Final Year Project Document

Project Name: Self-flying drone

By Jinwei Yuan

Student Number: 12170438

Supervisor: Petronel Bigioi

Course: Computer Science and Information Technology

NUIG

Statement of Originality

I hereby declare that this report is my original work except where stated.

Signature: ___________________________
Name: ___________________________
Date: ___________________________

Acknowledgements

I would like to
thank my supervisor Petronel Bigioi for the chance and advises.
thank Istvan Andorko for the great continuous support .
thank Fotonation for technique and other support.
This project is not able to be done without their help.

Contents

1 Introduction

1.1 Overview & Objectives

This project is aimed to build an application that is able to control a drone remotely, stream video using a wifi connection and process images in real-time from both main camera and bottom camera.

The drone control is implemented using dedicated UDP commands, and which follow the specifications of the ARDrone Developer Guide.

The drone will detect human faces and will identify its owner. If the owner is not identified, the drone will try to avoid interacting with a stranger. Once the owner is identified, the drone will follow the owner around a room, while in the same time keeping a safe distance.

The structure of the report is the following. Chapter 2 will describe software design aspects of the proposed drone control application. Chapter 3 will describe the implementation of the application while providing details about the library interaction, data flow and image analysis. Chapter 4 will describe the tests carried out for both the software and drone safety aspects point of view. Chapter 5 presents future work ideas, while conclusions are drawn in Chapter 6 of the report.

1.2 Literature Review

The current section presents a literature review regarding different technologies which were used during the development of the proposed application.

1.2.1 Socket (UDP, TCP)

User Datagram Socket: the UDP datagram is sent from one side without a connection to the receiving side. Destination IP and destination port and length of UDP should be known. No response will be returned by the receiving side whether it receives the UDP datagram or not. It is connectionless and unreliable [8] [9].
Figure 1 shows the basic idea about how UCP works. [28]

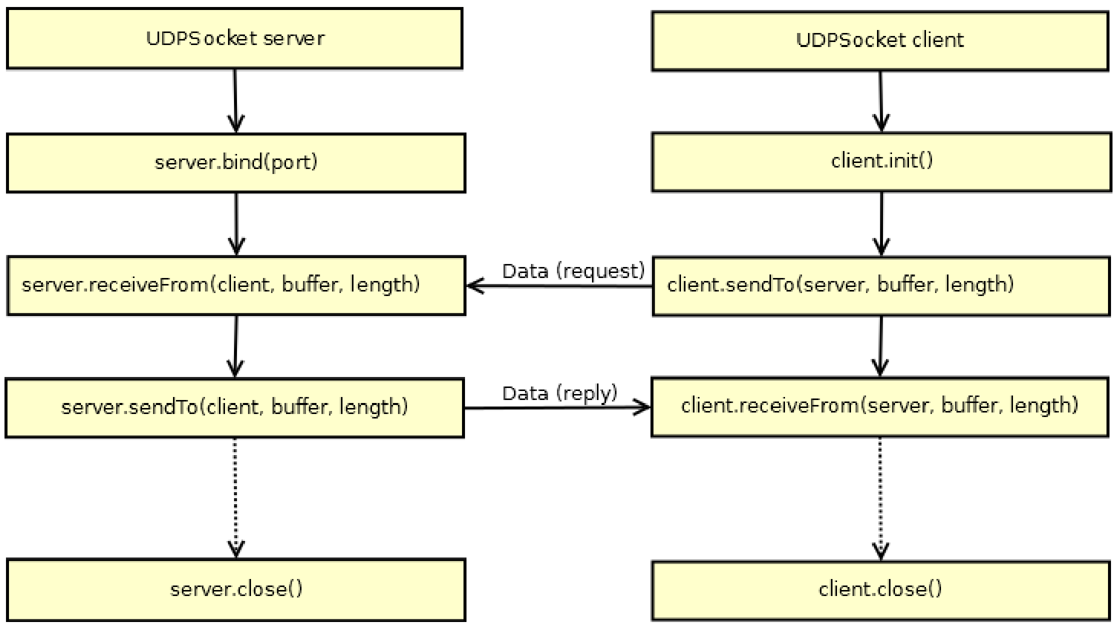

UDPSocket server
server.bind(port)
server.receiveFrom(client, buffer, length)
server.sendTo(client, buffer, length)
server.close()
UDPSocket client
client.init()
client.sendTo(server, buffer, length)
client.receiveFrom(server, buffer, length)
client.close()
Data (request)
Data (reply)

Figure 1. An idea about how UDP works

Transmission Control Protocol: TCP provides reliable data transmission. One device needs to connect with the other device before data transmission. In order to receive data correctly, the receiving side responses an acknowledgement number to the sending side when a packet is received successfully. Sequence number is used to determine whether a packet is duplicated or not. [10][11]
Figure 2 shows the basic idea about how TCP works. [28]

