Kinect Hand Gesture Drone Flight Controller

Darren Jody van Roodt

Thesis presented in fulfilment
of the requirements for the degree of
Bachelor of Science (Honours)
at the University of the Western Cape

Supervisor: Mr Mehrdad Ghaziasgar
Co-supervisor: Mr Reg Dodds

June 2016

Declaration

I, Darren Jody van Roodt, declare that this thesis "*Kinect Hand Gesture Drone Flight Controller*" is my own work, that it has not been submitted before for any degree or assessment at any other university, and that all the sources I have used or quoted have been indicated and acknowledged by means of complete references.

Signature: .... Date: ....

Darren Jody van Roodt.

Abstract

Controlling a drone using hand gestures is done by capturing the user's hands and arms and recognising the command that is being gestured. This project will attempt to develop a computerized hand gesture control system to control a drone during flight. The system will observe the user's hand gestures through a sequence of frames and convert the recognised gestures into signals to a drone, commanding it to move. The project will regard the hands and arms as the regions of interest. The user will be asked to make certain hand gestures which will be correctly identified by the system.

Key words

HDI

OpenCV

Acknowledgment

I would like to thank my supervisors Mr. M. Ghaziasgar and Mr. R. Dodds for their support and encouragement during my Honours year.

Contents

List of Tables

List of Figures

Glossary

HDI	Human drone interaction.
OpenCV	Open source library for computer vision
OpenKinect	Open source library for kinect
SVM	Support vector machines

Chapter 1

Background

This project is an attempt to control a drone with hand gestures. It entails developing an application that uses a Microsoft Kinect, which will be pointed at a user, to recognize hand gestures from the user. This will involve recognizing hand gestures first. Later it can be extended to more complex gestures. The application will then take the recognized hand gestures and convert them to signals to command the drone.

Chapter 2

User Requirements Document

2.1 Introduction

This section describes the problem from the user's point of view. It states the problem domain and the functionality of the program.

2.2 User's view of the problem

Controlling a drone is normally done through controllers, smart phones and tablets. The user wants a drone control system which uses natural movement as a means of controlling a drone.

2.3 Domain of the problem:

Personal drones are becoming popular in everyday environments. Therefore the domain of the problem is Human and Drone Interaction (HDI).

2.4 Description of the problem:

The user requires a system that uses natural movement to control a drone. The system will capture natural gestures and recognise. Through recognising the natural gestures the system can send signal commands to the drone, which will control it.

2.5 What is the solution for the problem?

The system that the user wants is a natural gestures drone controller. The natural gestures which the system will recognise is hand gestures. The solution for the suggested system will use a Microsoft Kinect to capture the hand gestures of the system user. Then system will have to recognise the hand gesture, every hand gesture is a control command for the drone. Therefore after recognising the hand gesture a command signal will be send to the drone to control it. The system will have a predefined hand gestures.

2.6 What is expected from the Software?

The Software is expected to detect the user's hands and arms using a Microsoft Kinect 360. Then the system will recognise hand gestures that the user makes to fly the drone. After recognising the hand gestures, a signal is sent to the drone which will make the drone move according to the gesture. The system will have a user friendly interface.

2.7 What is not expected from the Software?

The software will not track multiple users and it will not monitor users who are not directly facing the Kinect. The system is not expected to turn on the drone, the users need to turn it on using the button.

2.8 Conclusion

The above section describes what the user requires from the system that will allow them to control a drone with hand gestures. It also describes the expectations of the system to define the scope of the project. Chapter 3 provides the developers design of the system based on the users requirements.

Chapter 3

Requirements Analysis Document

3.1 Introduction

In Chapter 2, the user specified the need for a hand gesture controlled drone system. This chapter provide the design of the system based on the user requirements of the previous chapter. This will be done by breaking down the problem into high level parts and identifying the main parts to solve the problem.

