particular locations.

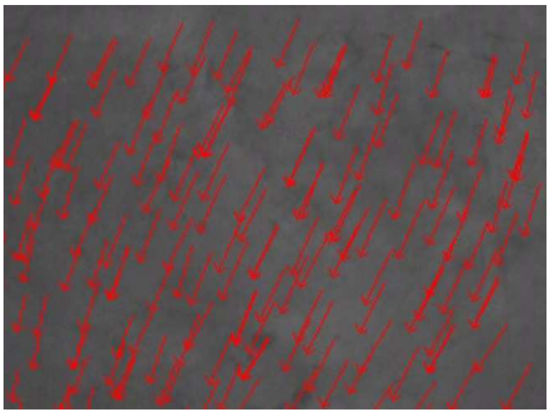
To implement the 2-D optical flow algorithm, the well-known OpenCV library from Intel can be used. OpenCV library also provides the 'goodFeaturesToTrack' function [71] which is able to find predefined number of feature points on an image and these feature points are selected particularly suitable for the later 'calcOpticalFlowPyrLK' function, which calculates the optical flow for a sparse feature set using the iterative Lucas-Kanade method with pyramids [78]. The implementation result can be found in Fig. 6.3. In this case, the camera sensor looks vertically downward on the indoor floor. By examining the flow patterns, the 3-D motion of the camera, thus also the UAV motion, can be vividly observed. However, it is indeed very complicated to recover the full 3-D motion from 2-D optical flow in a rigorous way. The following content will propose two efficient methods to compute the UAV 3-D velocity with fine accuracy. Method 1 is based on a forward-facing camera which looks at objects at different depths and Method 2 is based on a downward-facing camera which looks at visual features on the same ground plane. Both of them have used supplementary information from other sensors on the UAV avionic system, without which the 2-D to 3-D recover problem is extremely difficult to be solved.

6.1.2 3-D Motion Estimation via Optical Flow - Method 1

Refer to Fig. 6.4, let the camera motion expressed as a translation, $\boldsymbol{T} = (T_x, T_y, T_z)^{\mathrm{T}}$ and a rotation, $\Omega = (\omega_x, \omega_y, \omega_z)^{\mathrm{T}}$, P is the 3-D position of the observed point in the camera frame, expressed as $(X, Y, Z)^{\mathrm{T}}$. The equation of perspective projection expressed in the camera frame


(a) Time step 1


(b) Time step 2


(c) Time step 3

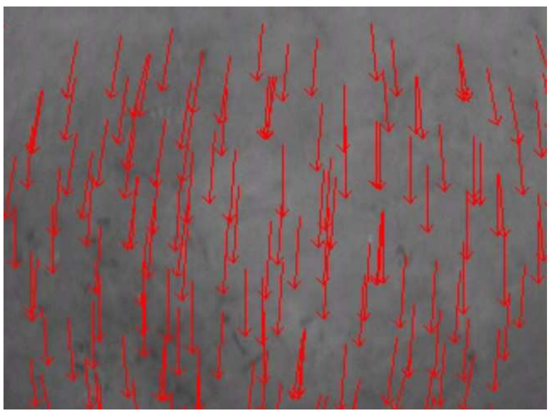

(d) Time step 4

Figure 6.3: The 2-D optical flow implementation result

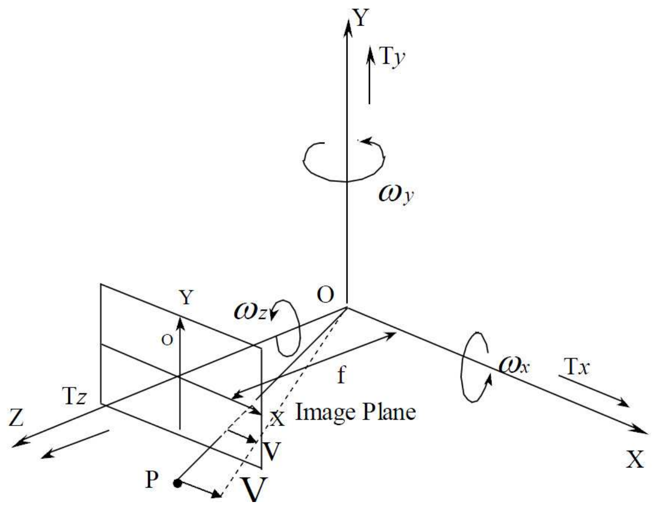


Figure 6.4: 3-D motion of camera

is very simple:

$$x = f \cdot \frac{X}{Z}, \tag{6.6}$$

$$y = f \cdot \frac{Y}{Z}, \tag{6.7}$$

where f is the focal length of the camera, which can be pre-calibrated, and (x, y) represent the pixel position of the projected point on the 2-D image. Taking derivatives with respect to time on both sides:

$$\begin{aligned} x' = v_x = f(\frac{X'}{Z} - \frac{XZ'}{Z^2}), \\ y' = v_y = f(\frac{Y'}{Z} - \frac{YZ'}{Z^2}). \end{aligned} \tag{6.8}$$

On the LHS, v_x, v_y are the 2-D optical flow which has already been obtained. On the RHS, there are X', Y', Z'. If combined together, they represent the 3-D linear velocity of the feature points w.r.t the camera, $\mathbf{V}$, and

$$\mathbf{V} = -\mathbf{T} - \Omega \times P, \tag{6.9}$$

or

$$\begin{aligned} X' &= -T_x - \omega_y Z + \omega_z Y, \\ Y' &= -T_y - \omega_z X + \omega_x Z, \\ Z' &= -T_z - \omega_x Y + \omega_y X. \end{aligned} \tag{6.10}$$

Substituting (6.10) into (6.8), it can be derived that:

$$\begin{aligned} v_x &= \frac{T_z x - T_x f}{Z} - \omega_y f + \omega_z y + \frac{\omega_x xy}{f} - \frac{\omega_y x^2}{f}, \\ v_y &= \frac{T_z y - T_y f}{Z} - \omega_x f + \omega_z x + \frac{\omega_y xy}{f} - \frac{\omega_x y^2}{f}. \end{aligned} \tag{6.11}$$

The first term involving T_x, T_y, T_z is related to optical flow caused by translational motion, while the other terms involving ω_x, ω_y, ω_z are related to optical flow caused by rotational motion. It can be easily seen that if a set of (T_x, T_y, T_z, Z) satisfy the equation, so does another set (kT_x, kT_y, kT_z, kZ), where k is a scale ambiguity between the translational motion and the

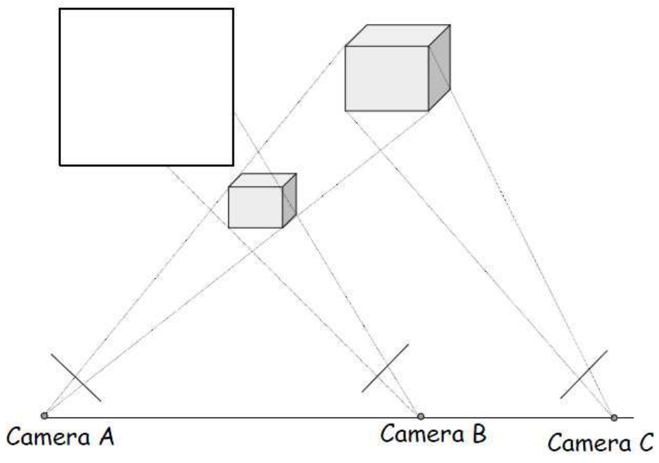


Figure 6.5: Scale ambiguity between translation amount and depth

feature depth. This scale ambiguity problem is well-known in the computer vision society and Fig. 6.5 gives a graphical illustration. Another challenging problem here is that the equations are nonlinear. Simple linear least square methods cannot be applied directly. Instead, high-dimensional searching algorithms or iterative methods are needed which hinder the calculation to be carried out onboard of the UAV in real time. Fortunately, these two problems only exist when information despite vision is not allowed to be used. On the UAV platforms used for this indoor navigation project, there are also valuable information from two other sensors, namely rotational motion (ω_x, ω_y, ω_z) from IMU and feature depth information from the laser range sensor. If we rearrange (6.11), it becomes

$$\begin{aligned} v_x - (-\omega_y f + \omega_z y + \frac{\omega_x xy}{f} - \frac{\omega_y x^2}{f}) &= \frac{T_z x - T_x f}{Z}, \\ v_y - (-\omega_x f + \omega_z x + \frac{\omega_y xy}{f} - \frac{\omega_x y^2}{f}) &= \frac{T_z y - T_y f}{Z}, \end{aligned} \tag{6.12}$$

where the LHS terms are all measurable. Let

$$\begin{aligned} V_x &= v_x - (-\omega_y f + \omega_z y + \frac{\omega_x xy}{f} - \frac{\omega_y x^2}{f}), \\ V_y &= v_y - (-\omega_x f + \omega_z x + \frac{\omega_y xy}{f} - \frac{\omega_x y^2}{f}), \end{aligned} \tag{6.13}$$

then

$$\begin{aligned} T_z x - T_x f - Z V_x = 0, \\ T_z y - T_y f - Z V_y = 0. \end{aligned} \tag{6.14}$$

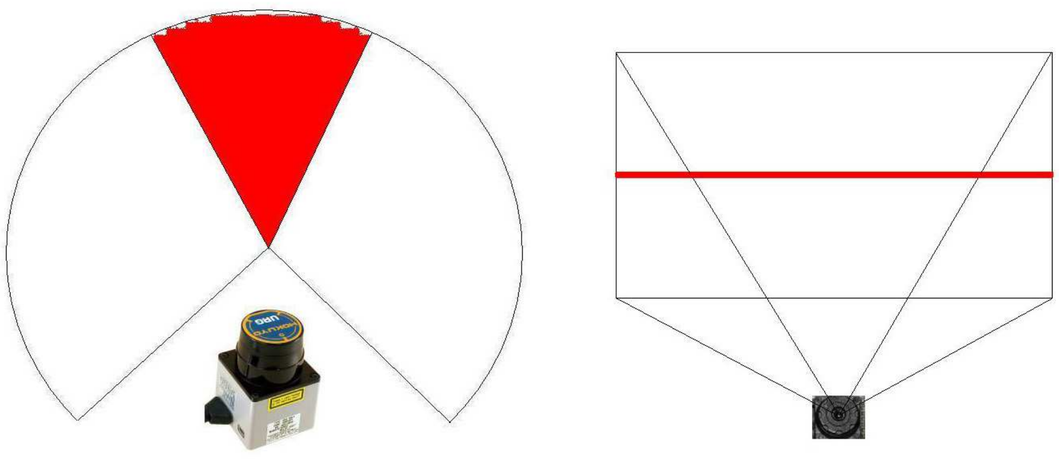


Figure 6.6: Measurement correspondence between laser scanner and camera

Eliminating Z,

$$T_z x V_y - T_x f V_y - T_z y V_x + T_y f V_x = 0. \tag{6.15}$$

For n feature points, the following linear equation set can be formed:

$$\begin{bmatrix} -fu_{1y} & fu_{1x} & xu_{1y} - yu_{1x} \\ -fu_{2y} & fu_{2x} & xu_{2y} - yu_{2x} \\ \vdots & \vdots & \vdots \\ -fu_{ny} & fu_{nx} & xu_{ny} - yu_{nx} \end{bmatrix} \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \tag{6.16}$$

Methods like Singular Value Decomposition (SVD) can be used to solve $(T_x, T_y, T_z)^{\mathrm{T}}$ up to the scale k:

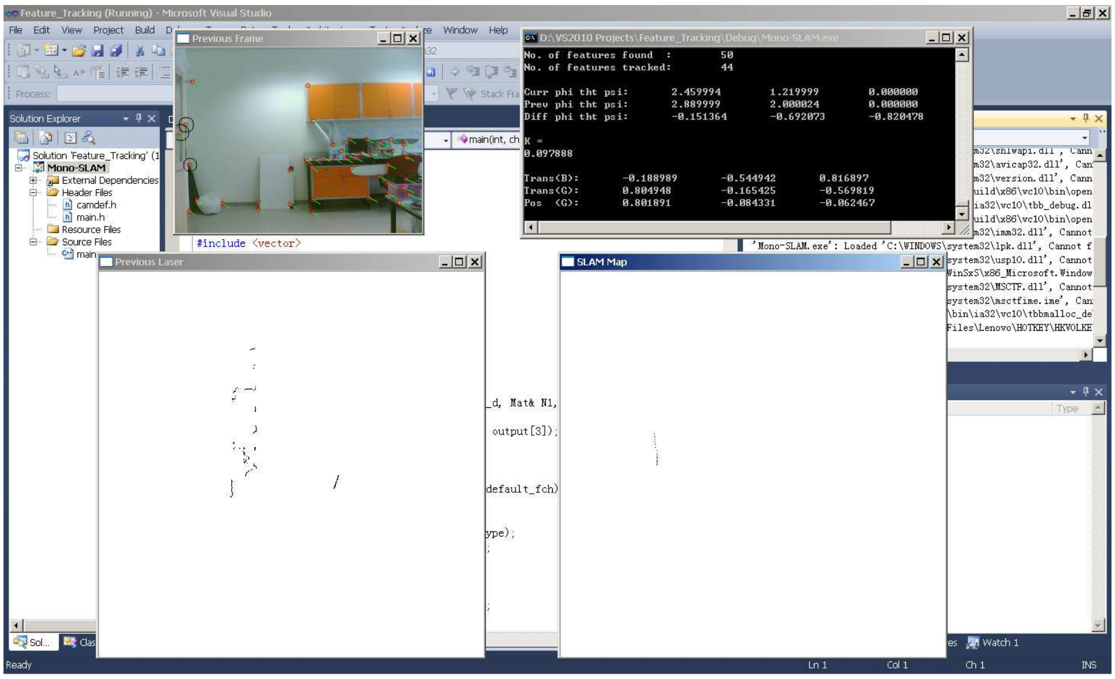
$$\begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix} = k \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \tag{6.17}$$

Till now, $(t_x, t_y, t_z)^{\mathrm{T}}$ can be determined but k is still unknown. To determine k, depth information from the observed feature points need to be known. This depth information can be acquired by including measurements from the laser scanner. Fig. 6.6 shows a scenario when the laser scanner and the camera both pointing forward. It can be seen that the laser scanner measures object distances in a fan-shaped 2-D plane, while the camera sees objects in a front-expanding

3-D space. By projecting the laser scanner's scanning plane onto the camera image, it actually forms a horizontal strip roughly at the center of the image. Hence, optical flows calculated within this strip region will have their corresponding feature depths known via the laser scanner measurements. By using this depth information, we can substitute (6.17) and features with known depth information Z back to (6.14) and obtain k in a least square manner. In this way, the translational motion $k \times (t_x, t_y, t_z)^{\mathrm{T}}$ in the camera frame can be obtained. By rotating this vector from the camera frame to the UAV-carried NED frame, the UAV velocity can be obtained. Furthermore, by integrating this term, the UAV NED position can also be estimated.

Experiments have been carried out to verify the aforementioned visual odometry method. The UAV platform is held in hand and moved around in an indoor room with a speed of roughly 0.5 m/s. The UAV was given a throttle channel input at trimming value through out the test so that the rotors were spinning, thus generating a realistic amount of vibrations. After logging all the sensor data, a C++ program is written to test the performance of the proposed algorithm. Figs. 6.7–6.18 have shown progressively how the position of the UAV has been estimated. For each figure, the top-left image shows the 2-D optical flows, within which the black circles indicate feature points being used to calculate k, the scale of translation, with the radius of the circle proportional to the feature depth. The bottom-left image displays laser scanner information, where objects within 4 meters in the UAV horizontal plane can be detected. The bottom-right image displays the estimated position of the UAV in a 2D trajectory manner.

The motion estimation is good except at $t = 30$ s, the direction of motion becomes a bit erroneous. By analyzing the data, it is found that there is artificial magnetic field at that location, which badly affects the yaw measurement from the IMU. This reflects the limitation of estimating rotational motion of the UAV by IMU only. If algorithms based on laser scanner and vision can also provide this information, then the estimated results can be fused in, thus suppressing conditional error from individual sensors. Another problem of this kind of dead-reckoning algorithms is a position drift. For the experiment mentioned just now, although the UAV has physically come back to its initial position, the estimated path does not precisely close the loop. In fact, if only motion between consecutive frames can be estimated, pure integration always results in position drift in practice. However, if the 3-D positions of a few strong feature points or landmarks can be remembered, thus recognized when they re-enter the camera view, then this drift can be compensated. It is generally called the *loop closure* technique in vision-based SLAM.


Feature_Tracking (Running) - Microsoft Visual Studio
Previous Frame
D:\VS2010 Projects\Feature_Tracking\Debug\Mono-SLAM.exe
Previous Laser
SLAM Map
Solution Explorer
Watch 1
Ready

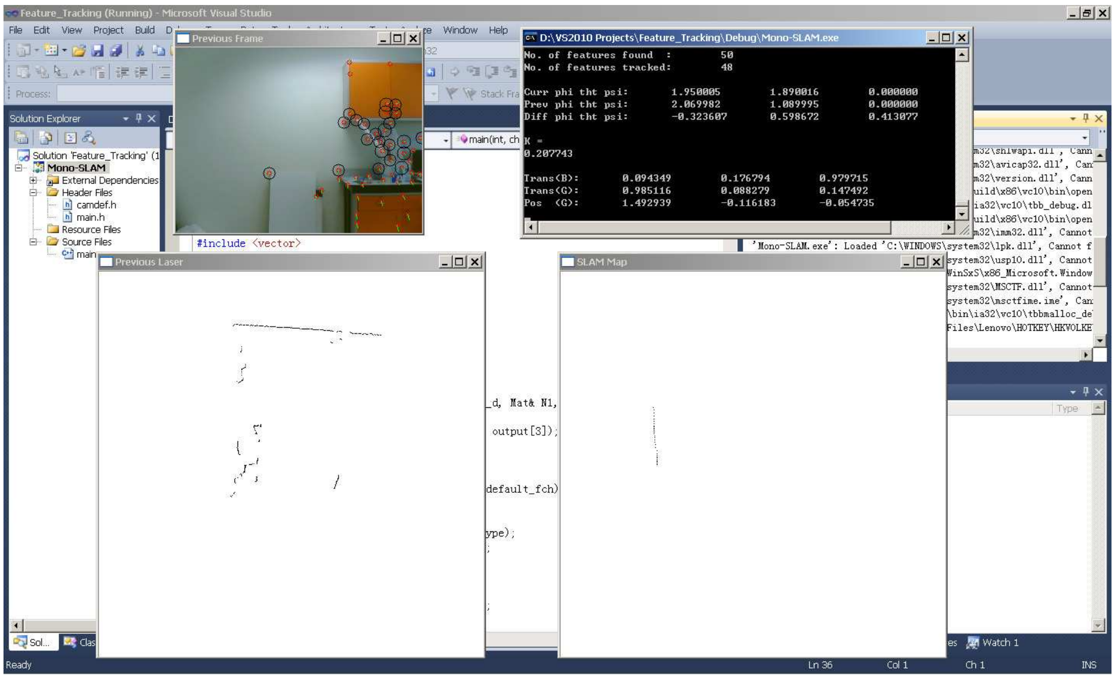
```
No. of features found  :        50
No. of features tracked:        44

Curr phi tht psi:       2.459994        1.219999        0.000000
Prev phi tht psi:       2.889999        2.000024        0.000000
Diff phi tht psi:       -0.151364       -0.692073       -0.820478

K =
0.097888

Trans(B):       -0.188989       -0.544942       0.816897
Trans(G):       0.804948        -0.165425       -0.569819
Pos  (G):       0.801891        -0.084331       -0.062467
```

Figure 6.7: Localization result at $t = 5.0$ s


Feature_Tracking (Running) - Microsoft Visual Studio
Previous Frame
D:\VS2010 Projects\Feature_Tracking\Debug\Mono-SLAM.exe
Previous Laser
SLAM Map
Solution Explorer
Watch 1
Ready

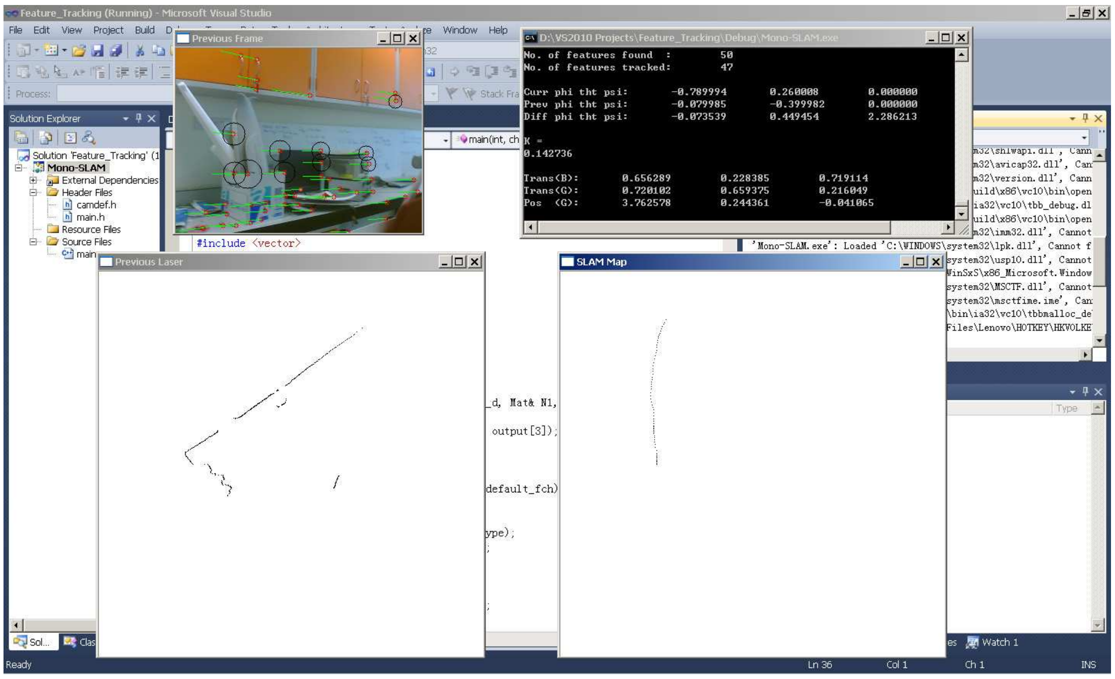
```
No. of features found  :        50
No. of features tracked:        48

Curr phi tht psi:       1.950005        1.890016        0.000000
Prev phi tht psi:       2.069982        1.089995        0.000000
Diff phi tht psi:       -0.323607       0.598672        0.413077

K =
0.207743

Trans(B):       0.094349        0.176794        0.979715
Trans(G):       0.985116        0.088279        0.147492
Pos  (G):       1.492939        -0.116183       -0.054735
```

Figure 6.8: Localization result at $t = 8.1$ s


Feature_Tracking (Running) - Microsoft Visual Studio
Previous Frame
D:\VS2010 Projects\Feature_Tracking\Debug\Mono-SLAM.exe
Previous Laser
SLAM Map
Solution Explorer
Ready

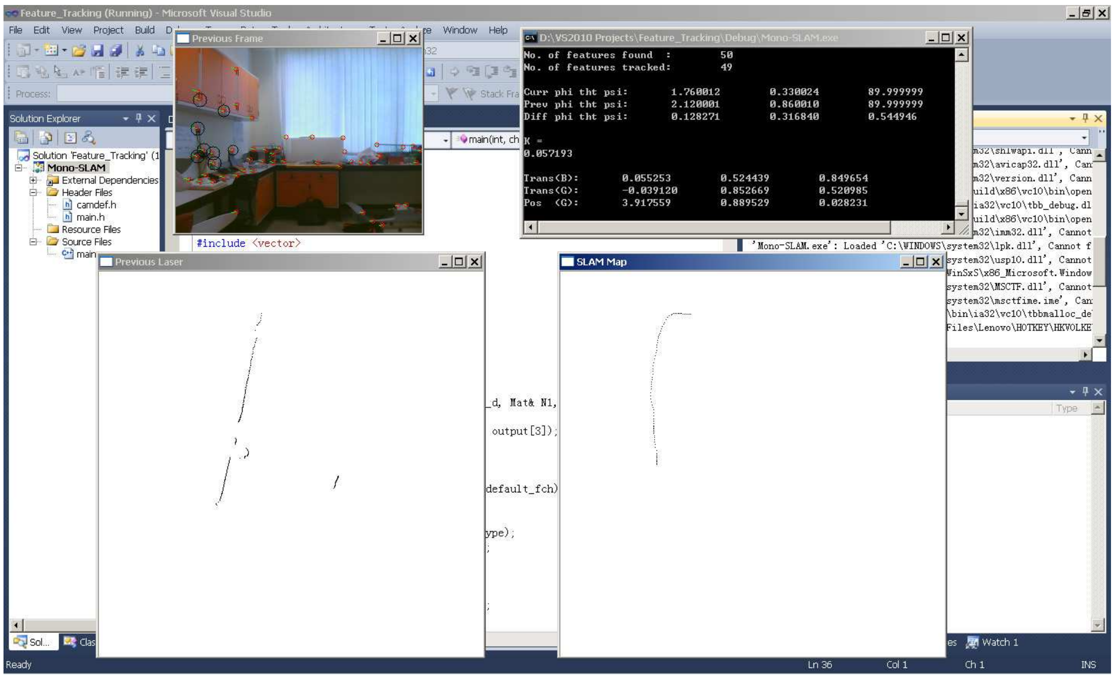
```
No. of features found  :        50
No. of features tracked:        47

Curr phi tht psi:       -0.789994       0.260008        0.000000
Prev phi tht psi:       -0.079985       -0.399982       0.000000
Diff phi tht psi:       -0.073539       0.449454        2.286213

K =
0.142736

Trans(B):       0.656289        0.228385        0.719114
Trans(G):       0.720102        0.659375        0.216049
Pos  (G):       3.762578        0.244361        -0.041065
```

Figure 6.9: Localization result at $t = 11.7$ s


Feature_Tracking (Running) - Microsoft Visual Studio
Previous Frame
D:\VS2010 Projects\Feature_Tracking\Debug\Mono-SLAM.exe
Previous Laser
SLAM Map
Solution Explorer
Ready

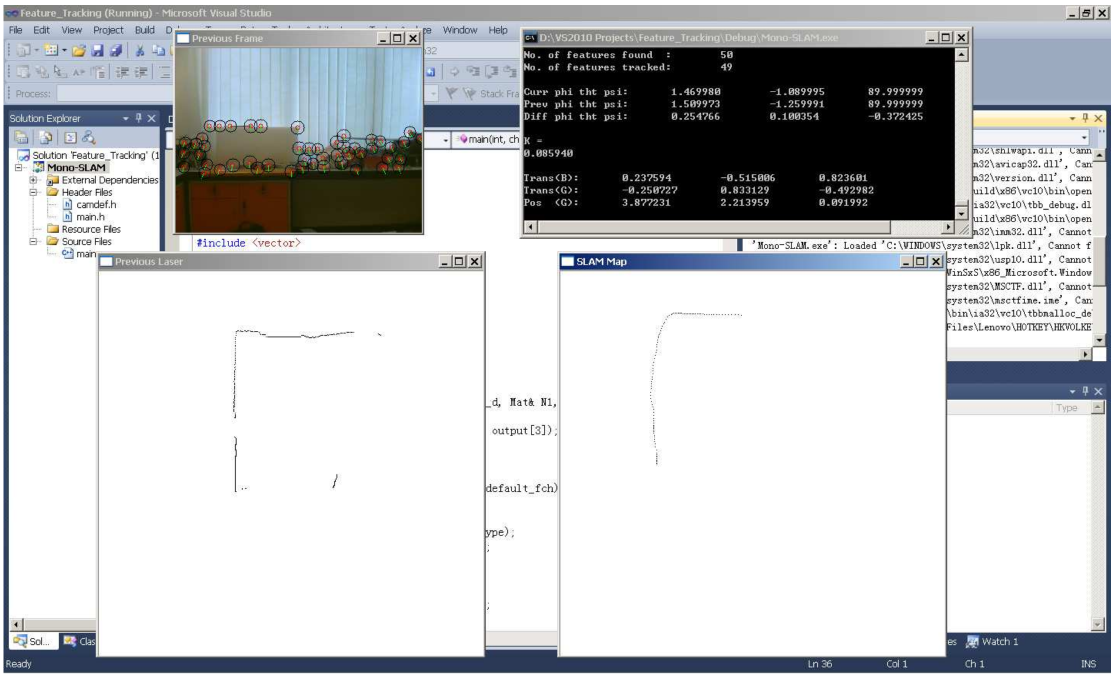
```
No. of features found  :        50
No. of features tracked:        49

Curr phi tht psi:       1.760012        0.330024        89.999999
Prev phi tht psi:       2.120001        0.860010        89.999999
Diff phi tht psi:       0.128271        0.316840        0.544946

K =
0.057193

Trans(B):       0.055253        0.524439        0.849654
Trans(G):       -0.039120       0.852669        0.520985
Pos  (G):       3.917559        0.889529        0.028231
```

Figure 6.10: Localization result at $t = 13.8$ s


Feature_Tracking (Running) - Microsoft Visual Studio
Previous Frame
D:\VS2010 Projects\Feature_Tracking\Debug\Mono-SLAM.exe
Previous Laser
SLAM Map
Solution Explorer
Ready

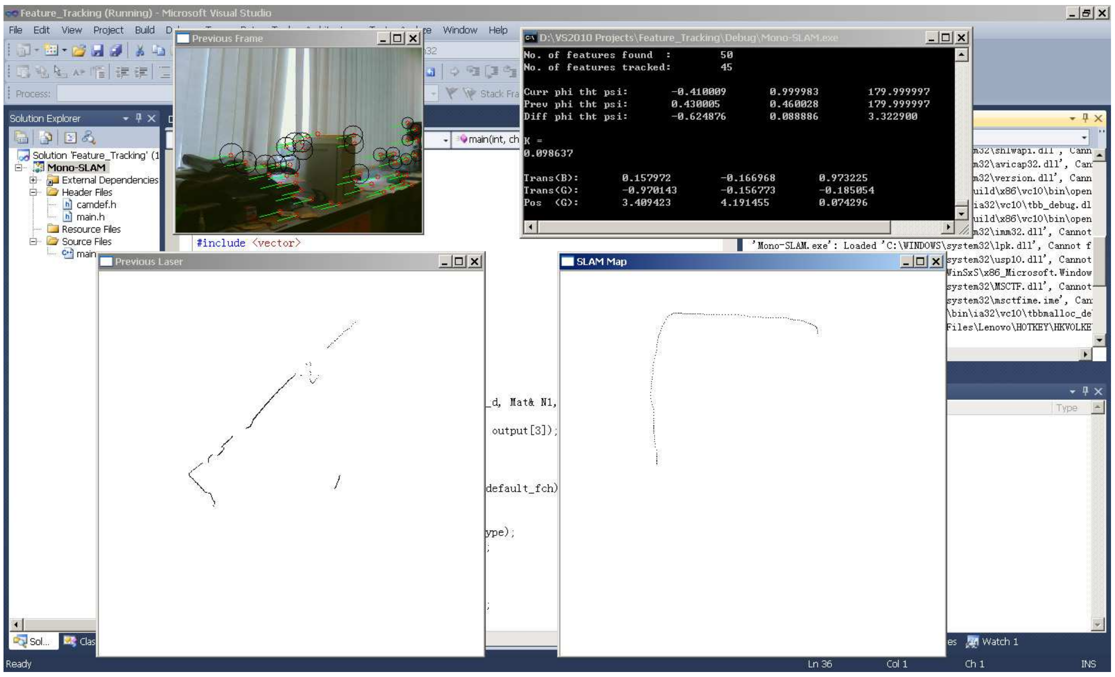
```
No. of features found  :        50
No. of features tracked:        49

Curr phi tht psi:       1.469980        -1.089995       89.999999
Prev phi tht psi:       1.509973        -1.259991       89.999999
Diff phi tht psi:       0.254766        0.100354        -0.372425

K =
0.085940

Trans(B):       0.237594        -0.515006       0.823601
Trans(G):       -0.250727       0.833129        -0.492982
Pos  (G):       3.877231        2.213959        0.091992
```

Figure 6.11: Localization result at $t = 16.8$ s


Feature_Tracking (Running) - Microsoft Visual Studio
Previous Frame
D:\VS2010 Projects\Feature_Tracking\Debug\Mono-SLAM.exe
Previous Laser
SLAM Map
Solution Explorer
Ready

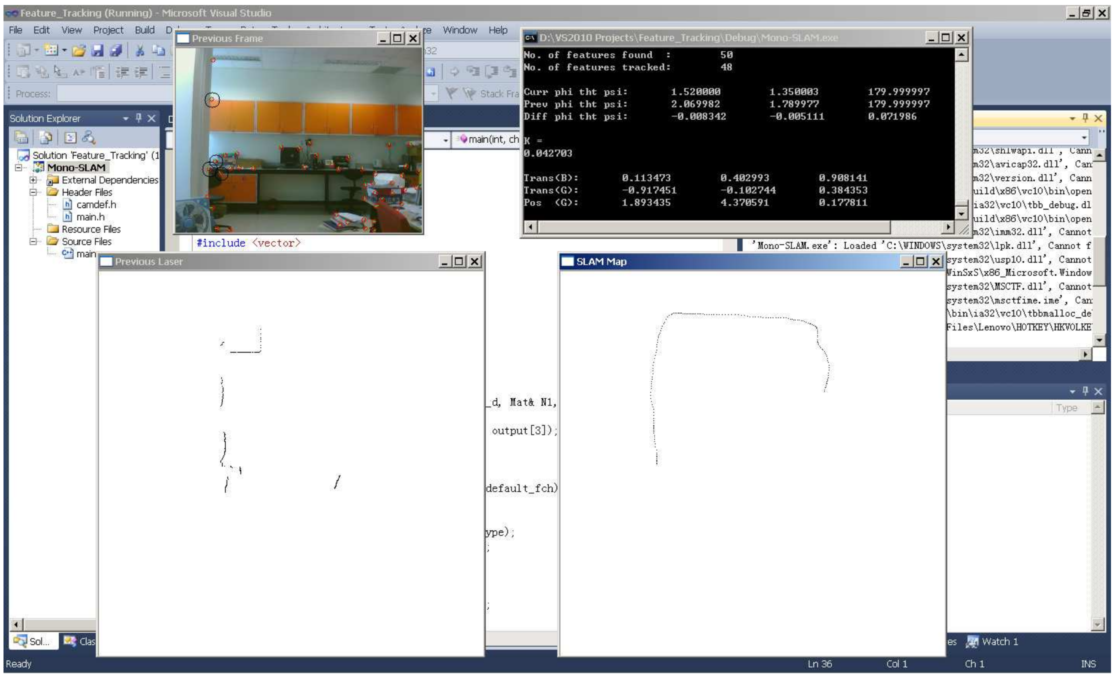
```
No. of features found  :        50
No. of features tracked:        45

Curr phi tht psi:       -0.410009       0.999983        179.999997
Prev phi tht psi:       0.430005        0.460028        179.999997
Diff phi tht psi:       -0.624876       0.088886        3.322900

K =
0.098637

Trans(B):       0.157972        -0.166968       0.973225
Trans(G):       -0.970143       -0.156773       -0.185054
Pos  (G):       3.409423        4.191455        0.074296
```