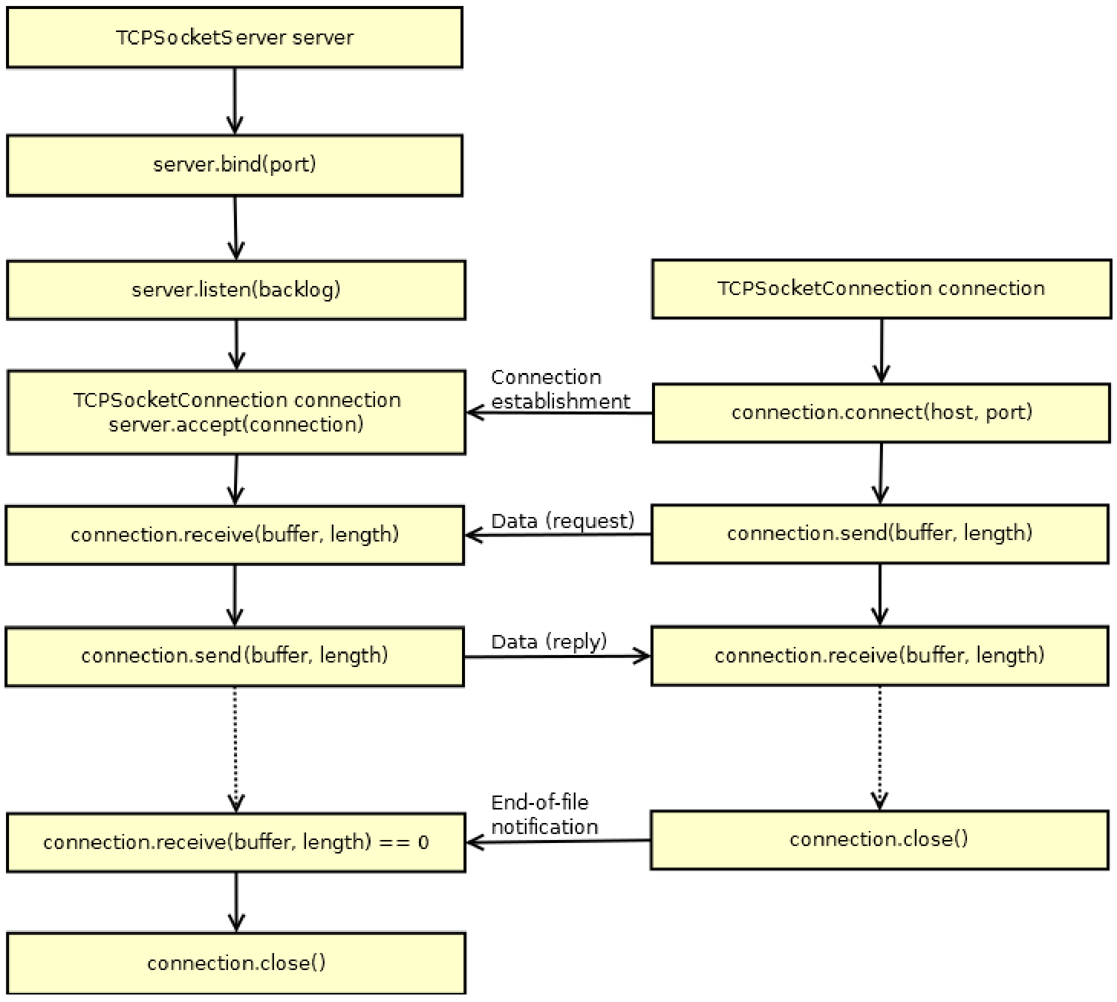

TCPSocketServer server
server.bind(port)
server.listen(backlog)
TCPSocketConnection connection
server.accept(connection)
connection.receive(buffer, length)
connection.send(buffer, length)
connection.receive(buffer, length) == 0
connection.close()
TCPSocketConnection connection
connection.connect(host, port)
connection.send(buffer, length)
connection.receive(buffer, length)
connection.close()
Connection establishment
Data (request)
Data (reply)
End-of-file notification

Figure 2. Idea about how TCP works

Both TCP and UDP are part of Internet Protocol Suite and they are on the transport layer.

1.2.2 Video streaming

Traditionally, real time video streaming is UDP based (including UDP, RTP, RTCP and etc.). UDP is fast because it is response-less. But the drone uses TCP to do real time video streaming, I did a search and found Real Time Streaming Protocol (RTSP) can do real time video streaming via both TCP and UDP. RTSP is in application layer of Internet protocol suite. RTSP uses TCP in most cases. Video stream, connection and control are available in RTSP. But I think UDP has better user experience if Net Speed is not very stable and fast. [12] [22]

1.2.3 Using third party libraries (both static and dynamic)

Using static library: when header files are included and libraries (.lib) are linked, the built program will have the codes that are used from the libraries. For coding, it is easy but the target program can be large. [13][14]
Using dynamic: comparing with static library, header files are not necessary, only

Dynamic-Link Libraries are need. For coding, “LoadLibrary” and “GetProcAddress” are needed in order to use the functions from DLLs but the built program is smaller compare to the static one. When running the program, DLL is need as well. [13][15]

1.2.4 Face Detection and Face Recognition

Before face recognition, face is needed to be detected first. There are ways to detect faces by colour and motion. For colour, distinguish face colour and from background colour. For motion, faces and eyes often moves. [16][17]
Edges and “weak-classifier cascades” are good and able to be used when above methods are not available. [17][18]
For face recognition, Principle Component Analysis (Using Eigen-face) is a method for recognition. The brief idea is:
Suppose a matrix F stands for all pixels training face samples (Greyscale), each row vector is a face.
Calculate the covariance matrix of F, which is a square (n*n).
Calculate the Eigen values and Eigen vectors. This can form Eigen faces.
Then use a sample face, average face and an Eigen face to calculate a weight. After calculating weights for all Eigen faces, a vector is formed with these weights.
Do the same thing for a new face, get another vector.
Finally, calculate the Euler distance between these two vectors. For more details, check [19][20][21]

2 Design

The current section presents three aspects about the project design which includes the UI definition & User Experience, Analysis of possible designs, and Final Design.

2.1 Define UI& User Experience

The drone is controlled using buttons in a dedicated app. The app streams video though main (front) camera and bottom camera. To make the drone fly easily, I want the drone to follow recognised people, for example, the drone recognises me and if I move away from the drone, the drone will fly towards until a safe distance which is calculated by the application.

.

2.2 Analysis of possible designs

The initial idea was to develop an iOS app and to control the drone from an iPhone or iPad. This approach was risky due to unavailability of development tools and libraries (ARDroneLib and ControlEngine from ARDrone SDK 201).
My backup plan was to use a windows environment on a PC and control the drone using image information from its cameras.

2.3 Final Design

Initially I tried to develop an iOS-based application. I found that the neither ARDroneLib nor ControlEngine(Xcode Project) is unavailable to use and build with the current XCode. I decided not to use the libraries from ARDrone SDK 201 but I could not make the UDP work correctly on iOS. In order to make sure that I finish the project on time, I had to switch to windows environment and use C to implement.
Figure 3, 4 and 5 are examples of proposed UI design. Figure 3 shows that the images when the drone is on the ground. Figure 4 shows the image when the drone takes off and hover. Figure 5 shows that the drone flies towards an object (a Pringles can).

