Vision-based Sensory System in the Indoor Applications of Unmanned Aerial Vehicle

Submitted by
Wang Yuxiang

Department of Electrical & Computer Engineering

In partial fulfillment of the
requirements for the Degree of
Bachelor of Engineering
National University of Singapore

Abstract

This thesis is devoted to the design and implementation of a vision-based onboard sensory system of an indoor unmanned aerial vehicle in its autonomous flight along a colored track. Specifically, attitude, position and velocity of the indoor helicopter with respect to the track can be efficiently estimated using vision and inertial measurement at more than 20 Hz on an embedded computer.

The core of the project is the vision-based algorithm that recovers the pose of helicopter based on vanishing geometry. In particular, vanishing line and a vanishing point of ground plane are calculated from a set of equally-spaced parallel lines. By using such geometries, an elegant closed-form expression is derived to compute the rotation matrix of the camera. This rotation matrix is then used in a constrained estimation of the camera location. Moreover, a fast line detection algorithm is proposed where one dimensional edge detections are performed to a sampled subset of image pixels. In the latest development of the project, readings of an inertial measurement unit is integrated so as to refine the vision algorithm. This involves the fusing of pose readings and the estimation of velocity from position and acceleration via Discrete Kalman Filtering.

The system has been successfully applied to a mini-UAV, codenamed "Mer-Lion" in the Singapore Amazing Flying Machines Competition 2011. The UAV was able to autonomously follow the track and complete a series of tasks on its own, thus is awarded the overall championship.

Acknowledgement

Author of the paper would like to express his sincere gratitude to the project supervisor Prof. Ben M. Chen, direct supervisor Dr. Lin Feng and fellow researchers Swee King, Wang Fei, Shiyi, Sing Jie, Sharon, Yi Ling for their invaluable support and encouragement.

Contents

List of Figures

List of Tables

List of Symbols

$\boldsymbol{X}$	Image point in homogeneous coordinate
X, Y	Image point coordinate
$\boldsymbol{x}$	World point in homogeneous coordinate
x, y, z	World point coordinate
W, w	Augmented dimension coordinate
$\mathbb{P}^i$	Perspective space of dimension i
$\boldsymbol{X}_\infty$	Vanishing point on image
$\boldsymbol{L}_\infty$	Vanishing line on image
$\boldsymbol{l}_\infty$	Vanishing line in world frame
$\tilde{\boldsymbol{C}}$	Camera center in world frame
$\boldsymbol{K}$	Intrinsic matrix/calibration matrix
f_x, f_y	Focal length in pixel unit
c_x, x_y	Principal displacement in pixel unit
s	Skew factor
$\boldsymbol{M}$	Extrinsic matrix $\boldsymbol{M} = [\boldsymbol{R}\vert\boldsymbol{t}]$
$\boldsymbol{R}$	3D Rotation matrix
$\boldsymbol{r_1}, \boldsymbol{r_2}, \boldsymbol{r_3}$	Columns of rotation matrix $\boldsymbol{R}$
r_{ij}	Entry at location i, j of rotation matrix

$\boldsymbol{t}$	Translation vector
$\boldsymbol{P}$	Camera projection matrix
$\boldsymbol{H}$	Homography
θ	Pitch angle
ϕ	Roll angle
ψ	Yaw angle
a_x, a_y, a_z	Linear acceleration in body frame
$\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D}$	System matrices in linear system

List of Abbreviations

ACELFS	Automated Cross Embedded Linux From Scratch
ARM	Advanced RISC Machine
AVATAR	Autonomous Vehicle Aerial Tracking and Reconnaissance
COM	Computer-On-Module
DLT	Direct Linear Transform
d.o.f	Degree of Freedom
ETH	Eidgenšsische Technische Hochschule
FIFO	First In First Out
FPS	Frame Per Second
GCS	Ground Control Station
GNU	GNU is Not Unix (recursively)
GPS	Global Positioning System
HSV	Hue Saturation Value
IDE	Interactive Development Environment
IMU	Inertial Measurement Unit
ISO	Exposure index specified by International Organization for Standardization
MAST	Micro Autonomous System Technology
MEMS	Microelectromechanical systems

NED	North-East-Down
OMAP	Open Multimedia Application Platform
OS	Operating System
QoS	Quality of Service
R/C	Radio Control
RANSAC	RANdom SAmple Consensus
RGB	Red Green Blue
SAMFC	Singapore Amazing Flying Machine Competition
SLAM	Simultaneous Localization and Mapping
SVD	Singular Value Decomposition
TCP/IP	Transfer Control Protocol/Internet Protocol
UART	Universal asynchronous receiver/transmitter
UAV	Unmanned Aerial Vehicle
UDP	User Datagram Protocol
USB	Universal Serial Bus
USC	University of Southern California
V4L2	Video For Linux Two
WiFi	Trademark for IEEE 802.11x standards

Chapter 1

Introduction

1.1 GPS free: flying in indoor environment

The unavailability of satellite signal in indoor context has forced autonomous robots, such as unmanned aerial vehicles (UAVs), to seek other forms of localization methods. One possible solution is to use combinations of ultrasonic or infrared range finders. However, these types of sensors are not suitable for complex environment. Slightly better are the class of more sophisticated sensors such as laser scanner. Yet, it is not feasible to install these sensors onto the airborne platform considering their weight and high power consumption. Due to the low cost and flexibly programmable nature of cameras, vision-based localization has become a popular choice for researchers and engineers.

1.2 Challenges of an embedded vision system

There are pros and cons in adopting a vision system. While it is light-weighted, low cost, high resolution and provides rich information, vision algorithms that make use of such information are usually too bulky to be implemented on embedded environment [2].

To tackle the problem, one common configuration is to transmit the image frame back to a ground control station (GCS) for processing and then send processed results to UAV for flight control and navigation, as in [3, 4]. The performance of this configuration depends on the quality of the data-link between UAV and GCS, hence is subject to electronic interference, traffic jam and physical barriers, such as a wall. More critically, the delay incurred in this method is likely to cause serious problems in UAV control.

Therefore, the major challenge of developing an vision system for UAV is to perform everything on-board with the limited power of computation.

Besides the lack of computational power, the uniqueness of indoor UAV flight poses further constraints to the vision system. They are:

(1) Lack of maneuvering space

(2) Blurry images due to vibration

The lack of space to maneuver limits the size of the UAV to the level of mini-UAV and micro-UAV. Such constraint severely limits the number of sensors, computational instruments and batteries that can be carried onboard. The narrow space also imposes stringent design parameters for controller and the precision of sensor measurement, e.g. vision measurement.

During indoor flight, quality of the images captured will be influenced by the severe vibration of the brushless motors and rotor blades. This can be detrimental to a number of popular image processing algorithms.

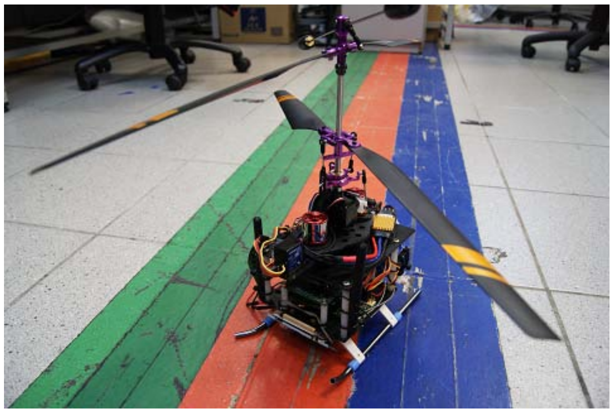


Figure 1.1: "MerLion" sitting on the colored track

1.3 An efficient vision sensory system

While it is difficult to develop embedded vision system, it is by far the most reachable way to attain autonomous flight. Thanks to the works in [3], the platform design and image capturing utility is available from the very beginning of this project.

In this thesis, an efficient vision-based sensory system is proposed for UAV's indoor autonomous flight along a colored track (see Figure 1.1). This system features full onboard processing, recovers the UAV's attitude and location accurately and is robust against various fly conditions, such as vibration. Furthermore, it attains a real-time performance of more than 20 Hz. Major innovations of this system over the predecessor in [3] is summarized below:

(1) Enlarged field of view

(2) Linear-time robust line detection method

(3) Exact pose and position measurement

(4) Fusing the Inertial Measurement Unit (IMU) readings

(5) Velocity estimation with Kalman Filter

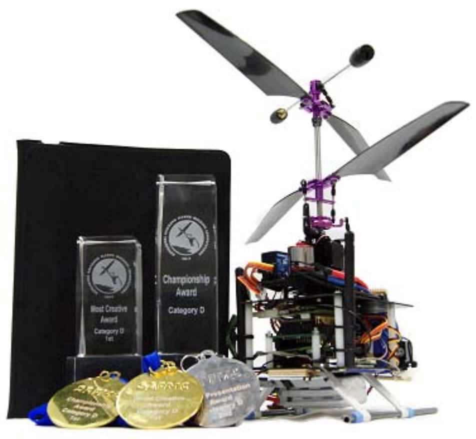


Figure 1.2: "MerLion" winning SAFMC Grand Awards

Figure 1.3: SAFMC 2011

1.4 Singapore Amazing Flying Machine Competition

To test out the system, this vision-based sensory system is installed on our mini-UAV and entered for the Singapore Amazing Flying Machine Competition (SAFMC). This coaxial rotorcraft named "Merlion" won the overall championship and most creative award after completing a set of challenges automatically. The tasks include: A gate, a beam, a hoop, a symbol table to be detected via video link and a target for releasing payload.

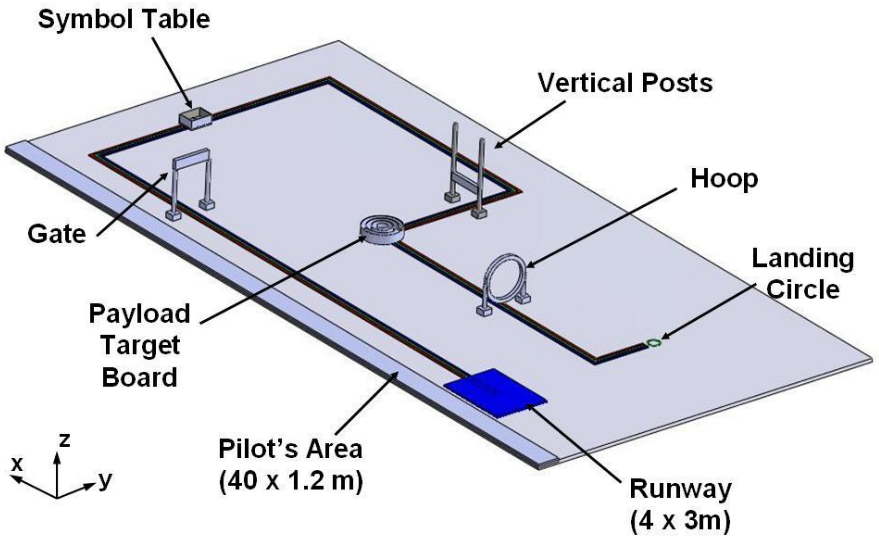


Figure 1.4: SAFMC Challenges: Category D

1.5 Related works

PixHawk Project The PixHawk project from ETH has been very successful in embedded computer vision on UAV. Similar to us, they target at indoor flight and onboard processing. They have provided an entire documentation of the cross-compilation toolchain and drivers of onboard cameras over the OpenEmbedded platform. As is pointed out in [3], the PixHawk team is using exactly the same onboard processor as MerLion, we can hence expect similar processing capability and performance in both system. It is further noted that PixHawk team also implemented a track following algorithm for autonomous flight. Nonetheless, their track is based on some existing barcode patterns where software that analyzes such patterns efficiently is readily available online. As PixHawk did not disclose any information regarding their vision measurements, The performance comparison of their system to ours remains unknown.

MAST-Micro Autonomous System Technology Prof. Vijay Kumar and his team have developed very impressive indoor maneuvers of UAV [5,6]. Yet, their applicability is limited to a special room where the so-called ”indoor GPS system” is available via the scene reconstruction using multiple cameras. Instead of the ego-centric view where camera is mounted onboard the robot, MAST's system requires a god's eye view of the world.

AVATAR- Autonomous Vehicle Aerial Tracking And Reconnaissance The USC's Autonomous Flying Vehicle project has been very successful in developing UAV's autonomous flight in urban environment. The 3rd Generation of their helicopter AVATAR is equipped with a frontal stereo camera and two fish-eye cameras on the side for optical flow [7]. While they have shown great capability of combining stereo and optical flow in its vision measurement system, it is not feasible to implement the same configuration on MerLion. The size and payload of the AVATAR is sufficient to carry very powerful computers onboard and interface multiple cameras.

1.6 Organization of the document

The content of this thesis is divided into four parts. In Chapter 1, a brief overview of the whole system of Merlion will be given with focus on software and hardware related to the onboard vision system. It follows by Chapter 2 that describes the vision-based pose estimation and localization where only vision information is used. Chapter 3 documents the several attempts in fusing IMU and vision measurement. The results turn out to be ideal in several aspects. Finally, Chapter 4 presents the track following algorithm that is used in the competition.

Chapter 2

Embedded Platform Design

2.1 Overview of the MerLion

MerLion is a mini scale coaxial rotorcraft designed and built by our group of five FYP students. It weighs 768 g and is capable of indoor autonomous hovering and track following operations. The hardware of MerLion is summarized in the following diagram. For details of the chip selections, and performance consideration, please refer to [8].

The information flow of the system is summarized in Figure 2.2. Essentially, image captured by the onboard camera is processed by the vision processing system and attitude and position of the UAV is extracted from the image. As we can see in the block diagram, vision processing system can also take in IMU measurement to refine the position estimation. This will be discussed in Chapter 4.

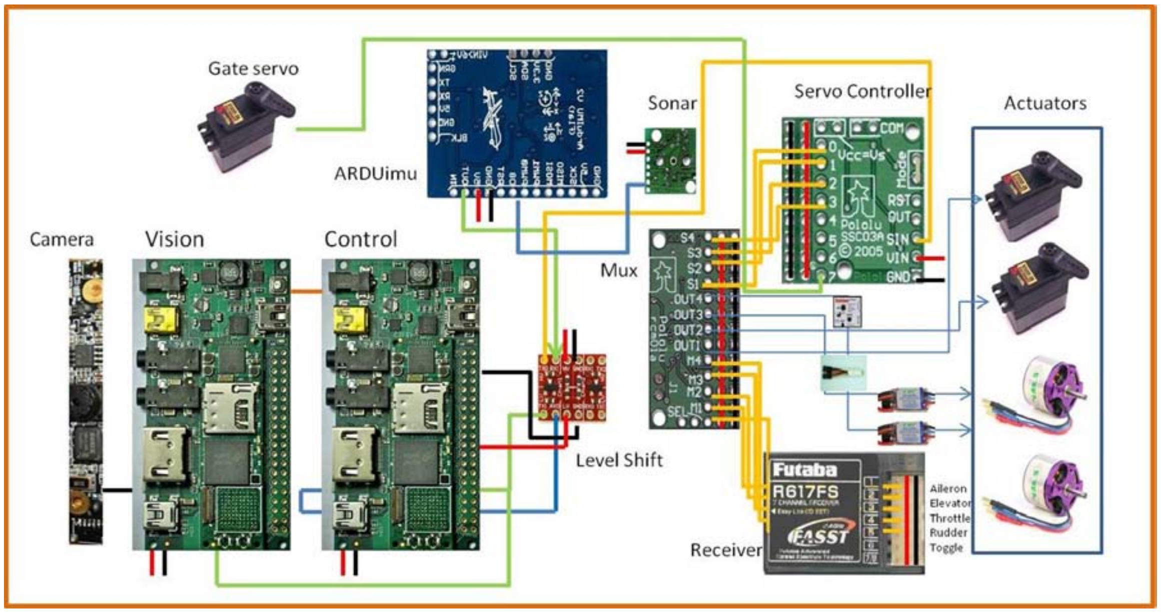


Figure 2.1: Hardware summary diagram

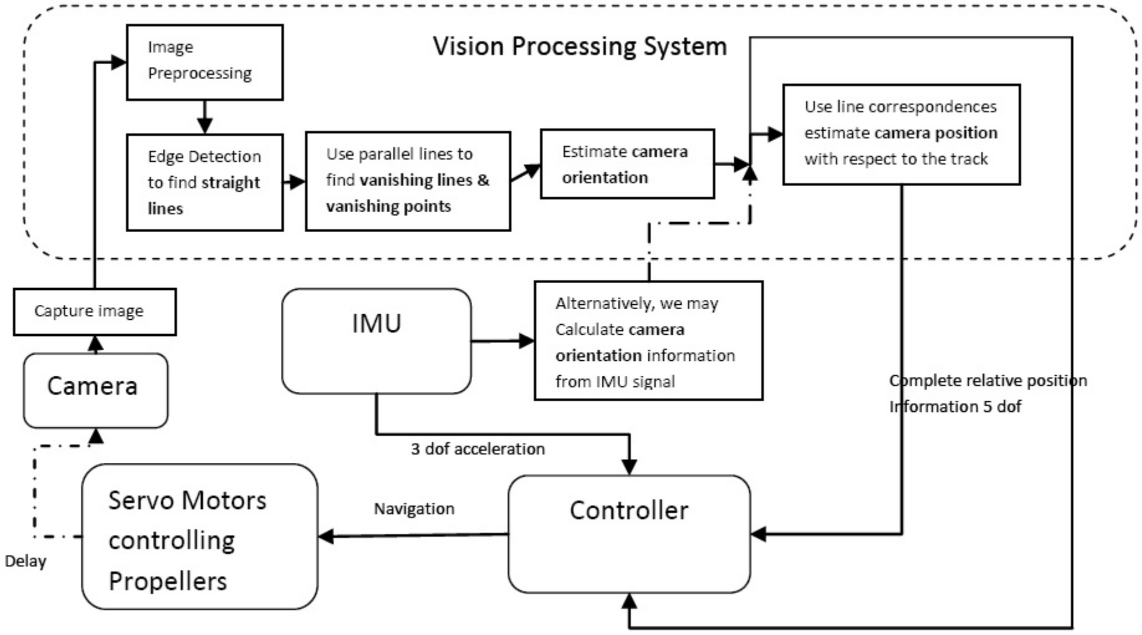

Vision Processing System
Image Preprocessing
Edge Detection to find straight lines
Use parallel lines to find vanishing lines & vanishing points
Estimate camera orientation
Use line correspondences estimate camera position with respect to the track
Capture image
Camera
IMU
Alternatively, we may Calculate camera orientation information from IMU signal
Complete relative position Information 5 dof
3 dof acceleration
Navigation
Servo Motors controlling Propellers
Controller
Delay

Figure 2.2: Block diagram of MerLion's information flow

2.2 Platform of the embedded vision system

Now let us take a closer look of the platform design for embedded vision system. In order to realize airborne onboard vision processing, the system must meet the following criteria.

