The University Of Sheffield.

Mouse Control by Hand Gestures

Student: Peng Cheng
Supervisor: Yoshi Gotoh
Department: Computer Science
Course: Msc in Software System and Internet Technology

This report is submitted in the partial fulfilment of the requirement for the degree of Msc Software System and Internet Technology.

Declaration

All sentences or passages quoted in this report from other people's work have been specifically acknowledged by clear cross-referencing to author, work and page(s).

Any illustrations that are not the work of the author of this report have been used with the explicit permission of the originator and are specifically acknowledged.

I understand that failure to do the amounts of plagiarism and will be considered grounds for failure in this project and the degree examination as a whole.

Name: _____

Signature: _____

Date: _____

Abstract

At present, there is a pressing demand for motion sensing technology in the worldwide market, but the supply side is still slightly less. The main reason is that the current motion sensing technology is not mature enough for massive using and popularizing.

This project aims to design a software platform to allow people to interact with a personal computer that almost everyone will have and work with it by using body actions. The relevant body actions are hand gestures.

For now, this project has met the basic requirements of my design. Users can interact with the system via three kinds of hand gestures. The accuracy of recognition is acceptable. However, there are some problems in the project. For example, the recognition will be disturbed by lighting, complex background, etc. or the noise cannot be removed perfectly. Therefore, the project still has left room to grow in order to be better and easier to use.

Table of Contents

Table of Figures:

1. Introduction

1.1. Background

At present, according to different principles and application ways, the evolution of motion sensing technology in the world can be divided into two categories, which are mechanical motion tracking and electronic motion tracking.

Currently, there is a pressing demand for motion sensing technology in the worldwide market, but the supply side is still slightly less. The main reason is that the current motion sensing technology is not mature enough for massive using and popularizing.

One major flaw of current motion sensing technology is limited usage scope, which is still left room for technology to grow.

Some more technologies which are more portable are expected by consumers.

1.2. Aims and Objectives

The project aims to design a software platform to allow people to interact with a personal computer that almost everyone will have and work with it by using body actions. The relevant body actions are hand gestures.

In order to achieve the project aims, below is the specific objectives which can be reached during the process:

Compared with the Leap Motion controller, the usage scope of this finished system may be extended.

Figure 1 Leap motion controller

As you can see, the figure 1 shows the leap motion controller.

Cameras would be defined/chosen as the main device to help computers catch body actions.

For Mouse's Both-handed Operation, scaling function can be replaced and implemented when camera catches corresponding body actions.

For Mouse's One-handed Operation, the below mouse actions can be replaced and implemented when camera catches corresponding body actions.

1. Left clicking and right clicking.
2. Cursor moves.
3. Scroll up and scroll down.

2. Literature Review

2.1. Relevant Technologies

In this project, there are two parts – hand gesture tracking and system control. The hand gesture tracking has some steps.[i]

1. Image acquisition

2. Data processing
3. Gesture segmentation
4. Compare gesture with default data
5. Gesture recognition

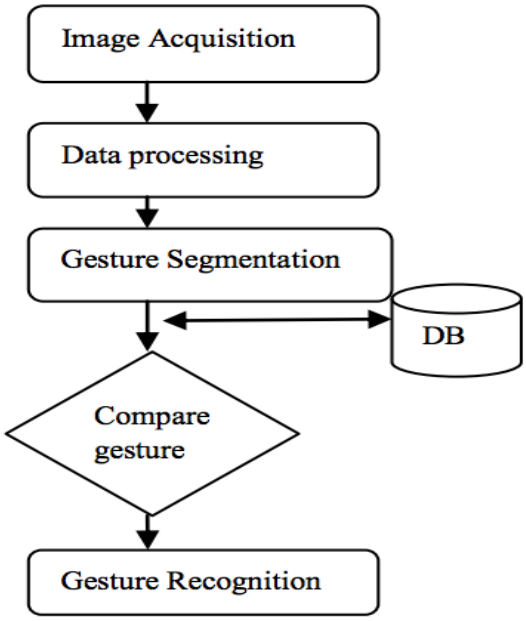

Figure 2 Hand gesture tracking steps [1]

The controlling part is just simulating the input of mouse and keyboard.

2.1.1. Data Acquisition

There are three ways to collect raw data.

The first is to use some wearable devices such as a glove. The device will measure many joints of the hand via a 6 DOF (The Degree of Freedom) tracking. Generally, this will consist of 1-2 gloves used to collect hand orientation and position data.

[1] Pradipa, R. and Kavitha, S., (2014). Hand Gesture Recognition – Analysis of Various Techniques, Methods and Their Algorithms. *International Journal of Electronics and Computer Science Engineering.*

The second is to use a vision-based approach (e.g. cameras) to gather pictures of the user's hands and the cameras will collect a necessary number of pictures per second and then the image processing program will do posture recognition and gesture recognition to find the position of hands.[ii]

The third way is a mixed approach combining the previous two approaches with the goals of realizing recognition with better accuracy via using two data streams to decrease the number of errors.

Wearable Devices: The most popular wearable device used to collect hand gesture is the glove. Many kinds of sensors are embedded in gloves and they can measure finger movement.

Vision-Based Approach: Comparing to wearable devices, the user does not need to wear any extra device and just need some cameras to collect raw data of posture and gesture recognition via vision-based approach. In other words, this approach will not limit movement of users comparing to the traditional approaches of interaction.[iii]

There are three key points in vision-based approach.[iv]

The first is the placement of cameras. The visibility's level of the user's hands that will be tracked should be the largest to get a more accurate recognition result so the placement of cameras will be much critical.[v]

The second is the visibility. It is necessary to reduce occlusion problems in vision-based approach with better visibility to the cameras. Generally, the extraction of hand data will be simpler if the hands can be more visible.

The amount of cameras which are used to track is also important in this approach because if there are more cameras, the hand data will be more accurate.[vi]

Conclusion:

Methods	Cost	User Comfort	Hand Anatomy	Calibration	Portability
Wearable Devices	Higher	Lesser	High	Critical	Lesser
Vision-Based Approach	Lesser	Higher	Less	Not Critical	Higher

Figure 3 Comparison diagram of two approaches

2.1.2. Data Analysis

All raw data must be analysed to determine whether the data is valid. In other words, the data with recognized postures or gestures correctly will be filtered as valid data.

There are six algorithms used to analyse data – Template Matching, Feature Extraction, Active Shape Model, the Principal Component, the Linear Fingertip Model and Causal Analysis.

Template Matching:

This is the simplest method to recognize hand postures. This method will check if the data can be classified as a piece of record of a set of default data. There are two parts in this approach. The first is to create some templates via gathering data for every posture in the set of posture. Another part is finding the template that most similar to the current data via comparing the current data collected by the sensor with the default data.[vii]

Feature Extraction Analysis:

In this approach, the basic information of the raw data will be used to generate advanced information and the generated information will be used to recognize postures and gestures. This approach is robust and can be used to

recognize many kinds of gestures and postures. In addition, the accuracy of this system is over 97%.[viii]

Active Shape Model:

This approach is used to find some features in some still images. The contour of the image will be moved iteratively to nearby edges and finally the contour will be deformed to fit the feature's shape. This approach is used in every frame and the feature's position will be as the original approximation for the next frame.

Principal Component Analysis:

This approach is statistical and used to decrease the dimensionality of some data sets.

The decreasing of the data set is by transforming the old data to a new set of variables that are ordered so that the first few variables contain most of the variation present in the original variables.[ix]

The original data set is transformed via computing the eigenvectors and eigenvalues of the data set's covariance matrix.

When dealing with image data is that it is highly sensitive to position, orientation and scaling of the hand in the image.

Linear Fingertip Model:

This approach is used only when the fingertips are applied to input data. It assumes finger movements are linear and there is almost no or little rotational movement.

The model uses only the fingertips as input data and permits a model that represents each fingertip trajectory through space as a simple vector.

Once the fingertips are detected, their trajectories are calculated using motion correspondence. The postures themselves are modelled from a small training set by storing a motion code, the gesture name and direction and magnitude vectors for each of the fingertips. The postures are recognized whether all the direction and magnitude vectors match (within some threshold) a gesture record in the training set. [x]

System testing showed good recognition accuracy (over 90%), but the system did not run in real time and the posture and gesture set should be expanded to determine whether the performance of this technique is high.

Casual Analysis:

This approach is vision-based. It extracts some information from some video streams via actions in the scene. Actually, the system will extract a feature set and the set will include wrist acceleration and deceleration, some work against gravity, size of gesture or posture, area between arms, angle between arms and verticality. There will be more researches on this technique to determine whether it is robust enough when it is used.

Conclusion:

Algorithms	Usage	Advantages	Disadvantages
Template Matching	Wearable Devices and Vision-Based Approach	Simplest Accurate for small set of postures Small amount of calibration	Not for hand gestures Does not work for large posture sets
Feature Extraction	Wearable Devices	Both postures and gestures Layered architecture	Computationally expensive
Active Shape Models	Vision-Based Approach	Real time recognition Both hand postures and gestures	Tracks only the open hand
Principal Components	Wearable Devices and Vision-Based Approach	Recognize on the order of 25 to 35 postures	More training needed
Linear Fingertip Models	Vision-Based Approach	Simple Good recognition accuracy	Not real time Recognizes small set of postures
Causal Analysis	Vision-Based Approach	Uses information about how human interact	Limited gestures No orientation

Figure 4 Comparisons of data analysis algorithms

2.1.3. Gesture Segmentation

This step is much important in gesture tracking. An efficient gesture segmentation is the key of success to all gesture tracking and recognition because of coms challenges of vision-based approaches like complex lighting condition, skin colour detection, etc.