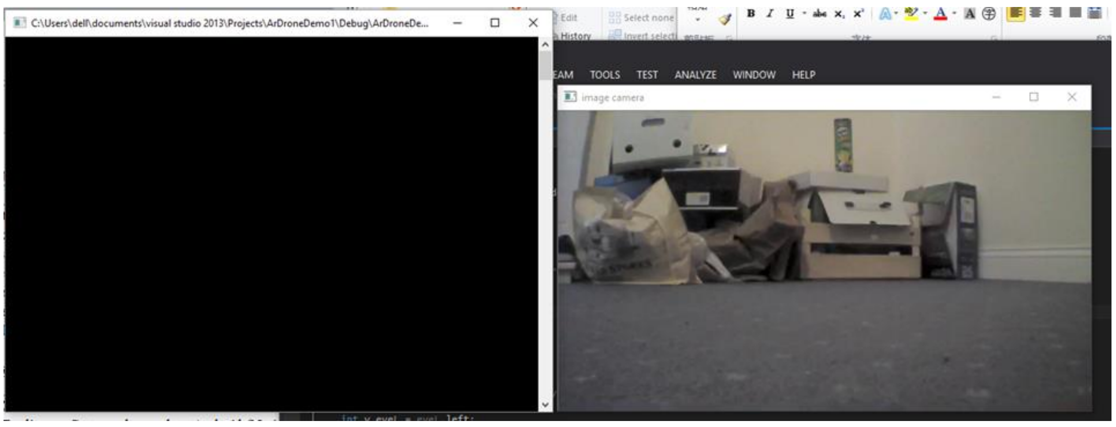


Figure 3. UI Example - 1

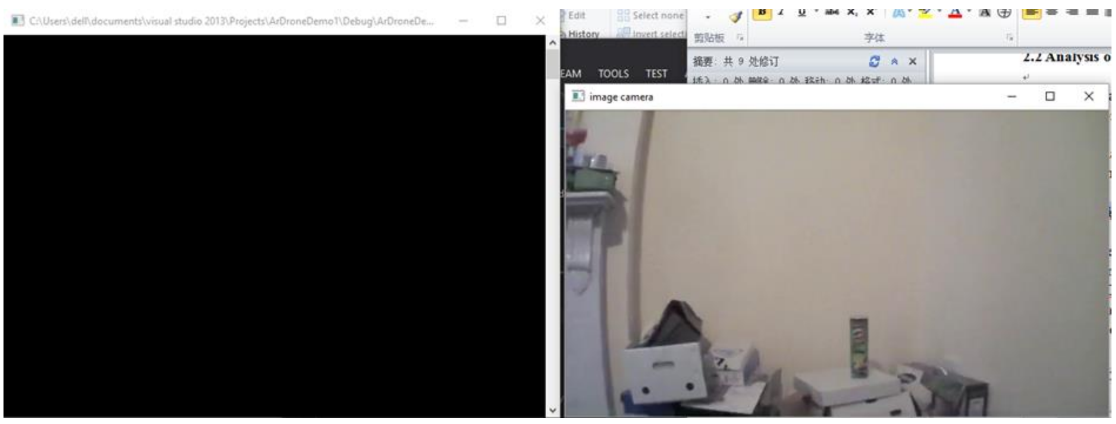


Figure 4. UI Example - 2

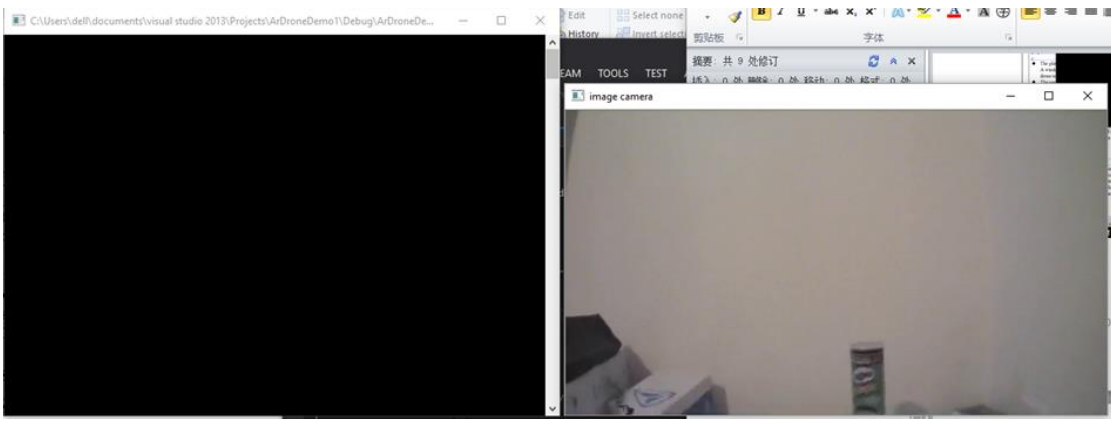


Figure 5. UI Example - 3

The instructions about how to use a keyboard to control the drone: ‘W’ is flying forward (suppose that the main camera is the front), ‘S’ is flying backward, ‘A’ is flying leftward, ‘D’ is flying rightward, ‘J’ is spinning left, ‘L’ is spinning right, ‘I’ is rising in the air, ‘K’ is going down, ‘Space’ is used for taking off and landing. The keyboard can be used as a safety backup.

The drone lifts off and looks for a person, then tries to identify the person. All faces and their eyes are outline by rectangles on the screen as well as LEDs on the drone turn red. When the drone recognizes me, the drone will fly following my face and the LEDs become orange. When I am closer than the safety distance (around 1 meter) to the drone, it will keep flying away from me until it reaches a safety place. When I am too far than the safety distance to the drone, it will fly towards until it reaches the safety place. When I move left or right, the drone will fly left or right. When my face goes up and down, the drone will fly up and down. For the people that the drone does not recognize, it will only do the fly-away movement.

Figure 6 shows that the LEDs of the drone are green normally when on ground. Figure 7 shows that LEDs turn red when a face is detected. Figure 8 shows that LEDs turn orange when someone is recognized. Figure 9 shows that the drone detects and recognised my face.

