A Study of Micro-UAV and its Applications

Tianpeng Gou

An engineering project report submitted
in partial fulfillment for the degree of
Bachelor of Engineering
in the
Faculty of Engineering and IT
University of Sydney

Supervisor: Dr Xiaofeng Wu

October 2014

Statement of Student Contribution

- I carried out the research and selection of micro-UAV, and the final selection was proved by supervisor
- I performed a comprehensive study of the selected micro-UAV- Crazyflie 1.0 by my own, on both hardware and software.
- I practice all the project build-up with the aid of open source software.
- I performed a series of auto-tracking of micro-UAV tasks and analyse the result by myself.
- I cited all the materials and pictures in this report which are not belonged to my own work.

The above represents an accurate summary of the student's contribution.

Signed..... student .....(supervisor)

Abstract

During the last decade, UAV development has experienced a transaction from large size fixed-wing based military supportive/surveillance aircraft to mini size urban adoptive 4-blade rotatory copter. Recent years, more and more research groups have put interest and effort into developing a much smaller size UAV to fit into a closed indoor area. As plenty of advanced technologies have been converted to civilian use, and the life-style's transaction from outdoor to indoor, the developer tends to create hand-size 'toy' to keep people entertained while pushing the frontier of micro-technology.

Aerospace Department of USYD has started to develop Mircro-UAV to serve multiply purposes. One of the research interests is to assist the functionality of the Planetary Rover developed by the ACFR. Hence, in this project, an open-platform micro-UAV/quadcopter called Crazyflie, developed in Sweden, will be studied. In addition, Crazyflie will also be used to test out the feasibility of automatically tracking flight with motion camera.

Figure 1: Crazyflie Prototype

Acknowledgment

The author wishes first to express his deep gratitude to Dr. Xiaofeng Wu. Not only his constant guidings and suggestions to author to carry out the study, but also his inspiration and warm personality have won author's highest respect.

The author also wishes to thank his parents, and his friends Jie Cao, Xuan Qin, Youzhi Hu and Xiaodie Zhang for giving him help and supporting him to complete this project.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Brief Development History of UAV

Over the last decade, UAV has gained a large amount of usage both in the perspective of military and civilian. Especially, during this period, UAV development has experienced a transaction from large size fixed-wing based military supportive/surveillance aircraft to mini size urban adoptive 4-blade rotatory copter.

(a) Global Hawk
(www.xconomy.com)

(b) Quad Copter
(www.aeroquad.com)

Figure 2: Development of UAVs

As shown in Figure 2, the distinguish features of two types of UAVs has told their functionality in different areas. Where the Global Hawk has a good fame of effectively detecting hostile movement in high-risk enemy zone without endangering man force, and Quad Copter has been broadly deployed to the large populated urban area to minimize workload and carryout the mission beyond civilian ability. Such like fertilizing the corps, so it can be 3 times faster than tradition way, or filming mission, so it can be conducted by civilians with basic training and low cost other than the helicopter way, or even delivering a pizza, so you will never worry about customer's complaint due to the traffic jam!

The extensive usage of those UAVs requires them to be able to operate in a long-range, so manual control is not functional when UAV is far from control station. Apart from the advanced satellite communication and guidance ability of military UAV, the civilian one can rely on the GPS to set waypoints to automatically guide

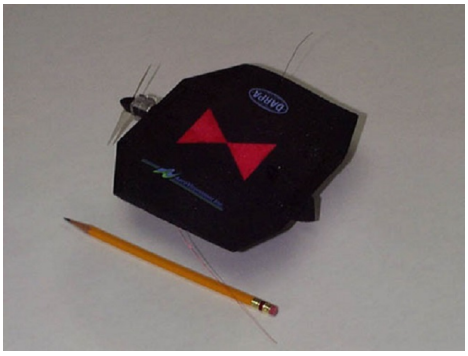
the UAV when it is not within the radio receiving range.

Recent years, more and more research groups have put interest and effort into developing a much smaller size UAV to fit into a closed indoor area. As plenty of advanced technologies have been converted to civilian use, and the life-style's transaction from outdoor to indoor, the developer tends to create hand-size 'toy' to keep people entertained while pushing the frontier of micro-technology.

(a) CIA Bio-Insectothopter
(www.alert5.com)

(b) Micro-QuadCopter
(www.kitguru.net)

(c) Micro-Helis UAV
(www.amevoice.com)



(d) BlackWidow UAV
(www.draganfly.com)

Figure 3: Conceptional Micro-UAVs

Such examples can be found in Figure 3, the Bio-Insectothopter is modeled by the inspiration of dragonfly to simulate the behavior of natural insect, thus CIA could manipulate it into a secret mission, just like the 007 movie! Or a more realistic and easy-to-get Micro-QuadCopter is made to either manually play a indoor dogfight or automatically fly in a formation like a swarm. Other examples such like the military surveillance Micro-Helis and Black Widow are also the development of Micro-UAVs in minimizing size of conventional aircraft and helicopter.

1.2 Project Interest of Mircro-UAV

Initially, like the other facilities around the world, Aerospace Department of USYD has started to develop Mircro-UAV to serve multiply purposes. One of the research interests is to assist the functionality of the Planetary Rover developed by the ACFR, shown in Figure 4. The idea is that when the rover is deployed on any foreign plant, the restricted vision may cause rover blindly enter deep hole or gap so the rover would never get back into normal plane again and fail the mission. Therefore, a device that can foresee the condition of terrain around the rover is in-need to prevent such issue happens.

Figure 4: Mawson the Planetary Rover
(acfr.usyd.edu.au)

Hence, one of solutions is to allow the rover to carry a Micro-UAV, and quadcopter is preferred due to its non-mechanical control and ease of takeoff and landing ability just like conventional helicopter. The mission of this mini-quadcopter is to automatically take off from the rover's platform and enter the detecting range with proper guidance. When the mini-quadcopter is in position, the sonar or camera device embedded on it can transmit terrain data to the rover and warn any danger if possible.

As a challenge, this project will carry out based on finding a suitable Micro-UAV (i.e. mini-quadcopter) and test on the viability on close-range guidance method such liking motion detection or image processing technique.

1.3 Layout of Project

1. Introducing the main role of this project - Crazyflie 1.0.
2. Review and Explain the features of Crazyflie.
3. Analyse the hardware and accessories of Crazyflie and this project.
4. Analyse the software of Crazyflie and this project.
5. Step-by-step project software setup procedures.
6. Introduce and explain the image processing technique - OpenCV.
7. Perform the auto-tracking and tuning of Crazyflie.
8. Discuss the results and plan the future work.

2 Literature Review - UAV Sening Technology

The sensing is a big issue for this project, especially in achieving accurate tracking. According to reference [1], [2], [3] and [4], the sensing techniques (except the basic accessories Gyro, Accelerometer, Magnetometer and IMU) can be summarized as follows:

2.1 Global Positioning System

A space-based global navigation satellite system that provides reliable absolute location anywhere on Earth, when there is an unobstructed line of sight to four or more GPS satellites. GPS report aircraft-specific information such as speed, bearing and altitude.

2.2 Laser Range Finder

A device which uses a laser beam to determine the distance to an object. LRFs operate on the time of flight principle by sending a laser pulse in a narrow beam towards the object and measuring the time taken by the pulse to be reflected off the target and returned to the sender. LRFs variety of computer vision-related field, offering high-precision scanning abilities, with either single-face or 360-degree scanning modes.

2.3 Ultrasonic Sensor

It generates high frequency sound waves and evaluates the echo which is received back by the sensor. To determine the distance to an object, the sensor calculate the time interval between sending the signal and receiving the echo.

2.4 Infrared Sensor

It transmits infrared pulses and measures the time it takes to return back, estimating the distance. It works for shorter distances than the ultrasonic sensors.

2.5 Pressure Sensor

It generates a signal as a function of the pressure imposed. In aircraft, rockets, satellites and weather balloons, pressure sensors are used for altitude sensing, using

the relationship between changes in pressure relative to the altitude.

2.6 Imaging Sensor

Visual sensing is especially attractive because it is passive, non-contact, very versatile and low-cost. It provides a tremendous amount of information about a uav's environment, and it is potentially the most powerful source of information among all the sensors used on robots to date. However, it introduces several technical challenges: the mapping from the image plane to 3-dimensional coordinates can be significantly non-linear. Image processing is computationally intensive and introduces latencies from the time of capture to the time measurements are available. Occlusions, poor lighting, or failure of the image processing algorithms can result in loss of measurements for long periods of time.

2.7 Conclusion

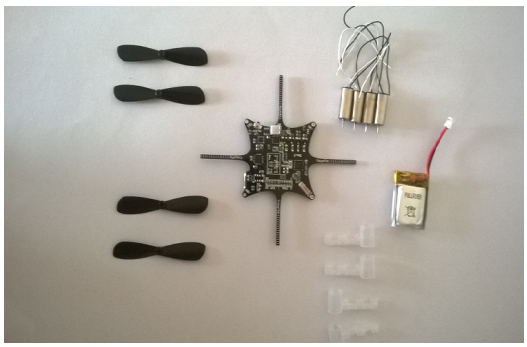
In terms of size of the micro-UAV, the best fit sensing technique for this project would be using imaging sensors, since the test environment can be placed at enclosed indoor area, thus the success rate can be vastly increased.