There are six approaches used in segmentation – background subtraction, thresholding, anticipated static gesture set, hand segmentation using HSV colour space and sampled storage approach, hand segmentation using lab colour space and hand tracking and segmentation algorithm.

Background Subtraction:

It is also known as Foreground Detection, and is a technique to extract image's foreground.

The rationale in the approach is detecting moving objects via comparing the current frame with the previous frame and then extract the moving objects (image's foreground).

However, in general, the background subtraction is used on some static backgrounds and if the backgrounds are active, it will work incorrectly such as the background is the rain, clouds, etc.

There are various background subtraction algorithms such as MOG Background Subtraction, Subspace Learning Background Subtraction, Statistical Background Subtraction, Fuzzy Background Subtraction, RPCA Background Subtraction, Traditional and Recent Approaches for Background Subtraction, etc.

The MOG (Mixture of Gaussians) Background Subtraction is the most famous background subtraction algorithm. It models every pixel as a mixture of Gaussians and uses an on-line approximation to update the model. Actually, in the MOG algorithm, it assumes that every pixel's intensity values in the video can be modelled using a Gaussian mixture model. A simple heuristic determines which intensities are most probably of the background. Then the pixels which do not match to these are called the foreground pixels. Foreground pixels are grouped via using 2D connected component analysis.

Thresholding:

Thresholding is the simplest method of image segmentation. From a grayscale image, thresholding can be used to create binary images.

The simplest thresholding methods replace in an image with a black pixel if the image intensity $I_{i,j}$ is less than some fixed constant T (that is, $I_{i,j} < T$), or a white pixel if the image intensity is greater than that constant.

Figure 5 Original image[2]

[2] Wikipedia. (2016). *Thresholding (image processing)*. [online] Available at: https://en.wikipedia.org/wiki/Thresholding_(image_processing) [Accessed 29 Aug. 2016].

Figure 6 Result when the thresholding is used[3]

As you can see, the figure 5 is an image and the figure 6 is a result when a threshold effect is used on the figure 5.

To make thresholding completely automated, it is necessary for the computer to automatically select the threshold T. Sezgin and Sankur categorize thresholding methods into the following six groups based on the information the algorithm manipulates.

1. Histogram shape-based methods, where, for example, the peaks, valleys and curvatures of the smoothed histogram are analysed.
2. Clustering-based methods, where the grey-level samples are clustered in two parts as background and foreground, or alternately are modelled as a mixture of two Gaussians.

---

[3] Wikipedia. (2016). *Thresholding (image processing)*. [online] Available at: https://en.wikipedia.org/wiki/Thresholding_(image_processing) [Accessed 29 Aug. 2016].

3. Entropy-based methods result in algorithm that use the entropy of the foreground and background regions, the cross-entropy between the original and binarized image, etc.
4. Object Attribute-based methods search a measure of similarity between the grey-level and binarized images, such as fuzzy shape similarity, edge coincidence, etc.
5. Spatial methods that use higher-order probability distribution and/or correlation between pixels.
6. Local methods adapt the threshold value on each pixel to the local image characteristics. In these methods, a different T is selected for each pixel in the image.

In addition, colour images can also be thresholded. One approach is to designate a separate threshold for each of the RGB components of the image and then combine them with an AND operation. This reflects the way the camera works and how the data is stored in the computer, but it does not correspond to the way that people recognize colour. Therefore, the HSL and HSV colour models are more often used; note that since hue is a circular quantity it requires circular thresholding. It is also possible to use the CMYK colour model.

Anticipated Static Gesture Set:

Static gesture is a specific posture or gesture assigned with meaning. Application interface will be provided after recognition of specified posture for action. Simplicity and user friendliness were taken into consideration for the design of anticipated posture set. The gesture's centre of hand will be passed as a mouse cursor.

Hand Segmentation Using HSV Colour Space and Sampled Storage Approach:

This approach has been developed and tested for green colour glove. In this approach, segmentation based on colour is attempted using HSV colour space. The H, S and V separation uses following equations.

$V=\max\{R, G, B\}$

$\delta = V-\min\{R, G, B\}$

$S=\delta/V$

Hand Segmentation Using Lab Colour Space:

The input captures the RGB image that will be converted to lab colour space. Convolution operation was applied on binary images for the segmentation. In CIE-L, a, b co-ordinates, where L defines lightness, a represent red/green value and b denotes the blue/yellow colour value. The a axis and +a direction shift towards red while along the b axis and +b movement shift toward yellow. Once the image gets converted into a and b planes, thresholding was finished. Morphological processing was done to get the superior hand shape. This algorithm works for skin colour detection but it was sensitive for complex background.

There are 7 steps in this algorithm:

1. Capture the image.
2. Read the input image.
3. Convert RGB image into lab colour space.
4. Convert the colour values in I into colour structure specified in c-form.
5. Compute the threshold value.
6. Convert intensity image into binary image.
7. Performing morphological operations such as erosion.

Hand Tracking and Segmentation Algorithm:

This approach is robust in skin colour detection and removal of complex background. However, it was attempted on the hand detection and segmentation. In addition, mean shift algorithm was used in hand tracking.

Odd frame has been considered for fast processing. For better performance user's skin colour sample was passed and HSV histogram was created. CamShift function within the OpenCV library is used for tracking and detection. Edge traversal algorithm usage would get fine contour of the hand shape. As dynamic background was considered, while capturing the user's gesture after edge from the background. In an attempt to only identify the boundary of user's hand edge, traversal algorithm was devised.

There are 9 steps in this algorithm:
1. Capture the image frames from camera.
2. Process odd frames, track the hand using CamShift function by providing skin colour.
3. Samples at the run time.
4. HSV histogram is created and the experimented threshold value is passed to the CamShift function for tracking required hand portion.
5. Segment the required hand portion from Image.
6. Find the edges by using Canny edge detection.
7. Dilate the image.
8. Erode the image.
9. Apply edge traversal algorithm to get final contour.

2.1.4. Gesture Recognition

There are six approaches in gesture recognition – hidden Markov model, YUV colour space and camshaft algorithm, using time flight camera, Naïve Bayes' Classifier, 3D hand model and appearance-based approach.

Hidden Markov Model:

This approach is important to deal with the dynamic aspects of gestures. Gestures are extracted from a sequence of images via tracking the skin colour blobs corresponding to the hand into a body-face space centred on the face of

the user. The goal is to recognize two classes of gestures – deictic and symbolic. The image is filtered using a fast look-up indexing table.[xi]

After filtering, skin colour pixels are gathered into blobs. Blobs are statistical objects based on the location (x, y) and the colourimetry (Y, U, V) of the skin colour pixels in order to determine homogeneous areas.

YUV colour space and CamShift algorithm:
This approach is important to deal with recognition of hand gestures.

There are 5 steps in this algorithm:
At first, a digital camera records a video stream of hand gestures.
All the frames are taken into consideration and then using YUV colour space skin colour based segmentation is performed.
The YUV colour system is employed for separation chrominance and intensity. The symbol Y indicates intensity while UV specifies chrominance components.
Now the hand is separated using CamShift algorithm. Since the hand is the largest connected region, we can segment the hand from the body.
After this is finished, the position of the hand centroid is calculated in each frame. This is done by first calculation the 0^{th} and 1^{st} moments and then using this information the centroid is calculated.
Now the different centroid points are joint to form a trajectory. This trajectory shows the path of the hand movement and thus the hand tracking procedure is determined.

Using Time Flight Camera:
In this approach, x and y projections of the image and optional depth features are used for gesture classification. In addition, the 3-direction time-of-flight (TOF) sensor is used in the system so this approach is good at simplifying hand segmentation. The gestures used in the system show a good separation potential along the two image axes. Hence, the projections of the hand onto the x and y axis are used as features for the classification.

Naïve Bayes' Classifier:

This approach is effective and fast in static hand gesture recognition and it is based on classifying the different gestures. In addition, it is independent on skin colour. The gestures are extracted from each frame of video, with a static background.

There are some steps in this algorithm:[xii]

Segment and label the objects of interest and to extract geometric invariants from them.

Classify gestures by using a K-nearest neighbour algorithm aided with distance weighting algorithm (KNNDW) to provide suitable data for a locally weighted Naïve Bayes' classifier. The invariants of each region of interest are the input vector for this classifier, while the output is the type of the gesture.

Locate the specific properties of the gesture that are needed for processing in the system.

3D Hand Model-Based Approach:

This approach is based on the 3D kinematic hand model with DOF.

In addition, this algorithm estimates the hand parameters by comparisons between the input images and possible 2D appearance projected by the 3D hand model. And it is ideal for realistic interactions in virtual environments.[xiii]

Appearance-Based Approach:

Some image features are used in this approach in order to model the visual appearance of the hand and then, the parameters will be compared with the extracted image features from video stream. In addition, real time performance due to the easier 2D image features that are employed. A straight forward and simple approach that is often utilized.