Figure 6.12: Localization result at $t = 21.1$ s



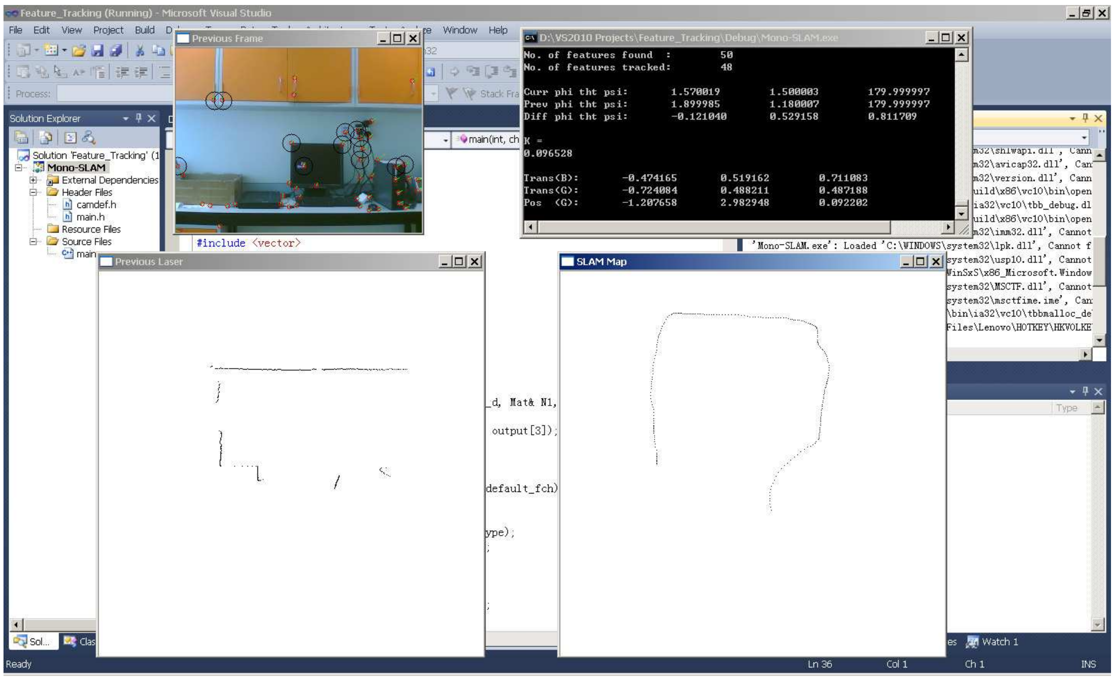
```
D:\VS2010 Projects\Feature_Tracking\Debug\Mono-SLAM.exe
No. of features found  :        50
No. of features tracked:        48

Curr phi tht psi:       1.520000        1.350003        179.999997
Prev phi tht psi:       2.069982        1.789977        179.999997
Diff phi tht psi:       -0.008342       -0.005111       0.071986

K =
0.042703

Trans(B):       0.113473        0.402993        0.908141
Trans(G):       -0.917451       -0.102744       0.384353
Pos  (G):       1.893435        4.370591        0.177811
```

Figure 6.13: Localization result at $t = 25.0$ s



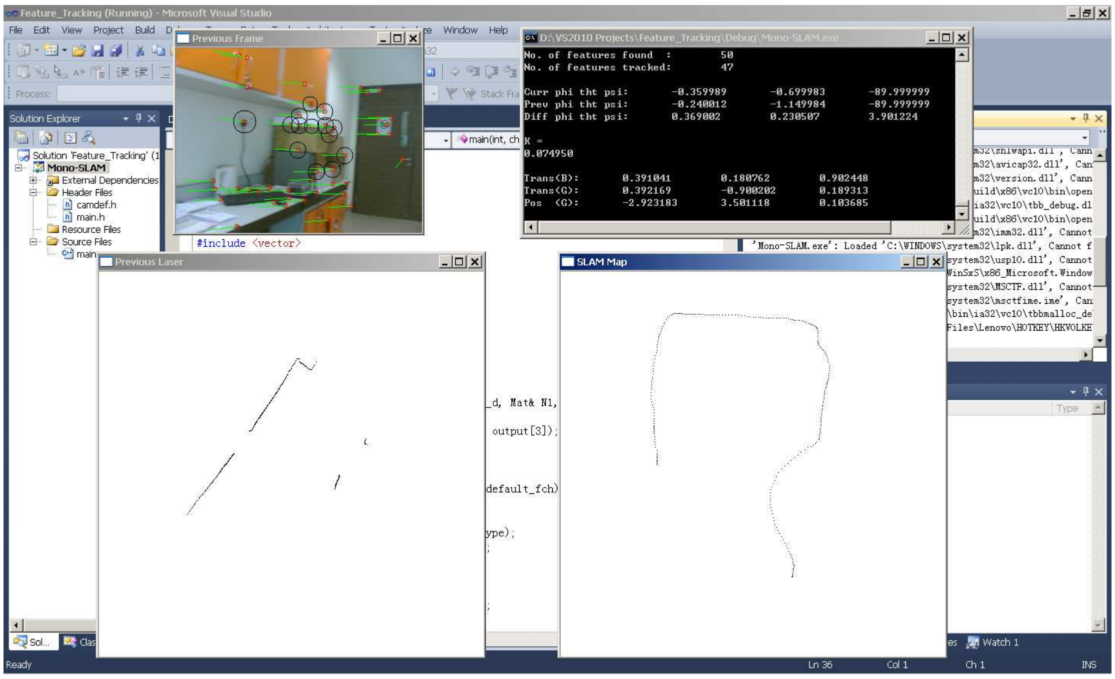
```
D:\VS2010 Projects\Feature_Tracking\Debug\Mono-SLAM.exe
No. of features found  :        50
No. of features tracked:        48

Curr phi tht psi:       1.570019        1.500003        179.999997
Prev phi tht psi:       1.899985        1.180007        179.999997
Diff phi tht psi:       -0.121040       0.529158        0.811709

K =
0.096528

Trans(B):       -0.474165       0.519162        0.711083
Trans(G):       -0.724084       0.488211        0.487188
Pos  (G):       -1.207658       2.982948        0.092202
```

Figure 6.14: Localization result at $t = 31.5$ s



Feature_Tracking (Running) - Microsoft Visual Studio
Previous Frame
D:\VS2010 Projects\Feature_Tracking\Debug\Mono-SLAM.exe
Previous Laser
SLAM Map
Solution Explorer

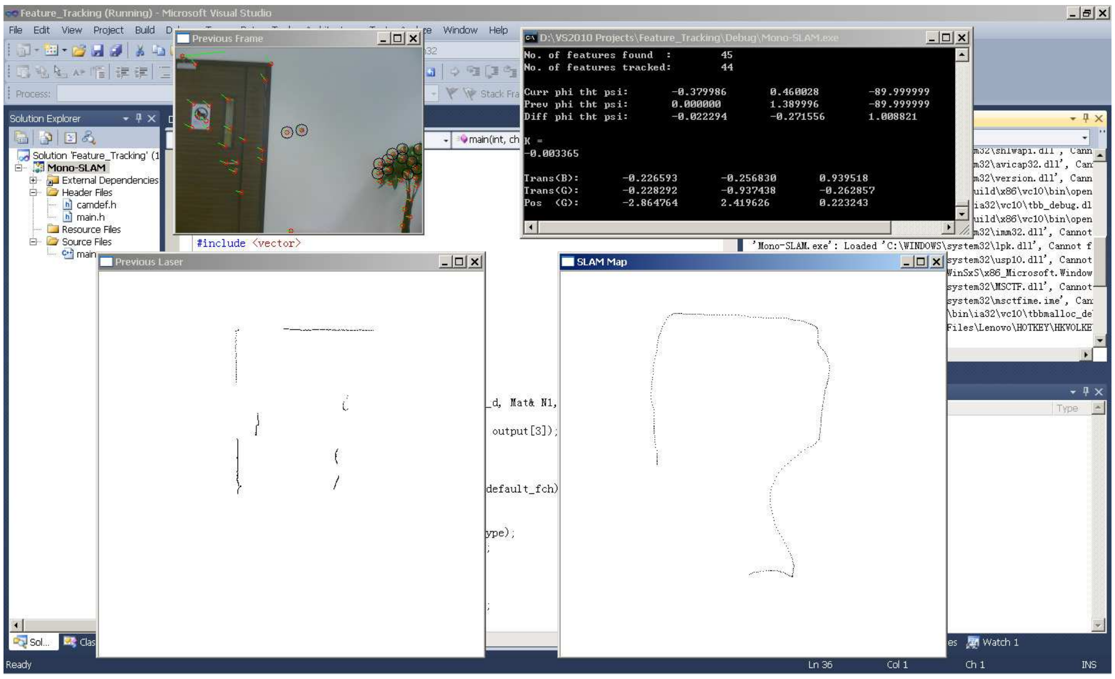
```
No. of features found  :        50
No. of features tracked:        47

Curr phi tht psi:       -0.359989       -0.699983       -89.999999
Prev phi tht psi:       -0.240012       -1.149984       -89.999999
Diff phi tht psi:       0.369002        0.230507        3.901224

K =
0.074950

Trans(B):       0.391041        0.180762        0.902448
Trans(G):       0.392169        -0.900202       0.189313
Pos  (G):       -2.923183       3.501118        0.103685
```

Figure 6.15: Localization result at $t = 34.8$ s



Feature_Tracking (Running) - Microsoft Visual Studio
Previous Frame
D:\VS2010 Projects\Feature_Tracking\Debug\Mono-SLAM.exe
Previous Laser
SLAM Map
Solution Explorer

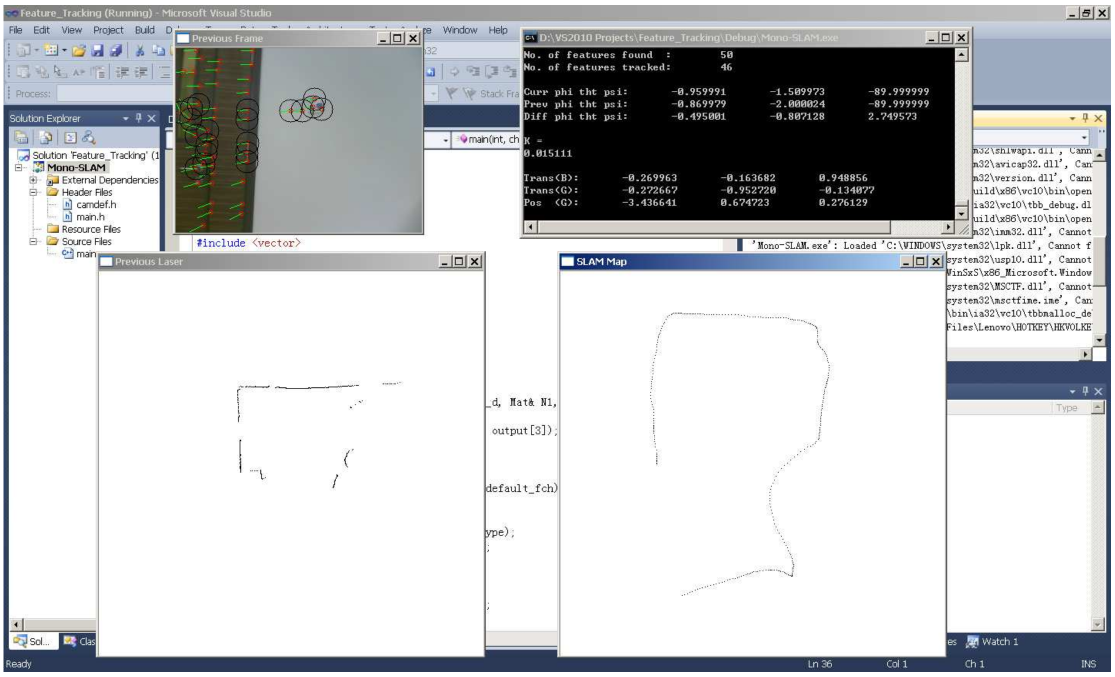
```
No. of features found  :        45
No. of features tracked:        44

Curr phi tht psi:       -0.379986       0.460028        -89.999999
Prev phi tht psi:       0.000000        1.389996        -89.999999
Diff phi tht psi:       -0.022294       -0.271556       1.008821

K =
-0.003365

Trans(B):       -0.226593       -0.256830       0.939518
Trans(G):       -0.228292       -0.937438       -0.262857
Pos  (G):       -2.864764       2.419626        0.223243
```

Figure 6.16: Localization result at $t = 38.1$ s


Feature_Tracking (Running) - Microsoft Visual Studio
Previous Frame
D:\VS2010 Projects\Feature_Tracking\Debug\Mono-SLAM.exe
Previous Laser
SLAM Map
Solution Explorer

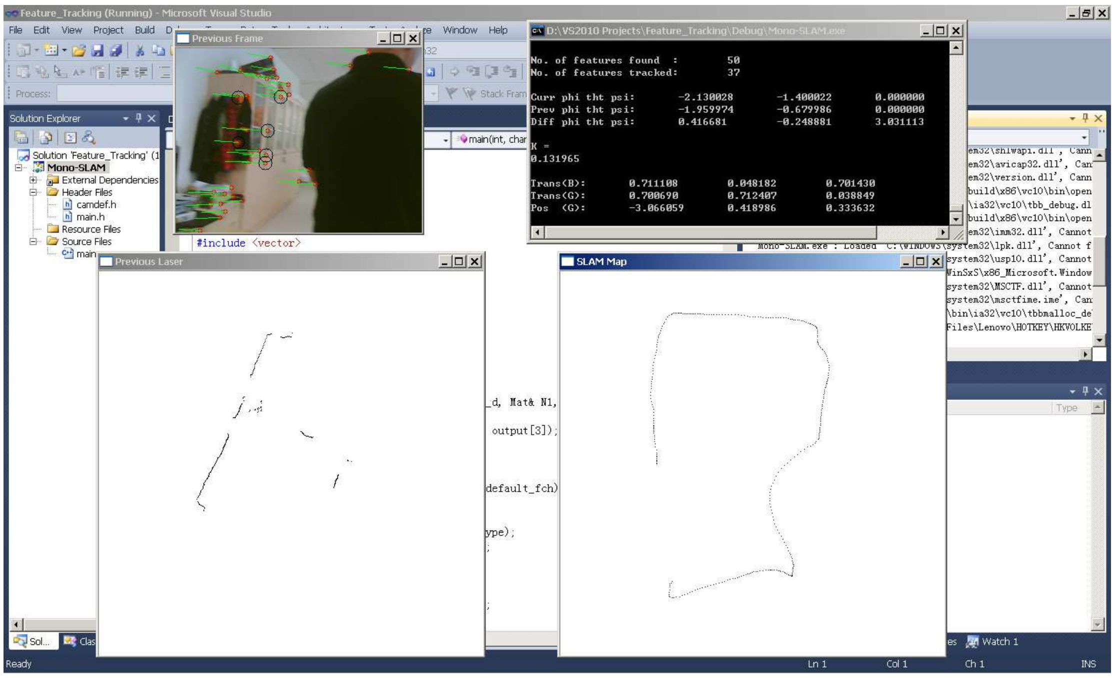
```
No. of features found  :        50
No. of features tracked:        46

Curr phi tht psi:       -0.959991       -1.509973       -89.999999
Prev phi tht psi:       -0.869979       -2.000024       -89.999999
Diff phi tht psi:       -0.495001       -0.807128       2.749573

K =
0.015111

Trans(B):       -0.269963       -0.163682       0.948856
Trans(G):       -0.272667       -0.952720       -0.134077
Pos  (G):       -3.436641       0.674723        0.276129
```

Figure 6.17: Localization result at $t = 40.8$ s


Feature_Tracking (Running) - Microsoft Visual Studio
Previous Frame
D:\VS2010 Projects\Feature_Tracking\Debug\Mono-SLAM.exe
Previous Laser
SLAM Map
Solution Explorer

```
No. of features found  :        50
No. of features tracked:        37

Curr phi tht psi:       -2.130028       -1.400022       0.000000
Prev phi tht psi:       -1.959974       -0.679986       0.000000
Diff phi tht psi:       0.416681        -0.248881       3.031113

K =
0.131965

Trans(B):       0.711108        0.048182        0.701430
Trans(G):       0.700690        0.712407        0.038849
Pos  (G):       -3.066059       0.418986        0.333632
```

Figure 6.18: Localization result at $t = 42.6$ s

6.1.3 3-D Motion Estimation via Optical Flow - Method 2

The aforementioned 3-D motion estimation method via a forward-facing camera has two practical issues. First, the correspondence between the laser range information and the depth of visual features may not be exactly matched if the relative pose between the laser scanner and the camera are not carefully calibrated. In fact, it is practically difficult to do so unless an innovative calibration method can be invented. In this work, the relative pose between the two sensors is roughly obtained by ruler measurement, which may has resulted in additional inaccuracy for the estimated 3D motion. Second, while a forward-facing camera works well to estimate the UAV velocity in the body-frame y and z directions, it may have poorer performance in the x direction due to numerical issues (x direction is where the camera is facing towards). By considering the fact that the z direction position and velocity information can be accurately acquired by other sensors, such as a barometer, a sonar or a laser scanner, it is more preferable to mount the camera facing vertically downwards so that the UAV x and y direction velocities can be better estimated. This configuration also takes great advantages from the assumption that the floor of an indoor environment is usually flat and vision algorithms involving feature points on the same plane can be largely simplified, thus easier to be implemented onboard.

Having said so, a neater method is proposed to estimate the UAV 3-D motion with the camera facing downward. As the indoor ground is usually flat, all observed visual features can be assumed to be co-planar in the 3-D space. Then an important concept called *homography* can be exploited. *Homography* is a term in computer vision to describe the linear position relationship between co-planar feature points projected onto two different 2-D images. Suppose a downward-facing camera on the UAV takes image of the ground during flight. Given two consecutive images taken at time t_1 and t_2, the corresponding visual features' pixel positions in the 1st and 2nd images are theoretically related by a 3 by 3 matrix $\mathbf{H}$, provided that the image scene all belong to the same plane [44]. $\mathbf{H}$ is called the *homography* matrix.

This *homography* matrix carries valuable information about the UAV motion from t_1 to t_2. If $\mathbf{R}$ and $\mathbf{T}$ are the inter-frame rotation and translation of the UAV from t_1 to t_2, $\mathbf{N}$ is the unit-length normal vector of the ground plane resolved in the camera frame at t_1, and d is the UAV altitude with respect to the ground plane, then the homography matrix $\mathbf{H}$ can be expresses as [32]:

$$\mathbf{H} = \mathbf{R} + \frac{1}{d}\mathbf{T}\mathbf{N}^{\mathrm{T}} \tag{6.18}$$

The decomposition of $\mathbf{H}$ into $\mathbf{R}$, $\mathbf{T}$ and $\mathbf{N}$ is doable but quite complicated and it usually results in large numerical errors in practice. Fortunately, $\mathbf{R}$, $\mathbf{N}$, d are known in our case since the IMU sensor can provide Euler angle estimation at every moment and the altitude of the camera with respect to the ground plane can be obtained via barometer or range sensors. If we have UAV attitude angles ϕ_1, θ_1, ψ_1 at t_1 and ϕ_2, θ_2, ψ_2 at t_2, then

$$\mathbf{N} = \begin{bmatrix} -\sin\theta_1 \\ \sin\phi_1\cos\theta_1 \\ \cos\phi_1\cos\theta_1 \end{bmatrix} \tag{6.19}$$

and

$$\mathbf{R} = \mathbf{R}_{\mathrm{b/n}}(t_2)\mathbf{R}_{\mathrm{n/b}}(t1) \tag{6.20}$$

where $\mathbf{R}_{\mathrm{b/n}}$ is the rotational matrix to convert 3-D points from the inertia frame to the UAV body frame and $\mathbf{R}_{\mathrm{n/b}}$ is vice versa and they are transpose of each other. Hence, with $\mathbf{H}$, $\mathbf{R}$, $\mathbf{d}$ and $\mathbf{N}$ known, the translational motion, $\mathbf{T}$ can be calculated as:

$$\mathbf{T} = d(\mathbf{H} - \mathbf{R})\mathbf{N} \tag{6.21}$$

As mentioned previously, indoor UAV height measurement can be obtained via various choices of sensors. Here we briefly discuss one of the methods which is based on the barometer. When a reference pressure P_0 (the pressure measured right before taking off) is selected, then the z direction position of the UAV at any moment after can be calculated by the following formula:

$$z_g = -44307\left[1 - \left(\frac{P}{P_0}\right)^{0.1902}\right], \tag{6.22}$$

where P is the current pressure measured by the barometer in Pascal. Although more accurate altitude measurement can be obtained by using a second laser scanner, which will be discussed in Chapter 9, the implementation result presented at the end of this chapter is based on the barometer-only setup. If substituted by more accurate height measurement, the result is expected to be even better.

Although visual odometry can be obtained via the above two methods, their raw estimation results have problems including low update rate, relatively large noise, and more severely, outliers. Fusion with IMU acceleration will largely solve the problems and provide a smoother velocity and position estimation which is needed for appropriate flight control implementation. To fuse these information, Kalman filter is one of the best choices. The next section will discuss how to apply Kalman filter to fuse the visual odometry result from Method 2 together with IMU measurements. The same concept can be applied to visual odometry from Method 1 or other kind of velocity or position estimations.

6.1.4 Fusion with IMU Data via Kalman Filter

The Kalman filter framework describes a discrete-time linear system as follows,

$$\begin{aligned}\mathbf{x}(k+1) &= \mathbf{A}\mathbf{x}(k)+\mathbf{B}(\mathbf{u}(k)+\mathbf{w}(k)),\\ \mathbf{y}(k) &= \mathbf{C}\mathbf{x}(k)+\mathbf{v}(k),\end{aligned} \tag{6.23}$$

where $\mathbf{x}$, $\mathbf{u}$ and $\mathbf{y}$ are the state, input and measurement vectors respectively. $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ are system matrices with appropriate dimensions. $\mathbf{w}$ and $\mathbf{v}$ are input and measurement noises, which are assumed to be Gaussian with zero means and covariance matrices Q and R respectively. The main objective of Kalman filter is to estimate $\hat{\mathbf{x}}(k|k)$ at the time step k with the measurement $\mathbf{y}(k)$, input $\mathbf{u}(k-1)$ and the previously estimated state $\hat{\mathbf{x}}(k|k-1)$. If system 6.23 is observable, then the statistically optimal estimator is given as follows,

Time Update:

$$\hat{\mathbf{x}}(k|k-1) = \mathbf{A}\hat{\mathbf{x}}(k-1)+\mathbf{B}\mathbf{u}(k-1), \tag{6.24}$$

$$\mathbf{P}(k|k-1) = \mathbf{A}\mathbf{P}(k-1)\mathbf{A}^{\mathrm{T}}+\mathbf{B}\mathbf{Q}\mathbf{B}^{\mathrm{T}}, \tag{6.25}$$

Measurement Update:

$$\mathbf{H}(k) = \mathbf{P}(k|k-1)\mathbf{C}^{\mathrm{T}}(\mathbf{C}\mathbf{P}(k|k-1)\mathbf{C}^{\mathrm{T}}+\mathbf{R})^{-1}, \tag{6.26}$$

$$\hat{\mathbf{x}}(k) = \hat{\mathbf{x}}(k|k-1)+\mathbf{H}(k)(\mathbf{y}(k)-\mathbf{C}\hat{\mathbf{x}}(k|k-1), \tag{6.27}$$

$$\mathbf{P}(k) = (\mathbf{I}-\mathbf{H}(k)\mathbf{C})\mathbf{P}(k|k-1), \tag{6.28}$$

where $\mathbf{H}(k|k-1)$ is a feedback gain matrix and $\mathbf{P}(k|k-1)$ is the covariance of the state estimation error that is defined as

$$
\begin{aligned}
\mathbf{P}(k|k-1) &= E[\mathbf{x}(k) - \hat{\mathbf{x}}(k|k-1)][\mathbf{x}(k) - \hat{\mathbf{x}}(k|k-1)]^{\mathrm{T}}, && (6.29)\\
\mathbf{R}(k) &= E\{\mathbf{v}(k)\mathbf{v}^{\mathrm{T}}(k)\}, && (6.30)\\
\mathbf{Q}(k) &= E\{\mathbf{w}(k)\mathbf{w}^{\mathrm{T}}(k)\}. && (6.31)
\end{aligned}
$$

Here, $E\{*\}$ denotes expectation.

To apply Kalman filter in this indoor UAV state estimation problem, the motion model and the measurement model need to be first defined, namely the state vector $\mathbf{x}$, input vector $\mathbf{u}$, measurement vector $\mathbf{y}$ and the $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ system matrices. For the motion model, the simple point mass kinematics model can be used, where in NED or ground frame, position can be integrated by velocity and velocity can be integrated by acceleration. Hence,

$$
\begin{aligned}
\mathbf{x} &= \begin{bmatrix} x_g & y_g & z_g & u_g & v_g & w_g \end{bmatrix}^{\mathrm{T}},\\
\mathbf{u} &= \begin{bmatrix} ax_g & ay_g & az_g \end{bmatrix}^{\mathrm{T}},\\
\mathbf{y} &= \begin{bmatrix} z_g & u_g & v_g \end{bmatrix}^{\mathrm{T}},
\end{aligned}
$$

where x_g, y_g, z_g are the NED position coordinates, u_g, v_g, w_g are the NED velocity elements, and ax_g, ay_g, az_g are the NED acceleration elements. Furthermore,

$$
\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 & 0.02 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0.02 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0.02 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix},
$$

$$\mathbf{B} = \begin{bmatrix} 0.005 & 0 & 0 \\ 0 & 0.005 & 0 \\ 0 & 0 & 0.005 \\ 0.02 & 0 & 0 \\ 0 & 0.02 & 0 \\ 0 & 0 & 0.02 \end{bmatrix},$$

$$\mathbf{C} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}.$$

While the above have been numerically defined, the input noise matrix $\mathbf{Q}$ and the measurement noise matrix $\mathbf{R}$, being both positive definite and diagonal, can be selected by logging real flight test data; the diagonal elements in $\mathbf{Q}$ represent the acceleration measurement noises and the diagonal elements in $\mathbf{R}$ represent the noises of height measurement and velocity estimation from visual odometry.

In theory, the above method can be applied to any indoor UAV platform equipped with an IMU sensor and a camera. However, the performance depends on the quality of sensors and the computation power of the onboard processors. Unfortunately, the coaxial platform, being payload restricted, is carrying an ArduIMU with poor measurement accuracy and its vision computer is a 720 MHz Gumstix Overo Fire, which can merely execute a real-time (10 Hz) optical flow algorithm at about 40×30 pixel resolution. At such low pixel resolution, the velocity measurements are too noisy to be used. As such, the aforementioned algorithms are implemented on the quadrotor platform instead. The quadrotor platform is equipped with a high-performance IG-500N IMU sensor and its vision system constitutes a Firefly FMVU-03MTM/C-CS USB camera and a relatively powerful fit-PC2 computer. The vision algorithm implemented onboard runs at 10 Hz based on a 320×240 resolution image. The control computer keeps sending UAV roll, pitch, yaw, altitude to the vision computer and the vision computer, through optical flow computation, sends back the computed linear velocity to the control computer. A Kalman filter runs in the control computer by taking in all the information from IMU's acceleration, vision's velocity and pressure sensor's altitude.

Experiments have been carried out to verify the performance of this state estimation method. First, the UAV is manually flied with a sequence of actions including taking-off, flying forward,

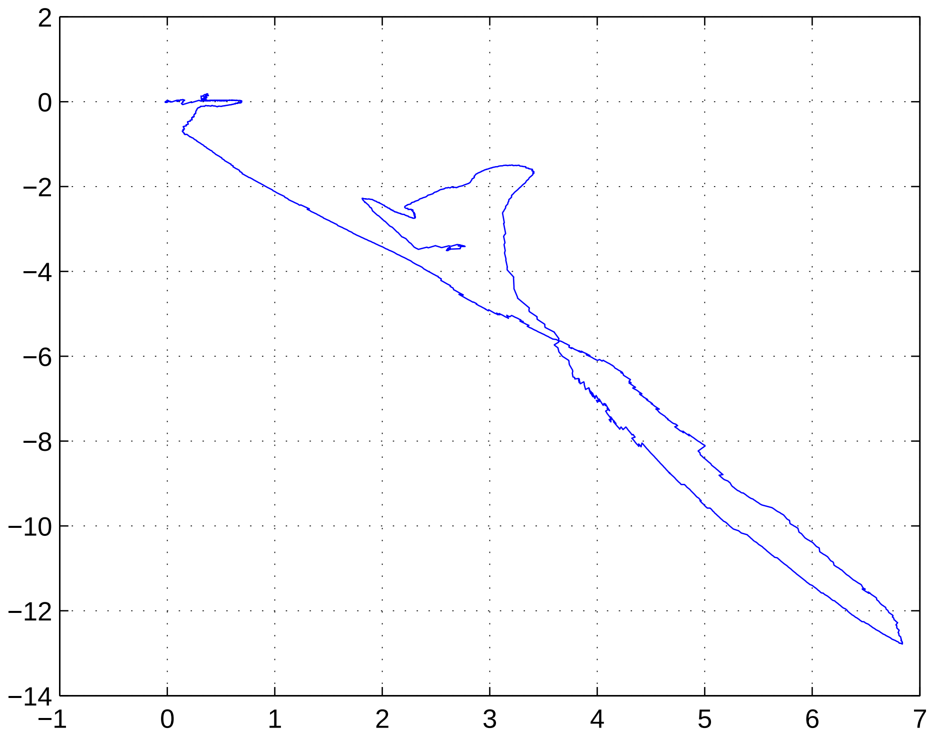


Figure 6.19: The estimated NED-frame x, y-axis positions

flying backward and landing. The furthest point to the take-off point is 15 meters away in distance. Fig. 6.19 shows the estimated 2-D position of the UAV, which illustrates the forward and backward paths almost coincide. For the same experiment, Fig. 6.20 and Fig. 6.21 show the estimated 2-D velocities against time. Although there is no ground truth to be compared with, the signals are all smooth and reasonable.

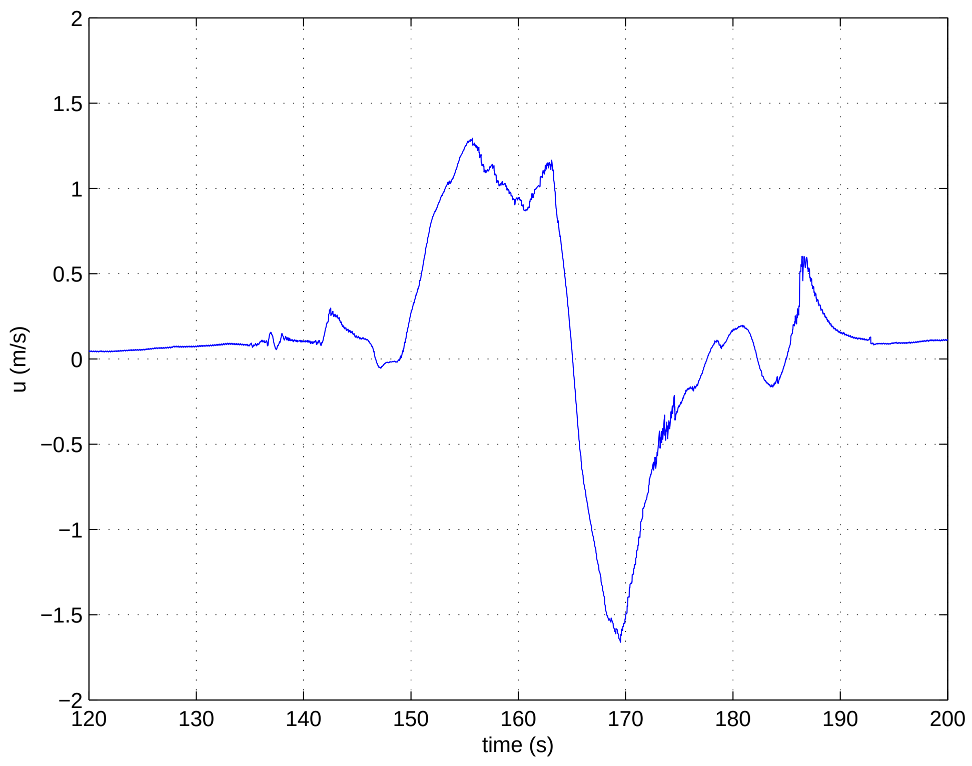


Figure 6.20: The estimated body-frame x-axis velocity

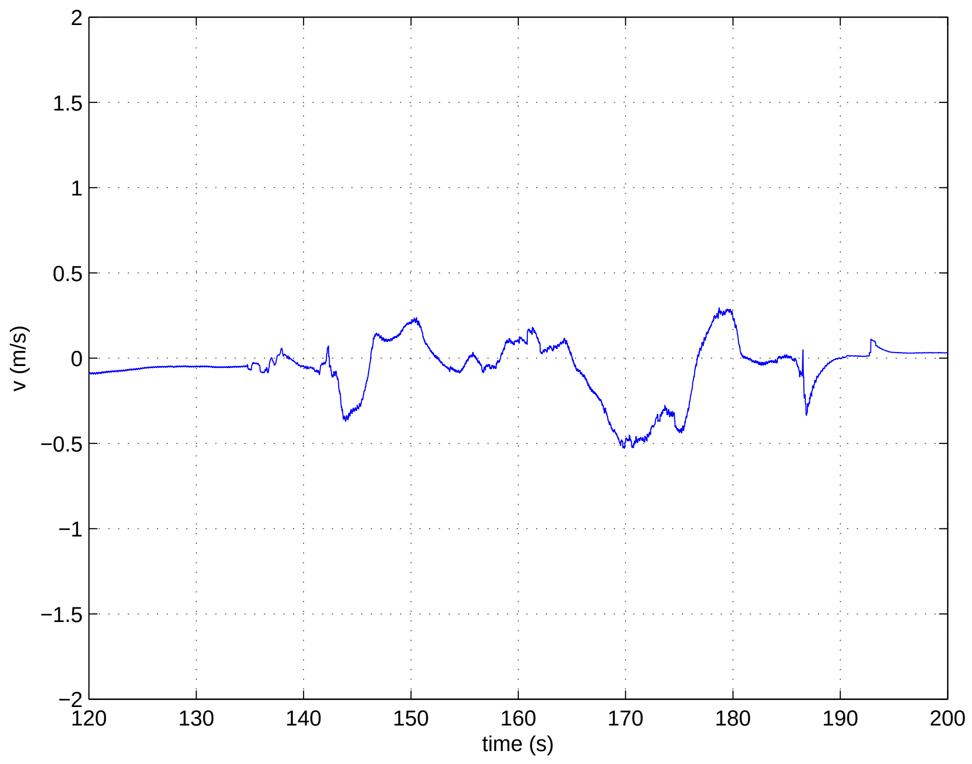


Figure 6.21: The estimated body-frame y-axis velocity

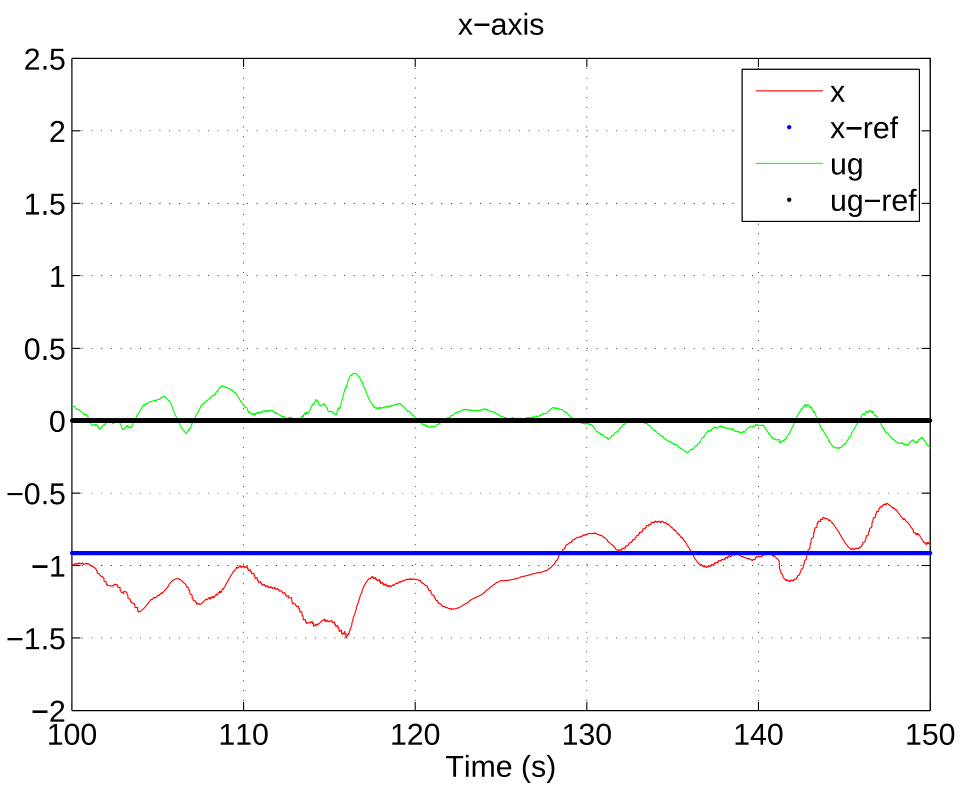


Figure 6.22: The x-axis references and actual values in m or m/s

To further confirm the performance of the proposed visual odometry and data fusion method, an autonomous hover test via the estimated position and velocity feedback is carried out. For the whole time duration, from $t = 100s$ to $t = 150s$, the UAV is commanded to hold its position based on the estimated position and velocity. A few intentional disturbances are given to the UAV to see how it recovers (see Figs. 6.22–6.25). The outer-loop position and velocity tracking performance is illustrated in Figs. 6.26–6.29. According to human observation from various angles, the position drift is almost zero and the overall flight performance is very stable.