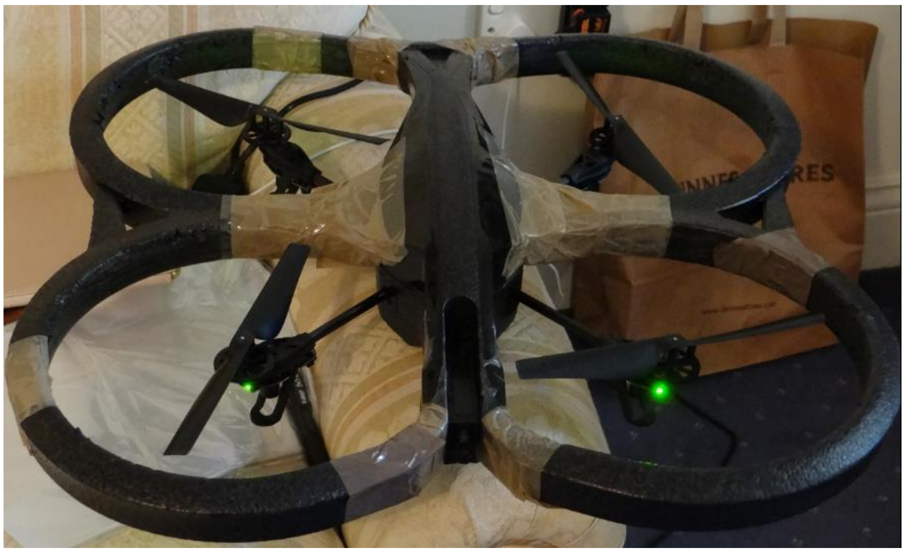


Figure 6. Green LEDs for normal working conditions

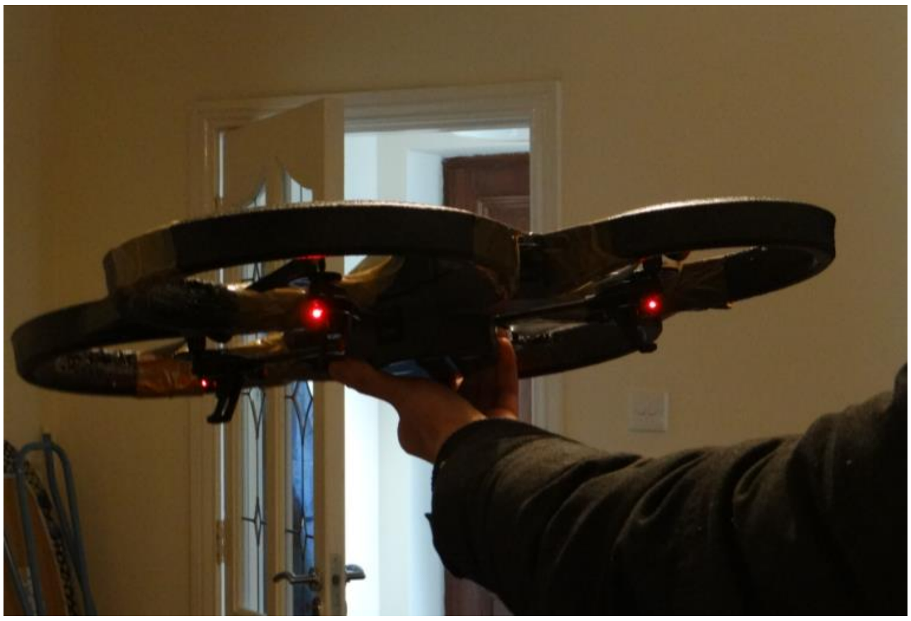


Figure 7. Red LEDs for when a face is detected

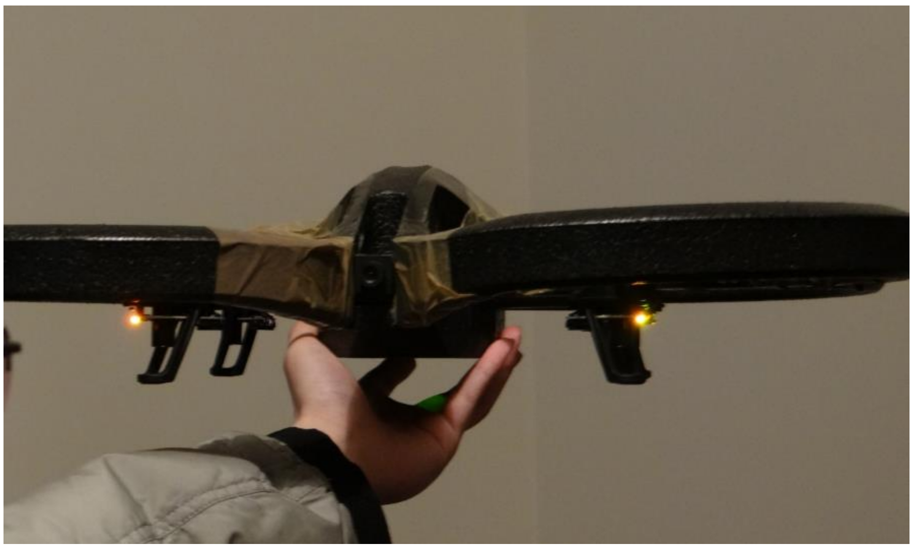
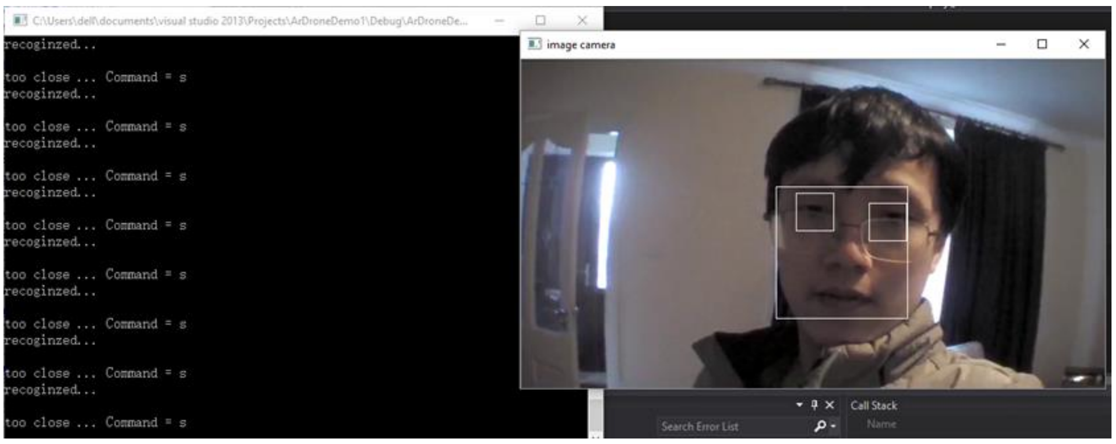