2.1.5. Simulating mouse/keyboard

The simulator of mouse/keyboard can simulate input of mouse/keyboard at the bottom layer of system and all input will be recognized as which are from real input device. Nearly all actions of mouse/keyboard can be simulated such as moving, clicking, scrolling, etc.

In fact, so many testers/developers write automated functional tests which follow a prescriptive, possibly branching, possibly dynamic “user workflow”. The script can check for changes in the GUI itself, operating system environment, file system, database table and records, network, internet or extranet URLs/pages/web services, etc. In other words, anywhere that there could be changes can be checked by automated script/program. And many automated script/simulator of mouse/keyboard can actually do some tasks like human beings such as PyAutoGUI, WATSUP (Windows Application Test System Using Python), IEC, Pywinauto, etc.

PyAutoGUI:
It is a cross-platform GUI automation Python module for human beings and used to programmatically control the mouse and keyboard. All actions of mouse/keyboard can be simulated by this module.[xiv]

WATSUP:
It is a toolkit that designed to allow the automated test of Windows applications. The system uses the “object-based” mechanism for identifying and invoking actions on controls and menu items.

IEC:
It is a Python library that designed to automate and control an IE (Internet Explorer) window. The user can use this library to navigate to web pages, read the values of various HTML elements, set the values of checkboxes, text boxes, radio buttons etc., click on buttons and submit forms.

Pywinauto:

It is a set of Python modules to automate the Microsoft Windows GUI.
It itself is very simple to allow you to send mouse and keyboard actions to windows dialogs and controls, but actually is has supported many complex controls now.

Conclusion:

Techniques	Platforms	Target Objects	Functions
PyAutoGUI	Mac, Linux, Windows	All applications	All actions of mouse and keyboard
WATSUP	Windows	All applications	All actions of mouse and keyboard
IEC	Windows	IE	All actions of mouse and keyboard
Pywinauto	Windows	All applications	Part actions of mouse and keyboard

Figure 7 Comparisons of simulators

2.1.6. Additional Features

There are various techniques that can be used in the project.

Suzuki85:

This is an algorithm proposed by Satoshi Suzuki in 1985. It is an extension of the border following algorithm. [xv]

The border following algorithm is one of the fundamental techniques in the processing of digitized binary images. It derives a sequence of the coordinates or the chain codes from the border between a connected component of 1-pixels (1-component) and a connected component of 0-pixels (background or hole).

The extended algorithm of Suzuki has two main extensions:
1. To put a unique mark on each border rather than to adopt the same marking procedure for every border (border labelling).
2. To add a procedure for obtaining the parent border of the currently followed border.

In addition, there are 4 definitions in this extended algorithm.

D1: In the 4- (8-) connected case, a 1-pixel (i, j) having a 0-pixel (p, q) in its 8- (4-) neighbourhood is called a border point. It is also described as “a border point between a 1-component S_1 and a 0-component S_2”, if (i, j) is a member of S_1 and (p, q) is a member of S_2.

D2: For given two connected components S_1 and S_2 in a binary picture, if there exists a pixel belonging to S_2 for any 4-path from a pixel in S_1 to a pixel on the frame, we say that S_2 surrounds S_1. If S_2 surrounds S_1 and there exists a border point between them, then S_2 is said to surround S_1 directly.

D3: An outer border is defined as the set of the border points between an arbitrary 1-component and the 0-component which surrounds it directly. Similarly, we refer to the set of the border points between a hole and the 1-component which surrounds it directly as a hole border. We use the term “border” for either an outer border or a hole border. Note that the hole border is defined as a set of 1-pixels (not 0-pixels) as well as the outer border. And additionally, for an arbitrary 1-component of a binary picture its outer border is one and unique. For any hole its hole border (the border between that hole and the 1-component which surrounds it directly) is also unique.

D4: The parent border of an outer border between a 1-component S_1 and the 0-componenet S_2 which surrounds S_1 directly is defined as: (1) the hole border between S_2 and the 1-component which surrounds S_2 directly, if S_2 is a hole; (2) the frame of the picture, if S_2 is the background.

The parent border of a hole border between a hole S_3 and the 1-component S_4 which surrounds S_3 directly is defined as the outer border between S_4 and the 0-component which surrounds S_4 directly.

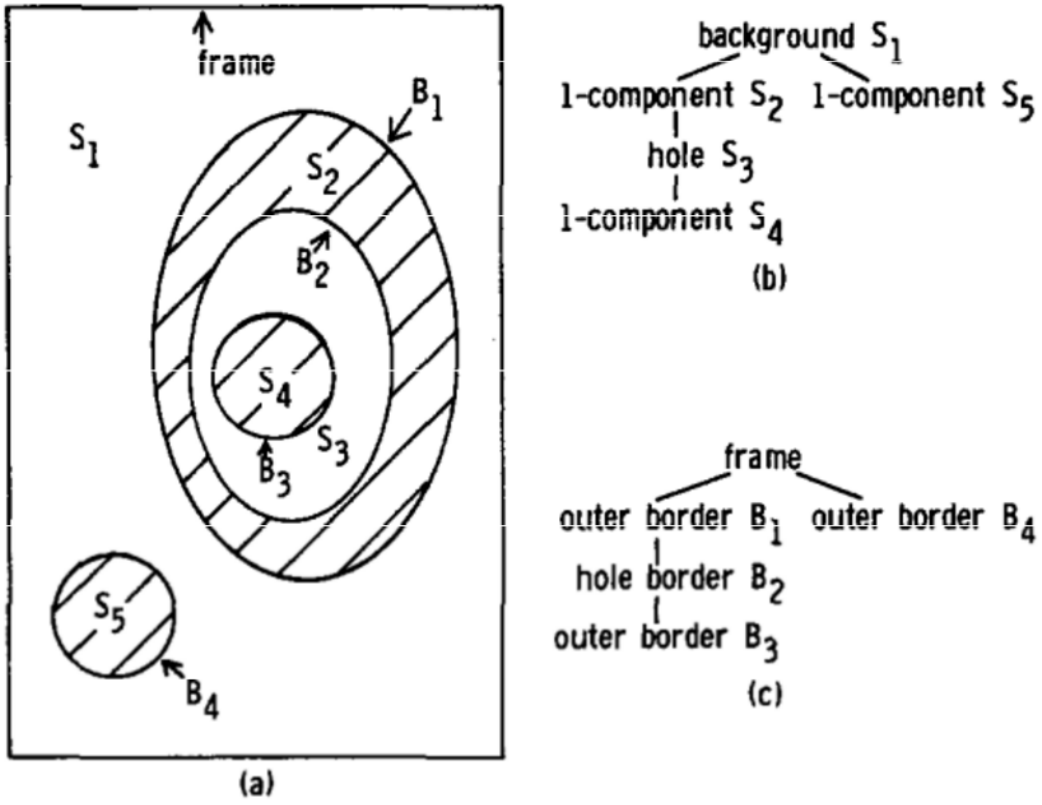

Figure 8 Surroundings among connected components and among borders[4]

Briefly, the process of extraction of contours is what is in the image below.

[4] Suzuki, S., (1985). Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, *30*(1), pp.32-46.

Figure 9 Process of contours' extraction[5]

As it displays in the figure 9, the system read the grey image at first, and then find the largest contour of the whole image. Finally, it figures out the real contour of the image.

Sklansky82:

This is an algorithm proposed by Jack Sklansky in 1982. It can be used to find the convex hull of any simple polygon specified by a sequence of m vertices.

All earlier convex hull algorithm is limited to polygons which remain simple when locally non-convex vertices are removed but we could find the convex hull of any simple polygon with complexity O (m) via this algorithm.

2.2. Related Works

There are many relevant products.

For example, Leap Motion controller is the most popular one. The Leap Motion controller is designed to be placed on a computer, facing upward and it can also be mounted onto a virtual reality headset. This device used two monochromatic IR cameras and three infrared LEDs. In addition, it only

[5] Opencvpython.blogspot.co.uk. (2016). *OpenCV-Python: Contours - 2 : Brotherhood*. [online] Available at: http://opencvpython.blogspot.co.uk/2012/06/contours-2-brotherhood.html [Accessed 29 Aug. 2016].

observes about 1 meter. With this device, the user can use his hands to control his computer.

Of course, the gesture tracking can also be implemented via software instead of only hardware.

Arc Soft has a pure software solution for gesture tracking. It used natural human hand gestures with one or both hands, such as wave, grab and move, as well as face, eye and finger motions that allow us to interact with our devices without actually touching them. In addition, this technology can support single-lens devices such as a consumer webcam and some stereoscopic devices.[xvi]

3. Requirement Analysis

3.1. Requirements

Firstly, the aim of our project is to detect and track hand gestures in real-time and then, control a computer as a mouse does.

In the system, some elements can affect the results of testing.