3 Mircro-UAV: Crazyflie 1.0

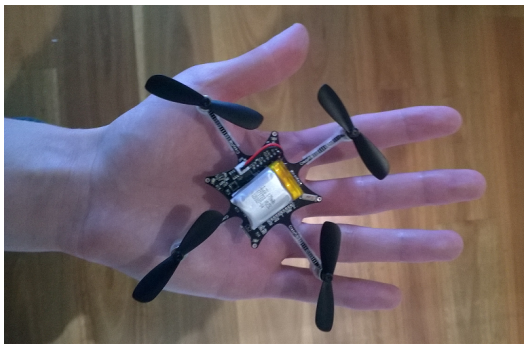
Quadcopter is suitable for this project majorly due to its non-mechanical design, where the flight attitude control is achieved by adjusting the voltage input of each motor, i.e. differ the voltage input of two adjacent motors while synchronizing two opposite ones to yaw. Due to the fact that re-design a new quadcopter from scratch is quite time-consuming, therefore picking a open platform quadcopter is the best choice for this project since the main interest is on testing the close-range guidance of the mini-robot. After a number of researches and reviews, the Swedish designed **Crazyflie** enters the sight and finally becomes the main role of this project.

3.1 Crazyflie Review

The Crazyflie Nano/Mini Quadcopter is a miniature quadcopter that fits in ones hand. It only weights about 19 grams and is 9 cm motor-to-motor. As shown in Figure 5, the Crazyflie comes as a kit, which means that assembly has to be done. This includes soldering the motors to the PCB as well as attaching the motor mounts and battery.



(a) Crazyflie Components



(b) Crazyflie Ready-to-Fly

Figure 5: Crazyflie 1.0

It's main purpose is to be a versatile development platform that can be used to experiment, develop and explore a lot of different areas of technology. The quadcopter is flown by connecting a USB radio dongle and a controller to a host system, and then running the Crazyflie client software. For input a wide range of controllers can be used when connected to your PC. In this project, first generation Crazyfile 1.0 will be used.

3.2 Crazyflie 1.0 Features and Specifications

Crazyflie platform is much more than just a quadcopter. It consists of lots of different parts working together to complete the system. That means that developers can dive into any given area and experiment, learn and improve. Each of these areas can be explored separately using other boards, but with the Crazyflie you have a complete system built around what you are experimenting with. Here's basic components and features of Crazyflie 1.0 according to Crazyflie 1.0 Manual Book[1].

- **Mini/Nano Size**

  Crazyfile 1.0 is very lightweight and small, with only 19 g and 90 mm between two opposite motor. It is one of the world smallest quadcopter.

- **Long Flight Time**

  Crazylie 1.0 carrying fully charged standard 170mAh Li-Po battery achieves 7 minutes long flight time, it uses standard micro-USB connector for charging which takes 20min.

- **Radio Communication**

  Crazyflie 1.0 features On-board low-energy radio @ 1mW based on the nRF24L01+ chip, up to 80m range (environment dependent) when using the Crazyradio USB dongle.

- **MCU/Gyro/Accelerometer**

  Crazyflie 1.0 embeds with powerful 32 bit MCU: STM32F103CB @ 72 MHz (128kb flash, 20kb RAM), 3-axis high-performance MEMs gyros with 3-axis accelerometer: Invensense MPU-6050.

- **Accessories**

  Crazyflie 1.0 has expansion header 2x10 pins 1.27mm pitch including power, I2C/UART, SPI/ADC. Header also contains ARM Cortex 10-pin JTAG (header not included). It also has 4-layer low noise PCB design with separate voltage regulators for digital and analog supply.

Crazyflie includes components are widely used and easy-to-get. The space-saving architecture such like part of the board as motor arm and battery laying on the

[1]Crazyfile 1.0 Manual Book is referred to http://www.bitcraze.se/crazyflie/

top of board is very impressive. In addition, the fairly small size of Crazyflie gains advantage of excellent impact durability.

3.3 Crazyflie Public Declaration

The Crazyflie platform is an open development platform consisting of open hardware and open source firmware/software. The reason for making it open is to allow users to be able to hack, modify, experiment and learn from our platform. The projects are developed using only open tools. Since the project it self is open so the users to have access to the tools needed for viewing, editing and using them.

There is a number of different licenses that Crazyflie project has to comply:

- All firmware is licensed **GPL**[1]**v3+**
- The PC applications and API are licensed **GPLv2+**
- All hardware design files are licensed **CC BY-NC-SA 3.0**[2]

Inspired by the open-source spirit, all the virtual tools used in the project will be based on this. Plus all the hardwares come from the daily life, are also easy-to-get and cheap in price.

[1]GPL: General Public License

[2]CC BY-NC-SA 3.0: Creative Commons Attribution-ShareAlike 3.0 License

4 Crazyflie Hardware Analysis

4.1 MCU: STM32F103CB

The STM32F103xx[1] medium-density performance line family incorporates the high-performance ARM Cortex^{TM}-M3 32-bit RISC core operating at a 72 MHz frequency, high-speed embedded memories (Flash memory up to 128 Kbytes and SRAM up to 20 Kbytes), and an extensive range of enhanced I/Os and peripherals connected to two APB buses. All devices offer two 12-bit ADCs, three general purpose 16-bit timers plus one PWM timer, as well as standard and advanced communication interfaces: up to two I2Cs and SPIs, three USARTs, an USB and a CAN.

The devices operate from a 2.0 to 3.6 V power supply. They are available in both

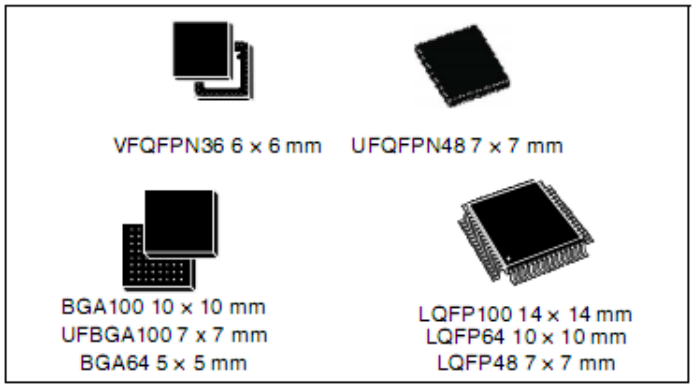


Figure 6: STM32F103xx Layout

the –40 to +85 C temperature range and the –40 to +105 C extended temperature range. A comprehensive set of power-saving mode allows the design of low-power applications.

The STM32F103xx medium-density performance line family includes devices in six different package types: from 36 pins to 100 pins. Depending on the device chosen, different sets of peripherals are included, the description below gives an overview of the complete range of peripherals proposed in this family.

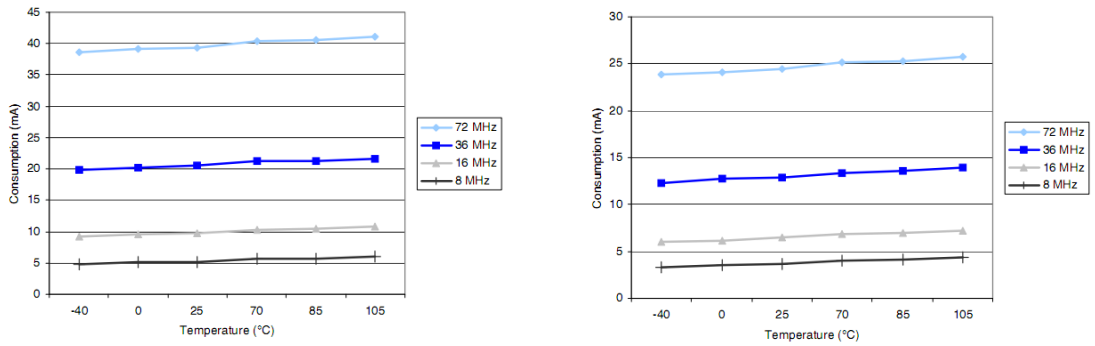
These features make the STM32F103xx medium-density performance line microcontroller family suitable for a wide range of applications such as motor drives, application control, medical and handheld equipment, PC and gaming peripherals, GPS

[1]STM32F103xx: the data information is referred to http://www.st.com/

platforms, industrial applications, PLCs, inverters, printers, scanners, alarm systems, video intercoms, and HVACs.

The RTC and the backup registers are supplied through a switch that takes power either on VDD supply when present or through the VBAT pin. The backup registers are ten 16-bit registers used to store 20 bytes of user application data when VDD power is not present.The real-time clock provides a set of continuously running counters which can be used with suitable software to provide a clock calendar function, and provides an alarm interrupt and a periodic interrupt. It is clocked by a 32.768 kHz external crystal, resonator or oscillator, the internal low-power RC oscillator or the high-speed external clock divided by 128. The internal low-power RC has a typical frequency of 40 kHz. The RTC can be calibrated using an external 512 Hz output to compensate for any natural crystal deviation. The RTC features a 32-bit programmable counter for long-term measurement using the Compare register to generate an alarm.