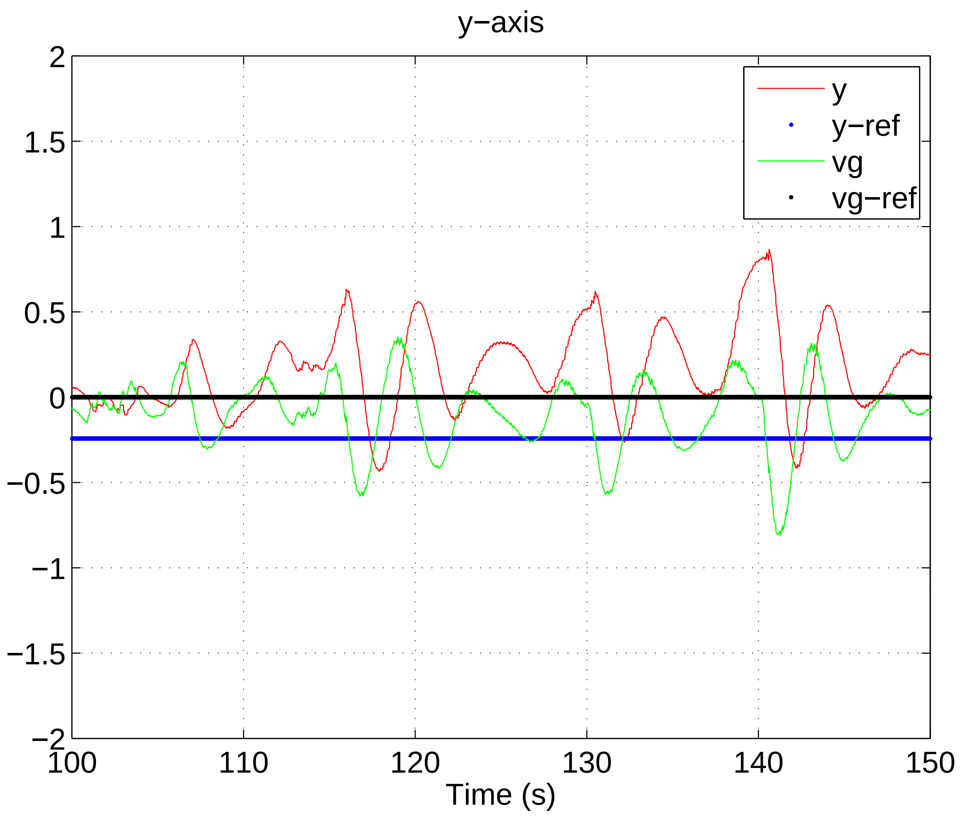


Figure 6.23: The y-axis references and actual values in m or m/s

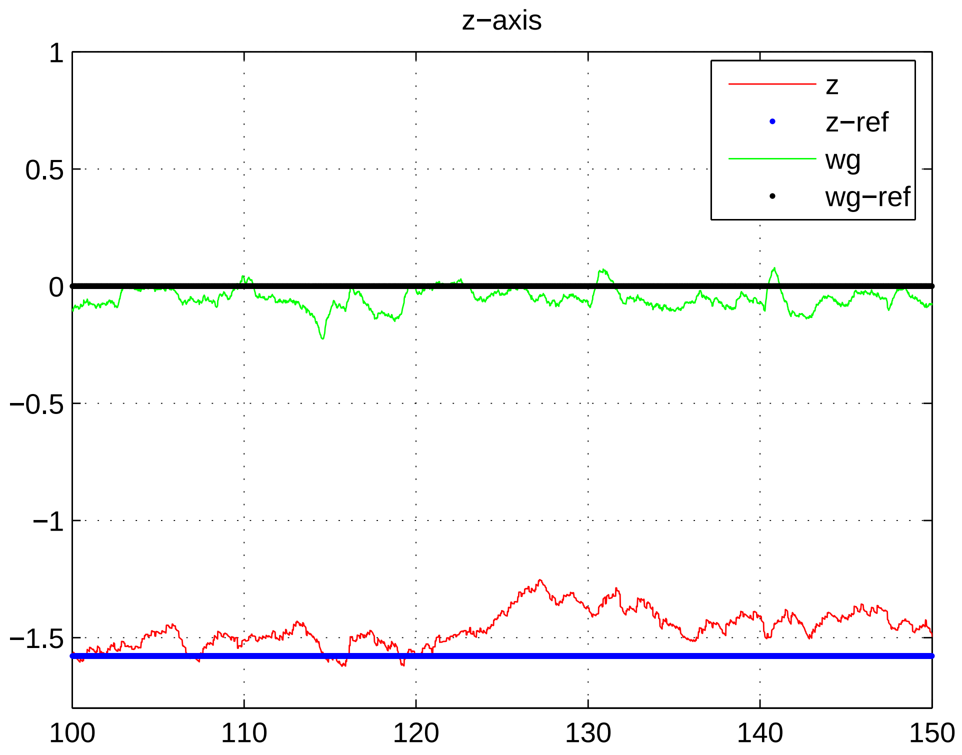


Figure 6.24: The z-axis references and actual values in m or m/s

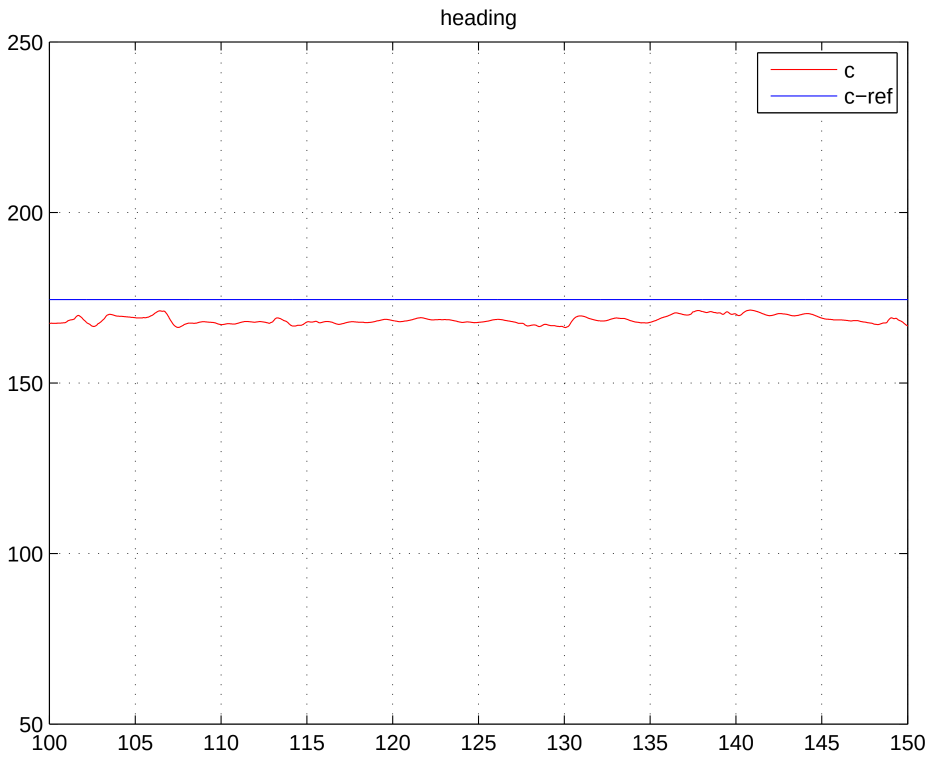


Figure 6.25: The yaw references and actual values in deg or deg/s

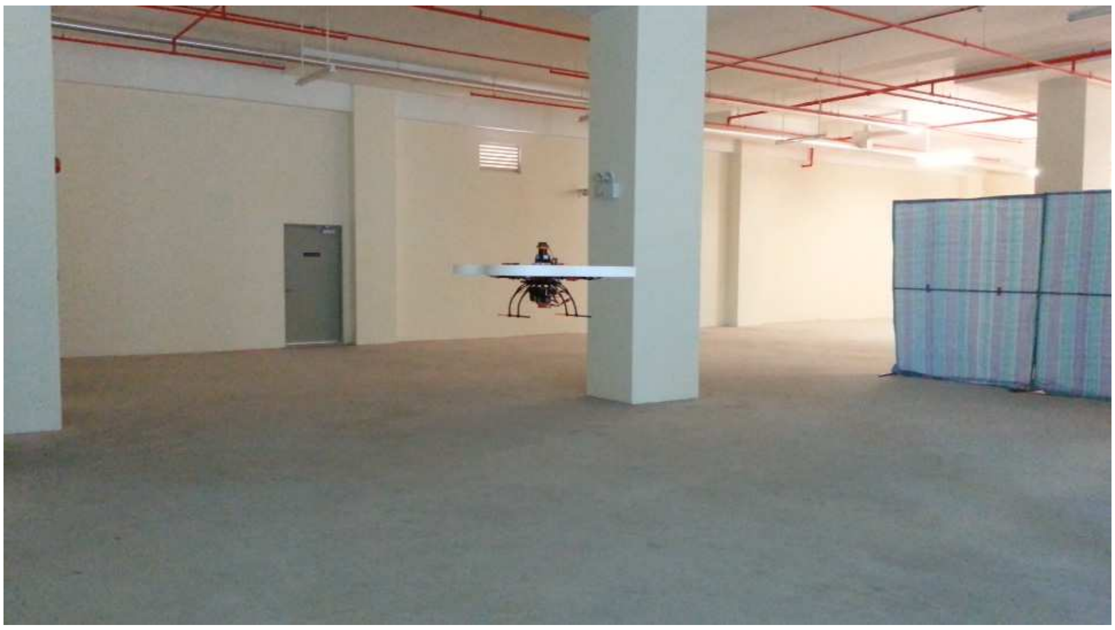


Figure 6.26: Quadrotor position hold via optical flow (Moment 1)

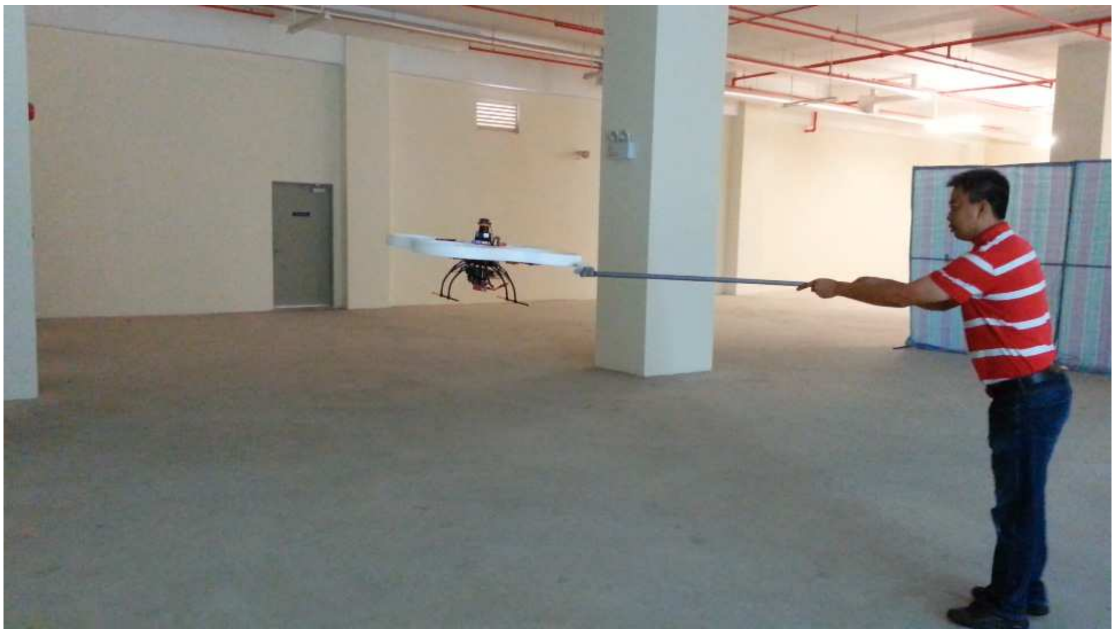


Figure 6.27: Quadrotor position hold via optical flow (Moment 2)

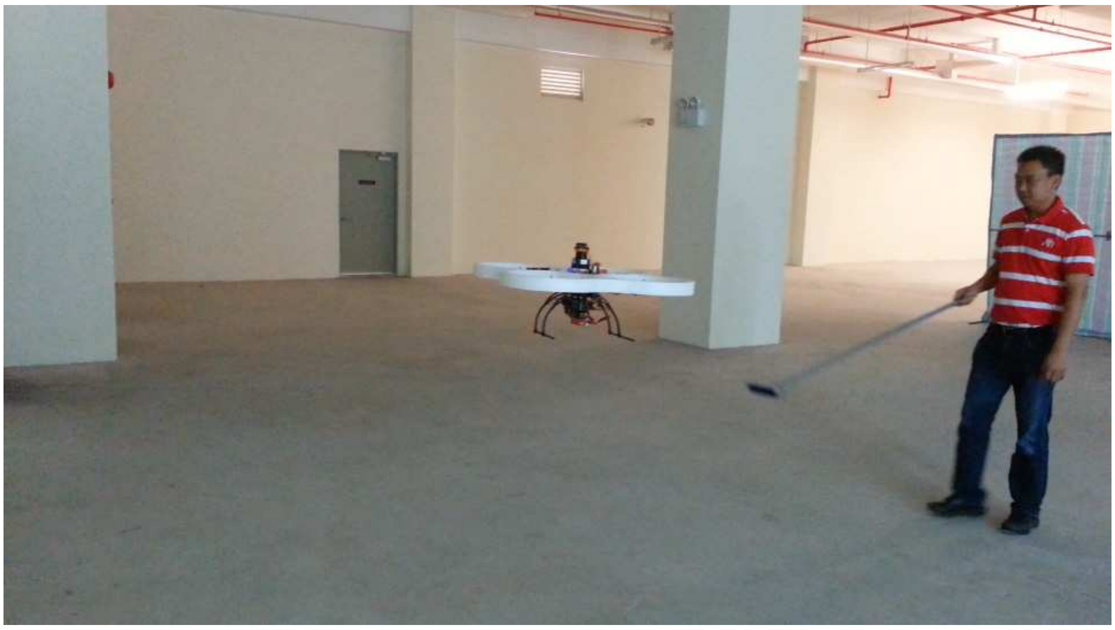


Figure 6.28: Quadrotor position hold via optical flow (Moment 3)

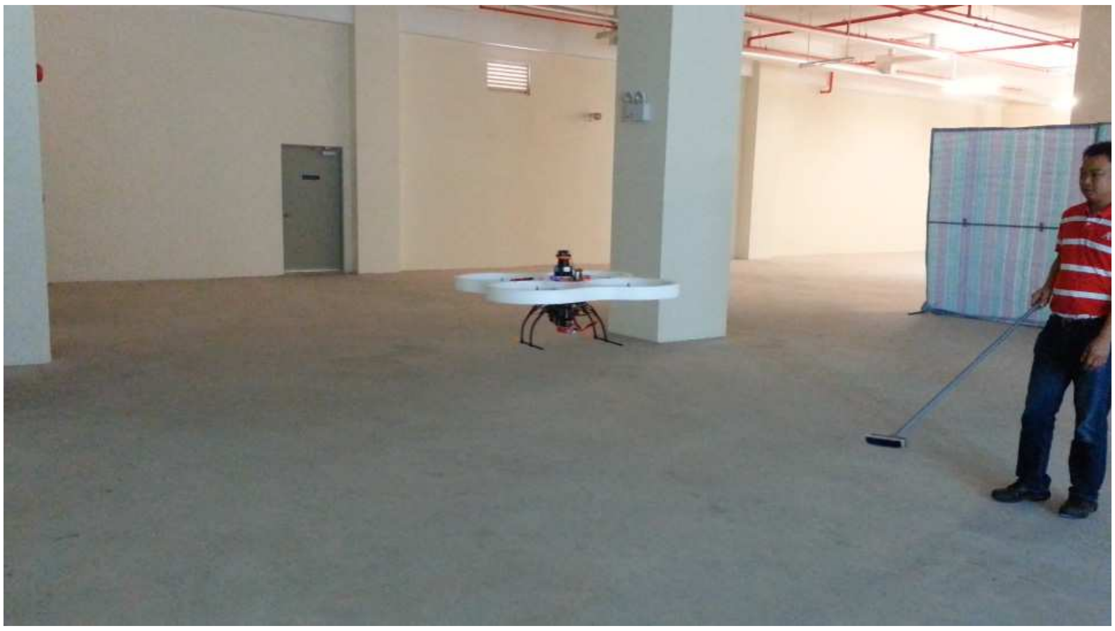


Figure 6.29: Quadrotor position hold via optical flow (Moment 4)

6.2 Laser Odometry

Similar to the visual odometry case, algorithms based on measurements from laser scanner can also be developed to provide UAV inter-frame motion estimation by comparing consecutive laser scans. In this chapter, the iterative closest point algorithm (ICP) will be explored and used as another odometry information for the indoor UAV.

The ICP algorithm is a method of fitting points in a target frame to points in a control frame by rigid transformation (rotation and translation) [3]. The ultimate goal of the algorithm is to minimize the sum of squared errors with respect to the target points and their corresponding closest control points. An initial coarse estimate of motion is needed to align the target points and control points roughly. The basic component of the algorithm calculates the smallest distance between each point in the target image to a point in the control image. These calculated points are then used to form a translational vector and a rotational matrix that is applied over all points in the target image to adjust them towards the control image. This processes is repeated numerous times, thus an iterative algorithm, with the end result being a target image with points that are within a specified squared error distance of their corresponding points in the control image.

The ICP algorithm is a very appropriate technique that can be used on laser scanner data. As the laser scanner acquires indoor object points in a 2-D plane, consecutive scans can be compared to compute an accurate 2-D motion of the UAV. By accumulating this 2-D motion

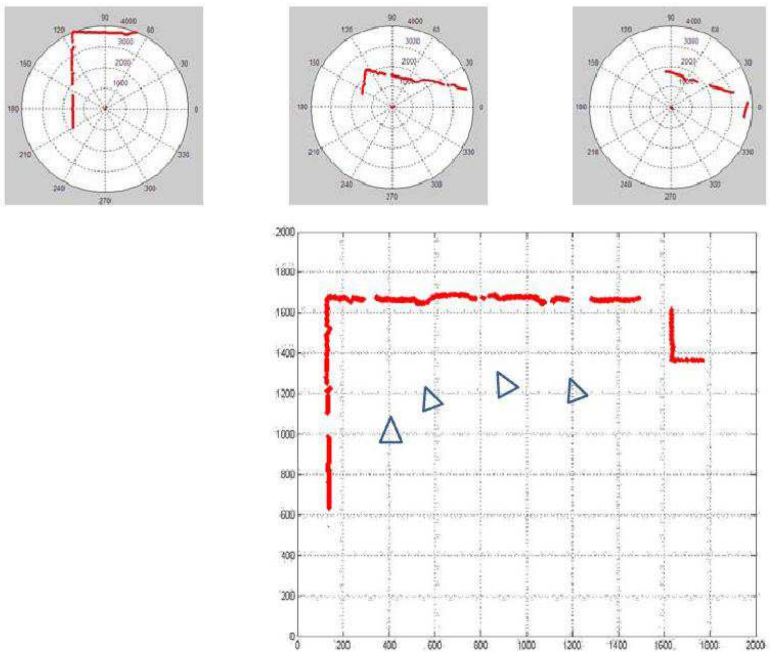


Figure 6.30: Using ICP for SLAM

frame by frame, the 2-D position of the UAV with respect to its initial position can be obtained also. On the other hand, if we assume that the estimated position is correct, then all frames of the laser-scanned points can be transformed back to the initial frame, and the map of the indoor environment can be generated. The basic idea is illustrated in Fig. 6.30. The four sub-plots in the upper portion of the figure represent four frames of laser scanner data. By the ICP algorithm, the position and orientation of the UAV can be estimated for each instance, and the map of the indoor environment can be gradually generated. Again, the laser scanner based ICP algorithm is odometry (integration) based. Hence, the estimated position will drift in the long run and the generated map will be inconsistent thereafter.

6.2.1 Assumptions and Issues

There are several assumptions and issues need to be stated before the algorithm itself is explained. Some of the issues are unavoidable in general indoor navigation cases and some of them can be caused by hardware limitations of sensors. However, they usually result in minor inaccuracies in the final result or are subject to special conditions which rarely happen in a normal indoor environment.

Unique 2-D Plane Assumption

To apply the scan matching algorithm, the measurements from different frames of laser scan are assumed to be in the same 2-D plane, which means the UAV is at a constant height with zero roll and pitch angles. In practice, there is always minor error in the UAV height control and small deflections in the roll and pitch motion, even when the UAV is at a near-hover condition. To correct the roll and pitch offsets, each scanned point needs to be compensated individually according to their different radial directions and the roll-pitch angles at that particular moment. However, the roll and pitch measurements can be noisy if a low-cost IMU is used and the compensation may not be very accurate. In addition, it must be further assumed that all scanned objects are vertically homogeneous, such as walls, poles and other vertically structured objects. Fortunately, it is usually the case in a man-made indoor environment.

Overlap Assumption

For the ICP algorithm to function properly, there must be enough overlap between the two consecutive laser scans. When the algorithm tries to match all the target points to the control points, it is best if all the matching pair physically exists. This assumption can be valid if the UAV cruising speed does not exceed certain threshold and the scanning frequency of the laser scanner is fast enough. For the case of UAV indoor navigation, since the UAV cruising speed can be controlled and update frequency of the Hokuyo laser scanner is fast enough (10 Hz or above), this overlap assumption can be always met.

Limited Range Issue

Two laser scanners with different range limits can be used for this research work. The 30 m UTM-30LX should have sufficient measurement range for all kinds of indoor environments, while the 4 m URG-04LX may not be sufficient for a few cases. In order to have sufficient number of scanned points, there have to be natural objects like walls, pillars in the range of laser sensor while the UAV flies. Hence, if the 4 m laser scanner is used, this laser-scanner-based ICP is only applicable to small classrooms or lab rooms, but not for the large-scale halls or auditoriums. However, the 30 m laser scanner on the quadrotor UAV literally has no such issue.

Degenerate Cases

Except for the cases when all object distances exceed the measurement range, the algorithm will also fail when the UAV surroundings are too simple. For example, only a single straight line is detected by the laser scanner. This happens when the UAV flies forward along a wall on its left side but the front wall has not come into range yet. For such cases, the two consecutive laser scanner data will be nearly the same and the forward motion of the UAV cannot be distinguished.

Initial Estimation

It is important that an initial guess for the transformation that maps the target points to the control points is known so that ICP can applied with better performance. There are two approaches if a high-end laser scanner is used. One is to assume that every pair of consecutive scans are already close enough. So the initial guess is zero rotation and zero translation. By executing the algorithm with large number of iterations, the solution hopefully converges to the global optimum. The second approach is to obtain a rough estimation of the motion between two frames by other sensors, such as the rotational motion from the IMU sensor and translational motion from the visual odometry. The first approach may consume greater computational power because of more iterations while the second approach is prone to measurement noises from the other sensor source and sensor synchronization needs to be ensured. As of writing, only the first approach has been successfully implemented.

6.2.2 The ICP Algorithm

Fig. 6.31 has shown the procedures of a standard ICP algorithm. The algorithm starts by initializing the coarse transformation (alignment) and an infinitely large error. Then it calculates the point correspondences between the target frame and the control frame using the nearest neighbor rule. Based on the obtained correspondence, the optimal transformation can be found and applied to the target frame. If the alignment error is smaller than a pre-defined threshold, then the algorithm stops. Else, it iterates back to the *Calculate correspondence* step, followed by re-calculation of the alignment. While other steps are straight forward, the *Calculate correspondence* and *Calculate alignment* steps deserve extra explanation as they rely on vigorous mathematical derivations. The following contents will thus explain these two steps in detail and the formulations are based on a general 3-D point cloud case. To apply it to the case of 2-D

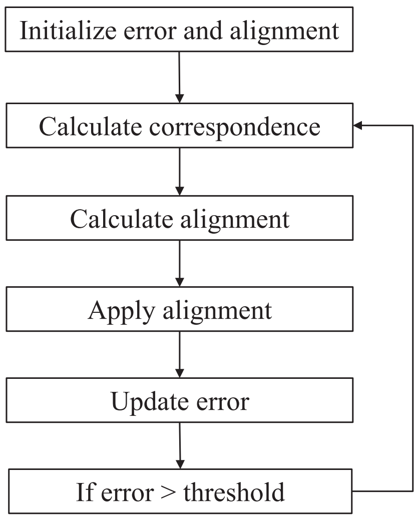


Figure 6.31: Procedures of the ICP algorithm

laser scanner, the third dimension of all input points can be set to zero and only the x-y-plane translation and rotation is to be extracted from the result.

Correspondence Calculation

This step aims to find the point matching pairs between the target frame and the control frame based on the nearest neighbor rule. The distance between two points in the 3-D space can be calculated as

$$d_{i,j} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2}, \tag{6.32}$$

where (x_i, y_i, z_i) is a point in the target frame and (x_j, y_j, z_j) is a point in the control frame. This calculation needs to be performed for each valid point in the target frame against all points in the control frame. Among all points in the control frame, the one produces the smallest value for this calculation is stored as the closest point for the current point in the target model, and then an association $a_{i,j}$ is created. However, a naive implementation of this step will be computationally intensive despite its logic simplicity. To make it less time consuming, data structures favoring efficient searching, such as the K-D tree [11], can be used.

Alignment Calculation

Let $P = \{p_i | i = 1, 2, ..., n\}$ and $Q = \{q_i | i = 1, 2, ..., n\}$ be the matched target points and the corresponding control points respectively. It is desired to find a rigid body transformation that optimally aligns the two sets of points in the least squares sense, i.e., to seek a rotational matrix R and a translational vector t such that

$$(R,t) = \arg\min \sum_{i=1}^{n} \omega_i ||(Rp_i + t) - q_i||^2, \tag{6.33}$$

where $w_i > 0$ is a weighting factor for each point pair.

Computing Translational Vector

Assume R is fixed and denote $F(t) = \sum_{i=1}^{n} \omega_i ||(Rp_i + t) - q_i||^2$. The optimal translation can be found by taking the derivative of F w.r.t t and searching for its roots:

$$\begin{aligned} 0 &= \frac{\partial F}{\partial t} = \sum_{i=1}^{n} 2w_i(Rp_i + t - q_i) \\ &= 2t(\sum_{i=1}^{n} w_i) + 2R(\sum_{i=1}^{n} w_i p_i) - 2\sum_{i=1}^{n} w_i q_i. \end{aligned} \tag{6.34}$$

If we define

$$\bar{p} = \frac{\sum_{i=1}^{n} w_i p_i}{\sum_{i=1}^{n} w_i}, \;\; \bar{q} = \frac{\sum_{i=1}^{n} w_i q_i}{\sum_{i=1}^{n} w_i}, \tag{6.35}$$

by rearranging the terms in (6.34), one can get

$$t = \bar{q} - R\bar{p} \tag{6.36}$$

In physical meanings, the optimal translation t maps the transformed weighted centroid of P to the weighted centroid of Q. Now substitute the optimal t back into the objective function:

$$\begin{aligned} \sum_{i=1}^{n} \omega_i ||(Rp_i + t) - q_i||^2 &= \sum_{i=1}^{n} w_i ||Rp_i + \bar{q} - R\bar{p} - q_i||^2 \\ &= \sum_{i=1}^{n} w_i ||R(p_i - \bar{p}) - (q_i - \bar{q})||^2 \end{aligned} \tag{6.37}$$

If we redefine the terms as follows:

$$x_i := p_i - \bar{p}\,, \;\; y_i := q_i - \bar{q} \tag{6.38}$$

then it is equivalent to seek for the optimal rotational matrix R such that

$$R = \arg\min \sum_{i=1}^{n} w_i ||Rx_i - y_i||^2 \tag{6.39}$$

Computing Rotational Matrix

Expand the expression of a part of the objective function derived above:

$$\begin{aligned}
||Rx_i - y_i||^2 &= (Rx_i - y_i)^{\mathrm{T}}(Rx_i - y_i) \\
&= (x_i^{\mathrm{T}} R^{\mathrm{T}} - y_i^{\mathrm{T}})(Rx_i - y_i) \\
&= x_i^{\mathrm{T}} R^{\mathrm{T}} R x_i - y_i^{\mathrm{T}} R x_i - x_i^{\mathrm{T}} R^{\mathrm{T}} y_i + y_i^{\mathrm{T}} y_i \\
&= x_i^{\mathrm{T}} - y_i^{\mathrm{T}} R x_i - x_i^{\mathrm{T}} R^{\mathrm{T}} y_i + y_i^{\mathrm{T}} y_i \\
&= x_i^{\mathrm{T}} - 2 y_i^{\mathrm{T}} R x_i + y_i^{\mathrm{T}} y_i
\end{aligned} \tag{6.40}$$

Throwing away the two terms not related to R, minimizing the objective expression is equivalent to maximizing another simplified expression:

$$\arg\min(-2 \sum_{i=1}^{n} w_i y_i^{\mathrm{T}} R x_i) = \arg\max \sum_{i=1}^{n} w_i y_i^{\mathrm{T}} R x_i \tag{6.41}$$

Note that

$$\sum_{i=1}^{n} w_i y_i^{\mathrm{T}} R x_i = \mathrm{tr}(W Y^{\mathrm{T}} R X) \tag{6.42}$$

where $W = \mathrm{diag}(w_1, ..., w_n)$ is an $n \times n$ diagonal matrix; Y is a $3 \times n$ matrix with y_i as its columns and X is a $3 \times n$ matrix with x_i as its columns. Therefore, a rotational matrix R that maximizes tr$(WY^{\mathrm{T}}Rx)$ needs to be found. By using the property of tr(AB) = tr(BA), we have

$$\mathrm{tr}(W Y^{\mathrm{T}} R X) = \mathrm{tr}(R X W Y^{\mathrm{T}}). \tag{6.43}$$

Let $S = XWY^{\mathrm{T}}$ and take singular value decomposition (SVD) of S:

$$S = XWY^{\mathrm{T}} = U \Sigma V^{\mathrm{T}}. \tag{6.44}$$

Then,

$$\text{tr}(RXWY^{\text{T}}) = \text{tr}(RS) = \text{tr}(RU\Sigma V^{\text{T}}) = \text{tr}(\Sigma V^{\text{T}}RU) \tag{6.45}$$

Note that V, R and U are all orthogonal matrices, so $M = V^{\text{T}}RU$ is also an orthogonal matrix. This means all entries m_{ij} of M are smaller or equal to 1 in magnitude. Also note that Σ is a diagonal matrix with non-negative values $\sigma_1, \sigma_2, \sigma_3 \geqslant 0$ on the diagonal. So:

$$\text{tr}(\Sigma M) = \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & \sigma_3 \end{bmatrix} \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix} = \sum_{i=1}^{3} \sigma_i m_{ii} \leqslant \sum_{i=1}^{3} \sigma_i \tag{6.46}$$

Therefore, the trace is maximized if $m_{ii} = 1$, which means M is the identity matrix:

$$I = M = V^{\text{T}}RU, \tag{6.47}$$

which leads to:

$$V = RU \quad \text{and} \quad R = VU^{\text{T}}. \tag{6.48}$$

One issue with this method is the ambiguity between rotation and reflection. If the $\det(VU^{\text{T}}) = -1$, the calculated R matrix is actually a reflection. For the case of pure rotation, $\det(VU^{\text{T}})$ should be 1. So when $\det(VU^{\text{T}}) = -1$, the next best alternative, which is a local maxima, needs to be found. If we look at the trace again, it is a function of M's diagonal values:

$$\text{tr}(\Sigma M) = f(m_{11}, m_{22}, m_{33}) = \sigma_1 m_{11} + \sigma_2 m_{22} + \sigma_3 m_{33} \tag{6.49}$$

By considering m_{ii}'s as variables, the domain of (m_{11}, m_{22}, m_{33}) is a subset of $[-1, 1]^3$. The function f is linear in the m_{ii}'s, thus it attains its extrema on the boundary of the domain. Since the domain here is rectilinear, the extrema will be attained at the vertices $(\pm 1, \pm 1, \pm 1)$. After $(1, 1, 1)$ has been ruled out (it is a reflection), the next best alternative is $(1, 1, -1)$:

$$\text{tr}(\Sigma M) = \sigma_1 + \sigma_2 - \sigma_3 \tag{6.50}$$

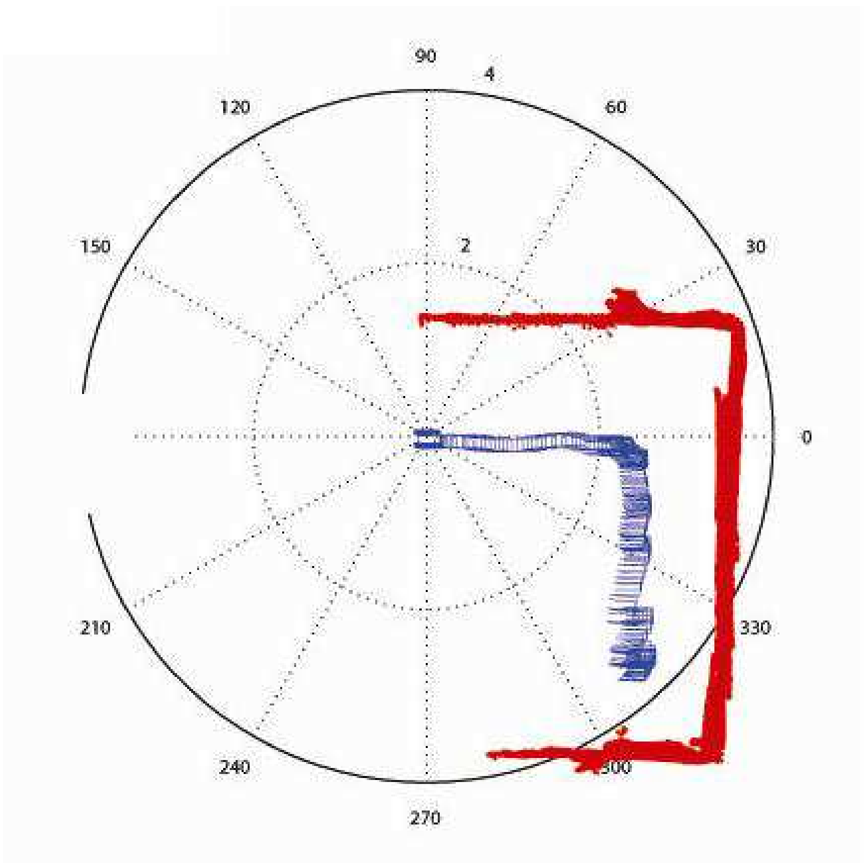


Figure 6.32: ICP result from simulation

To summarize, we can write a general formula that encompasses both cases, namely $\det(VU^{\mathrm{T}}) = 1$ and $\det(VU^{\mathrm{T}}) = -1$:

$$R = V \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \det(VU^{\mathrm{T}}) \end{bmatrix} U^{\mathrm{T}} \tag{6.51}$$

6.2.3 Simulation and Flight Test Results

This ICP algorithm has been firstly off-line implemented in MATLAB. The estimated UAV location (the track starting from the origin) and the generated map (the boundary) are shown in Fig. 6.32. Actual flight tests are also conducted with the same algorithm running onboard. The mapping and localization result is displayed on the GCS, and Fig. 6.33 is a screen capture. It can be seen that the indoor walls become thicker after the whole flight test, indicating a small drift of the estimated position.