Gestures:

1. Moving the cursor with one finger.
2. Scrolling up/down/left/right with two fingers.
3. Left/Right clicking with a combined gesture. (one/two fingers and fist)
4. Dragging the cursor with a combined gesture.
5. More…

Background:

1. Monotone white wall.
2. Monotone wall (other colour except white).
3. Mixed background.

4. More…

Illuminance:

1. Brightest sunlight. (about 120000 lux)
2. Bright sunlight. (about 111000 lux)
3. Shade illuminated by entire clear blue sky at midday. (about 20000 lux)
4. Typical overcast day at midday. (about 1000 – 2000 lux)
5. Dark room. (about <1 lux)
6. More…

For data that used for testing, we prepared several scenes/examples to check if the system can/cannot work or how much accuracy will the system have for each example.

The first testing scene is that a user operates the system and he is in a room with monotone white walls. In addition, it is a sunny day so the bright sunshine gets inside the room via a window.

The second testing scene is that a user operates the system and he will only use one finger to try to move the cursor. In addition, it is a sunny day so the bright sunshine gets inside the room via a window. However, he will test the system in several rooms with different walls.

The third testing scene is that a user operates the system and he will use one fingers to try to move the cursor. In addition, he will test the system in several rooms with monotone white walls. However, the illuminance of every room is different.

3.2. Analysis

We have analyzed the requirements of the project and the data we will test and then, we draw the conclusion that there are 3 stages that would be

performed in the project. They are Data Capturing, Tracking and Processing and Motion Control.

3.2.1. Data Capturing

The issues related with this stage are what technology to use for collecting raw data from the hand. Generally, two types of technologies are available for collecting this raw data.

The first one is using input device, which measures a number of joint angles in the hand. Accuracy of an input device depends on the type of bend sensor technology used; usually, the more accurate the device is, the more expensive it is.

Another way of collecting raw data is to use computer vision. In a vision-based solution, one or more cameras placed in the environment record hand movement. By using a hand posture or gesture-based interface, the user does not want to wear the device and be physically attached to the computer. If vision-based solutions can overcome some of their difficulties and disadvantages, they appear to be the best choice for raw data collection.

3.2.2. Tracking and Processing

The issues related with this stage are which technology is the best to track the hand gesture.

First of all, the hand tracking can be divided into several stages and they are Segmentation and Recognition.

A lot of segmentation techniques can be used such as Background Subtraction, Thresholding, Anticipated Static Gesture Set, Hand

Segmentation Using HSV Color Space and Sampled Storage Approach, Hand Tracking and Segmentation (HTS) Algorithm, etc.

In addition, a number of recognition techniques are available such as Template Matching, Feature Extraction, Active Shape Models, etc.

All of these techniques all have some different advantages and disadvantages. For instance, the Active Shape Model can be applied to analyzed data because it can be used to do real time recognition and both hand postures and gestures can be recognized. However, obviously, its disadvantages will make this project hard to recognize some gestures when the hand is closing such as if the user's hands are clenched and then, he does some gestures, maybe the program will not recognize his gestures correctly.

3.2.3. Motion Control

The issues related with this stage are how the mouse can be replaced. There are several techniques of simulating mouse. They are PyAutoGUI, WATSUP, IEC, Pywinauto, etc.
It is easy to simulate mouse with them. For instance, the PyAutoGUI is a cross-platform GUI automation Python module for human beings and used to programmatically control the mouse and keyboard. All actions of mouse/keyboard can be simulated by this module.

3.3. Project Testing and Evaluation Strategy

The data used for testing is real-time via camera in laptop. They have been introduced briefly above and some more specific details will be introduced in chapter 5 later.
There are many points that need to be evaluated in the project. For example, in the tracking and processing stage, the most important point is that we need to ensure whether the background has been removed or not because if the system cannot remove background correctly, the hand gesture will cannot be

recognized correctly and the system will not work. Another one is whether the gesture has been recognized. As if the gesture has not been recognized correctly, the relevant control command will not be executed correctly.

4. Design

The design chapter laid out the most suitable techniques for implementing the project and the conceptual designs in chapter 3 will be introduced in details in the design chapter.

In this paper, I propose an adaptive hand detection approach by using information from the camera of laptop and track the hand using vision-based method.

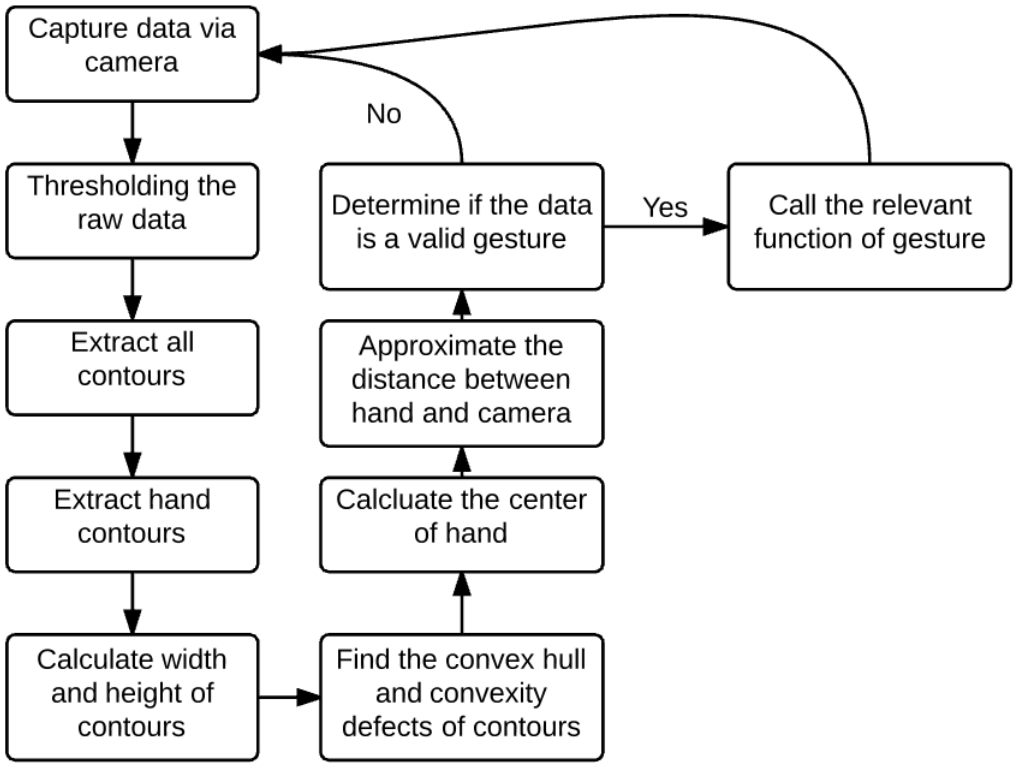

Figure 10 The process of the whole system

My system follows the process what is in the figure 10.

