

Recognising Human Motion and Gestures in Video

A DISSERTATION SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE BSC (HONS) COMPUTER SCIENCE

AUTHOR: [REDACTED]

SUPERVISOR: JOHN DARBY



DECLARATION

No part of this project has been submitted in support of an application for any other degree or qualification at this or any other institute of learning. Apart from those parts of the project containing citations to the work of others, this project is my own unaided work.

Signed





Abstract

This is a project that aims to build a system that can recognise human gestures from a live camera feed. Those gestures will then be shown on the display in order to judge their accuracy. There have been many attempts at integrating videos and real life, a notable example is the Kinect. This projects shows that the ability to implement that, albeit a simple alternative can be done in an accurate manner even on a machine with non-dedicated hardware.

Contents

Abstract.....	2
Introduction	4
Background	4
Assessment of the problem	5
Aim:	5
Objectives:	5
Report Chapters	5
Literature review.....	5
Terminology techniques employed	6
Similar studies of human gesture recognition.....	8
Design.....	9
Gestures to Be Implemented	10
The fist.....	10
The Wave (open palm).....	10
The Point	10
The Peace Sign	10
The Rock Sign	10
Design steps	10
Methodology.....	11
Implementation	11
Justification pieces of code	18
Testing and evaluation.....	19
Processing issues:.....	19
Light variations.....	19
Hair colour.....	20
Facial hair	20
Other objects in room.....	21
Skin tone	21
Clothing	22
Glove colour	24
Glove Type	25
User with a different hand size.....	26
Other areas of testing	27
Processor usage	27
Memory usage	28

Accuracy	28
Code rectifications	29
Adjustments to the HSV Values:	30
Limit to the angle between fingers:	30
Adjustment to the BRG Values:	30
Background decided:	30
Erosion and Dilation values amended:	30
Adjust light levels:	31
Conclusion.....	31
References & Bibliography.....	32

Introduction

Background

“For reactive computers, embodied robots, or artificial creatures to effectively interact with us, it will be necessary for them to discern the various types of human movements that abound in the world. “ (J. Davis)

In the advent of humans and computers becoming more and more integrated, it is important that the computers are able to know what the human is doing. For example, a car should know when someone is nearby to avoid hitting them, but also be aware if someone is signalling it. Alternatively, a deaf person who is signing to a camera could have his or her gestures decoded and transcribed for someone who cannot sign to understand.

There are millions of humans with cameras, or with access to them. There are many potential things that can be improved upon by being able to track their own movement. People are able to see what to do in the gym, and when they then go on to mimic those lessons can see how accurate what they are doing is. There can be people who are learning how to perform tasks that are very intricate, such as playing the piano. Being able to track a person’s finger movements when playing the piano could allow someone who does not know how to play the instrument that well, or at all, the ability to follow along, or learn faster through an ability to determine mistakes and areas for growth.

Virtual and augmented reality both are heavily reliant on motion tracking. To have a program that recognises what gestures someone is making can make it much easier to augment the surroundings and produce different outcomes for each different movement the person makes.

This shall be a project of image manipulation in order to extract movements made, and differentiate between them in order to know what action is being performed. This can be anything as simple as a fist, an open palm, or even a peace sign.

Assessment of the problem

Aim:

The program shall read an input from a video and then display what the person within the video is gesturing. This could be waving, pointing or another gesture. The program will use thresholding to remove any stationary objects and focus on the person, and then after analysing the relative positions of the figure, show what is being done.

Objectives:

After researching into what has been done in the field already. There will be a further exploration of ways to remove clutter from videos. This shall be done in order to prevent clutter from affecting the mapping of the individual.

The programming of the different gestures shall then be programmed, with the specific selected being listed and clear to differentiate between. When doing this the most and least complex gestures shall be assessed, and the ability to add them in time shall be considered.

The aim is to study the OpenCV library and extract the relevant data from the video to recognise the gestures.

The different gestures shall then be added into the code. Research shall be conducted into the different methods of doing so, to determine which is best for this project. Whether that be tracking the points on the body as they move or identifying the differences in the frame, ensuring the calculations of the furthest points are relative to the ratio of a person, and identifying the gestures that way.

