Target Tracking from a UAV based on Computer Vision

Yuhan Zhang

Thesis submitted to the faculty of the Virginia Polytechnic Institute and State University in partial fulfillment of the requirement for the degree of

Master of Science

In

Mechanical Engineering

Alexander Leonessa, Chair

Kevin B. Kochersberger

Steve C. Southward

April 26, 2018

Blacksburg, VA

Keywords: UAV, control, computer vision

Target Tracking from a UAV based on Computer Vision

Yuhan Zhang

Abstract

This thesis presents the design and build of tracking system for a quadrotor to chase a moving target based on computer vision in GPS-denied environment. The camera is mounted at the bottom of the quadrotor and used to capture the image below the quadrotor. The image information is transmitted to computer via a video transmitter and receiver module. The target is detected by the color and contour-based detection algorithm. The desired pitch and roll angles are calculated from the position controller based on the relative position and velocity between the moving target and the quadrotor. Interface between PC and quadrotor is built by controlling the PWM signals of the transmitter for command transmission.

Three types of position controllers including PD controller, fuzzy controller and self-tuning PD controller based on fuzzy logic are designed and tested in the tracking tests. Results on the corresponding tracking performances are presented. Solutions to improving the tracking performance including the usage of optical sensor for velocity measurement and high-resolution camera for higher image quality are discussed in future work.

Target Tracking from a UAV based on Computer Vision

Yuhan Zhang

General Audience Abstract

In this thesis, an automatic tracking system for a quadrotor based on computer vision in GPS-denied environment is studied and developed. A camera mounted on the quadrotor is used to "see" the moving target. The relative position and velocity between the quadrotor and the target can be obtained by a visual detection and tracking algorithm. Through the position controller, the desired pitch and roll angles are calculated to determine how much acceleration the quadrotor requires to chase the moving target and keeps the target within the detection range. Three position controllers are designed and tested, and their corresponding performances are compared and discussed.

Table of Contents

List of Figures

List of Tables

Chapter 1.

Introduction

Quadrotor, which is also called quadcopter, is a type of UAV (unmanned aerial vehicle). Unlike fixed wing aircrafts which generate lift force from the relative forward velocity and air pressure difference of the airfoil, a quadrotor generates lift force directly from four motors each equipped with a propeller. The four motors are generally distributed symmetrically on the quadrotor frame, two rotating in clockwise direction and two in counterclockwise direction. By changing the rotation speed of each motor, the propellers are able to generate the desired thrust and torques to reach desired attitude angles, including pitch, roll and yaw, and to realize vertical and horizontal movements. Nowadays quadrotors are obtaining increasing popularity because they are much easier to control and do not require large taking off or landing space like fixed wings airplanes. Their application areas include aerial photography, monitoring, security and other industries.

The objective of this thesis is to study and develop a method by which a quadcopter can track a moving target using visual detection and tracking, and thereby always hover above the target in a small range while maintaining a constant altitude. A flight mode called “Follow-me” mode which realizes target tracking by GPS already exists for commercial and open source flight controllers such as DJI and Ardupilot respectively. This flight mode requires a GPS USB dongle, or a Bluetooth GPS module connected to laptop. Before flight, a MAV (Micro Aerial Vehicle) link connection should be established between the quadrotor and computer using wireless telemetry. After taking off, the flight mode of the quadrotor should be switched into “Loiter” mode. In the Ground Control Station (GCS) software, which is a full-featured ground station application, by clicking on “Follow me”, the quadrotor will work under “Follow-me” mode. In this mode, the laptop sends the GPS location of its GPS module to flight controller through the telemetry link. After receiving the location information, compared with the quadrotor’s own GPS location, the quadrotor will move towards the computer GPS module and follow the computer GPS module if the

computer keeps moving. However, during testing, it was found that the quadrotor only followed the target (the computer GPS module) but could not hover directly above the target. The quadrotor cannot move simultaneously as the target moves and there was a certain distance left between the target and quadrotor, which may be caused by GPS signal transmission or flight controller response lag. Another issue is that the “Follow-me” mode requires a good GPS lock, low magnetic interference on the compass and low vibrations to achieve a good loiter performance. Thus, this mode is not available in the environment where GPS signals are poor, such as indoor environment.

To overcome these deficiencies, another method using computer vision to track the target automatically is studied and developed in this thesis. A camera is mounted on the quadrotor, and algorithm is developed to detect the target and send the movement commands to the quadrotor. The quadrotor can hover above the target and keep the target within the camera screen. GPS is not required in this method, so target tracking is capable of also working in GPS-denied environments.

Chapter 2.

System Overview

The diagram of the whole tracking system is shown in Figure 2-1.

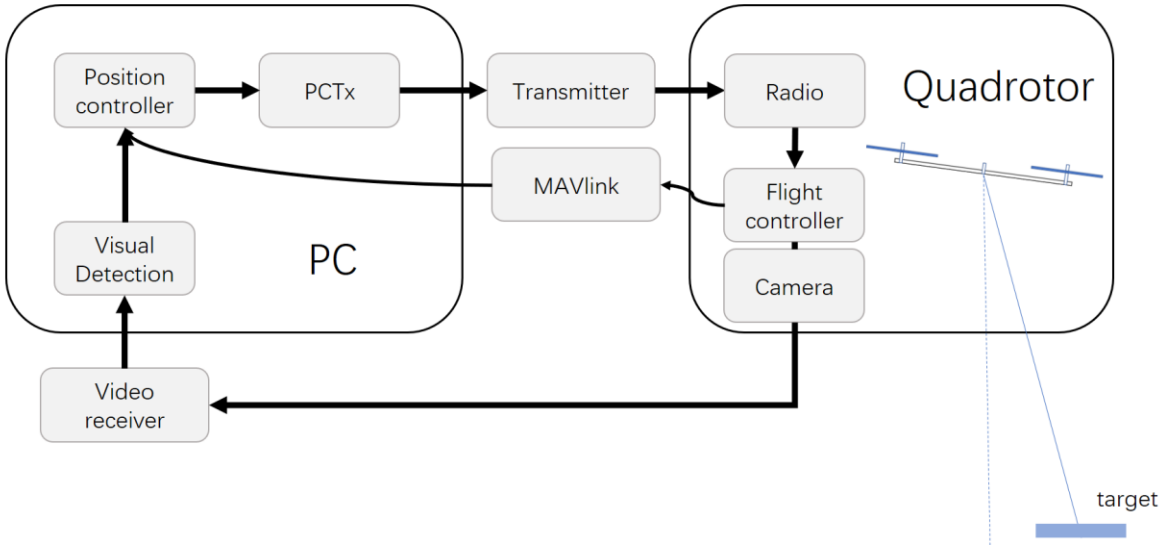


Figure 2-1 The system structure

The camera is mounted at the bottom of the quadrotor and is used to capture images below the quadrotor. Then the image frames are transmitted to a computer via a video transmitter and receiver module. Simultaneously, the information of quadrotor states including attitude angles and flight mode is transmitted to computer via telemetry using the MAVlink protocol. The computer detects the target in the image and calculates the relative position through a series of image processing functions. With the relative position calculated and the MAVlink message, the position controller is implemented to calculate desired attitude angles as outputs. The output angles are first transformed into corresponding Pulse Width Modulation (PWM) values and sent to transmitter using a Universal Serial Bus (USB) to Pulse Position Modulation (PPM) generator called PCTx. The transmitter sends desired attitude angle information to the flight controller on the quadrotor. Then the quadrotor responds and tracks the desire attitude angles using its internal attitude controller.

In this thesis, only the position controller is studied. The attitude controller used is an Open source controller that is a part of the Ardupilot firmware uploaded in the flight controller and not be coded. The whole system will be introduced in detail in separate parts in following chapters.

Chapter 3.

Quadrotor Setup

Since quadrotors are becoming increasingly popular nowadays, there are plenty of choices in the market. Quadrotors produced by DJI and Parrot are the most common civil UAVs, which are widely used in aerial photography and races with their robust flight performance

However, these quadrotors lack feasibility of assembling additional devices because of their fixed frame, limited load capacity, and lack of the support of an easy interface between quadrotor and computer. Thus, a self-assembled quadrotor is selected with a large frame which is capable of carrying extra devices.

3.1 Flight controller

Flight controller is the control system of the quadrotor. It is used to maintain the stability and provide controllability of the flight. As a small onboard computer, flight controller contains a processor and sensors including gyro sensor, accelerometer, magnetometer, barometer and GPS module. By processing data measured from sensors and radio commands, it gives corresponding speed signals to brushless motors to reach desired the attitude angles.

3.1.1 APM 2.6 introduction

In this thesis, the flight controller used is APM 2.6 (Ardupilot Mega 2.6). APM is an autopilot system based on Arduino Mega platform. APM can be used to control different types of aerial vehicle, including traditional helicopter, fixed-wing plane and multi-rotor helicopters. The software for APM is open source and is maintained and updated by a professional development team and supported by a large community.

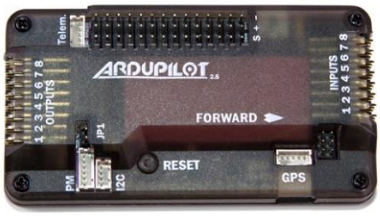


Figure 3-1 APM 2.6 Flight Controller

3.1.2 APM 2.6 setup

Before uploading software to APM physical board, GCS software “Mission Planner” should be downloaded and installed on the ground station computer first. Mission planner is a ground station of Ardupilot flight controller which is compatible with Windows system. Mission Planner has many useful features:

1. Firmware can be loaded into the flight controller board via Mission Planner.
2. Parameters of the controller can be tuned to optimize the flight performance.
3. Multiple functions and extra devices such as camera gimbal can be setup and tuned.
4. Flight logs created by flight controller can be downloaded and analyzed after flight to find reasons of flight problems.
5. When mission planner is connected to flight controller with telemetry, flight status can be monitored real time on the screen and the information can also be saved as telemetry logs for post-flight processing.

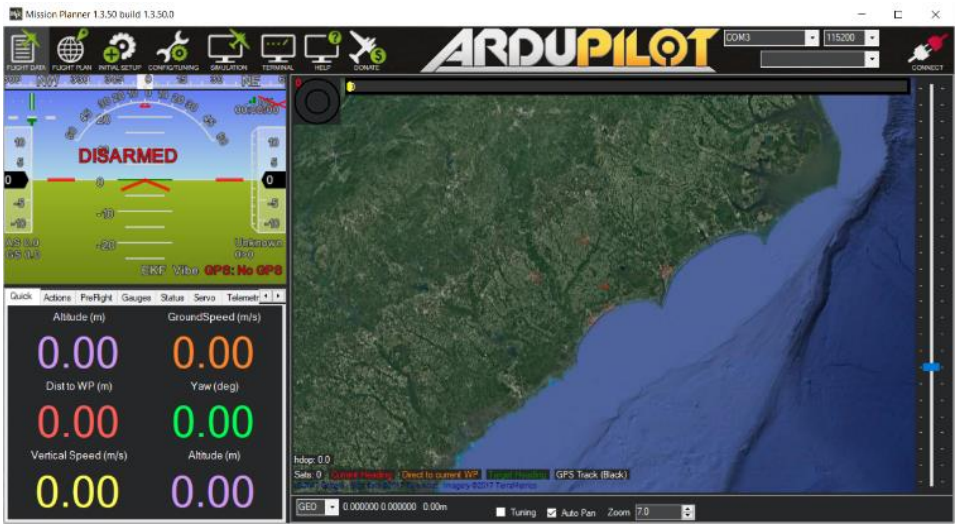


Figure 3-2 Mission Planner Screen

After Mission Planner is installed, the APM controller should be connected to computer using USB cable. Specific port should be selected for the board and Baud Rate is set to 115200. On the initial setup screen, Quad frame, which is shown in Figure 3-3 and labeled in the red square, should be selected, and firmware is then uploaded into APM board following the instructions of the software.

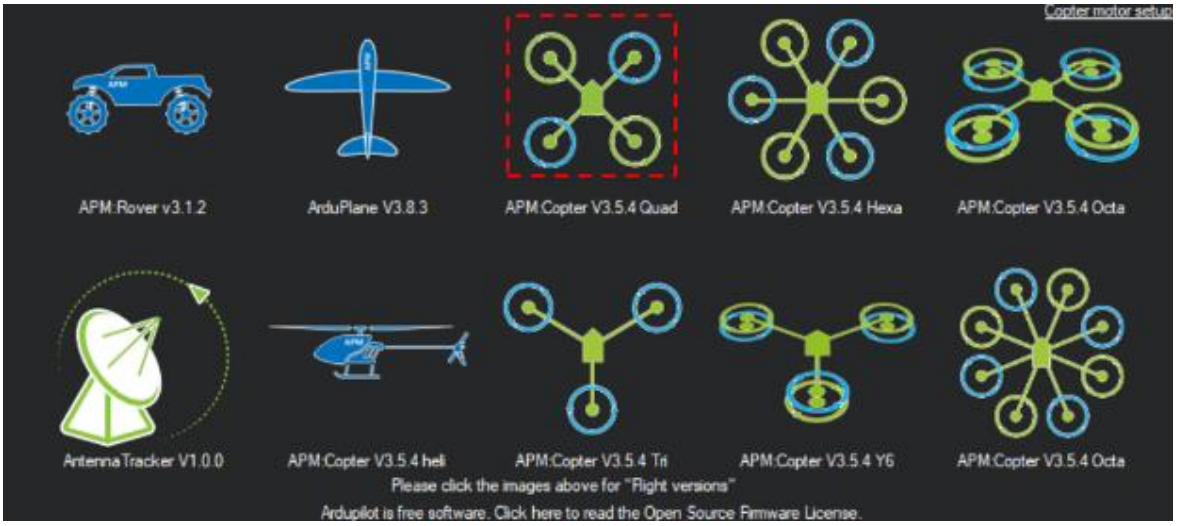


Figure 3-3 Frame selection in Mission Planner

After the firmware is uploaded successfully, parameters can be tuned in Mission Planner for an optimal performance. The basic control parameters for the quadrotor used in this thesis are shown in Figure 3-4. For different quadrotor frames, these parameters should be adjusted differently.

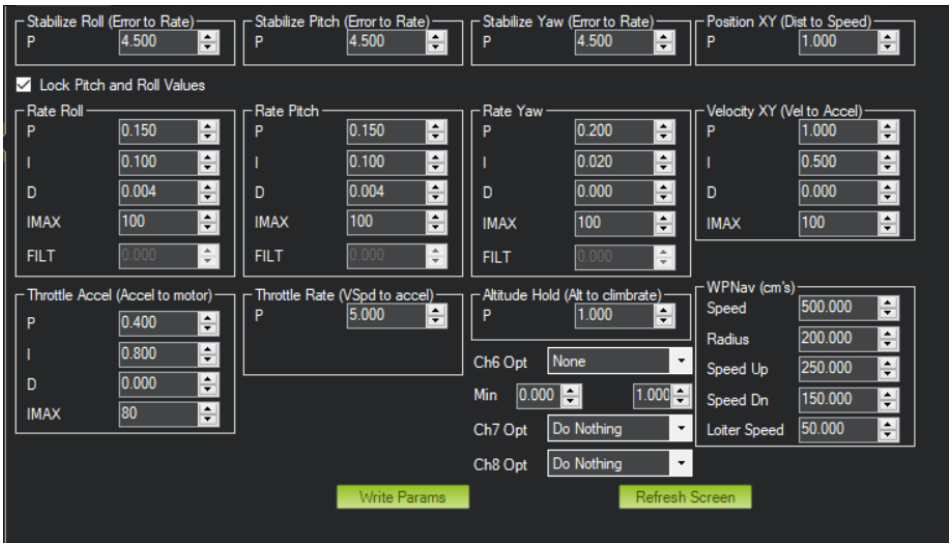


Figure 3-4 APM parameter setting

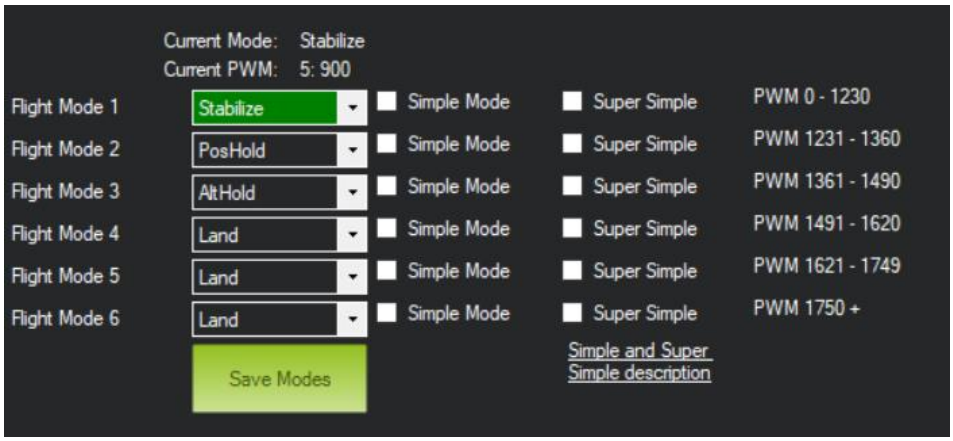


Figure 3-5 APM Flight mode setting

There are up to 6 flight modes which can be chosen for one APM autopilot. The setup is shown in Figure 3-5 above. Flight modes can be switched with the value change of channel 5 during flight.

“Stabilize” mode is a basic flight mode which does not require GPS lock. The pilot’s pitch and roll inputs directly control the quadrotor lean angles. When the roll and pitch sticks are released, the quadrotor can automatically level itself. The yaw input controls the change rate of quadrotor heading. If yaw stick is released, the quadrotor can keep its current heading. The throttle input controls the total thrust force. Under “stabilize” mode, the altitude is not maintained automatically, so throttle should be continuously adjusted during the flight to keep an altitude to avoid crashing on the ground.

In “Althold” (Altitude hold) mode, the altitude is automatically controlled by the flight controller to maintain current altitude when throttle is in the neutral position. If the throttle is above or below neutral position, the quadrotor will climb or descend at a pre-set speed, which can also be adjusted in Mission Planner. The control of pitch, roll and yaw is the same with stabilize mode. “Althold” mode does not need GPS lock.