4.1. UI Design

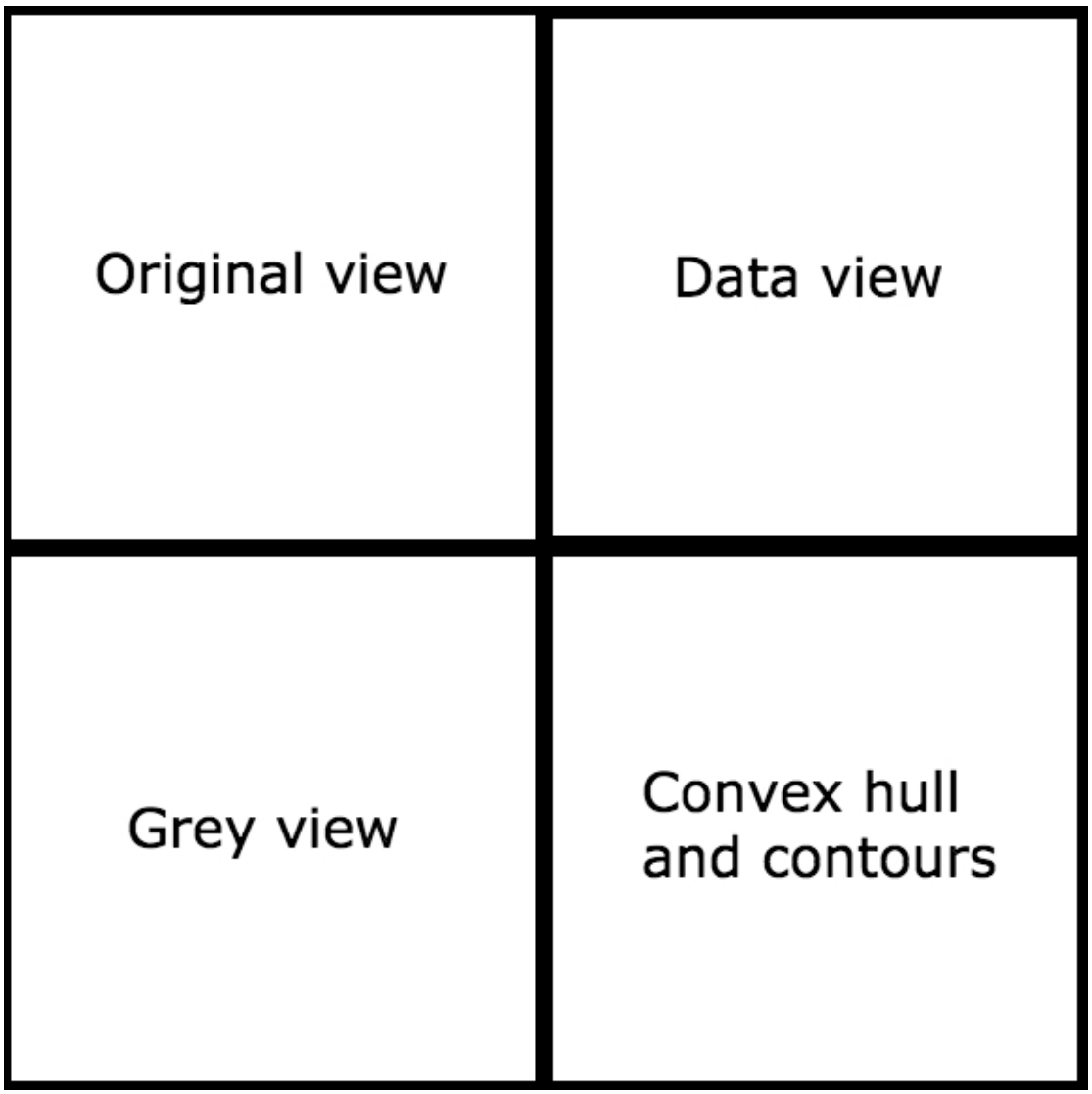

Figure 11 UI

The figure 11 shows the design for the UI of my system.

As you can see, there are 4 parts in the interface.

4.2. Data

As discussed before, the system will capture the data in real-time with camera.

And then, after collecting raw data, the data will be thresholded.

As discussed before, the thresholding is replacing in an image with a black pixel if the image intensity $I_{i,j}$ is less than some fixed constant T (that is, $I_{i,j} <$ T), or a white pixel if the image intensity is greater than that constant. In the example image on the right, this results in the dark tree becoming completely black, and the white snow becoming complete white.

In fact, the local thresholding algorithm will be used. The local algorithm is a method that adapts the threshold value on each pixel to the local image characteristics. In the local methods, a different T is selected for each pixel in the image.

In addition, the background subtraction and skin detection will also be used to improve the performance of the system. At first, the skin detection model will be applied to the raw data to retrieve the first result and the MOG background subtraction will be applied to remove all static objects and then the final result can be generated.

Generally, only hands and the head will be in the final result image. So it is easy to do some hand recognitions.

4.3. Tracking and Processing

All contours will be retrieved from the binary image. The Suzuki85 algorithm (the border following algorithm proposed by Suzuki in 1985) will be used to do this work. This algorithm is an extended version of the border following algorithm which discriminates between outer borders and hole borders and the extensions are:

1. To put a unique mark on each border rather than to adopt the same marking procedure for every border (border labelling).
2. To add a procedure for obtaining the parent border of the currently followed border.

In general, the contours are a useful tool for shape analysis and object detection and recognition. When the system retrieves all contours, it can figure out which is the hand contour, the convex hull, etc.

Then, the system will find the largest contour from detected contours and the largest contour is assumed to be the hand contour.

In addition, an approximation method will be applied to the hand contours we got to improve the accuracy of results. Actually, the approximation method just removes all curves which are too short and makes the contours straighter.

Then, the system will calculate the up-right bounding rectangle of points in every contour and find the convex hull of points in every contour with the Sklansky82 algorithm that has O (N logN) complexity in the current implementation. In addition, all convexity defects of every contour will be found.

The figure 12 illustrates the concept of convexity defects using an image of the human hand. The dark contour line is the convex hull around the hand. Each of the gridded regions (A, B, C … H) is the convexity defect in the hand contour relative to the convex hull. For a single convexity defect, there is a start point (p_s), depth point (p_d), end point (p_e) and depth length (l_d) as shown in the figure 12. We find all the points and store them in an array for further analysis.

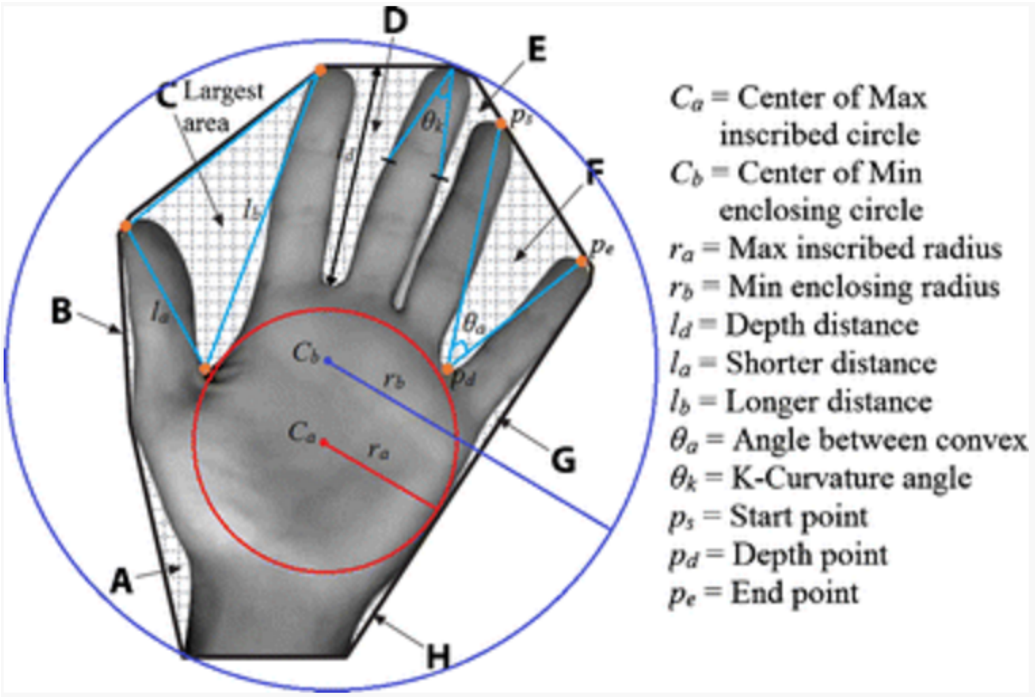

Figure 12 Convex hull and convexity defects of hand[6]

Then, the system will figure out the centre of the hand. Actually, it calculates the shortest distance of each point in the contour to the contour perimeter, and the point with largest distance is the centre of the maximum inscribed circle. This process is quite computationally intensive. In order to speed up this process, first we downscale the image and further limit the region of interest (ROI) to the middle of the contour's bounding rectangle. Finally, we only search for every N-point instead of all points in the contour. In my design, the value of N will be 4 for balanced performance and accuracy.

[6]Yeo, H., Lee, B. and Lim, H. (2013). Hand tracking and gesture recognition system for human-computer interaction using low-cost hardware. *Multimedia Tools and Applications*, [online] 74(8), pp.2687-2715. Available at: http://link.springer.com/article/10.1007/s11042-013-1501-1 [Accessed 29 Aug. 2016].

Figure 13 Limiting region of interest[7]

Finally, the system will use the radius of the inscribed circle to approximate a distance between hand and camera.

In order to get the distance between the camera and the hand, we need the camera width, the camera height and the radius of the inscribed circle.

4.4. Gesture

The system will determine if the gesture is valid. As discussed before, we have the convex hull and convexity defects of all hand contours so we could know the number of fingers with valid convexity defects. If the number is not appropriate, the system will capture new data again.

Then if the gesture is valid, it will call any relevant gesture functions. For instance, if the number of fingers is 1, the movement function will be called to move the cursor. In this system, some gestures are available now:

1. Moving the cursor with one finger
2. Scrolling up/down/left/right with two fingers

[7] Yeo, H., Lee, B. and Lim, H. (2013). Hand tracking and gesture recognition system for human-computer interaction using low-cost hardware. *Multimedia Tools and Applications*, [online] 74(8), pp.2687-2715. Available at: http://link.springer.com/article/10.1007/s11042-013-1501-1 [Accessed 29 Aug. 2016].

3. Left Clicking with a combined gesture (one finger and clench fist)
4. Right Clicking with a combined gesture (two fingers and clench fist)

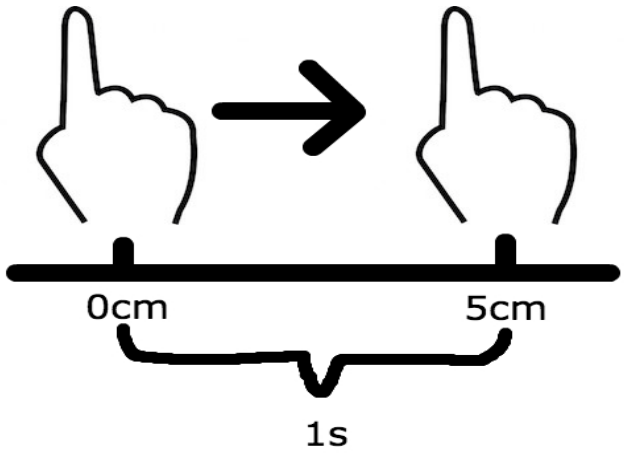

Figure 14 Hand tracking

The figure 14 shows the gesture for moving the cursor. As you can see, the speed and distance that the user moves his finger will be calculated and reflected on the cursor with an appropriate scale. The user can move his hand (stretching out one finger) to all directions. For example, if he moves his hand to right, the cursor will move to right relatively.