Figure 8. Orange LEds when face is recognized



C:\Users\dell\documents\visual studio 2013\Projects\ArDroneDemo1\Debug\ArDroneDe...

```
recoginzed...

too close ... Command = s
recoginzed...

too close ... Command = s
recoginzed...

too close ... Command = s
recoginzed...

too close ... Command = s
recoginzed...

too close ... Command = s
recoginzed...

too close ... Command = s
recoginzed...

too close ... Command = s
recoginzed...

too close ... Command = s
```

Figure 9. UI screenshot with detected face and eyes

3 Implementation

3.1 Application interaction with libraries

Libraries used: OpenCV, Fotonation face libraries, FFMPEG, windows socket

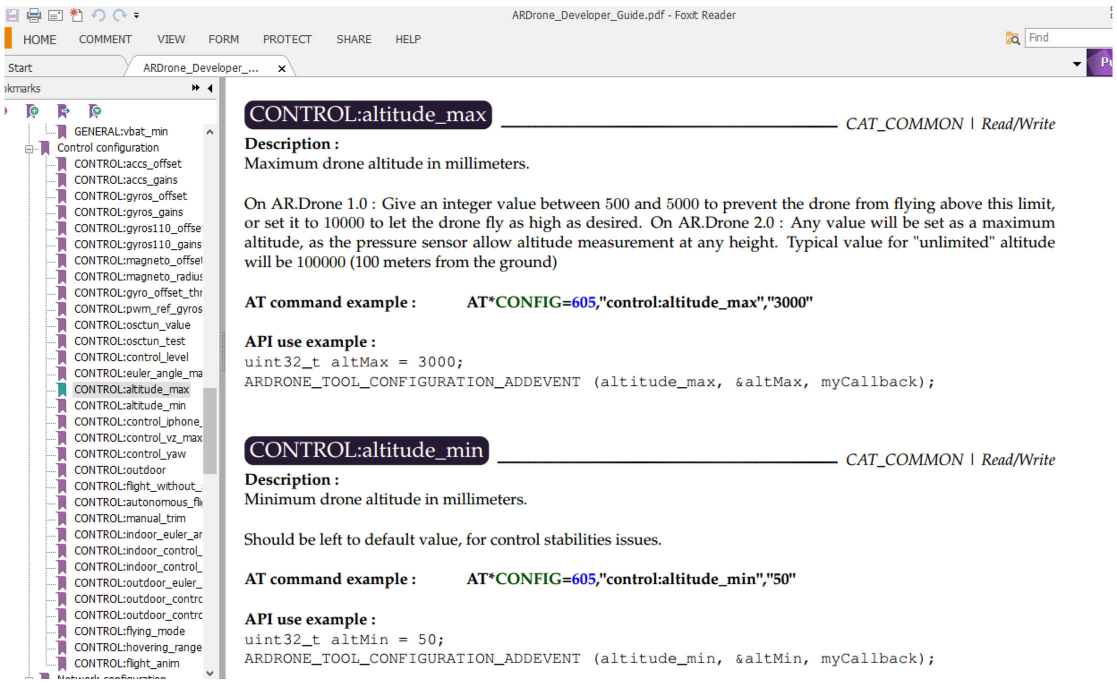
OpenCV library is used to show image on screen, listen to keyboard on the video window so that when a key is pressed, the application will get the information (index) of the key immediately. The draw function is used to draw a rectangle when faces are detected. [1]

Fotonation face libraries are provided by Fotonation. These libraries are run–time and they are able to do face detection, face recognition and return the information about the detected face such as the location of the face and the location of the eyes, and the information about whether a face is recognized or not. [26]

FFMPEG is used to catch TCP video stream, decode the video and save the image data to memory. Zeranoe FFMPEG Build is used. [2] [3]

Windows Sockets 2 is used in the application to send strings to drone to control the drone through UDP. [27]

The ARDrone_SDK_2_0_1 from Parrot is not used but the ARDrone Developer Guide here. [6]
Figure 10 shows a part of API use examples from the ARDrone Developer Guide.



CONTROL:altitude_max — *CAT_COMMON | Read/Write*

Description :
Maximum drone altitude in millimeters.

On AR.Drone 1.0 : Give an integer value between 500 and 5000 to prevent the drone from flying above this limit, or set it to 10000 to let the drone fly as high as desired. On AR.Drone 2.0 : Any value will be set as a maximum altitude, as the pressure sensor allow altitude measurement at any height. Typical value for "unlimited" altitude will be 100000 (100 meters from the ground)

AT command example : **AT*CONFIG=605,"control:altitude_max","3000"**

API use example :
```
uint32_t altMax = 3000;
ARDRONE_TOOL_CONFIGURATION_ADDEVENT (altitude_max, &altMax, myCallback);
```

CONTROL:altitude_min — *CAT_COMMON | Read/Write*

Description :
Minimum drone altitude in millimeters.

Should be left to default value, for control stabilities issues.

AT command example : **AT*CONFIG=605,"control:altitude_min","50"**

API use example :
```
uint32_t altMin = 50;
ARDRONE_TOOL_CONFIGURATION_ADDEVENT (altitude_min, &altMin, myCallback);
```

Figure 10. API example

3.2 Data flow

The data flow of controlling through image is shown in Figure 11.