“Poshold” (position hold) mode is similar to “Althold” mode in pitch and roll control, but it requires GPS lock. Under “Poshold” mode, the altitude can also be maintained automatically, and the flight controller can calculate angle compensation for wind disturbance. So, when the pitch and roll sticks are released, the quadrotor can maintain its current location. The hovering performance, however, depends on how good the GPS signal is. The lean angles are now the sum of angle transferred from stick input and angle compensation. If the sticks are releases

when quadrotor is moving, the quadrotor will brake and stop quickly.

"Land" mode enables the quadrotor to move straight down and land steadily, which can avoid hard crash when landing manually. This mode is also GPS independent.

In the tracking test, quadrotor takes off in "Stabilize" mode. Automatic tracking tasks only run under "Althold" mode. "Poshold" mode is only used when quadrotor is controlled to move to a desired location manually or emergency occurs for an immediate brake.

3.1.3 Parameter tuning

The flight performance varies with different controller parameters and by adjusting these parameters on Mission Planner, the performance can be optimized.

For parameters in rate pitch and roll, the most important parameter 'P' determines how large the desired rotation rate converted from angle error should be. Higher P can make the quadrotor more responsive and lower P smoother. If P is too large, the quadrotor may oscillate and if too small, the quadrotor will be sluggish.

In rate yaw, the P values works similarly. If P is too large, the quadrotor heading may oscillate and if too small, it is likely that the heading is hard to maintain because of small torque in yaw direction.

In altitude tuning, the P value in altitude hold controls the climb or descent rate converted from altitude error. Large P can make the quadcopter more aggressive to keep the altitude and may be jerky if it is set too large. P value in throttle rate controls climb and descent rate when quadrotor is controlled to move up or down.

The throttle acceleration parameters convert acceleration error to motor outputs. The ratio of P and I is required to maintain $1/2$ when these parameters being modified. These parameters are not recommended to be increased unless the quadrotor are powerful copters and these parameters should both be reduced by 50%.

3.2 Radio setup

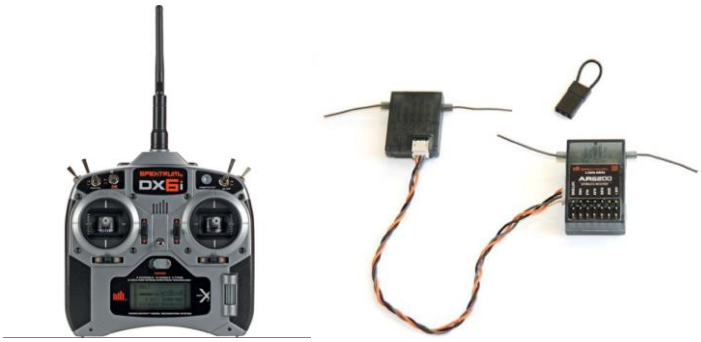


Figure 3-6 Radio transmitter and receiver

The radio system, which is consist of a transmitter and receiver, transmits motion commands to flight controller. The transmitter and receiver used in this thesis are DX6i and AR6200 produced by Spektrum. DX6i is a 6-channel and 10-model memory full range DSM2 2.4GHz radio transmitter for airplanes and helicopters. It is a popular transmitter which is widely used in unmanned aerial vehicle fly and race. AR6200 is a 6-channel DSM2 2.4GHz radio receiver which is compatible with DX6i.

These two devices should be bound first, so the receiver only recognizes the bounded transmitter. At least five radio channels should be used in quadrotor control. Channel 1 to 5 correspond to roll, pitch, throttle, yaw and flight modes. Generally, except channel 5, channel 1 to 4 do not need to be specifically set up. The setting of channel 5 is shown in Figure 3-7 below. In “MIX” option in Adjust menu of transmitter, the value of channel can be mixed with other channels. In this case, Gyro corresponds to channel 5 and it is mixed with itself. Thus, with different “ELE D/R”, “FLAP” and “AIL D/R” switch combinations, channel 5 varies and flight mode can be switched. The relationship between switch combination and flight mode is shown in Table 3-1.

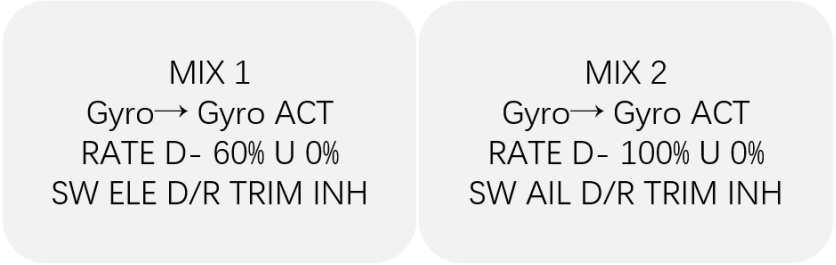


Figure 3-7 Channel 5 setting

ELE D/R	FLAP	AIL D/R	Flight mode
0	0	0	Stabilize
1	0	0	Poshold
1	1	1	Althold
1	1	1	Landing

Table 3-1 Switch combination and flight mode relationship

3.3 ESC

ESC (Electronic Speed Controller) is used to control the brushless motor speed according to input signal. The ESC used in this thesis is “Rctimer OPTO-30a ESC” and its specification are shown in Table 3-2 below.

Figure 3-8 OPTP 5K-30A ESC

Input voltage (V)	DC 6-16.8
Current (A)	30
Size (mm*mm*mm)	36*26*7
weight (g)	32

Table 3-2 ESC specification

The power and ground wires are connected to a power distribution board which is used to split the battery connectors into four, so that one battery can be connected to four ESCs. The signal wire is connected to the APM controller output and receives the speed request. ESC outputs are three DC currents which can drive the brushless motors.

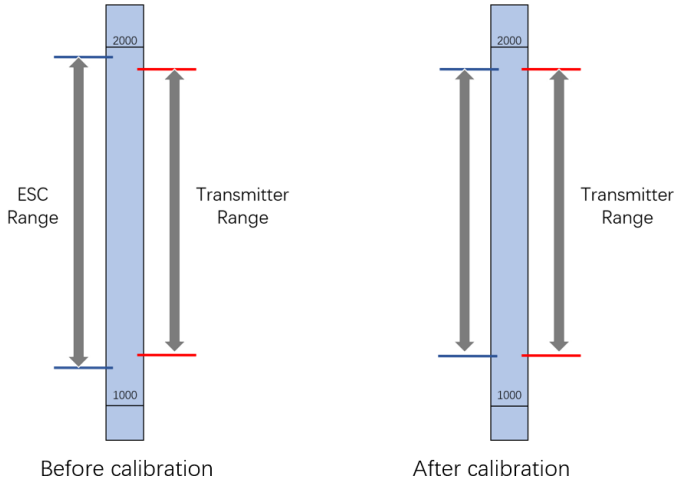


Figure 3-9 ESC range before and after calibration

ESC calibration is essential before flight. Without calibration, ESC does not have knowledge of the maximum and minimum PWM values of the input signal, and the limits and ranges of ESCs and transmitter do not match. Thus, even for a same speed request, four motors can spin in different speeds, which will cause severe imbalance of the quadrotor. After calibration, each ESC will have the same standard and their ranges and limits are consistent.

3.4 Brushless motor

The model of brushless motor used is 5010-360kv Rctimer brushless motor and its specifications are shown below in Table 3-3. KV number is the most important parameter for a brushless motor. 360kv means the rotation speed of the motor increases 360 rpm when the input voltage increases 1 volt with no load applied on the motor.

Figure 3-10 5010-360kv brushless motor

Motor diameter (mm)	50
Motor thickness (mm)	10
Weight (g)	90
KV (rpm/v)	360

Table 3-3 Motor specification

3.5 Propeller

The quadrotor is designed to carry one extra camera, a video transmission system and two Li-Po batteries. So large lift force is required to guarantee the quadrotor can take off and has enough force left for flight maneuvers. Propeller 1755 is selected in this thesis. When the quadrotor is powered by a 4S Li-Po battery, the lift force provided by 1755 propeller and 360KV motor is shown in Table 3-5 below.

Figure 3-11 1755 Propeller

Material	Carbon Fiber
Length (inch)	17
Pitch (inch)	5.5
Weight (g)	24

Table 3-4 1755 Propeller specification I

	No load		On load			Load type
Voltage (V)	Current (A)	Speed(rpm)	Current (A)	Pull (g)	Power (W)	Propeller
14.8	0.3	5328	1.8	400	26.6	1755
			4.7	800	69.6	
			8.5	1170	125.8	

Table 3-5 1755 Propeller specification II

3.6 Telemetry module

Telemetry module is used to build a wireless connection with flight controller and Mission Planner. Through telemetry it is possible to receive and monitor flight status on computer during flight. The telemetry module used is YKS 3DR Radio Telemetry Kit which is compatible with APM 2.6. The frequency band is 915MHz. When connecting using telemetry, baud rate should be set to 57600 instead of 115200. The connection should be built before APM board is armed. If the board is already armed, it is possible that wireless connection will be rejected.

Figure 3-12 Telemetry Kit

3.7 Video transmission module

Video transmitter is used to capture camera images and transmit video signals to computer. AKK TS832 and RC 832 FPV Audio Video transmitter and Receiver, and TOTMC USB 2.0 Video Capture Adapter are used. The transmitter is mounted on the quadrotor and connected to

the camera signal wire. The receiver is placed on the ground and connected to video capture adaptor, which is connected to computer via USB.

The video signal is first collected by the transmitter and then sent to RC832 Receiver wirelessly at 5.8 GHz working frequency. Then the video signal is collected by the computer through the video capture adaptor. The video resolution supported is 720 * 480 pixels at 30 frames per second (fps). With the image transmission module, the camera works as a remote webcam of the computer. The image can be shown on the screen easily by creating a class *VideoCapture()* in C++. The lag of image transmission can be neglected, so the visual detection can be considered as being processed in real time.

There are three wireless transmission used in flight, the radio transmitter, telemetry and video transmitter. They should have different working frequencies to avoid being affected by each other. In this thesis, working frequencies for these three systems are 2.4 GHz, 915 MHz and 5.8 GHz respectively.

Figure 3-13 Video transmitter, receiver and USB adaptor

3.8 Camera module

A camera is mounted on the quadrotor to capture the images below and detect targets for local navigation. In this thesis, a BitEye Mini FPV camera is selected. This camera uses Sony ICX639/638BK CCD image sensor with 700 TVL horizontal resolutions and 2.88 mm lens, and works under 5~25V DC, which can easily be powered by 2~6S Li-Po Batteries.

Figure 3-14 Bit Eye camera

There are two types of image sensors widely used nowadays, CCD (charge-coupled device) and CMOS (Complementary metal–oxide–semiconductor). These two image sensors are similar in many aspects but have an important difference in collecting and storing light signals for every pixel. CCD camera usually uses a global shutter. In global shutter mode, all pixels are exposed simultaneously. So, if the object is moving or changing its shape quickly, CCD camera is more likely to capture all details of the object in every frame. CMOS camera usually uses rolling shutter. In rolling shutter mode, the pixels are exposed one row by one row. Thus, some parts of the moving object may be exposed repeatedly and loses its true shape. The example of two shutters is shown in Figure 3-15. In the left figure, the rotating fan loses its own shape in rolling shutter. But in the right figure, the fan keeps its own shape in the captured image in global shutter.

In the tracking task, both the target and quadrotor are moving. And the camera also shakes in a high frequency with small amplitude, which is caused by motor vibration. So, CCD camera is recommended in this thesis, which can be more helpful in keeping target's real shape and making the detection more robust.

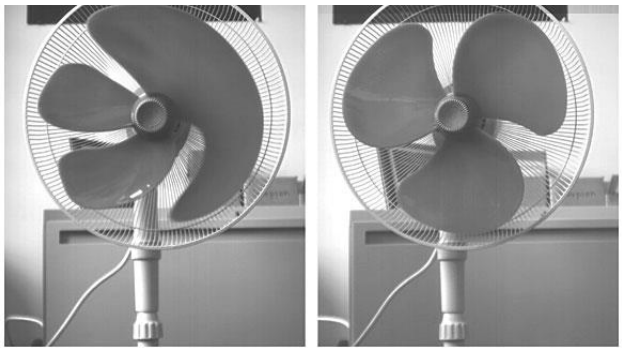


Figure 3-15 Comparison between rolling and global shutter [19]

3.9 Gimbal setup

APM supports several optional hardware including camera gimbal. The gimbal can stabilize the camera up to three axis directions by connecting to three output channels. In this thesis, the camera is expected to point downward and keep the same heading with the quadrotor body, so only tilt (pitch) and roll direction is used. Pan (yaw) direction is not activated.

Flight performance can be affected by the weight of the quadrotor. It is recommended that the quadrotor can still hover around 50% throttle and not exceeding 70% after the camera gimbal is mounted on the frame. Otherwise, it may cause quadrotor wobble during flight and has less power left to keep the attitude under disturbance. Thus, the camera gimbal consists of a 3D-printed frame and two SG-50 servo motors, which are less than 200g and have negligible impact on flight. The printed gimbal is shown in Figure 3-16.

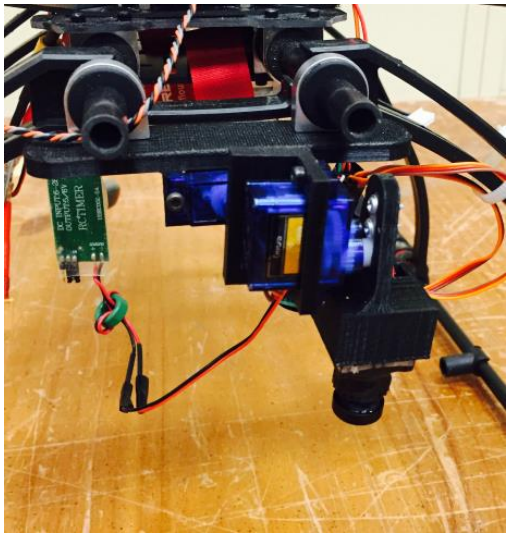


Figure 3-16 3D-printed Gimbal

For APM board, the signal wires of pitch and roll servo motor are connected to pin A11 and A10 as shown in Figure 3-17. The power and ground wires of two servo motors cannot be connected to the board directly. If two servo motors are powered by the board, APM may be brownout when motor moves during the flight. Thus, these two servo motors should be powered by external BECs (Battery Elimination Circuit) or regulators.

Figure 3-17 Camera gimbal connection [20]

The gimbal parameter setting is shown in Figure 3-18 below. By clicking “stabilize tilt” and “stabilize roll” option, the gimbal can have a real-time angle compensation. When the quadrotor has pitch or roll rotation, the corresponding servo motor rotates same degree in opposite direction. Thus, the camera direction is maintained constant and always points downwards. The minimum and maximum values of servo limits which correspond to angle limits varies between different motors and should be well calibrated to make sure the camera is perfectly pointing straight down. If the motor rotates in an opposite direction, it can be reversed by clicking “Reverse” option.

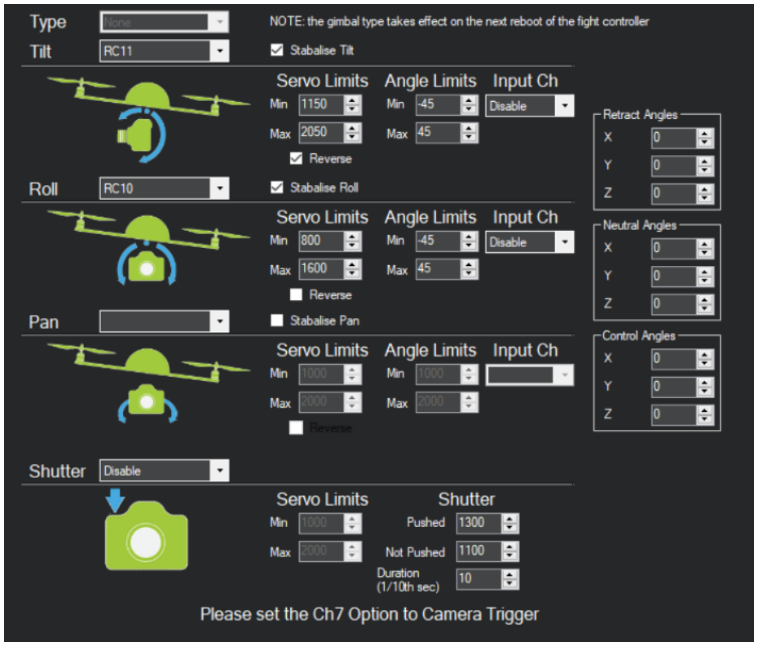


Figure 3-18 Camera gimbal setup

3.10 Battery

Li-Po (lithium polymer) battery is the most common power supply for quadrotors. Two Li-Po batteries are used in this thesis. One 4S (16.8 V) battery is used to power APM board, servo motors on camera gimbal and four ESCs. Another 3S (11.1 V) battery is used to power camera and video transmitter. The advantage of using two separate batteries is to decrease current disturbance caused by ESCs on video transmission system and provide an image signal as stable and clear as possible.

Battery is the main weight source of the whole quadrotor. These two batteries have 700g weight altogether. To decrease the effect of additional moment caused by battery, it is recommended to mount batteries close to the frame geometric center. With a fully charged 4S battery, quadrotor can fly around 20 minutes. Additional battery can be used to extend flight duration, but it also increases load and decreases the ability to recover attitude from wind disturbance.

Figure 3-19 Li-Po battery

3.11 Connection

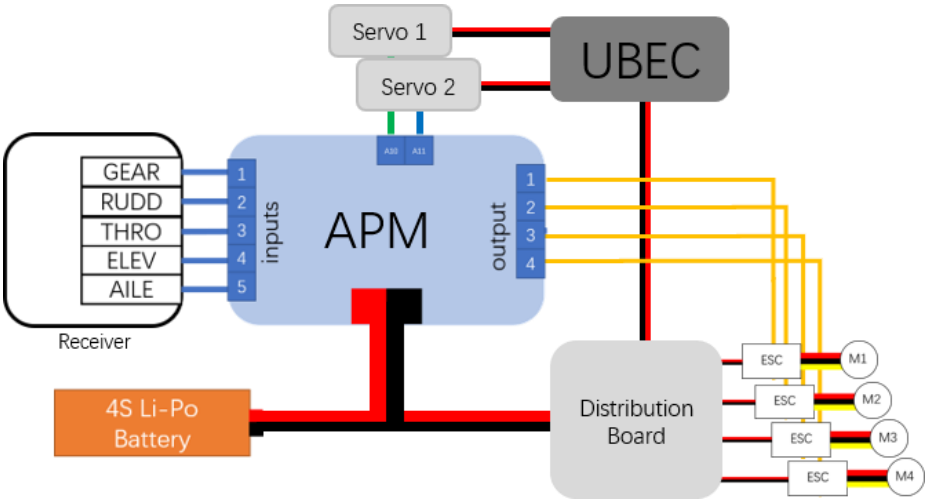


Figure 3-20 APM connection

Connection of APM autopilot is shown in Figure 3-20. A 4S Li-Po battery is connected to a power module for APM and split into two connectors. One connector is a 6-pos connector which provides 5.3V to power the flight controller board and provides current and voltage measurements. The other connector is connected to power distribution board and split into five sub-connectors. Four of these connectors are used to power ESCs separately, and one is connected to a UBEC (Universal Battery Elimination Circuit) to provide a constant 5V to power two servo motors. The signal wires should be connected to APM as shown in Figure 3-20.