(1) Fast processing

(2) Light weight

(3) Low power consumption

(4) Support available on the internet

Henceforth, we need a powerful yet small microprocessor, a reasonably well-performed camera, and a software infrastructure that facilitates the development. After careful selection, we have decided on the following software and hardware combinations.

2.2.1 Hardware platform

Microprocessor: Gumstix Overo Fire COM

Relevant specifications of the embedded computer are summarized in Table 2.1.

The small size, high performance, low cost and integrated WiFi connectivity of this computer-on-module (COM) has made it an ideal choice for our application.

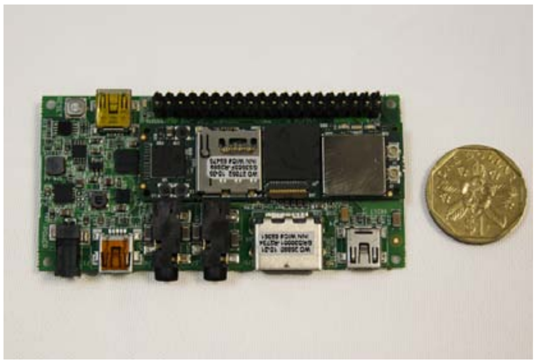
Camera: HP DV3000 Webcam

This camera is chosen in an ad-hoc way. The system is not very particular about webcam specifications, although a large field of view is recommended. In fact, any USB webcams that are light-weighted and support 320×240 frame capturing at 30 FPS will perform the same in our system.

Processor	Texas Instruments OMAP 3530 Applications Processor: - ARM Cortex-A8 CPU - C64x+ digital signal processor (DSP) core
Clock(MHz)	720 MHz
Memory	256MB RAM 256MB Flash
Features	OMAP3530 Application Processor 802.11b/g wireless communications Bluetooth communications microSD card slot
Size	17mm × 58mm × 4.2mm (0.67 in. × 2.28 in. × 0.16 in.)
Weight	5.6g

Table 2.1: Gumstix Overo Fire Specifications

Through experiment, we found that the vertical angle of view of this webcam is about 45 degrees and the horizontal angle of view is about 60 degrees. This is sufficient for our application.


(a) Gumstix Overo Fire

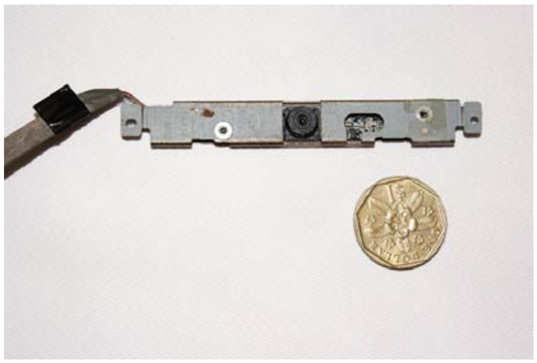

(b) HP Webcam on DV3000 Laptop

Figure 2.3: Illustration of hardware

2.2.2 Software platform

Operating system and cross compilation toolchain

The operating system (OS) is the core of the software platform as it includes a complete set of hardware drivers, resource manager, file system and toolchain for software development etc. In this project, GNU/Linux is the choice. We choose this OS for three reasons. It is free, well-supported online and the drivers to all hardware of interest are readily available in the latest kernel build.

This project adopts a customized version of Linux OS which is compiled and built completely from source packages. The procedure is automated by the Auto Cross Embedded Linux From Scratch (ACELFS). Although through time, certain links are broken and some packages are updated to new version, the build script of ACELFS is still of great convenience once a few tweaking here and there are done correctly. Details of ACELFS can be obtained in its developer's thesis [cite Jun Jie].

It is further noted that besides a Linux OS, ACELFS also constructed a cross-compilation tool chain for Overo Fire COM's ARM processor. With this set of cross-compilation tools, prefixed "arm-unknown-linux-gnueabi-", we can develop software on common desktop or laptop with i386 architecture and build it for running on ARM architecture. This approach is faster, less invasive and much more developer-friendly. In fact, we have set up this cross-compilation tool chain in Eclipse IDE, which greatly simplified the management of large software project. An illustration of the cross compilation and the cross development cycle are given in the following two diagrams.

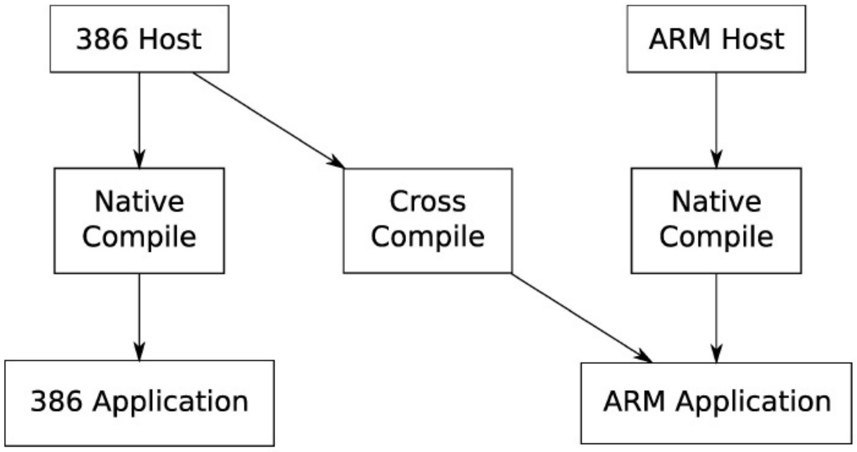


Figure 2.4: Illustration of different types of compilation

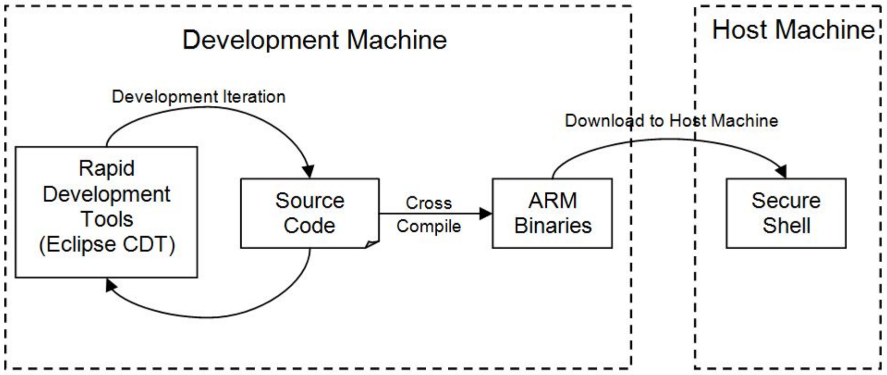


Figure 2.5: Illustration of cross development of software

Open source packages used

OpenCV [9] The best open source computer vision library in C and C++. Implementations of image data structure, matrix manipulation as well as singular value decomposition are used intensively in this project.

Levmar [10] This is an efficient C++ implementation of Levenberg Marquadt Optimization. It is used for non-linear refinement in this project.

Matlab camera calibration toolbox [11] Excellent toolbox for camera calibration using matlab. It has an intuitive GUI and many options for advanced level configuration. It is used for off-line calibration of camera's intrinsic parameters.

V4L2 API While Video for Linux Two library is provided in Linux kernel, making use of the driver to capture frame in user-defined program requires the source code of this package. According to the discussion in [3], opencv's grabbing function cvQueryFrame is much slower and less stable compared to v4l2.

2.3 Video link and GCS development on iPad

To monitor the flight state of UAV and to assist the testing of vision algorithms, a ground control station (GCS) is developed on a favorite platform: iPad(see Figure 2.6). Details of the iPad development and explanations of its functionality are available in [12]. The focus of this section will be on the smart interface between iPad and UAV, which consists of (1) Video link (2) Online tuning of HSV parameters (3) Dispatching/receiving high level command.

Figure 2.6: Apple's new tablet PC: iPad

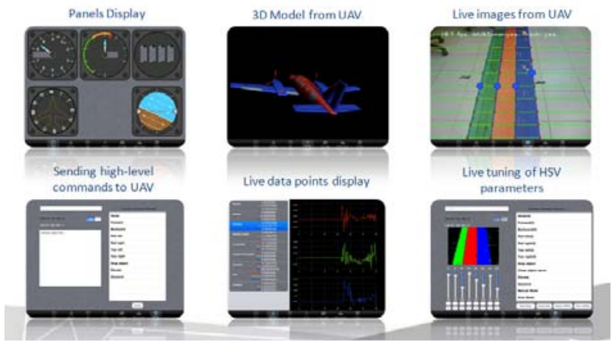


Figure 2.7: Illustration of iPad GCS app

2.3.1 Video Link

Unreliable data link

802.11g WiFi connection is the obvious choice considering its availability on Gumstix, signal range and the build-in robustness of this protocol. Yet, in real conditions, especially in a public place where dozens of WiFi access points and ad-hoc devices are filling the frequency, the QoS of this wireless link can deteriorate significantly. In order not to influence the performance of the vision processor, UDP is adopted over TCP/IP, because in UDP, no handshaking and error retransmission are enforced.

Error in transmission

In case of error in transmission or temporary broken data link, UAV side will not be affected at all. It will continue to transmit images as if there is no problem. This ensures the immediate recovery of video display at GCS side once the connection is restored.

Live video feed at 4-5 FPS

In our design, the vision processor will compress an image frame into JPEG and transmit it back to GCS every 5 processed frames, which renders a live video feed of about 4 FPS on iPad display. This frequency is chosen such that it does not lag the vision measurement, yet appears smooth on the live video feed. Experiments have shown that the bottleneck of this video link is not the capacity of the wireless link but rather the image compression.

2.3.2 Online tuning

The second important augmentation of the iPad GCS to UAV is the online tuning mechanism of image processing parameters. In the tuning mode, parameters can be tuned online by simply scrolling the bar up and down on the touch screen until the ideal outcome is displayed on the video feed. This ensures easy adjustment of thresholds in different lighting conditions. An illustration of this function and why it is useful will be discussed in the Chapter 3.

2.3.3 High level command

Lastly, iPad GCS is able to control the UAV in semi-autonomous mode, which means to control the UAV motion manually without affecting its inner stability.

High level instruction is send to UAV based on the accelerometer measurement of the iPad. In this way, pilot holding the iPad is just like holding the swashplate of the helicopter directly.

2.4 Onboard Communication of devices

Onboard of MerLion, communications between devices are required to convey sensor measurements to controllers. Specifically, both vision and control processors demands measurements of IMU and vision processor need to pass the vision guidance to control processor. Unlike the communication to GCS, onboard communication requires high level of reliability. Thus it must be carried through a wired connection and adopt certain error checking mechanism. To cater the need, a simple packet structure is designed (see Table 2.2).

header	size	Data payload	ChecksumA	ChecksumB

Table 2.2: Serial Packet Structure

UART serial communication is chosen because of its availability on all devices. On the vision processor for instance, interface to the serial device and the data buffer have been taken care of by Linux kernel. All we have to do is to read/write the Tx/Rx buffers. The serial port settings are summarized in Table 2.3.

However, there is still a challenge: the sample rates of devices are different! While ArduIMU and control processor runs at 50 Hz, vision processor can only reach an average of 23 Hz. If the asynchronous behavior is not reconciled nicely, either will the system suffer substantial delay and eventually cause buffer overflow, or will the waiting time for new packet lengthened the running time per iteration. Something must be done.

Settings	**Flags**	**Explanation**
Baud rate	B38400	38400 baud per second
Control options (c_cflag)	~PARENB ~CSTOPB ~CSIZE CS8 ~CNEW_RTSCTS	8N1: no parity, 1 stop bit 8 data bit, Disable hardware flow control
Local Options (c_lflag)	~ICANON ~ECHO ~ECHOE ~ISIG	Set RAW mode input Disable echo Disable special signals
Input Options (c_iflag)	IGNPAR IGNBRK IGNCR	Ignore parity Ignore BREAK Ignore carriage return
Output Options (c_oflag)	~OPOST	Set RAW mode output

Table 2.3: Setting Serial port for RAW mode datalink

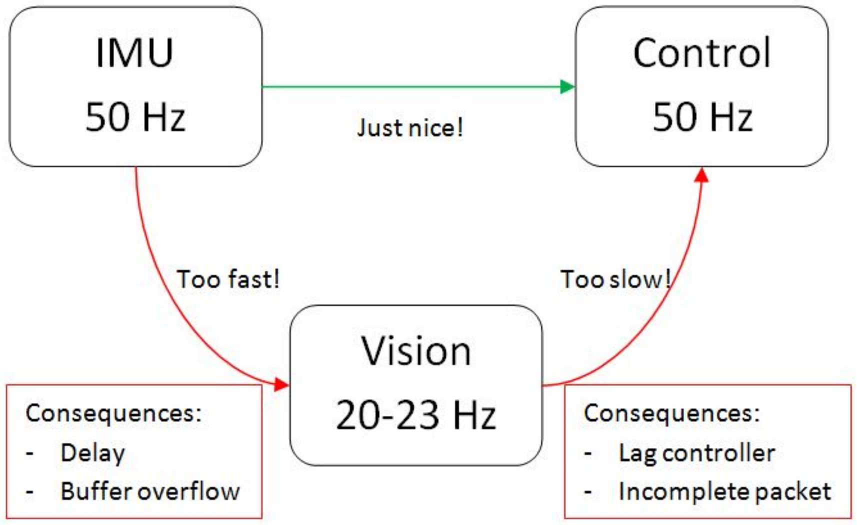


Figure 2.8: Illustration of inhomogeneous sample rate

To solve this problem, we came up with two separate algorithms to receive UART stream. Steps of the algorithms are summarized in the charts below. Using these algorithms, all communications are carried out in an up-to-date, polling manner.

IMU to Vision (High to Low)

(1) Read everything in the serial buffer to array buff[512].

(2) Search the array for packet header "DIYd", keep track of the latest two detections.

(3) Try reading the last packet.
If it is complete and passed the checksum, go to (5) else proceed to (4).

(4) Try reading the second last packet.
if it passed the checksum, proceed to (5), else return failure.

(5) Save the packet read and return success.

Vision to Control (Low to high)

Declare a global FIFO queue structure of bytes using array.

For every loop:

(1) Read everything in serial buffer and push into the queue.

(2) Search for packet header from the head of queue. if found, go to proceed to (3), else empty the queue and return failure.

(3) Try read the packet, if packet not complete, return failure;else if checksum fail go to (4), else go to (5).

(4) Pop the packet and go to (2).

(5) Save the packet, pop the packet from queue and return success.

Chapter 3

Vision-based pose estimation and localization

3.1 Background

3.1.1 problem description

As is described in the introduction, the purpose of the vision sensory system is to compensate the loss of GPS signal in indoor environment. To perform the same function as GPS, frames captured by a camera must be processed in a way such that the location and orientation of the camera can be extracted. This is often characterized as the camera calibration problem for extrinsic parameters, which is not easy in general unless certain assumptions are made about the environment, such as known corner locations or orthogonality of lines.