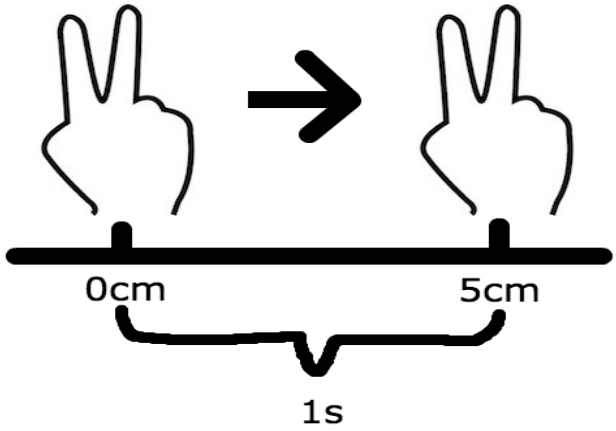

Figure 15 Hand tracking

The figure 15 shows the gesture for scrolling. The speed and distance will be calculated in this gesture, too. The user can move his hand (stretching out two fingers and two fingers must have an appropriate angle) to all directions. For example, if he is viewing some news in a website and he moves his hands down, the page will move down relatively.

Figure 16 Hand tracking[8]

The figure 16 shows the gesture for left clicking. The user can show this 'ok' gesture to the system and the control system will call the clicking function.

5. Implementation and Testing

In this chapter, we will implement this project and do some testing.

5.1. Data

The data we use to test is a real-time raw data from the camera. All gestures with a different background will be tested.

[8] D3thflcq1yqzn0.cloudfront.net. (2016). [online] Available at: https://d3thflcq1yqzn0.cloudfront.net/024565706_prevstill.jpeg [Accessed 31 Aug. 2016].

Gestures:

1. **Move the cursor**
2. **Scroll up/down/left/right**
3. **Left/Right click**

Background:

1. **Monotone white wall**
2. **Monotone wall (Other color)**
3. **Mixed background**

In addition, different illuminance will be thought during the implementation, too.

5.2. Implementation/Testing

The software will be implemented with OpenCV and Python.

5.2.1. Software UI

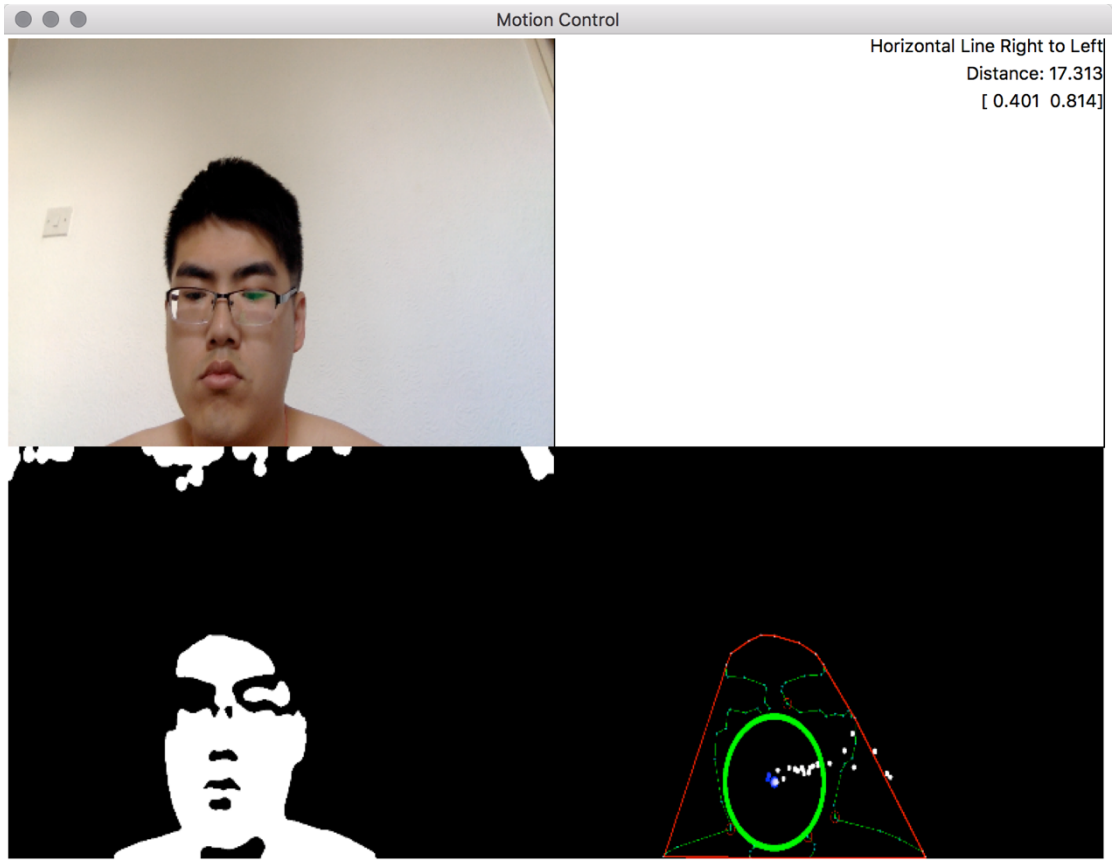

Figure 17 UI

The figure 17 is an example for the UI of my system.

There are 4 parts in the UI.

1. The top-left part is the real-time image captured by the camera.
2. The top-right part is the window to display some data of gesture.
3. The bottom-left part is the real-time grey image that processed by skin detection and background subtraction.
4. The bottom-right part is the contour and the convex hull of it. The green circle is the inscribed circle of contour and the white point is the center of it. In addition, the red line is the convex hull of the contour.

5.2.2. Gesture

As discussed before, my system will support various gestures.

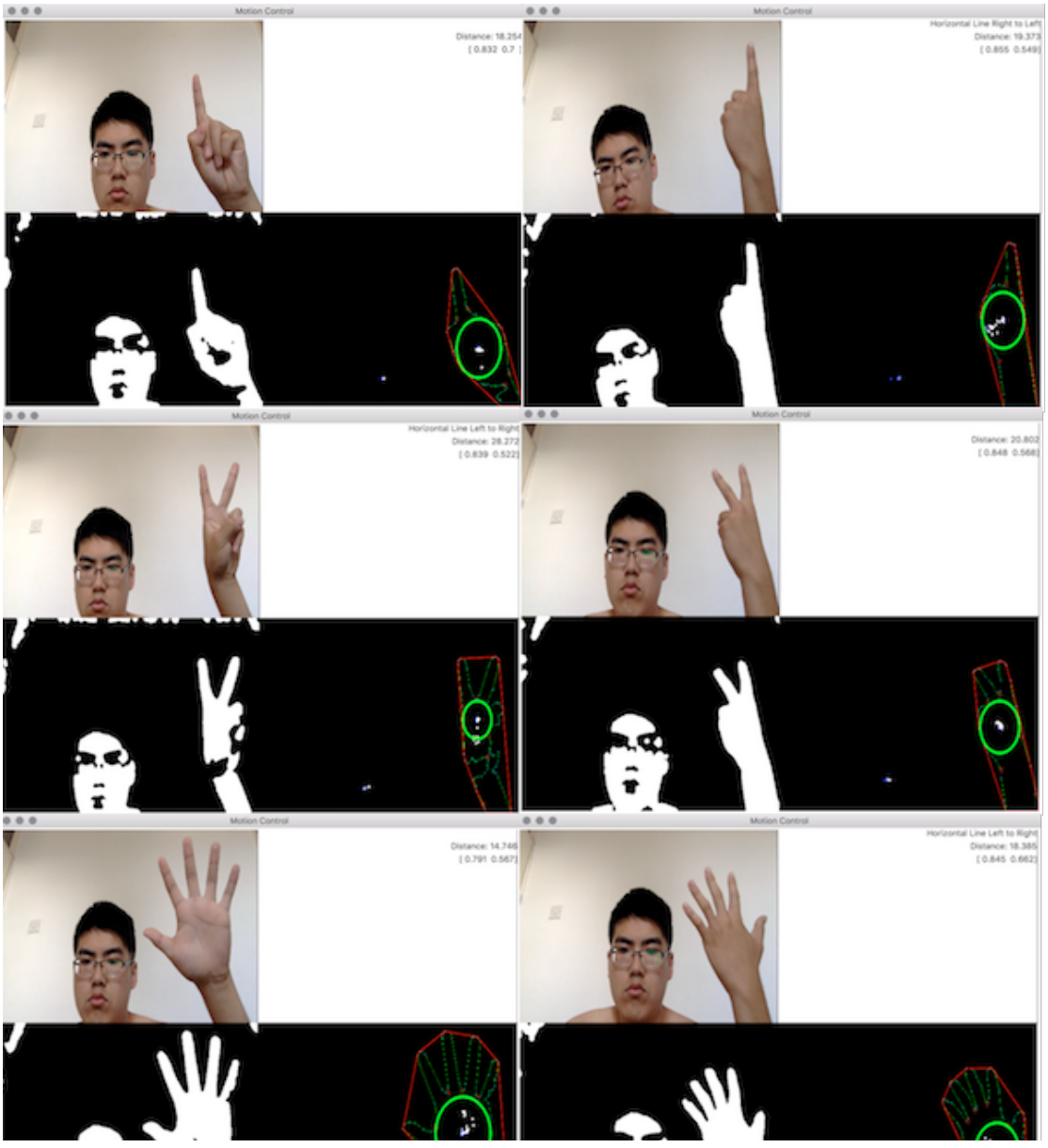

Figure 18 Gesture detection

As you can see, my system can detect both sides of the hand and gestures with any number of fingers. In addition, in the system, most gestures are classified by the number of fingers such as one finger to move the cursor, two fingers to scroll, etc.