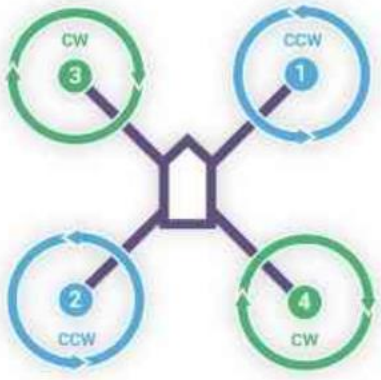


Figure 3-21 Motor connection for QUAD frame [21]

The motor connection of a Quad frame is shown in Figure 3-21. The rotation direction of motors and propellers should strictly follow the Figure 3-21, otherwise the propellers can cause wrong torques and the quadrotor will be in out of control and result in a hard crash.

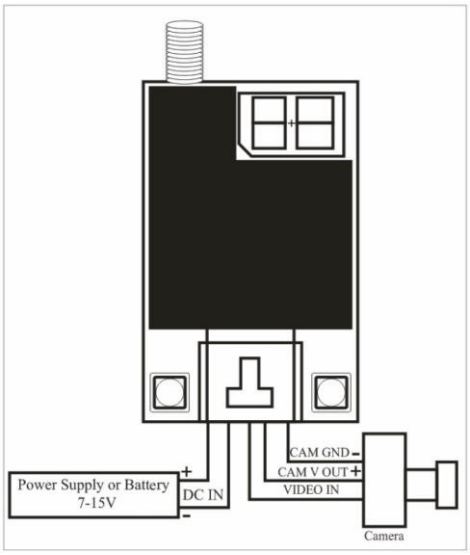


Figure 3-22 Connection of video transmission module

The connection of video transmission module is shown in Figure 3-22 above. An individual 3S battery powers the video transmitter and camera. These devices are totally isolated from the previous circuit system, so the noise from ESCs and motors on image signals will be reduced effectively. The assembled quadrotor is shown in Figure 3-23.

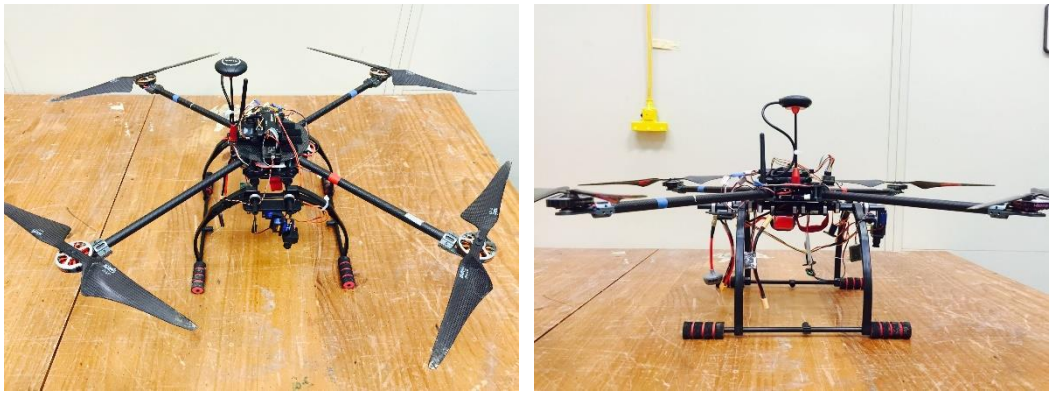


Figure 3-23 The assembled quadrotor

Chapter 4.

Target Detection

Without GPS navigation, the relative position between quadrotor and target can only rely on the target visual detection in the camera image. Images of the surface underneath the quadrotor are captured and transmitted to the computer. The target can be detected after performing a series of image processing operations, and its pixel location can be used to calculate relative position and velocity. The relative position and velocity are the inputs to the controller that in turn determines the values of desired attitude angles.

During the visual detection process, there are two requirements. First, the detection should be accurate enough. Only intended target should be detected, not other objects, which guarantees that the quadrotor moves in the right direction. Second, the image processing should be fast, which requires that the computation time should be minimized. As a result, the frequency of the output signal can be high enough for the quadrotor to respond in time to follow the changing distance of the quadrotor and target. The target selected in this thesis was selected specially to meet these requirements, viz can be easily detected in the image and not easily misidentified with false positives. The image processing utilizes both color-based and contour-based detection algorithm. In this chapter, Section 4.1 and Section 4.2 deal with detection strategy and target selection. Furthermore, from Section 4.3 to Section 4.5, camera calibration, 3D reconstruction and gimbal usage are discussed respectively to improve the detection accuracy in both software and hardware aspects.

4.1 Detection strategy

The visual detection and tracking code was written and compiled in Visual Studio C++ and OpenCV (Open Source Computer Vision) is used. OpenCV is a powerful library of programming functions in computer vision. After the image is received by the computer, it is stored as a RGB color model as a $720 \times 480 \times 3$ matrix. 720 and 480 are the width and height of the image and the third dimension "3" represents the red, green and blue channels

independently. The RGB color model is a color model that adds red, green and blue light together with different weight to produce different colors. Each pixel has three channels with values from 0 to 255.

At the very beginning of the study of target detection, color-based detection by detecting a specific color such as red or blue block was first tested. The target was placed on a grass field outside and fully exposed under bright sunshine during the initial flight test. The light reflection on the surface on the target and ground can cause an obvious chromatic aberration in the camera, so the object is likely to show a different color. For example, the RC car which is used to drag the target is shown in Figure 4-1 below. The camera captures its bright red outer case under indoor light. However, when it is exposed to outdoor light, it completely loses its original color. After the color filter is applied on the image, the colorful block may not be filtered out entirely but can only remain as an irregular part. To avoid this potential problem, the target which only contains black and white parts is selected, because black (0,0,0 in RGB model) and white (255,255,255 in RBG model) are two colors that have the largest difference in RBG model and can be still distinguished even in harsh light environments. The color-based detection is not enough to find the specific target. Contour detection is also used to recognize the patterns on the target to eliminate noise and determine the target location.

Figure 4-1 Chromatic aberration example

4.1.1 Gray scale

The image is turned into gray scale first. Gray scale is an image where each pixel only contains a single value which represents the intensity information of the light amount. The transformation equation is

$$Y = 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B \qquad 4\text{-}1$$

where Y is the pixel gray scale value and R, G and B are the red, green and blue value in GRB color mode. The gray scale image is stored as a 720×480 matrix. The width and height are not changed but the third dimension is reduced to one.

4.1.2 Binary image

After the grayscale image is obtained, it is turned into a binary image. The binary image is a digital image where each pixel only contains 1 or 0, which means the image only has two colors, white for 1 and black for 0. The binary image can be easily calculated from a gray scale image by setting a threshold:

$$dst(x,y) = \begin{cases} maxVal & if\ src(x,y) > threshold \\ 0 & otherwise \end{cases} \quad 4\text{-}2$$

where $src(x,y)$ is the input pixel value and $dst(x,y)$ is the output pixel value in (x,y) location. If the intensity of the pixel is greater than the threshold, the new value is reset to be the maximum value. Otherwise, the value is 0. In this case, $maxVal$ is 1 and the threshold can be set to a high value, for example 240 is selected in this thesis.

As mentioned in Section 4.1, the target only contains white and black parts. With the selection of high threshold, the binary image transformation can be considered as a strong white color filter. As the result, only white part of the target and some small noises remain in the binary image. The small noises are caused by some light reflective spots on the grass field and salt-and-pepper noise in video transmission. The black part of the target and the remaining field the environment such as the grass field will all be turned into 0 (black) in the binary image.

4.1.3 Median filter

Salt-and-pepper noise is a common noise occurring in video transmission. It appears as white or black pixels in random position, maybe black pixels in light parts or white pixels in dark parts, or both. It is usually caused by a sudden disturbance in image signals. Salt-and-pepper noises exist as small white points in binary images like other reflective spots in the background. Each point creates extra contours and takes additional loops to determine whether it is the target or not. These noises should be removed before the next step to reduce the number of detection

loops and decrease programming processing time.

Median filter is a nonlinear digital filter which is powerful in removing salt-and-pepper noise and smoothing the image. Median filter computes the median value of the pixel under the kernel window and replace the central pixel with the median value, which is highly effective in removing salt and pepper noise. In OpenCV, the corresponding function is

$$\text{void } medianBlur(InputArray\ src, OutputArray\ dst, int\ ksize).$$

The input array is a binary image, and $ksize$ is the aperture linear size, which must be odd and greater than 1. The input image will be smoothed by using this median filter function with $ksize \times ksize$ aperture. The image example of salt-and-pepper noise is shown in Figure 4-2. The left image is the original image which shows the salt-and-pepper noise are these white points in dark part and black point in light parts. The right image shows the result after the median filter is applied with median filter. As a result, almost all noises have been removed.

Figure 4-2 Salt-and-pepper noise and median filter example [22]

4.1.4 Edge detection

After two median filters are applied, only white parts of the target and the noise remain in the binary image. Edge detection is used in this step to find the edges of the image. The function *Canny* is called in visual studio.

$$Void\ Canny(InputArray\ image, Output\ edges, double\ threshold1, double\ threshold2,$$
$$int\ apertureSize = 3, bool\ L2gradient = false)$$

The input image is the binary image after two median filters. The output image is the edge image with the same size. $Threshold1$ and $threshold2$ are lower and upper threshold for the

hysteresis procedure. $ApertureSize$ is the aperture size of Sobel operator, which is set to be the default value of 3. $L2gradient$ decides the method of calculation of gradient, the default value *false* is enough in this thesis.

Canny edge detector is a famous and widely used edge detection method. The steps of the algorithm are shown below.

1. A gaussian filter is applied to filter out noise first.

$$K = \frac{1}{159}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix} \quad 4\text{-}3$$

K is the gaussian filter matrix with size of 5.

2. Intensity gradient of every pixel is calculated. Two directions of Sobel Operator are applied on each pixel.

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} and\ G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

where G_x is the gradient strength in x direction and G_y is in y direction. Total gradient strength can be calculated in two ways. When $L2gradient$ is set to the default value false, the gradient strength is the L1 norm.

$$G = |Gx| + |Gy| \quad 4\text{-}4$$

When the $L2gradient$ is set to true, a more accurate method is used by using L2 norm.

$$G = \sqrt{(Gx)^2 + (Gy)^2} \quad 4\text{-}5$$

The direction is calculated by

$$\theta = arctan\left(\frac{Gy}{Gx}\right) \quad 4\text{-}6$$

and then rounded to 0, 45, 90 and 135.

3. Non-maximum suppression is applied to thinning the edge. The gradient of pixels around edges may all have large values, which makes the edge quite blurred. In non-maximum suppression, the pixel gradient is compared with the neighbor pixels in the gradient direction. If it is not the local maxima, it will be set to zero. Thus, only one pixel

in this line remains and can be regarded as the candidate edge while other unwanted lines are removed.

4. Two thresholds are used to filter out the edges.

   1. If the pixel gradient is greater than the upper threshold, it is considered as the edge.

   2. If the pixel gradient is smaller than the lower threshold, it is eliminated.

   3. If the pixel is between upper and lower threshold, it is considered as edge only when it is connected to a pixel considered as edge.

4.1.5 Contour detection

After the edge image is obtained, another function $findcontours$ in OpenCV is called.

$$Void\ findcontours(InputOutputArray\ \boldsymbol{image}, OutputArrayOf\ Arrays\ \boldsymbol{contours}, OutputArray\ \boldsymbol{hierarchy}, int\ \boldsymbol{mode}, int\ \boldsymbol{method}, Point\ \boldsymbol{offset}\ =\ Point()).$$

The input of the function is the edge image calculated in last step and there are two outputs. Each element of contours is the array containing all pixel locations of the detected contours. The output array hierarchy contains topology information of the contours. For ith contour, $hierarchy[i][0]$, $hierarchy[i][1]$, $hierarchy[i][2]$ and $hierarchy[i][3]$ are indexes of next contour and previous contour at the same hierarchy level, and first child contour at lower hierarchy and parent contour at higher hierarchy correspondingly. Hierarchy represents the relationship between a contour and other contours outside, inside or next to it.

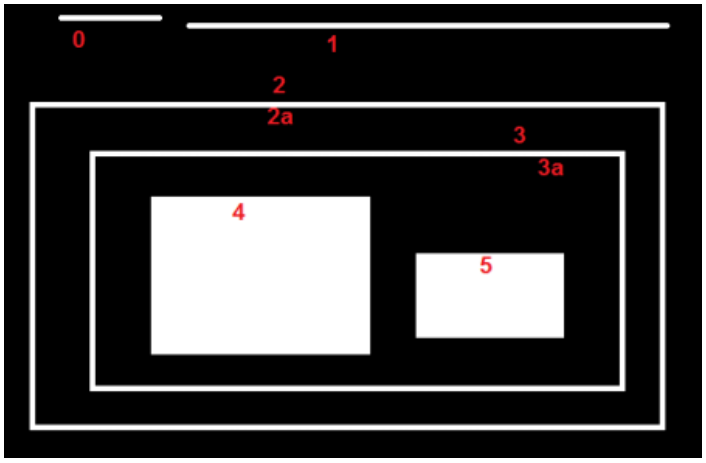


Figure 4-3 Hierarchy example [23]

The hierarchy example is shown in Figure 4-3. Contour 0, contour 1 and contour 2 are the most external contours in the image, which are considered in the same hierarchy level. Contour 2a is the child contour of contour 2 because it is enclosed completely by contour 2 and contour 2 is the parent contour of contour 2a. Similarly, contour 3 is the child contour of contour 2a and contour 3a is the child contour of contour 3. Contour 4 and 5 are in the same hierarchy level and are both the child contours of contour 3a. Contour 4 and 5 are the next and the previous contour for each other. In hierarchy, if there are no contours having corresponding relationship, -1 is stored. Otherwise, contour index is stored in the element.

The parameter mode should be set to be RETR_TREE so that the full hierarchy for all contours is built. And method should be set to CV_CHAIN_APPROX_NONE. All contour points are stored. Any two subsequent pixel locations have maximum distance of 1 in horizontal or vertical directions.

After all contour points are stored and their hierarchy relationship is calculated, the target can be detected by distinguishing its special pattern.

4.2 Target selection

The target should have two properties, particularity and distinguishability. Two targets with special patterns are selected in this thesis.

4.2.1 QR code

QR code (Quick Response Code) is a type of matrix barcode which contains information about the corresponding item and is widely used in automotive industry today. During the test, the QR code is displayed in the grass field. Compared with the background, QR code is unique enough that it can be hardly mixed up with other objects or patterns.

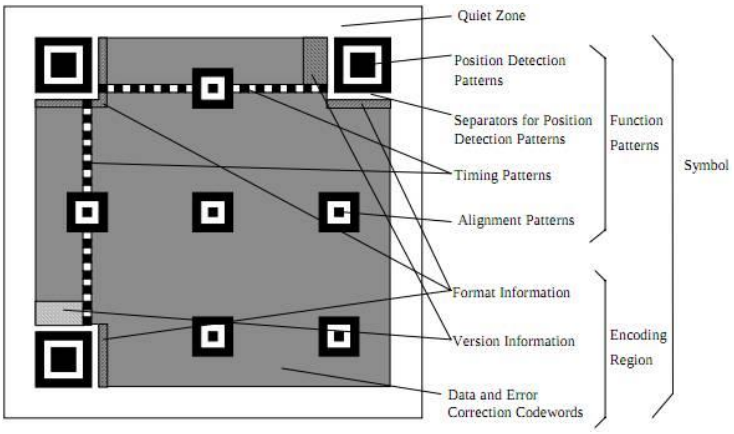


Figure 4-4 QR code structure [24]

The structure of a QR code is shown in Figure 4-4. Three position detection patterns consist of three repeated squares are located at the left-upper, right-upper and left-lower corners of the QR code. Other parts are not used in detection. The center of QR code is located by finding these patterns first. There are several methods to detect position pattern such as by calculating the area ratios or side length ratios of three squares. In this thesis, the position pattern is detected by finding repeated contours. The flow diagram is shown in Figure 4-5 below.

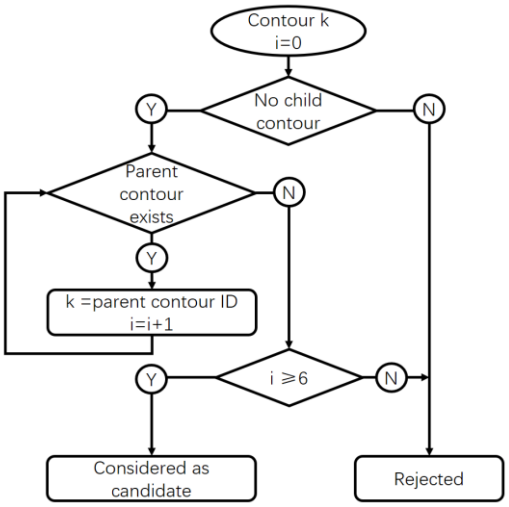


Figure 4-5 Contour detection flow diagram

In the edge image, each position pattern creates three edges. Each edge has two contours, one inner contour and one outer contour. Thus, each position pattern results in six repeated concentric contours. If a contour has at least five repeated parent contours and the centers of these contours are all located together, it is considered as the most inner contour of one position pattern. Its mass center is also the pattern's center.