Researching and finding previous variations of this project that have been done in order to come up with a reasonable accuracy rate and a visualisation that is friendly to the end users shall also be done.

A series of tests shall then be ran, using the data and variables that have been acquired in order to test whether my program is able to read all of the gestures and from multiple videos.

If it is found that the accuracy rate is not that high, then refining of the program using the test data shall take place in order to ensure the most accuracy.

Report Chapters

Literature review

Terminology techniques employed

As a very similar method to solving this problem has been done before; which shall be followed loosely in this project, the main steps that were done by another team (S. ANDRESEN)

First of all there will be a need to subtract the background of the live feed. There are a few ways that this can be done, with varying success levels. The main way to do so is with the RGB colour levels, however there are some methods involving the HSV colour channel that have proven equally, if not more effective (D. BORA¹, A. GUPTA, ET AL). The HSV channel shall be used as the previous study encourages confidence that the difference in results that will be yielded is more than negligible, and not a change in processing power needed. In order to process the background either each channel can be done separately and then added together to make a final layer, or they can all be processed together for slightly less accuracy. Depending on the time it takes to do so, the method that shall be implement will be to process each colour channel separately if time allows.

The input will need to differentiate between the clothing the subject is wearing and the skin. This is so that the hand can be extracted from the image and not the entire arm. Due to this there needs to be a skin colour extraction done also. This will be done using the HSV values or thresholding, as that could remove much more from the program. It will need to be considered which clothing colours the program user anyone else involved in the testing process uses. The skin colour constraints will also be a large factor. As skin types are so varied in the area in which I am testing, if it was wanted to enable many others to participate then there would need to be a way to allow the range of skin tones to be larger than just a few people; which is why the option to have the test user have a black glove appeals. During this project the test shall be performed on a human wearing a black glove. This will enable anyone to be able to use the program, without much need to have much worry about skin tone.

The YCrCb model will "reduce the effect of illumination conditions and segment hand gesture from complex backgrounds or different illumination conditions." (Z. QIU-YU, L, JUN-CHI ET AL). This was one of the reasons that colour scape was considered, as it is particularly imperative as I there can be no background subtraction without a background reference picture in the chosen software. Depending on where the user is there will be different lighting and background environments also. Through using this black glove method the aim is to increase the accuracy of the program output and thus have those lighting and background variations not be the cause of error.

In order to show the extracted parts of the image in clarity, there needs to be contour extraction (BRADSKI, G.), which will enable the sight of all of the fingers and rest of the hand outlined clearly. This works by finding the area of the image that outline a chosen area, and forms a shape, if there has been processing on the image to reduce items in the background, it would be a very simple case of outlining the hand and being able to generate variables from that point.

Through the use of thresholding, there is ability to remove a lot of the noise from the video feed and make the outlines and area recognised as the hand even clearer and less prone to error (BRADSKI, G.). Also, if required, erosion and dilation techniques shall be performed to further refine the output .The outlines of the hand need to be as clear as possible in order to be able to tell the variation in

the movements. Another tool that can be used is Gaussian smoothing. This will clear many imperfections and remove many parts of the background in order to make the foreground easier to separate from the background. The image displayed could be in any of the states mentioned above.

Black objects will need to be removed out of the room unless they are small. This is because the way in which any thresholding would work would allow the black objects to stay in detection alongside the glove, only showing the biggest contour is an option, however if there are objects bigger than the hand in relation to the video then the program would still not function accurately.

As the face is a large object and a very similar skin tone to the hands, it is worth detecting and removing it in order to just have the hands being processed for gestures [15]. The face being removed will prevent errors and issues found in previous work where the face is larger than or just as visible as the hands and is mistaken as one or confuses the program. This will also allow for a cleaner look on the output.

A step that will be taken will be to find the convex hull in order to get the shape of the hand. After that is done the convexity defects will also be found. K-curvature is a method of tracking the movements of the fingers. It will then be possible to tell which position each finger is in relative to the others and enable the ability to find differences between the gestures. [3] There are other methods in various libraries that find the similarities in a shape. This is not affected by the size of the shape and so is integral in ensuring the gestures that are the same, but in different positions will be recognised. An example would be holding up an index finger, if it were that the person rotated their hand the gesture would still be the same, and the program should be able to recognise that, rather than need to have all potential angles and distance from the screen programmed in separately.