Self-localization in an unknown environment is generally termed vision-based Simultaneous Localization and Mapping (Vision SLAM) where a map of the environment and robot's location within the environment is computed simultaneously. This is usually carried out via stereo [13], multiple view or structure from motion,

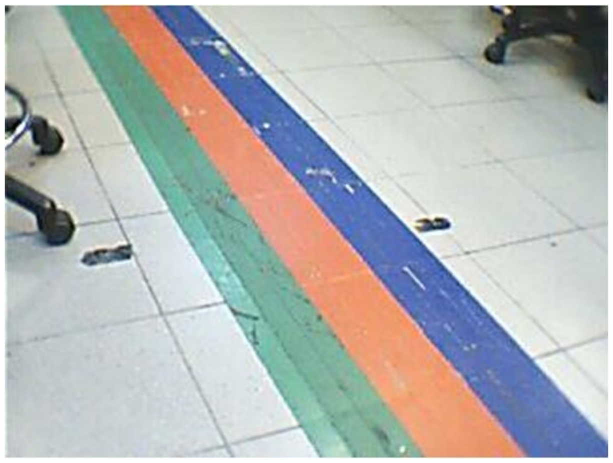


Figure 3.1: Illustration of the colored track

e.g. monoSLAM [14]. In theory, it is possible to reconstruct the 3D model of an environment up to projective ambiguity for uncalibrated camera and up to metric ambiguity for calibrated camera [15]. We will not be able to discuss this general case in this thesis.

In this problem, we are looking at a fixed pattern on the ground plane with its metrics given a priori. Given this RGB colored track on the floor (see Figure 3.1), onboard vision system should be able to give measurements about UAV's **heading**, **height** and **lateral displacement** with respect to the track. In fact, as will be shown later in this chapter, it is possible to recover five out of the six extrinsic parameters of the camera given the simple geometry as in Figure 3.1. The only missing value is the longitudinal position along the track which is not a well-defined value anyway given the reference structure.

3.1.2 previous work

Two former students have worked on this subject in 2009 and 2010. They came up with the following two designs:

Vision Positioning System of "PetiteLion" [4] This system adopts various image processing techniques such as HSV color segmentation, morphological filtering, Canny Edge Detection and Hough transform for line detection. While it is workable, the high computational cost forced the system to process image remotely at GCS.

Onboard Vision Navigation System of "KingLion" [1] This system features a highly efficient line extraction method by considering only the boundary of the image frame (illustrated in Figure 3.2). The corresponding threshold of 1D edge detection is selected adaptively via entropy minimization which yields a degree of robustness against change in lighting conditions. The drawbacks are obvious too. As is shown in Figure 3.3, defects or reflections at critical region(the image boundary) will cause the algorithm to fail.

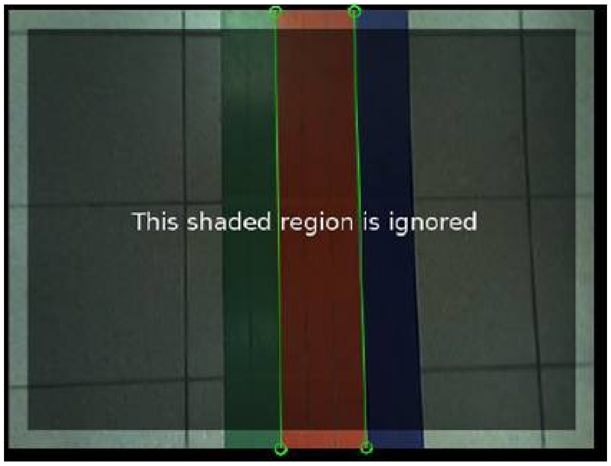


Figure 3.2: Illustration of line detection in [1]

Despite their difference in image processing algorithms, the two systems share the same limitations.

(1) Small field of view, easy to lose target.

(2) Use only middle band of the track, failed to exploit all information from the track.

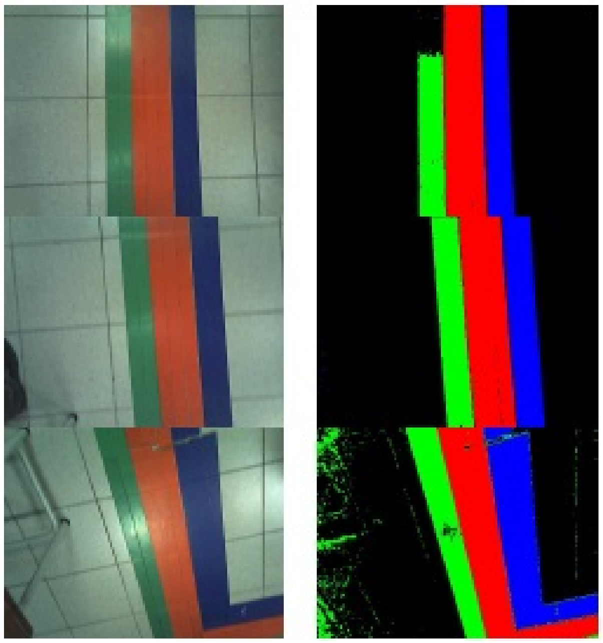


Figure 3.3: Illustration of possible failures in [1]

(3) Ambiguity between θ and y.

(4) Near hovering assumption: Limited applicability on coaxial rotorcraft.

(5) Measurements are qualitatively correct,but not theoretically plausible.

3.1.3 The new method and vanishing geometry

To overcome these limitations, we propose a simple and direct solution:

Look forward

By placing the camera somewhat towards the frontal direction, a larger portion of the track can be captured in image frame (illustrated in Figure 3.4). Since the track far away from camera is more stable in image, they are less likely to deviate outside the image frame. Also, we can make use of the perspectivity to compute the geometric features at the infinity such that the estimation of position and orientation can be done in a decoupled way.

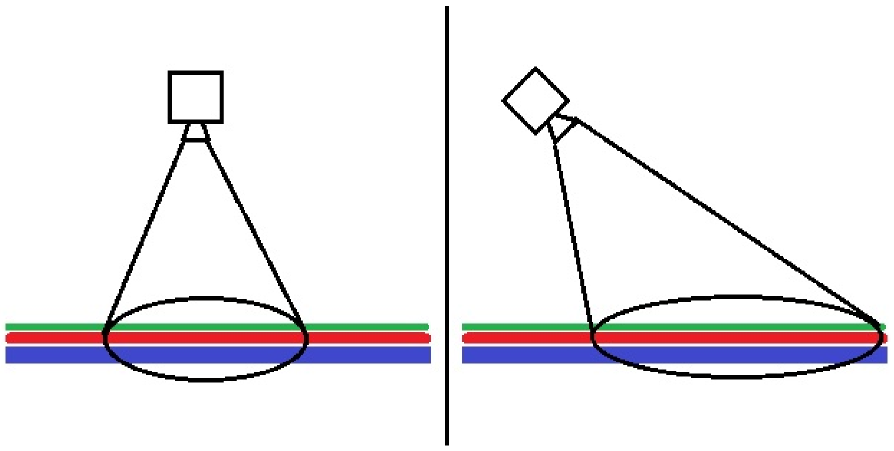


Figure 3.4: Looking forward broadens the robot's horizon

To do this, the geometry of vanishing point and vanishing line plays an important role.This is because because points at infinity are invariant to the Euclidean translation of view point, which is intuitive as we all have the experience on the train that the mountains far away move slower than the trees along the railway. Towards infinity, for instance the moon, almost remain stationary as we walk.

Not surprisingly, vanishing features can be used to extract the camera pose. Various researchers have worked on vanishing point and line. Wang and Tsai transformed vanishing line directly to roll angles in [16], but extra information was required to uniquely determine yaw and pitch. Vanishing points were also used for pose estimation in [17–19], but they required the points to be pointing at orthogonal directions. In addition, a pose estimation using single vanishing point normal to ground plane was proposed by Xu in [20]. Yet, it recovers only pitch and roll.

Although aforementioned works identified the importance of vanishing geometry, they failed to address the common situation where only one vanishing point is available in the image,e.g. road following.This is exactly the case in our situation where the UAV is to fly autonomously according to the colored track on

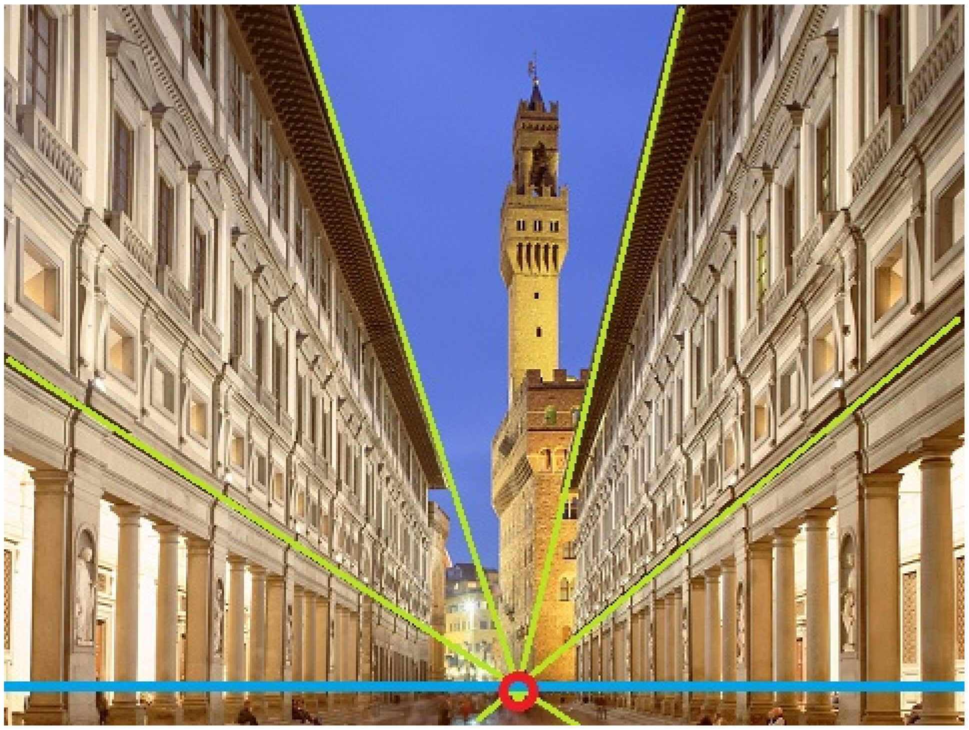


Figure 3.5: Perspectivity: vanishing geometry of Uffizi Museum, Florence

the floor (illustrated in Figure 3.1). Though in a sense, vision path following is considered as a solved problem, the existing approaches do not require explicit recovery of the camera pose [21–23]. The concept of vanishing point was only used to provide heading feedback [23]. Even for the visual servoing of the UAV, near hovering assumptions (zero pitch and roll angles) are being made, such as in [1, 4, 24]. The near hovering assumption might be reasonable for low-speed coaxial rotors with self-balancing hardware, but is not valid for quad-rotor and conventional helicopter.

In this chapter, we describe an efficient algorithm to estimate UAV pose as well as vertical and lateral displacement with respect to the track in Figure 3.1. An elegant closed-form expression of rotation matrix using vanishing point and line is derived, which reveals the underlying connections between pose and vanishing geometry. The rotation matrix is further applied in a constrained estimation of the camera's external matrix, in which the lateral and vertical displacement of the camera relative to the track are found by solving a system of over-determined linear equations. The measurements obtained will provide essential information for accurate control of the UAV and facilitate various autonomous tasks. Furthermore, the algorithm described in this chapter is accepted for poster presentation at the IAPR Conference of Machine Vision Application [25]. A large portion of this chapter will be identical to that paper, but with more explanations.

3.1.4 Interlude: pinhole camera model

As a small interlude, we will describe the pinhole camera model in computer vision. This model will be used intensively in the remaining part of the thesis. Pinhole camera model regard a camera as a simple perspective projection from 3D world to 2D plane. As in (3.1), the homogeneous representation of 3D world point $\boldsymbol{x} = [x\ y\ z\ 1]^{\mathrm{T}}$ is projected to the 2D point $\boldsymbol{X} = [X\ Y\ 1]^{\mathrm{T}}$.

$$\boldsymbol{X} \propto \boldsymbol{K}[\boldsymbol{R}|\boldsymbol{t}]\boldsymbol{x} = \boldsymbol{P}\boldsymbol{x} \tag{3.1}$$

where $\boldsymbol{P}$ is a 3×4 projection matrix.

$$\boldsymbol{K} = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

is the intrinsic matrix, and

$$\boldsymbol{M} = [\boldsymbol{R}|\boldsymbol{t}]$$

encapsulates the extrinsic parameters which include position and orientation of the camera.

$\boldsymbol{R}$ is the rotation matrix and $\boldsymbol{t}$ is a 3×1 translation vector.

Let the camera center in world frame be $\tilde{\boldsymbol{C}}, \boldsymbol{t} = -\boldsymbol{R}\tilde{\boldsymbol{C}}$.

Strictly speaking, (3.1) is only correct for camera obscura (without lens), but in reality, this model describes modern cameras just as well given thin lens assumption. Minor image abberations and lens distortion are usually ignored, though there exist methods to rectify these distortions. In this thesis, only radial distortion is considered.

3.2 Image Processing and Feature Extraction

3.2.1 Linear-time Line extraction

The high complexity of Canny edge detection and Hough Transform makes it unsuitable for onboard processing. In this particular case where a track of colored bands is available, much more efficient algorithms can be expected. Recall that the method described in [1]extract lines by performing 1D edge detection on the outer boundary of the image(see Section 3.1.2). It lowers the complexity significantly but its results are vulnerable to various scenarios. Inspired by the idea, we come up with a more reliable algorithm.

The steps are summarized below:

(1) Binary search for the top-most horizontal pixel sequence that contains [Green Red Blue] pattern;

(2) Take samples evenly in the lower half of image;

(3) Preprocess each sample, perform 1D edge detections and then search for the segment groups that contain [Green Red Blue] pattern;

(4) Rectify radial distortion;

(5) Fit line using least square methods;

(6) If the error is above a threshold, RANSAC is triggered. The line is then fit again with outliers excluded.

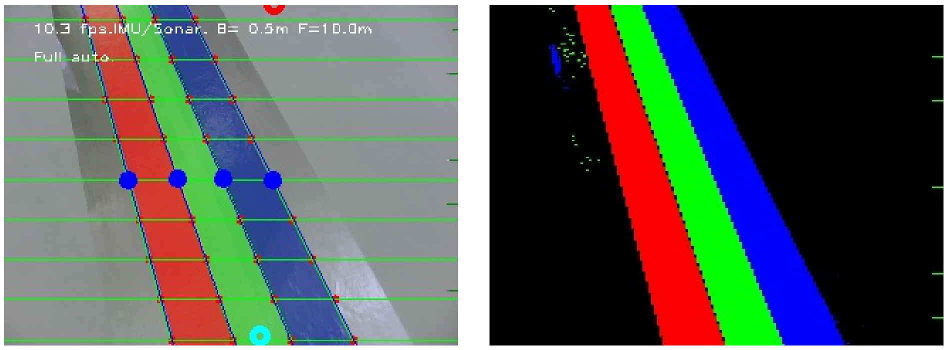
1D Edge Detection

1D Edge Detection is the fundamental building block of this line extraction method.

Almost exactly the same gradient-based edge detector as in [1], except that we added a pre-processing step, which eliminated the need of adaptive thresholding. While the authors of [1] claims its adaptive thresholding approach to be robust against change in lighting condition, it can only robustly detect the two lines in the middle. This is not acceptable in this application.

For the pre-processing step, an HSV domain color segmentation is developed. A set of properly tuned thresholds can segment the image nicely such that the edge detection becomes trivial. An illustration of the color segmentation is given in Figure3.6. Note that for illustration purposes, the algorithm is applied to the entire image instead of only the sampled strip of pixels.

Indeed, this HSV color segmentation is sensitive to lighting conditions. Thus it must be tuned for every environment the UAV flies in. As the lighting infrastructure is usually fixed for a particular indoor environment, this approach is good enough. For easy tuning of the HSV parameters, an App is developed on the iPad GCS. The App features real-time display of processed images while the user modifies a particular value by simply scrolling a bar on the touchscreen. With this App, tuning of HSV parameter can be done in seconds. Details of the iPad GCS development can be found in [12]. In fact, implementing the HSV color segmentation yields further significance. It helps the system to handle blurry images, in a



(a) original image (b) pre-processed image

Figure 3.6: Illustration of color segmentation.

sense that this color segmentation will automatically identify a good edge within the overlapping regions of two colors. As we'll show later, the simulation result confirms the system robustness against vibration.

RANSAC line fitting

RANSAC is a generic algorithm in fitting a particular model with data that contains large outliers. It is used here to eliminate outliers and attain a more better line fitting results in case of track and image defects. An illustration is available at Figure 3.7(b), in which the fourth and last two samples of the right most line are excluded from the computation of the line.