In the detection loop, every contour is scanned one at a time. If it does not have enough pixels, it will be considered as a noise contour. Otherwise, its child contour will be checked next. If it has no child contour, which means it is in the most inner hierarchy level, then its parent contour will be checked. Otherwise, it is rejected directly because there should be no other patterns in the most inner square of the position pattern. If the parent contour exists, the loop will continue to find its parent contour and repeat the process until no parent contour in higher hierarchy level exists and a counter records the number of loops. If it is greater than 6, which means the contour has at least 5 repeated parent contours, and it is determined to be the position detection patterns. After scanning all contours, the centers of three position pattern can be calculated by calculating the contour mass center. First the spatial and central moments can be calculated using function

Moments moments(InputArray array, bool binaryImage = false).

Input array is the contour points.

The output spatial moments are calculated by

$$m_{ji} = \sum_{x,y} array(x,y) \cdot x^j \cdot y^i \quad 4\text{-}7$$

The central moments are calculated by

$$mu_{ji} = \sum_{x,y} array(x,y) \cdot (x - \bar{x})^j \cdot (y - \bar{y})^i \quad 4\text{-}8$$

and center mass is

$$\overline{x} = \frac{m_{10}}{m_{00}} \quad \text{and} \quad \overline{y} = \frac{m_{01}}{m_{00}} \quad 4\text{-}9$$

The three center points are recorded as $A(x_A, y_A)$, $B(x_B, y_B)$ and $C(x_C, y_C)$.

Then the distances of these three points are calculated.

$$L_{AB} = \sqrt{(x_A - x_B)^2 + (y_A - y_B)^2} \quad 4\text{-}10$$

$$L_{AC} = \sqrt{(x_A - x_C)^2 + (y_A - y_C)^2} \quad 4\text{-}11$$

$$L_{BC} = \sqrt{(x_B - x_C)^2 + (y_B - y_C)^2} \quad 4\text{-}12$$

The center of the QR code is the midpoint of the longest distance. For example, L_{AB} is longest and center pixel location is $(\frac{(x_A - x_B)}{2}, \frac{(y_A - y_B)}{2})$.

Thus, the position of the QR code is finally determined. However, during the test, when the

quadrotor reaches an altitude higher than 5 meters, the size of the whole QR code is small and the position patterns are even smaller because its area is less than one ninth of the whole QR code. The black squares in the most center position may only occupy several pixels and the width of the white outliers may only occupy one pixel. The FPV camera cannot provide a very high image quality because of the salt-and-pepper noise and its low resolution. As a result, these three detection zones sometimes cannot be recognized simultaneously. Thus, QR code is not used any longer and another target was developed to solve this problem.

4.2.2 Ring

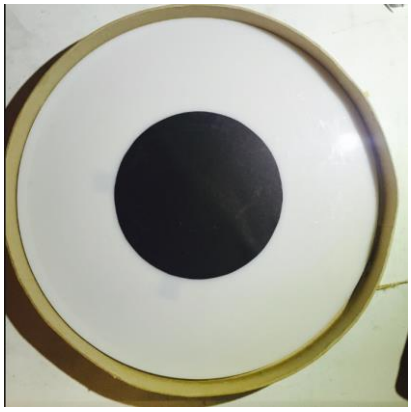


Figure 4-6 Target of ring shape

In a QR code, only the position pattern is useful in target detection. Thus, the target can be simplified and only a ring is used as shown in Figure 4-6. The advantage of using a single ring is that it can be made into the same size of a whole QR code, which guarantees that all parts can occupy enough pixels to keep the details of the target clear enough to be recognized when the quadrotor reaches a high altitude.

The white part of the ring is not made of cardboard but a piece of translucent diffuser material to reduce the chromatic aberration as much as possible. The diffuser material can spread out lights to create soft light. Thus, the pixels of the white ring can reach almost maximum value in grayscale under sun light in an outdoor environment and can be easily filtered out by setting a high threshold. The black circle in the center is less prone to reflective light and remained completely. However, compared with QR code, only a ring may be not unique enough. Thus, an extra criterion is made to eliminate other objects which may have the similar repeated

contours. The basic algorithm calculates the distances between centers of each circle. The centers of the target should be highly concentric. If several candidate targets are found, the most concentric ring is selected as the target.

The detection process example is shown in figures below.

Figure 4-7 Original and gray scale image example

The target is placed on a chair in the laboratory. The indoor light is not strong enough, so the target is illuminated by LED placed behind. In Figure 4-7, left figure shows the origin image in RGB mode and left is the gray scale image, where the white ring has a very high light intensity in gray scale.

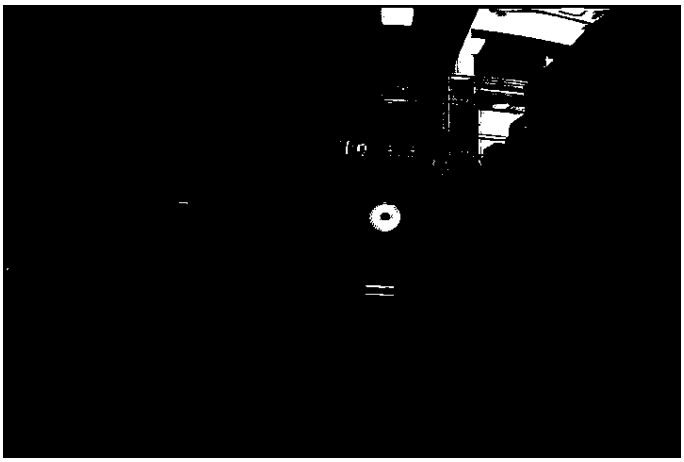


Figure 4-8 Binary image example

Then the gray scale image is turned into the binary image as shown in Figure 4-8. The white ring is preserved almost completely. Besides the target, there are other objects left including light reflective parts and some noise.

Figure 4-9 Image example after two median filters

In next step, median filter is applied twice. Most noises are removed, and the ring part becomes smoother.

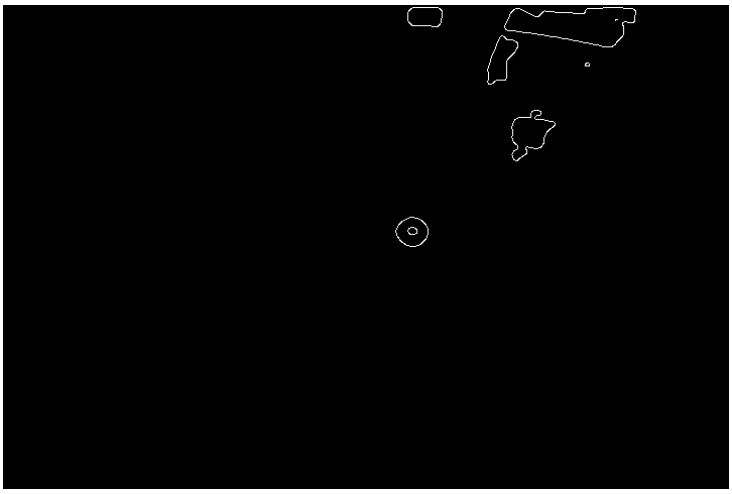


Figure 4-10 Edge image example

Edge detection is applied in Figure 4-10. Two concentric circles are shown in the image, and there are only several distractor edges remaining which will be rejected by the detection algorithm.

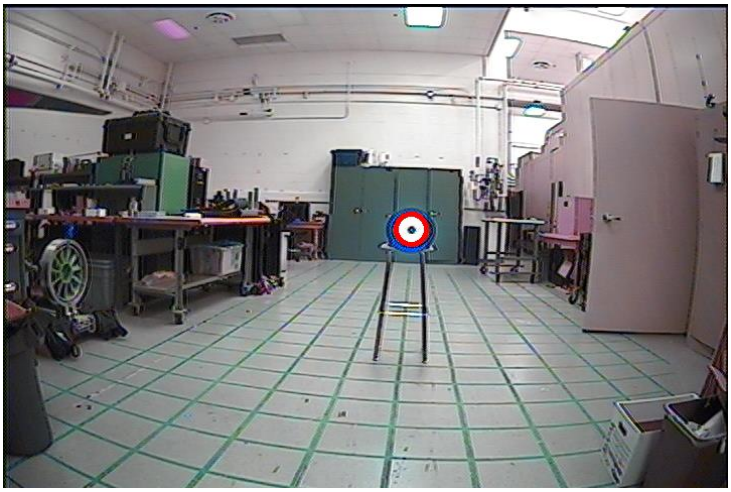


Figure 4-11 Target detection example

As shown in Figure 4-11, the target is found and labeled as a red circle in the image.

4.3 Camera calibration

After the position of the target is determined, position correction from the distortion should be done to make the position more accurate before being sent to the controller. The FPV camera has a common problem like all other pinhole cameras, the distortion, which is caused by unideal pinhole projection. Barrel and pincushion are two common types of distortion which are shown in Figure 4-12.

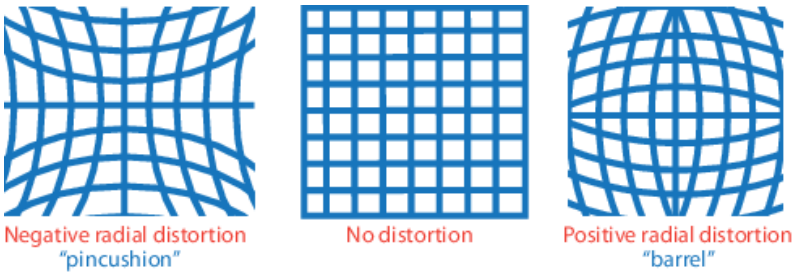


Figure 4-12 Distortion types

Due to distortion, the position of the target obtained may not be accurate enough, which may result in a bias error of the desired attitude angles. To eliminate this error, camera parameters should be calibrated. Camera calibration is used to estimate the parameters of lens and image sensor of a camera. With these parameters, the distortion can be corrected, and the distance can be calculated in terms of real world units.

For camera calibration, a checkerboard is used. The checkerboard pattern can provide accurate intersections of lines and help to determine calibration parameters. The focal length in horizontal and vertical directions, f_x and f_y, and the optical center (principal center), c_x and c_y, and radial distortion coefficients, k_1 and k_2, will be calculated after camera calibration. The calibration parameters are shown in Table 4-1 below.

k_1	radial distortion coefficient	-0.3575
k_2		0.13
p_1	Tangential distortion coefficient	0
p_2		0
f_x	focal length in x axis	484.2883
f_y	focal length in y axis	435.8663
c_x	optical length in x axis	362.61
c_y	optical length in y axis	240.2852
t_x	Tangential distortion in x axis	0
t_y	Tangential distortion in y axis	0

Table 4-1 Calibration parameter

The distortion model is shown below.

$$x_{distorted} = x(1 + k_1 \cdot r^2 + k_2 \cdot r^4 + k_3 \cdot r^6) \quad 4\text{-}13$$

$$y_{distorted} = y(1 + k_1 \cdot r^2 + k_2 \cdot r^4 + k_3 \cdot r^6) \quad 4\text{-}14$$

where $(x_{distorted}, y_{distorted})$ are the distorted points, x and y are the undistorted pixel locations in normalized image coordinate. The normalized image coordinate is calculated from the original coordinate by translating the origin to the optical center and then dividing by the focal length.

To calculate corrected locations, the bivariate quartic equations should be solved. Using OpenCV function

$$void\ undistortPoints(InputArray\ src, OutputArray\ dst, InputArray\ cameraMatrix, inputArray\ distCoeffs, InputArray\ R = noArray(), inputArray\ P = noArray())$$

The input is the pixel location in distorted image.

$$camera\ matrix = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \text{ and } distCoeffs = (k_1, k_2, p_1, p_2)$$

R and P are just empty. The output is the normalized point set.

Suppose the output is (u', v') and (x', y') is its real pixel location.

$$x' = f_x \cdot u' + c_x \quad 4\text{-}15$$

$$y' = f_y \cdot v' + c_y \quad 4\text{-}16$$

(x', y') is the point in original coordinate whose origin is the left-upper corner of the image. A coordinate transformation is required to move origin to the image center for convenience.

$$\tilde{x} = x' - c_x \quad 4\text{-}17$$

$$\tilde{y} = y' - c_y \quad 4\text{-}18$$

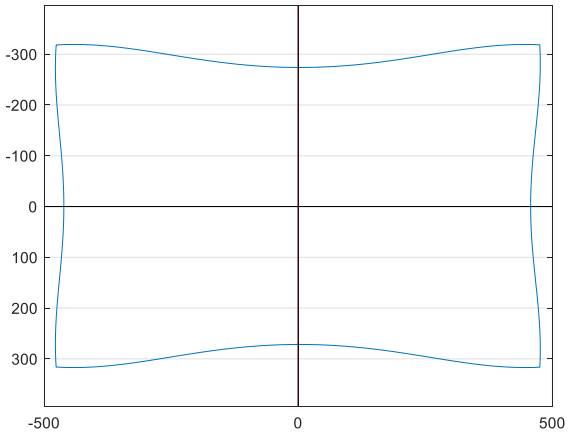


Figure 4-13 Final pixel coordinate

The new coordinate is shown in Figure 4-13 above. The blue frame is the image boundary undistorted and origin is in the image center. Thus, the sign of pixel location directly shows the position relationship between the quadrotor and the target.

After the undistorted pixel position of the target is calculated, Kalman filter can be

implemented for position estimation. As shown in the left figure in Figure 4-14, the red curve is the original pixel position of the target in x direction and blue curve is the estimated position based on Kalman filter. The pixel distance between the original target and estimated target is shown in the right figure.

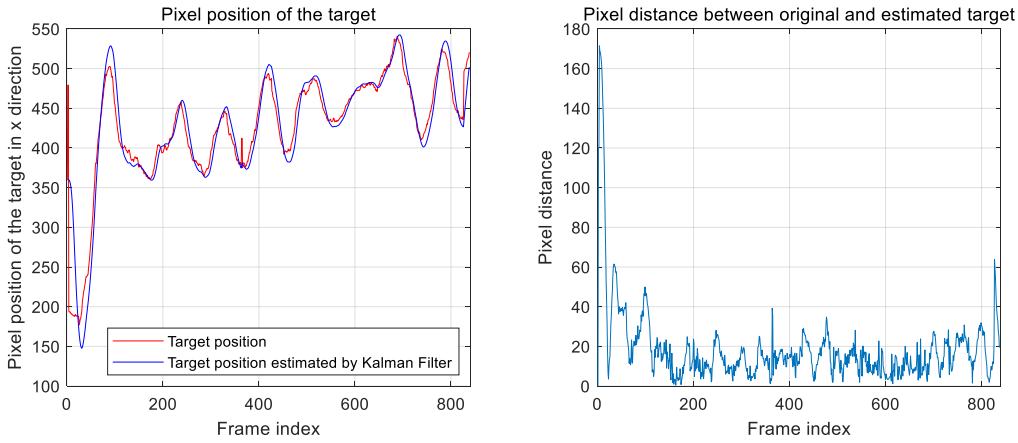


Figure 4-14 The original and estimated pixel position of the target

4.4 3D reconstruction

The pixel location of the target can be directly used as the controller input. However, the altitude of the quadrotor may change slightly during the flight and the real length of a pixel changes. By reconstructing the pixel to real world coordinate, a more precise estimate of the position error can be obtained. The relationship between image and world coordinate is shown below.

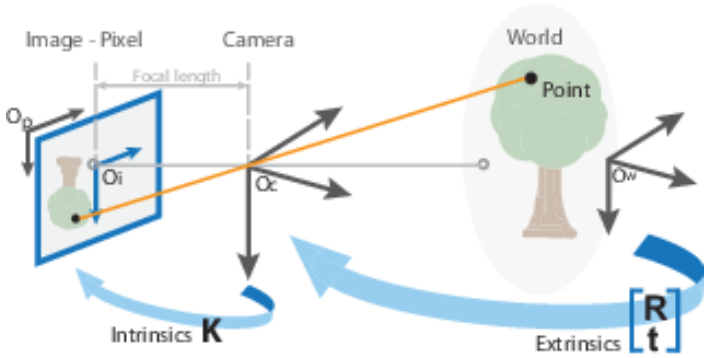


Figure 4-15 Relationship between image can world coordinate

$$w[x'\ y'\ 1] = [X\ Y\ Z\ 1]P \tag{4-19}$$

$$P = \begin{pmatrix} R \\ t \end{pmatrix} K \tag{4-20}$$

$$Z = H \tag{4-21}$$

where w is scale factor, x' and y' are image points. X, Y and Z are world points and Z is equal to the altitude H. P and t are rotation and translation extrinsic matrix. There is no rotation and translation of FPV camera, thus they are set to be identity matrix and zero vector. K is the camera matrix. Then the relative position in the real word coordinate can be calculated,

$$X = \frac{H \cdot \tilde{x}}{f_x} \tag{4-22}$$

$$Y = \frac{H \cdot \tilde{y}}{f_y} \tag{4-23}$$

4.5 Camera Gimbal

In this test, a pitch-and-roll camera gimbal is mounted on the quadrotor. The gimbal gives pitch and roll angle compensation in real time and enables the camera always to point downwards. If the gimbal is not mounted, the position of the target can still be calculated but

in a more complex method which is introduced below.