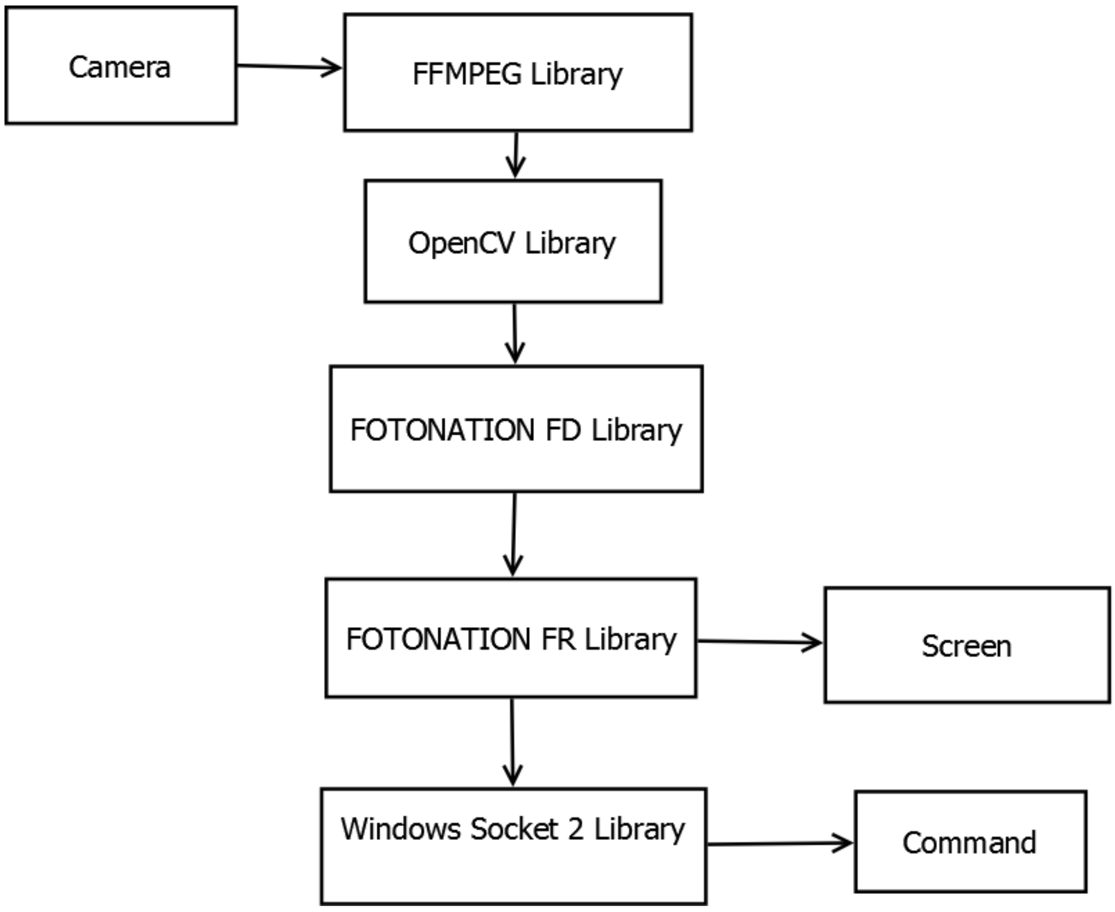


Figure 11. Image dataflow

Here are the details about the blocks in Figure 11.

- Camera: The image data from the camera of the drone is sent through TCP by the drone. The IP address is "192.168.1.1", the port is 5555.
- FFMPEG Library: This app uses this library to connect to the drone, receive TCP video stream, and decode video and store the decoded frame data in memory.
- OpenCV Library: The frame data is copied to a BGR24 IplImage (IC) [23]. Then convert the data from RGB to Grayscale and store the data in another IplImage (IB).
- FOTONATION FD Library: This app uses Fotonation Face Detection Library to detect faces from the current frame data and gives the information about the detected faces and eyes.

- FOTONATION FR Library: This app uses Fotonation Face Recognition Library to identify faces from the current frame data. Then the app will know whether a face is recognized or not.
- Screen: All detected faces and eyes are outlined with rectangles. The video is shown to the screen.
- Windows Socket 2 Library: This app uses Windows Socket 2 to send UDP commands to the drone.
- Command: The controlling UDP commands are determined by whether faces are detected or not and whether faces are recognized or not. The command data is consisted of a string, which is called AT Command and described by Parrot. For example, from the ARDrone Developer Guide, "AT*REF=%d,290717696\r", which is landing. [6] [7]

The data flow of controlling through keyboard is shown in Figure 12.

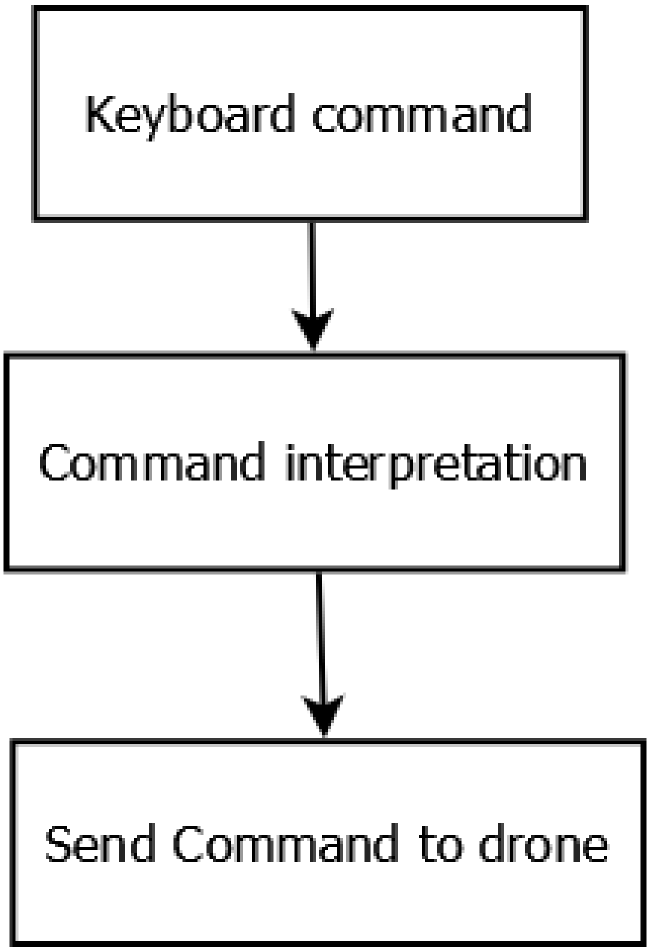


Figure 12. Control flow

Here are the details about the blocks of Figure 12 above.

1. Keyboard command: If any key of keyboard is pressed, the application will catch the information of the pressed key.
2. Command interpretation: The app creates AT Commands (Specific string) based

on what information is got from the pressed key.

3. Send Command to drone: The AT command resides in a UDP packet which will be sent.

For better development and testing, the application will write logs for its actions.

3.3 Image analysis

The image is updated and shown on run-time.

Faces and eyes from the image of main camera are detected by the face detection library. When the coordinate of top left corner, the width, and the height of a rectangle is known, this rectangle can be drawn by replacing the pixel data on rectangle to 255 which is white. I used a function from OpenCV to draw a rectangle for now.