5.2.3. Motion control

Moving cursor:

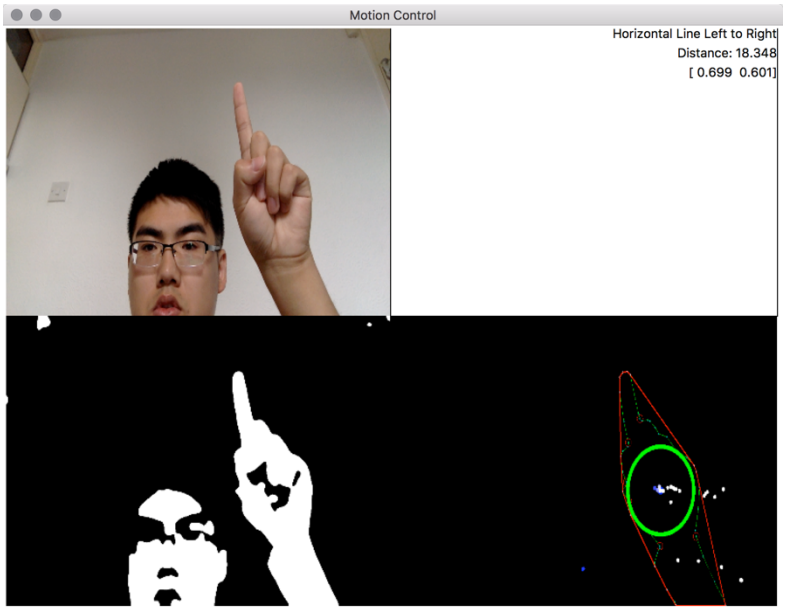

Figure 19 Before moving hand

This is the screenshot that I have not moved my hand. At first, I put my hand here close to my hand.

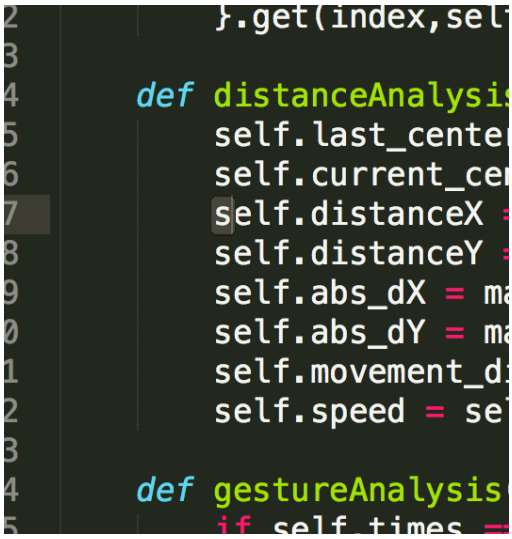

Figure 20 The state of cursor before moving hand

Because the screenshot cannot catch the cursor so I select some texts. As you can see, the figure 20 shows the cursor is in the letter - ‘s’ at the line 7 now.

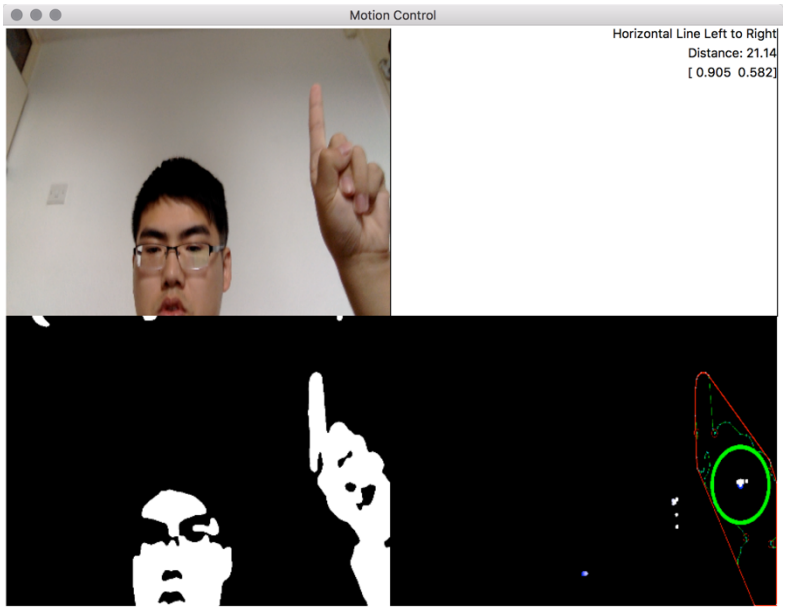

Figure 21 After moving hand

This is the screenshot that I have moved my hand. As you can see, I have moved my hand to right.

```
    }.get(index,self.moveCursor(self.distanceX

def distanceAnalysis(self, current_center):
    self.last_center = self.current_center
    self.current_center = current_center
    self.distanceX = (self.current_center[0] -
    self.distanceY = (self.current_center[1] -
    self.abs_dX = math.pow(self.distanceX, 2)
    self.abs_dY = math.pow(self.distanceY, 2)
    self.movement_distance = math.pow(self.abs_
    self.speed = self.movement_distance / 0.1

def gestureAnalysis(self, defect_points, curre
    if self.times == 0 :
        self.setDefaultData(current_center)
        self.times = 1
```

Figure 22 The state of cursor after moving hand

As you can see, the cursor is at '[0]' so it moves from the left 's' to the right as the hand moves from the left to the right.

Scrolling up/down/left/right:

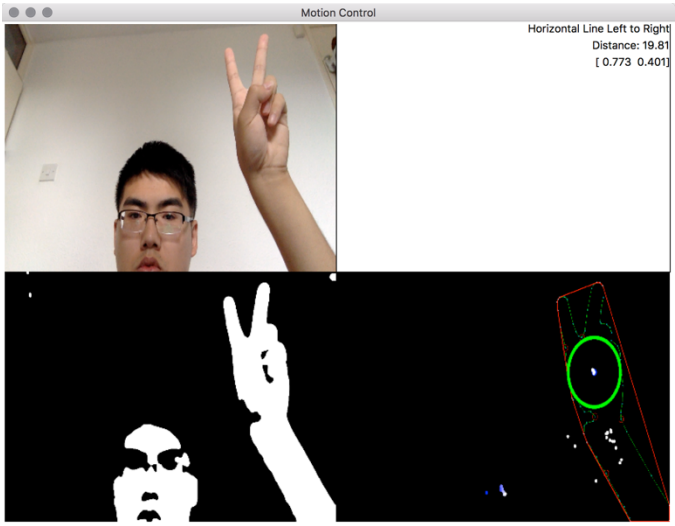

Figure 23 Before moving hand

This is the screenshot that I have not moved my hand. At first, I put my hand close to my head.

```
            moveY = self.screenHeight - self.currentMouseY
        elif estimatedY < 0 :
            moveY = 0 - self.currentMouseY
        pyautogui.moveRel(moveX, moveY, duration = duration)

    def clickMouse(self, buttonName, number):
        pyautogui.click(button = buttonName, clicks = number)

    def dragCursor(self, buttonName, distanceX, distanceY, sp
        pyautogui.mouseDown(button = buttonName)
        self.moveCursor(distanceX, distanceY, speed)
        pyautogui.mouseUp(button = buttonName)

    def scroll(self, distanceX, distanceY):
        if self.abs_dX <= self.abs_dY :
            pyautogui.scroll(distanceY)
        else :
            pyautogui.hscroll(distanceX)

    def setDefaultData(self, original_center):
        self.last_center = original_center
        self.current_center = original_center

    def reflectControll(self, index):
        return {
        '1':self.moveCursor(self.distanceX, self.distanceY, s
        }.get(index,self.moveCursor(self.distanceX, self.dist

    def distanceAnalysis(self, current_center):
        self.last_center = self.current_center
```

Figure 24 The state of page before moving hand

This is the state of page when I have not moved my hand. As you can see, page is from line 26 to line 55 now.