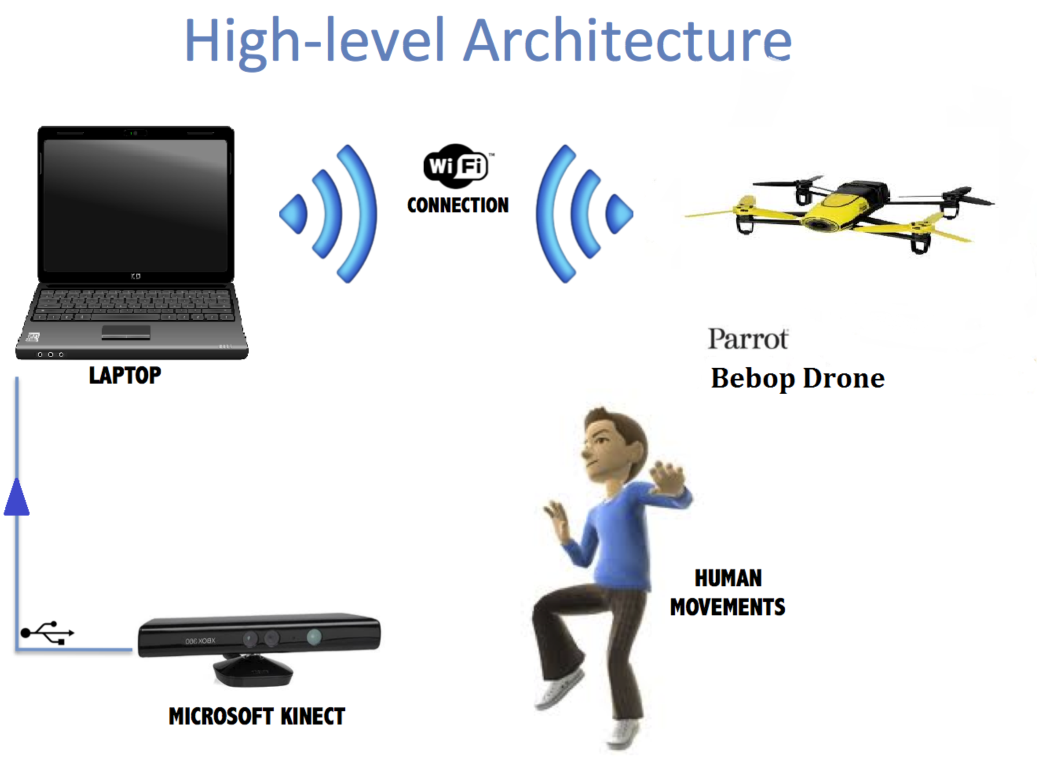

Figure 3.1: High Level Architecture

3.2 High level breakdown of problems

The application captures live videos of the subject, who is going to control the drone. The subject has to stand directly in front of the Kinect camera and facing it. When the subject is directly facing the Kinect then the subject has to perform the hand gestures which have been predefined to control the drone.

Recognising hand gestures needs to be accurately done. The video taken when the subject does the gesturing will have to track the hand and process the hand gestures. The processing of the video will be done with OpenCV, is a computer vision and machine learning software library which provides efficient implementation of many computer vision algorithms (Bradski and Kaehler, 2008). In order to send control commands to the drone the application has to transfer UDP(User Datagram Protocol) packets over WiFi.

3.3 Deeper analysis of the problem

3.3.1 Predefined hand gestures

A set of hand gestures is needed to control the drone. In this section a set of predefined hand gestures will be giving to control the drone in a very basic manure.

The hand gesture to make the drone take off and land is showing your palms to the kinect.

The next hand gesture will command the drone to do an emergency landing, the hand gesture is to cross your arm.

In order to make the drone fly forwards and backwards the user needs to clinch a fist with both hands, which creayes a virtual yoke, then to move your hand forward and backwords. Keep the same yoke form and turn your hand left to turn left and the same goes for turning right.
These hand gestures are basic for drone control. In a later stage more complex movements can be added.

3.3.2 Hands gestures and tracking

The application will accept video input from the Kinect. The region of interest is the hands. The application needs to be able to track the hands and also detect the gestures made by the user of the application. OpenCV will be used to track the hands and to determine which hand gestures were made.

3.3.3 Drone control

In order to control the drone the user will perform a predefined hand gesture. If the gesture is to rotate right then the drone needs to rotate right. In order to control the drone we need to understand the packets that are sent to the drone from the standard drone controller. Drones use WiFi to communicate with the controller. The packets that are sent over the WiFi of the drone uses UDP. Therefore if the user perform a gesture, the drone will get the commands via network packets.