The distance between a detected face and the drone is calculated based on that the pixel-distance of detected faces is known. In the distance calculation, I am using a formula, which also uses sensor and lens specifications. The details and formula are shown below.

$$Distance[mm] = \frac{GenericDistBtwEyes[mm] + FocalLength[mm]}{DistEye[pix] + {SensorWidth}/{ImgWidth}}$$

Distance: the distance between a camera and a face, a constant [mm]

GenericDistBtwEyes: the generic Inter - Pupillary Distance of a human, a constant [mm]

FocalLength: the focal length of the camera lens, a constant [mm]

DistEye: the distance between two eyes, as returned by the Face Detection library [pix]

SensorWidth: the width of the sensor, a constant [pix]

ImgWidth: the width of the image, a constant [pix]

Figure 13 shows how a digital camera catches images from light through as well as how lens, light and focal length look like. [24]

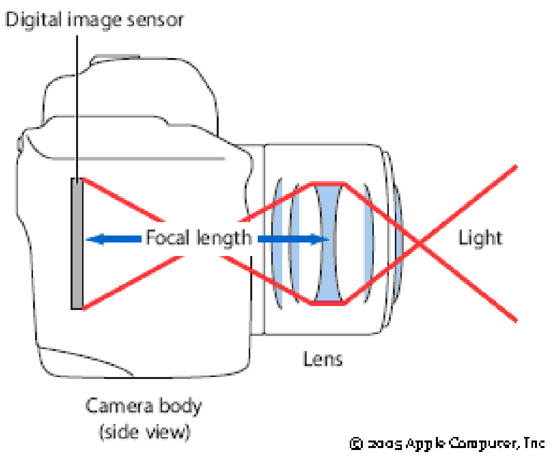


Figure 13

I have accessed the image from the bottom camera. As part of future work, I am planning to detect lines. More details will be provided in the future work chapter (chapter 5) of this report.

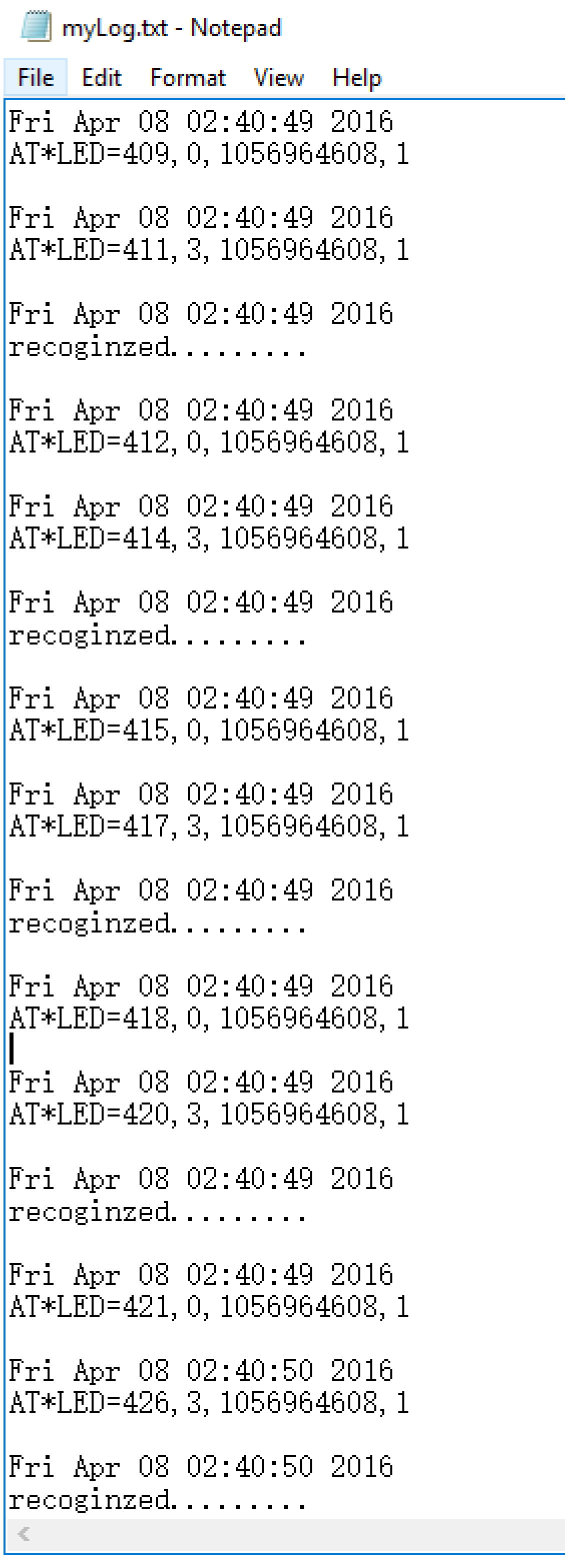
4 Testing

4.1 The experimental environment

The test was done inside a room with at least 10 square meters and about 3 meters height empty space. 2 to 3 people took part in the test and all of 3 had the ability to leave away from the drone in a short time. So I think the environment is safe for testing.

4.2 Testing

The program was run, let the drone detect and recognise my face first. Then pressed space, W, S, A, D, I, K, J, L a variety of times and different orders to see whether the drone would fly forward, backward, leftward, rightward, up, down, spin left and spin right or not. In most cases, the drone can be controlled through keyboard correctly.
Then I stood in front of the drone, I moved nearby with my face opposite to the drone to see whether the drone would recognise and follow my performance.
Let other test members did the same things as I did. Then the results were seen.
After testing the program, check the logs. The image libraries from Fotonation were well implemented. In almost all the cases, the faces are detected and recognized correctly according the drone movement and logs. For example, I checked most recent logs, and it shows that my face from most recent frames is recognized.
Figure 12 shows a part of information of the log file created by the application.



```
Fri Apr 08 02:40:49 2016
AT*LED=409,0,1056964608,1

Fri Apr 08 02:40:49 2016
AT*LED=411,3,1056964608,1

Fri Apr 08 02:40:49 2016
recoginzed.........

Fri Apr 08 02:40:49 2016
AT*LED=412,0,1056964608,1

Fri Apr 08 02:40:49 2016
AT*LED=414,3,1056964608,1

Fri Apr 08 02:40:49 2016
recoginzed.........

Fri Apr 08 02:40:49 2016
AT*LED=415,0,1056964608,1

Fri Apr 08 02:40:49 2016
AT*LED=417,3,1056964608,1

Fri Apr 08 02:40:49 2016
recoginzed.........

Fri Apr 08 02:40:49 2016
AT*LED=418,0,1056964608,1

Fri Apr 08 02:40:49 2016
AT*LED=420,3,1056964608,1

Fri Apr 08 02:40:49 2016
recoginzed.........

Fri Apr 08 02:40:49 2016
AT*LED=421,0,1056964608,1

Fri Apr 08 02:40:50 2016
AT*LED=426,3,1056964608,1

Fri Apr 08 02:40:50 2016
recoginzed.........
```