The current consumption, shown in Figure 7, is a function of several parameters and factors such as the operating voltage, ambient temperature, I/O pin loading, device software configuration, operating frequencies, I/O pin switching rate, program location in memory and executed binary code.



(a) Current Consumption Peripherals (b) Current Consumption Non-Peripherals

Figure 7: MCU Current Consumption

The low power consumption features sleep, stop and standby modes with 72 MHz high frequency plus RTC enchantment has made this MCU the perfect one for mini-UAV flight controller.

4.2 Gyro/Accelerometer: MPU-6050

The MPU-60X0[1] is the world's first integrated 6-axis MotionTracking device that combines a 3-axisgyroscope, 3-axis accelerometer, and a Digital Motion Processor™ (DMP) all in a small 4x4x0.9mmpackage. With its dedicated I2C sensor bus, it directly accepts inputs from an external 3-axis compass to provide a complete 9-axis MotionFusion™ output. The MPU-60X0 MotionTracking device, with its 6-axis integration, on-board MotionFusion™, and run-time calibration firmware, enables manufacturers to eliminate the costly and complex selection, qualification, and system level integration of discrete devices, guaranteeing optimal motion performance for consumers.

(a) MPU-6050 Actual

(b) MPU-6050 Orientation of Axes

Figure 8: MPU-6050

The MPU-60X0 consists of three independent vibratory MEMS rate gyroscopes, which detect rotation about the X-, Y-, and Z- Axes. When the gyros are rotated about any of the sense axes, the Coriolis Effect causes a vibration that is detected by a capacitive pickoff. The resulting signal is amplified, demodulated, and filtered to produce a voltage that is proportional to the angular rate. This voltage is digitized using individual on-chip 16-bit Analog-to-Digital Converters (ADCs) to sample each axis. The full-scale range of the gyro sensors may be digitally programmed to 250, 500, 1000, or 2000 degrees per second (dps). The ADC sample rate is programmable from 8,000 samples per second, down to 3.9 samples per second, and user-selectable low-pass filters enable a wide range of cut-off frequencies.

The MPU-60X0's 3-Axis accelerometer uses separate proof masses for each axis. Acceleration along a particular axis induces displacement on the corresponding proof

[1]MPU-60X0: the data information is referred to http://www.invensense.com/

mass, and capacitive sensors detect the displacement differentially. The MPU-60X0's architecture reduces the accelerometers' susceptibility to fabrication variations as well as to thermal drift. When the device is placed on a flat surface, it will measure 0g on the X- and Y-axes and +1g on the Z-axis. The accelerometers' scale factor is calibrated at the factory and is nominally independent of supply voltage. Each sensor has a dedicated sigma-delta ADC for providing digital outputs. The full scale range of the digital output can be adjusted to 2g, 4g, 8g, or 16g.

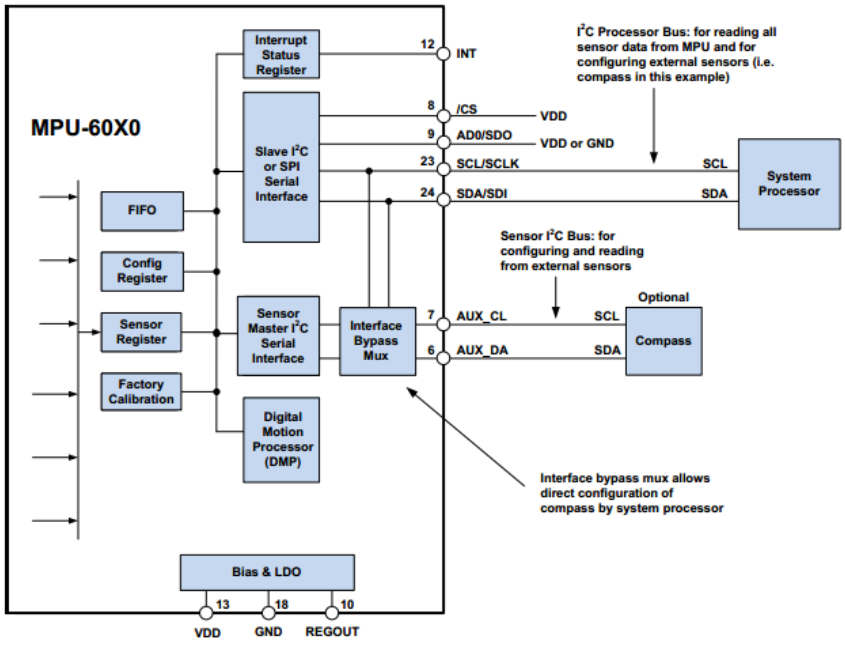


Figure 9: Sensor Fusion with I2C Interface

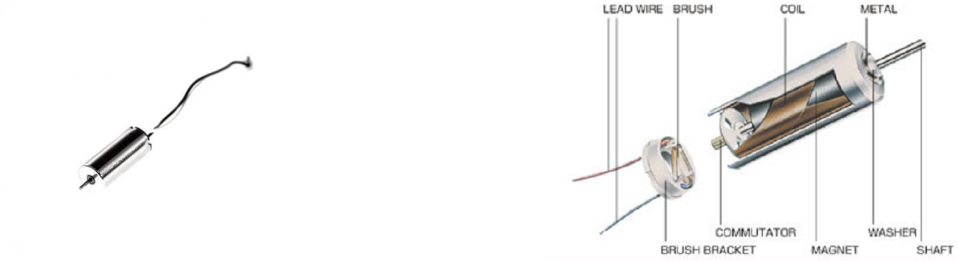
In Figure 9, the system processor is an I2C master to the MPU-60X0. In addition, the MPU-60X0 is an I2C master to the optional external compass sensor. The MPU-60X0 has limited capabilities as an I2C Master, and depends on the system processor to manage the initial configuration of any auxiliary sensors. The MPU-60X0 has an interface bypass multiplexer, which connects the system processor I2C bus pins 23and 24 (SDA and SCL) directly to the auxiliary sensor I2C bus pins 6 and 7.

4.3 Crazyflie Accessories

The Crazyflie accessories have been designed to fit the unique feature of this mini-UAV to perform mutl-tasks in regardless of its raw prototyping look. These accessories will be addressed in the following section.

4.3.1 Coreless DC motor (7x16 mm)

Coreless motor is a kind of DC motor with a rotor but without an iron core. Instead, it contains a permanent magnet inside and coil outside, shown in Figure 10.



(a) Coreless Motor Actual

(b) Coreless Motor Components

Figure 10: Coreless Motor

There is a few of advantages by eliminating the iron core from the motor, which includes it produces smaller rotational inertial thus rotate faster and it has smoothers rotation due to the absence of magnetic.

Diameter	7.0mm	Wire Length	32.0mm
Length	16.0mm	Rated Current	1000mA
Shaft Length	3.5mm	Rated Voltage	4.2V
Shaft Diameter	0.8mm	Kv	14000rpm/V
Weight	32.0mm		

Table 1: Motor Specification

By looking at the specification of this coreless motor, the Kv is as high as 14000rpm/V, so that when Crazyflie reaches it peak 3.7 V (Lipo Battery), the rotation can arrive at 51000rpm. This means the extremely rapid rotation shall produce sufficient lift to let Crazyflie maintain desired flight.

4.3.2 LiPo battery (240mAh)

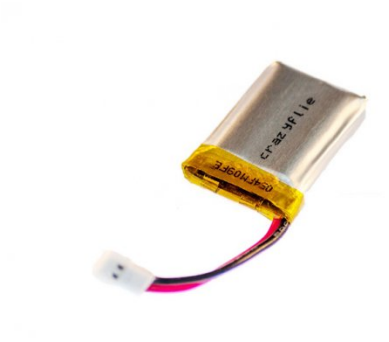


Figure 11: LiPo Battery

Operate Temperature	-20 - 60 C	Charge	2C
Capacity	240mAh	Weight	7.1g
Normal Voltage	3.7V	Cell Size	20x7x30mm
Discharge	15C		

Table 2: Lipo Battery Specification

4.3.3 CW+CCW propellers

Figure 12: Propellers

Blade Size	45mm
Fits Shaft	0.8mm

Table 3: Blade Specification

4.3.4 Crazyradio - nRF24LU1+ USB radio dongle (2.4Ghz)

The Crazyradio is the official radio dongle for the Crazyflie Nano Quadcopter. It is a 2.4GHz USB radio dongle based on the nRF24LU1+ chip from Nordic Semiconductor.

Figure 13: Radio Dongle

The design includes a 2x5 header (not mounted) where the dongle can be connected and powered from another source than USB. It also has two indicator LEDs.