Implications in efficiency

There are several implications on efficiency within this new algorithm. First of all, sampling the image downsizes the image from 320×240 to $320 \times k$, where k is the number of samples. This reduces the two-dimensional problem to a multiple of one-dimensional ones. Secondly, the process of undistortion can now be performed only to the points of interest. Thirdly, RANSAC is less taxing with small k. It is even feasible to try out all combinations. Lastly, compared to Hough Transform,

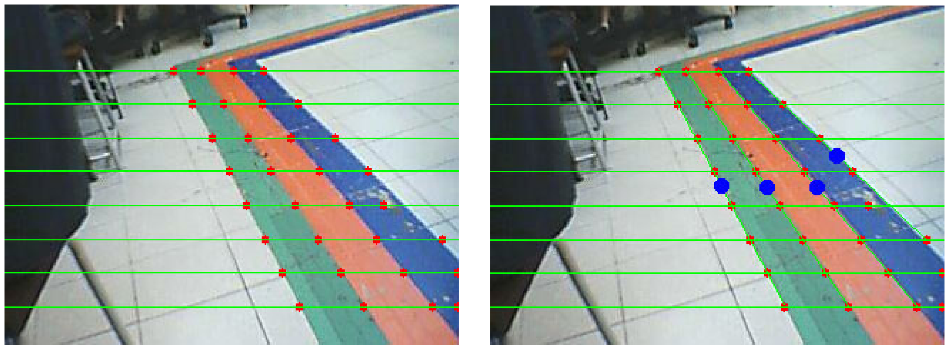

(a) Edge detection (b) RANSAC line fitting

Figure 3.7: Illustration of line segmentation.

this method saves the trouble in filtering and reordering of detected lines. In overall, this line detection algorithm retains a complexity of $O(n)$ when k is small and is thus an ideal method in our application.

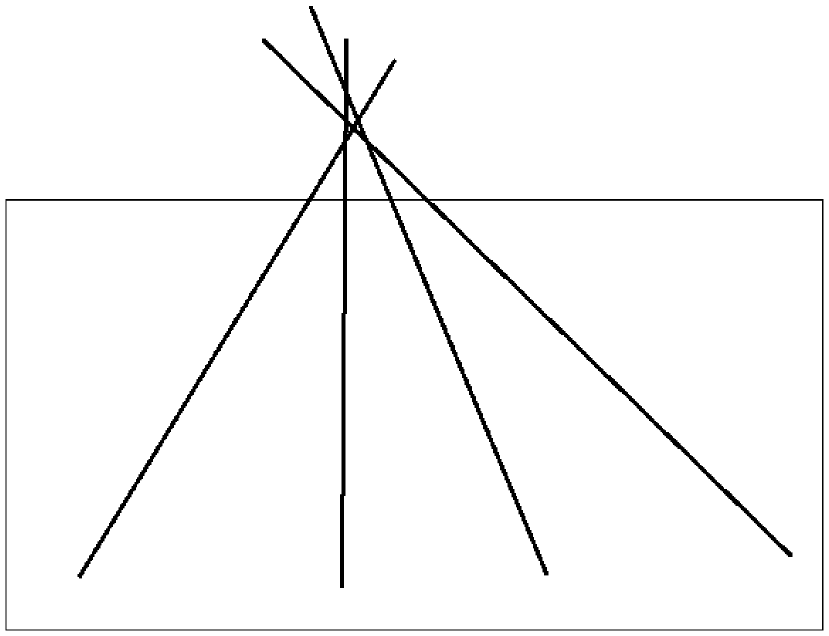
3.2.2 Finding the Vanishing Point and Vanishing Line

The next step is to calculate the vanishing point and vanishing line from the lines we obtained.

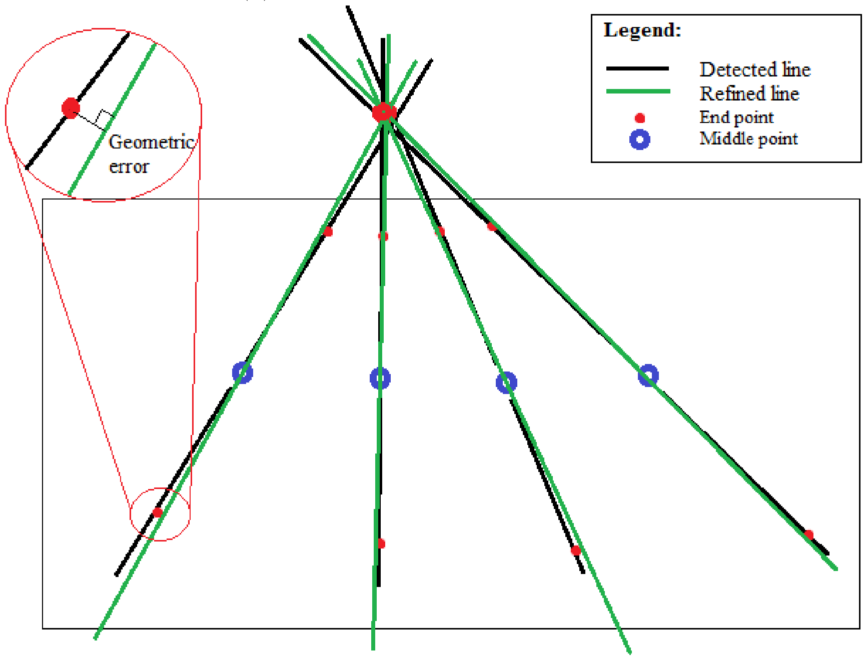
Vanishing Point

In theory, parallel lines should intersect at one single vanishing point on image. However, due to the noisy real image, this is often not the case (see Figure 3.8(a)).

To solve this numerical problem, we first apply normalized direct linear transformation (DLT) to formulate the problem into a system of linear equations. Then we apply SVD to find a linear least square solution that minimizes algebraic error. This algebraic error, however, does not attain the actual optimal. If we connect the intersection point and the middle point of each detected line segment and form a new line, a better cost function will be the sum of squared pixel distance between one end point of the line segment to the new line (illustrated in Figure 3.8(b)). To



(a)The over-determined problem.



(b)Minimizing geometric error.

Figure 3.8: An illustration of finding vanishing point.

minimize this geometric error which is non-linear, we apply Levenberg-Marquadt optimization and iteratively refine the solution. In this way, we obtained a better set of lines and an optimal intersection point simultaneously.This method is the gold standard method in computer vision described in [15].

Vanishing Line

According to [26], vanishing line can be expressed analytically with three coplanar equally-spaced parallel lines. With four such lines on ground planes in our case, the problem can be formulated into an over-determined system of linear equations. Nonetheless, such formulation is not explicitly disclosed in that paper. Hence, our formulation will be presented here as an example of applying Schaffalitzky's findings [26].

A group of parallel lines can be expressed in standard form:

$$\boldsymbol{L}_\lambda : ax + by + \lambda = 0$$

or its normal vector $\boldsymbol{L}_\lambda$ in matrix form:

$$\boldsymbol{L}_\lambda = \begin{bmatrix} 0 & a \\ 0 & b \\ 1 & 0 \end{bmatrix} \begin{pmatrix} \lambda \\ 1 \end{pmatrix} \tag{3.2}$$

which defines a projection from 1D projective space $\mathbb{P}^{\mathbf{1}}$ to 2D projective space $\mathbb{P}^{\mathbf{2}}$. By the definition of projective space, this relation is invariant to projective transformation (homography). Note that when λ equals to consecutive integers, the line group has equal spacing. Define $\boldsymbol{M}^{-\mathrm{T}}$ to be the line transformation matrix between world plane and image plane. By multiplying $\boldsymbol{M}^{-\mathrm{T}}$ on both side of (3.2), we obtain a projection matrix from points in $\mathbb{P}^{\mathbf{1}}$ to images of $\boldsymbol{L}_\lambda$.

$$\boldsymbol{A} = \boldsymbol{M}^{-\mathrm{T}} \begin{bmatrix} 0 & a \\ 0 & b \\ 1 & 0 \end{bmatrix}$$

Also note that the first column of the projection matrix in (3.2) is the vanishing line in $\mathbb{P}^{\mathbf{2}}$, then the first column of $\boldsymbol{A}$ is the vanishing line in the image. Let $\boldsymbol{A} = [\boldsymbol{a}_1^{\mathrm{T}},\ \boldsymbol{a}_2^{\mathrm{T}},\ \boldsymbol{a}_3^{\mathrm{T}}]^{\mathrm{T}}$, and the i-th image line is given by $\boldsymbol{L}_i = [x_i,\ y_i,\ 1]^{\mathrm{T}}$. Corresponding i-th object point $\boldsymbol{X}_i = [i,\ 1]^{\mathrm{T}}$. For each correspondence, the two linear equations are:

$$\begin{bmatrix} 0 & -\boldsymbol{X}_i^{\mathrm{T}} & y_i\boldsymbol{X}_i^{\mathrm{T}} \\ \boldsymbol{X}_i^{\mathrm{T}} & 0 & -x_i\boldsymbol{X}_i^{\mathrm{T}} \end{bmatrix} \begin{pmatrix} \boldsymbol{a}_1 \\ \boldsymbol{a}_2 \\ \boldsymbol{a}_3 \end{pmatrix} = \mathbf{0} \tag{3.3}$$

A total of four equally-spaced lines give 8 independent equations, which is more than enough to determine the matrix $\boldsymbol{A}$ and hence the vanishing line. An illustration of the detected vanishing line and vanishing point is shown in Fig. 3.9.

3.3 Pose Estimation and Constrained Localization

Without losing any generality, we choose the world coordinate frame with respect to the track. Specifically, the forward middle line of the track is x direction and the orthogonal direction to the right is y direction. By convention this is a right-hand coordinate, thus z-axis points vertically downwards (see Fig. 3.10). This world coordinate system is chosen in the same fashion as the North-East-Down(NED)

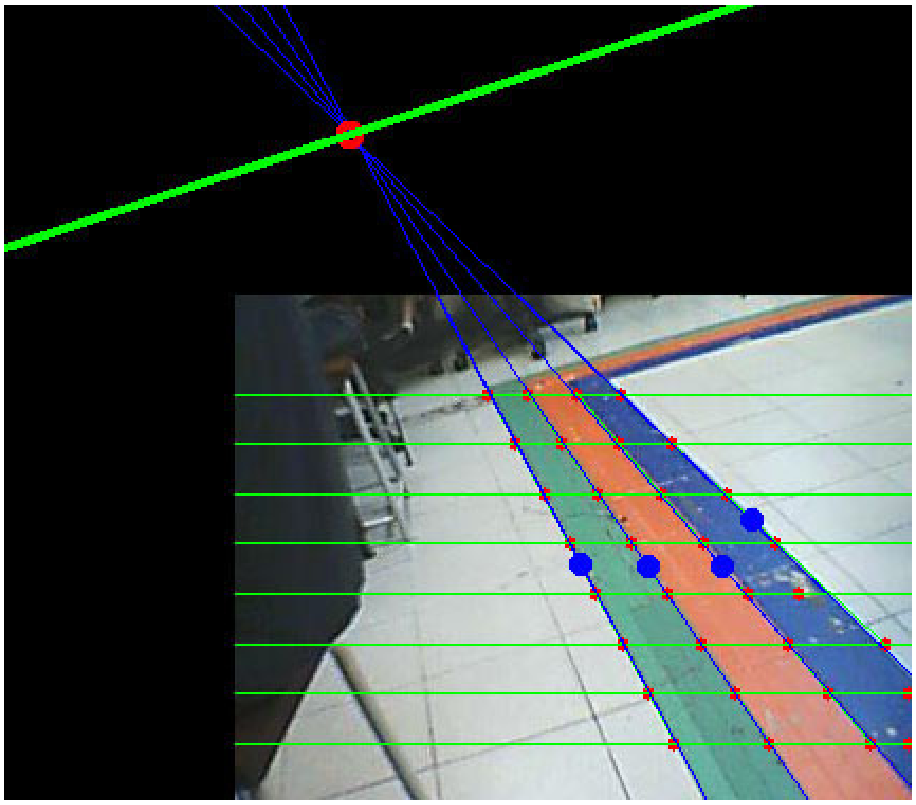


Figure 3.9: The detected vanishing point and line.

frame, which preserves the conventional sign and meaning of the yaw, pitch and roll angle.

3.3.1 Pose Estimation

Given the vanishing point on image $\boldsymbol{X}_\infty = [X,\ Y,\ 1]^\mathrm{T}$, we have

$$\boldsymbol{X}_\infty \propto \boldsymbol{K}[\boldsymbol{R}|\boldsymbol{t}]\begin{pmatrix}1\\0\\0\\0\end{pmatrix} = \boldsymbol{K}[\boldsymbol{r}_1\ \boldsymbol{r}_2\ \boldsymbol{r}_3\ \boldsymbol{t}]\begin{pmatrix}1\\0\\0\\0\end{pmatrix}$$

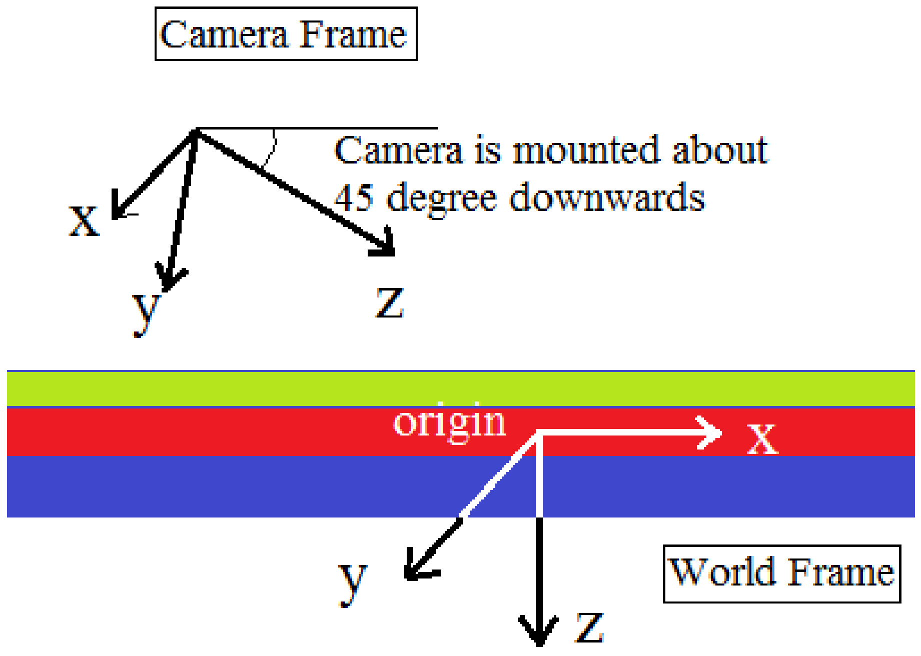


Figure 3.10: The illustration of the coordinate systems.

where $\boldsymbol{R} = [\boldsymbol{r}_1\ \boldsymbol{r}_2\ \boldsymbol{r}_3]$. Hence, we obtain

$$\boldsymbol{r}_1 = \frac{\boldsymbol{K}^{-1}\boldsymbol{X}_\infty}{||\boldsymbol{K}^{-1}\boldsymbol{X}_\infty||}$$

Let the ground plane vanishing line $\boldsymbol{L}_\infty = [p,\ q,\ 1]^{\mathrm{T}}$. Since all points on ground plane has a form $\boldsymbol{x} = [x,\ y,\ 0,\ w]^{\mathrm{T}}$, the projection matrix reduces to a homography.

$$\boldsymbol{H} = \boldsymbol{K}\begin{bmatrix} \boldsymbol{r}_1 & \boldsymbol{r}_2 & \boldsymbol{t} \end{bmatrix} \tag{3.4}$$

Thus the transformation equation is:

$$\boldsymbol{L}_\infty \propto \boldsymbol{H}^{-\mathrm{T}}\boldsymbol{l}_\infty = \boldsymbol{H}^{-\mathrm{T}}\begin{pmatrix}0\\0\\1\end{pmatrix}.$$

Inverting the homography to another side and expand the term, we obtain

$$\begin{pmatrix}\boldsymbol{r}_1^{\mathrm{T}}\\\boldsymbol{r}_2^{\mathrm{T}}\\\boldsymbol{t}^{\mathrm{T}}\end{pmatrix}\boldsymbol{K}^{\mathrm{T}}\boldsymbol{L}_\infty \propto \begin{pmatrix}0\\0\\1\end{pmatrix}$$

The first two equations imply that the vector $\boldsymbol{K}^{\mathrm{T}}\boldsymbol{L}_\infty$ is orthogonal to both $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$. This is exactly the characteristic of $\boldsymbol{r}_3$. Thus, up to direction ambiguity, we have derived:

$$\boldsymbol{r}_3 = \pm\frac{\boldsymbol{K}^{\mathrm{T}}\boldsymbol{L}_\infty}{||\boldsymbol{K}^{\mathrm{T}}\boldsymbol{L}_\infty||}$$

The other vector $\boldsymbol{r}_2$ is simply $\boldsymbol{r}_3 \times \boldsymbol{r}_1$ by right hand rule. Taking into consideration the physical meaning of $\boldsymbol{r}_3$: the unit vector in z direction of the world frame represented in camera frame, we can easily resolve the ambiguity by looking at camera's principal ray direction.