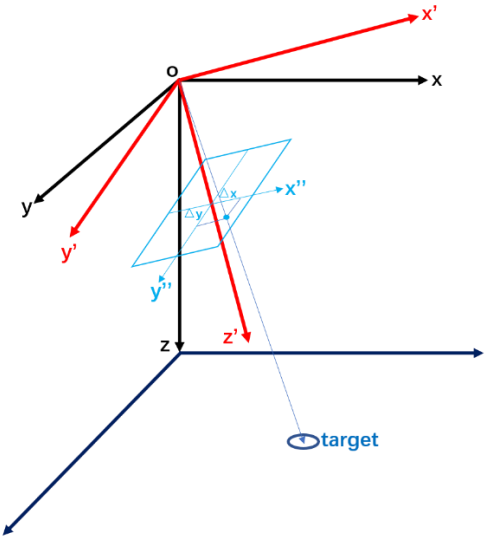


Figure 4-16 Coordinates for position calculation without camera gimbal

As shown in Figure 4-16, the black coordinate is the inertial coordinate, the red coordinate is the body coordinate of the quadrotor frame with pitch θ and roll φ, and light blue coordinate is the camera plane. The relationship between inertial coordinate and frame coordinate can be built by two rotation matrices:

$$R_\theta = \begin{bmatrix} cos\theta & 0 & sin\theta \\ 0 & 1 & 0 \\ -sin\theta & 0 & cos\theta \end{bmatrix} \text{ and } R_\varphi = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\varphi & -sin\varphi \\ 0 & sin\varphi & cos\varphi \end{bmatrix}$$

And then $P = R_\theta \cdot R_\varphi \cdot Q$ where P is the point in body coordinate and Q is the point in inertial coordinate. Suppose the pixel location of the projection of the target on camera plane is denoted as $(\Delta x, \Delta y)$. Then the direction vector from the origin to target in body coordinate can be written as $v_b = (\frac{\Delta x}{f_x}, \frac{\Delta y}{f_y}, 1)$. The direction vector in inertial coordinate can be calculated by $v_i = (R_\theta \cdot R_\varphi)^{-1} \cdot v_b = (\Delta x', \Delta y', \Delta z')$. Suppose the target location in inertial coordinate is denoted as (X, Y, H) and its direction vector can be written as $V_i = (X, Y, H)$. The two direction vectors v_i and V_i are parallel and their cross product should be zero.

$$V_i \times v_i = 0 \qquad 4\text{-}24$$

By simplifying Equation (4-24), the position of target can be calculated from

$$\frac{X}{\Delta x'} = \frac{Y}{\Delta y'} = \frac{H}{\Delta z'} \qquad 4\text{-}25$$

The pitch, roll and altitude can be received by the computer via MAVlink, then the target position can be calculated. However, this method is still not recommended for two disadvantages. First, this method results in extra computation time. Second, it can reduce effective detection range of the camera.

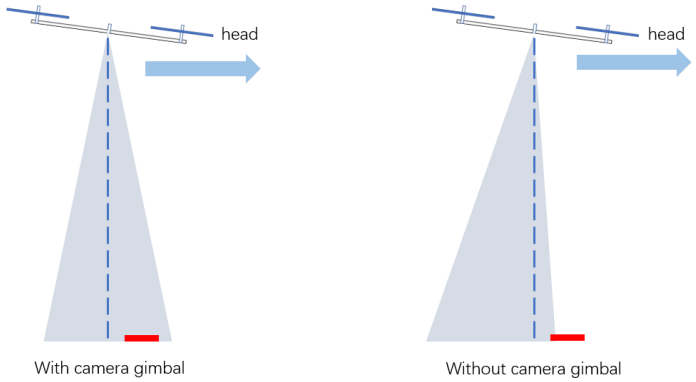


Figure 4-17 Camera range with and without gimbal

As shown in Figure 4-17 above, assume the target is in front of the quadrotor. The quadrotor needs to move forward to track the target, which requires a negative pitch. The negative pitch will result in the backward movement of the projection of the focal center of the camera. Thus, the camera can only see less in the front view, which can increase the possibility of losing the target.

Chapter 5.

MAVlink

Mission Planner can achieve real-time communication between computer and flight controller by MAVlink protocol. However, it does not provide communication channel with other software applications. Thus, it is necessary to build a communication channel in Visual Studio to receive real-time flight information.

MAVlink (Micro Air Vehicle Link) is a protocol for communication with unmanned vehicle and is used by PX4, Pixhawk, APM and other flight controllers. The basic unit of a message is a frame. The structure is shown in Figure 5-1 and Table 5-1 below.

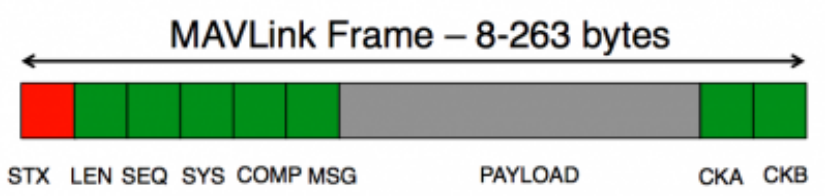


Figure 5-1 MAVlink structure [25]

Byte Index	Content	Value	Explanation
0[STX]	Packet start sign	v1.0: 0xFE (v0.9: 0x55)	The start of a new packet.
1[LEN]	Payload length	0 - 255	Length of the following payload.
2[SEQ]	Packet sequence	0 - 255	Counting send sequence, used to detect packet loss
3[SYS]	System ID	1 - 255	ID of the SENDING system. Allows to differentiate different MAVs on the same network.
4[SOMP]	Component ID	0 - 255	ID of the SENDING component. Allows to differentiate different components of the same system.
5[MSG]	Message ID	0 - 255	ID of the message - the ID defines what the payload “means” and how it should be correctly decoded.
6 to (n+6) [PAYLOAD]	Data	(0 - 255) bytes	Data of the message, depends on the message id.

Table 5-1 MAVlink structure [26]

Two most important parts of a frame is message ID and payload. The former indicates what content this frame belongs to and the later contains detail information. MAVlink supports over 200 contents, but not all of them are used by APM. In Table 5-2, the messages used by APM are listed.

ID	message
0	heartbeat
1	system status
2	system time
24	GPS raw data
27	IMU raw data
29	pressure
30	attitude
33	global position
35	RC channel
36	servo output
42	current
62	fixed wing navigation
74	HUD
152	memory
163	AHRS
165	battery information

Table 5-2 message ID for APM

Using the source code provided by MAVlink in GitHub, a communication channel in Visual Studio can be built. For each message frame, its ID should be first identified. Then the detail information can be decoded by using functions defined in corresponding header files.

Chapter 6.

Command transmission

After input attitude angles are determined through the controllers, they should be sent to the flight controller in real time. MAVlink protocol enables the realization of this function by switching the flight controller into onboard mode and sending desired attitude angles in MAVlink message package through telemetry directly. But under this mode, the flight controller can be armed without transmitter and receiver, which means the radio part is totally disabled. It is hard to take reaction such as shutting down the motors at once if the flight controller loses control or immediate maneuvers should be taken. But it can be easily realized when the flight controller is controlled by the transmitter, such as shutting down the motors by pressing throttle cut button, switching to “land” mode or triggering failsafe. For safety reasons, it is best to use a transmitter to control the flight controller. Then a communication between transmitter and computer is required to build. A USB-to-PPM generator called PCTx is used in interface of computer and transmitter.

6.1 Working principle

The output signal of each radio channel is a PWM signal. PWM (Pulse Width Modulation) is a modulation system for encoding a message into a pulsing signal. For a common PWM signal, the period is 20ms and the lasting time of high voltage (800μs to 2000μs), which is the pulse width, will be transferred into corresponding throttle, attitude angle and flight mode.

Channel	Dead zone (μs)	Max(μs)	Min(μs)	Trim(μs)
RC 1	30	1937	1109	1523
RC 2	30	1936	1109	1523
RC 3	30	1937	1109	1507
RC 4	40	1937	1110	1523
RC5	30	1803	986	/

Table 6-1 Radio channel setting

The detail parameters of radio channels are shown in Table 6-1 above. The parameters may differ in different transmitters and change slightly after radio calibration.

During objecting tracking task, channel 5 should be kept constant in the range of altitude flight mode, and throttle and yaw should be kept in the trim (neutral) value, which means the current altitude and yaw are maintained. These three channels are constant during the tracking. Channel 1 and 2, the roll and pitch angles, are the only two variables should be controlled. The values of these two input channels are transferred into desired pitch and roll linearly in following equation.

$$desired\ angle = \begin{cases} \dfrac{rc - (trim + deadzone)}{rc\_max - (trim + deadzone)} \times angle\_\max & if\ rc > (trim + deadzone) \\ 0 & if\ (trim - deadzone) \le rc \le (trim - deadzone) \\ \dfrac{rc - (trim - deadzone)}{rc\_min - (trim + deadzone)} \times angle\_\min & if\ rc < (trim - deadzone) \end{cases} \quad 6\text{-}1$$

where rc is the input signal pulse width, $angle\_max$ and $angle\_min$ are the maximum and minimum lean angles in all fight modes, which are set to be 10°. These two values can be tuned in parameter setting via Mission Planner.

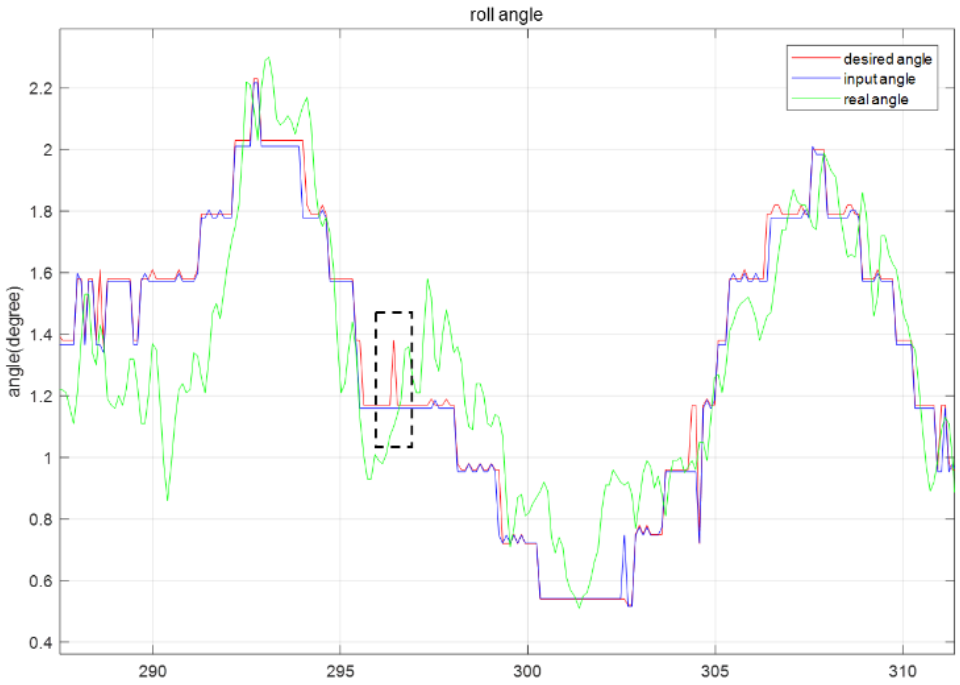


Figure 6-1 Log of desired, input and real angle

Three angles recorded in flight log are plotted in Figure 6-1. The blue curve is the input angle corresponding to rc value and red curve is desired angle calculated by flight controller. Green curve is the real angle measured by IMU. The desired angel and input angle almost overlap, which can prove the effectiveness of this transmission method. There are some points not matching marked in the dashed black rectangle. This is caused by the data loss when logging. The desired angle is recorded but rc value is missed. But the following angle rise in real angle shows the flight did receive the command to increase roll angle and responded.

One important function of DX6i is its master-slave function. One DX6i can be operated as a slave transmitter when its trainer cord is connected to another master transmitter. When the trainer switch is held, the master transmitter sends a single PPM signal which contains all information of its own PMW channels through trainer cord to slave transmitter. PPM (Pulse Position Modulation) is another form of signal modulation. Unlike PWM signal, it can carry the information of several channels in a single signal but not in several signals separately. PPM and PWM are interconvertible and the relationship is show in Figure 6-2 below.

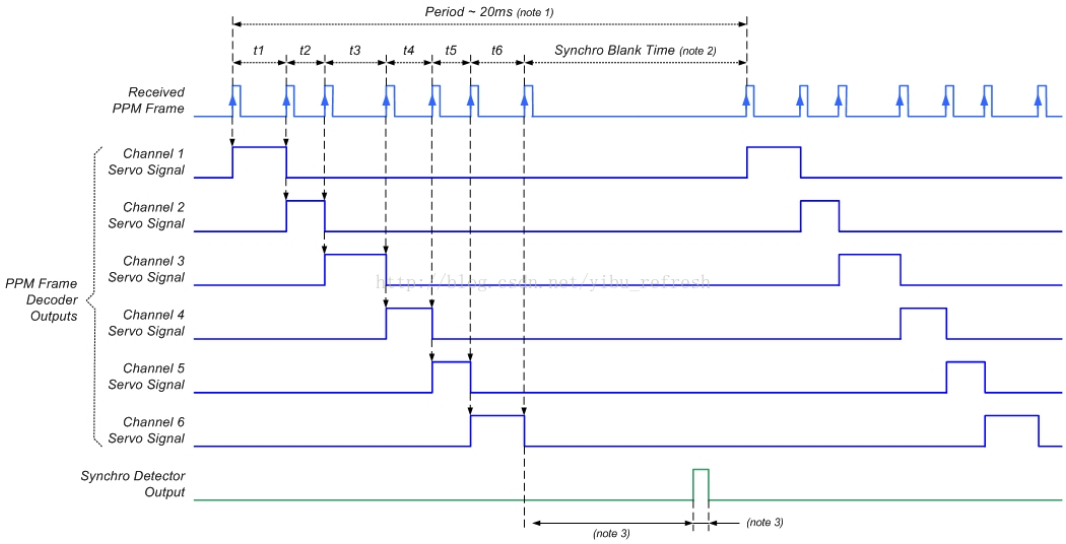


Figure 6-2 PWM and PPM signals [27]

After receiving this PPM signal, the slave transmitter only generates and sends the same output PWM signals of the master transmitter, but not itself. By using this function, a USB to PPM generator can be used as a “fake” master transmitter, and then the transmitter output signals can be totally controlled by computer.

6.2 PCTx

PCTx is a Spektrum transmitter to PC interface produced by Endurance RC. Essentially, it is a USB to PPM generator. It receives input message via USB port on PC and generate up to 9 output channels, and its output is a single PPM signal with 50Hz. Each channel has 1024 steps, which is linearly corresponding to PWM value of the transmitter. The curve fitting equation is shown below.

$$pctx = \frac{rc-1078.1}{0.8096} \quad 6\text{-}2$$

where $pctx$ is the PCTx input value (from 0 to 1024). The parameters may be slightly different with different transmitters.

With the source code provided by Endurance RC, interface between computer and PCTx is built in Visual Studio using C++. Then the whole signal transmission system is complete. After desired pitch and roll are calculated, the angle values are first transferred into corresponding PCTx value and sent to PCTx via USB port. Then PCTx generates PPM output signal which contains all received information. Then PPM signal is received by DX6i transmitter through trainer cord and transferred into corresponding PWM signals for each channel. And then the transmitter sends these signals to the receiver of the flight controller. The flight transfers PWM value into desired angles at last.

During the tracking task, the trainer switch is released at first and the quadrotor is manually controlled to take off. When the target is shown in the screen, the trainer switch is held and then the quadrotor is automatically controlled by the computer. If an unexpected situation arises, trainer switch can be released at once and the quadrotor is controlled manually again for emergency landing or crash avoidance, which meets the safety requirement.

Chapter 7.

Modeling

A quadrotor is an underactuated system with a lower number of actuators than degrees of freedom. The inputs of the quadrotor are only four rotating speeds of brushless motors which produce thrust forces, but the quadrotor body can have translation and rotation motion in six dimensions. The modeling of a quadrotor is studied in this chapter. Section 7.1 defines two frames. Section 7.2 deals with force of motors and Section 7.3 deals with the torque. The quadrotor model is studied in Section 7.4.

7.1 Frame definition

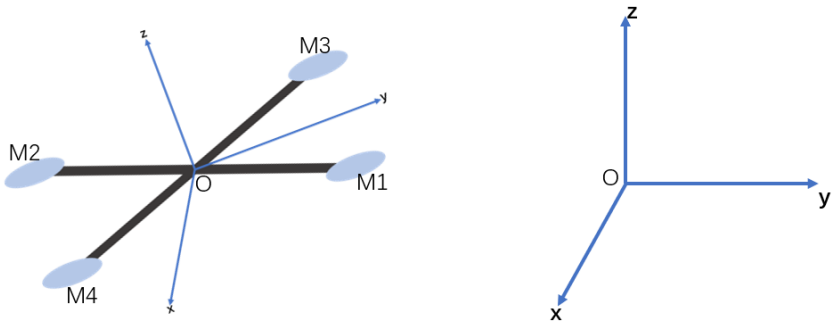


Figure 7-1 Body frame and inertial frame

Two frames are defined in dynamic modeling. The body frame is defined on the quadrotor. The x and y directions are pointing right and forward respectively. The z direction points upward from the body. The inertial frame is defined on the ground. The z direction points opposite to the direction of the gravity.

7.2 Forces

The torque of a brushless motor can be written as

$$\tau = K_t \cdot (I - I_0) \qquad 7\text{-}1$$

where τ is the motor torque, I is the input current and I_0 is the input current with no load on

the motor. During the flight, the current on the motor is much larger than when it is no load, thus, $I \gg I_0$. Therefore, equation (7-1) can be simplified to

$$\tau = K_t \cdot I \tag{7-2}$$

The voltage on the motor can be represented as:

$$V = I \cdot R_m + K_v \cdot \omega \tag{7-3}$$

where V is the voltage across motor, R_m is the internal resistance of the motor, K_v is a constant and ω is the motor rotation speed. The inertial resistance is small and can be neglected, thus, $R_m \approx 0$. Equation (7-3) can then be simplified to:

$$V = K_v \cdot \omega \tag{7-4}$$

Using equations (7-2) and (7-4), the power equation can be obtained:

$$P = V \cdot I = \frac{K_v}{K_t} \cdot \tau \cdot \omega \tag{7-5}$$

where P is the motor power.

According to the law of conservation of energy, the electric energy exerted by the motor is equal to the work done to overcome air friction when motor is rotating with the propeller attached. The power can then be written as:

$$P = T \cdot v_h \tag{7-6}$$

where T is thrust force and v_h is the air velocity.

The relationship between thrust force and air velocity is given by

$$v_h = \sqrt{\frac{T}{2\rho A}} \tag{7-7}$$

where ρ is the density of air and A is the frontal area of the propeller [28].

The motor torque is proportional to thrust force:

$$\tau = K_\tau \cdot T \tag{7-8}$$

where K_τ is a proportional coefficient.

Therfore, the thrust force equation can be obtained by

$$\mathrm{T} = \left(2\rho A \frac{{K_v}^2 {K_\tau}^2}{{K_t}^2}\right) \cdot \omega^2 = k \cdot \omega^2 \tag{7-9}$$

k is a constant for convention and $k = \left(2\rho A \frac{{K_v}^2 {K_\tau}^2}{{K_t}^2}\right)$.