3.4 Existing solutions

(Boudjit et al., 2013) has created a AR. Drone control of Flight Operator using the Depth Map of Microsoft Kinect. Using the depth map will render a skeleton from the user who is standing in front of Kinect. Then the user has to perform a set of gestures to control the drone. The 3D coordinates , (x,y,z) are used to control the drone. The system is accurate only over a useful range. (Sanna et al., 2013) created a drone controller which uses FAAST(Flexible Action and Articulate Skeleton Toolkit), in which they use the skeleton render of the user to determine which gesture is being made. The latency of system is 2 seconds.

3.5 Best solution

The best solution is be to detect hand gestures and recognise the commands for the drone. Then send the command to the drone via WiFi as an UDP packet.

This solution can be extended to us human movement to control drones in real-time. Tools need to implement the solution is a Microsoft Kinect and a computer will OpenCV. In order to track the arms system will use the skeleton approach to locate the arm and determine the position.

3.6 Testing the solution

The solution will be tested by calculating the accuracy of hand gestures of controlling the drone. Also, testing to determine if the application can send control commands to the drone in real time and if the system can determine the hand gesture in real time will be carried out.

3.7 Conclusion

This chapter discussed the designer interpretation of the problem and provides a framework to start developing with. The next chapter displays the project plan.

Chapter 4

User Interface Specification

4.1 Introduction

This chapter defines what the user interface is going to looks like and how the user interacts with the program. In this chapter the GUI will be defined, including all the components and the functionality of the GUI.

4.2 Description of the user interface

The user interface of this system is complex because we need to get live video from the kinect and the drone. The user interface also needs the status for the drone like the battery level , flying status and the X,y,z coordinates and the status of the hand recognition for example the recognised gesture and the time to execute the gesture.

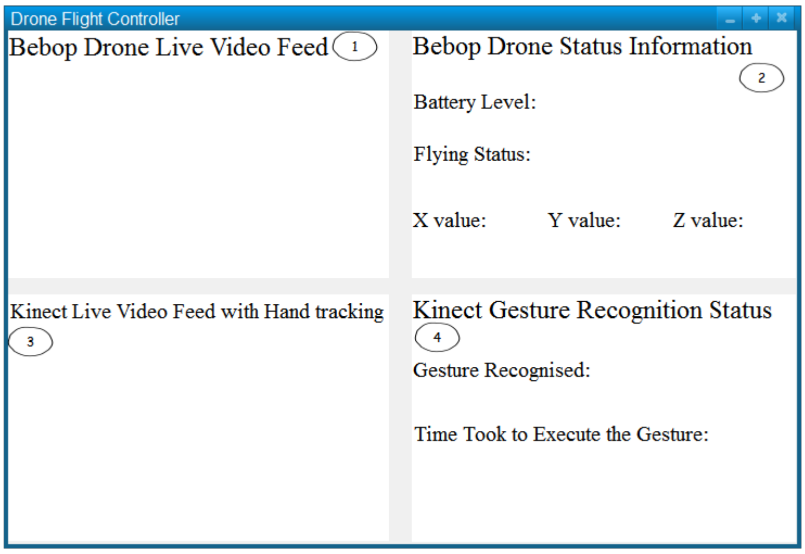
Drone Flight Controller
Bebop Drone Live Video Feed
1
Bebop Drone Status Information
2
Battery Level:
Flying Status:
X value:
Y value:
Z value:
Kinect Live Video Feed with Hand tracking
3
Kinect Gesture Recognition Status
4
Gesture Recognised:
Time Took to Execute the Gesture:

Figure 4.1: Main Screen

4.3 Main Screen

The system starts with the main screen that shows the live video feeds from the kinect and the drone. The drone needs to be connected via WiFi and the kinect needs to be connected via USB. If the drone and the kinect is not connected the system will not be able to work. Therefore this needs to be done before running the system.

1 Drone Live Video Feed: The user will be able see what the drone is pointing too. It helps seeing were the drone is pointing in order to control it.

2 Drone Status Information: The user will be able to see the drones battery level and flying status on this panel. User can also see the x,y.z coordinates of the drone.

3 Kinect Live Video Feed and Hand Tracking: In panel the users will be able to see the themselves. The users hands will be circled to show that the hands are being tracked. In order to track the hands the user needs to initialize the tracking with a closed fingers and their palms facing the kinect.

4 Kinect Gesture Recognition Status: In this panel the user will be able to see if the system is recognising their hand gesture correctly and see how long in seconds it takes for the drone to execute the gesture command.

4.4 Conclusion

This chapter discussed user interface with regards to the main windows and how the user interacts with the application.

