

# **Kinect Hand Gesture Drone Flight Controller**

---

Darren Jody van Roodt  
Supervisor: Mr M Ghaziasgar  
Co-Supervisor: Mr R Dodds

# A Quick Recap...

## ➤ What is the aim of the System?

The system allows users to control a drone with a set of predefined hand gestures.

## ➤ Limitations:

Does not track more than one user at a time. Users of the system should turn the drone on.

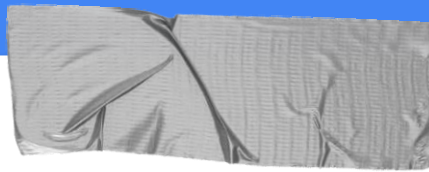
## ➤ Expectations:

System should detect users hands.

Recognise hand gestures, which will control the drone.

The system will have a User friendly interface.






# Overview

- User Interface Specifications
- High-Level Design
- Low-Level Design
- Prototype Demo


# User Interface Specification

Drone Flight Controller

Bebop Drone Live Video Feed (1)



Kinect Live Video Feed with Hand tracking (3)



Bebop Drone Status Information (2)

Battery Level:

Flying Status:

X value:      Y value:      Z value:

Kinect Gesture Recognition Status (4)

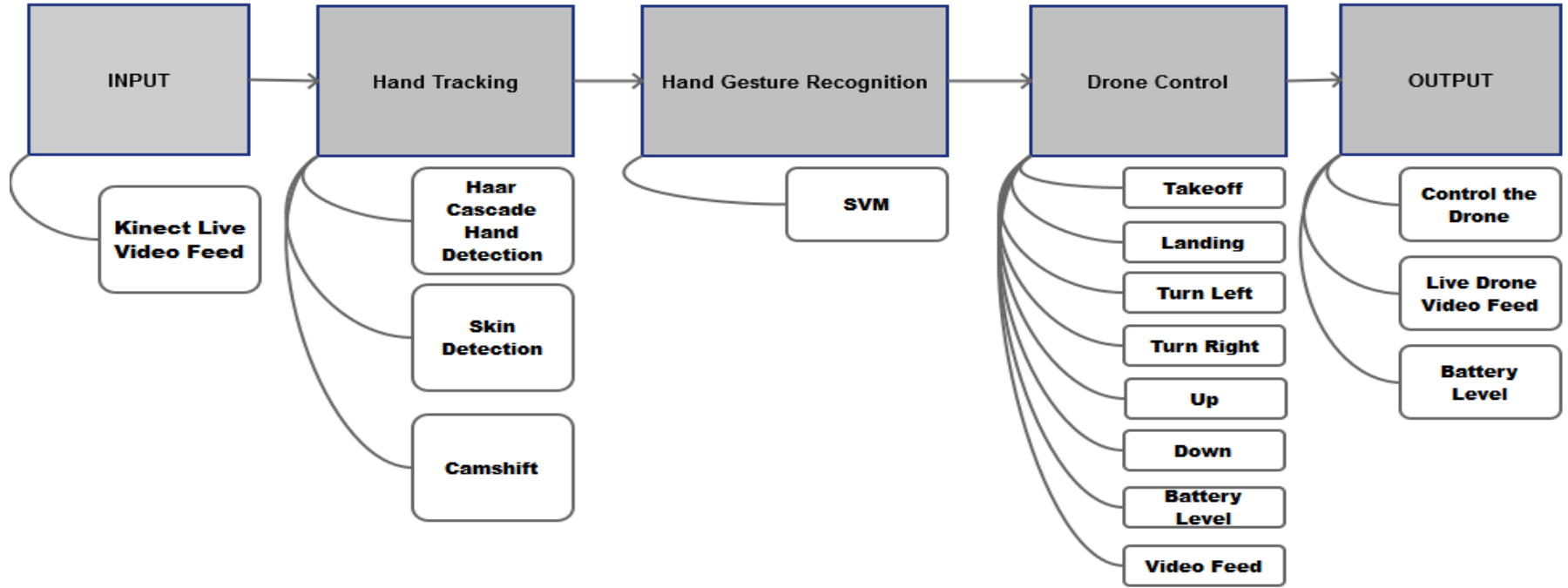
Gesture Recognised:

Time Took to Execute the Gesture:

# High-Level Design



# Low-Level Design

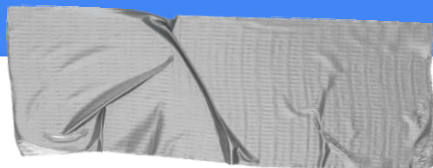




# Input

- Kinect provides a live video feed as input.
- In order to use the Kinect the system makes uses of OpenKinect which allows the system to retrieve the video from the kinect with the programming language Python.