There are four brushless motors mounted on the quadrotor. So, the total thrust force is the sum of the four thrust forces produced by each motors:

$$T_t = \sum_{i=1}^{4} T_i = k \begin{bmatrix} 0 \\ 0 \\ \sum_{i=1}^{4} {\omega_i}^2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ T_z \end{bmatrix} \quad 7\text{-}10$$

where T_t is total thrust force, and i denotes the motor order.

7.3 Torque

When the propeller is rotating, the surrounding air creates drag force which is given by

$$F_D = \frac{1}{2}\rho \cdot C_D \cdot A \cdot v^2 \quad 7\text{-}11$$

where F_D is the drag force, ρ is the air density, C_D is the drag coefficient related to the Reynolds number, A is the reference area and v is the flow velocity relative to the propeller [28].

Assuming the surrounding air velocity is zero, the propeller velocity is equal to the relative flow velocity. Therefore, $v = \omega \cdot R$, where R is the propeller radius.

The torque caused by the drag force is given by

$$\tau_D = R \cdot F_D = \frac{1}{2}\rho \cdot C_D \cdot A \cdot R^3 \cdot \omega^2 = b_d \cdot \omega^2 \quad 7\text{-}12$$

where b is the drag coefficient for convention and $b = \frac{1}{2}\rho \cdot C_D \cdot A \cdot R^3$.

The total torque for one motor about z-axis is given by

$$\tau_z = \tau_D + I_M \cdot \dot{\omega} \quad 7\text{-}13$$

where I_M is the moment of inertial of a brushless motor and propeller in z-axis and $\dot{\omega}$ is the angular acceleration. Assuming a steady state flight condition, the angular acceleration is close to zero. Therefore, $\dot{\omega} \approx 0$. Equation (7-13) can be simplified to:

$$\tau_{zi} = b_d \cdot {\omega_i}^2 \quad 7\text{-}14$$

where i denotes the motor order.

For an X-Quad frame, the body axis is not located on the arms. Thus, complete torques about three axes are related to all motors.

$$\tau_\varphi = \frac{\sqrt{2}}{2} Lk({\omega_1}^2 + {\omega_3}^2 - {\omega_2}^2 - {\omega_4}^2) \quad 7\text{-}15$$

$$\tau_\theta = \frac{\sqrt{2}}{2} Lk({\omega_2}^2 + {\omega_3}^2 - {\omega_1}^2 - {\omega_4}^2) \quad 7\text{-}16$$

$$\tau_\psi = b({\omega_1}^2 + {\omega_2}^2 - {\omega_3}^2 - {\omega_4}^2) \quad 7\text{-}17$$

where τ_φ, τ_θ and τ_ψ are torques about x, y and z axis respectively and L is the arm length of the quadrotor.

The total torques are written as

$$\tau_B = \begin{bmatrix} \tau_\varphi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} \quad 7\text{-}18$$

7.4 Model

The position of the quadrotor mass center is denoted by $X = (x, y, z)^T$ in inertial frame and the attitude angle is expressed as $\Theta = (\varphi, \theta, \psi)^T$, where $\varphi, \theta \; and \; \psi$ are the roll, pitch and yaw angle in inertial frame respectively. Ω denotes the angular velocity in the body frame along the axis of rotation. Ω is a vector along the axis and not equal to $\dot{\Theta}$, which is the first time derivate of the attitude angle. The relationship between these two parameters [29] is:

$$\Omega = \mathrm{W} \cdot \dot{\Theta}$$

$$\text{where } W = \begin{bmatrix} -\sin(\theta) & 0 & 1 \\ \cos(\theta)\sin(\varphi) & \cos(\varphi) & 0 \\ \cos(\theta)\cos(\varphi) & -\sin(\varphi) & 0 \end{bmatrix} \quad 7\text{-}19$$

The body frame and inertial frame can be related by using a rotation matrix, which is defined as

$$R = \begin{bmatrix} c\theta \cdot c\psi & s\varphi \cdot s\theta \cdot c\psi - c\psi \cdot s\psi & c\varphi \cdot s\theta \cdot c\psi + s\varphi \cdot s\psi \\ c\theta \cdot s\psi & s\varphi \cdot s\theta \cdot s\psi + c\varphi \cdot c\psi & c\varphi \cdot s\theta \cdot s\psi - s\varphi \cdot c\psi \\ -s\theta & s\varphi \cdot c\theta & c\varphi \cdot c\theta \end{bmatrix} \quad 7\text{-}20$$

where $s(\cdot)$ and $c(\cdot)$ represents $sin(\cdot)$ and $cos(\cdot)$.

By neglecting the air friction, the quadrotor body is only under gravity and motor thrust. The motor and propellers are mounted in the horizontal plane of the body; therefore the forces on the quadrotor are all along z-axis in the body frame. Therefore, the linear motion equation can be written as:

$$m\ddot{X} = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + R \cdot T_t \tag{7-21}$$

where m is the quadrotor mass and g is the gravitational acceleration.

According to Euler's Equation of rigid body dynamics,

$$I\dot{\Omega} = -\Omega \times I\Omega + \tau_B \tag{7-22}$$

where I is the inertia matrix given by $I = \begin{bmatrix} I_x & 0 & 0 \\ 0 & I_y & 0 \\ 0 & 0 & I_z \end{bmatrix}$.

By substituting Equation (7-19) into Equation (7-22), the rotation motion equation can be obtained:

$$\ddot{\Theta} = (\mathrm{IW})^{-1}\left(\tau_B - I\dot{W}\dot{\Theta} - \mathrm{W}\dot{\Theta} \times \mathrm{IW}\dot{\Theta}\right) = \begin{bmatrix} \widetilde{\tau_\varphi} \\ \widetilde{\tau_\theta} \\ \widetilde{\tau_\psi} \end{bmatrix} \tag{7-23}$$

The dynamical model is listed here:

$$\ddot{x} = \frac{1}{m}(\cos(\varphi) \cdot \sin(\theta) \cdot \cos(\psi) + \sin(\varphi) \cdot \sin(\psi))T_z \tag{7-24}$$

$$\ddot{y} = \frac{1}{m}(\cos(\varphi) \cdot \sin(\theta) \cdot \sin(\psi) - \sin(\varphi) \cdot \cos(\psi))T_z \tag{7-25}$$

$$\ddot{z} = \frac{1}{m}(\cos(\varphi) \cdot \cos(\theta))T_z - g \tag{7-26}$$

$$\ddot{\varphi} = \widetilde{\tau_\varphi} \tag{7-27}$$

$$\ddot{\theta} = \widetilde{\tau_\theta} \tag{7-28}$$

$$\ddot{\psi} = \widetilde{\tau_\psi} \tag{7-29}$$

Chapter 8.

Control

The control system is comprised of two portions:

1. **Attitude control**: the control system operated in the autopilot system, which is used to keep the quadrotor body in desired attitude angles.
2. **Position control**: the control system operated on computer, which is used to control the quadrotor to move to desired position in horizontal level.

The attitude control of the quadrotor is introduced in Section 8.1, and several position control systems are studied in section 8.2.

8.1 Attitude control

In this thesis, the built-in attitude controller in Ardupilot is used directly. The control diagram is shown in Figure 8-1 below. The controller input is the target angle and outputs are motor velocity commands. The angle error between target and actual angles are converted into a desired rotation rate using a "square root proportional" controller. Then the rotation rate error between desired and actual rate is converted into the speed command of brushless motor through a PID controller.

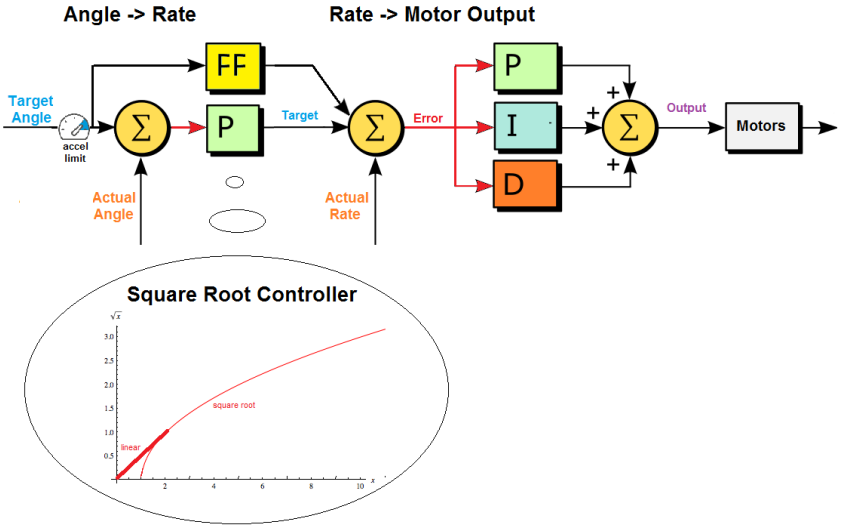


Figure 8-1 Attitude Control diagram [30]

8.2 Position control

The tracking task is operated under “Althold” mode. In “Althold” mode, the altitude is maintained constant by the APM automatically using the following control algorithm presented below. The altitude error is first converted into the desired climb or descent rate using a proportional controller. Afterwards, the rate error is converted into the desired acceleration through a PID controller. Next, the acceleration error is converted into the motor command using a second PID controller.

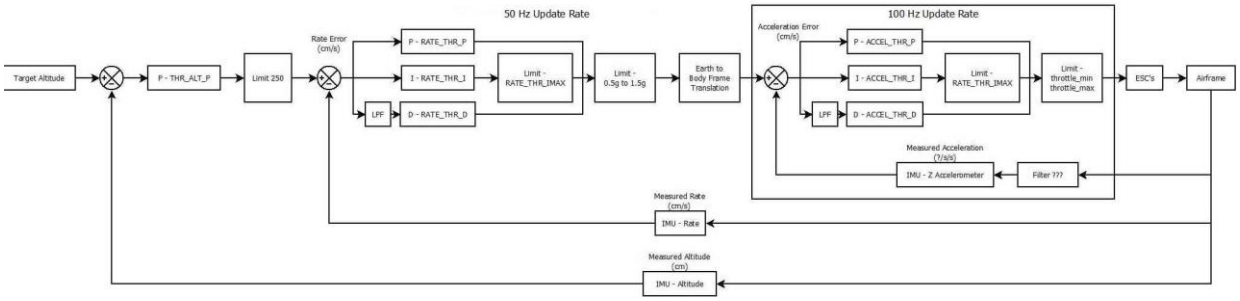


Figure 8-2 Altitude control diagram [31]

When the altitude is held constant, the gravity is balanced out by the total lift force: therefore, the total force in z axis in the inertial frame is zero. Thus, the motion equation in z direction can be written as:

$$\frac{1}{m}(\cos(\varphi)\cdot\cos(\theta))T_z = g \tag{8-1}$$

and the total thrust force is:

$$T_z = \frac{mg}{\cos(\varphi)\cdot\cos(\theta)} \tag{8-2}$$

Pitch and roll angles aresufficient to control the quadrotor’s horizontal motion, therefore the yaw angle is held constant. The x direction of the inertial frame is defined to be same as the quadrotor heading, thus $\psi = 0$. The motion equation in x and y direction can be written as:

$$\ddot{x} = \frac{1}{m}(\cos(\varphi)\cdot\sin(\theta))T_z \tag{8-3}$$

$$\ddot{y} = -\frac{1}{m}\sin(\varphi)T_z \tag{8-4}$$

Substituting Equation (8-2) into Equation (8-3) and (8-4),

$$\ddot{x} = tan(\theta)g \tag{8-5}$$

$$\ddot{y} = -\frac{\tan(\varphi)}{\cos(\theta)}g \tag{8-6}$$

Assuming the pitch and roll angles are small during flight ($\theta < 10°$ and $\varphi < 10°$):

$$tan(\theta) \approx \theta,\ tan(\varphi) \approx \varphi \text{ and } \cos(\theta) \approx 1$$

Using the above small angle approximation, equation (8-5) and (8-6) can be linearized into:

$$\ddot{x} = \theta \cdot g \tag{8-7}$$

$$\ddot{y} = -\varphi \cdot g \tag{8-8}$$

The position of the target is denoted as (x_t, y_t). In the tracking task, the states of the target are not measurable and predictable. It is assumed that the target does not have a large acceleration, therefore $\ddot{x_t} \approx 0$ and $\ddot{y_t} \approx 0$. Without using any information from GPS module, there are no sensors measuring the current position and velocity of the quadrotor. So, the only measurable states are relative position and relative velocity calculated from target pixel location, which are given by

$$x_t - x = X = \frac{H \cdot \tilde{x}}{f_x} \tag{8-9}$$

$$y_t - y = Y = \frac{H \cdot \tilde{y}}{f_y} \tag{8-10}$$

$$\dot{x_t} - \dot{x} = \dot{X} = \frac{H \cdot \dot{\tilde{x}}}{f_x} \tag{8-11}$$

$$\dot{y_t} - \dot{y} = \dot{Y} = \frac{H \cdot \dot{\tilde{y}}}{f_y} \tag{8-12}$$

The position controller is designed to stabilize the system by driving $(x - x_t) \to 0$ and $(y - y_t) \to 0$.

8.2.1 PD controller

The motions in the x and y directions are controlled by the pitch and roll angle separately. First, the position control in x direction is considered. The control law is given by

$$\ddot{x} = k_p(x_t - x) + k_d(\dot{x}_t - \dot{x}) \quad \text{8-13}$$

$$\text{where } k_p > 0 \text{ and } k_d > 0$$

8.2.1.1 Stability analysis

To verify the stability of the system, the candidate Lyapunov function is presented below:

$$V = \frac{k_p}{2k_d}(x_t - x)^2 + \frac{1}{2k_d}(\dot{x}_t - \dot{x})^2 \quad \text{8-14}$$

$$\dot{V} = \frac{k_p}{k_d}(x_t - x)(\dot{x}_t - \dot{x}) + \frac{1}{k_d}(\dot{x}_t - \dot{x})(\ddot{x}_t - \ddot{x})$$

$$\dot{V} = (\dot{x}_t - \dot{x})\left(\frac{k_p}{k_d}(x_t - x) + \frac{1}{k_d}\left(\ddot{x}_t - k_p(x_t - x) - k_d(\dot{x}_t - \dot{x})\right)\right)$$

$$\dot{V} = -k_d(\dot{x}_t - \dot{x})^2 + \frac{1}{k_d}(\dot{x}_t - \dot{x})\ddot{x}_t$$

As assumed before, for small $\ddot{x}_t \approx 0$:

$$\dot{V} = -k_d(\dot{x}_t - \dot{x})^2 \leq 0$$

Therefore, the system is stable.

Similarly, $\ddot{y} = k_p(y_t - y) + k_d(\dot{y}_t - \dot{y})$ and the desired output pitch and roll angles are

$$\theta_d = \frac{\ddot{x}}{g} \text{ and } \varphi_d = -\frac{\ddot{y}}{g} \quad \text{8-15}$$

If large angles are expected,

$$\theta_d = \text{atan}\left(\frac{\ddot{x}}{g}\right) \text{ and } \varphi_d = -\text{atan}\left(\frac{\ddot{y}}{g} \cdot \cos(\theta)\right) \quad \text{8-16}$$

Large attitude angle inputs are not recommended in this thesis. A $10°$ lean angle can provide an acceleration of $1.7\ {}^{m}/_{s^2}$, which is sufficient for a normal moving target. Large input angles lead to a large proportional coefficient, which may cause a high overshoot and oscillation of the whole system.

8.2.1.2 Derivate estimator

The relative velocity is calculated from the pixel velocity. The pixel velocity (pixel location change rate) can be calculated in a traditional way by

$$\dot{\tilde{x}} = \frac{\tilde{x}_k - \tilde{x}_{k-1}}{\Delta t_k} \tag{8-17}$$

where $\tilde{x}_k$ and $\tilde{x}_{k-1}$ are the pixel location in the current frame and previous frame respectively, and Δt_k is the time interval between two frames.

The adaptive variable structure derivate estimator (AIVSDE) [32] is used to calculate the velocity in this thesis. For an input signal $r(t)$, the output signal $x_0(t)$ and the switching function $\sigma(t)$ can be constructed by

$$\dot{x}_0(t) = \hat{k}_r(t) \cdot sgn(\sigma) + k_b \cdot \sigma(t) - \frac{k_1}{k_2} \cdot x_0(t) \tag{8-18}$$

$$\sigma(t) = k_2 \cdot (r(t) - x_0(t)) + k_1 \cdot \int_0^t (r(\tau) - x_0(\tau))\, d\tau \tag{8-19}$$

where k_b, k_1 and $k_2 \geq 0$ and the adaptive gain $\hat{k}_r(t)$ is given by the law

$$\dot{\hat{k}}_r = \begin{cases} \alpha_r, & for\ |\sigma(t)| \geq \mu \\ 0, & for\ |\sigma(t)| < \mu \end{cases} \tag{8-20}$$

where $\alpha_r\ and\ \mu \geq 0$. The output signal $x_0(t)$ approaches $r(t)$, and $\dot{x}_0(t)$ estimates the first derivative of $r(t)$.

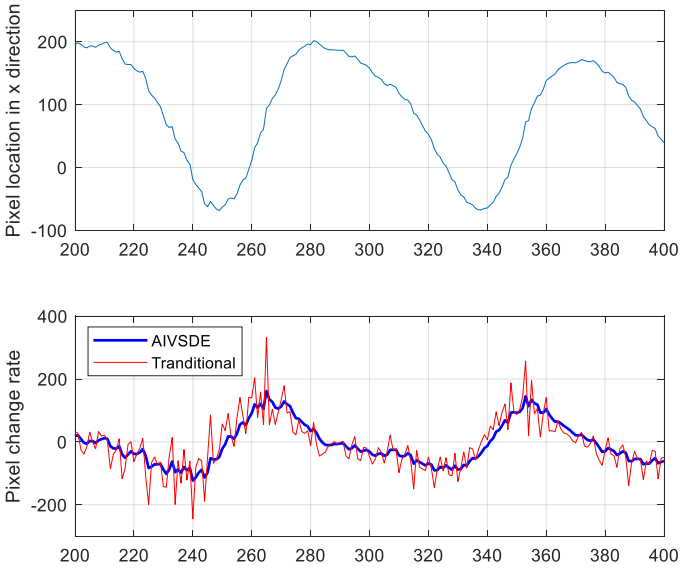


Figure 8-3 Comparison of two velocity estimation

The performance of AIVSDE is shown in Figure 8-3 above. In the lower figure, the traditional velocity estimation computed using Equation (8-16) is represented by the red curve. It can be seen that the AIVSDE is significantly smoother than the traditional estimate.