Figure 25 After moving hand

This is the screenshot that I have moved my hand. As you can see, I have moves my hand from up to down.

```
44
45      def setDefaultData(self, original_center):
46          self.last_center = original_center
47          self.current_center = original_center
48
49      def reflectControll(self, index):
50          return {
51          '1':self.moveCursor(self.distanceX, self.dis
52          }.get(index,self.moveCursor(self.distanceX,
53
54      def distanceAnalysis(self, current_center):
55          self.last_center = self.current_center
56          self.current_center = current_center
57          self.distanceX = (self.current_center[0] - s
58          self.distanceY = (self.current_center[1] - s
59          self.abs_dX = math.pow(self.distanceX, 2)
60          self.abs_dY = math.pow(self.distanceY, 2)
61          self.movement_distance = math.pow(self.abs_d
62          self.speed = self.movement_distance / 0.1
63
64      def gestureAnalysis(self, defect_points, current
65          if self.times == 0 :
66              self.setDefaultData(current_center)
67              self.times = 1
68          else :
69              self.distanceAnalysis(current_center)
70              self.finger_number = len(defect_points)
71              #if self.speed != 0 :
72                  #self.reflectControll(self.finger_nu
73
```

Figure 26 The state of page after moving hand

This is the state of page when I have moved my hand. As you can see, page is from line 44 to line 73 now. Page is actually scrolled down.

6. Results and Discussion

6.1. Results

I invited 20 classmates to test this system and the results have been displayed with tables below.

Test scene 1:

The requirement for a gesture	If meet	Testing results (Success/All)	Successful percentage of results	Background and illuminance
Moving the cursor	✔	19/20	95%	The white monotone wall and the sunny day
Scrolling up/down/left/right	✔	17/20	85%	The white monotone wall and the sunny day
Left/Right clicking	✔	20/20	100%	The white monotone wall and the sunny day
Dragging the cursor	✖	0/0	0%	The white monotone wall and the sunny day

Figure 27 Testing results

In the test scene 1, as you can see, the background and the illuminance of every test room is the same.

For moving the cursor: This function has been implemented and 19 of 20 people can use it successfully. Another one is too far away from the camera so the system fails to capture his hands.

For scrolling: This function has been implemented and 17 of 20 people can use it successfully. Another two people fold their fingers so the system recognizes their gesture as a gesture of the moving cursor (one finger). The other one is too close to the camera so the system fails to recognize the hands.

For clicking: This function has been implemented and 20 of 20 people can use it successfully.

Test scene 2:

The requirement for a background	If meet	Testing results (Success/All)	Successful percentage of results	The testing gesture and illuminance
Monotone white wall	✔	19/20	95%	Moving the cursor and the sunny day
Monotone orange wall	✖	0/20	0%	Moving the cursor and the sunny day
Mixed background	✔	9/20	45%	Moving the cursor and

				the sunny day

Figure 28 Testing results

In the test scene 2, as you can see, the testing gesture and the illuminance of every test room is the same.

For monotone white wall: 19 of 20 people can use it successfully. Another one is too far away from the camera so the system fails to capture his hands.

For monotone orange wall: 0 of 20 people can use it successfully. The colour of wall is too similar to the colour of skin and the skin detection is used in the system so the system cannot capture hands.

For mixed background: 9 of 20 people can use it successfully. The lighting has been reflected by the orange wall so sometimes the system will fail to recognize the hand.

Test scene 3:

The requirement for a illuminance	If meet	Testing results (Success/All)	Successful percentage of results	Background and the testing gesture
Brightest sunlight	✔	13/20	65%	White monotone wall and moving the cursor
Bright sunlight	✔	19/20	95%	White monotone wall and moving the cursor
Shade illuminated by the entire clear blue sky at the midday	✔	19/20	95%	White monotone wall and moving the cursor
Typical overcast day at the midday	✔	14/20	70%	White monotone wall and moving the cursor
Dark room	✖	0/20	0%	White monotone wall and moving the cursor

Figure 29 Testing results

In the test scene 3, as you can see, the testing gesture and the background of every test room is the same.

For the brightest sunlight: 13 of 20 people can use it successfully. The lighting is so too bright so the system has been disturbed and it cannot capture the hands properly.

For bright sunlight: 19 of 20 people can use it successfully. This illuminance is appropriate for the system so the system almost recognizes hands properly.

For shade illuminated: 19 of 20 people can use it successfully. This illuminance is appropriate for the system so the system almost recognizes hands properly.
For the typical overcast day: 14 of 20 people can use it successfully. The lighting is a little dark so the system cannot recognize some hands with a deeper color.

For the dark room: 0 of 20 people can use it successfully. The lighting is too dark so the system cannot capture any hand at all.

6.2. Evaluation

At this version of the system, there are some problems now.

1. The user should use the system close to a monotone white background now and I will modify the background subtraction algorithm to improve the scope of the system's valid recognition later.
2. The user should use the system with an appropriate illuminance. 20000 lux is the most appropriate illuminance now. This level is about a shade illuminated by entire clear blue sky at midday.
3. Just few gestures can be supported. Many more gestures can be added to this system later.

4. The system is unstable. Sometimes the program will crash or the system will run slowly. The code can be optimized to improve the performance of the system.
5. The algorithms to remove the noise is not good enough so sometimes much noise will affect the recognition of the system and I will modify it later.
6. Sometimes the cursor will jump everywhere because of much noise. I will implement an exception checking module to avoid this situation.

7. Conclusion

The project aims to design a software platform to allow people to interact with a personal computer that almost everyone will have and work with it by using body actions and makes it more portable for every user.

The process of developing this system is not smooth and there are several problems during it. For example, the field of image processing is new for me and the process of studying it cost me much time. In addition, many algorithms are difficult so I have to focus on learning them. However, it is easy for me to use OpenCV and program with Python. I have implemented some fancy functions in the system such as the system can track real-time speed and the moving distance of the user's hands, the system can detect both sides of a hand, etc.

For development of this system, there are 6 chapters in the document for it.

The chapter 1 gives a briefly introduction for this project and states the goals of the development of this project.

The chapter 2 states some relevant technologies of gesture recognition and mouse simulating.

The chapter 3 involves requirements gathering and analyzing. In addition, some testing scenes have been described in this chapter.

The chapter 4 describes some designs for the project. Many ides have been considered in this chapter to get more desirable results. In addition, it is convenient for making a fancier UI to use the OpenCV.

The chapter 5 states some details of the implementation process and in the chapter 6, all testing scenes discussed in the chapter 3 have been implemented and the results is satisfied.

To summarize, an interactive software system was developed and it is easy and portable for users to control the personal laptop.

Reference

[i] Utsumi, A., Miyasato, T., Kishinoand, F. and Nakatsu, R., (1995). Real-time Hand Gesture Recognition System. *Proc. of ACCV '95*, pp. 249-253.

[ii] Pradipa, R. and Kavitha, S., (2014). Hand Gesture Recognition – Analysis of Various Techniques, Methods and Their Algorithms. *International Journal of Electronics and Computer Science Engineering.*

[iii] Aarti, M. and Ruchika., (2013). Gesture Technology: A Review. *International Journal of Electronics and Computer Science Engineering*.

[iv] Garg, P., Aggarwal, N. and Sofat, S., (2009). Vision based hand gesture recognition. *inProc. World Acad. Sci., Eng. Technol.*, pp. 972–977.

[v] Neumann, J., Fermuller, C. and Aloimonos, Y., (2002). A hierarchy of cameras for 3D photography. *in Proc. Int. Symp. 3D Data Process. Vis. Transmiss*, pp. 2–11.

[vi] Manresa, C., Varona, J., Mas, R. and Perales, F., (2000). Real-Time Hand Tracking and Gesture Recognition for Human-Computer Interaction. *Electronic Letters on Computer Vision and Image Analysis*, pp.96-104.

[vii] Stenger., (2006). Template based Hand Pose recognition using multiplecues. *7th Asian Conference on Computer Vision*.

[viii] Yinghui, Zhou., Lei, Jing., Junbo, Wang. and Zixue, Cheng., (2012). Analysis and Selection of Features for Gesture Recognition Based on a Micro WearableDevice. *International Journal of Advanced Computer Science and Applications.*

[ix] Dardas, N.H., and Petriu, E.M., (2011). Hand gesture detection and recognition using principal component analysis. *IEEE International Conference.*

[x] Oka, K., Sato, Y. and Koike, H., (2002). Real-time tracking of multiplefingertips and gesture recognition for augmented desk interface systems. *In IEEE International Conference on Automatic Face and Gesture Recognition.*

[xi] Yang, T., Xu, Y. and A., (1994). Hidden Markov Model for Gesture Recognition. *Carnegie Mellon Univ*.

[xii] Ziaie, P., M¨uller, T., Foster, M. and Knoll, A., (2000). A Na¨ive Bayes Classifier with Distance Weighting for Hand-Gesture Recognition. *Robotics and Embedded Systems.*

[xiii] Black, M. and Fleet, D., (2000). Stochastic tracking of 3D human figures using 2D image motion. *Sixth European Conference on Computer Vision*, pp.702-718.

[xiv] Pyautogui.readthedocs.io. (2016). *Welcome to PyAutoGUI's documentation! — PyAutoGUI 1.0.0 documentation*. [online] Available at: https://pyautogui.readthedocs.io/en/latest/ [Accessed 29 Aug. 2016].

[xv] Suzuki, S., (1985). Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, *30*(1), pp.32-46.

[xvi] Arcsoft.com. (2016). *Global leader in imaging technology - ArcSoft*. [online] Available at: http://www.arcsoft.com/index.html [Accessed 29 Aug. 2016].