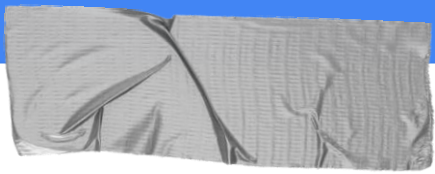


# Hand tracking

→ Hand tracking is a 3 step process:

1. Haar Cascade: Hand Detection
2. Skin Detection
3. Camshift





# 1) Haar Cascade

- Object Detection
- Viola & Jones Framework using Haar features
- `handCascade.detectMultiScale()`



## 2) Skin Detection

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

and

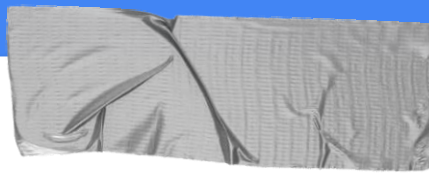
$$\frac{g-r}{r+g+b} + \frac{r+g+b}{3r} \leq 0.9498 \quad (6.2)$$

and

$$\frac{br+g^2}{gb} \leq 2.7775 : \quad \text{Rectangular Snip} \quad (6.3)$$

## 3) Camshift

- Camshift allow the system to track the user's hand in real time. In order to run camshift a step process is done
1. Select the region of interest of the probability distribution image to the entire image.
  2. Select an initial start for the Mean Shift search window.
  3. Calculate a colour probability distribution of the region centered at the window.
  4. Iterate Mean Shift algorithm to find the centroid of the probability image. Store the zeroth moment and the centroid location
  5. For the next frame centered the windows at mean location found in step4 and set the windows size to a function of zeroth moment. Go to step 3 and repeat.



## Hand Gesture Recognition

→ Hand Gesture Recognition use a Support Vector Machine to recognize hand gestures.

# Support Vector Machine

## Training

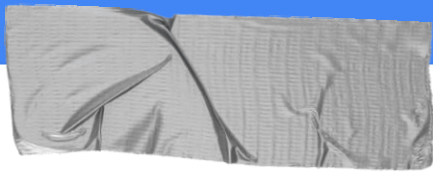
- **Labelled data is used as training data.**
- **Multiple hand gesture from different user's need to be captured to train the SVM.**

## Testing

- **Unlabelled data that the system has never seen.**
- **SVM predicts the hand gesture based on the labelled data.**

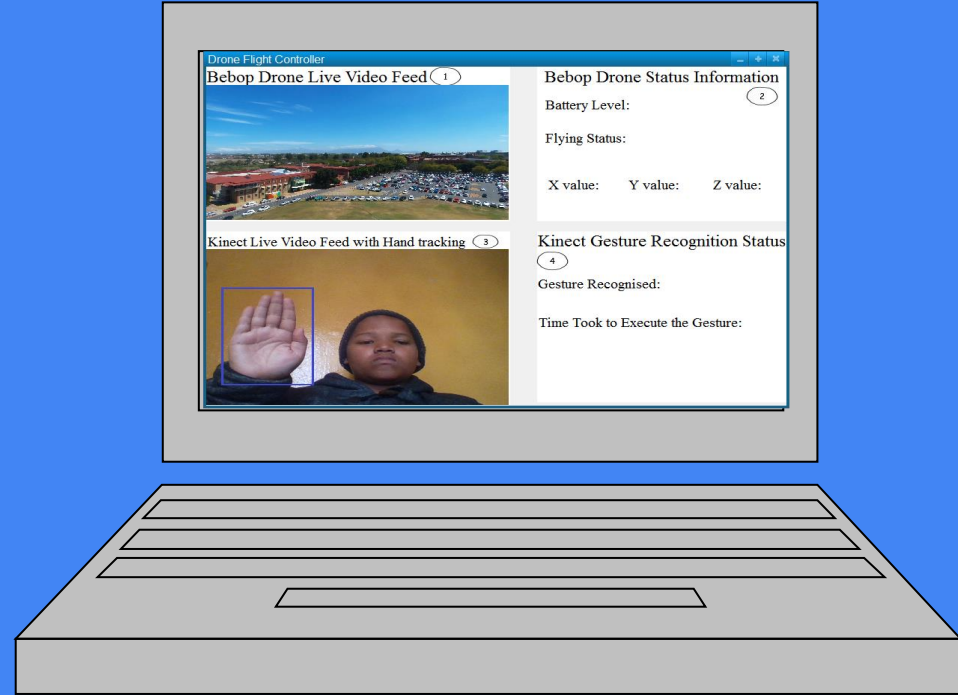
# Drone Control

- Drone Control allows the system to control the drone
- The control also allows the system to get the video feed and battery level of the drone
- Drone Control is the system of the drone which gives the drone commands to fly.



# Output

- The output from the system is the drone control or seeing the drone move when the gesture command it too.
- Other output are the live drone feed and battery level.



# Project Plan

## Research

- Learn to use OpenCV for image processing
- Learn to send commands to the drone

Term 1

Term 2

Term 3

Term 4

## Prototype

- Accurately locate region of interest

## Implementation

- Recognise a predefined set of hand gestures
- Send commands to drone via WiFi

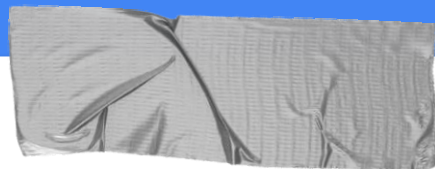
## Test and Evaluate

- Test the accuracy of the system and add more gestures
- Test the speed of detecting a gesture and improve
- Test the accuracy of sending a signal to the drone



# References

- 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.



# Prototype Demo

1. Hand Detection
2. Segment Hand
3. Skin Detection
4. Camshift

**Questions?**