8.2.2 Fuzzy control

The fuzzy controller is a controller based on control experience and fuzzy rules. The tracking error and its change amount are the inputs of fuzzy rules, which are denoted as e and Δe and control outputs are fuzzy rule outputs according to the rule table. Based on the error and error rate, the ijth fuzzy control rule can be expressed as

$$\text{If } e = \mu_i \text{ and } \Delta e = \nu_j, \text{ then } u = u_{ij}$$

The corresponding fuzzy rule table is shown in Table 8-1 below:

u		e			
		μ_1	μ_2	$\cdots$	μ_i
Δe	ν_1	u_{11}	u_{21}	$\cdots$	u_{i1}
	ν_2	u_{12}	u_{22}	$\cdots$	u_{i2}
	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
	ν_j	u_{1j}	u_{2j}	$\cdots$	u_{ij}

Table 8-1 Fuzzy rule table

The membership function is defined as

$$f_{ij} = \mu_i(e) \cdot \nu_j(\Delta e) \tag{8-21}$$

where μ_i and ν_j are the membership functions of the error and the error rate.

By the defuzzification process, the output is given by

$$u = \frac{\sum_{i,j} f_{ij} \cdot u_{ij}}{\sum_{i,j} f_{ij}} \tag{8-22}$$

The normalized pixel location and pixel difference of the target are proportional to the target position and velocity. Therefore, if changes in altitude are neglected, pixel location and pixel difference can be considered as error (e) and error rate (Δe). Thus, $e = \tilde{x}$ and $\Delta e = \Delta\tilde{x}$. The membership functions are listed below:

$$\mu_1 = e^{-(\frac{x+1}{\pi/6})^2} \quad 8\text{-}23$$

$$\mu_2 = e^{-(\frac{x}{\pi/6})^2} \quad 8\text{-}24$$

$$\mu_3 = e^{-(\frac{x-1}{\pi/6})^2} \quad 8\text{-}25$$

$$\nu_1 = e^{-(\frac{x+10}{2\pi})^2} \quad 8\text{-}26$$

$$\nu_2 = e^{-(\frac{x}{2\pi})^2} \quad 8\text{-}27$$

$$\nu_3 = e^{-(\frac{x-10}{2\pi})^2} \quad 8\text{-}28$$

Nine fuzzy rules are defined:

- If e is P and Δe is P, then u is BP.
- If e is P and Δe is Z, then u is MP.
- If e is P and Δe is N, then u is Z.
- If e is Z and Δe is P, then u is MP.
- If e is Z and Δe is Z, then u is Z.
- If e is Z and Δe is N, then u is MN.
- If e is N and Δe is P, then u is Z.
- If e is N and Δe is Z, then u is MN.
- If e is N and Δe is N, then u is BN.

u_{ij}		e		
		N	Z	P
Δe	N	BN	MN	Z
	Z	MN	Z	MP
	P	Z	MP	BP

Table 8-2 Fuzzy controller lookup table

BP, MP, Z, MN and BN represent big positive, median positive, zero, median negative and big negative. They are set to be 1, 0.5, 0, -0.5 and -1 respectively. The design of the fuzzy rules is explained below:

1. When the error and the error rate are both positive, it means the target is in the positive position relative to the quadrotor and moving away from the quadrotor. Therefore, the output should be positive big to produce an acceleration in the positive direction to drive the quadrotor to chase the target and keep the target within the detection range.
2. When the error is positive, and the error rate is close to zero, it means the target is in the positive position relative to the quadrotor but not moving away from the quadrotor in a high speed. Therefore, the output should be median positive to drive the quadrotor to move in the positive direction and reduce the distance to the target. In order to decrease the overshoot, the output should not be too large.
3. When the error is positive, and the error rate is negative, it means the target is in the positive position relative to the quadrotor but moving towards the quadrotor. Therefore, the output should be close to zero to reduce the overshoot.
4. When the error is close to zero, and the error rate is positive, it means the target is directly below the quadrotor but moving in the positive direction. Therefore, the output should be median positive to drive the quadrotor to move in the same direction with the target and avoid the potential increase of the distance.
5. When the error and the error rate are both close to zero, it means the target is directly below the quadrotor and they are relative static. Therefore, the output should be zero to keep the current states.
6. Other conditions are symmetric to the conditions listed above.

8.2.3 Self-tuning PD control

PD controller can be combined with fuzzy logic in parameter tuning. The proportional and derivate term can be tuned with the change of error and error rate by fuzzy logic to improve the control performance. The structure of the self-tuning PD controller based on fuzzy logic is shown in Figure 8-4 below.

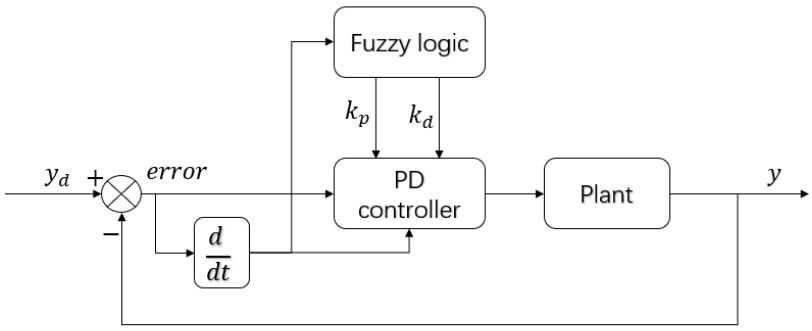


Figure 8-4 Self-tuning PD controller diagram

Two coefficients c_p and c_d are obtained from fuzzy logic. By multiplying the coefficient, self-tuning terms become $k_p{}' = c_p \cdot k_p$ and $k_d{}' = c_d \cdot k_d$. Therefore, the expression for the self-tuning PD controller becomes:

$$\ddot{x} = c_p \cdot k_p(x_t - x) + c_d \cdot k_d(\dot{x}_t - \dot{x}) \quad 8\text{-}29$$

For a PD controller, the proportional term P is used to increase the response speed. Large P can increase response speed but may also cause overshoot. Small P decreases respond time and increases settling time. Derivate term D is used to predict the error change and avoid overshoot.

Based on the knowledge of proportional and derivate term, the membership functions and fuzzy rule tables are shown below.

$$\mu_1 = e^{-(\frac{x+1}{\pi/9})^2} \quad 8\text{-}30$$

$$\mu_2 = e^{-(\frac{x+0.5}{\pi/9})^2} \quad 8\text{-}31$$

$$\mu_3 = e^{-(\frac{x}{\pi/9})^2} \quad 8\text{-}32$$

$$\mu_4 = e^{-(\frac{x-0.5}{\pi/9})^2} \quad 8\text{-}33$$

$$\mu_5 = e^{-(\frac{x-1}{\pi/9})^2} \quad 8\text{-}34$$

$$v_1 = e^{-(\frac{x+10}{\pi})^2} \quad 8\text{-}35$$

$$v_2 = e^{-(\frac{x+5}{\pi})^2} \quad 8\text{-}36$$

$$v_3 = e^{-(\frac{x}{\pi})^2} \quad 8\text{-}37$$

$$v_4 = e^{-(\frac{x-5}{\pi})^2} \quad 8\text{-}38$$

$$v_5 = e^{-(\frac{x-10}{\pi})^2} \quad 8\text{-}39$$

c_p		e				
		NB	NM	Z	PM	PB
Δe	NB	B	B	B	S	B
	NM	B	M	B	S	B
	Z	B	M	M	M	B
	PM	B	S	B	M	B
	PB	B	S	B	B	B

Table 8-3 Fuzzy rule table for K_p

c_d		e				
		NB	NM	Z	PM	PB
Δe	NB	B	M	S	M	S
	NM	M	M	M	M	S
	Z	S	M	B	M	S
	PM	S	M	M	M	M
	PB	S	M	S	M	B

Table 8-4 Fuzzy rule table for K_d

NB, NM, Z, PM and PB represent negative big, negative median, zero, positive median and positive big. B, M and S are set to be 1.5, 1 and 0.5.

In this study, the error and error rate are divided into five cases. The tuning principle of the proportional term is explained here:

1. When the magnitude of the error is big, it means the target is far away from the

quadrotor and close to the detection boundary. In order to prevent the target from getting out of the detection range, no matter what the value of the error rate is, k_p should be big to produce a large feedback force to decrease the error as soon as possible.

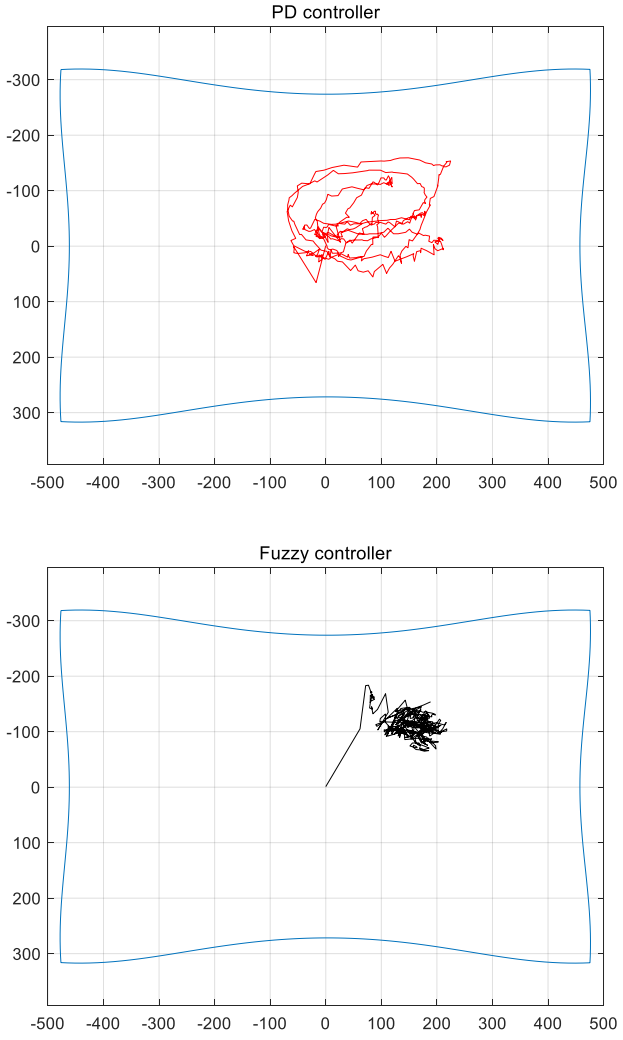
2. When the magnitude of the error is median, and the error rate is in the opposite direction, it means the target has an acceptable distance to the quadrotor and is moving towards the quadrotor. Therefore, k_p is set to be small to reduce the feedback output to avoid overshoot. If the error rate is zero or median in the same direction with the error, k_p is set to median. The output should not be large to increase overshoot, or small to increase the distance. If the error rate is big in the same direction, it means the target is moving away from the target and it is possible to approach or even get out of the detection boundary. Therefore, k_p should be big to produce a large feedback force to drive the quadrotor to move in the same direction with the target and keep the distance not increasing.
3. When the error and the error rate are both around zero, it means the target is directly below the quadrotor. Therefore, k_p is set to median to produce moderate feedback output to reduce the error. Otherwise, k_p should be set to big to restrain error increase.
4. When the magnitude of the error is big, and the error rate is in the same direction, it means the target is getting close to the detection boundary. Therefore, k_d should be big to increase the feedback force to drive the quadrotor to chase the target. If the error rate is median in the same direction, k_d can be set to median. If the error rate is around zero or in the opposite direction, k_d should be set to small to reduce the restrain effect.
5. When the error is median, k_d does not have to be specified and can be set to median.
6. When the error is around zero, the feedback output due to the derivate part is used to restrain the movement trend. k_d should not be too large to create oscillation or too small to lose its function of restraining the distance change rate. Therefore, k_d value is selected to decrease with the increase of the error rate magnitude.

Chapter 9.

Test and Results

The automatic tracking is tested outside on a large grass field for safety. The target can move on the grass field by being dragged by a RC car. GPS data is not used in the automatic tracking, but two GPS modules are mounted on the RC car and the quadrotor to collect path data.

9.1 Static target tracking



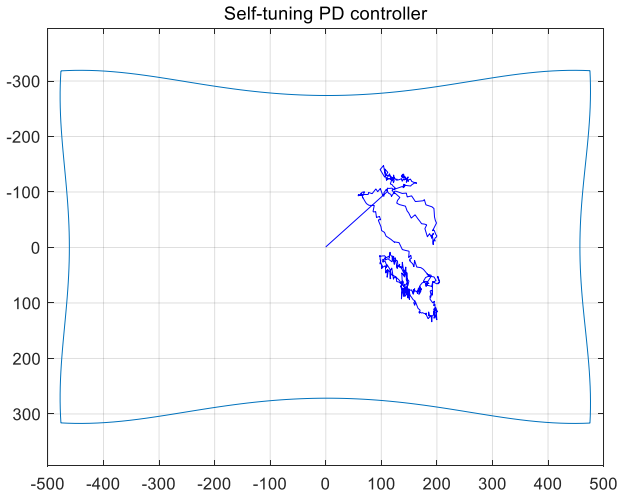


Figure 9-1 Pixel location path for static target tracking I

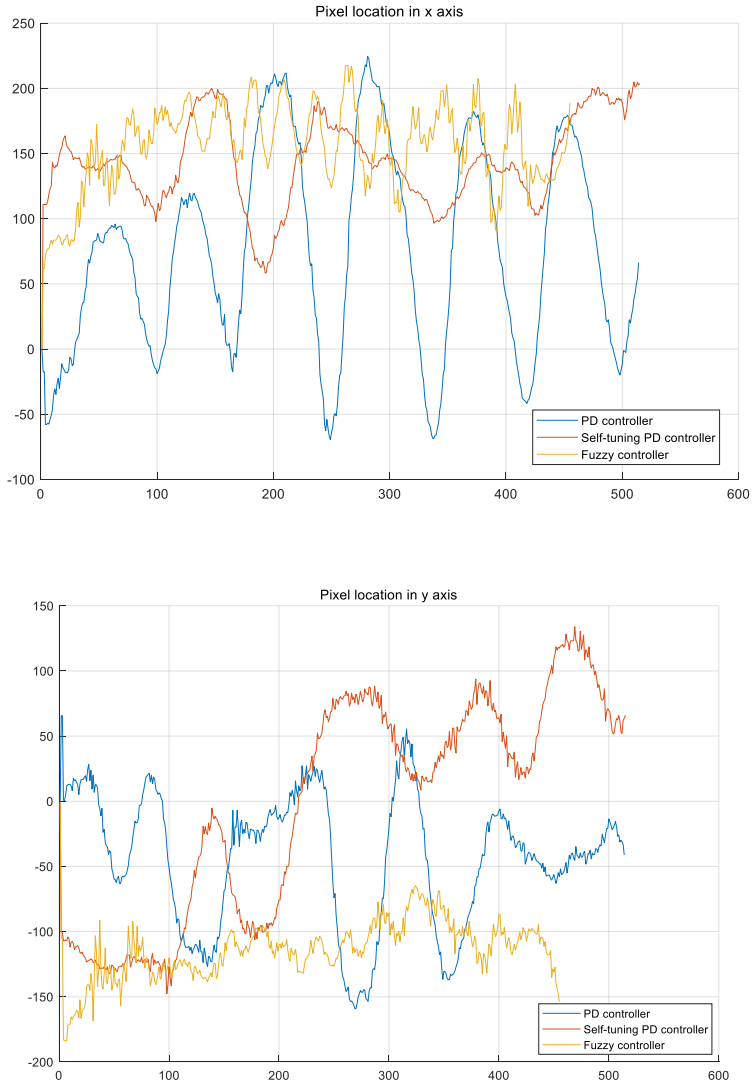


Figure 9-2 Pixel location path for static target tracking II

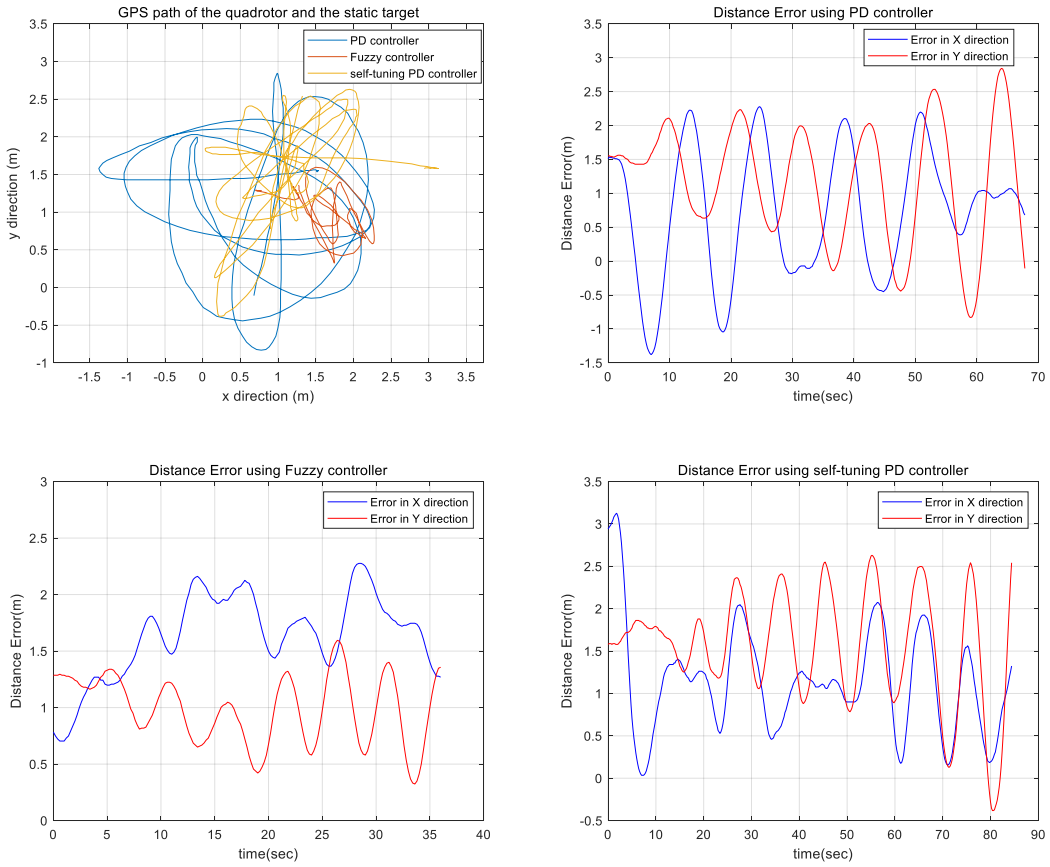


Figure 9-3 GPS path of the quadrotor and the static target

The test field is chosen to be outside, so wind disturbance is inevitable even though the wind speed is low (below 5 km/h). Pixel locations for three controllers are shown in Figure 9-1 and Figure 9-2. The path of the quadrotor is shown in Figure 9-3 based on GPS data. The upper left figure shows three flight paths using different controllers when the quadrotor tracked static target. The upper right, lower left and lower right figures show the distance error between the quadrotor and the target in both x and y directions when using PD controller, Fuzzy controller and self-tuning PD controller respectively.