The dongle is delivered with open source firmware and Python[1] drivers that enables easy use. The dongle can communicate with any project using a compatible Nordic Semiconductor radio chip. The firmware can be freely modified as it is open source and can be compiled with an open source compiler (sdcc).

Frequency	2.4Ghz
Output Power	1mW
Channels	125
Data Rate	2Mbps
Range	80m

Table 4: Radio Dongle Specification

A bootloader on the dongle enables firmware updates without using any external programmer. It can also be programmed over SPI, which is available in the header. This is recommended when developing for it.

[1]Python: A object-oriented programming language for API development

4.3.5 Controller - PS3 Controller

The Crazyflie development team has integrated a number of control methods including PS3 pad, Xbox pad and Android cellphone client. In this project, the controller used to manually command the Crazyflie is the common, easy-to-get PlayStation3 gameing pad, is the one shown in Figure 14.

Figure 14: PS3 Controller Sample
(www.playstations.com)

The way it works is that the controller's driver has been written into the client of Crazyflie. As shown in Figure 15, when controller is connected to the PC, and the Crazyflie itself is linked to the PC client through the radio dongle, then all the command made by the controller will sent to Craztflie by the radio dongle after filtering and processing through the API, and the detail information of the client interface will be provided in Chapter 4.

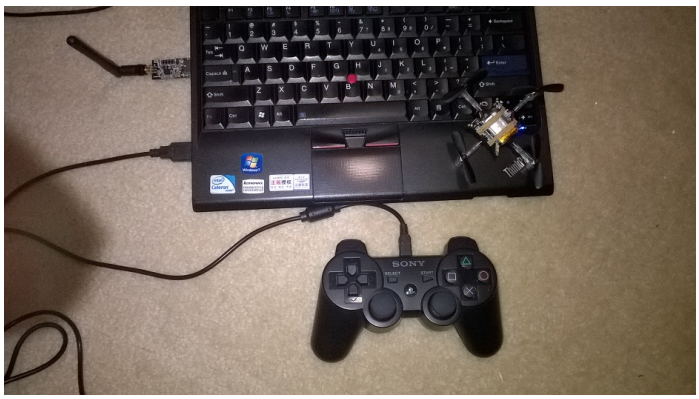


Figure 15: PS3 Controller in Setup

4.4 Kinect - Motion and Image Capture Device

The main interest as addressed at the beginning of this report is to test the feasibility of mirco-UAV being deployed and tracked by the carrier(i.e. Planetary Rover), plus safely returning back to it.

Initial assumptions were made, such as installing sonar or infrared device to tell the distance of Crazyflie away from carrier. Also, the barometer embedded on Crazyflie could activate the altitude-hold functionality once the reference point is made. However, the attitude of Crazyflie is difficult to accurately locate and control, and the multi-communication tasks may cause chaos in the development process.

Therefore, one reasonable solution is to use the motion and image process device. In detail, if such device is mounted on the carrier, it performs like a motion surveillance camera to tell the location of any moving target fitting inside the camera image, also means UAV's 2-D location is indicated. Meanwhile, the infrared camera on it(if applicable) will provide the distance from the target UAV to the camera, which means the third dimension information is secured. Hence, the 3-D all-in-one detecting device is preferred.

Figure 16: Kniect - Motion Sensing Device

Kinect, a motion sensing device, as shown in Figure 16, is developed by Micro-Soft to enhance the gameplay of its console Xbox. Kinect sensor is a horizontal bar connected to a small base with a motorized pivot and is designed to be positioned lengthwise above or below the video display. The device features an RGB camera, depth sensor and multi-array microphone running proprietary software which provide full-body 3D motion capture, facial recognition and voice recognition capabilities.

The depth sensor consists of an infrared laser projector combined with a monochrome CMOS sensor, which captures video data in 3D under any ambient light conditions. The sensing range of the depth sensor is adjustable, and Kinect software is capable of automatically calibrating the sensor based on gameplay and the player's physical environment, accommodating for the presence of furniture or other obstacles.

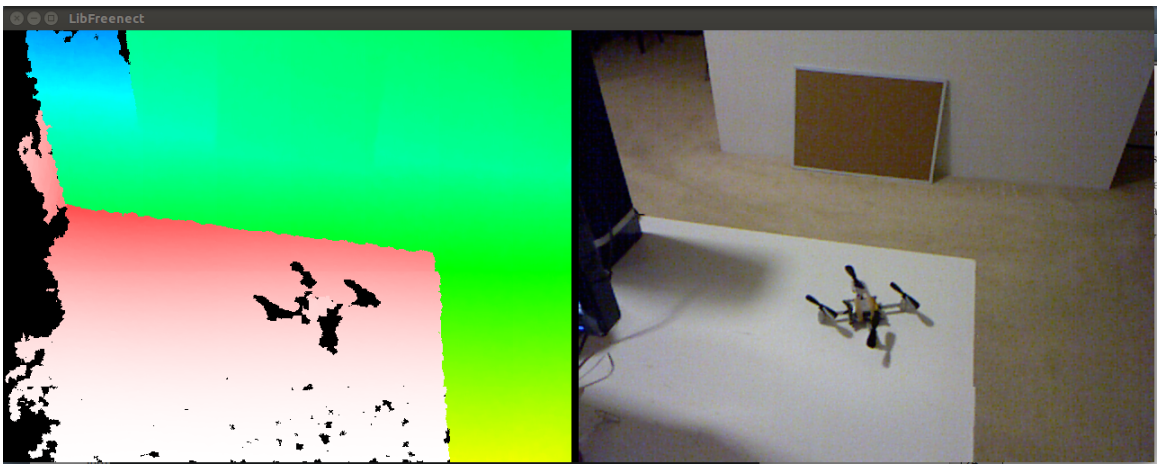


Figure 17: Kniect - Depth Sensor Example

In Figure 17, it shows how the depth sensor indicates the distance from the object to Kinect. The 'black' area is the blind spot, which means the object is too close to the sensor, and the distance is getting further by showing the color transformation from warm to cool hue. But, obviously Crazyflie is too small to be exactly captured in the depth sensor. Also, the narrow side-way of Crazyflie will be tracked during the flight, which makes the depth sensing even harder.

Hence, a fairly large and firm object is desired to be mounted on Crazyflie to act as an tracking object. The theoretical lift capacity of Crazyflie is 10-15 grams(despite the weight of Crazyflie itself is 19 grams, powerful like an ant), so this object must be within this weight limit. After a serie of tests, the best tracking object has been selected, which will be illustrated in section 3.5.

4.5 Tracking Target Design

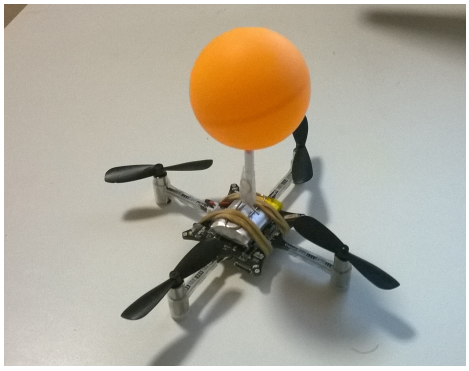
The primary intention is to select a round object. The benefit of this is during the tracking process, no matter how much Crazyflie yaws due to the feedback error, there is always a 'circle' appears on the image.

(a) Pingpong Ball

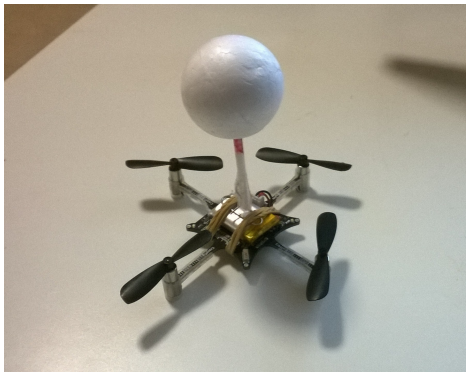
(b) Foam Ball

Figure 18: Tracking Target Selections

After research, the best options are either pingpong ball or foam ball, shown in Figure 18. Their natural light weights make them the suitable candidates. The first test is conducted on the pingpong ball, the standard weight of it is 2.7 grams. After installing it on Crazyflie with a 8cm long plastic stick and a rubber band, and doing a test flight, it indicates that during manoeuvres, the center of gravity of Crazyflie has been shifted so that it makes Crazyflie quit difficult to correct itself to steady flight once the attitude control input is released. Therefore, this indicates the hollow design of pingpong ball gives it downside such as unsteady center of gravity during flight.



(a) Pingpong Ball



(b) Foam Ball

Figure 19: Crazyflie with Tracking Balls

The second test is with the foam ball, the full-filling round foam with a weight of only 0.5 gram makes it advantageous over the pingpong ball. The setup is shown in Figure 19, after a few flight tests, the shift of center of gravity can barely be detected.

Hence, the extremly light foam ball has been selected and enter the tracking test, which will be specifically explained in the following chapters.

5 Crazyflie Software Analysis

The software setup illustrated in this chapter is accomplished on the Linux based operating system, Ubuntu[1], a version of 14.04 LTS.