Figure 14. Log file sample

5 Future work discussions

5.1 Technology discussions and proposal for future work

This application is written in C on MS VS2013. Different from C like an object-oriented language, Java, I think a ‘big’ project should be better considered as a Christmas tree (layer by layer), for example, make AT command as an abstract class and other AT command such as take-off, land, and rise as subclasses. These classes have common methods which are set types, parameters and sending command, class design should be based on behaviour (method or function) in my opinion. It will make development more organized, managed and further used.
I used C in this project and implement most behaviours of the drone as functions. And I think structure is an important way to know if the windows thread is used [4]. For the future, implement this project in a different and object-oriented language may improve the quality of the project.
For information about abstract class Java, see [25]

5.2 More effects and app features

Get the NVDATA from the drone such as the information of the drone battery.
Improve frames per second.
More controls using faces.
The image from the bottom camera can be used to detect lines and a rectangle: firstly, find a straight line (L) from edges (by Canny algorithm), then find two lines which are parallel with the straight line (L) and two lines which are perpendicular to the straight line (L). After, the rectangle based on four lines from step 2 is detected. Finally, this rectangle can be used to let the drone fly along or land.
More information is shown on the image showing window. For example, show “recognized” on the recognized face and “unrecognized” on the unrecognized face.

5.3 Discussion of the response time on the user interface

Image shown is not quite accurate or not with high FPS when the keyboard is keep pressed and the face detection and recognition is on. This happens to only show image until both of the detection and the recognition is finished.

5.4 Other topics that were difficult and want to discuss possible improvement

Increase FPS:
The frame per second (FPS) is low because the application only show image after face detection and recognition and drawing rectangles are finished. This will make the time between showing one image and showing next image longer. So if I increase the FPS, the video from the video streaming window will look more frequently which will lead to a better user experience.

Improve the structure of codes:

In this project, I use C and put all the methods and global variables in one file and what I want is to improve the structure to make the codes look more managed and object-oriented. If I improve the structure of the codes, the project can be easily implemented in object-oriented languages such as Java.

6 Conclusions

The objective of this project is to build an application that is able to control a drone remotely, stream video from both cameras with a WIFI connection and process images in real-time from both main camera and bottom camera. The drone is successfully been controlled thought dedicated UDP commands. The drone can detect human faces, identify its owner, and follow the owner as well as keeping a safe distance.
Through this project, I am more familiar with UDP, TCP, video streaming, using both static and dynamic libraries, face detection face recognition, designing a project and testing faces.

References

1. OpenCV, OpenCV API Reference, http://docs.opencv.org/2.4/modules/refman.html .
2. FFMPEG, FFmpeg Documentation, https://ffmpeg.org/doxygen/2.8/
3. Zeranoe, Zeranoe FFMPEG Builds, https://ffmpeg.zeranoe.com/builds/
4. Dranger.com, Tutorial 04: Spawning Threads, http://dranger.com/ffmpeg/tutorial04.html
5. John Canny, A Computational Approach to Edge Detection, http://cmp.felk.cvut.cz/~cernyad2/TextCaptchaPdf/A%20Computational%20Approach%20to%20Edge%20Detection.pdf
6. Parrot, AR.Drone Developer Guide, pg1 to pg116
7. Shwetak, AR Drone Control API Notes, https://abstract.cs.washington.edu/~shwetak/classes/ee472/assignments/lab4/drone_api.pdf
8. Gorry Fairhurst, The User Datagram Protocol(UDP), http://www.erg.abdn.ac.uk/users/gorry/course/inet-pages/udp.html#Anchor-Larzon-14210
9. John Postel, User Datagram Protocol, http://www.ietf.org/rfc/rfc768.txt
10. Linktionary.com, TCP(Transmission Control Protocol), http://www.linktionary.com/t/tcp.html
11. Oracle, System Administration Guide vol3 chapter 4 Overview of TCP/IP, https://docs.oracle.com/cd/E19455-01/806-0916/6ja85398k/index.html
12. H. Schulzrinne, A. Rao, R. Lanphier, Real Time Streaming Protocol, https://www.ietf.org/rfc/rfc2326.txt
13. Abhijit Saha, Static and Dynamic Libraries, http://www.geeksforgeeks.org/static-vs-dynamic-libraries/
14. Wikipedia, Static library, https://en.wikipedia.org/wiki/Static_library
15. Microsoft, Dynamic-Link Libraries, https://msdn.microsoft.com/en-us/library/windows/desktop/ms682589(v=vs.85).aspx
16. Robert Frischholz, The Face Detection & Recognition Homepage, https://facedetection.com/
17. Robert Frischholz, The Face Detection Algorithms & Techniques, https://facedetection.com/algorithms/
18. Paul Viola, Michel J.Jones, Robust Real-Time Face Detection, http://www.vision.caltech.edu/html-files/EE148-2005-Spring/pprs/viola04ijcv.pdf
19. Face-rec.org, ALGORITHMS, http://www.face-rec.org/algorithms/
20. M. Turk, A. Pentland, Eigenfaces for Recognition, Journal of Cognitive Neuroscience
21. H. Moon, P.J. Phillips, Computational and Performance aspects of PCA-based Face Recognition Algorithms, Perception
22. Howdy Pierce, The many ways to stream video using RTP and RTSP, https://cardinalpeak.com/blog/the-many-ways-to-stream-video-using-rtp-and-rtsp/
23. OpenCV, IplImage Struct Reference, http://docs.opencv.org/trunk/d6/d5b/structIplImage.html#gsc.tab=0
24. dptips-central.com, http://www.dptips-central.com/image-files/focal_length.gif
25. oracle, Abstract Methods and Classes, https://docs.oracle.com/javase/tutorial/java/IandI/abstract.html
26. FotoNation, http://www.fotonation.com/
27. Microsoft, Windows Socket 2, https://msdn.microsoft.com/en-us/library/windows/desktop/ms740673(v=vs.85).aspx

28. Mbed.org, Socket, https://developer.mbed.org/handbook/Socket