Chapter 5

High Level Design

5.1 Introduction

This chapter focuses on high level design of the system. These will given in a detailed breakdown of the technical solution.

5.2 Breakdown of technical solution into subsystem

The technical solution is based on five essential parts, namely, input, hand tracking, hand gesture recognition, drone control and output.In the figure 5.1 shows the high level breakdown of the system and in next chapter it will be explained in detail.

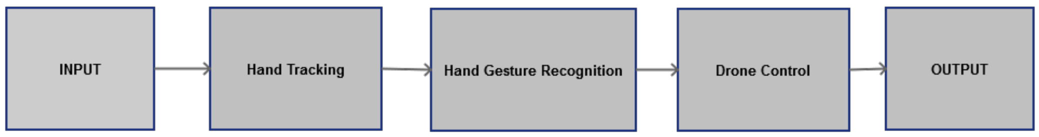

Figure 5.1: The Breakdown of the Technical Solution

5.3 Description of each subsystem and their interaction

5.3.1 Input

The input will be live video from the kinect. Where the user will be gesturing to control the drone.

5.3.2 Hand Tracking

The hands tracking is a three step process, which will be explained in the Low Level Design chapter. The kinect video will be used to find and track the hand.

5.3.3 Hand Gesture Recognition

Now that the hands are being tracked the user is able to perform hand gesture. The hand gestures will be recognized by using machine learning process.

5.3.4 Drone Control

The Drone control will take the results of the hand gesture recognition and send the commands to the drone to perform.

5.3.5 Output

The output will be the drone being controlled by the hand gestures also video and battery output.

5.4 Conclusion

This chapter explains the high level design of the system. It also describes the subsections in the technical breakdown.

Chapter 6

Low Level Design

6.1 Introduction

In this chapter the high level design will be described in more details rather than an a overview. This chapter will describe the subsections of the high level design including the methods used to execute the subsections.

6.2 Breakdown of High Level Design

6.2.1 Input

- Kinect Live Video Feed: The kinect video is allows the user to interact with the system. Which allows the user to gesture and control the drone. Capturing the video from the kinect the OpenKinect which is an open source software. OpenKinect allow the system to use the kinect for the video.

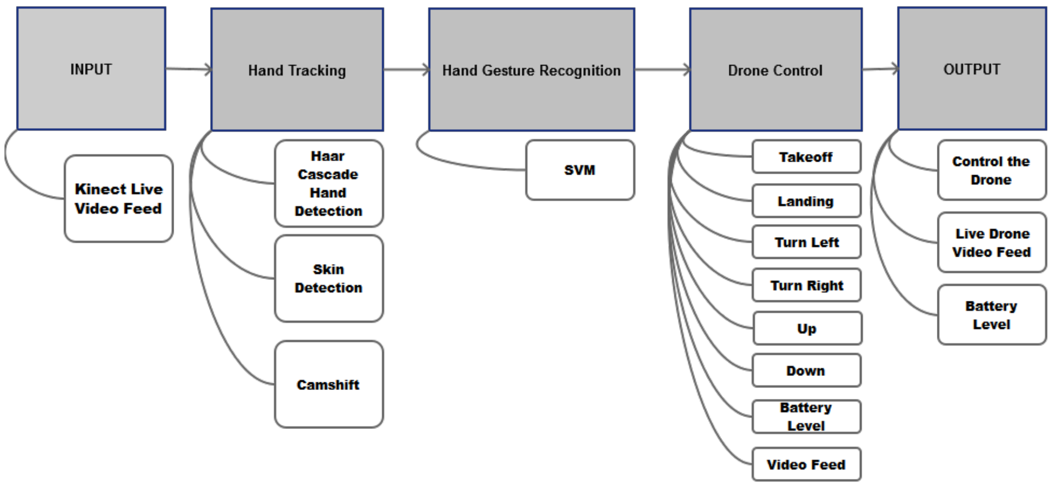

Figure 6.1: The Breakdown of the Technical Solution

6.2.2 Hand Tracking

Hand tracking is a three step process:

6.2.2.1 Haar Cascade Hand Detection

For rapid hand detection the system will implement haar cascade object detection by (Paul Viola, 2004). The haar cascade requires a grayscale frame from the input video and it return a vector of the hands location. The haar cascade uses haar features to detect the hands. Haar feature are represented by different orientations of black and white rectangles.