To sum up the derivation, rotation matrix can be expressed elegantly with one vanishing point and vanishing line:

$$\boldsymbol{R} = \left[\frac{\boldsymbol{K}^{-1}\boldsymbol{X}_\infty}{||\boldsymbol{K}^{-1}\boldsymbol{X}_\infty||} \quad \pm\frac{(\boldsymbol{K}^{\mathrm{T}}\boldsymbol{L}_\infty)\times(\boldsymbol{K}^{-1}\boldsymbol{X}_\infty)}{||\boldsymbol{K}^{\mathrm{T}}\boldsymbol{L}_\infty||\boldsymbol{K}^{-1}\boldsymbol{X}_\infty||} \quad \pm\frac{\boldsymbol{K}^{\mathrm{T}}\boldsymbol{L}_\infty}{||\boldsymbol{K}^{\mathrm{T}}\boldsymbol{L}_\infty||}\right] \tag{3.5}$$

3.3.2 Constrained Localization

We should notice that no information is given along the track's longitudinal direction, thus it is infeasible to recover x coordinate of camera center from the images. As a result, a shifting world coordinate system is adopted such that x is always zero.

Based on this assumption and the rotation matrix in Part A, the task of self-localization reduces to a constrained estimation of only two values, i.e., the lateral displacement and height. Given metric information of the track, the solution can be obtained using correspondences. Let the camera center $\tilde{\boldsymbol{C}} = [0,\ y,\ z]^{\mathrm{T}}$, we have

$$\boldsymbol{t} = -\boldsymbol{R}\tilde{C} = -y\boldsymbol{r}_2 - z\boldsymbol{r}_3 \tag{3.6}$$

Also let $\boldsymbol{l}_i$ be Line i in on ground plane and let $\boldsymbol{L}_i$ be the image of $\boldsymbol{l}_i$ in image plane. Since the lines are all parallel to x direction, we have a reduced form of $\boldsymbol{l}_i = [0,\ 1,\ w_i]^{\mathrm{T}}$. Then the homography between ground plane and image plane relating the line correspondences can be written as:

$$\boldsymbol{l}_i \propto \begin{pmatrix} \boldsymbol{r}_1^{\mathrm{T}} \\ \boldsymbol{r}_2^{\mathrm{T}} \\ -y\boldsymbol{r}_2^{\mathrm{T}} - z\boldsymbol{r}_3^{\mathrm{T}} \end{pmatrix} \boldsymbol{K}^{\mathrm{T}}\boldsymbol{L}_i$$

By the direct linear transform formulation:

$$\begin{bmatrix} 0 & -w_i(\boldsymbol{K}^{\mathrm{T}}\boldsymbol{L}_i)^{\mathrm{T}} & (\boldsymbol{K}^{\mathrm{T}}\boldsymbol{L}_i)^{\mathrm{T}} \\ w_i(\boldsymbol{K}^{\mathrm{T}}\boldsymbol{L}_i)^{\mathrm{T}} & 0 & 0 \end{bmatrix} \begin{pmatrix} \boldsymbol{r}_1 \\ \boldsymbol{r}_2 \\ -y\boldsymbol{r}_2 - z\boldsymbol{r}_3 \end{pmatrix} = \boldsymbol{0}$$

Apparently, only the first equation is relevant, so for each i, we have:

$$[(\boldsymbol{K}^{\mathrm{T}}\boldsymbol{L}_i)^{\mathrm{T}}\boldsymbol{r}_2 \;\; (\boldsymbol{K}^{\mathrm{T}}\boldsymbol{L}_i)^{\mathrm{T}}\boldsymbol{r}_3]\begin{pmatrix} y \\ z \end{pmatrix} = -w_i(\boldsymbol{K}^{\mathrm{T}}\boldsymbol{L}_i)^{\mathrm{T}}\boldsymbol{r}_2$$

Piling up the equations and applying singular value decomposition (SVD), the least square solution of y and z is obtained. Note that there is only one useful equation from each line correspondence and only two line correspondences are linearly independent. Thus, we have just enough equations for the two unknowns.

Likewise, counting the overall number of equations used and degree of freedom is interesting too. All in all, four linearly independent equations coming from a pencil of lines and one extra constraint of equal spacing add up to a total of five independent constraints. Correspondingly, there are exactly five degrees of freedom to be determined.

In a nutshell, by considering the geometric meaning of vanishing features, this algorithm nicely by-passes the non-linearity within the quadratic equations that define rotation matrix, and hence makes the process of calibrating the camera's extrinsic parameters more efficient.

3.3.3 From camera to UAV

The geometry between camera and UAV

By assuming the camera center to be coincide with the UAV's center of gravity, the x and y we obtained above is the UAV's x and y. The pose however, requires a further transformation. Assuming camera and UAV body are connected rigidly, there is a unique transformation that maps takes the calibrated rotation matrix of the camera to the UAV body frame and vice versa. In fact, we only need to consider a rotation matrix $\boldsymbol{R}_{UAV2Cam}$, which is defined as a rotation that maps point $\boldsymbol{x}_{UAV}$ in UAV's body frame to point $\boldsymbol{x}_{cam}$ in camera frame.

$$\boldsymbol{x}_{Cam} = \boldsymbol{R}_{UAV2Cam}\boldsymbol{x}_{UAV}$$

Then the UAV's attitude with respect to ground frame is encapsulated in

$$\boldsymbol{R}_{UAV} = (\boldsymbol{R}_{UAV2Cam}^{\mathrm{T}}\boldsymbol{R}_{Cam})^{\mathrm{T}} \tag{3.7}$$

and the inverse relation:

$$\boldsymbol{R}_{Cam} = \boldsymbol{R}_{UAV2Cam}\boldsymbol{R}_{UAV}^{\mathrm{T}} \tag{3.8}$$

Following the aerospace convention of Euler angle, the rotation matrix can be expressed by the sine and cosine of the three Euler angles, namely, yaw(ψ), pitch(θ), roll(ϕ).

$$\boldsymbol{R} = \begin{bmatrix} \cos\theta\cos\psi & -\cos\theta\sin\psi + \sin\phi\sin\theta\cos\psi & \sin\phi\sin\psi + \cos\phi\sin\theta\cos\psi \\ \cos\theta\sin\psi & \cos\theta\cos\psi + \sin\phi\sin\theta\sin\psi & -\sin\phi\cos\psi + \cos\phi\sin\theta\sin\psi \\ -\sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta \end{bmatrix} \tag{3.9}$$

Thus given R we can inversely calculate ϕ θ ψ as follows:

$$\begin{aligned} \psi &= \tan^{-1}\frac{r_{21}}{r_{11}} \\ \theta &= \sin^{-1} r_{31} \\ \phi &= \tan^{-1}\frac{r_{32}}{r_{33}} \end{aligned}$$

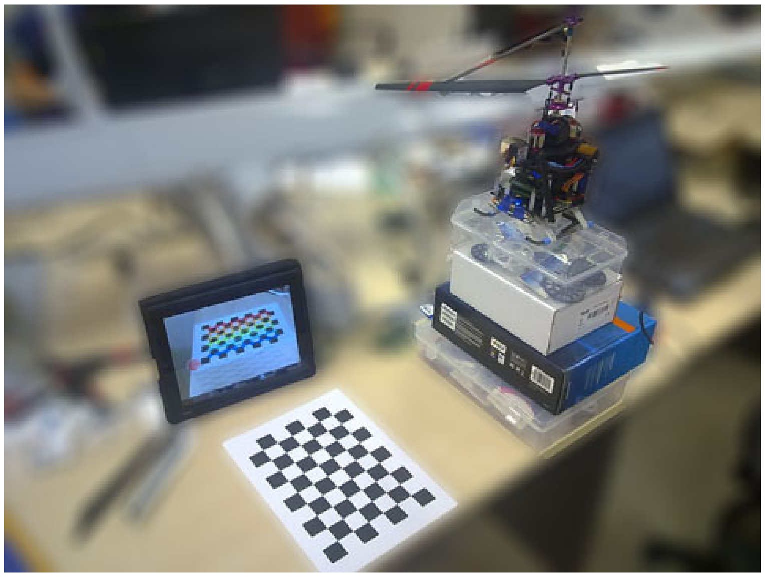


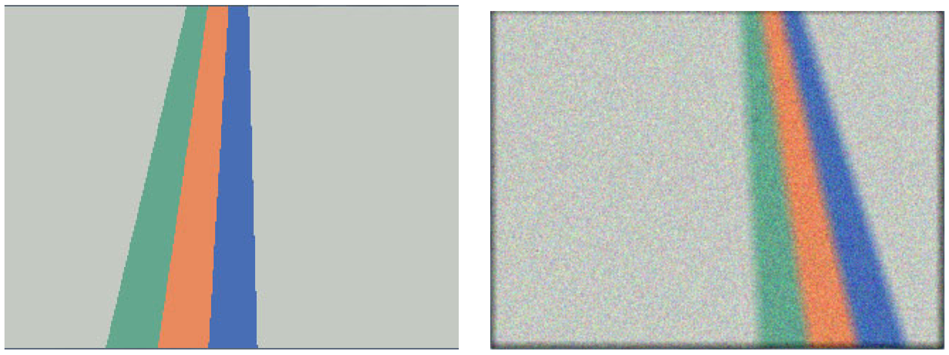
Figure 3.11: Calibrating the camera-UAV relation

Off-line calibration

Now the problem is how to find $\boldsymbol{R}_{UAV2Cam}$. It is difficult to mount the camera to UAV accurately to a specific orientation. That is why a calibration program using vision is implemented. By placing a checkerboard on the ground,and placing the UAV body in a way such that it see the pattern and its body frame aligns to the grid direction. Under this setup, we calibrate the camera's external parameters by looking at the checkerboard. Then the matrix $\boldsymbol{R}_{UAV2Cam}$ is the transpose of the calibrated rotation matrix $\boldsymbol{R}$. Again, iPad GCS is used to provide user friendly interface for this function. An illustration is given in 3.11.

3.4 Simulation and flytest

In order to verify the integrity of the algorithm, a series of simulation and fly-tests were carried out.



a. Ideal image b. Blurry and noisy image

Figure 3.12: Illustration of simulation conditions

3.4.1 Simulation

Methodology

In the simulation, a virtual UAV flies according to a predefined trajectory above the track. Video frames generated by the virtual onboard camera are then fed into the vision algorithm realized using OpenCV.

For comparison, two fly conditions are assumed: (a) ideally stable quasi-static flight with normal lighting condition; (b) UAV flight with vibration and high-ISO camera configuration (simulated using blurring and Gaussian noise). A sample of these images is given in Fig. 3.12.

Simulation Results

The results of both conditions are shown in Fig. 3.13. As we may observe, the measurement (solid blue line) follows the reference (dashed red line) closely. There is no significant difference between ideal case and deteriorated case. This implies that the algorithm is robust against harsh conditions.

3.4.2 Fly Test

Methodology

Similar to simulation, two experiments are conducted:

(1) Hand-held UAV motion: for easy maneuver of decoupled testing of different channels.

(2) R/C UAV flight: actual application scenario; performance subjected to severe vibration and noise.

In both experiments, measurements are compared to the IMU reading of yaw/pitch/roll angles.

Fly Test Results

The angle measurements of vision and IMU are plotted in comparison in Figure 3.14. In both experiments, vision provides sensitive and drift-free yaw and pitch readings. The performance is comparable to, if not better than that of IMU. Roll angle, however, is noisier than the IMU readings and exhibits fallacious behavior when the angle gets large. The position measurements of the first experiment is also plotted (see Figure 3.15), motion on both directions during the experiment are reflected as it is on the plot. The average frame rate in the two experiments is 25.6 FPS. Such real-time performance is sufficient for UAV's indoor navigation.

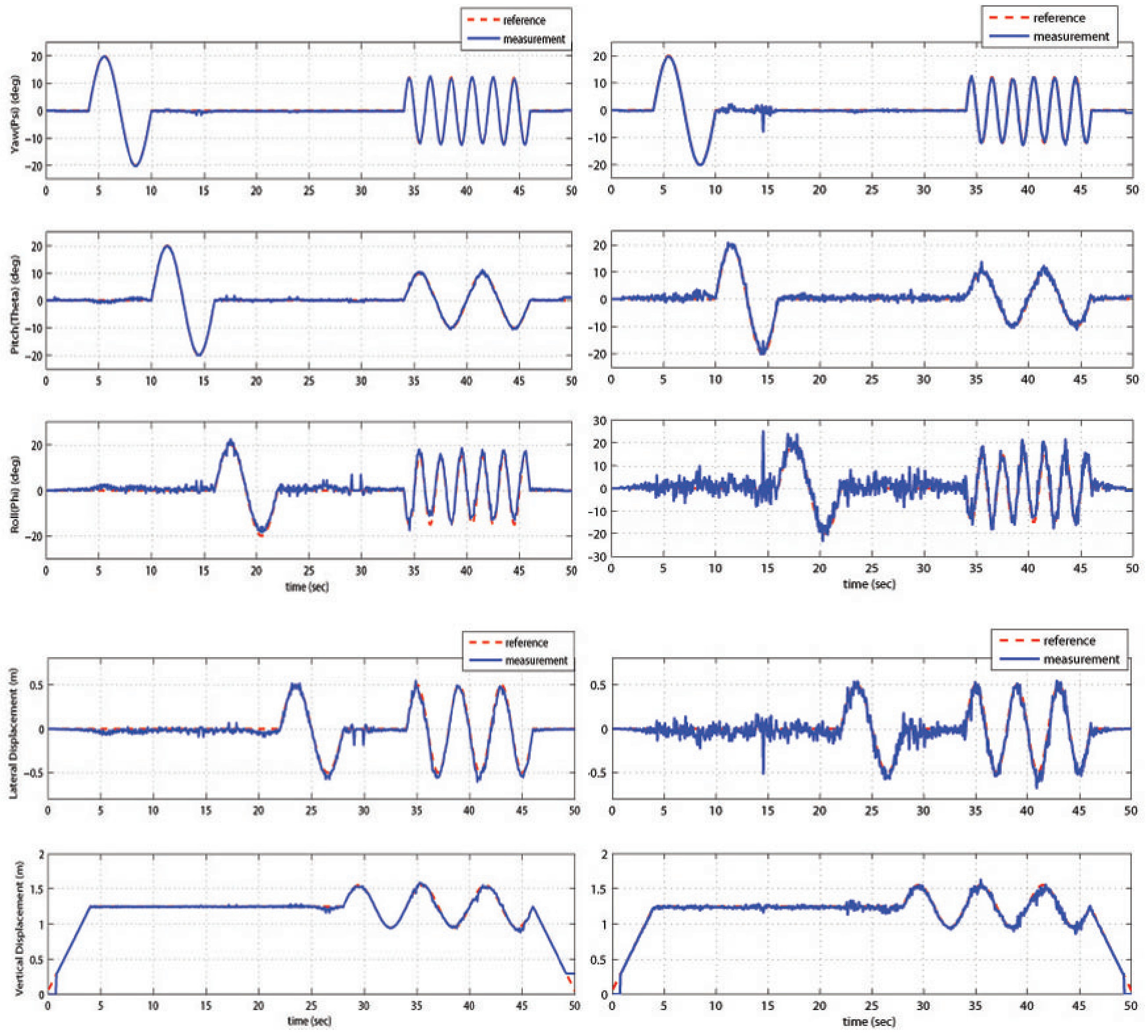


Figure 3.13: Simulation Results.
Left: Ideal Image; Right: Blurry AND noisy image.