With the PD controller, the quadrotor can hover above the target but keeps moving around. The route of pixel location of the target occupies the largest area.

The performance of the fuzzy controller is better than the PD controller. The quadrotor can almost hover in a stable position. However, the attitude angles change much more rapidly and intensively during the hovering. The reason may be that the outputs of the fuzzy controller are not smooth, and the quadrotor oscillates between positive and negative angles frequently. As a

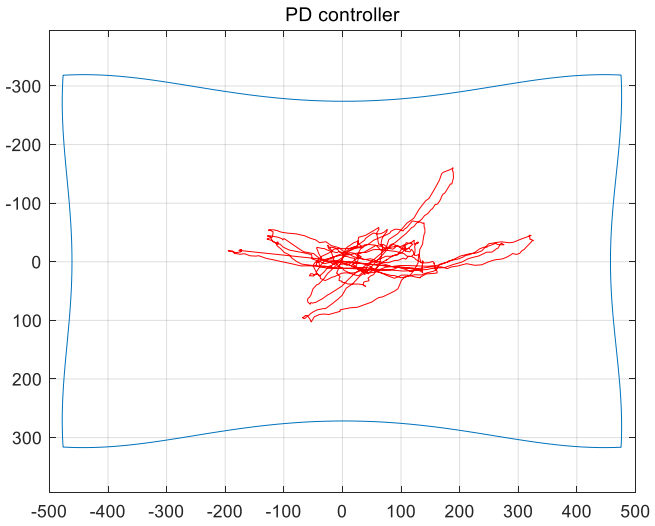
result, the camera shakes even though it is stabilized by a camera gimbal, and the blurry image caused by shaking can lead to the failure of vision detection.

With the self-tuning PD controller, the quadrotor almost hovers stably near the target, and the attitude changes smoothly.

The reason that the quadrotor cannot hover perfectly on the target but instead keeps some distance is because of the offset error. In the “althold” mode, if the attitude inputs are zero pitch and roll, the quadrotor can still move and drift in some directions. This offset error is caused by external forces such as wind disturbance and hardware issues including small imbalance of the frame and small noises in accelerometer and IMU.

9.2 Moving target tracking

In the moving target tracking test, fuzzy controller is not tested for its shaking screen. Pixel locations are shown in Figure 9-4 and Figure 9-5. The flight path of the quadrotor in horizontal plane is shown in Figure 9-6 and the 3D flight path is created by Google Earth and shown in Figure 9-7.



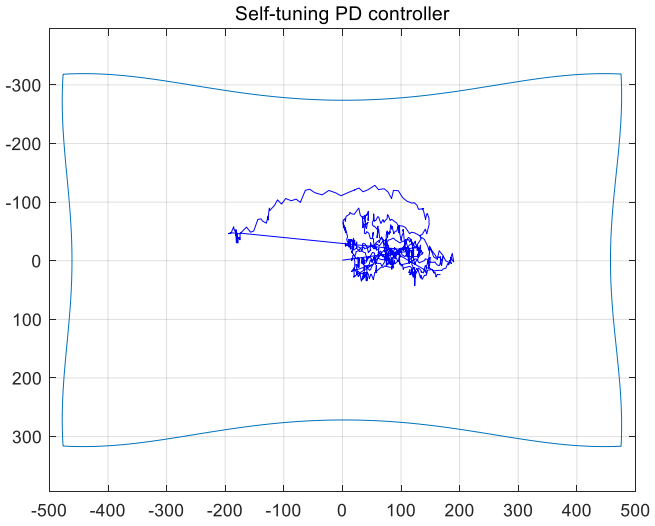


Figure 9-4 Pixel location path for moving target tracking I

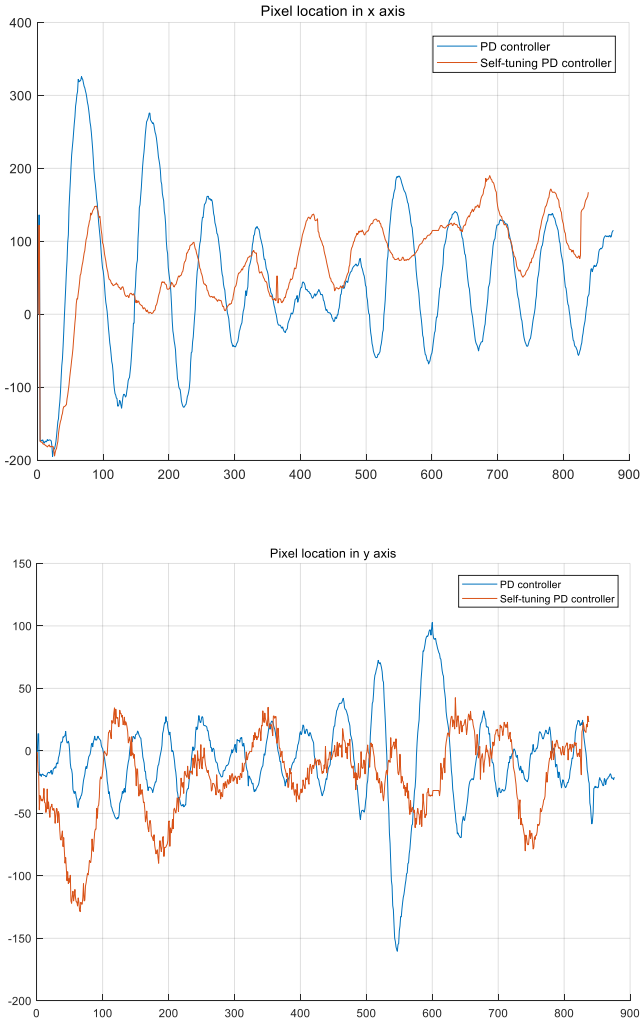


Figure 9-5 Pixel location path for moving target tracking II

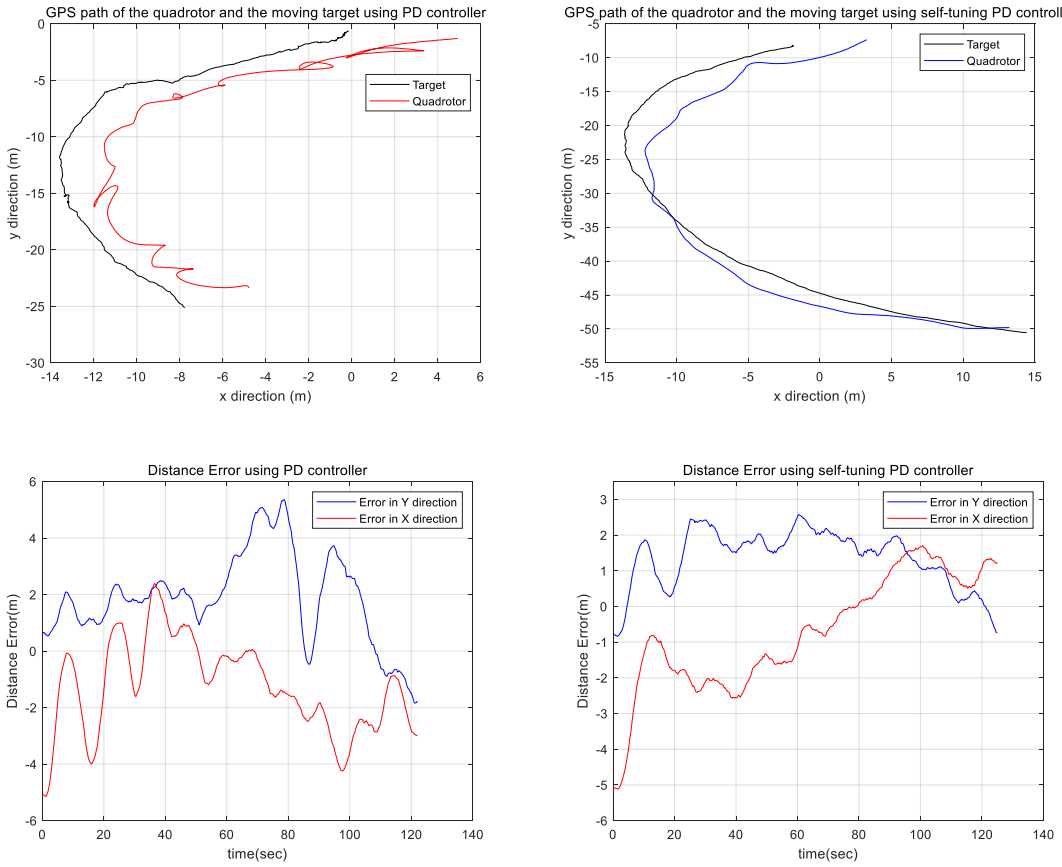


Figure 9-6 GPS path of the quadrotor and the moving target

As shown in Figure 9-4 and Figure 9-5, the quadrotor can hover above the moving the target and keep target within the camera detection range. For the self-tuning PD controller, the pixels can almost focus around origin and flight path is smoother. Figure 9-6 shows the flight path of the quadrotor and moving path of the target when the quadrotor tracked the moving target. The two upper figures show the paths of the target and the quadrotor when using PD controller and self-tuning PD controller respectively. The two lower figures show the distance error between the quadrotor and the moving target in both x and y directions when using PD controller and self-tuning PD controller respectively.

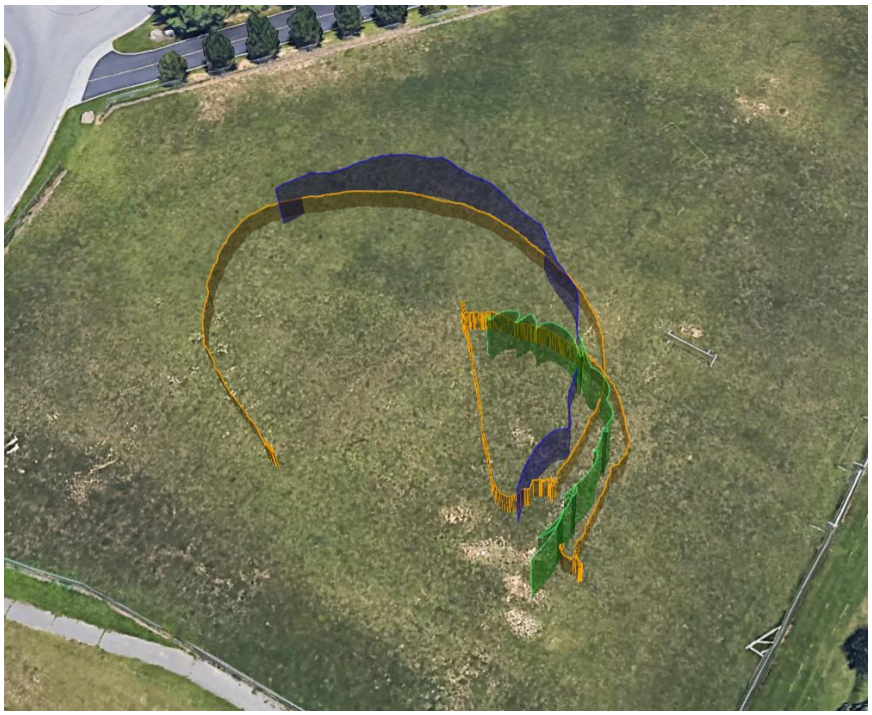


Figure 9-7 Flight path in Google Earth

Chapter 10.

Conclusion and Future Work

In this project, an automatic tracking system for a quadrotor based on computer vision is built. In the flight test, the quadrotor can follow the moving target while keeping it around the center of camera screen. Although satisfactory, several possible modifications that can be implemented to improve the tracking performance.

First, the visual detection is one of the most important processes, but the detection accuracy is affected and limited by low image quality using wireless video transmission. An on-board CPU with high-resolution camera can be used and mounted on the quadrotor. The image processing and position calculation can be down directly onboard. Instead of sending video signal wirelessly to a computer, only the position information is required to be sent. High-resolution camera can guarantee the details of the target and increase detection accuracy. As a result, the quadrotor can also fly at a higher altitude and obtain a wider detection range. Furthermore, On-board image processing removes the process of video transmission, which can effectively reduce the image noise and increase calculation speed.

Second, the optical flow sensor can be used to measure the ground speed of the quadrotor. Angle compensation can be calculated by implementing extended Kalman filter, so that the quadrotor can accurately maintain position even in a windy environment.

References

[1] Celine Teuliere, Laurent Eck, Eric Marchand. *Chasing a moving target from a flying UAV*. IROS IEEE 2011.

[2] García Carrillo, L.R., Dzul López, A.E., Lozano, R., Pégard, C. *Quad Rotorcraft Control Vision-Based Hovering and Navigation.*

[3] Miguel A. Olivares-Mendez, Somasundar Kannan, Holger Voos. *Vision Based Fuzzy Control Autonomous Landing with UAVs: From V-REP to Real Experiments.* 23rd Mediterranean Conference on Control and Automation (MED), 2015.

[4] Denys Bohdanov. *Quadrotor UAV Control for Vision-based Moving Target Tracking Task.*

[5] Miguel A. Olivares-Mendez, Somasundar Kannan, Holger Voos. *Setting Up a Testbed for UAV Vision Based Control Using V-REP & ROS: A Case Study on Aerial Visual Inspection.* International Conference on Unmanned Aircraft Systems (ICUAS), 2014.

[6] Miguel A. Olivares-Mendez, Somasundar Kannan, and Holger Voos. *V-REP & ROS Testbed for Design, Test, and Tuning of a Quadrotor Vision Based Fuzzy Control System for Autonomous Landing.*

[7] Miguel A´ ngel Olivares Me´ndez. *Soft-Computing Based Visual Control for Unmanned Vehicles.*

[8] J. E. Gomez-Balderas, G. Flores, L. R. García Carrillo, R. Lozano. *Tracking a Ground Moving Target with a Quadrotor Using Switching Control*. Journal of Intelligent and Robotic Systems, Springer Verlag (Germany), 2013.

[9] JeongWoon Kim, David Hyunchul Shim, James R. Morrison. *Tablet PC-based Visual Target-Following System for Quadrotors*. Journal of Intelligent and Robotic Systems, April 2014.

[10] JeongWoon Kim, David Hyunchul Shim. *A Vision-based Target Tracking Control System of a Quadrotor by using a Tablet Computer*, 2013 International Conference on Unmanned Aircraft Systems (ICUAS), May 2013.

[11] Doyoon Kim, Sung Kyung Hong. *Target pointing and Circling of a Region of interest with Quadcopter*. International Journal of Applied Engineering Research Volume 11.

[12] Ehsan Abbasi, Mohammad Mahjoob.*Quadrotor UAV Guidence For Ground Moving Target Tracking*.Journal of Advances in Computer Engineering and Technology, 2016.

[13] Daniel H. S. De Mel, Karl A. Stol, Jay A. D. Mills, Blair R. Eastwood. *Vision-Based Object Path Following on a Quadcopter for GPS-Denied Environments*. 2017 International Conference on Unmanned Aircraft Systems (ICUAS).

[14] Jia Yang, Xing Huo, Bing Xiao, Zhenzhou Fu, Chaofan Wu, Yiran Wei. *Visual Servo Control of Unmanned Aerial Vehicles: An Object Tracking-based Approach*. 2017 29th Chinese Control And Decision Conference (CCDC).

[15] Ardupilot Copter Website. http://ardupilot.org/copter/index.html

[16] DX6I Instruction Manual

[17] Mathworks. What is Camera Calibration? https://www.mathworks.com/help/vision/ug/camera-calibration.html

[18] QGROUNDCONTROL. Mavlink. http://qgroundcontrol.org/mavlink/start

[19] CCD & CMOS Cameras. Image downloaded from https://www.qualitymag.com/articles/91698-ccd-cmos-cameras-101

[20] Camera Gimbal with Servos. Image downloaded from http://ardupilot.org/copter/docs/common-camera-gimbal.html

[21] Connection of ESCs and Motors. Image downloaded from http://ardupilot.org/copter/docs/connect-escs-and-motors.html

[22] Example of salt-and-pepper noise and the performance of Median filter. Image downloaded from https://en.wikipedia.org/wiki/Median_filter

[23] OpenCV, Contours Hierarchy. Image downloaded from https://docs.opencv.org/trunk/d9/d8b/tutorial_py_contours_hierarchy.html

[24] Scanning QR Codes. Image downloaded from http://aishack.in/tutorials/scanning-qr-codes-1/

[25] QGROUNDCONTROL. Packet Anatomy. Image downloaded from http://qgroundcontrol.org/mavlink/start

[26] QGROUNDCONTROL. Packet Anatomy. Table from http://qgroundcontrol.org/mavlink/start

[27] PPM outpout. Image downloaded from
https://hackaday.io/project/2465-manucon-a-glove-based-controller/log/8254-ppm-output
[28] Clancy, L. J. *Aerodynamics*. Pitman Publishing Limited, London
[29] Goldstein, H. *Classical Mechanics*.
[30] Copter Attitude Control. Image downloaded from
http://ardupilot.org/dev/docs/apmcopter-programming-attitude-control-2.htm
[31] Altitude Hold Mode. Image Downloaded from
http://ardupilot.org/copter/docs/altholdmode.html#altholdmode
[32] Chih-Chiang Cheng, Ming-Wen Chang. *Design of Derivate Estimator Using Adaptive Sliding Mode Technique.* American Control Conference, 2006.