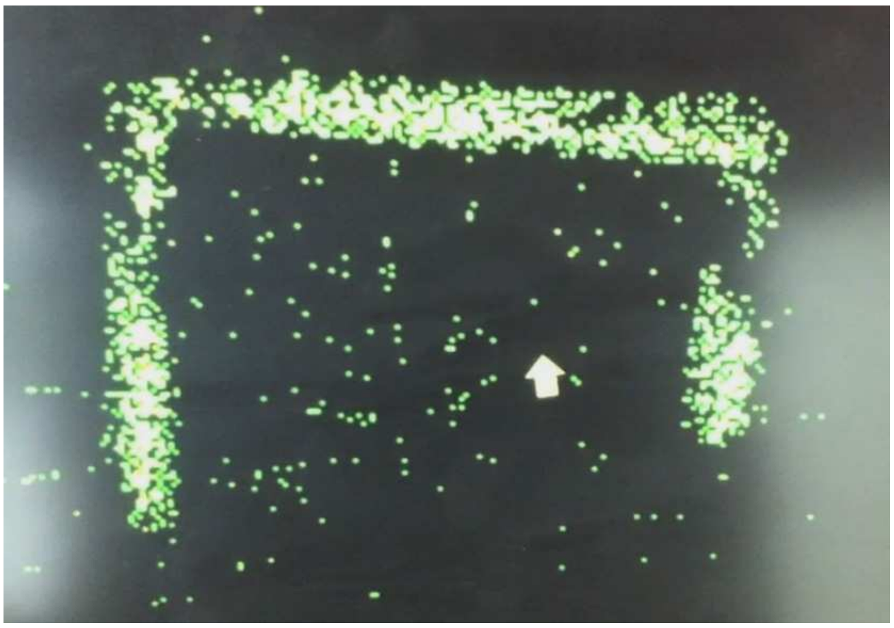


Figure 6.33: ICP result for a real flight test

Chapter 7

Path Planning Based on Local Laser Information

7.1 Background and Motivation

Path planning has been extensively studied in the robotics community. The general definition of path planning is to find a collision-free path in a known or unknown environment with static or dynamic obstacles. The traditional path planning algorithms can be categorized into three types, namely the road map methods, the cell decomposition methods and the potential field methods.

The road map approaches, e.g. visibility diagram [79] and Voronoi diagram [4], attempt to form a network connecting the current robot configuration to the destination configuration in all possible intermediate configurations. Then the path planning problem can be reduced to a searching problem in this configuration network for predefined optimum cost functions. Although optimal, this kind of method usually needs the full global information and its bundled calculation nature inherently limits its application to only off-line implementations.

The cell decomposition approaches have been widely used and are based on the concept of decomposing the set of free configurations into non-overlapping regions called ‘cells’, eg. [39] and [33]. The adjacency relationship of these cells is then represented in a ‘connectivity graph’, which will be searched for a path. The problem of this method is that all the cells and the connectivity graph must be constructed before the path searching algorithm takes place, and the amount of this pre-processing computation grows exponentially with the dimension of the configuration.

The potential field approaches, eg. [84] and [8], normally employ repulsive fields around obstacles and an attractive field around the goal. The gradient of the resultant potentials will guide the controlled robot to move towards the goal while avoiding obstacles in a natural way. One major drawback of these potential field methods is that there usually exists local minimums to the resultant potential fields which may trap the robot at that point infinitely. However, by manipulating the 'goal' or doing special case decisions, the local minimum problem can be largely avoided in practical situations. One good feature about this method is that it can be implemented in a way that only local information is needed, i.e. without knowing the global map.

For the first two approaches, the path planning problems are solved based on the assumptions of a known map and known UAV states. That means the SLAM problem, which will be discussed in Chapter 8, needs to be solved first if there is no global information about the UAV position and the indoor map. However, the indoor SLAM problem itself is very challenging, and implementing a high-performance SLAM algorithm onboard of the indoor UAV is extremely difficult. Hence in this chapter, we seek for a potential field based path planning algorithm which only relies on local map information, yet still be able to guide the UAV to fly in an indoor environment without collisions. It is definitely not the ultimate solution, but reasonable enough for UAVs equipped with a laser scanner to carry out autonomous exploration in a relatively clean indoor room. In addition, because of its simplicity, the proposed path planning algorithm can be easily realized onboard of the indoor UAV in real time.

7.2 Local Wall Following Strategies

In this section, a UAV wall following path planning solution that only relies on local laser scanner measurements, i.e. no global map or self-location information is needed, is proposed. It is a fairly universal strategy for indoor environments enclosed by vertical walls. Like many other local-map path planning methods, this algorithm utilizes the concept of the artificial potential field. All measurements from the laser scanner are treated as obstacles and they exert a repulsive force on the UAV. Besides, to let the UAV keeps moving forward, there is a constant attractive force coming from a virtual target two meters ahead from the UAV body. If the UAV is commanded to follow the wall on its left, then this virtual target is placed at the left-front of the UAV heading. If the the UAV is commanded to follow the wall on its right, then the target is placed at the right-front of the UAV heading. The followings are the formulation:

$$\boldsymbol{F} = \boldsymbol{F}_{\text{att}} - \boldsymbol{F}_{\text{rep}} \tag{7.1}$$

where

$$\boldsymbol{F}_{\text{att}} = e^{-\frac{|\boldsymbol{T}|^2}{2\sigma_1^2}}\boldsymbol{T} \tag{7.2}$$

and

$$\boldsymbol{F}_{\text{rep}} = \sum_{k=n_1}^{n_2} e^{-\frac{|\boldsymbol{W}(k)|^2}{2\sigma_2^2}} \frac{\boldsymbol{W}(k)}{\sigma_2 K} \tag{7.3}$$

$\boldsymbol{F}$ is the resultant force of the artificial potential field, with $\boldsymbol{F}_{\text{att}}$ being the attractive force generated by the virtual target and $\boldsymbol{F}_{\text{rep}}$ being the repulsive force generated by the wall obstacles. $\boldsymbol{T}$ and $\boldsymbol{W}(k)$ are unit vectors pointing towards the virtual target and the scanned points from the UAV CG respectively. σ_1 and σ_2 represent the stiffness of the Gaussian-shaped potential fields which can be tuned for different indoor situations. In (7.3), all laser scanner measurements indexed from n_1 to n_2 will be examined one by one. Invalid measurements, such as out-of-range data, will be dropped, and K is the total number of valid measurements afterwards.

Although $\boldsymbol{F}$ a virtual force, it can be interpreted as other physical entities in practice. In this implementation, we let the UAV 2-D velocity reference be proportional to $\boldsymbol{F}$, and by integration, it also forms the 2-D position reference. For the z direction, the UAV is ordered to maintain a predefined height with respect to the flat indoor floor. In addition, to determine the UAV heading reference, another algorithm is running at the same time to determine the UAV yaw angle reference at every time step, and it runs as follows:

1. If the UAV is to fly along the left side wall, omit the scanned points on the right side. If the UAV is to fly along the right side wall, omit the scanned points on the left side.
2. For all remaining scanned points, calculate the best straight line fit via least square optimization.
3. The UAV heading reference can be generated by deviating from the currently measured heading by a fraction α% of the difference between the fitted line gradient and the current heading (α needs to be tuned for UAVs with different yaw dynamics).

7.3 Simulation and Flight Test Results

By combining the potential field algorithm with the line fitting algorithm, the UAV outer-loop references can be comprehensively calculated. Before carrying out the actual flight tests, a MATLAB program is written to simulate the performance of this algorithm. A virtual map and a virtual laser scanner sensor, both to real-life scale, are coded for the purpose of navigation simulation. The dynamic model of the indoor UAV under closed-loop control is also integrated to make the simulation result more realistic. In Fig. 7.1, the sub-figure on the left is the global view of the indoor environment, which includes the walls, pillars and the UAV position and heading information. The sub-figure on the right shows the laser scanner measurements in the UAV body frame. The control reference to the UAV are purely calculated from the information from the right sub-figure, while the contents in the left sub-figure are only for display purposes and not available to the navigation algorithm.

The simulation results are shown in Figs. 7.1–7.6. This time, the virtual UAV is commanded to follow the wall on its left. The parameter configuration of path planning algorithm is:

$$\begin{cases} \sigma_1 &= 2, \\ \sigma_2 &= 1, \\ \alpha &= 5. \end{cases} \tag{7.4}$$

The same algorithm has been implemented on the UAV onboard system and actual flight tests have been carried out. The quadrotor platform, equipped with the 30 m's Hokuyo laser scanner, has performed an autonomous wall following flight in an indoor hall successfully. In this implementation, the position and velocity of the UAV are mainly obtained by visual odometry mentioned in Chapter 6. Figs. 7.7–7.14 sequentially show 8 instances of the flight, with the left sub-figure showing the physical flying condition and the right sub-figure showing the local laser scanner data. From the figures, one can observe that the UAV follows the initial wall on its left and then encounters and avoids a pillar and a protruding corner in the indoor hall. At the final stage of the flight, the UAV has started to follow the second long wall. If it continues, the UAV should be able to finish navigating through the whole indoor hall.

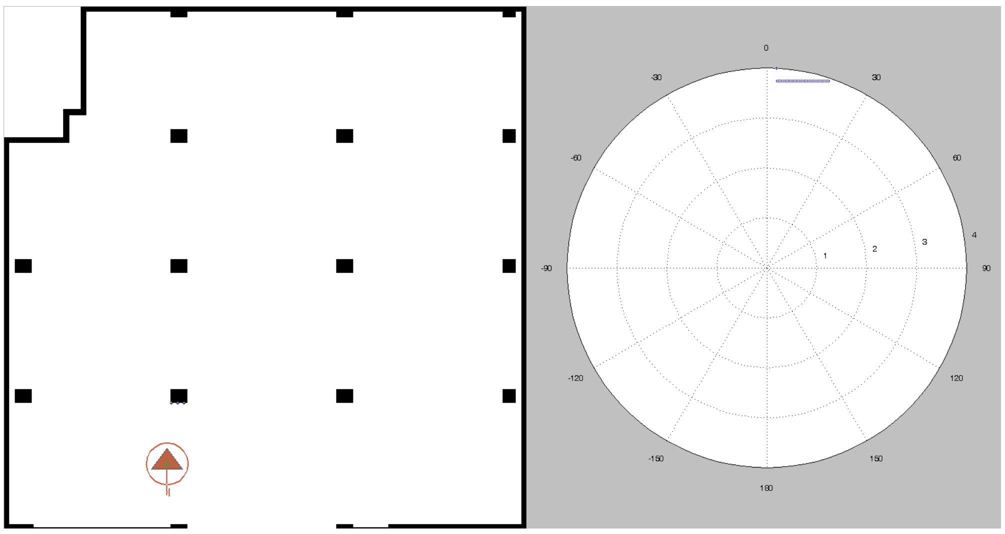


Figure 7.1: Wall-following strategy simulation result 1

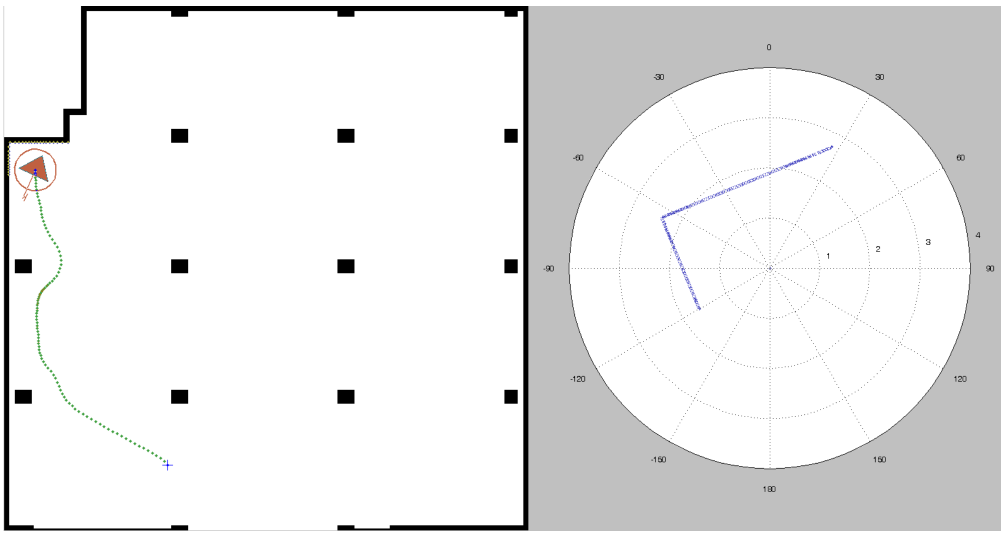


Figure 7.2: Wall-following strategy simulation result 2

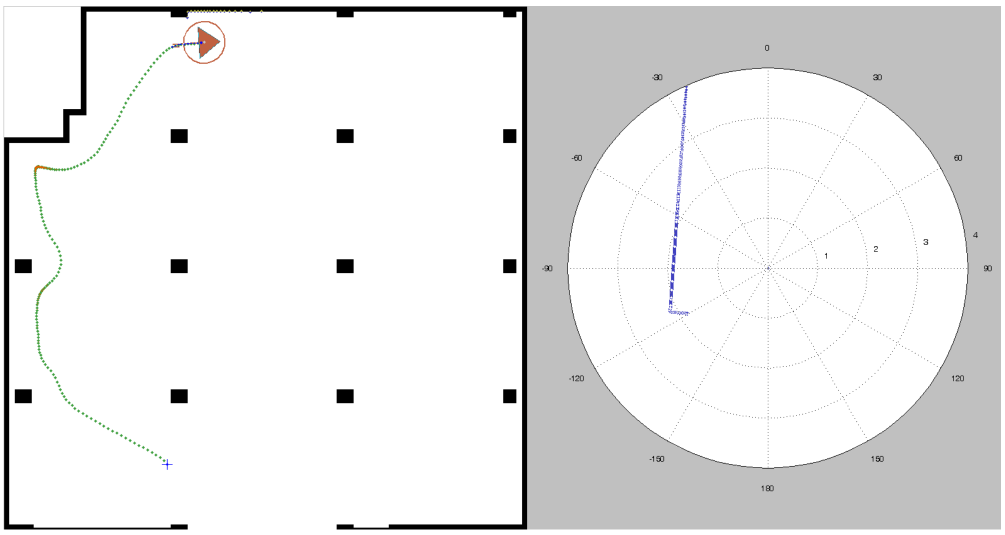


Figure 7.3: Wall-following strategy simulation result 3

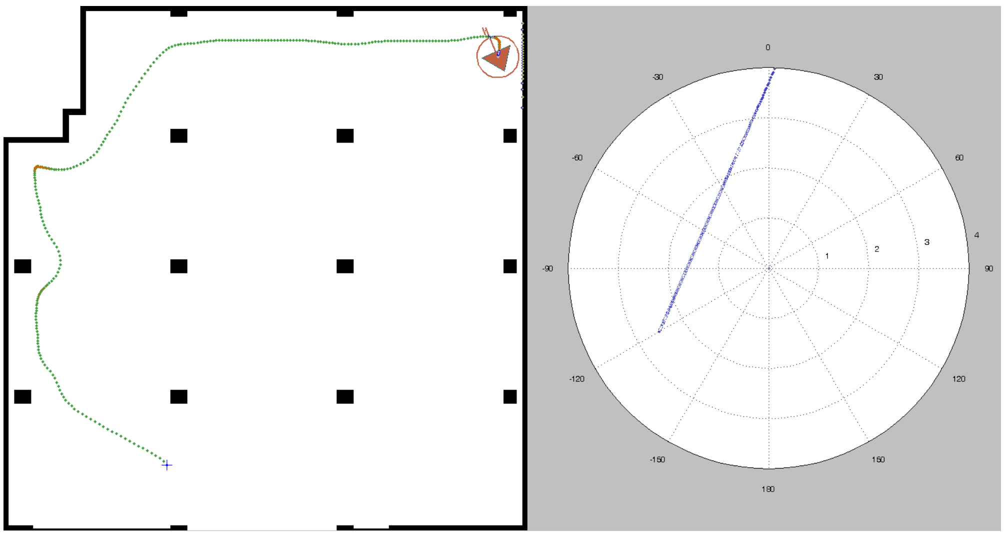


Figure 7.4: Wall-following strategy simulation result 4

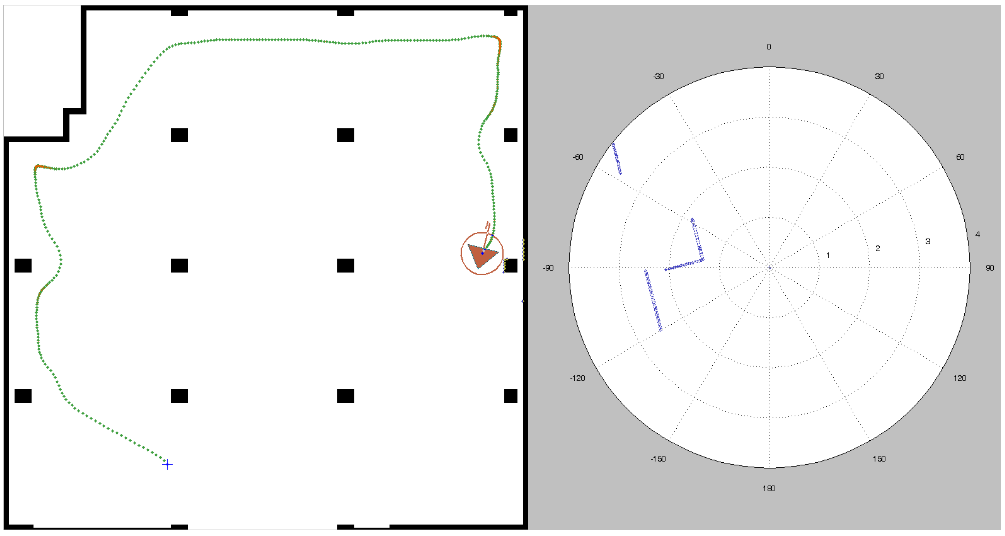


Figure 7.5: Wall-following strategy simulation result 5

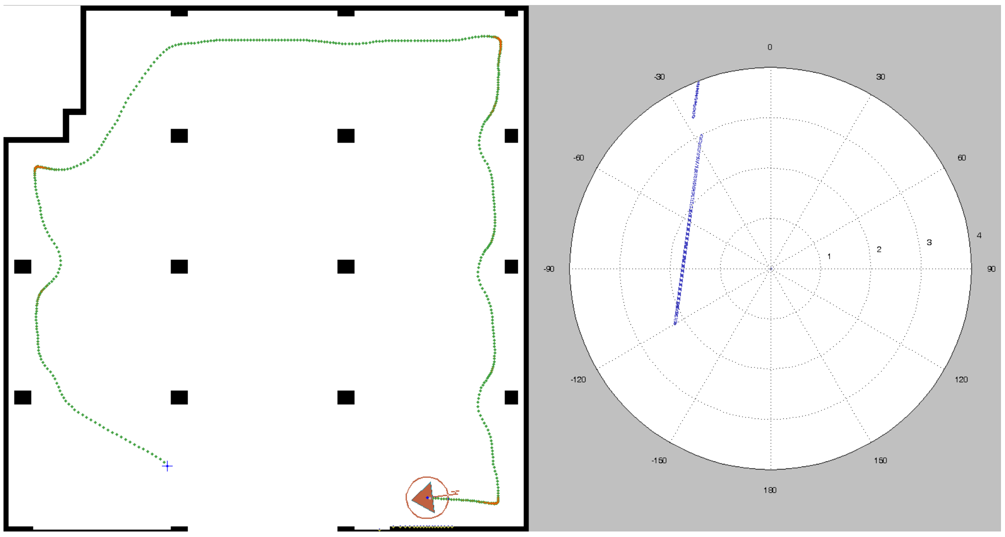


Figure 7.6: Wall-following strategy simulation result 6

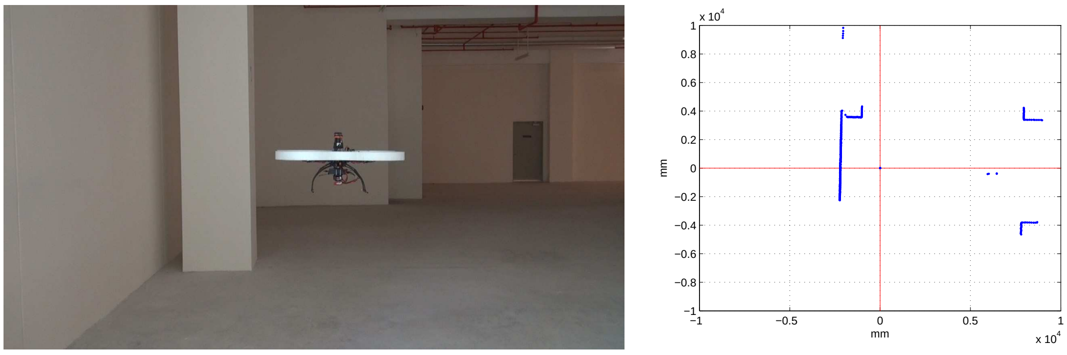


Figure 7.7: Wall-following flight test: hover and get prepared

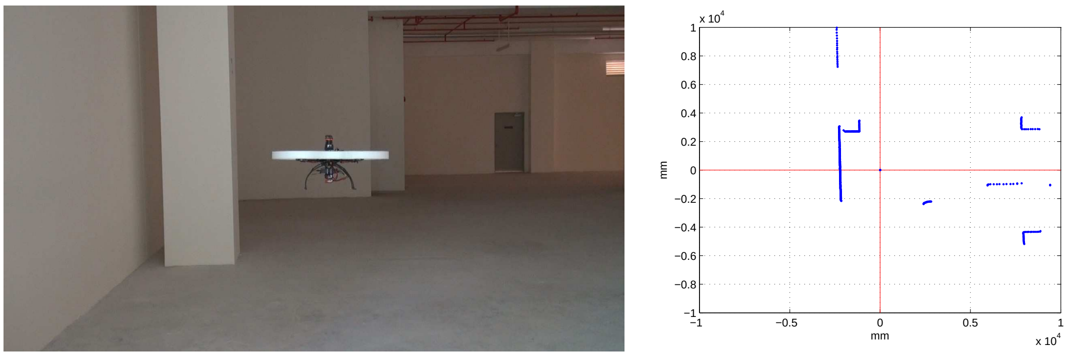


Figure 7.8: Wall-following flight test: start moving forward

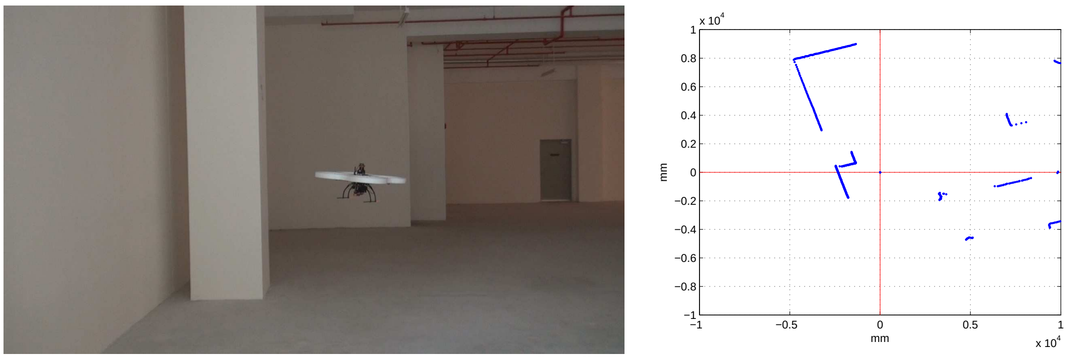


Figure 7.9: Wall-following flight test: avoid a pillar

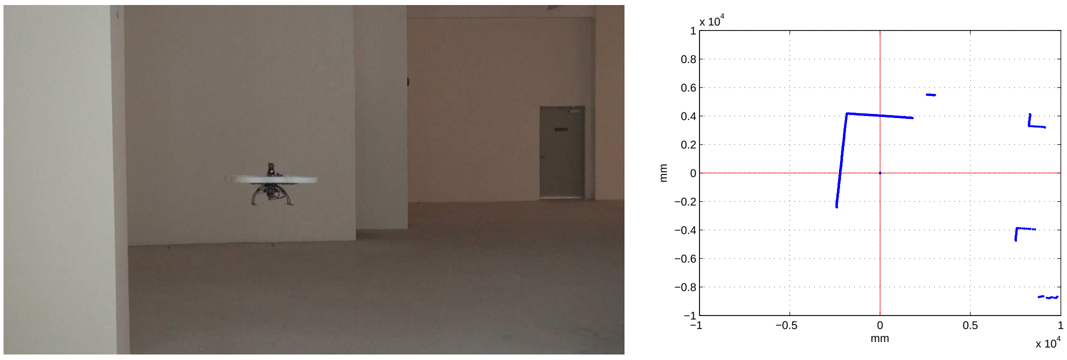


Figure 7.10: Wall-following flight test: fly back to wall

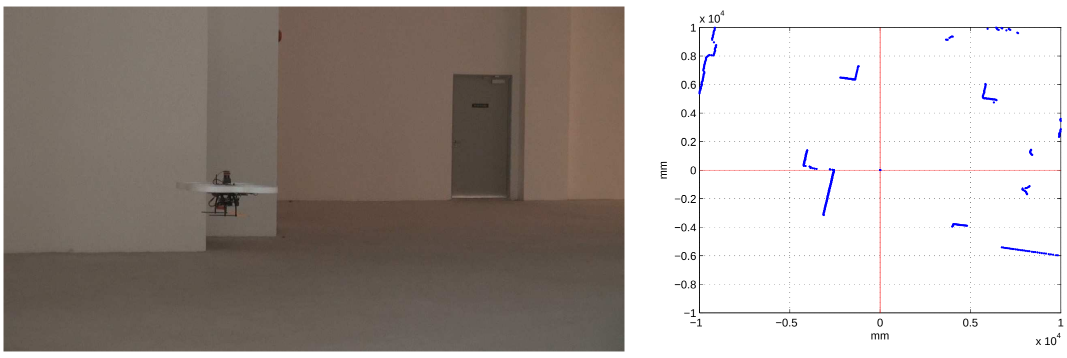


Figure 7.11: Wall-following flight test: encounter a frontal wall

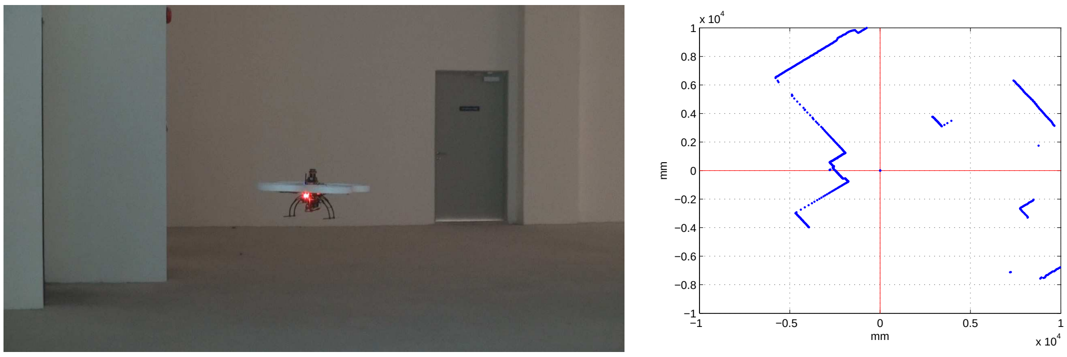


Figure 7.12: Wall-following flight test: go around the corners

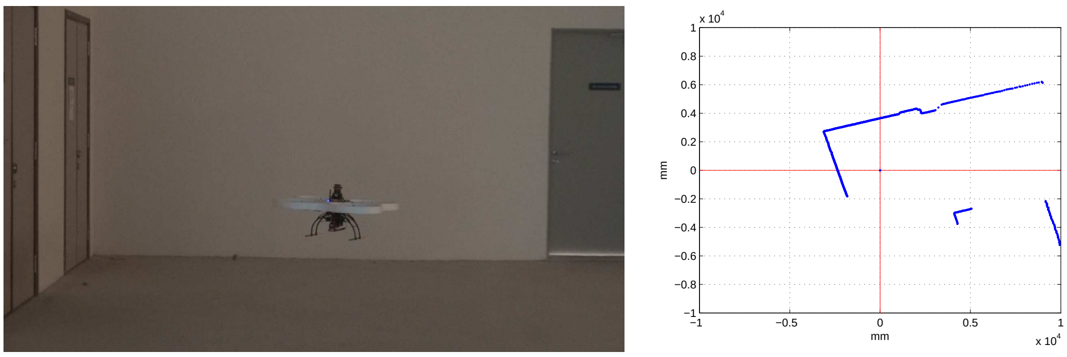


Figure 7.13: Wall-following flight test: encounter the 2nd frontal wall

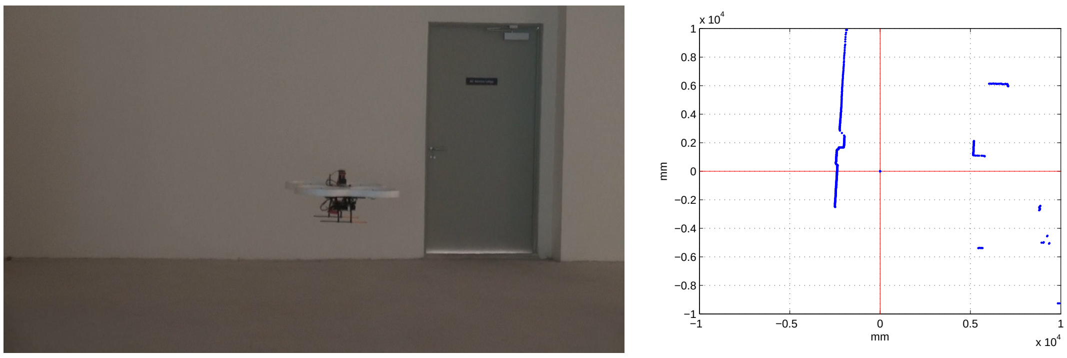


Figure 7.14: Wall-following flight test: start following a new wall

Chapter 8

Laser SLAM for Unknown Indoor Environments

The previous chapter has proposed methods to estimate UAV velocity by analyzing inter-frame visual images and inter-frame laser scanner measurements. Although position information can be estimated thereafter by dead reckoning, it will face the drifting problem eventually. To obtain better estimation of the UAV position, the SLAM technique has to be considered. Moreover, SLAM is not only for UAV localizing, it also produces a map of the environment, which is essential for most indoor UAV applications and it is the foundation for UAV autonomous path planning.