An upside of using this system other than Windows is due to fact that when connecting any hardware to the PC, Windows system requires 'driver' to run this hardware, this means software is priority. Instead, Linux system identifies the hardware as a ready-to-go device, in most cases, 'driver' is not required. This is quite beneficial for hardware still undergoing developing and unstable phase, so it is suitable for this project.

5.1 Crazyflie GUI Client

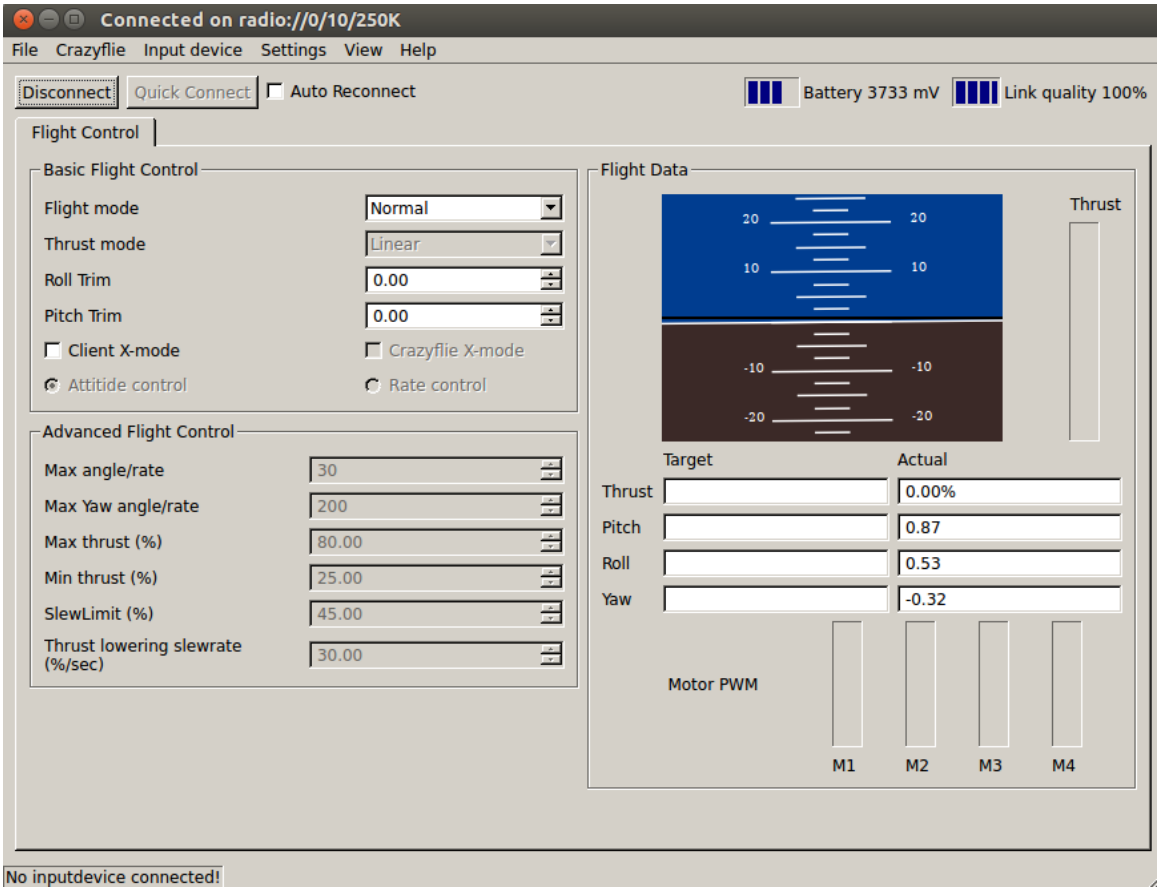


Figure 20: Crazyflie Client Interface

[1]Ubutu, an open source Linux operating system, http://www.ubuntu.com/

The Crazyflie official supplies the GUI client to add the user's flying experience, as shown in Figure 20. The purpose of this GUI is to control/log/bootload[1] the Crazyflie. User needs to download this client from the development hub[2].

5.1.1 Installation

After downloading and extracting the client, the following Python and USB supporting packages need to be installed for the PC ultilities to works, these include: Python2.7, pygame, PyUSB, liusb and Python bingdings for Qt4. The command line is:

```
>> sudo apt-get install python2.7 python-usb python-pygame python-qt4
```

After the package installations are done, then the client can be opened by:

```
>> cd <cfclient folder>
>> sudo python2 bin/cfclient
```

5.1.2 Setup

1. Start up the GUI application.
2. Insert the controller and raido dongle to pc.
3. Before pressing 'Connect' make sure that the controller is working as expected and that the thrust is zero. The joystick values should be visible in the flight data box under target.
4. Press 'Connect'.
5. Wait for the radio scanning to complete.
6. Double-click the Crazyflie. There's one debug URI for UI testing so you should connect to the URI that begins with radio://XX/XX/XX.
7. The client will now connect to the Crazyflie and handshake.
8. Once the handshake is done, it is ready to start flying the Crazyflie.

[1]bootload: To flash the firmware to Crazyflie through radio dongle.

[2]hub: https://bitbucket.org/bitcraze/crazyflie-pc-client/downloads/cfclient-2013.4.2.tar.gz

5.1.3 Features

- Flight Control: By selecting Advanced in the Flight mode drop down you will be able to set the Advanced Flight Control settings. When changed they take effect immediately.
- Parameters: This tab allows you to view and edit the parameters in the Crazyflie. Edited values will not be sent to the Crazyflie until the Update values button is clicked.
- Log TOC: Shows the table of contents for the loggable variables.
- Console: Shows printouts from the Crazyflie.
- Bootloader: For updating the Crazyflie firmware there's the possibility to enter bootloader mode and flash new firmware from within the client. The bootloader mode is accessed from the menu Bootloader.

In addition, it is possible to set another channel to communicate with the Crazyflie. It can be wise to do this if there exist other wireless networks that can interfere, especially WiFi. It is also possible to permanently store the trim values for pitch and roll. To edit the configuration block you will have to enter bootloader mode as described in Bootloader above.

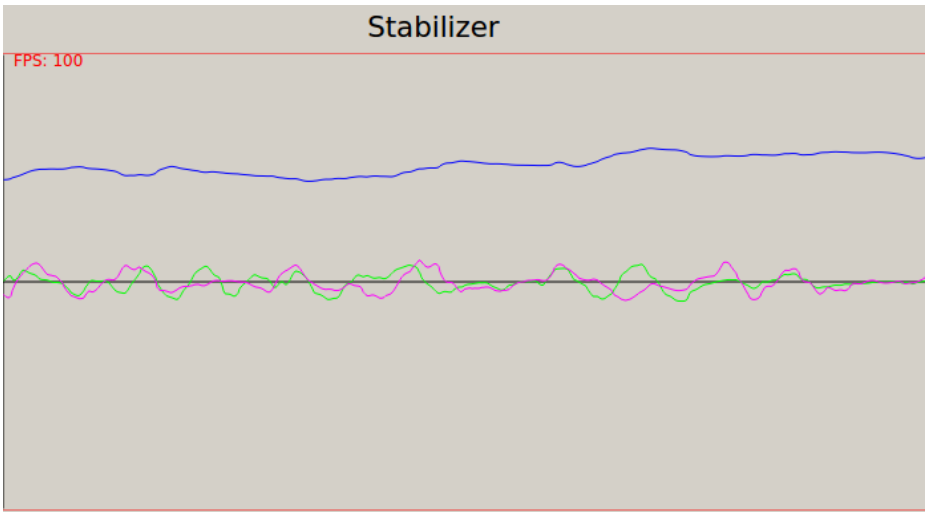


Figure 21: Flight Attitude Log

In Figure 21, it is an example of flight attitude log during steady level flight.

5.2 Crazyflie Firmware

5.2.1 Crazyflie Firmware

The Crazyflie's firmware can be found in the open source git hub[1], and it simply has following packages, and is for familiarizing the coding strategy, deep study is not required.

```
Folder description:
./              | Root, contains the Makefile
 + init         | Contains the main.c
 + config       | Configuration files
 + drivers      | Hardware driver layer
 |  + src       | Drivers source code
 |  + interface | Drivers header files. Interface to the HAL layer
 + hal          | Hardware abstaction layer
 |  + src       | HAL source code
 |  + interface | HAL header files. Interface with the other parts of
 the program
 + modules      | Firmware operating code and headers
 |  + src       | Firmware tasks source code and main.c
 |  + interface | Operating headers. Configure the firmware environment
 + utils        | Utils code. Implement utility block like the console.
 |  + src       | Utils source code
 |  + interface | Utils header files. Interface with the other parts of
 the program
 + scripts      | Misc. scripts for LD, OpenOCD, make, version control, ...
 |              | *** The two following folders contains the unmodified
 files ***
 + lib          | Libraries
 |  + FreeRTOS  | Source FreeRTOS folder. Cleaned up from the useless files
 |  + STM32F... | Library folder of the St STM32 peripheral lib
 |  + CMSIS     | Core abstraction layer
```

[1]hub: https://github.com/bitcraze/crazyflie-firmware

5.2.2 Crazyfile Radio Firmware

Crazyflie Radio is able to flash the firmware of Crazyflie and also modify the channel informations. The firmware of Crazy Radio can be found in git hub[1].