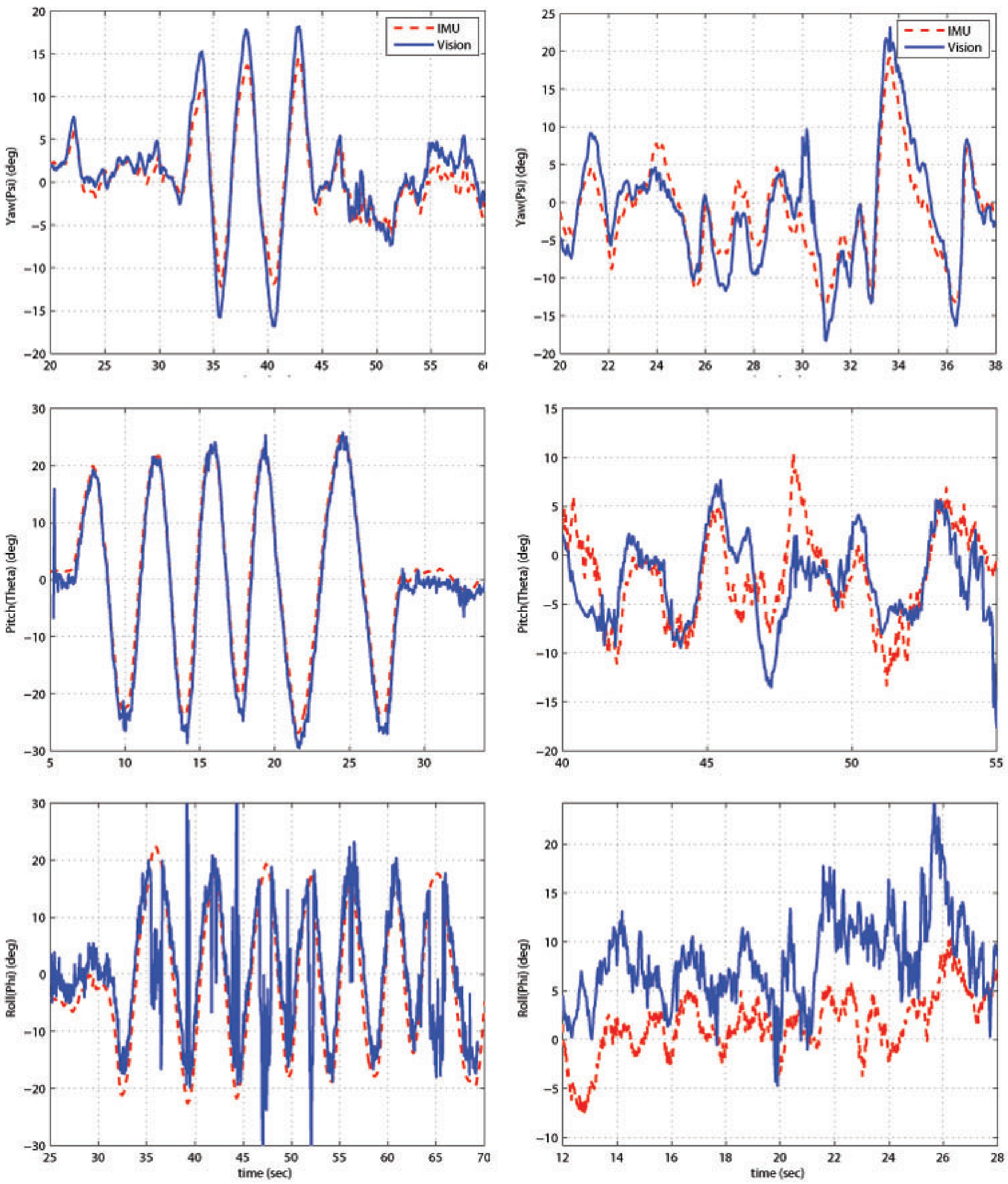
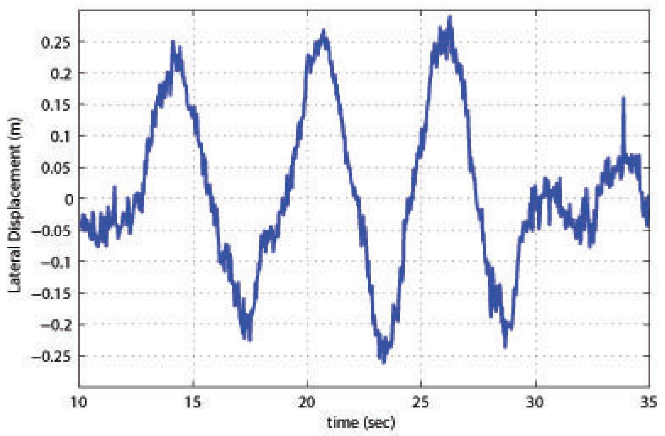


Figure 3.14: FlyTest Results. Left: Hand-hold Motion; Right: R/C Motion.



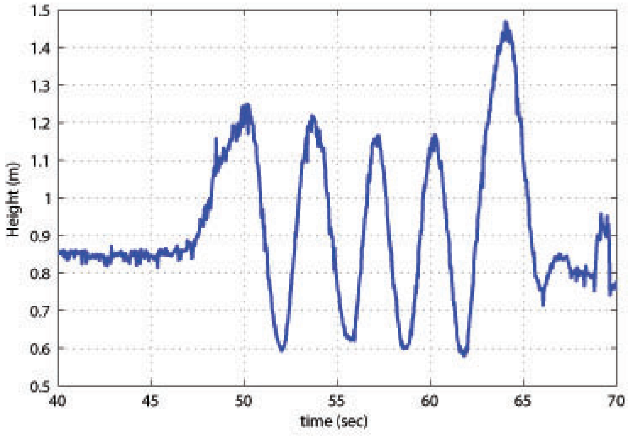


Figure 3.15: Hand-held Motion Position Plot. Left: Deviation; Right: Height.

Chapter 4

Sensor fusion: combining IMU measurement

4.1 Weakness of vision-only measurement

Through simulation and experiments, we have discovered the strength and weakness of the vision-only algorithm. Among all five measurements, z, ψ and θ are always good while ϕ and y measurements are noisier and sometimes contain large outliers. It turns out that the symptom gets worse when UAV ascends in height. It also occurs that the measurements become unstable when one of the color bands is truncated by image boundary as in Figure 4.1.

As the algorithm makes use of the equal spacing condition, it is natural that the results appear more sensitive to noise than the calculation of vanishing point. When UAV gains height, width of each color band appears only a few pixels on image. Bear in mind that it is not the width that matters, but rather the relationship between each widths. In extreme cases, plus and minus 2 pixels in the line position can be detrimental.

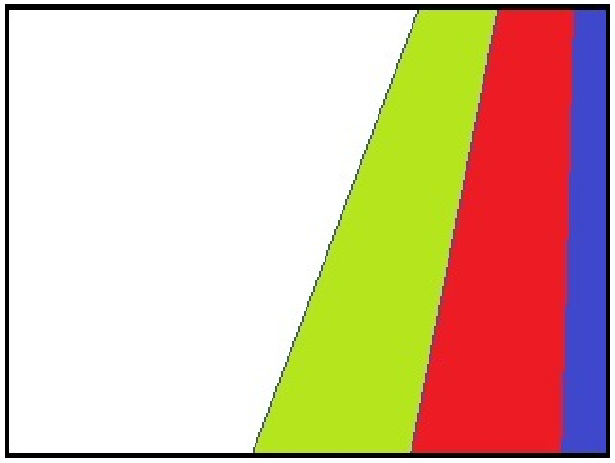


Figure 4.1: Boundary condition: right most line not visible

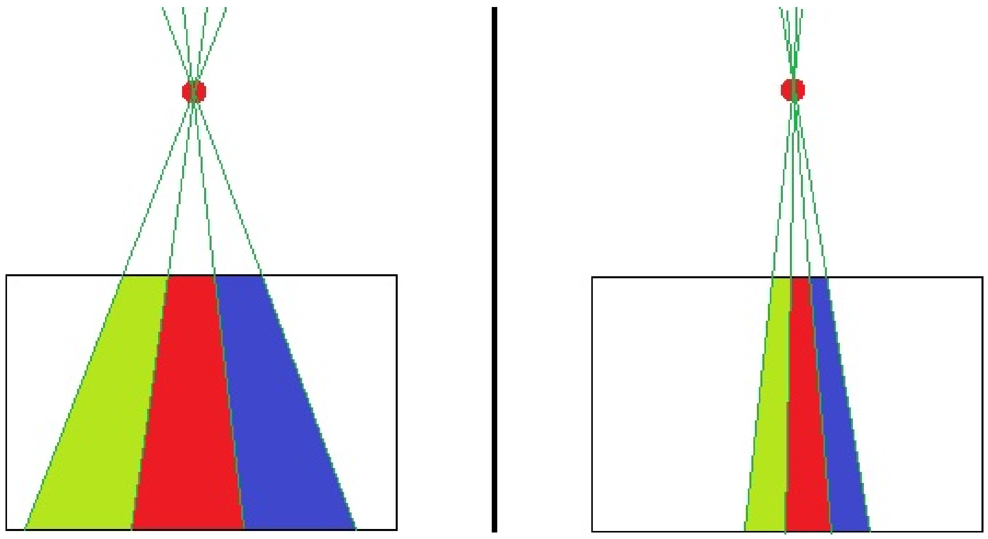


Figure 4.2: Illustration of difference in height

Likewise, the estimation of pitch is also affected by the increasing height of UAV. It is easy to visualize that the pitch measurement is closely related to the Y coordinate of vanishing point. As is shown in Figure 4.2, Y coordinate of vanishing point will be less accurate because the lines in graph appear less oblique. As a result, perspectivity is weakened.

The error in pose estimation will cause the estimation of y and z to be inaccurate too. y is noisier than z because y has larger correlation to ϕ measurement.

Empirically, when UAV is flying below 1.5 meters, vision measurements are good. When it reaches 2 meters and above, noise in ϕ and y becomes unacceptable.

4.2 Using IMU attitude to refine pose estimation

Having identified the crux of the issue, an instant solution emerges naturally, which is to combine the IMU readings. Nowadays, low cost inertial sensors powered by MEMS technology have become sufficiently mature. The IMU used in this project can reliably provide measurements of flight attitude (ϕ and θ) at 50 Hz. By combining the vanishing point from vision and attitude from IMU, we can calculate camera pose in a more reliable way.

4.2.1 What does vanishing point alone tells us?

We have derived in Section3.3 the relationship between rotation matrix and vanishing features which comprise of one vanishing point and one vanishing line. In fact, the vanishing point alone offers some interesting inference on flight attitude. By (3.5), $\boldsymbol{r}_1 = \dfrac{\boldsymbol{K}^{-1}\boldsymbol{X}_\infty}{||\boldsymbol{K}^{-1}\boldsymbol{X}_\infty||}$. So the first column of $\boldsymbol{R}_{Cam} = [\boldsymbol{r}_1, \boldsymbol{r}_2, \boldsymbol{r}_3]$ is known. Recall that in (3.7),

$$\boldsymbol{R}_{UAV} = (\boldsymbol{R}^{\mathrm{T}}_{UAV2Cam}\boldsymbol{R}_{Cam})^{\mathrm{T}} \tag{4.1}$$

So the first row of $\boldsymbol{R}_{UAV}$, denoted by $\boldsymbol{h}^{\mathrm{T}}$, can be calculated by:

$$\boldsymbol{h}^{\mathrm{T}} = (\boldsymbol{R}^{\mathrm{T}}_{UAV2Cam}\boldsymbol{r}_1)^{\mathrm{T}}$$

Expand $\boldsymbol{h}$ using direct cosine format as in (3.9), we have:

$$\boldsymbol{h} = \begin{pmatrix} h_1 \\ h_2 \\ h_3 \end{pmatrix} = \begin{pmatrix} \cos\theta\cos\psi \\ -\cos\theta\sin\psi + \sin\phi\sin\theta\cos\psi \\ \sin\phi\sin\psi + \cos\phi\sin\theta\cos\psi \end{pmatrix}$$

So, the yaw angle ψ can be expressed directly with entries of $\boldsymbol{h}$ and IMU pose ϕ and θ:

$$\psi = \arctan(\frac{\sin\theta\sin\phi}{\cos\phi} - \frac{h_2\cos\theta}{h_1\cos\phi})$$

Note that when IMU reading is not available, we can assume $\phi = 0$ or $\phi = \theta = 0$ and proceed with the algorithm. Typically, when UAV flies higher than 2 meters, it is advisable to make near-hovering assumption so as to prevent large error in the estimated roll from confounding the y, z measurement.

When $\phi = 0$, it is easy to show that:

$$\psi = -\arcsin h_2$$
$$\theta = \arctan\frac{h_3}{h_0}.$$

When $\phi = \theta = 0$, then

$$\psi = -\arctan\frac{h_2}{h_0}.$$

4.3 Patching the algorithm: using a subset of features

Although the geometry of the track contains four parallel lines, we don't have to use all four lines to compute the 5 unknowns.

Two parallel lines are enough to intersect at vanishing point. Three equally spaced parallel lines are sufficient for vanishing line. Therefore, when IMU readings are given, two lines are enough to compute camera pose. As for the constrained estimation of y an z, two equations are sufficient according to our formulation in Section 3.3.2.

Essentially, we can make use of this property to patch the algorithm so that the boundary condition described in Figure 4.1 is no longer an issue.

By counting the number of non-boundary sample points, whether a line is well-conditioned is obtained. The pose estimation and localization problems are then classified into four cases.

(1) All four lines are well-conditioned.

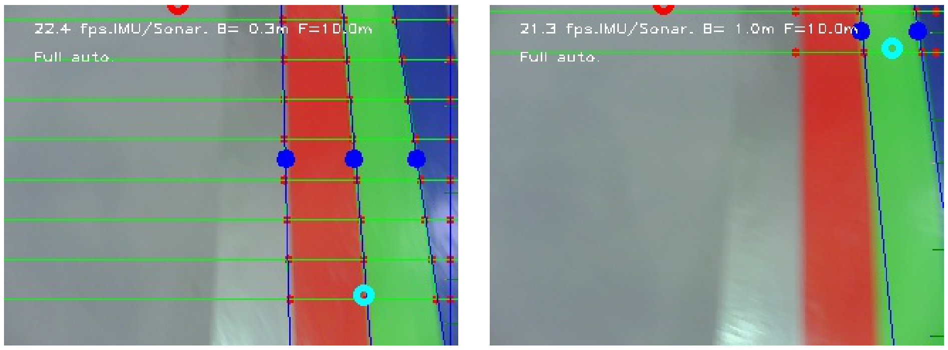
(2) Only left three lines are well-conditioned.

(3) Only right three lines are well-conditioned.

(4) Only middle two lines are well-conditioned.

For each case, our pose estimation and localization algorithms are formulated differently.

This is the actual algorithm that is implemented on MerLion, an illustration is given in Figure 4.3.


22.4 fps.IMU/Sonar. B= 0.3m F=10.0m
Full auto.
21.3 fps.IMU/Sonar. B= 1.0m F=10.0m
Full auto.

(a) case(2): Using left 3 lines (b) case(4): Using middle 2 lines

Figure 4.3: Subset estimation triggered in actual flight.

4.4 Velocity estimation via Kalman Filtering

Fusion of vision and IMU measurements can be further extended to the velocity estimation of UAV. By the law of physics, velocity $v = \frac{dx}{dt} = v_0 + \int adt$. Now that x is measured by vision and acceleration a is given by IMU. We should be able to calculate v by either differentiate x or integrate a.

In reality, however, differentiation of x directly will amplify the noise and integration of a will suffer from severe drift. In order to filter out the noise and compensate the drift at the same time, we borrowed the Discrete Kalman Filter in [27].

4.4.1 Discrete Kalman Filtering

The general discrete-time linear system is formulated as follows,

$$\begin{cases} \boldsymbol{x}(k+1) = \boldsymbol{A}(k)\boldsymbol{x}(k) + \boldsymbol{B}(\boldsymbol{u}(k) + \boldsymbol{w}(k)), \\ \boldsymbol{y}(k) = \boldsymbol{C}(k)\boldsymbol{x}(k) + \boldsymbol{v}(k). \end{cases} \tag{4.2}$$

where $x \in R^n$, $u \in R^p$ and $y \in R^m$ are state, input and measurement. $\boldsymbol{A}(k)$,$\boldsymbol{B}(k)$ and $\boldsymbol{C}(k)$ are system matrices. w and v are input and measurement noise, which is assumed to be zero-mean Gaussian noise. Assume (4.2) is observable, Kalman filtering is performed via time update and measurement update every time a new measurement is available.

Time update

$$\begin{cases} \hat{x}(k|k-1) = \boldsymbol{A}\hat{\boldsymbol{x}}(k-1) + \boldsymbol{B}\boldsymbol{u}(k-1), \\ \boldsymbol{P}(k|k-1) = \boldsymbol{A}\boldsymbol{P}(k-1)\boldsymbol{A}^{\mathrm{T}} + \boldsymbol{B}\boldsymbol{Q}\boldsymbol{B}^{\mathrm{T}}. \end{cases} \tag{4.3}$$

Measurement update

$$\begin{cases} \boldsymbol{H}(k) = \boldsymbol{P}(k|k-1)\boldsymbol{C}^{\mathrm{T}}(\boldsymbol{C}\boldsymbol{P}(k|k-1)\boldsymbol{C}^{\mathrm{T}} + \boldsymbol{R})^{-1}, \\ \hat{\boldsymbol{x}}(k) = \hat{\boldsymbol{x}}(k-1) + \boldsymbol{H}(k)(\boldsymbol{y}(k) - \boldsymbol{C}\hat{\boldsymbol{x}}(k|k-1), \\ \boldsymbol{P}(k) = (\boldsymbol{I} - \boldsymbol{H}(k)\boldsymbol{C})\boldsymbol{P}(k|k-1) \end{cases} \tag{4.4}$$

In the equation, $\boldsymbol{H}(k)$ is the feedback gain matrix, and a priori $\boldsymbol{P}(k|k-1)$ is the covariance of the state estimation error, which is defined in the following formula:

$$\begin{cases} \boldsymbol{P}(k|k-1) = E\left\{[\boldsymbol{x}(k) - \hat{\boldsymbol{x}}(k|k-1)][\boldsymbol{x}(k) - \hat{\boldsymbol{x}}(k|k-1)]^{\mathrm{T}}\right\}, \\ \boldsymbol{R}(k) = E\left\{\boldsymbol{v}(k)\boldsymbol{v}^{\mathrm{T}}(k)\right\}, \\ \boldsymbol{Q}(k) = E\left\{\boldsymbol{w}(k)\boldsymbol{w}^{\mathrm{T}}(k)\right\}. \end{cases} \tag{4.5}$$

4.4.2 Ground Velocity Estimation

In our vision system, the following model is developed based on the kinematical model of the helicopter. In this model,

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{I} & T_s\boldsymbol{I} \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix}, \boldsymbol{B} = \begin{bmatrix} \frac{T_s^2}{2}\boldsymbol{I} \\ T_s\boldsymbol{I} \end{bmatrix}, \boldsymbol{C} = \begin{bmatrix} \boldsymbol{I} & 0 \end{bmatrix} \tag{4.6}$$

where T_s is the sampling period.