In literature, there are three prominent approaches in solving the SLAM problem. The first and also the most classical one is the Kalman filter (KF) based SLAM, which also includes its variants such as the Extended Kalman filter (EKF) and the Unscented Kalman filter (UKF). The next type of SLAM is based on the Particle filter concept. The most representative example is the FastSLAM. Last but not least, is the Graph-based SLAM in which a graph with nodes and edges representing the robot poses and the inter-pose constraints need to be constructed and solved by optimization techniques. In this work, the first two approaches will be explored with special attention given to a customized FastSLAM algorithm. The Graph-based SLAM has not been explored in this thesis. However, it is definitely another promising direction to achieve robust and accurate indoor SLAM which deserves in-depth investigation for future studies.

In addition, this chapter tries to solve the UAV indoor SLAM problem by focusing on the measurements from a 2-D scanning laser range finder. Due to its measurement nature, the result after applying the SLAM algorithm is also 2-D, or pseudo-3-D if necessary assumptions about the 3-D indoor environment can be made. Nevertheless, the discussed ideas and concepts are possible to be extended to the 3-D case if 3-D laser scanners or stereo cameras are used. However, the difficulty level of real-time onboard implementation will increase drastically due to computational constraints and the releasing of essential assumptions.

8.1 General SLAM Problems

In the robotics community, *mapping* is the task of modeling the environment surrounding the robot, which includes position and orientation of landmark features, while *localization* is to estimate the pose of the robot inside the map. It is not difficult to solve either of them if the information from the other is known. However, if the map of the environment and the robot's pose are both unknown, these two problems tangle together and will result in an intractable *chicken-or-egg* problem and people usually call it the SLAM problem. The followings will mathematically define a general SLAM problem. It should be also noted that the SLAM technique is not only useful for indoor navigation, but also widely used for outdoor, undersea, underground and space applications.

Let the pose of the robot at time t be denoted by s_t and the complete trajectory of the robot denoted as s^t. Then,

$$s^t = \{s_1, s_2, \dots, s_t\}. \tag{8.1}$$

Assume that the environment consists of a set of N immobile landmarks. The set of N landmark locations will be written as $\{\theta_1, \theta_2, \dots, \theta_N\}$. For notation simplicity, the whole map will be written as Θ.

As the robot moves through the environment, it collects relative information about its own motion. This information can be obtained using odometers attached to the wheels of a ground robot, dead reckoning by readings from the inertia measurement unit, or simply observing the control commands executed by the robot. Regardless of their origins, all these motion information is referred to as a control in general. The control at time t will be written as u_t. The set of all control executed by the robot is written as u^t. So

$$u^t = \{u_1, u_2, \dots, u_t\}. \tag{8.2}$$

As the robot moves through the environment, it also observes its nearby landmarks. The observation at time t is z_t and the set of all observations collected by the robot is z^t. So

$$z^t = \{z_1, z_2, \ldots, z_t\}. \tag{8.3}$$

In the SLAM literature, it is sometimes assumed that the observation of one landmark θ_n is distinctive enough to be differentiated from other landmarks. The variable n represents the identity of the landmark being observed. In practice, the identity of the landmarks usually cannot be guaranteed and this poses a big problem to all SLAM implementations. Here, we first assume that the landmark identities are known. This assumption will be released and the solutions to it will be discussed in Section 8.3. Let the identity of the landmark corresponding to the observation z_t be denoted as n_t, where $n_t \in \{1, \ldots, N\}$. The set of all data associations is written n^t. So

$$n^t = \{n_1, n_2, \ldots, n_t\}. \tag{8.4}$$

Using the notation defined above, the primary goal of a full SLAM problem is to recover the whole history of the robot pose s^t and the map Θ, given the set of noisy controls u^t and observations z^t. In formal probabilistic notation, this is expressed as,

$$p(s^t, \Theta | z^t, u^t, n^t). \tag{8.5}$$

However, the above posterior is rather complicated. Common robotics applications only need the current robot pose to be estimated, thus making the computation trackable and possibly real-time. Hence, instead of estimating (8.5), we need to only compute

$$p(s_t, \Theta | z^t, u^t, n^t). \tag{8.6}$$

A graphical overview of the SLAM problem is illustrated in Fig. 8.1. To solve the SLAM problem, the *motion model* (see Fig. 8.2) of the robot and the *measurement model* (see Fig. 8.3) of the sensor are needed. For the motion model, it is a representation of the robot's current state by examining its previous state and the current control input, which is

$$p(s_t | s_{t-1}, u_t). \tag{8.7}$$

For the measurement model, it is the observation or sensor model relates measurement with the

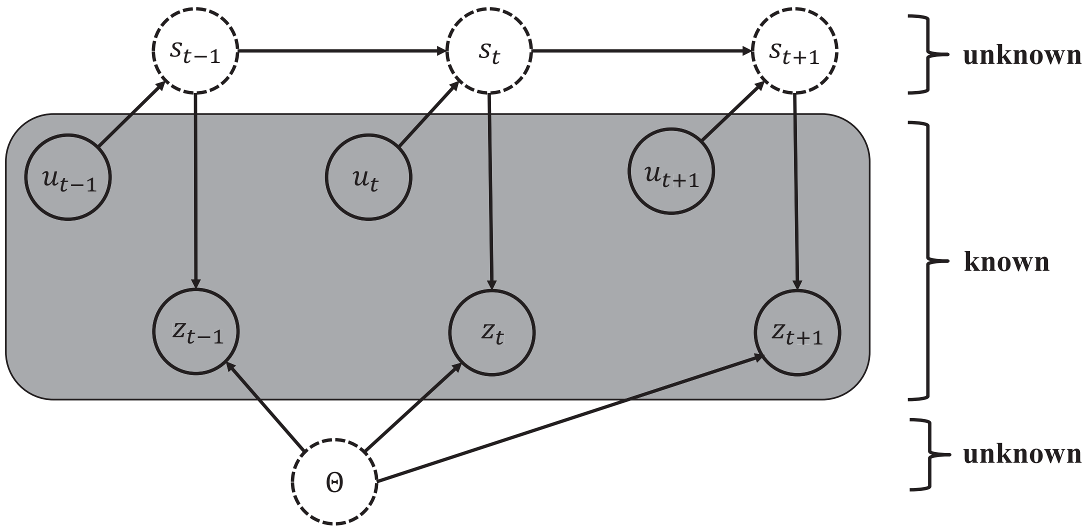


Figure 8.1: Graphical model of the SLAM problem

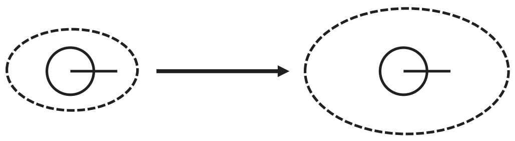


Figure 8.2: Motion model of a robot

robot's pose and the map. It can be represented as

$$p(z_t|s_t,\Theta). \tag{8.8}$$

With the problem well defined, different methods to solve the SLAM problem have bee proposed and they usually make a few assumptions about the statistical distribution of the controls and measurements, and also about the type of motion and measurement models. The next section will introduce the KF, EKF and UKF SLAM techniques in which a parametric (Gaussian) distribution of the control inputs and the measurements are assumed, while the motion or

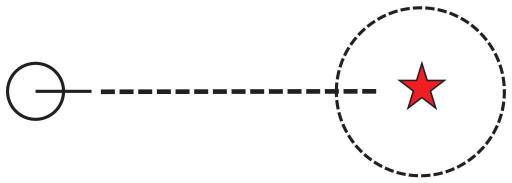


Figure 8.3: Measurement model of a robot sensor

measurement processes can be linear or nonlinear.

8.2 KF, EKF and UKF SLAM Approaches

Although the main contribution in solving the UAV indoor SLAM problem in this research work is a customized FastSLAM algorithm, which will be discussed in the Section 8.3, the KF, EKF and UKF based SLAM methods will be introduced first as they share a few similar concepts and notations with the Particle filter based SLAM and their inherent problems have motivated the discovery of FastSLAM.

8.2.1 Kalman Filter SLAM

The original KF algorithm was proposed long time ago in [34]. The application of KF filter to the SLAM problem has been well documented in [76]. The KF SLAM assumes that the motion model and the measurement model of the robotic system are both linear, and the control inputs and measurements are all Gaussian variables. Hence, every signal in the analysis can be written in a Gaussian parametric form as

$$\mathrm{p}(x) = \det(2\pi\Sigma)^{-\frac{1}{2}}\exp\left(-\frac{1}{2}(x-\mu)^{\mathrm{T}}\Sigma^{-1}(x-\mu)\right) \tag{8.9}$$

or

$$x \sim \mathcal{N}(\mu, \Sigma) \tag{8.10}$$

where x is a random variable with Gaussian distribution, while μ and Σ are its mean (scalar or vector) and variance (scalar or matrix) respectively. The SLAM problem can be solved by constructing a state variable x_t which includes both the estimation of the robot pose s_t and the estimation of the map Θ_t. KF represents the SLAM posterior as a high-dimensional, multi-variate Gaussian function parameterized by a mean μ_t and a covariance matrix Σ_t. The mean describes the most likely state of the robot and the landmarks, whereas the covariance matrix encodes noises and correlations between all pairs of state variables. So the problem becomes estimating the probability of

$$p(s_t, \Theta | z^t, u^t, n^t) = \mathcal{N}(x_t; \mu_t, \Sigma_t), \tag{8.11}$$

where

$$x_t = \{s_t, \theta_1, \ldots, \theta_N\} \tag{8.12}$$

$$\mu_t = \{\mu_{s_t}, \mu_{\theta_{1,t}}, \ldots, \mu_{\theta_{N,t}}\} \tag{8.13}$$

$$\Sigma_t = \begin{bmatrix} \sigma_{s_t,s_t} & \sigma_{s_t,\theta_1} & \cdots & \sigma_{s_t,\theta_N} \\ \sigma_{\theta_1,s_t} & \sigma_{\theta_1 s_t} & \sigma_{\theta_1\theta_2} & \cdots \\ \vdots & \sigma_{\theta_2,\theta_1} & \ddots & \vdots \\ \sigma_{\theta_N,s_t} & \cdots & \cdots & \sigma_{\theta_N,\theta_N} \end{bmatrix}_t \tag{8.14}$$

.

It is well known that for a linear time invariant system

$$\begin{cases} x_t = Ax_{t-1} + Bu_t + \epsilon_t, \\ z_t = Cx_t + \delta_t, \end{cases}, \tag{8.15}$$

where x_t, u_t, z_t are the state, control and measurement vectors respectively. A, B, C are system matrices with appropriate dimensions. ϵ and δ are the input and measurement noises, which are assumed to be Gaussian with zero means and covariance matrices of Q and R respectively. The main objective of Kalman filter is to estimate the mean μ and variance Σ of x at the time step t with the control input u_t, the measurement z_t and the previously estimated state, $x_{t-1} \sim \mathcal{N}(\mu_{t-1}, \Sigma_{t-1})$. If system 8.15 is observable, then the statistically optimal estimation process is given as follows:

1. take inputs $(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t)$
2. $\bar{\mu}_t = A\mu_{t-1} + Bu_t$
3. $\bar{\Sigma}_t = A\Sigma_{t-1}A^{\mathrm{T}} + Q$
4. $K_t = \bar{\Sigma}_t C^{\mathrm{T}}(C\bar{\Sigma}_t C^{\mathrm{T}} + R)^{-1}$
5. $\mu_t = \bar{\mu}_t + K_t(z_t - C\bar{\mu}_t)$
6. $\Sigma_t = (I - K_t C)\bar{\Sigma}_t$
7. return (μ_t, Σ_t)

8.2.2 Extended Kalman Filter SLAM

However, KF assumes that the motion model and the measurement model are both linear. In cases where the processes are nonlinear, EKF can be used to solve the estimation problem by linearizing the formulation at the most likely state of the system. Note that when the processes are nonlinear, EKF may not be the optimal estimator. It is just a practical solution to apply the KF concept to nonlinear systems. Thus, for the following nonlinear system:

$$\begin{cases} x_t & = & g(u_t, x_{t-1}) + \epsilon_t \\ z_t & = & h(x_t) + \delta_t \end{cases}, \tag{8.16}$$

where $g(*)$ and $h(*)$ are multi-dimensional nonlinear functions describing the motion model and measurement model respectively, and ϵ_t and δ_t are still zero mean Gaussian noises with covariance matrices Q and R respectively. Let G_t and H_t be the Jacobian matrices of g and h with respect to x, or in detail,

$$G_t = \begin{bmatrix} \frac{dg_1}{dx_1} & \frac{dg_1}{dx_2} & \cdots & \frac{dg_1}{dx_m} \\ \frac{dg_2}{dx_1} & \frac{dg_2}{dx_2} & \cdots & \frac{dg_2}{dx_m} \\ \vdots & \vdots & \cdots & \vdots \\ \frac{dg_m}{dx_1} & \frac{dg_m}{dx_2} & \cdots & \frac{dg_m}{dx_m} \end{bmatrix}, \quad H_t = \begin{bmatrix} \frac{dh_1}{dx_1} & \frac{dh_1}{dh_2} & \cdots & \frac{dh_1}{dx_m} \\ \frac{dh_2}{dx_1} & \frac{dh_2}{dh_2} & \cdots & \frac{dh_2}{dx_m} \\ \vdots & \vdots & \cdots & \vdots \\ \frac{dh_l}{dx_1} & \frac{dh_l}{dx_2} & \cdots & \frac{dh_l}{dx_m} \end{bmatrix} \tag{8.17}$$

where l and m are the dimension of measured outputs and the dimension of state respectively. With G_t and H_t evaluated at the current operating point, the EKF estimation can be processed as follows:

1. take inputs $(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t)$
2. $\bar{\mu}_t = g(u_t, \mu_{t-1})$
3. $\bar{\Sigma}_t = G_t \Sigma_{t-1} G_t^{\mathrm{T}} + Q$
4. $K_t = \bar{\Sigma}_t H_t^{\mathrm{T}} (H_t \bar{\Sigma}_t H_t^{\mathrm{T}} + R)^{-1}$
5. $\mu_t = \bar{\mu}_t + K_t(z_t - h(\bar{\mu}_t))$
6. $\Sigma_t = (I - K_t H_t)\bar{\Sigma}_t$
7. return (μ_t, Σ_t)

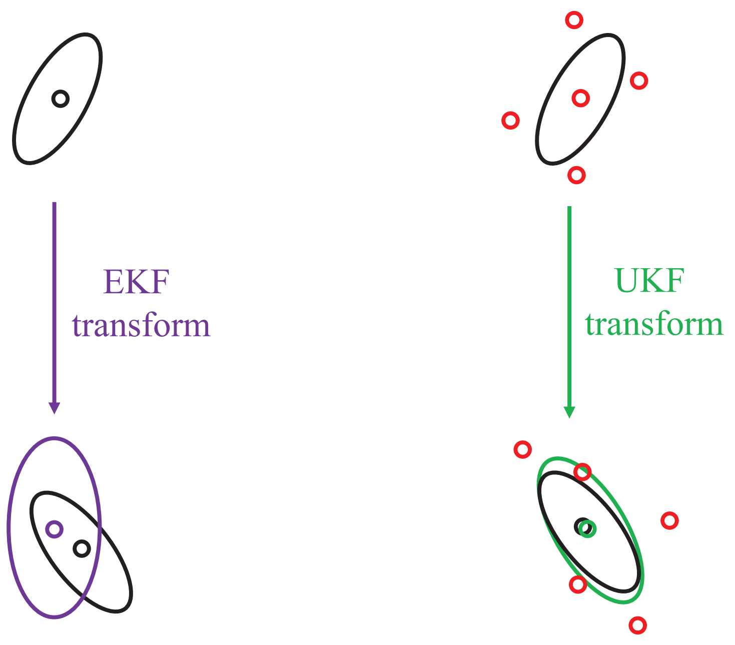


Figure 8.4: UKF vs. EKF

8.2.3 Unscented Kalman Filter SLAM

While EKF handles the nonlinear processes by linearizing the models at the current operating point, i.e. considering only 1st order term from Taylor expansion, UKF tries to estimate the process noise in a more accurate way by bringing the concept of sigma points. The idea to propagate noise in UKF is as follows:

1. Sample a set of sigma points with different weights around the operating point;
2. Transform the sigma points through the nonlinear function;
3. Compute an approximate Gaussian from the transformed points with resultant weights.

As shown in Fig. 8.4, by choosing appropriate sigma points, the unscented transform sometimes works better than the linearization result from EKF. However, it is indeed a challenging problem to decide where to put these sigma points $\chi^{[i]}$ and their respective weights $w^{[i]}$ appropriately. If we only consider selecting $\chi^{[i]}$ and $w^{[i]}$ to fulfil the following conditions:

$$\begin{aligned}\sum_i w^{[i]} &= 1,\\ \mu &= \sum_i w^{[i]}\chi^{[i]},\\ \Sigma &= \sum_i (\chi^{[i]}-\mu)(\chi^{[i]}-\mu)^{\mathrm{T}},\end{aligned}$$

then there is no unique solution of χ_i and w_i. A common approach to this problem is to choose the sigma points in a symmetric way as follows:

$$\begin{aligned} \chi^{[0]} &= \mu, \\ \chi^{[i]} &= \mu + \left(\sqrt{(n+\lambda)\Sigma}\right)_i \qquad \text{for} \quad i = 1, \ldots, n, \\ \chi^{[i]} &= \mu - \left(\sqrt{(n+\lambda)\Sigma}\right)_{i-n} \quad \text{for} \quad i = n+1, \ldots, 2n, \end{aligned} \tag{8.18}$$

where the $(*)_i$ means the i-th column vector of the matrix $*$, and n controls the total number of sigma points. Thereafter, we can compute

$$\begin{aligned} w_{\mathrm{m}}^{[0]} &= \tfrac{\lambda}{n+\lambda}, \\ w_{\mathrm{c}}^{[0]} &= w_{\mathrm{m}}^{[0]} + (1 - \alpha^2 + \beta), \\ w_{\mathrm{m}}^{[i]} = w_{\mathrm{c}}^{[i]} &= \tfrac{1}{2(n+\lambda)} \quad \text{for} \quad i = 1, \ldots, 2n. \end{aligned} \tag{8.19}$$

where $w_{\mathrm{m}}^{[i]}$ and $w_{\mathrm{c}}^{[i]}$ are the weights to calculate the transformed mean and variance respectively. The calculations are as follows:

$$\mu' = \sum_{i=0}^{2n} w_{\mathrm{m}}^{[i]} g(\chi^{[i]}), \tag{8.20}$$

$$\Sigma' = \sum_{i=0}^{2n} w_{\mathrm{c}}^{[i]} \left(g(\chi^{[i]}) - \mu'\right)\left(g(\chi^{[i]}) - \mu'\right)^{\mathrm{T}}. \tag{8.21}$$

α, β, λ are parameters subject to the following constraints:

$$\begin{aligned} \alpha &\in (0, 1], \\ \beta &= 2 \quad \text{(optimal choice for Gaussian distribution)}, \\ \lambda &= \alpha^2(n+\kappa) - n, \ \text{with} \ \kappa \geq 0. \end{aligned}$$

Hence, by utilizing the concept of sigma points, UKF SLAM updates the robot pose and map with the following procedures:

1. take inputs $(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t)$
2. $\chi_{t-1} = (\mu_{t-1},\ \mu_{t-1} \pm \gamma_1\sqrt{\Sigma_{t-1}},\ \mu_{t-1} \pm \gamma_2\sqrt{\Sigma_{t-1}},\ \ldots)$
3. $\bar{\chi}_t = g(u_t, \chi_{t-1})$
4. $\bar{\mu}_t = \sum_{i=0}^{2n} w_{\mathrm{m}}^{[i]} \bar{\chi}_t^{[i]}$

```
5.  $\bar{\Sigma}_t = \sum_{i=0}^{2n} w_{\mathrm{c}}^{[i]} (\bar{\chi}_t^{[i]} - \bar{\mu}_t)(\bar{\chi}_t^{[i]} - \bar{\mu}_t)^{\mathrm{T}} + Q$
6.  $\bar{\chi}_t = (\bar{\mu}_t,\ \bar{\mu}_t \pm \gamma_1 \sqrt{\bar{\Sigma}_t},\ \bar{\mu}_t \pm \gamma_2 \sqrt{\bar{\Sigma}_t},\ \dots)$
7.  $\bar{Z}_t = h(\bar{\chi}_t)$
8.  $\hat{z}_t = \sum_{i=0}^{2n} w_{\mathrm{m}}^{[i]} \bar{Z}_t^{[i]}$
9.  $S_t = \sum_{i=0}^{2n} w_{\mathrm{c}}^{[i]} (\bar{Z}_t^{[i]} - \hat{z}_t)(\bar{Z}_t^{[i]} - \hat{z}_t)^{\mathrm{T}} + R$
10. $\bar{\Sigma}_t^{x,z} = \sum_{i=0}^{2n} w_{\mathrm{c}}^{[i]} (\bar{\chi}_t^{[i]} - \bar{\mu}_t)(\bar{Z}_t^{[i]} - \hat{z}_t)^{\mathrm{T}}$
11. $K_t = \bar{\Sigma}_t^{x,z} S_t^{-1}$
12. $\mu_t = \bar{\mu}_t + K_t(z_t - \hat{z}_t)$
13. $\Sigma_t = \bar{\Sigma}_t - K_t S_t K_t^{\mathrm{T}}$
14. return $\mu_t$, $\Sigma_t$
```

8.2.4 Problems of KF, EKF, UKF SLAMs

The KF, EKF, UKF SLAM methods and their variants have been extensively used in robotics applications [73, 50]. However, they have common limitations which hinder them to be expanded to applications that need larger maps, longer navigation time and more noisy measurements to be handled.

The first problem of the KF types of SLAM is the *curse of dimensionality*. For example, when a robot moves in a 2-D plane, the state vector to be estimated is of dimension $2N + 3$, where N is the number of landmarks, since three dimensions are needed to represent the pose of the robot and two dimensions are needed to confirm the position of each landmark. In consequence, the covariance matrix is of size ($2N + 3$) by ($2N + 3$). Thus, the number of parameters needed to describe the posterior is quadratic with respect to the number of landmarks in the map. It should be noted that large-dimensional matrix computations, such as calculating the inverse of a matrix, are usually time consuming. When the robot or UAV moves, more and more new landmarks will be discovered and included into the state. In the long run, it will easily make the algorithm inefficient and thus impossible to be run in real time. This is seen as one of the main drawbacks of the KF types of SLAM methods, as their computation complexity is quadratic.

Second, although the EKF and UKF SLAM methods try to solve the problem caused by nonlinear motion model and measurement model, they are still not the optimal estimator for general nonlinear cases. The approximation made by them is good if the true models are approximately linear and if the discrete time step of the filter is small. However, in most practical operations, motion models and measurement models can be highly nonlinear. Applying these methods blindly may not guarantee a good overall results, or sometimes the filter even diverges.

Except for the above two shortcomings, the KF types of SLAM methods also suffer from the problem of wrong data associations. These SLAM methods usually maintain a single data association hypothesis per observation, typically chosen using a maximum likelihood heuristics, i.e., if the probability of an observation coming from any of the currently observed landmarks is low, the possibility of a new landmark is considered. If the data association chosen by the heuristic is incorrect, the effect of incorporating this observation into the filter can never be removed. If too many observations are incorporated into the filter with wrong data associations, the filter will easily diverge. This is a well known failure mode of the KF types of SLAM.

To overcome these problems, a lot of SLAM variants have been proposed by researchers. Some of them tried to exploit the sparsity of the matrix updating step in EKF [29, 77]. Some have proposed more robust methods of data association [7, 53]. Among them, the FastSLAM [48] is one of the most promising methods to improve both the robustness and efficiency of the algorithm. Unlike many other methods which factorize the SLAM problem spatially, FastSLAM factorizes the SLAM posterior over time using the path of the robot. The resulting algorithm scales logarithmically with the number of features in the map. In addition, FastSLAM originates from the particle filter, retaining different data association hypotheses to different particle solutions. The particles with wrong data associations can be completely forgotten in the long run. In the next section, the FastSLAM framework is adopted while the type of map features has been extended from the classical corner-only features to both corner and line features.

8.3 A Customized FastSLAM Algorithm

This section adopts the FastSLAM framework and applies it in a structured indoor environment, structured in the sense that the environment is purely constituted by vertical and straight walls. By realizing that common SLAM or FastSLAM solutions only consider point or corner features, this work moves one step further by bringing in line segment features into the algorithm as a supplement. It is believed that by using more types of map features into the FastSLAM particle filter, the performance could be more robust. Additionally, since most filter based SLAM algorithms prioritize localization rather than mapping (the point-feature-based sparse map usually cannot provide sufficient information of the environment for human or computer interpretation), there still needs an stand-alone mapping algorithm that can generate dense and more meaningful maps based on the localization result. That makes the whole solution even more tedious and computationally requiring. In the case of a clean indoor environment where corners and line segments are enough to describe the room or corridor setups, the proposed solution can do localization more robustly and at the same time, to generate a 'meaningful map' in an efficient way.

8.3.1 Algorithm Overview

The customized FastSLAM algorithm mentioned here engages a particle filter to represent the probability distribution of the UAV's pose at time t, denoted as s_t. The covariance of the UAV pose is represented by a distribution of M particles, and it is assumed that the m^{th} particle $P^{[m]}$ knows exactly where the vehicle's position and orientation are, without uncertainty. Hence, instead of having a huge covariance matrix, such as that of the EKF algorithm, this algorithm has many small covariance matrices for each combination of the map features and one of the possible UAV poses. This avoids the inverse calculation of a large matrix, which is the most computationally expensive step in those conventional SLAM algorithms. Each particle carries its own map. Similar to the EKF SLAM approaches, each feature in the map is assumed to have a Gaussian distribution, which can be described by its mean $\mu_{n,t}^{[m]}$ and variance $\Sigma_{n,t}^{[m]}$, where n is the associated index of the map features.

Whenever a new frame of laser scan is available, the algorithm converts the raw range data into a set of measurement features $\{z_{1,t}, z_{2,t}, \ldots, z_{i,t} \in z_t\}$ described by their respective mean $\mu_{i,t}^{z}$ and variance $\Sigma_{i,t}^{z}$. This step is referred to as **feature extraction**, and in our case, features include both corners and line segments. The **motion estimation** step is in charge of predicting the displacement of the vehicle between two adjacent frames, say u_t, according to the vehicle's motion equation or simply by scan matching. Based on this predicted displacement, the particles are propagated according to the motion model with pre-defined noises Σ_{u_t}. This is called a **proposal generation**. In the next **measurement update** step, measurements with respect to all particles are associated with the existing features in the map through per-particle **data association**. A weight $w_t^{[m]}$ is computed for each particle according to how well the measurements fit the features in that particular particle's map. At the same time, the existing map features in the particle can be updated, too. Finally the weighted particles are **re-sampled** to generate the new probability distribution of the vehicle's pose. In summary, the essential steps in a FastSLAM algorithm are shown below:

1. Feature Extraction;
2. Motion Estimation and Proposal Generation;
3. Per-particle Data Association;
4. Per-particle Measurement Update;
5. Importance Weighting and Resampling.

The next section will expand all the above steps sequentially, and the solution to include line segment features for this framework will be explained whenever necessary. However, there is one important simplification to the algorithm that needs to be stated clearly here. When this work applies EKF to the per-particle measurement update, the parameters describing the extracted line segment or corner features are assumed to be directly measured. In other words, the measurement model is simply a one-to-one copy of the EKF state variables with some predefined noises, and before EKF updating, all state variables and measurement variables have already been converted to the global frame according to the estimated UAV pose in that particular particle at that particular moment.

8.3.2 Feature Extraction

The objective of this feature extraction step is to convert a frame of raw measurement points into a set of measurement features, $\{z_{1,t}, z_{2,t}, \ldots, z_{n,t} \in z_t\}$, and at the same time, to establish the probability distribution of the extracted features, $P(z_{i,t})$. In the context of this report, it is assumed that the environment is structured and can be well described by line segments and corners with their descriptive parameters in Gaussian distribution. Hence, the features to be extracted are line segments and corners, and each of them can be represented by a vector mean and a square matrix covariance with the following notations (also see Fig. 8.5 for a graphical illustration):

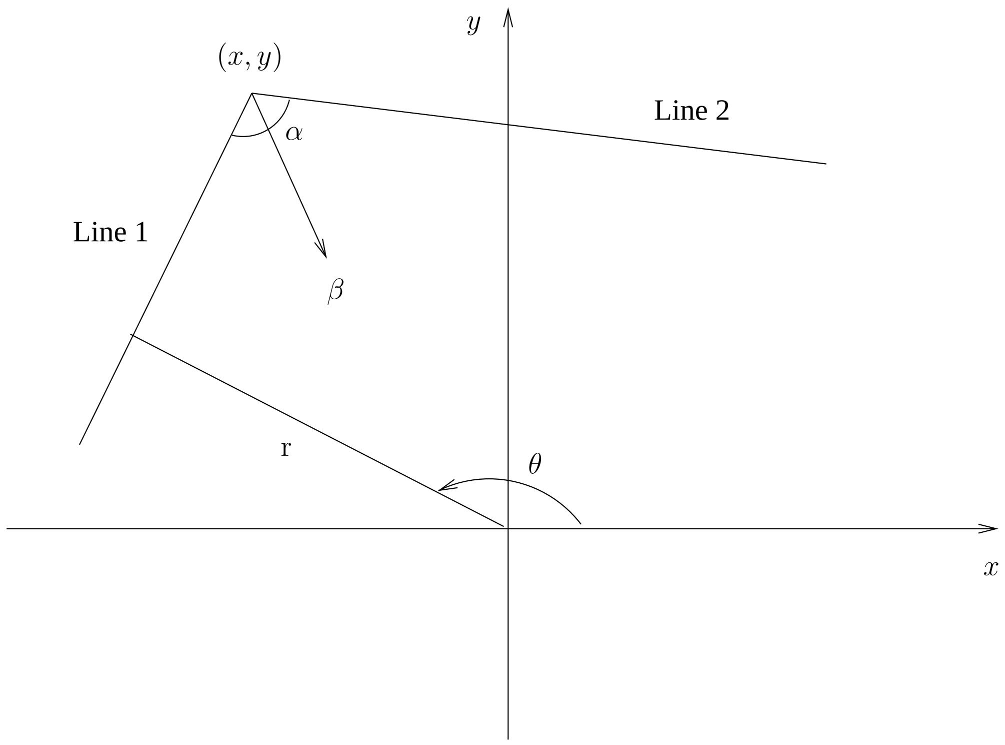


Figure 8.5: Parameters to describe line and corner features

- **Lines:**

$$z_l = \mathcal{N}(r, \theta; \mu_l^z, \Sigma_l^z) \tag{8.22}$$

$$\mu_l^z = \{\mu_r, \mu_\theta\} \tag{8.23}$$

$$\Sigma_l^z = \begin{bmatrix} \sigma_{rr} & \sigma_{r\theta} \\ \sigma_{\theta r} & \sigma_{\theta\theta} \end{bmatrix} \tag{8.24}$$

- **Corners:**

$$z_c = \mathcal{N}(x, y, \alpha, \beta; \mu_c^z, \Sigma_c^z) \tag{8.25}$$

$$\mu_c^z = \{\mu_x, \mu_y, \mu_\alpha, \mu_\beta\} \tag{8.26}$$

$$\Sigma_c^z = \begin{bmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{x\alpha} & \sigma_{x\beta} \\ \sigma_{yx} & \sigma_{yy} & \sigma_{y\alpha} & \sigma_{y\beta} \\ \sigma_{\alpha x} & \sigma_{\alpha y} & \sigma_{\alpha\alpha} & \sigma_{\alpha\beta} \\ \sigma_{\beta x} & \sigma_{\beta y} & \sigma_{\beta\alpha} & \sigma_{\beta\beta} \end{bmatrix} \tag{8.27}$$

The process of converting a frame of raw scan points into a set of corners and line segments can be divided into four steps, namely clustering, line fitting, line merging and filtering, and corner extraction. The clustering task is to group the raw data points in such a way that points within the same group belong to the same line segment as reasonable as possible. A recent review of line extraction algorithm [55] concludes that the *Split-and-Merge* and the *Incremental* are the two preferred algorithms, with *Split-and-Merge* being more speedy and *Incremental* being more robust. Our approach uses the *Split-and-Merge* method since it is good enough for a clean indoor environment. It is a recursive line extraction algorithm with the following steps:

1. Start with all input points.
2. Connect the first point and the last point with a line.
3. Calculate the perpendicular distances of all other intermediate points with respect to the line segment obtained in the previous step.
4. Search for the point that has the largest distance and compare this distance with a defined threshold.
5. If the maximum distance is less than the threshold, all points between the first point and the last point belong to the same line; Else, recursively call Step 1 with (first point, max point) and (max point, last point).

Split-and-Merge in our case, only clusters points into groups. The next step, line fitting, is the actual line feature extraction step to get the individual line parameters and covariance. For this step, since the uncertainties of the points belong to the same line are different, as a result of projecting elliptical shaped uncertainty (as the radial and angular component are different) of varied magnitude (measurements are more noisy for points at longer distance from the sensor). Such variations in uncertainty make fitting lines to a group of points a more difficult task. As the uncertainty of each of the point is different, points have to be weighted when fitting the line. Furthermore, the weight of the point is dependent on the heading of the line. As a result, no close form formula has been found in solving the line fitting problem. Instead, an iterative method that maximizes the likelihood of the line has been derived in [59] and it works as follows:

1. Assume an initial heading of the line by connecting the first and the last point.
2. Find the radial position and its variance based on the weight obtained by projecting uncertainties of the point onto the perpendicular direction of the line.
3. Calculate the iterative incensement for heading.
4. Repeat the Step 1 to Step 3 until heading converges.