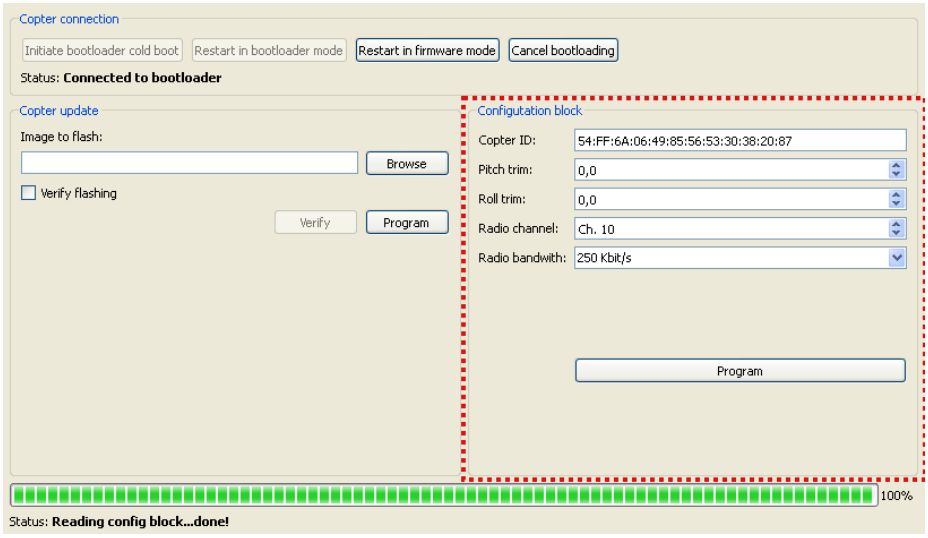


Figure 22: Radio Bootloader and Channel

To upgrade the Radio firmware:

```
>> cd crazyradio-firmware
>> sudo python usbtools/launchBootloader.py
Launch bootloader .
Bootloader started

>> cd crazyradio-firmware
>> sudo python usbtools/nrfbootload.py flash cradio-0.52.bin
```

Also, in Figure 22, it is possible to set another channel to communicate with the Crazyflie. It can be wise to do this if there exist other wireless networks that can interfere, especially WiFi. It is also possible to permanently store the trim values for pitch and roll.

[1]hub: https://bitbucket.org/bitcraze/crazyradio-firmware/downloads/cradio-0.52.bin

5.3 Open-Kinect Firmware

OpenKinect is an open community of people interested in making use of the amazing Xbox Kinect hardware with our PCs and other devices. The community is currently working on free, open source libraries that will enable the Kinect to be used with Windows, Linux, and Mac.

In this project, the tracking mission will be achieved by linking the open-kinect source libraries to the Crazyflie client so that the kinect will take control of the flight. To manually install with Ubuntu command line, it looks like this:

```
>> sudo apt-get install git-core cmake freeglut3-dev ...
   pkg-config build-essential libxmu-dev libxi-dev libusb-1.0-0-dev
>> sudo git clone git://github.com/OpenKinect/libfreenect.git
>> cd libfreenect
>> mkdir build
>> cd build
>> cmake ..
>> make
>> sudo make install
>> sudo ldconfig /usr/local/lib64/
```

To open the freenect client, insert command line:

```
>>  sudo freenect-glview
```

Figure 23: Openkinect Setup - Ball Clearly Identified

6 Image Processing Technique

The purpose of studying the image processing technique is to implement the relevant programming functions into the control system in order to identify and track the designated target. In this project, the main focus is on studying and employing OpenCV techniques[1].

6.1 OpenCV Review

OpenCV is an open source, cross-platform library that provides building blocks for comupter vision experiments and applications. It provides high-level interfaces for capturing, process, and presenting image data, i.e. abstracting details about camera hardware and array allocations.

In order to comply with the Crazyflie API's Python background, OpenCV's Python bindings is employed to explore solutions to requirements in a high-level language and in a standardized data formate that is interoperable with scientific libraries such as NumPy and SciPy. Thus, to support the functionality of Kinect, the follwing libraries need to be installed.

- **NumPy**: A dependency of OpenCV's Python bindings. It provides numeric computing functionality, including efficient arrays.
- **SciPy**: A scientific computing library thatis closely related to NumPy. It is nor required by OpenCV but it is useful for manipulating the data in OpenCV images.
- **OpenNI**: An optional dependency of OpenCV. It adds support for certain depth camera.
- **SensorKinect**: An OpenNI plugin and optional dependency of OpenCV. It adds support for Kinect depth camera.

[1]OpenCV: A library of programming functions mainly aimed at real-time computer vision, developed by Intel Russia research center.

6.2 OpenCV Setup

To install OpenCV 2.4.9 on Ubuntu 14.04 operating system, install development environment to build OpenCV:

```
>> sudo apt-get install build-essential cmake pkg-config
```

Install Image I/O libraries:

```
>> sudo apt-get install libjpeg62-dev
>> sudo apt-get install libtiff4-dev libjasper-dev
```

Install the GTK dev library:

```
>> sudo apt-get install  libgtk2.0-dev
```

Install Video I/O libraries:

```
>> sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev
```

Install the Python development environment and the Python Numerical library:

```
>> sudo apt-get install python-dev python-numpy
```

Install the Qt dev library:

```
>> sudo apt-get install libqt4-dev
```

Download OpenCV 2.4.*:

```
>> mkdir xxx
>> cd xxx
>> sudo wget http://sourceforge.net/projects/opencvlibrary/files/opencv-unix/2.4.9
>> tar -xvf OpenCV-2.4.*.tar.bz2
```

Create and build directory and configure OpenCV with cmake:

```
>> cd OpenCV-2.4.*
>> mkdir build
>> cd build
>> cmake -D CMAKE_BUILD_TYPE=RELEASE -D CMAKE_INSTALL_PREFIX=/usr/local
    -D WITH_TBB=ON -D BUILD_NEW_PYTHON_SUPPORT=ON -D WITH_V4L=ON
    -D INSTALL_C_EXAMPLES=ON -D INSTALL_PYTHON_EXAMPLES=ON
    -D BUILD_EXAMPLES=ON -D WITH_QT=ON -D WITH_OPENGL=ON ..
```

Compile and finalize installation:

```
>> sudo make
>> sudo make install
```

6.3 OpenCV Examples

OpenCV provides the ability to identify the shape of the object, i.e. the red circle marking the foam ball in Figure 24. This is simply a demo of showing the capability of OpenCV locating the target. Later in the tracking section, the red circle will be dynamically locating the crazyflie in real-time flight.