Let p_g, v_g be the position and velocity in world frame, a_{nb} be the IMU measured body frame acceleration and $\boldsymbol{R}_{gb}$ is the rotation matrix that transform world frame vector to body frame vector, then the state variable $\boldsymbol{x} := \begin{pmatrix} p_g \\ v_g \end{pmatrix}$, vision measurement $\boldsymbol{y} := p_g$, input $\boldsymbol{u} := \boldsymbol{R}_{gb}^T \boldsymbol{a}_b + g \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$.

Following the computation in 4.3 and 4.5, we can perform the Kalman filtering for every loop and obtain an optimal estimation of current velocity as in $\hat{\boldsymbol{x}}(k+1|k)$.

Since initial condition is not important in Kalman filtering, initial state variable and acceleration are assigned to null when the filter first starts.

Performance of this Kalman filtering formulation is verified and discussed in the following session.

4.5 Experiments

4.5.1 Methodology

To verify the improved algorithm, fly-test experiments are conducted again. As in Section 3.4.2, the experiments are conducted in two ways:

(1) Hand-hold oscillation

(2) R/C flight

An automatic flight test is also conducted. The results are omitted in Section 4.5.2 and 4.5.3 because for vision/IMU measurements, there is no difference to (2) in terms of test conditions. In Section 4.5.4, however, velocity estimations of this automatic flight is given.

In the plots, yaw measurements are plotted in comparison to the IMU measured yaw angle. Similarly, height measurements are compared to the output of a sonar sensor. In Section 4.5.4, velocity measured using Kalman Filter is compared to two other ways of calculating velocity, namely, integration of acceleration and differentiation of position.

For integration, simple summation of acceleration value times sampling time is used. For differentiation, we adopt a Washout Filter which makes use of the current velocity and the difference between the latest two position measurement. In this way, we managed to reduce the noise of this differentiation process to certain extent.

4.5.2 Pose Plot

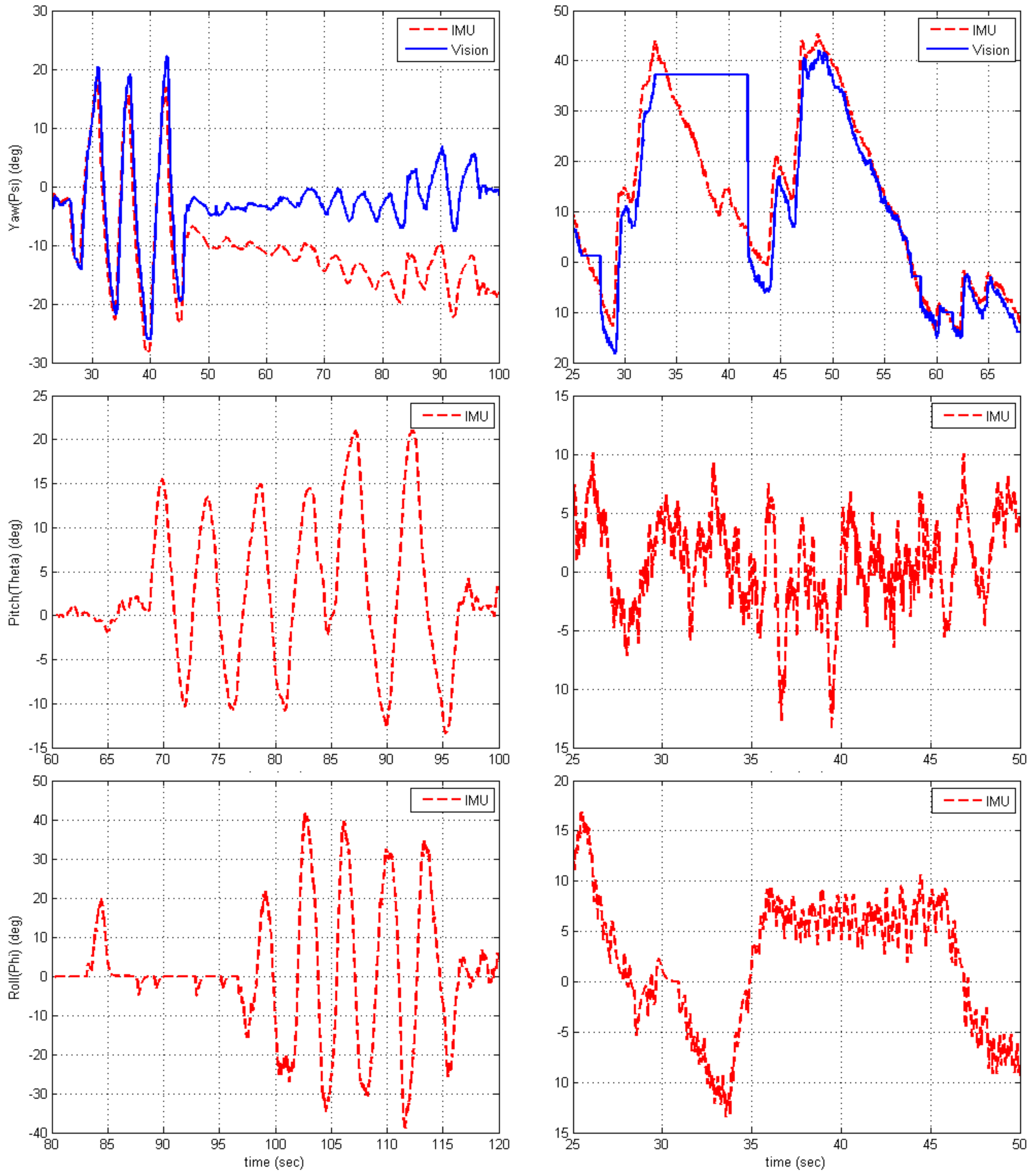


Figure 4.4: Fly-test results: Pose.
Left: Hand-held Oscillation; Right: R/C Test Flight,

4.5.3 Position Plot

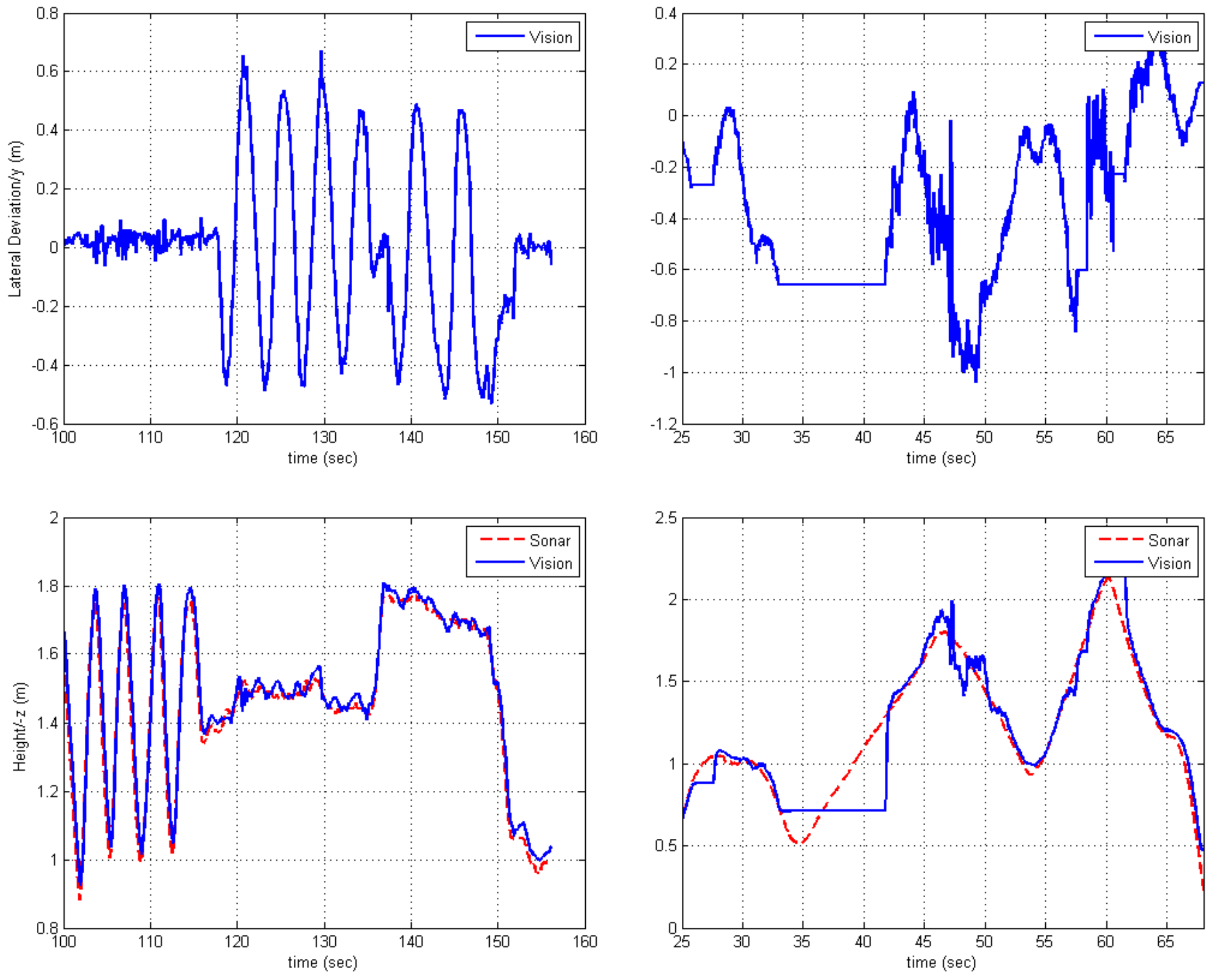
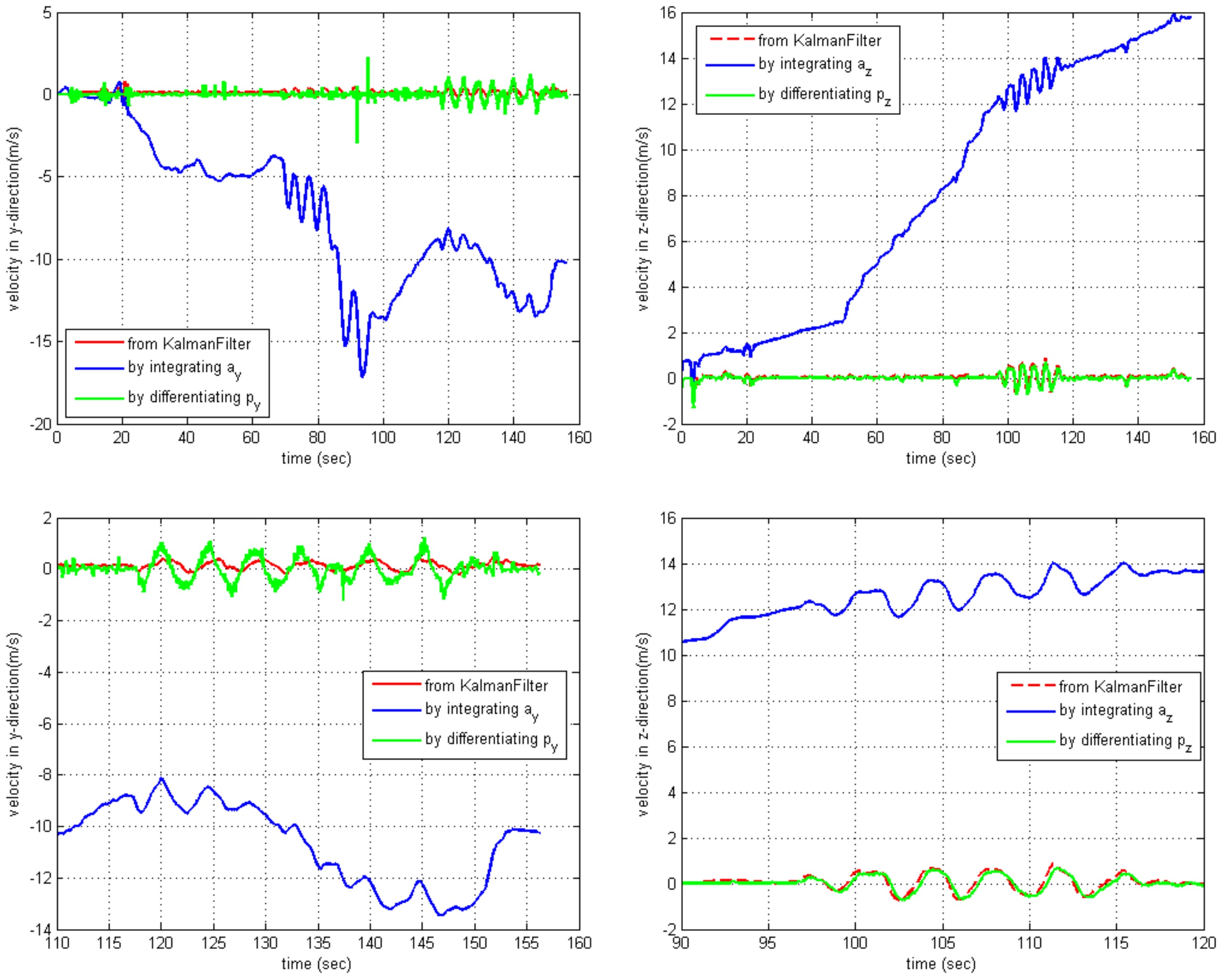


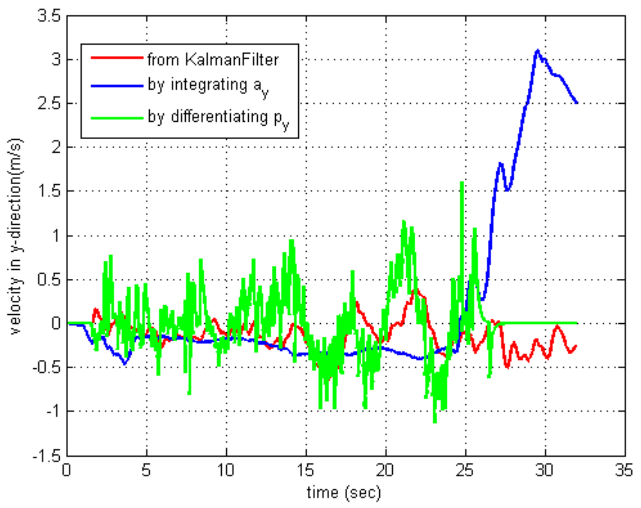
Figure 4.5: Fly-test results: Position.
Left: Hand-held Oscillation; Right: R/C Test Flight,
From left to right, top to bottom: (a),(b),(c),(d).

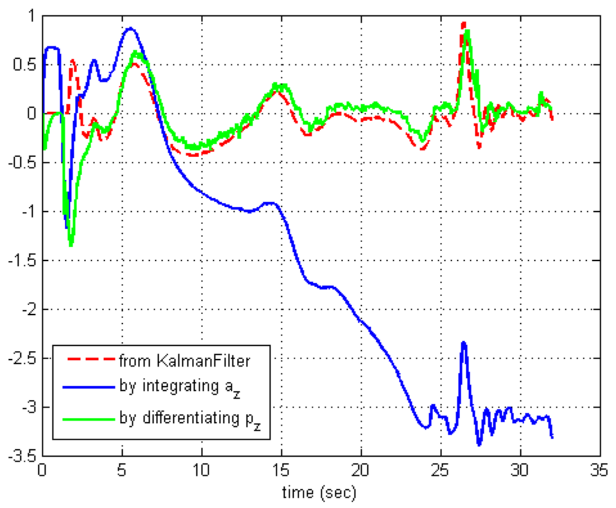
4.5.4 Velocity Plot



Left:(a) lateral velocity; Right:(b) vertical velocity,
Left:(c) Zoomed-in view of v_y; Right:(d) Zoomed-in view of v_z,

Figure 4.6: Comparison velocity in Hand-held oscillation.





Left:(a) lateral velocity; Right:(b) vertical velocity,

Figure 4.7: Comparison velocity in automatic flight.

4.5.5 Discussions

On Pose Estimation

As the IMU on MerLion does not contain magnetometer, it suffers from drift in heading angle all the time. The degree of drift varies. It can be very large (as in Figure 4.4(a)), or very small (as in Figure 4.4(b)). ψ from vision however, is always drift-free.

As for pitch(θ) and roll(ϕ), the IMU output is very good at hand-held motion and acceptable during actual flight. Note that there is no significant outliers in ϕ measurements similar to Figure 3.14.

On Position Estimation

This set of plots suggested that, by combining IMU output of θ and ϕ in the constrained localization, measurements of y, z have become less sensitive to change of height. This is illustrated in Figure 4.5. As the height rises from 1.2m to 1.8m at about $t = 136s$, there is no significant disparity between the two portions of measurements. Although in overall, the performance/noisiness of the measurements are similar to vision-only algorithm.

On Velocity Estimation and Kalman Filtering

As we can see from the velocity plots, the integration approach drifts very fast. While differentiation approach gives drift free measurements, the noise is usually great especially when the UAV is in flight and suffers a lot of vibration. The discrete Kalman filter integrates the two approaches and comes up with good results.