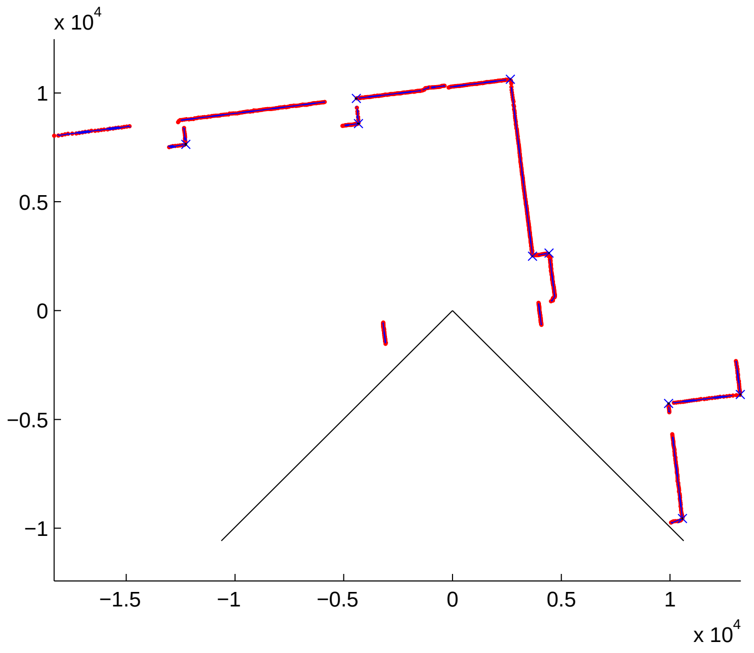


Figure 8.6: Lines (blue) and corners (black) extracted from a frame of raw laser scanner data (red)

5. Calculated the remaining terms in the covariance matrix for line parameters.

The detailed formulation of the iterative line parameter extraction and covariance estimation can be found in [59] and it will not be repeated here. However, the results can be appreciated in Fig. 8.7, where the extracted lines have dotted boundaries at both sides of the line, representing the 3-sigma uncertainty region.

For line merging and filtering, it tries to minimize the errors in the previous clustering step by merging lines that are obtained from the same feature but split into different clusters by mistake. Adjacent lines extracted in the previous step are examined by some merging criteria. Mahalanobis distance can be used in this case for probabilistically better judgement. Those lines satisfying the merging criteria are re-joined together to form the same single line feature. On the other hand, lines which are too short in length are normally unstable to be used as map features, thus can be discarded.

For corner extraction, adjacent lines are extended and their intersecting point is taken to be the position of the corners. Direction (β) and angle (α) of all corners are also computed from the line parameters. Covariance of the corner can be obtained from line covariances. Similar as line extraction, formulas to calculate the covariance of the corner from the information of the lines have been well derived in [57] and will not be restated here.

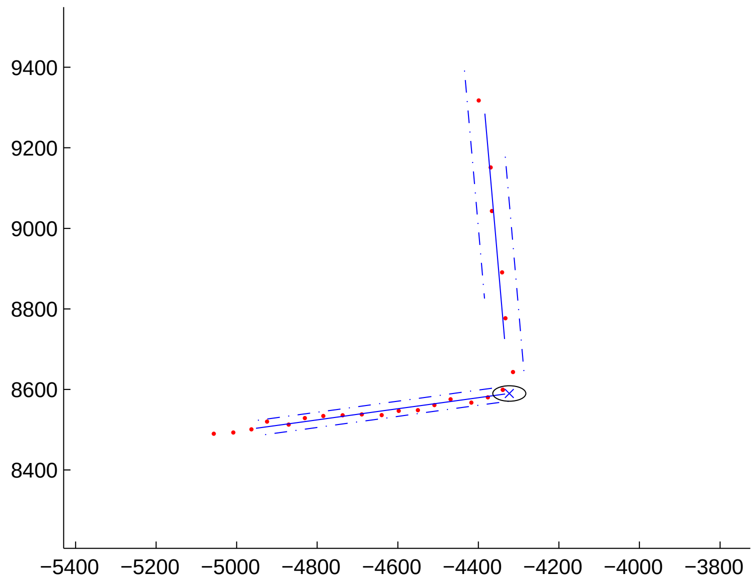


Figure 8.7: Line feature and corner feature with 3-sigma uncertainty region

The algorithm was implemented in MATLAB and laser scanner data recorded by the on-board avionics system of a quadrotor UAV was used for off-line verification, and the result is shown in Figs. 8.6–8.7. One can see that the feature extraction algorithm can successfully capture all possible lines and corners, even for line features that are only a few centimeters apart.

8.3.3 Motion Estimation and Proposal Generation

This step gives a rough prediction about the UAV motion from the previous frame to the current frame. Concurrently, the uncertainty caused by this motion is propagated. The predicted body-frame displacement of the UAV position (Δx_t, Δy_t) and heading (Δc_t) are assumed to be random variables on their own and distributed normally around their expected values. So,

$$\Delta c_t = \mathcal{N}(\Delta\bar{c}_t, \sigma_{\Delta c}), \tag{8.28}$$

$$\Delta x_t = \mathcal{N}(\Delta\bar{x}_t, \sigma_{\Delta x}), \tag{8.29}$$

$$\Delta y_t = \mathcal{N}(\Delta\bar{y}_t, \sigma_{\Delta y}), \tag{8.30}$$

where $\Delta\bar{x}_t$, $\Delta\bar{y}_t$, $\Delta\bar{c}_t$ are the expected values and $\sigma_{\Delta x}$, $\sigma_{\Delta y}$, $\sigma_{\Delta c}$ are their respective standard deviations which can be defined by the following equations:

$$\sigma_{\Delta c} = \alpha_1|\Delta\bar{c}_t| + \alpha_2, \tag{8.31}$$

$$\sigma_{\Delta x} = \alpha_3|\Delta\bar{x}_t| + \alpha_4, \tag{8.32}$$

$$\sigma_{\Delta y} = \alpha_3|\Delta\bar{y}_t| + \alpha_4. \tag{8.33}$$

α_1, α_2, α_3, α_4 denote proportional and additive noise parameters that can be tuned for practical implementations.

There are various ways to obtain the average motion estimation, i.e., to calculate $\Delta\bar{x}_t$, $\Delta\bar{y}_t$, $\Delta\bar{c}_t$. One solution is to use the UAV control input (value of PWM signals fed to motors or servos) with a precise UAV dynamic model. However, this method relies too much on the accuracy of the UAV motion model and different UAVs usually have very different model structures and parameters. That means the implemented solution will be only suitable for one particular UAV, while a lot of things need to be re-formulated and re-tuned if porting to another platform. The second solution is to use motion estimation from another sensors, such as that provided by IMU dead reckoning or a stand-alone vision system. However, there will be a lot of miscellaneous problems like sensor synchronization, inter-system communication delay, etc. that need to be solved. Meanwhile, we prefer using the scanning laser range finder as the single sensor for the problem of SLAM so that the solution keeps simple and platform independent.

Estimating the displacement of two adjacent scans from laser scanner is also referred to as the scan matching problem. This problem has been looked into by many researchers. The available solutions can be categorized into two types, according to whether the displacement is calculated from features extracted from the scan or solely from the raw data. For the raw data scan matching, the most widely used solution is the ICP algorithm. ICP tries to minimize the distance between two sets of points iteratively and for every iteration, it assumes that the closest point pairs in the two consecutive scans are the same point in the real environment. The ICP method is capable of producing very accurate results. There are published works, in which the motion estimation produced by the ICP alone can yield quite satisfying results, even in a 3 dimensional space with 6 DOF [46]. However, the drawback of the ICP method is its expensive computational load. In contrast, the feature based scan matching algorithms usually need much less computational power. Moreover, making use of the features extracted in the previous feature extraction step can even further reduce the computational time. However, this will strengthen the correlation between displacement estimation and the measurement update, as essentially we are using the same information to predict and update. Such strong correlation might result in overconfidence in the estimated pose of the UAV. However, since real-time implementation will be the ultimate goal, the feature based scan matching approach is chosen despite sacrificing a bit to the overall performance.

Therefore, the corner features extracted in two consecutive frames will be corresponded and a closed-form calculation can be used to estimate the UAV motion, namely a rotation $\mathbf{R}$, followed by a translation $\mathbf{T}$. The formulation goes as follows:

1. Check corner feature correspondences based on their pair-wise Mahalanobis distances.
2. Organize corner features in such a way that p_i or $[px_i \;\; py_i]^{\mathrm{T}}$ in the previous frame corresponds to q_i or $[qx_i \;\; qy_i]^{\mathrm{T}}$ in the current frame.
3. Calculate the centroid of the feature points for both frames, denoted by $\bar{p}$ and $\bar{q}$.
4. Form matrix $P = [p_1 - \bar{p}, \;\; p_2 - \bar{p}, \;\; \ldots \;\; , \;\; p_{\text{max}} - \bar{p}]$.
5. Form matrix $Q = [q_1 - \bar{q}, \;\; q_2 - \bar{q}, \;\; \ldots \;\; , \;\; q_{\text{max}} - \bar{q}]$.
6. Let $[U, S, V] = \text{SVD}(PQ^{\mathrm{T}})$ and $d = \text{sign}(\text{det}(PQ^{\mathrm{T}}))$,
7. Then $\mathbf{R} = V[1 \;\; 0; \;\; 0 \;\; d]U^{\mathrm{T}}$ and $\mathbf{T} = R\bar{p} - \bar{q}$, that leads to $\Delta\bar{c}_t = -\text{atan2}(\mathbf{R}(1), \mathbf{R}(2))$, $\Delta\bar{x}_t = \mathbf{T}(1)$, $\Delta\bar{y}_t = \mathbf{T}(2)$.

As mentioned previously, the motion estimation result will be applied to all particles with random additive and multiplicative noises. So for particle $P^{[m]}$,

$$\begin{aligned} \Delta c_t^{[m]} &= \Delta\bar{c}_t(1 + \alpha_1 \text{randN}(1)) + \alpha_2 \text{randN}(1), && (8.34) \\ \Delta x_t^{[m]} &= \Delta\bar{x}_t(1 + \alpha_3 \text{randN}(1)) + \alpha_4 \text{randN}(1), && (8.35) \\ \Delta y_t^{[m]} &= \Delta\bar{y}_t(1 + \alpha_3 \text{randN}(1)) + \alpha_4 \text{randN}(1), && (8.36) \end{aligned}$$

where randN(1) represents a function that can generate a random value from a standard normal distribution, and the updating equations are as follows:

$$P_t^{[m]}.c = P_t^{[m]}.c + \Delta c_t^{[m]}, \tag{8.37}$$

$$P_t^{[m]}.x = P_t^{[m]}.x + \cos(P_t^{[m]}.c)\Delta x_t^{[m]} - \sin(P_t^{[m]}.c)\Delta y_t^{[m]}, \tag{8.38}$$

$$P_t^{[m]}.y = P_t^{[m]}.y + \sin(P_t^{[m]}.c)\Delta x_t^{[m]} + \cos(P_t^{[m]}.c)\Delta y_t^{[m]}. \tag{8.39}$$

8.3.4 Per-particle Data Association

The objective of this step is to find data pairs that associate contemporary measurement features with the existing features in the map. As usual, the confidence of all the data association pairs will be calculated. Unlike other SLAM algorithms such as EKF-SLAM or GraphSLAM, FastSLAM performs data association on each of the particles instead of on the entire frame. Particle-wise data association has both advantages and disadvantages. It brings extra robustness to the algorithm, especially when particles are widely dispersed in space. However, repeating the data association algorithm many times limits the complexity / dimensions that this algorithm can handle. There is a wide range of data association algorithms available, among which the following 3 classes are the most popular.

The first class considers each feature independently. It aims to maximize the probability of associating map features with measured features without excluding repeated matches. The individual compatibility nearest neighbor (ICNN) algorithm is one of the famous examples. On the other hand, the second class makes sure the associations are consistent in a sense that no duplicated associations can be possibly made for the same feature. Such algorithms include sequential compatibility nearest neighbor (SCNN) and joint compatibility branch and bond (JCBB). The third class of data association algorithms takes one step further. It considers data associations in the previous iteration. In other words, it tries to maximize the whole probability history. When making data associations, previous decisions are examined and modified. Examples of such algorithms include the tree-structured searching algorithm developed in [30].

Not surprisingly, algorithms of lower complexity does not produce as robust results as compared to that by complex algorithms. Hence, choosing an algorithm that balance well between computational load and robustness is a curtail task in the entire implementation of the SLAM algorithm. At the moment, the JCBB method is used for data association in this work, as it provides a robust yet efficient solution.

8.3.5 Per-particle Measurement Update

Since every particle has its own map and the estimation of landmark features in each map is conditioned on the corresponding particle's path, there are $(N_c + N_l)$ low-dimensional EKFs for each particle, where N_c and N_l are the number of corner features and number of line features in the map respectively. Moreover, we need to do map update for all M particles, that means there are in total $(N_c + N_l) \times M$ EKFs, and for each of them, the updating rule is as follows (line features and corner features can be updated with the same formula):

$$Z_{n,t} = \Sigma^{[m]}_{n,t-1} + \Sigma^{z}_{n,t}, \tag{8.40}$$

$$K^{[m]}_{n,t} = \Sigma^{[m]}_{n,t-1} Z^{-1}_{n,t}, \tag{8.41}$$

$$\mu^{[m]}_{n,t} = \mu^{[m]}_{n,t-1} + K^{[m]}_{n,t}(z_{n,t} - \hat{z}_{n,t}), \tag{8.42}$$

$$\Sigma^{[m]}_{n,t} = (I - K^{[m]}_{n,t})\Sigma_{n,t-1}. \tag{8.43}$$

8.3.6 Particle Importance Weighting and Resampling

Samples from the proposal distribution, i.e., particles after motion update, are distributed according to $p(s^t|z^{t-1}, u^t)$, where x^t means all the time history of x: $\{x_1, x_2, \ldots, x_t\}$. This distribution most likely does not match the posterior probability $p(s^t|z^t, u^t)$. Importance weighting is to correct this difference by giving each particle a weight according to their probability of observing z_t at the time step t. So in a Gaussian distribution case,

$$w^{[m]}_t = \frac{1}{\sqrt{2\pi Z_{n,t}}} \exp\left\{-\frac{1}{2}(z_{n,t} - \hat{z}_{n,t})^{\mathrm{T}}[Z_{n,t}]^{-1}(z_{n,t} - \hat{z}_{n,t})\right\} \tag{8.44}$$

After the particles have been assigned their corresponding weights, a new set of samples can be drawn from the original set with probabilities in proportion to their weights. There are various ways to do this resampling process. Among them, the following algorithm does the job and it is very efficient:

1. Calculate W, the total weight of all M particles.
2. Generate a random number W' between 0 and W.
3. Deduct W by $w^{[1]}_t$, $w^{[2]}_t$, $w^{[3]}_t$, …, $w^{[i]}_t$ one by one until the result hits W'.
4. Particle i in the old set will be chosen as one element in the new set.
5. Repeat Step 2 to Step 4 M times to generate a whole new set of M particles.

8.4 Implementation Results

The proposed FastSLAM algorithm has been coded in MATLAB and off-line processed based on a sequence of laser scanner data logged while manual flying the quadrotor platform. Fig. 8.10(a) - 8.10(f) shows six moments of the SLAM results. For each moment, the left sub-figure shows the UAV body-frame laser scanner raw data and the extracted features (line segments and corners). Line segments are colored in blue, while the corners are indicated by short green lines with a number beside to show its pointing direction. The right sub-figure shows the UAV pose and the map building in progress. The UAV pose is represented by a red cross, which is from the highest weighted particle. The blue cloud of crosses around it are the possible UAV poses from the other particles. It can be seen that, the green line segments and blue corners naturally form a vivid map of the indoor environment with straight walls and sparsely distributed pillars. The result is much better than the case of dead reckoning from point cloud ICP method, which was introduced in the last chapter (see Fig. 8.8) or dead reckoning from feature based motion estimation only (see Fig. 8.9).

However, there is an obvious issue from the reconstructed map. There are repeated line features and corner features. This problem not only makes the map noisy, but also complicates the computation by introducing unnecessary number of features. By investigation, it is believed that overconfidence in the motion estimation and feature extraction may cause unmatched features despite their same identity in the real environment. In consequence, the algorithm assumes that these unmatched features are new to the reconstructed map. This can be solved by relaxing the covariance calculation for both feature extraction and motion estimation or decrease the feature matching threshold. However, more faulty matches (match two physically different features into the same one) may occur which will degrade localization and mapping result in another way. A better solution is to implement a map management function so that features in the map are periodically checked and combined according to general knowledge about the indoor environment.

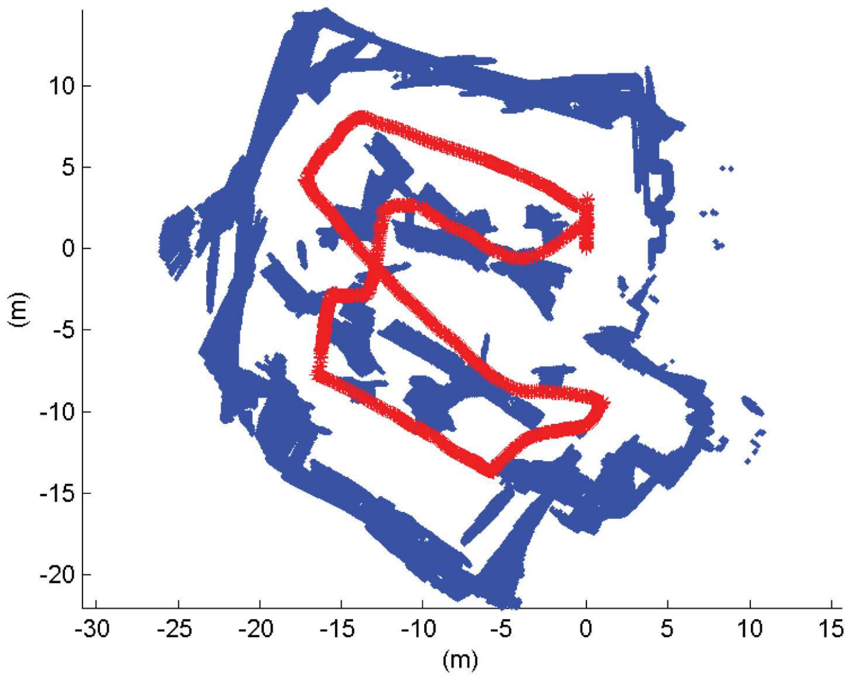


Figure 8.8: SLAM results via point cloud ICP

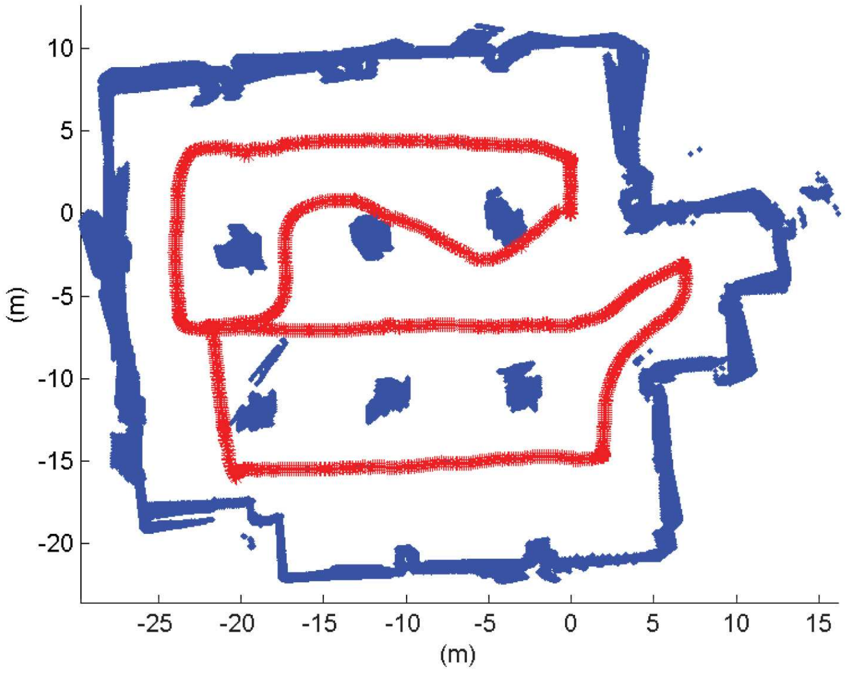
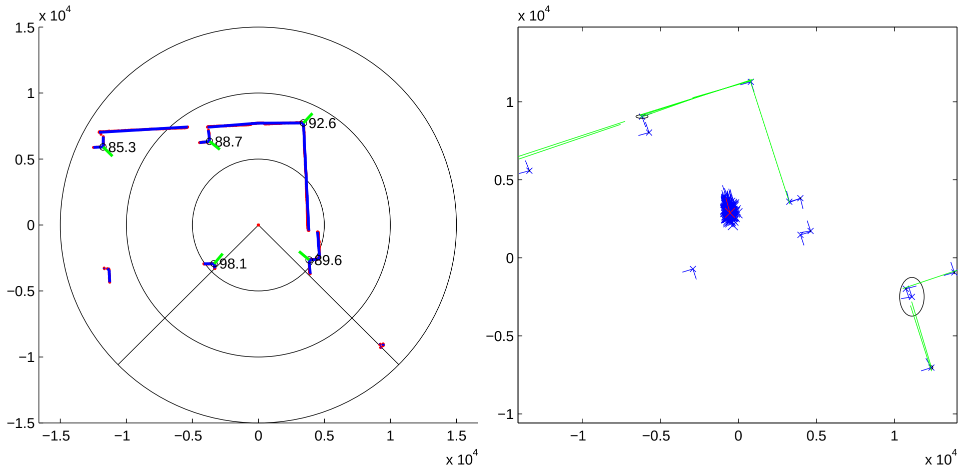
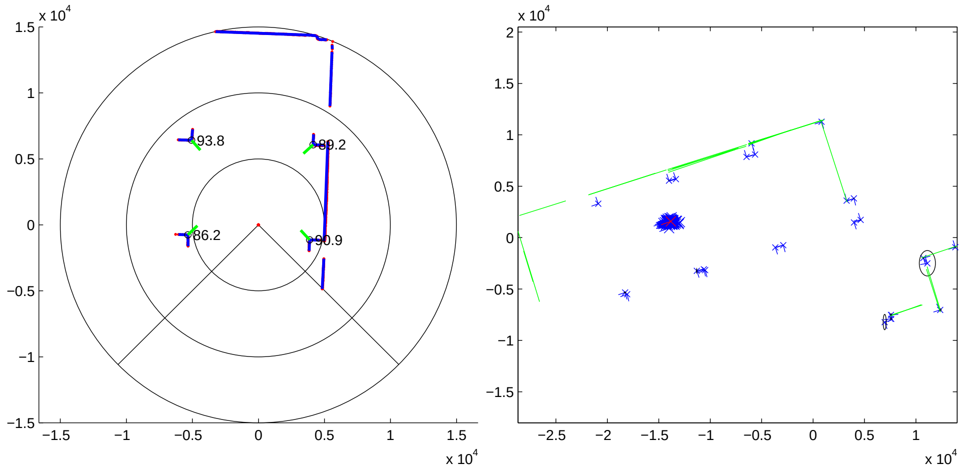
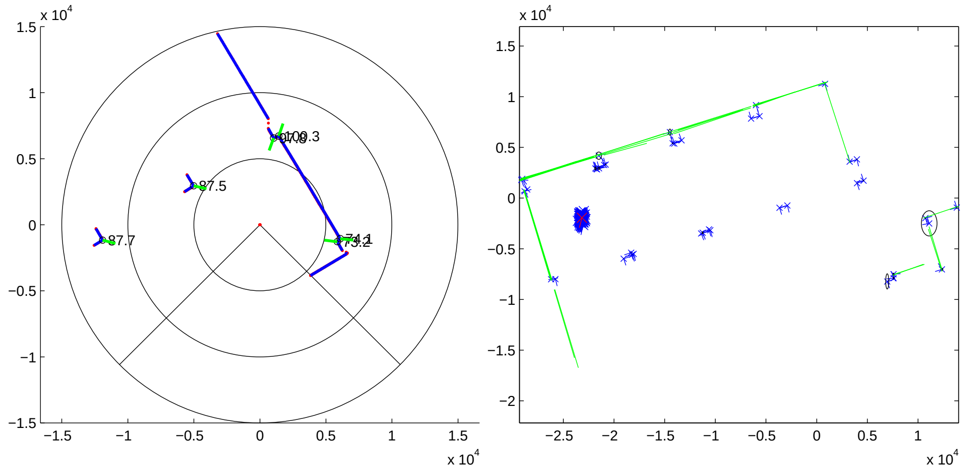
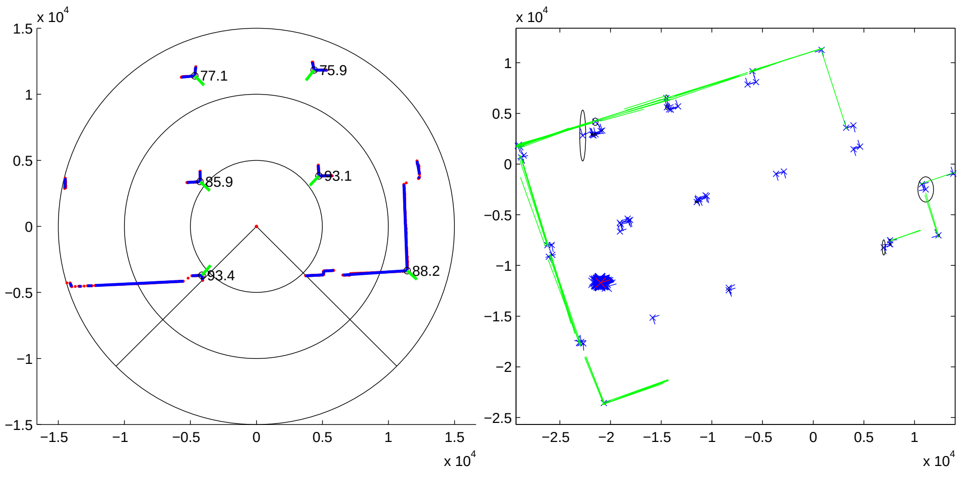


Figure 8.9: SLAM results via feature-based scan matching

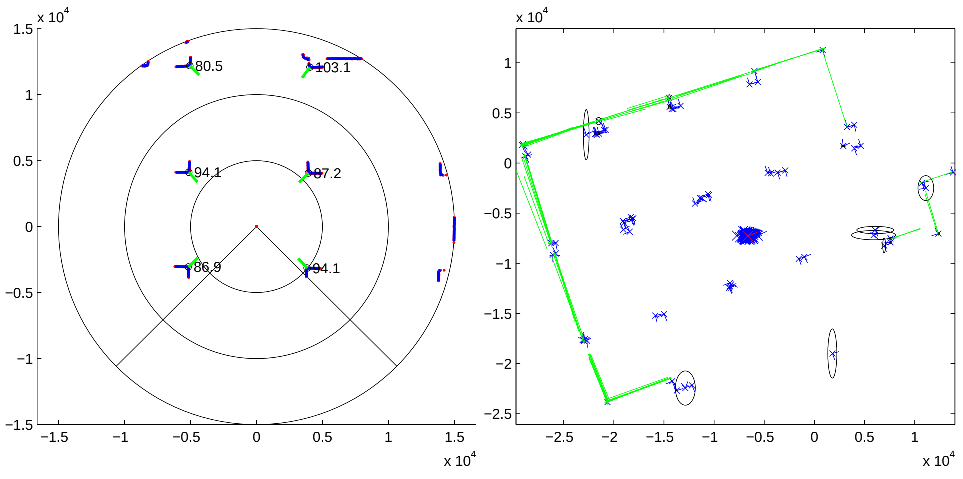

(a) Moment 1


(b) Moment 2

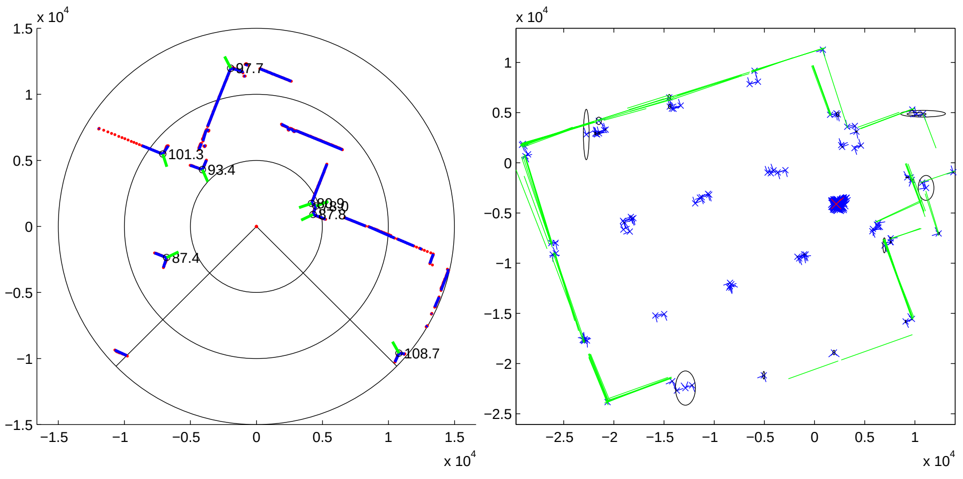

(c) Moment 3



(d) Moment 4



(e) Moment 5



(f) Moment 6

Figure 8.10: The customized FastSLAM result in an indoor hall with pillars

Chapter 9

Efficient Laser SLAM for Partially Known Indoor Environments

9.1 Background and Motivation

The previous chapter has discussed about the regular SLAM techniques which can be used by robots or autonomous vehicles to build up a map within an unknown indoor environment and at the same time to keep track of their own positions. In fact, many theoretical works and practical implementations of SLAM on ground robots [56, 89], and on UAV platforms [28, 49, 85] have been published in literature. However, few of these works have been considering the computation limitation on miniature indoor UAVs and they usually exploit the unlimited payload on ground robots or rely on high-bandwidth communication to the GCS where a powerful computer is running the most computationally intensive algorithm. In consequence, some of them only work in controlled lab environments with short and line-of-sight communication. But for real-life applications in which ideal communications cannot be guaranteed, the performance is expected to be poor. Although being relatively efficient already, the customized FastSLAM method mentioned in the previous chapter is still off-line so far. By utilizing a more powerful onboard processor or further optimizing the code, it may be able to run onboard of the UAV in real time. However, the expected difficulty level is still high, thus it will only be tried for future studies.

That being said, a more practical and robust navigation strategy should only rely on the UAV onboard computers for all necessary control and navigation functions. A few research groups are working towards this direction. In [47], an innovative laser-pointer-aided vision system is proposed to release the high computational load from dense image processing. [83] has demonstrated the possibility of real-time visual-inertial state estimation via a 1.6 GHz Atom computer onboard of the controlled UAV. In [25], hardware configuration has been optimized to achieve a highly efficient vision navigation system. The impressive work in [70] has pushed UAV onboard intelligence to the limit where a rather complicated indoor environment can be handled. Nevertheless, there must be a compromise between the complexity of the navigation algorithm and the complexity of the navigated environment under the current microprocessor technology.

In this chapter, it is intended to solve the indoor navigation problem solely onboard of the indoor UAV flying in a structured indoor environment. The algorithm can be designed very efficient because three assumptions about the indoor environment are made:

1. The environment can be described by sparse features, which include corners and straight lines;
2. The line features are orthogonal to each other or off-set by multiples of a constant angle displacement, such as 30° or 45°.
3. The coordinates of the corner features are known.

These assumptions appear to be strong but they still cover quite a lot of real-life conditions. First of all, Assumption 1 and 2 are usually met for indoor environments in modern man-made buildings. Moreover, the proposed algorithm will work as long as the majority of corner and line features in the target environment fulfills the assumptions. A few map noises will not affect the performance too much. Although Assumption 3 makes the algorithm not suitable for advanced tasks such as exploring a completely unknown environment, missions like UAV autonomous surveillance and patrolling are still doable if minimal information about the indoor environment is known. Nevertheless, the main advantage of the proposed method lies in its efficiency. With the three assumptions met, the UAV localization algorithm can be designed in an innovative way so that an ARM-based embedded computer is more than enough to handle the computation.

9.2 Efficient Localization for Partially Known Map

As stated previously, the main advantage of the proposed navigation algorithm lies in its efficiency. With Assumption 1,2,3 and the initial state of the UAV given, a feature matching based localization algorithm can be implemented to track the UAV pose in real time.

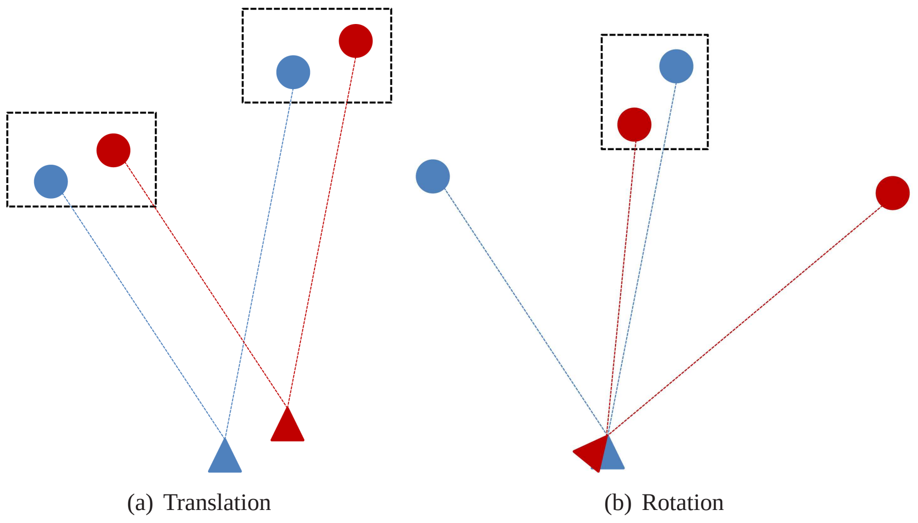


Figure 9.1: Feature matching result after a small motion

The UAV pose in the map frame can be represented by its 3-D coordinates x, y, z and heading angle ψ. Moreover, to differentiate the localization results from their respective sensor sources, we partition the UAV pose into two parts, namely the planar pose (x, y, ψ), and the vertical height z. The first part can be estimated by the horizontal scanning laser range finder, similar to a 2-D ground robot case, while the altitude of the UAV can be estimated by the second laser scanner.

9.2.1 Planar Localization

The planar localization algorithm via the first laser scanner contains the fundamental ideas that make the whole navigation solution fast and efficient. With Assumption 1, the conventional point cloud matching algorithm can be avoided, leaving the number of point matching pairs single digits as compared to the original thousands. With Assumption 2, the estimation of rotational motion can be done by comparing the difference between line gradients instead of relying on point feature matching, thus making the estimation of rotational motion decoupled from translational motion. This decoupling feature is very beneficial because rotational motion usually results in inconsistent point matching results, especially when the feature points are far away from the sensor source. From Fig. 9.1, one can see that the point matching result is correct in the first case which involves a small translation, but becomes totally wrong in the second case which involves a small rotation. As the method used in this paper estimates the rotational motion robustly and independently from the translational motion, the next stage point association and localization will have very stable performance.