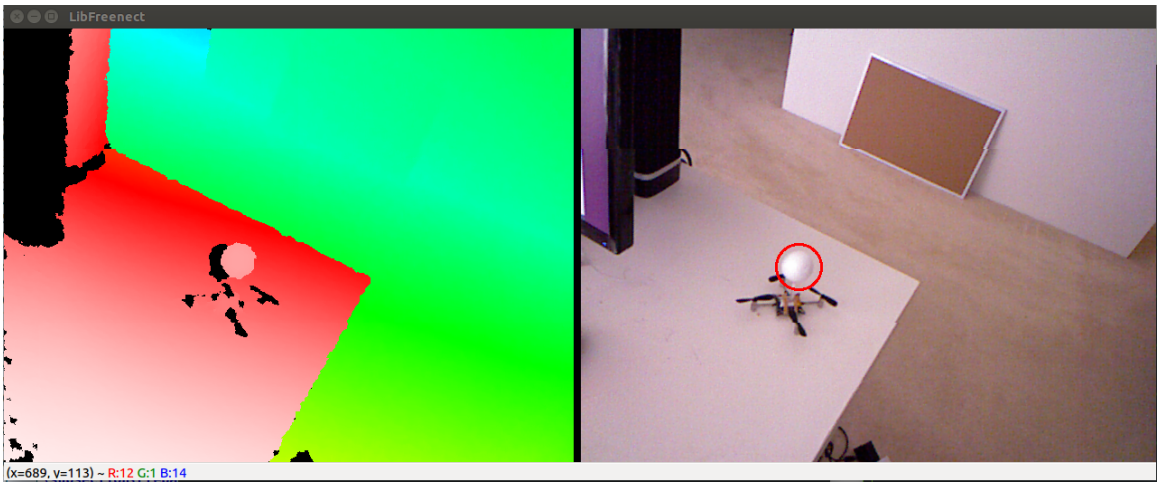


Figure 24: Crazyflie Marked by Red Circle

The simply 'red circle' can be written as:

```
import numpy as np
import cv2

# Load 'openkinect' image
img = cv2.imread('openkinect.png',1)
# Create circle with desired loaction(x,y), radius, color,thickness
cv2.circle(img,(880,290), 25, (0,0,255),2)
# Show the image
cv2.imshow('Kinect-Crazyflie',img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

7 Auto-Tracking

The objective of this part is to achieve the auto-flight without any manually control input or interference. One of the big challenges is to set up a less disturbed and enclosed area to conduct the flight test. Hence, the project takes place in a bed room like environment with a white background to minimize the color sensitivity.

The auto-tracking control software frame has been developed by Crazyflie team, and it's free to use and modify. Therefore, beside conducting the auto-tracking test, studying the frame and PID technique are also concerned.

7.1 Preparation

7.1.1 Red Target

The foam ball has been painted into 'Red' color, shown in Figure 25. This is caused by if any dark scheme appears in the track image, then the track circle will start to track the cool color target ball. So, painting the ball into red is a suitable option.

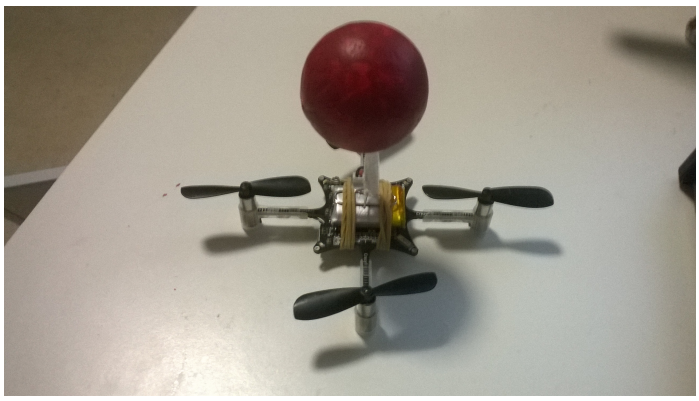
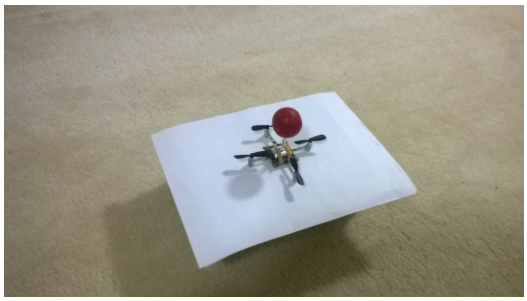
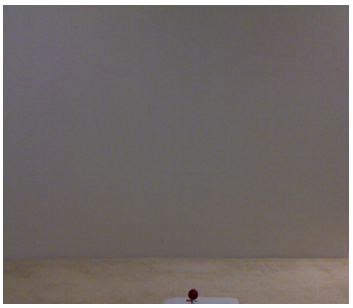


Figure 25: Red Target

Crazyflie is going to automatically takeoff and perform hovering. Hence, Crazyflie's red foam ball is expected to appear at the bottom of the tracking image, as shown in Figure 26. Once Crazyflie lifts off ground, red ball will start to approach the center of the image, as planned. During the process, the image processing tool will keep sending the location information to the control software in order to notify the flight command to adjust thrust setting to reach the designated location.



(a) Tracking Platform



(b) Tracking Background

Figure 26: Tracking Setup

7.1.2 Control Frame

The source code of the control frame can be found in "/crazyflie/lib/kinect" package. The control logics starts with define the track image (640x480) and enable the mouse control:

```
IMG_WIDTH = 640
IMG_HEIGHT = 480
cv.NamedWindow('LittleBlack Auto-Tracking', cv.CV_WINDOW_AUTOSIZE)
cv.SetMouseCallback('LittleBlack Auto-Tracking', self.mouse_ev)
```

Then, define the center tracking point 'cp' and create the circle in 'blue':

```
cv.Circle(img, (cp[0], cp[1]) , 10, cv.CV_RGB(0, 0, 255), 1)
```

Also, get the depth working, which is finding the distance from camera to Crazyflie :

```
img_th = cv.CreateImage(cv.GetSize(depth_image), 8, 1);
cv.InRangeS(depth_image, 10, 210, img_th);
depth = cv.Avg(depth_image, img_th)[0]
```

The position function can be defined in three dimensions:

```
depth_img = pretty_depth_cv(kinect_depth)
position = self._get_pos(self.img)
depth = self._get_depth(depth_img, debug=False)
```

Thus, the position of Crazyflie can be tracked by the dynamic position 'sp' representing in 'green' circle:

```
cv.Circle(self.img, (self.sp[0], self.sp[1]) , 10, cv.CV_RGB(0, 255, 0), 1)
```

7.2 PID Technique and Implementation

As the most automated control system, the PID controller need to be tuned to optimize the flight. The PID controller for Crazyflie auto-tracking has the following methodology:

The PID has following default values:

```
P=1.0, I=0.0, D=10.0
```

The set point of Crazylie will be corrected by:

```
Kp = P; Ki = I; Kd = D;

error = set_point - current_value

P_value = Kp * error
        if (self.last_value >= current_value):
            change = error - last_error
        else:
            change = 0.0

        if error > 0.0:
            I_value = Integrator * Ki
        else:
            I_value = (Integrator * Ki)

D_value = Kd * change
Derivator = error
Integrator = Integrator + error/200.0 # 200 is the max value

PID = P_value + I_value + D_value
```

These are general form of PID calculations, and the values in the test has the example like:

```
r_pid = PID_RP(P=0.05, D=1.0, I=0.00025, set_point=0.0)
p_pid = PID_RP(P=0.1, D=1.0, I=0.00025, set_point=0.0)
t_pid = PID(P=30.0, D=500.0, I=40.0, set_point=0.0)
```

Thus, in the real test, these P, I and D values will be tuned to smooth the flight in roll, pitch and thrust.

7.3 Stability Test

To begin the final stage of the auto-tracking, selecting a fine location of platform is crucial. The effective tracking range is from 0.3 to 1.5 meters. After setting up the Crazyflie on the platform, perform the auto-tracking by typing command:

```
>> sudo python2 ~/cfckinect
```

To be noted, the yaw correction is not applicable because the yaw behaviour can not be detected at this stage. Therefore, the yaw change must always be zero, and the accelerometer will record the zero yaw attitude and correct it if it changes.

The initial PID setting to correct roll, pitch and thrustwill be like these:

```
r_pid = PID(P=0.05, D=1.0, I=0.00025, set_point=0.0)
p_pid = PID(P=0.1, D=1.0, I=0.00025, set_point=0.0)
t_pid = PID(P=30.0, D=500.0, I=40.0, set_point=0.0)
```

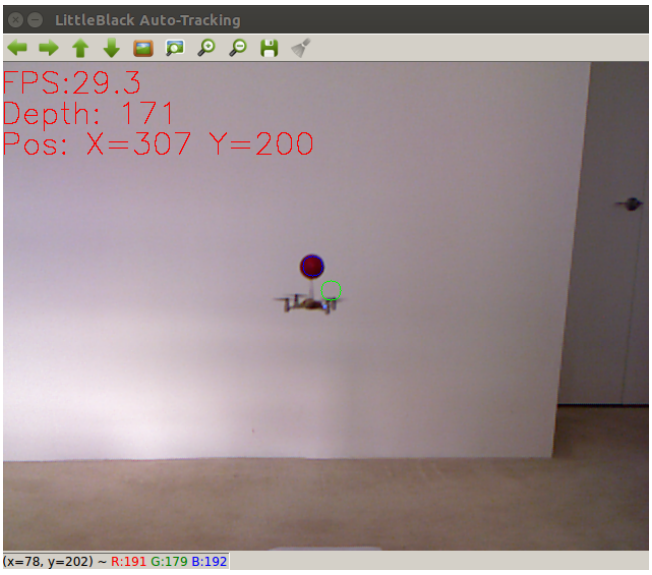


Figure 27: Auto-Tracking #1

As observed in Figure 27, the frame rate of Kinect keeps floating around 30 fps, which satisfies the tracking purposes. The 'Depth' and 'Pos' indicates the position (X,Y,Z) of Crazyflie inside the tracking image.

As explained before, the 'green' circle is the target of the ball and 'blue' one is the current position of it. The depth or distance from camera to Crazyflie will be located by the radius of the circle, which means Crazyflie will keep correcting its depth until the red foam ball fits into the blue circle, both circle in the image shares same radius.

Also, it can be notices that in Figure 27, Crazyflie can almost reach the 'green' circle, where the center of the image is. This small gap is caused by crazyflie over correcting itself when it's approaching the center. Hence, lower the 'P' value by 20% in the PID at a time to minimize the gain until it achieves better result, and the new PID setting looks like:

```
r_pid = PID(P=0.03, D=1.0, I=0.00025, set_point=0.0)
p_pid = PID(P=0.6, D=1.0, I=0.00025, set_point=0.0)
t_pid = PID(P=24.0, D=500.0, I=40.0, set_point=0.0)
```

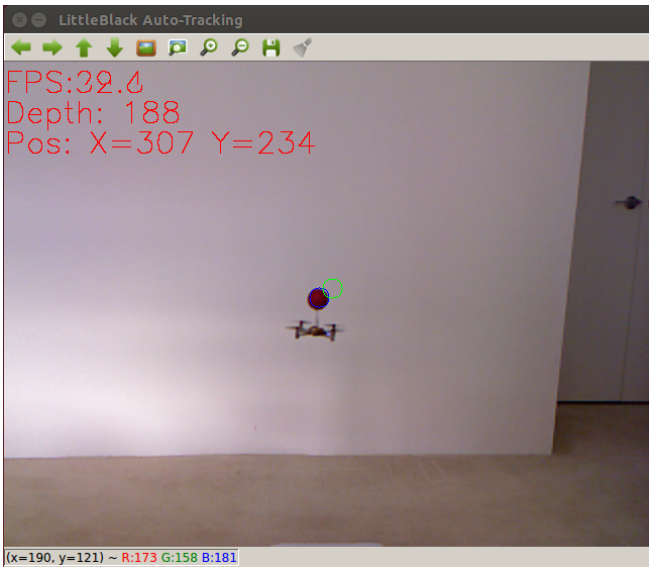


Figure 28: Auto-Tracking #2

After a serie of tests, a better result is obtained, as shown in Figure 28, the circles are touching each other, and by far this is the most steady and closest one.

In addition, the stability can be shown with the log ability of the stabilizer, where 'blue' is roll, 'green' is pitch and 'red' is yaw (not concerned in the tracking).

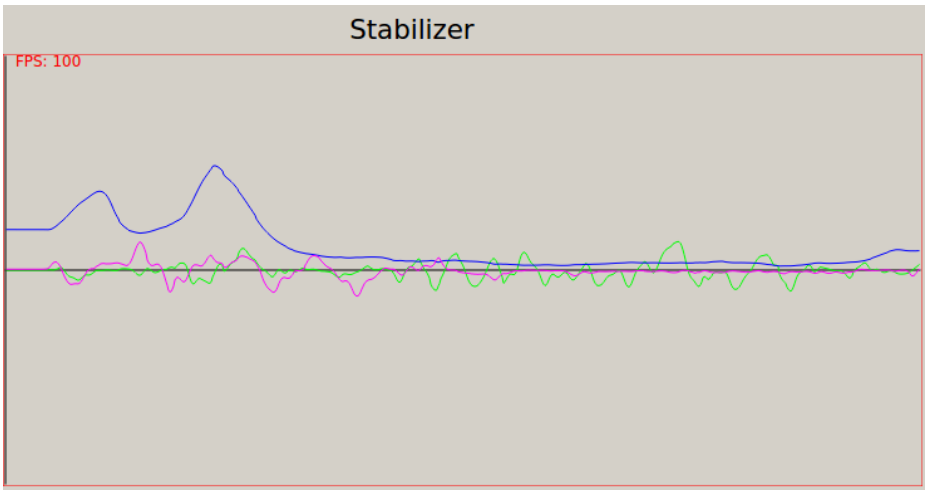


Figure 29: Auto-Tracking Stabilizer #1

The magnitude of the attitude variation is in degree. As expected, in the Figure 29, the #1 test, the attitude vibrating very much due to over-correcting.

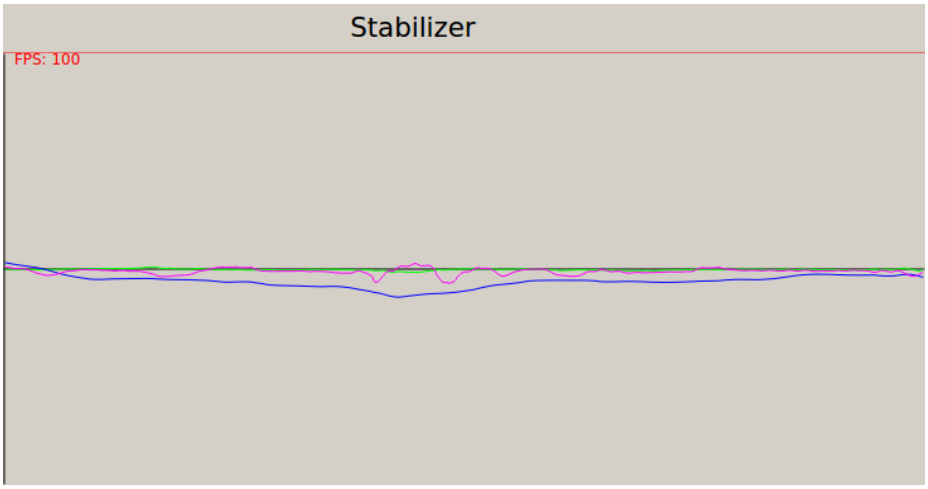


Figure 30: Auto-Tracking Stabilizer #2

After adjusting the PID setting for a number of tests, the #2 produces the desired result, as shown in Figure 30, the vibration magnitude has been much reduced, and steady flight has been achieved.

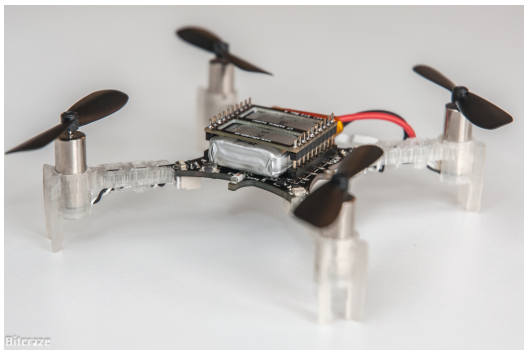
8 Conclusion and Future Work

8.1 Conclusion

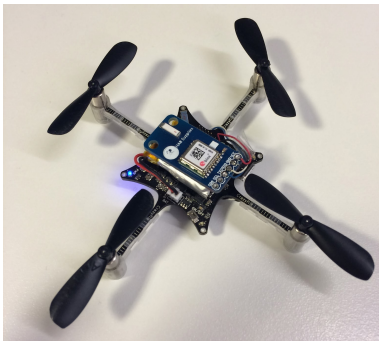
Firstly, the Crazyflie platform is still undergoing the development, so most of time the tests are not successful. A number of location adjustments have to be done to perform takeoff normally. When the tracking started after takeoff, red ball tends to track towards the camera and eventually crashes on it due to the lighting issue and paining qualities.

Secondly, the firmware, software and clients require a plenty of debugs to compile succesfully, plus the tuning of auto-tracking, which is quite time consuming process. However, the manually flight experience by using the control client is satisfying.

Finally, the Crazyflie project is a fun and exciting process, especially from opening the kit and manually assembling it to successfully seeing the first auto flight. Beside the auto-tracking project, there is still a lot of hacks and modifications can be done on Crazyflie.



(a) Crazyflie 2.0 Prototype



(b) Crazyflie with GPS module

Figure 31: Crazyflie's Future

In Figure 31, it shows the new generation of Crazyflie, 2.0. It brings a lot of new features which will further enhance the user's development experience, and also the Crazyflie with GPS module to test out the Crazyfle's capability of doing auto path travelling in regardless of the size and battery life.

8.2 Future Work

8.2.1 Trajectory Planning

Once the Crazyflie can achieve steady tracking, next line of the work is to do trajectory planning, such that it can automatically draw a circle in 2-d phase or even with the depth trajectory integrated into it's path. This requires the implementation of path algorithm and a numbers of experiments.

8.2.2 Device Implementation

In order to perform the recon , technically Crazyflie has to carry additional devices to measure the ambient conditions around the rover, such as gas content, humidity and pressure etc. The power to weight ratio and battery life are the issues. Future work should include improving the power out and battery life of Crazyflie without adding extra weight.

8.2.3 Formation Flight

After the solo tracking technology of Crazylie is mature, then it will be the time to proceed to implement multiple Crazyflies at the same time to test out the concept of formation flight or 'swarm', as shown in Figure 32. In this case, each of these Crazyflie could carry a specific module different to others to serve multiple purposes.

Figure 32: Micro-quadcopter 'swarm'

(www.unmanned.co.uk)

However, these formation flight currently can only be well done by using multiple motion cameras in an enclose area. If transfer it to outdoor scenario, the tracking and communication of multiple Crazyflies will be very challenging.

8.2.4 Shape Recognition

If Crazyflie is carrying module on top of it, the tracking target, i.e foam ball, appears to be distracting. A solution to this is to find out a way to let camera recognize the shape of Crazyflie instead of the foreign object implemented on it.

9 Reference

[1]: Carrillo,L. Lopez,A. Lozano,R. Pegard,C. 2013, "Quad Rotorcraft Control - Vision-Based Hovering and Navigation", p-35

[2]: Chao,H. Chen,Y. 2012, "Remote Sensing and Actuation Using Unmanned Vehicles", p-54

[3]: Guerrero,J. 2012, "Flight Formation Control", p-11

[4]: Jeschke,S. Liu,H. Shilberg,D. 2011 "Intelligent Robotics and Application"