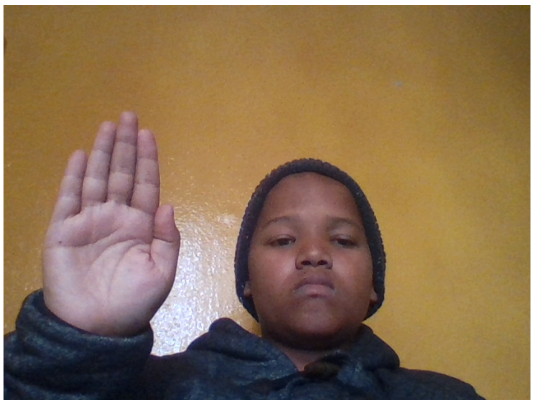

Figure 6.2: Before Hand Detection

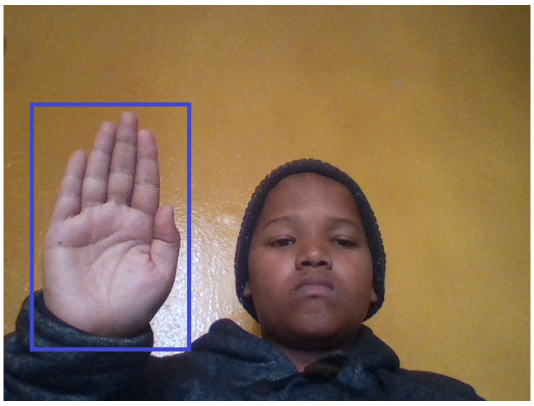

Figure 6.3: After Hand Detection

6.2.2.2 Skin Detection:

In order for the hand detection to work we need to do skin detection show that camshift can track the hands. The system does skin detection based on filtering the colour space RGB. The colour space needs to be normalised between 1 and 0. Where r is red an g is green and b is blue. Then if

$$\frac{3br^2}{(r+g+b)^3} > 0.1276 \tag{6.1}$$

and

$$\frac{g-r}{r+g+b} + \frac{r+g+b}{3r} <= 0.9498 \tag{6.2}$$

and

$$\frac{br+g^2}{gb} <= 2.7775 : \tag{6.3}$$

then skin else not skin

6.2.2.3 Camshift:

Camshift use a binary image, where white is skin and black other. Camshift will track the hand by setting a region of interest then it calculates the probability density function(pdf) for the skin pixels then it uses the moment of pdf the to track the hand (McKenna et al., 1999).

6.2.3 Hand Gesture Recognition

In order to recognise gestures the system will use a Support Vector Machine(SVM). The SVM will only work if its trained by different skin types and doing the predefined hand gestures.

- Training: In order to train the SVM a predefined hand gestures need to be performed by many users with different skin colours and hand sizes. This will be called the training set for the SVM from this predictions can be done on the training set.

- Prediction: The user of the system will perform a hand gesture then SVM will use the hand gestured frame to determine what gestures is being down.

6.2.4 Drone Control

- Takeoff: Makes the drone takeoff.
- Landing: Makes the drone land.
- Turn left: Makes the drone turn left.
- Turn right: Makes the drone turn right.
- Up:
  Makes the drone fly up.
- Down: Makes the drone fly down.
- Battery Level: Gives the battery level of the drone.
- Video Feed: Gives the live video feed of the drone.

6.2.5 Output

- Controlling the drone in flight
- Live Video of the Drone
- Battery level from the Drone

6.3 Conclusion

This chapter expressed a low level design, by breaking the high-level design into subsystems and explaining how they work together to make the system work as one.

Bibliography

Boudjit, K., Larbes, C., and Alouache, M. (2013). Control of flight operation of a quad rotor ar. drone using depth map from microsoft kinect sensor. *International Journal of Engineering and Innovative Technology*, 3(3):15–19.

Bradski, G. and Kaehler, A. (2008). *Learning OpenCV: Computer vision with the OpenCV library.* " O'Reilly Media, Inc.".

McKenna, S., Raja, Y., and Gong, S. (1999). Tracking colour objects using adaptive mixture models. *Image and Vision Computing Journal*, 17:223–229.

Paul Viola, M. J. J. (2004). Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154.

Sanna, A., Lamberti, F., Paravati, G., and Manuri, F. (2013). A kinect-based natural interface for quadrotor control. *Entertainment Computing*, 4(3):179–186.