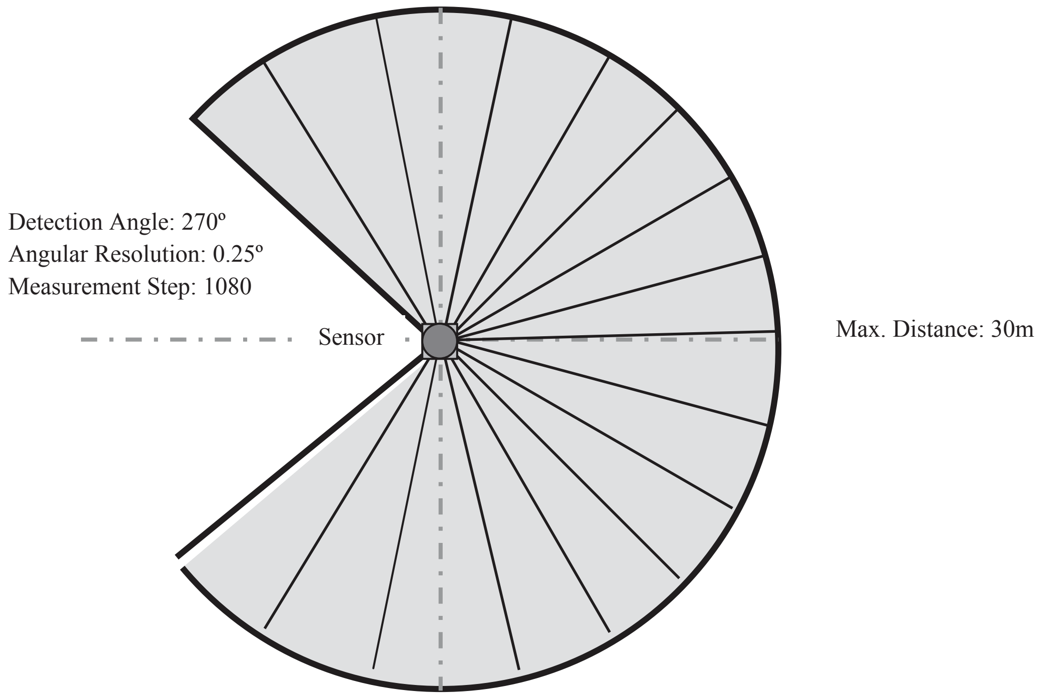


Figure 9.2: Hokuyo UTM-30LX laser range sensor

The planar localization algorithm will be explained in four steps, which include feature extraction, rotation tracking, corner feature association and position tracking.

Feature Extraction

The laser scanner used for this planar localization algorithm is a Hokuyo UTM-30LX sensor. Its specifications are shown in Fig. 9.2. For each frame of scanned data, the sensor will output 1081 integer numbers to represent the measured distances in millimeter from the rightmost angle to the leftmost angle sequentially. Each distance data is associated with its own angle direction, thus the data can be seen as in polar coordinates. A simple transformation can be applied to the raw measurement data to convert it from polar coordinates (r_k, θ_k) to Cartesian coordinates (x_k, y_k):

$$\begin{cases} x_k &= r_k \cos\theta_k \\ y_k &= r_k \sin\theta_k \end{cases} .$$

Then the *split-and-merge* algorithm [12] is applied to these array of 2-D points so that they can be grouped into clusters with each cluster belonging to a straight line feature. Here, the main steps of *split-and-merge* algorithm is summarized below with Fig. 9.3 giving a graphical

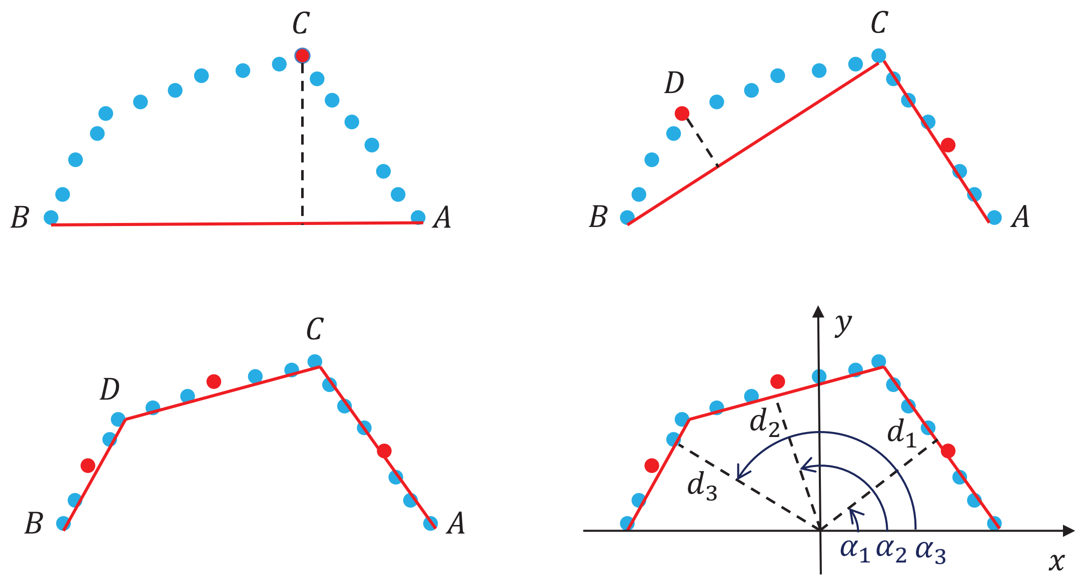


Figure 9.3: The *split-and-merge* and line extraction algorithm

illustration:

1. Connect the first point A and the last point B of the input data by a straight line.
2. Find point C among all data points that has the longest perpendicular distance to the line AB.
3. If this longest distance is within a threshold, then a cluster is created with points in between A and B.
4. Else, the input points will be split into two subgroups, A-C and C-B. For each group, the *split-and-merge* algorithm will be applied recursively.

After obtaining the clusters of points, two choices of line extraction methods can be used. The first is to use least square line fitting by considering all points in the cluster, while the second is to simply connect the first point and the the last point. Although the second method looks a bit harsh, these two methods surprisingly result in more or less the same quality of line features in a clean and structured indoor environment, thanks to the laser scanner's superior range accuracy and angular resolution. The second method actually triumphs in computational time and it is finally chosen as the way to get the line features. By convention, each line can be represented by two parameters, namely the line's normal direction α_k and its perpendicular distance to the center of laser scanner d_k. In the last sub-figure of Fig. 9.3, xy axes represent the laser scanner frame. Normal direction of the line is defined as the angle from the x-axis to the line normal, counterclockwise as positive.

Rotation Tracking

In this step, Assumption 2 will be utilized in an innovative way to keep track of the robot's heading direction ψ, which is defined as the angle from the map frame x-axis to the heading direction of the UAV, counterclockwise as positive if viewed from above. Without loss of generality, let the map frame x-axis align with one of the walls in the indoor environment. Then all the walls will have their directions at $n\alpha$, where α is the constant angle displacement and n can be any integers. Choose one of the walls currently observable and let its direction be β_l in the laser scanner frame. Then we have this wall's map frame direction β_m as:

$$\begin{aligned} \beta_m &= \psi_t + \beta_l \\ &= \psi_{t-1} + \Delta\psi_t + \beta_l \\ &= n_i\alpha. \end{aligned}$$

where ψ_t and ψ_{t-1} are the UAV headings in the current frame and previous frame respectively and $\Delta\psi_t$ is the inter-frame heading movement. Obviously, $(\psi_{t-1} + \Delta\psi_t + \beta_l)$ is divisible by α, which leads to

$$\Delta\psi_t = -\,[\,(\psi_{m,t-1} + \beta_l)\,\%\,\alpha\,], \tag{9.1}$$

where the operator $\%$ is defined in this paper as:

$$a\%b = \begin{cases} (a \mod b) & , \text{ if } (a \mod b) \le b/2 \\ (a \mod b) - b & , \text{ otherwise} \end{cases} \tag{9.2}$$

After obtaining $\Delta\psi_t$, the UAV heading can be updated as

$$\psi_{m,t} = \psi_{m,t-1} + \Delta\psi_t. \tag{9.3}$$

Using the above method, the UAV heading is tractable frame by frame provided that the initial heading $\psi_{m,0}$ is known. However, it should be noted that this heading tracking algorithm only works when the UAV inter-frame rotational motion is less than $\alpha/2$. Fortunately, a 10 Hz

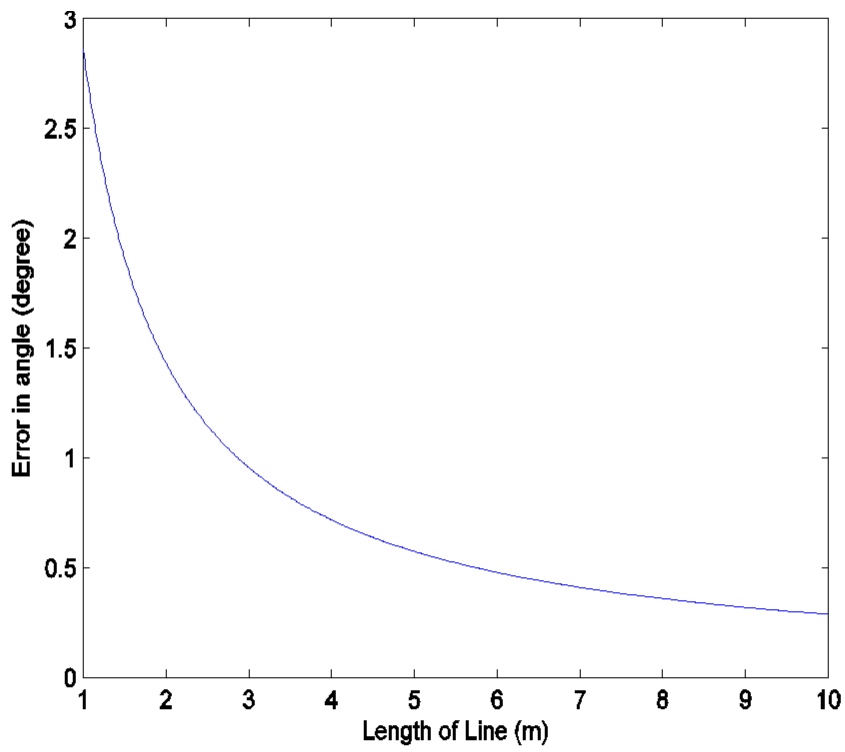


Figure 9.4: Heading error versus the length of the line

laser scanner is fast enough to handle the non-aggressive flight cases. In actual implementation, the longest line extracted for the current frame can be used for the heading alignment. This is because the error of extracted line gradient due to inaccurate end points is smaller if the line is longer. The theoretical relationship between the angle error and the length of the line being referenced is shown in Fig. 9.4.

Point Feature Association

The end points of the line clusters can be treated as local point features, in which some of them should physically associate with the known map corners. The next step is to associate these local point features to the globally known map features. This can be done by transforming the locally observed point features to the global map frame based on the information of previous-frame UAV position $[x_{t-1}, y_{t-1}]$ and the current-frame UAV heading ψ_t. As the UAV rotational motion has been resolved, the difference between the obtained feature points and the known map feature points should be caused by translational motion only. By considering the fact that this translational motion between frames of 10 Hz is very small, the nearest neighbor searching is more than enough to associate them well. The whole association process is done with the following steps:

1. Transform all local feature points $q_{j,l}$ in the laser scanner frame into the global map frame $q_{j,m}$ based on the UAV's current-frame heading and its previous-frame position.
2. For each transformed feature $q_{j,m}$, find its nearest map feature p_i.
3. Calculate the distance between $q_{j,l}$ and $q_{j,m}$, if the distance is within certain threshold, then an association ($n_{i,j}$) between the two feature points is created.

The 2-D transformation from the laser scanner local frame to the global map frame can be calculated as,

$$q_{j,m} = [x_{m,t-1}, y_{m,t-1}]^{\mathrm{T}} + R_t \times q_{j,l}, \tag{9.4}$$

where R_t is the rotation matrix from local frame to global frame calculated based on $\psi_{m,t}$,

$$R_t = \begin{bmatrix} \cos\psi_t & \sin\psi_t \\ -\sin\psi_t & \cos\psi_t \end{bmatrix}. \tag{9.5}$$

Position Tracking

Similar to the method in rotation tracking, the current position can be calculated based on the previous-frame position $[x_{m,t-1}, y_{m,t-1}]$ and an incremental change $[\Delta x_t, \Delta y_t]$:

$$[x_{m,t}, y_{m,t}] = [x_{m,t-1}, y_{m,t-1}] + [\Delta x_t, \Delta y_t], \tag{9.6}$$

where

$$\begin{bmatrix} \Delta x_t \\ \Delta y_t \end{bmatrix} = \frac{\sum\limits_{n_{i,j}} w_j (p_i - q_{j,m})}{\sum w_j}. \tag{9.7}$$

This incremental change can be calculated as an average displacement of all the associated features. By considering the laser scanner noise model, i.e. points further away are more noisy, the matched point features are given different weights w_j in calculating the average displacement. The closer the feature points, the larger the weight.

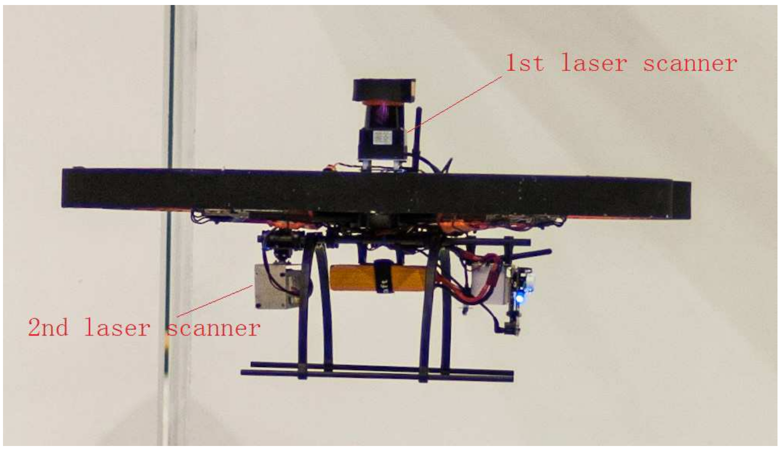


Figure 9.5: The dual laser scanner setup

9.2.2 Height Estimation

In an indoor environment with completely flat ground, UAV height measurement can be simply obtained via a sonar or a one-point laser range finder. However, for the cases when the UAV needs to fly over tables, chairs and window sills, these sensors will fail as the UAV cannot distinguish between the actual floor surface and the surfaces of other objects underneath. Barometer may be a candidate, but its accuracy does not meet the requirement for a UAV to fly in confined indoor environments. To solve this problem, a second laser scanner is mounted orthogonally to the first and a height calculation algorithm with robust floor identification is developed and integrated into the navigation system. Fig. 9.5 shows the dual laser scanner setup on the quadrotor platform.

This height calculation algorithm basically finds the furthest line parallel to the level plane and treat it as the ground. As shown in Fig. 9.2.2, the first step of the algorithm is line extraction, which can be done via the same *split-and-merge* method mentioned before. Since the obtained lines are still expressed in the laser scanner frame, their directions α_k should be compensated by the UAV pitch angle θ and compared with the normal line of the level plane. So

$$\Delta\alpha_k = \alpha_k + \theta - \pi/2 \tag{9.8}$$

If $|\Delta\alpha_i|$ is greater than a threshold, then the corresponding line is filtered out. The remaining lines are sorted by their perpendicular distances to the laser scanner and the furthest ones are kept. Among them, the longest line is believed to be the true ground. To make the estimation more accurate, this line will be recalculated using the least mean square fit by taking into account

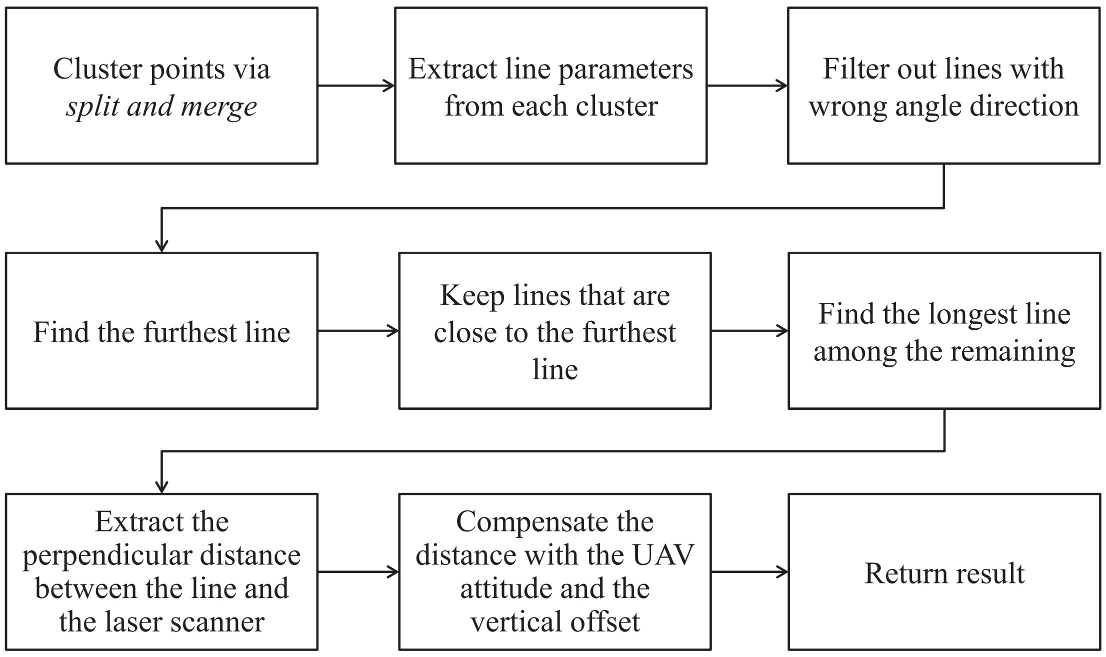

Cluster points via *split and merge*
Extract line parameters from each cluster
Filter out lines with wrong angle direction
Find the furthest line
Keep lines that are close to the furthest line
Find the longest line among the remaining
Extract the perpendicular distance between the line and the laser scanner
Compensate the distance with the UAV attitude and the vertical offset
Return result

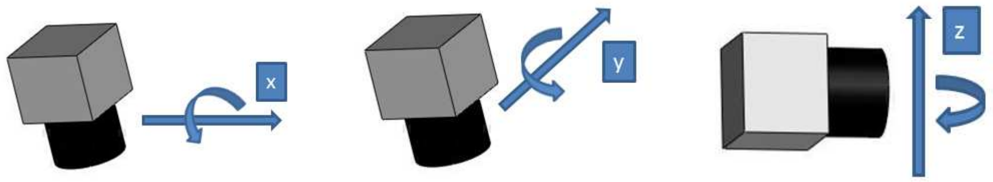
Figure 9.6: Flowchart of height estimation algorithm

all cluster points instead of only the two end points. Here, the efficiency retains as the least mean square optimization will be called only once. Finally, the perpendicular distance of obtained line to the laser scanner is compensated with the UAV roll angle ϕ, leaving the UAV height estimation to be:

$$h = r\cos(\phi) - \tilde{h} \tag{9.9}$$

where $\tilde{h}$ is the offset between the laser scanner center and the CG of the UAV. By using this method, as long as the laser scanner projects a portion of its laser beams on the true ground, a very accurate height measurement can be obtained. This basically solves the problem of scattered protruding objects on the ground when the UAV flies over.

However, using a high-end laser scanner to only estimate the height of the indoor UAV is a bit wasteful. To fully utilized its measurements, this second laser scanner is innovatively mounted on a servo motor mechanism which can be rotated in a ‘yawing’ mode (see ‘rolling’, ‘pitching’ and ‘yawing’ modes in Fig. 9.7). In this way, it can be used similar to a 3-D laser scanner and it helps to reconstruct a detailed 3-D map of the environment. The 3-D map reconstruction method will be explained in the next section.



(a) ‘Rolling’ mode (b) ‘Pitching’ mode (c) ‘Yawing’ mode (the chosen mode)

Figure 9.7: Rotating a 2-D laser scanner

9.3 3-D Map Reconstruction

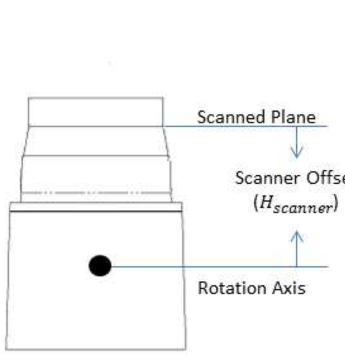
With the 3-D pose of the UAV obtained, every scan from the second rotatable laser scanner can be projected to the global frame and map reconstruction can be done in an accumulative way. The 3-D reconstruction method used here assumes that the indoor UAV localization problem has already been solved. If at one instant the UAV global position and orientation are known , and the relative position and orientation between the UAV body and the second laser scanner are also known, the points scanned by the second laser scanner can be transformed to the global frame in a rigorous way.

Except for the global frame and the UAV body frame, a third coordinate frame has to be considered here, namely the laser scanner frame. To recap the definitions, the global frame (or the map frame) is defined with its origin stationary at an arbitrarily defined position, such as the initial take-off point of the controlled UAV. Its x-axis points to the geometric north, y-axis points to the geometric east and z-axis points vertically downwards with respect to the surface of earth. The UAV body coordinate takes the UAV’s CG as origin and moves together with the UAV fuselage. Its x-axis points to the UAV heading direction, y-axis points to the right of the UAV body and z-axis points vertically down with respect to the UAV horizontal surface. For the laser scanner frame, its origin locates at CG of the laser scanner and moves together with the laser scanner body (laser scanner will be rotated by servo for a 3-D scan). Its x, y-axes are defined as that of Fig. 9.9 and its z-axis follows the right hand rule accordingly.

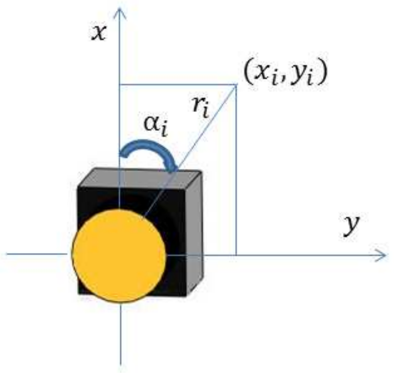
9.3.1 Transformation of 3-D Points

After defining the coordinates, converting the scanned points in their raw format to the corresponding global coordinates involve three steps, namely

1. Transforming scanned data points from polar coordinate to cartesian coordinate, both in



(a) Laser scanner side view



(b) Laser scanner coordinate definition

laser scanner frame;

2. Transforming measurement points from laser scanner frame to the UAV body frame;
3. Transforming measurement points from UAV body frame to the global frame.

Transform from Polar Coordinates to Cartesian Coordinates

Translating points in polar coordinates into points in Cartesian coordinate can be easily done as follows:

$$x_i = r_i \times \cos \alpha_i \tag{9.10}$$

$$y_i = r_i \times \sin \alpha_i \tag{9.11}$$

$$z_i = -H_{\text{scanner}} \tag{9.12}$$

with all symbols defined in Fig. 9.8(b).

Transform from Laser Scanner Frame to UAV Body Frame

The point cloud can then be rotated and translated into the UAV body frame. Rotation and translation can be done by multiplying the position vector $(x_i, y_i, z_i)^{\text{T}}$ by a rotational matrix (R) and adding a translational vector (T), expressed as follows:

$$(x_i, y_i, z_i)^{\text{T}}_{\text{b}} = R_{\text{b/s}} \times (x_i, y_i, z_i)^{\text{T}}_{\text{s}} + T_{\text{b/s}}, \tag{9.13}$$

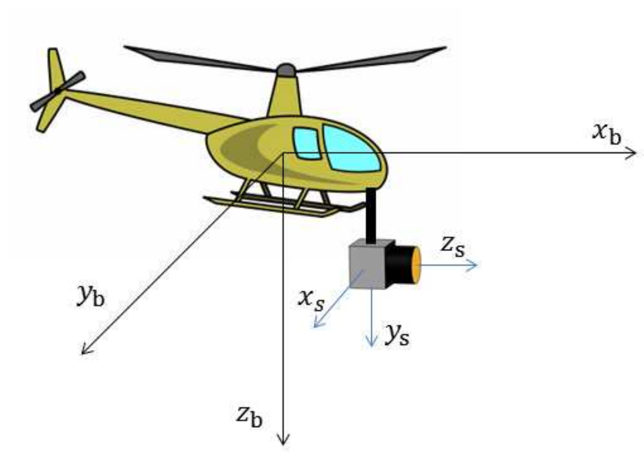


Figure 9.8: From laser scanner frame to UAV body frame

with

$$R_{\text{b/s}} = \begin{bmatrix} c_\psi c_\theta & c_\psi s_\theta s_\phi - s_\psi c_\phi & c_\psi s_\theta c_\phi + s_\psi s_\phi \\ s_\psi c_\theta & s_\psi s_\theta s_\phi + c_\psi c_\phi & s_\psi s_\theta c_\phi - c_\psi s_\phi \\ -s_\theta & c_\theta s_\phi & c_\theta c_\phi \end{bmatrix}_{\text{b/s}}$$

where ϕ, θ, ψ are the roll, pitch, yaw angles defined for the laser scanner with respect to the UAV body frame, and s_*, c_* denote $\sin(*)$, $\cos(*)$ respectively. The elements in the translational vector equals to the displacement between the origin of the UAV body frame and the origin of the laser scanner frame, expressed in the UAV body frame coordinates, that is:

$$T_{\text{b/s}} = (\Delta x_{\text{s}}\ \Delta y_{\text{s}}\ \Delta z_{\text{s}})^{\text{T}} \tag{9.14}$$

Transform from UAV Body Frame to Global Frame

Transforming 3-D points from UAV body frame to the global frame is similar to the previous step. The process is also a rigid body rotation followed by a vector translation, stated as follows:

$$(x_i, y_i, z_i)_{\text{g}}^{\text{T}} = R_{\text{g/b}} \times (x_i, y_i, z_i)_{\text{b}}^{\text{T}} + T_{\text{g/b}}, \tag{9.15}$$

with

$$R_{\text{g/b}} = \begin{bmatrix} c_\psi c_\theta & c_\psi s_\theta s_\phi - s_\psi c_\phi & c_\psi s_\theta c_\phi + s_\psi s_\phi \\ s_\psi c_\theta & s_\psi s_\theta s_\phi + c_\psi c_\phi & s_\psi s_\theta c_\phi - c_\psi s_\phi \\ -s_\theta & c_\theta s_\phi & c_\theta c_\phi \end{bmatrix}_{\text{g/b}}$$

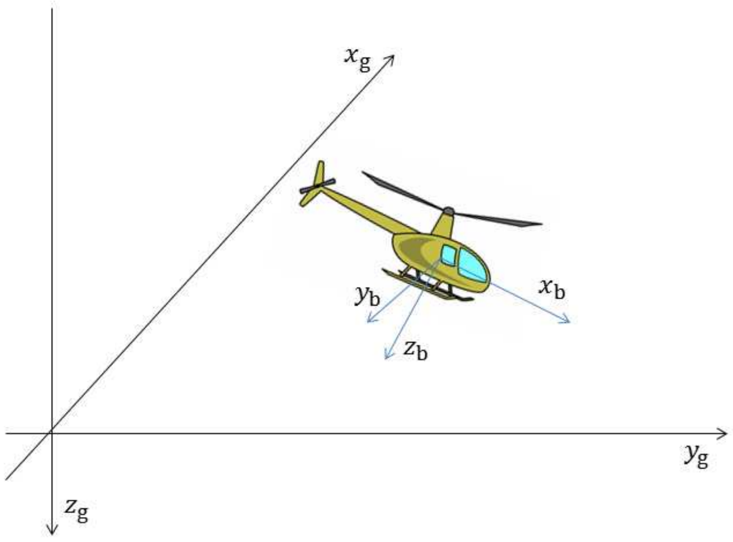


Figure 9.9: From UAV body frame to NED frame

where ϕ, θ, ψ are the roll, pitch, yaw angles defined for the UAV body respect to the global frame. The elements in the translational vector equals to the displacement between the origin of the global frame and the origin of the UAV body frame, expressed in the global frame coordinates, that is:

$$T_{\mathrm{g/b}} = (\Delta x_{\mathrm{g}}\ \Delta y_{\mathrm{g}}\ \Delta z_{\mathrm{g}})^{\mathrm{T}} \tag{9.16}$$

9.3.2 Map Representation and Management

For an indoor environment, the volume of exploration is usually limited and the worst case scenario in most cases can be foreseen. This makes a grid-based map representation plausible as the problem of memory explosion can be avoided. The grid map can be implemented by a 3-D array, with the index of an entry in this array represents a small cubical space in the environment. The volume of the cubical space is arbitrarily defined and is referred to as the resolution of the map. Obstacles in the environment can be represented by non-zero values of the entry at a particular index.

After projecting the scanned points from the laser scanner to the global coordinate frame, the points are incorporated into the 3-D map array. Points from a single scanned frame are processed sequentially. Calculating the index of the entry in the array that corresponds to the measured location is a critical step. The process can be done by dividing the x, y, z global coordinate values by the pre-defined map resolution. However, cautions must be taken for two aspects. Firstly, the index of an array can only be integers. Hence, floating point numbers resulted from the division must be rounded up or down. In addition, if the rounding process is done in a consistent manner for all directions, the product of index and resolution would recovers the furthest or nearest point of the cubical space. Secondly, as the index can only be non-negative, a constant offset need to be given to all positions of the points. This offset need to be maintained properly if the map is going to be expanded if larger area will be explored.

Updating the value of the map entry is trivial. Depending on whether the type of array is Boolean or integer, the entry can be set to 1 (true) or incremented correspondingly. The entry value can also be set as a probabilistic representation if the noise model of laser scanner measurement is used. In this way, only grids with a probabilistic value higher than a threshold are treated as obstacles and to be displayed or used for future 3-D path plan algorithm.

Choosing a proper resolution for the map is not a trivial task. Memory capacity is not the only constrain. A finer resolution would require denser points, thus more accurate sensor measurement is required and the UAV motion cannot be too fast. For a laser scanner with 4 meters' range and $0.36°$ angular resolution, adjacent points at a maximum range would be about 2.5 cm apart. If the scanner is roughly scanning perpendicular to the trajectory of the UAV at a speed of 0.5 m/s, the minimum distance between 2 points from adjacent frame would be greater than 5 cm. Based on the above calculation, 5 to 15 cm should be a reasonable range for the map resolution, depending on whether the UAV is flying slowly or fast.

9.3.3 Map Visualization

The aforementioned 3-D reconstruction method is logically simple but computationally intensive. As such, to make sure the map updating and visualization run in real time, the whole algorithm is executed on the GCS. During flight, the status of the UAV and the laser scanner data are remotely transmitted from the UAV onboard computer to the GCS at an updating rate of 10 Hz. After transforming the laser scanner raw data to the ground frame, the 3-D array representing the map is updated accordingly. At another thread, a visualization program written upon the OpenGL library is running to display the 3-D map grid by grid. For a better visualization, different colors are assigned to grids with different height values. The floor is colored in grey and from grey to green to blue, the grids being represented are higher and higher in the 3-D space. Figure 9.3.3 shows the reconstructed map as the right sub-figure, while the left sub-figure is a clean and ideal map of the environment pre-drawn as a reference. It can be seen that they agree with each other, especially when concerning about the positions of windows,

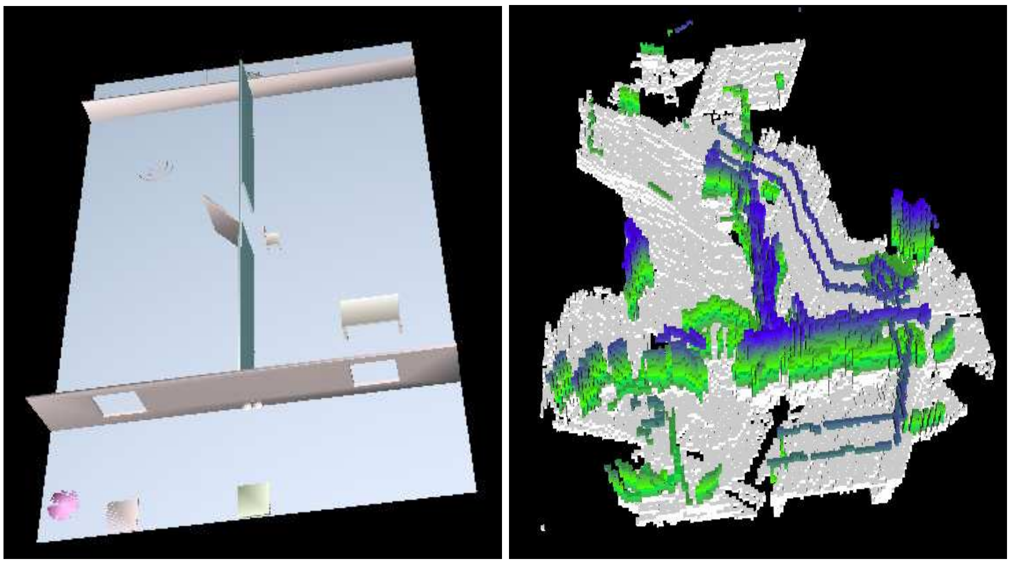

(a) The ideal 3-D map (b) The generated 3-D map

Table 9.1: Performance of the planar localization algorithm

Position (m)	RMS error (m)	Execution time (s)
$(-2,-4)$	$(0.03, 0.03)$	0.011
$(2,-5)$	$(0.03, 0.05)$	0.017
$(11,-3)$	$(0.06, 0.05)$	0.011
$(1,6)$	$(0.02, 0.03)$	0.009

door way and walls. The reconstructed map looks more noisy because the localization of the UAV platform is not perfect and there are still minor errors in the calibration of the laser scanner orientation and position with respect to the UAV body. Some obstacles in the reconstructed map are not seen in the ideal clean map, such as two pillars around the flight path. They are actually present physically.

9.4 Flight Test and Competition Results

The proposed control and navigation algorithms are implemented onboard of the quadrotor UAV. For the planar localization algorithm, the average computation time is about 12 ms for a single frame of laser scanner data. Table 9.4 shows the position error and computation time of the localization algorithm when the UAV is positioned at four stationary locations in the indoor environment. The Root-Mean-Square (RMS) error of the estimated position is very small.