While the drifts are compensated well, we may notice that the middle value of Kalman filter output is sometimes not zero from the zoomed-in plots (e.g., Figure 4.6(c)). There is a constant small bias in the results it provides. This is similar to the concept of steady state error in compensating drift and will certainly cause problems in control if the bias becomes significant.

Another observation is that the normal noise assumption we made in formulating Kalman filter is not necessarily true in real reality. Due to the nature of vision measurements, there are sparse but significance outliers in measurements especially when the track goes out of sight. Usually, it takes a long time for Kalman filter to re-converge to the correct estimation.

Chapter 5

Track following algorithm

Besides feeding measurements, the vision is also in charge of sending command to controller about how to follow the track, namely, vision navigation. It involves not only setting reference, but also how to continue following the track in case of turning.

5.1 Back-projection of image point

Before going to the actual algorithm, let us first introduce the back-projection of image point.

Usually, 3D world point cannot be uniquely determined given one image point. This world point can be anywhere on the ray connecting camera center and image point. However, in this special case, we can, because the objects of interest are all located on ground plane.

According to (3.4) of Section 3.3, it is a 3×3 homography $\boldsymbol{H}$ that maps the the world point $\boldsymbol{x} = [x, y, 1]^{\mathrm{T}}$ on ground plane to the image point $\boldsymbol{X} = [X, Y, 1]^{\mathrm{T}}$.

The back-projection equation is:

$$\boldsymbol{x} \propto \boldsymbol{H}^{-1}\boldsymbol{X} \tag{5.1}$$

This equation offers us flexibility in calculating the world coordinate of any object in the image.In other word, if we can detect the end of track whenever it appears in the image, the vision measurements are automatically upgraded to full six d.o.f.

5.2 Turning detection and classification

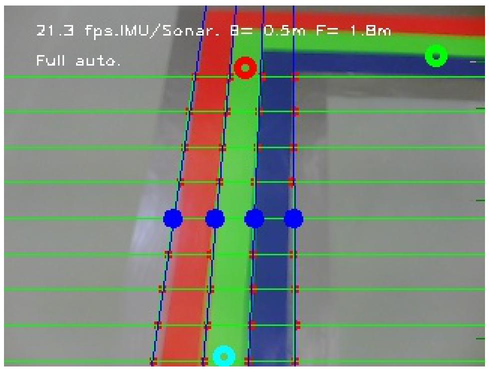
Recall the line detection algorithm described in Section 3.2.1, we searched for the end of track in image and conducted 1D edge detection for each sample strip. Two pieces of information can be used here in our algorithm to detect turning corners.

(1) The "end of track point" $\boldsymbol{X}$ in the image can be back-projected to 3D world and show us precisely how far away the end of track (if existed) is to the UAV.

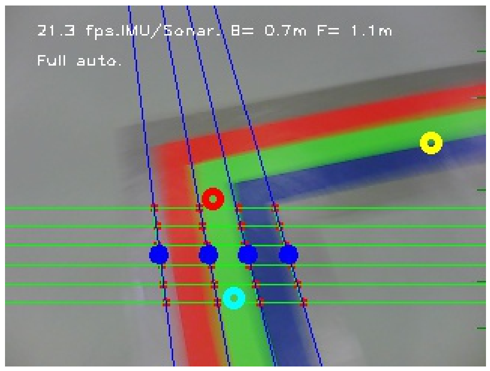
(2) The 1D edge detector can be reused for vertical strip of pixels to determine if the corner is a left turn or right turn.

Referring to Figure 5.1, signals of turning are issued whenever left or right turner is detected and the corner is within a pre-defined distance. The UAV receiving such signal will turn immediately towards the next portion of track. Usually, the measurements resume right away.

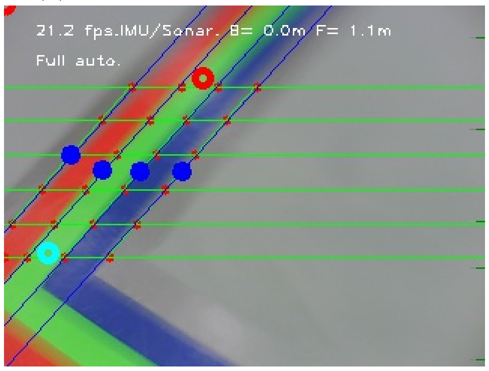
Sometimes when track appears very slanted to one side, as in Figure 5.2, there might be false detections of frontal distance x (see the "F=0.9" in Figure 5.2). Detection for left or right turnings will be triggered if this x is small enough. Note



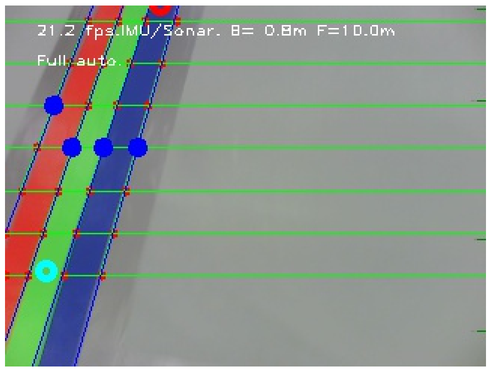
(a)Turning corner detected at 1.8m.



(b)Issue command when corner gets near enough



(c)Turned and identified new track right away.



(d)continue to follow track.

Figure 5.1: An illustration of turning detection.

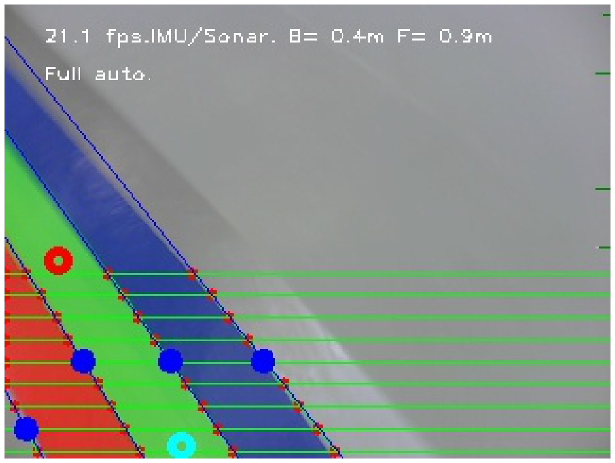


Figure 5.2: Illustration of possible false turning

Category	**Error Residual**
False detection of turning	0.6-3
True detection of turning	100-1000

Table 5.1: Classification of true and false detection

that if we take the left most pixel strip and conduct 1D edge detection, we will find the same [Blue Green Red] pattern as if there is a turn on the left.

Fortunately, such false alarms can be reliably distinguished from true turnings by testing whether the edge points of the vertical strip are lying on the corresponding lines features which we obtained earlier. The pixel distance between point and line can be calculated using $\frac{|\boldsymbol{X}\cdot\boldsymbol{L}|}{\sqrt{L_1^2+L_2^2}}$ if the point and line are represented in their homogeneous representation $\boldsymbol{X} = [X_1, X_2, 1]^{\mathrm{T}}$ and $\boldsymbol{L} = [L_1, L_2, 1]^{\mathrm{T}}$.

Based on our experiments, the average sum of square distances for true and false detections are given in Table 5.1.

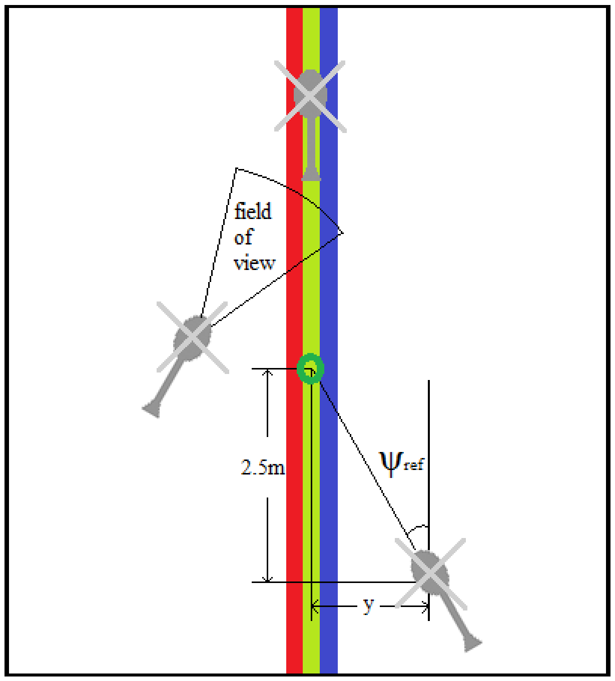


Figure 5.3: Illustration of Point Tracking

5.3 Keeping the landmark in the scene: Point tracking

Another prevailing issue about vision-based navigation is that the landmark may go outside the image frame. This is usually done by panning the camera manually or using a servo motor. In our case, as we assumed rigid connection between camera and UAV body, this configuration is not possible.

Given the fact that the airborne system is highly unstable by itself, we cannot rule out the possibility that the UAV will deviate away from the track from time to time. Although placing the camera 45 degree downwards solved the problem to some extent (compared to former works that look vertically downwards), it is still not sufficient to keep the track inside the field of view all the time.

The solution is simple. Instead of keeping the heading reference constant towards the x direction, we let the UAV to select the reference adaptively based on its current position with respect to the track. This heading reference is designed such that the UAV is always facing an imaginary point 2.5 meters away on the track. An illustration is given in Figure 5.3.

The formula to calculate heading reference ψ_{ref} is

$$\psi_{ref} = -\arctan(y/2.5)$$

This simple and elegant solution ensures that the measurements that are given to controller are continuous and available at all time. This is crucial to the precision control and maneuverability of the UAV.

Chapter 6

Conclusion

In this report, we described the design and implementation of a vision-based sensory system that provides multiple dimensions of measurements for indoor UAV and proper guidance for it to fly autonomously along a colored track. Let us reiterate the major achievements and innovations in this project.

First of all, through a complete study of the geometry within four equally spaced parallel lines, we provide critical insight into how hidden geometric entities such as a vanishing line (horizon) in indoor environment can provide useful information. Specifically, a new pose estimation and self-locating algorithm is proposed and discussed. This algorithm features an innovative linear-time line detection technique, an unconventional vanishing line estimation method, a constrained localization formulation and the derivation of the analytical expression of rotation matrix using geometry at infinity. Simulation and fly-test are conducted and the results are verified.

Secondly, based on the fly-test results, we identified the shortcomings of this vision-only algorithm and improved the algorithm by combining the readings of IMU. This provides considerable refinement in vision measurements. Furthermore, we explored the use of Kalman filtering in estimating the velocity of UAV. All these measurements are verified in the second fly-test.

Thirdly, we developed a set of practical methods to reliably fly along a colored track with left and right turnings. On both hardware level and software level, we attempted to extend the field of view so as to provide timely feedback throughout the course.

Last but not least, this successful design and implementation of vision-based sensory system on Coaxial Rotorcraft MerLion has won the group the Overall Championship and Most Creative Award in the Singapore Amazing Flying Machine Competition 2011.

Chapter 7

Future Works

In the future development of this vision-based sensory system for indoor UAV, the following can be done.

Barrier Avoidance Barrier avoidance is a simple but powerful feature we can implement. It can be performed via infrared sensor, stereo vision or other depth measuring mechanism. In the crowded indoor environment, a robust barrier avoidance mechanism is essential for the safety of both robots and human beings.

Vision-based velocity estimation While Kalman filter approach of velocity estimation is powerful, there are quite a few limitations as we discussed earlier. In fact, we might be able to measure velocity using vision alone by taking into consideration the temporal and spectral information among multiple frames. Optical flow is one of the technique that we might want to explore.

Change landmark As is remarked in this thesis, the geometric information of this RGB colored track is actually very little. In particular, there is no information at all in the longitudinal direction. To empower more sophisticated research such as SLAM, or should we wish to be more robust in the track following operation, we might wish to consider changing the landmark into barcode or other things that contains richer information.

Bibliography

[1] S. K. Phang, J. J. Ong, R. T. C. Yeo, B. M. Chen, and T. H. Lee, "Autonomous mini-uav for indoor flight with embedded on-board vision processing as navigation system," *Proceedings of the IEEE R8 International Conference on Computational Technologies in Electrical and Electronics Engineering*, pp. 722–727, 2010.

[2] O. Sidla, N. Brndle, W. Benesova, M. Rosner, and Y. Lypetskyy, "Embedded vision challenges," in *Smart Cameras* (A. N. Belbachir, ed.), pp. 99–117, Springer US, 2010.

[3] J. J. Ong, "Embedded Vision Indoor Guidance System for Unmanned Aerial Vehicles," B.Eng Thesis, National University of Singapore, Singapore, 2010.

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

[5] D. Mellinger, N. Michael, and V. Kumar, "Trajectory generation and control for precise aggressive maneuvers with quadrotors," in *In Proceedings of the International Symposium on Experimental Robotics*, (Delhi, India), Dec 2010. To Appear.

[6] N. Michael, J. Fink, and V. Kumar, “Cooperative manipulation and transportation with aerial robots,” *Autonomous Robots*, vol. 30, pp. 73–86, Jan 2011.

[7] S. Hrabar and G. S. Sukhatme, “Combined optic-flow and stereo-based navigation of urban canyons for a uav,” in *In Proceedings of the IEEE/RSJ Int. Conference on Intelligent Robots and Systems*, pp. 2–6, 2005.

[8] S. Ang, “Development of Mini Unmanned Helicopters for Cooperative Tasks,” B.Eng Thesis, National University of Singapore, Singapore, 2011.

[9] G. Bradski, “The OpenCV Library,” *Dr. Dobb’s Journal of Software Tools*, 2000.

[10] M. Lourakis, “levmar: Levenberg-marquardt nonlinear least squares algorithms in C/C++.” [web page] `http://www.ics.forth.gr/~lourakis/levmar/`, Jul. 2004. [Accessed on 31 Jan. 2005.].

[11] J. Y. Bouguet, “Camera calibration toolbox for Matlab,” 2008.

[12] S. J. Lee, “Development of Comprehensive Ground Station for Unmanned Aerial Vehicle with Intelligent Mobile Device,” B.Eng Thesis, National University of Singapore, Singapore, 2011.

[13] S. Takezawa, D. C. Herath, and G. Dissanayake, “Slam in indoor environments with stereo vision,” *2004 IEEERSJ International Conference on Intelligent Robots and Systems IROS IEEE Cat No04CH37566*, vol. 38, no. 1, pp. 1866–1871, 2004.

[14] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "MonoSLAM: Real-Time Single Camera SLAM," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 29, no. 6, pp. 1052–1067, 2007.

[15] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision.* 2nd edn., The Edinburgh Building, UK: Cambridge University Press, 2003.

[16] L. L. Wang and W. H. Tsai, "Camera calibration by vanishing lines for 3-d computer vision," *Pattern Analysis and Machine Intelligence, IEEE Transactions*, no. 13, pp. 370–376, 1991.

[17] Z. Kim, "Geometry of vanishing points and its application to external calibration and realtime pose estimation," tech. rep., UC Berkeley: Institute of Transportation Studies Retrieved from: http://escholarship.org/uc/item/1m71z3t3, 2006.

[18] C. Rother, "A new approach to vanishing point detection in architectural environments," *Image and Vision Computing*, vol. 20, pp. 647–655, 2002.

[19] L. Grammatikopoulos, G. Karras, and E. Petsa, "An automatic approach for camera calibration from vanishing points," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 62, pp. 64–76, 2007.

[20] W. Xu, P. Li, and B. Han, "An attitude estimation method for mav based on the detection of vanishing point," *Proceedings of the 8th World Congress on Intelligent Control and Automation*, pp. 6158–6162, 2010.

[21] S. A. S. C. Thorpe; M. H. Hebert, T. Kanade, "Vision and navigation for the carnegie-mellon navlab," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 10, pp. 362–373, 1988.

[22] Q. Fang and C. Xie, “A study on intelligent path following and control for vision-based automated guided vehicle,” *Proceedings of the 5th World Congress on Intelligent Control and Automation*, vol. 6, pp. 4811–4815, 2004.

[23] S.-P. Liou and R. C. Jain, “Road following using vanishing points,” *Computer Vision, Graphics, and Image Processing*, vol. 39, pp. 116–130, 1987.

[24] E. Frew, T. McGee, and Z. Kim, “Vision-based road-following using a small autonomous aircraft,” *Proceedings of Aerospace Conference 2004*, pp. 3006–3015, 2004.

[25] Y. X. Wang, “An efficient algorithm for uav indoor pose estimation using vanishing geometry[preprint],” *IAPR Conference on Machine Vision Applications*, 2011.

[26] F. Schaffalitzky and A. Zisserman, “Planar grouping for automatic detection of vanishing lines and points,” *Image and Vision Computing*, vol. 18, pp. 647–658, 2000.

[27] T. H. L. F. Lin, B. M. Chen, “Vision aided motion estimation for unmanned helicopters in gps denied environments,” *Proceedings of the 2010 IEEE International Conference on Cybernetics and Intelligent Systems*, pp. 64–69, June 2010.