To show its dynamic performance, the estimated UAV path after one complete flight is logged and shown in Fig. 9.10. It is actually the full flight path in accomplishing the 2013 Singapore Amazing Flying Machine Competition (SAFMC) Category D2 tasks. The NUS UAS

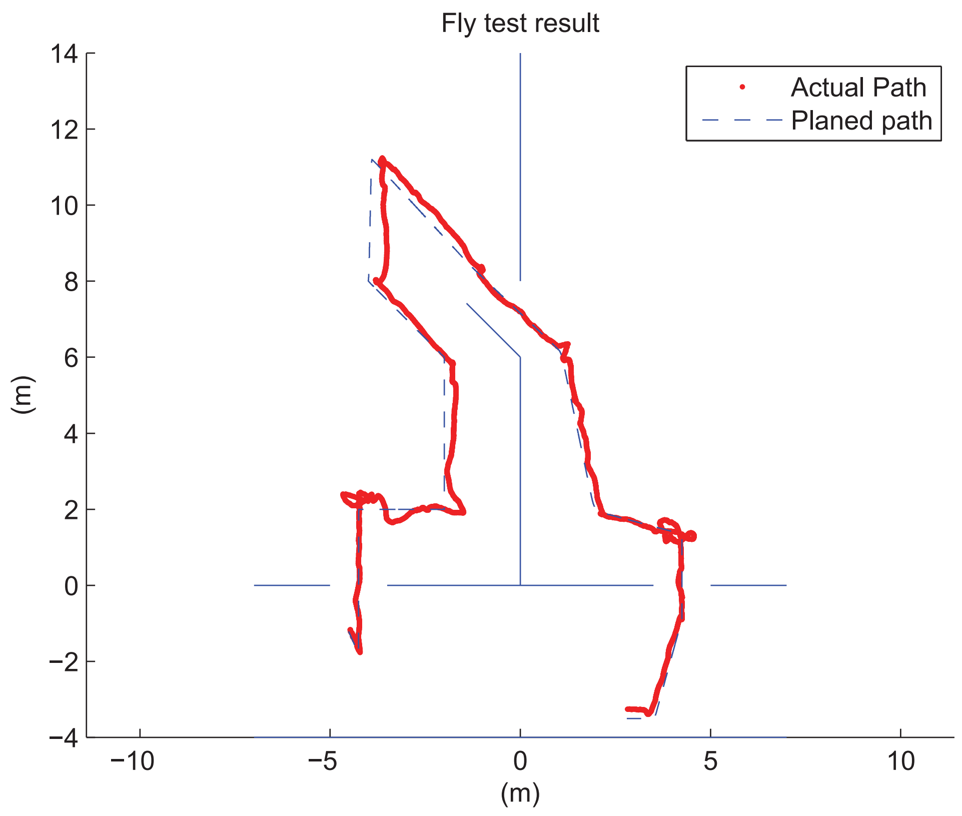


Figure 9.10: Localization result after one complete flight

Group has won all three top awards in the competition, namely the overall champion award, the best performance award, and the most creative award. Fig. 9.11 shows the overall setup of the competition. Figs. 9.12(a)–9.12(b) show two photo snaps in the competition, in which the quadrotor UAV flies through a window and drops a payload to a target location. These two tasks require very high localization accuracy as well as precise position control performance from the UAV system.

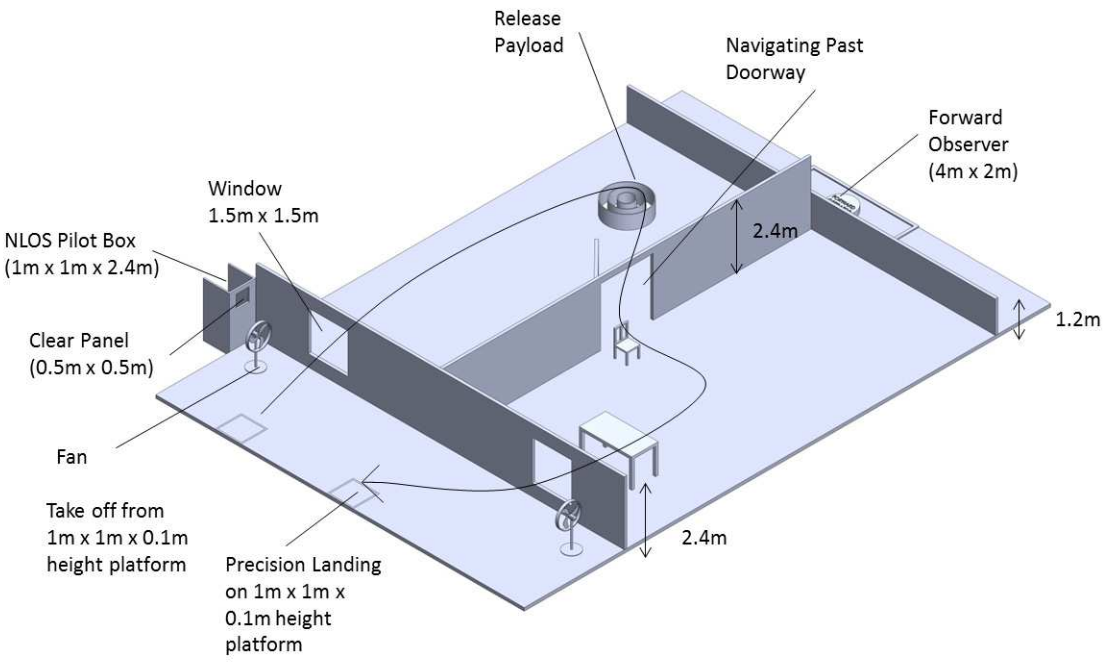
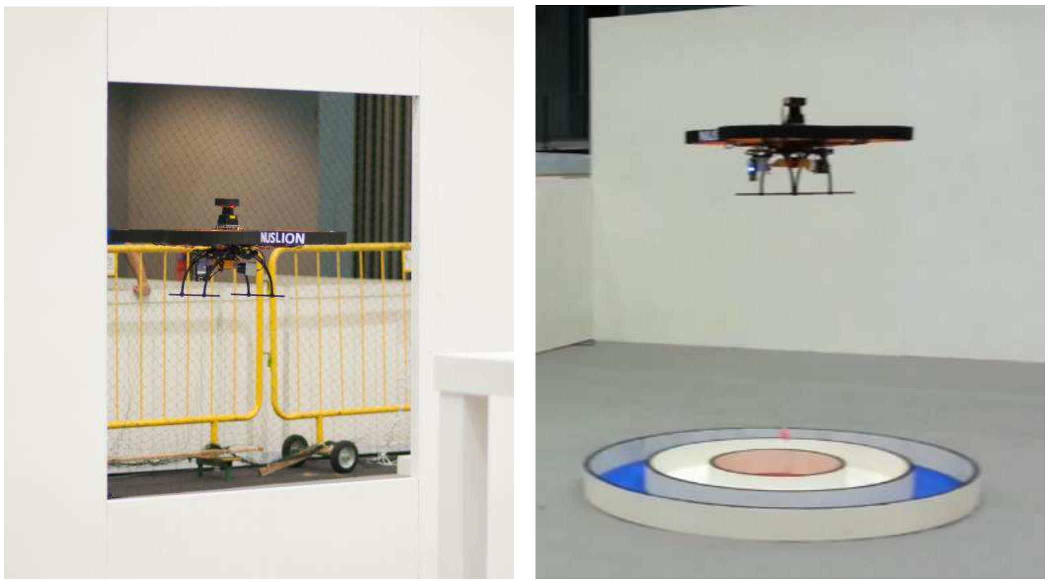

Release Payload
Navigating Past Doorway
Forward Observer (4m x 2m)
Window 1.5m x 1.5m
NLOS Pilot Box (1m x 1m x 2.4m)
2.4m
1.2m
Clear Panel (0.5m x 0.5m)
Fan
Take off from 1m x 1m x 0.1m height platform
Precision Landing on 1m x 1m x 0.1m height platform
2.4m

Figure 9.11: Competition setup in the SAFMC 2013



(a) Fly through a window (b) Drop a payload precisely

Figure 9.12: Fly-off in the SAFMC competition

Chapter 10

Conclusions and Future Works

In this thesis, the topic of UAV indoor navigation has been discussed in breadth and depth. With the general aim of establishing a comprehensive navigation system for indoor UAVs, this thesis has developed algorithms for various functions of the navigation system with consistent attention paying to onboard implementation and real-time computation. It is believed that an indoor UAV system is much more valuable if its core navigation algorithms can be executed without relying on external sensory information or external computational power.

An interesting exploration in this work is that two different types of indoor UAV platforms, namely the coaxial helicopter and the quadrotor helicopter, are constructed and their respective flight control laws and indoor navigation algorithms are implemented. Theoretically, all proposed ideas and algorithms in this thesis should be applicable to both platforms or even other types of UAV platforms. However, due to their different hardware limitations such as payload capacity, onboard sensor quality and processor computational power, implementation results obtained from the coaxial UAV do not always show robust and real-time performance, but the quadrotor UAV is stable and powerful enough to implement most of the algorithms with good performance. Nevertheless, the nonlinear coaxial helicopter model and its control method discussed in this thesis is actually a valuable contribution to the UAV modeling and control community.

In order to control the UAV at the navigation level without using GPS or other external motion capturing system, visual odometry and laser scanner based odometry have been developed. For visual odometry, two innovative methods are proposed. Both of them use only a single camera but mounted in two different orientations with respect to the UAV body. In general, the second method is currently more practical and convenient to be implemented while the first method can handle more general indoor setups but need further study if more accurate odometry is needed. In addition, it is found in both methods that by utilizing supplementary information from other sensors such as the IMU sensor and the laser scanner sensor, the vision algorithms can be largely simplified while retaining accurate and stable result. On the other hand, an ICP-based laser odometry method is also discussed in this thesis. While usable, it can only estimate 2-D motion of the UAV, and same as the visual odometry case, the problem of position drift still exists.

In order to solve or minimize the position-drift problem and at the same time to reconstruct a map for the indoor environment, studies about UAV indoor SLAM have been conducted. To overcome the limitation of the traditional KF, EKF and UKF based SLAM methods, a customized FastSLAM algorithm in cooperating both corner features and line features has been developed. By bringing the line features into the SLAM algorithm, which is not commonly seen in literature, the localization result becomes more robust and the landmark features naturally form a visually comprehensible map. However, only off-line results have been obtained so far because the logic complexity of this algorithm is high and MATLAB needs to be used first to verify its feasibility. If porting the algorithm to C++ language, the performance is expected to be real-time onboard.

Motivated by SAFMC 2013 and also trying to solve the SLAM problem in a partially known map with efficiency, an innovative localization method isolating rotational motion estimation from translational motion estimation has been proposed. Although several assumptions about the indoor environment need to be made, they are all reasonable assumptions that can be met by most modern man-made buildings. The assumptions are further verified to be reasonable as the same localization algorithm has been used in several UAV indoor demonstration events in which the indoor environments are quite different. Besides, a second laser scanner is installed onto the UAV platform orthogonally to the first to reliably estimate the UAV altitude. In this way, cases when the UAV flies over protruding objects on the ground can be handled. In addition, by rotating the second laser scanner via a servo motor, it becomes a pseudo 3-D laser scanner. When the UAV flies with its 3-D position calculated, a 3-D map of the environment can be reconstructed by accumulating scanned points by this rotating laser scanner.

Beside, to realize a complete an indoor UAV system, topics on sensor data fusion and UAV path planning are briefly explored also. A simple Kalman filter is used to fuse acceleration information from IMU, velocity information from visual or laser odometry, and position information from SLAM. A smooth estimation of the UAV position, velocity can be obtained in 50 Hz which is adequate enough for outer-loop feedback control. To let the UAV fly along the walls of an enclosed indoor room and to avoid obstacles automatically, a potential field based path planning method which only relies on local laser scanner measurements is developed and flight test has been carried out successfully.

Although all necessary functions of a UAV indoor navigation system have been developed in this thesis, there are still plenty of room for performance improvements, and some of the developed navigation functions are still not intelligent enough. The followings list a few future works that can be conducted to push this navigation system to a higher level of robustness and flexibility.

1. Although the big quadrotor platform is easier for onboard implementation of indoor navigation algorithms, the coaxial platform is still better in its form factor and energy efficiency. With more advanced sensor and processor technology in future, it may be possible to implement onboard autonomous indoor navigation on micro aerial vehicles (MAVs), which is defined as aerial vehicles with the largest dimension less than 15 cm.
2. This thesis accomplishes the so-called 2.5-D indoor navigation as it assumes all obstacles are vertically homogeneous. If the indoor environments are more complex and unstructured, the proposed algorithms will most likely fail. Hence, a true 3-D mapping and navigation solution is still open for further studies, and SLAM via 3-D laser scanner or stereo-vision could be the most promising directions.
3. The path plan algorithm used for this project is just a general wall following strategy with obstacle avoidance function. More meaningful optimization functions, such as energy, time, acceleration, and etc, can be considered to achieve better planning of 3-D trajectories. Furthermore, path planning can be formulated in a way that the target function favors the SLAM computation. If the main objective of the indoor flight is to obtain the originally unknown map, then this path-plan-SLAM-correlated formulation can make sure the map is thoroughly explored and the UAV motion planning at every time step should favor the estimation of unsure map features.
4. Although the proposed ideas and algorithms are for UAV indoor navigation, some of them can be extended to a more general navigation problem in GPS-denied environments, such as the urban canyon and the foliage cases. If a UAV navigation system can handle all

types of environments, its value and application range will be huge.

At the end of this thesis, it should also be highlighted that UAV-related research works and projects usually involve teamwork of people from different disciplines. This indoor UAV navigation system cannot be developed successfully without the help from all other members in the NUS UAS Group. On the other hand, the author of this thesis, has also been involved in UAV-related works other than indoor navigation during his Ph.D. studies. One major contribution is his involvement in the 2nd AVIC Cup - International UAV Innovation Grand Prix, which was held at the Airport of Miyun, Beijing, China, in September 2013. In this event, he lead a joint team from the NUS UAS Group and the Nanjing University of Science and Technology, and won the first place in the final round of the rotary-wing category competition. Although it was an outdoor UAV competition and the used platform was a single-rotor helicopter, similar concepts and methodologies about UAV modeling, control and navigation mentioned in this thesis can also be applied [81, 88].

Bibliography

[1] M. Achtelik, A. Bachrach, R. J. He, S. Prentice, and N. Roy. Stereo vision and laser odometry for autonomous helicopters in GPS-denied indoor environment. In *Proceedings of the SPIE Conference on Unmanned Systems Technology XI*, 2009.

[2] B. Arama, S. Barissi, and N. Houshangi. Control of an unmanned coaxial helicopter using hybrid fuzzy-PID controllers. In *Proceedings of the 24th Canadian Conference on Electrical and Computer Engineering*, 2011.

[3] K. Arun, T. S. Huang, and S. D. Blostein. Least-squares fitting of two 3-D point sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(5):698–700, 1987.

[4] F. Aurenhammer. Voronoi diagrams - a survey of a fundamental geometric data structure. *ACM Computing Surveys*, 23:345–405, 1991.

[5] A. G. Bachrach. Autonomous flight in unstructured and unknown indoor environments. Master's thesis, MIT, 2009.

[6] T. Bailey and H. Durrant-Whyte. Simultaneous localization and mapping (SLAM): Part II. *IEEE Robotics & Automation Magazine*, 13:108–117, 2006.

[7] Y. Bar-Shalom and T. Fortmann. *Tracking and Data Association*. Academic Press, San Diego, CA, USA, 1988.

[8] J. Barraquand, B. Langlois, and J. C. Latombe. Robot motion planning with many degrees of freedom and dynamic constraints. In *Proceedings of the Fifth International Symposium of Robotics Research*, 1989.

[9] H. Bay, T. Tuytelaars, and L. V. Gool. SURF: Speeded up robust features. In *Proceedings of the 9th European Conference on Computer Vision*, 2006.

[10] P. Beecher, A. Franklin, J. Gassaway, A. Oberste, and J. Wright. Virginia tech entry to the 2009 international aerial robotics competition, 2009.

[11] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18:509–517, 1975.

[12] G. Borges and M. J. Aldon. A split-and-merge segmentation algorithm for line extraction in 2D range images. In *Proceedings of the 15th International Conference on Pattern Recognition*, 2000.

[13] S. A. Bortoff. The University of Toronto RC helicopter. In *Proceedings of the 1999 IEEE International Conference on Control Applications*, 1999.

[14] A. R. S. Bramwell, G. Done, and D. Balmford. *Bramwell's Helicopter Dynamics*. Oxford, England: Butterworth-Heinemann, 2001.

[15] G. Cai, B. M. Chen, T. H. Lee, and K. Y. Lum. Comprehensive nonlinear modeling of an unmanned-aerial-vehicle helicopter. In *AIAA Guidance, Navigation and Control Conference and Exhibit*, 2008.

[16] G. Cai, F. Lin, B. M. Chen, and T. H. Lee. Systematic design methodology and construction of UAV helicopters. *Mechantronics*, 18:545–558, 2008.

[17] B. M. Chen. *Robust and H_∞ Control*. London: Springer-Verlag, 2000.

[18] B. M. Chen, T. H. Lee, and V. Venkataramanan. *Hard Disk Drive Servo Systems*. Advances in Industrial Control Series. Springer, New York, 2002.

[19] L. Chen and P. McKerrow. Modelling the lama coaxial helicopter. In *Proceedings of the Australasian Conference on Robotics and Automation*, 2007.

[20] Y. C. Chen, Y. Zhao, and H. K. Wang. Real time path planning for UAV based on Focused D. In *4th International Workshop on Advanced Computational Intelligence*, 2011.

[21] C. Coleman. A survey of theoretical and experimental coaxial rotor aerodynamic research. Technical Report TP-3675, NASA, California, 1997.

[22] Y. Deng, R. Tao, and J. Hu. Experimental investigation of the aerodynamic interaction between upper and lower rotors of a coaxial helicopter. *ACTA Aeronautica ET Astronautica Sinica*, 24:10–14, 2003.

[23] M. W. M. G. Dissanayake, P. Newman, S. Clark, H. Durrant-Whyte, and M. Csorba. A solution to the simultaneous localization and map building (SLAM) problem. *IEEE Transactions on Robotics and Automation*, 17(3):229–241, 2001.

[24] H. Durrant-Whyte and T. Bailey. Simultaneous localization and mapping: Part I. *IEEE Robotics Automation Magazine*, 13(2):99–110, 2006.

[25] S. Ehsan and K. D. McDonald-Maier. On-board vision processing for small UAVs: Time to rethink strategy. In *NASA/ESA Conference on Adaptive Hardware and Systems*, 2009.

[26] P. Fankhauser, S. Bouabdallah, S. Leutenegger, and R. Siegwart. Modeling and decoupling control of the coax micro helicopter. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011.

[27] S. G. Fowers, D. J. Lee, B. J. Tippetts, K. D. Lillywhite, A. W. Dennis, and J. K. Archibald. Vision aided stabilization and the development of a quad-rotor micro UAV. In *Proceedings of the 2007 IEEE International Symposium on Computational Intelligence in Robotics and Automation*, 2007.

[28] S. Grzonka, G. Grisetti, and W. Burgard. Towards a navigation system for autonomous indoor flying. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2878–2883, 2009.

[29] J. E. Guivant and E. M. Nebot. Optimization of the simultaneous localization and map-building algorithm for real-time implementation. *IEEE Transactions on Robotics and Automation*, 17:242–257, 2001.

[30] D. Hahnel, W. Burgard, B. Wegbreit, and S. Thrun. Towards lazy data association in SLAM. In *Proceedings of the 11th International Symposium of Robotics Research*, 2003.

[31] C. M. Harris and C. E. Crede. *Harris' Shock and Vibration Handbook 4th ed.* NY:McGraw-Hill, 1996.

[32] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004.

[33] V. Hayward. Fast collision detection scheme by recursive decomposition of manipulator workspce. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 1986.

[34] R. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME-Journal of Basic Engineering*, 82:35–45, 1960.

[35] K. Kang and J. V. R. Prasad. Development and flight test evaluations of an autonomous obstacle avoidance system for a rotary-wing UAV. *Unmanned Systems*, 1:3–19, 2013.

[36] J. Keller, D. Thakur, V. Dobrokhodov, K. Jones, M. Pivtoraiko, and J. Gallier. A computationally efficient approach to trajectory management for coordinated aerial surveillance. *Unmanned Systems*, 1:59–74, 2013.

[37] H. W. Kim and R. E. Brown. Coaxial rotor performance and wake dynamics in steady and manoeuvring flight. In *Proceedings of 62nd American Helicopter Society Annual Forum*, 2006.

[38] J. Kim and I. S. Kweon. Vision-based autonomous navigation based on motion estimation. In *Proceedings of the International Conference on Control, Automation and Systems*, 2008.

[39] K. Kondo. Motion planning with sixth degrees of freedom by multistrategic bidirectional heuristic free-space enumeration. *IEEE Transactions on Robotics and Automation*, 7:267–277, 1991.

[40] H. Lim and S. Machida. Mechanism and control of coaxial double contra-rotation flying robot. In *Proceedings of the International Conference on Control, Automation and System*, 2010.

[41] Y. C. Liu and Q. H. Dai. Vision aided unmanned aerial vehicle autonomy: An overview. In *Proceedings of the 3rd International Congress on Image and Signal Processing*, 2010.

[42] D. G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the International Conference on Computer Vision*, 1999.

[43] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 1981.

[44] Y. Ma, S. Soatto, J. Kosecka, and S. Sastry. *An Invitaion to 3D Vision*. New York: Springer, 2004.

[45] D. Mellinger and V. Kumar. Minimum snap trajectory generation and control for quadrotors. In *IEEE International Conference on Robotics and Automation*, 2011.

[46] G. Michalicek, D. Klimentjew, and J. Zhang. A 3D simultaneous localization and mapping exploration system. In *Proceedings of the International Conference on Robotics and Biomimetics*, 2011.

[47] M. K. Mohamed, S. Patra, and A. Lanzon. Designing simple indoor navigation system for UAVs. In *Proceedings of the 19th Mediterranean Conference on Control & Automation*, 2011.

[48] M. Montemerlo and S. Thrun. *FastSLAM: A Scalable Method for the Simultaneous Localization and Mapping Problem in Robotics*. New York: Springer, 2007.

[49] B. B. Moshe, N. Shvalb, J. Baadani, I. Nagar, and H. Levy. Indoor positioning and navigaton for micro UAV drones - work in progress. In *Proceedings of the IEEE 27th Convention of Electrical & Electronics Engineerings in Israel*, 2012.

[50] P. Moutarlier and R. Chatila. Stochastic multisensory data fusion for mobile robot location and environment modeling. In *Proceedings of the 5th International Symposium on Robotics Research*, 1989.

[51] P. Mukherjee and S. L. Waslander. Modeling and multivariable control techniques for small coaxial helicopters. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, 2011.

[52] D. Neamtu, R. Deac, R. D. Keyser, C. Ionescu, and I. Nascu. Identification and control of a miniature rotorcraft unmanned aerial vehicle (UAV). In *Proceedings of the IEEE International Conference on Automation, Quality and Testing, Robotics*, 2010.

[53] J. Neira and J. D. Tardos. Data association in stochastic mapping using the joint compatibility test. *IEEE Transactions on Robotics and Automation*, 17:890–897, 2001.

[54] P. Newman, D. Cole, and K. Ho. Outdoor SLAM using visual appearance and laser ranging. In *Proceedings of the IEEE Conference on Robotics and Automation*, 2006.

[55] V. Nguyen, A. Martinelli, and N. Tomatis. A comparison of line extraction algorithms using 2D laser rangefinder for indoor mobile robotics. In *International Conference on Intelligent Robotics and Systems*, 2005.

[56] A. Nuchter, H. Surmann, K. Lingemann, J. Hertzberg, and S. Thrun. 6D SLAM with an application in autonomous mine mapping. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2004.

[57] P. Nunez, R. Vazquez-Martin, J. C. del Toro, A. Bandera, and F. Sandoval. Natural landmark extraction for mobile robot navigation based on an adaptive curvature estimation. *Journal of Robotics and Autonomous Systems*, 56:247–264, 2008.

[58] A. Ollero and L. Merino. Control and perception ttechnique for aerial robotics. *Annual Reviews in Control*, 28:167–178, 2004.

[59] S. T. Pfister, S. I. Roumeliotis, and J. W. Burdick. Weighted line fitting algorithms for mobile robot map building and efficient data representation. In *International Conference on Robotics and Automation*, 2004.

[60] S. K. Phang, J. J. Ong, T. C. R. Yeo, B. M. Chen, and T. H. Lee. Autonomous mini-UAV for indoor flight with embedded on-board vision processing as navigation system. In *Proceedings of the IEEE R8 International Conference on Computational Technologies in Electrical and Electronics Engineering*, 2010.

[61] P. Pounds, R. Mahony, and P. Corke. Modelling and control of a large quadrotor robot. *Control Engineering Practice*, 18:691–699, 2010.

[62] Y. Qu, Q. Pan, and J. Yan. Flight path planning of UAV based on heuristically search and genetic algorithms. In *Proceedings of the IEEE 31st Annual Conference on Industrial Electronics Society*, 2005.

[63] O. Rand and V. Khromov. Aerodynamic optimization of coaxial rotor in hover and axial flight. In *Proceedings of the 27th international Congress of the Aeronautical Sciences*, 2010.

[64] A. Ravikanth, C. R. Raviteja, N. P. Kumar, C. Vamsimohan, R. S. Vikram, H. Rampal, G. Kashyap, M. Pradeep, and K. Kulkarni. Development of an autonomous aerial vehicle capable of indoor navigation. 2009 International Aerial Robotics Competition Entry, 2009.

[65] J. F. Roberts, T. Stirling, J. C. Zufferey, and D. Floreano. Quadrotor using minimal sensing for autonomous indoor flight. In *Proceedings of the European Micro Air Vehicle Conference and Flight Competition*, 2007.

[66] E. N. Sanchez, H. M. Becerra, and C. M. Velez. Combining fuzzy, PID and regulation control for an autonomous mini-helicopter. *Information Sciences*, 177:1999–2022, 2007.

[67] Z. Sarris. Survey of UAV applications in civil markets. In *Proceedings of the 9th Mediterranean Conference on Control and Automation*, 2001.

[68] D. Schafroth, C. Bermes, S. Bouabdallah, and R.Siegwart. Modeling, system identification and robust control of a coaxial micro helicopter. *Control Engineering Practice*, 18:700–711, 2010.

[69] D. Schafroth, C. Bermes, S. Bouabdallah, and R. Siegwart. Modeling and system identification of the muFly micro helicopter. *Journal of Intelligent and Robotic Systems*, 57:27–44, 2019.

[70] S. Shen, N. Michael, and V. Kumar. Autonomous multi-floor indoor navigation with a computationally constrained MAV. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 20–25, 2011.

[71] J. Shi and C. Tomasi. Good features to track. In *IEEE Conference on Computer Vision and Pattern Recognition*, 1994.

[72] D. H. Shim, H. Chung, H. J. Kim, and S. Sastry. Autonomous exploration in unknown urban environments for unmanned aerial vehicles. In *AIAA Guidance, Navigation and Control Conference*, 2005.

[73] R. C. Smith and P. Cheeseman. On the represnetation and estimation of spatial uncertainty. *International Journal of Robotics Research*, 5:56–68, 1986.

[74] M. Sugeno, I. Hirano, S. Nakamura, and S. Kotsu. Development of an intelligent unmanned helicopter. In *Proceedings of the 1995 IEEE International Conference on Fuzzy Systems*, 1995.

[75] A. Tayebi and S. McGilvray. Attitude stabilization of a VTOL quadrotor aircraft. *IEEE Transactions on Control Systems Technology*, 14:562–571, 2006.

[76] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, 2005.

[77] S. Thrun, Y. Liu, D. Koller, A. Y. Ng, Z. Ghahramani, and H. Durrant-Whyte. Simultaneous localization and mapping with sparse extended information filters. *The International Journal of Robotics Research*, 23:693–716, 2002.

[78] C. Tomasi and T. Kanade. Detection and tracking of point features. Technical report, Carnegie Mellon University, 1991.

[79] G. Vegter. The visibility diagram: A data structure for visibility problem and motion planning. *SWAT 90: 2nd Scandinavian Workshop on Algorithm Theory*, 447:97–110, 1990.

[80] B. Vidolov, J. D. Miras, and S. Bonnet. A two-rule-based fuzzy logic controller for contrarotating coaxial rotors uav. In *Proceedings of the IEEE International Conference on Fuzzy Systems*, 2006.

[81] F. Wang, P. Liu, S. Zhao, B. M. Chen, S. K. Phang, S. Lai, T. Pang, B. Wang, C. Cai, and T. H. Lee. Guidance, navigation and control of an unmanned helicopter for automatic cargo transportation. To be submitted to the 2014 Chinese Control Conference.

[82] H. Wang, D. Wang, X. Niu, and H. Duan. Modeling and hovering control of a novel unmanned coaxial rotor/ducted-fan helicopter. In *Porceedings of IEEE International Conference on Automation and Logistics*, 2007.

[83] S. Weiss, M. Achtelik, S. Lynen, M. Chli, and R. Siegwart. Real-time onboard visual-inertial state estimation and self-calibration of MAVs in unknown environments. In *IEEE International Conference on Robotics and Automation*, pages 957–964, 2012.

[84] T. S. Wikman and W. S. Newman. A fast, on-line collision avoidance method for a kinematically redundant manipulator based on reflex control. In *IEEE Internatinoal Conference on Robotics and Automation*, 1992.

[85] A. D. Wu and E. N. Johnson. Methods for localization and mapping using vision and inertial sensors. In *Proceedings of the AIAA Guidance, Navigatoin and Control Conference and Exhibit*, 2008.

[86] S. Wu, Z. Zheng, and K. Cai. Indoor autonomous hovering control for a small unmanned coaxial helicopter. In *Proceedings of the 8th IEEE International Conference on Control and Automation*, 2010.

[87] H. Yu, R. Beard, and J. Byrne. Vision-based navigation frame mapping and planning for collision avoidance for miniature air vehicles. *Control Engineering Practice*, 18:824–836, 2010.

[88] 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 vision system for a UAV transporting cargoes between moving platforms. Submitted to the 2014 Chinese Control Conference.

[89] Y. Zhuang, K. Wang, W. Wang, and H. Hu. A hybrid sensing approach to mobile robot localization in complex indoor environments. *International Journal of Robotics and Automation*, 27(2):198–205, 2012.

List of Publications

Journal Articles:

1. F. Wang, P. Liu, S. Zhao, B. M. Chen, T. H. Lee, S. K. Phang, S. Lai, T. Pang, et al., Development of an unmanned rotorcraft system for the International UAV Innovation Grand Prix. To be submitted for journal publication.
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 an unmanned helicopter transporting cargoes between moving platforms, Submitted for journal publication.
3. F. Wang, J. Q. Cui, B. M. Chen and T. H. Lee, A comprehensive UAV indoor navigation system based on vision optical flow and laser FastSLAM, *Acta Automatica Sinica*, Vol. 39, No. 11, pp. 1889-1900, November 2013.
4. 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.
5. F. Wang, S. K. Phang, J. J. Ong, B. M. Chen and T. H. Lee, Design and construction methodology of an indoor UAV system with embedded vision, *Control and Intelligent Systems*, Vol. 40, No. 1, pp. 22-32, January 2012.

Book Chapters:

1. F. Wang, J. Cui, B. M. Chen and T. H. Lee, Flight dynamics modeling of coaxial rotorcraft UAVs, *Handbook of Unmanned Aerial Vehicles* (Edited by K. P. Valavanis and G. J. Vachtsevanos), Springer (in press).

Conference Papers:

1. F. Wang, P. Liu, S. Zhao, B. M. Chen and T. H. Lee, Guidance, navigation and control of an Unmanned Helicopter for Automatic Cargo Transportation. To be submitted to 2014 Chinese Control Conference.
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 vision system for a UAV transporting cargoes between moving platforms, Submitted to 2014 Chinese Control Conference.
3. F. Wang, K. Wang, S. Lai, B. M. Chen and T. H. Lee, An efficient navigation solution for UAV in confined but partially known indoor environments. To be submitted to 2014 IEEE International Conference on Control & Automation.
4. J. Q. Cui, F. Wang, X. Dong, K. 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.
5. F. Lin, K. Z. Y. Ang, F. Wang, B. M. Chen, T. H. Lee, et al., Development of an unconventional unmanned coaxial rotorcraft: GremLion, Presented at the 15th International Conference on Human-Computer Interaction, Las Vegas, USA, July 2013.
6. B. X. Hon, H. Tian, F. Wang, B. M. Chen and T. H. Lee, A customized Fast-SLAM algorithm using scanning laser range finder in structured indoor environments, Proceedings of the 10th IEEE International Conference on Control and Automation, Hangzhou, China, pp. 640-645, June 2013.
7. F. Wang, J. Cui, S. K. Phang, B. M. Chen and T. H. Lee, A mono-camera and scanning laser range finder based UAV indoor navigation system, Proceedings of the 2013 International conference on Unmanned Aircraft Systems, Atlanta, GA, USA, May 2013.
8. J. Cui, F. Wang, Z. Qian, B. M. Chen and T. H. Lee, Construction and modeling of a variable collective pitch coaxial UAV, Proceedings of the 9th International Conference on Informatics in Control, Automation and Robotics, Rome, Italy, pp. 286-291, July 2012.

9. D. Luo, F. Wang, B. Wang and B. M. Chen, Implementation of obstacle avoidance technique for indoor coaxial rotorcraft with scanning laser range finder, Proceedings of the 31st Chinese Control Conference, Hefei, China, pp. 5135-5140, July 2012.

10. B. Wang, F. Wang, B. M. Chen and T. H. Lee, Robust flight control design for an indoor miniature coaxial helicopter, Proceedings of the 10th World Congress on Intelligent Control and Automation, Beijing, China, pp. 2918-2924, July 2012.

11. F. Wang, S. K. Phang, J. Cui, G. Cai, B. M. Chen and T. H. Lee, Nonlinear modeling of a miniature fixed-pitch coaxial UAV, Proceedings of the 2012 American Control Conference, Montreal, Canada, pp. 3863-3870, June 2012.

12. F. Wang, S. K. Phang, J. Cui, B. M. Chen and T. H. Lee, Search and rescue: a UAV aiding approach, Proceedings of the 23rd Canadian Congress on Applied Mechanics, Vancouver, Canada, pp. 183-186, June 2011.

13. F. Wang, T. Wang, B. M. Chen and T. H. Lee, An indoor unmanned coaxial rotorcraft system with vision positioning, Proceedings of the 8th IEEE International Conference on Control and Automation, Xiamen, China, pp. 291-296, June 2010.