One of the teams (H. YEO, B. LEE ET AL) found that the maximum inscribed and the minimum enclosing circle. This will enable the gestures of closed fist, and claw-like gestures whereby the difference is the distance of the fingers from the centre of the hand are the main differences to be recognised clearly.

In order to determine which hand is which, there will be a need to inspect the concavity defects. Assuming that there are fingertips, from the largest defect, which will be the thumb, it will then be possible to gather which hand it is by checking the vector.

The next part of the task will be recognising what the gesture is. For that there are two methods being debated

The first is to take pictures of a hand or use stock black and white images from a data set, with each picture doing a different gesture in order to have a complete image set to reference each hand movement to in the live feed. This would be trained using MATLAB in order to predict the next gesture. With this approach there can be as many gestures as possible and also many things can be added if there is an update to the pictures being stored in order to increase the accuracy of the read.

The second option is to evaluate the shape of the hand and outlines, along with the rest of the data in the output in order to determine which characteristics are there and match to the closest gesture. This approach will be harder to implement however it will also allow an unlimited amount of gestures to be input and added/amended, and also the amount of memory being used in the system

will be a lot less as there will be no images being loaded into the RAM slowing the program down, and also this would allow for the program to be much more adaptive and faster to change if there was an issue with a particular gesture being read in.

The final and most difficult part of the project will then be finding a way to deal with the movement of the hands to identify movements as well as gestures. This could be based on a certain amount of time and taking into account the difference in the movement between the pixels in the current state and the previous state, to an amount of maybe 100 frames. This difference will then show a diagram particular to the individual movement which could then be added in in a similar fashion to the gestures in order to incorporate waving and clapping, and potentially the clicking of a thumb and index finger. (J. DAVIS)

By doing this the software would become a lot more interactive, however it is reasonable to predict that time constraints may prevent the final part from occurring, as it will require additional methods to the ones mentioned earlier to learn.

Similar studies of human gesture recognition

There was a previous group that did the project in the same way that this project is being intended [1]. The main difference was that this group had designed the software with the intentions of using the Kinect in mind, as well as with a camera. Their paper is being used as the basis of this project. The main differences will be that in this project a different set of colour filters shall be used due to the difference in hand colour filtration. .

Also was the case of a Han Sung, who on a YouTube video produced a piece of software to enable finger drawing using OpenCV. This will be relevant to the last part of the work as the ability to remember where the gestures were previously and have the program use that to write, will be implemented in a way where it can be mapped to gesture movements as a whole and tell which is which.

There has also been another person who has completed a piece of software with similar aims (D. Michal). This person's software recognises the entire body as well as identifying the hands and finger movements. The speed and accuracy at which the software recognises the different gestures and hand signs that the person is making is something which is aimed to be achieved in this task. In the case of recognising the gestures; there was a study on recognising human emotion through body movement. In order to work out which movement was which, those doing the study use image flow diagrams (G. CASTELLANO, S. VILLALBA, ET AL)

"The SMI at frame t is generated by adding together the silhouettes extracted in the previous n frame and then subtracting the silhouette at frame t . " [9] This will enable me to get each gesture and visualise what each gesture is in order to pair it with either a matching image I have loaded into memory or a set of conditions that are particular to the gesture.

That one of the options that could be taken. The other is a complete mathematical representation of the fingers and the angles and vectors used shall be used to calculate the hand position. The latter option will be more of a complex process, however it will not be very processor intensive.

Design

The user shall need a camera on their device in order to run the software. They will also need to be in a well-lit room with an empty background in order to optimise results (as shown in subsequent sections)

Due to the fact that there will be a live video feed, the feed will be shown on the display with the modifications overlaid onto the image. The overlay will consist of multiple attributes.

The area in which the hand is shall be encased in a bounding box. This shall be the area in which the modifications and calculations take place. This shall serve as a visual cue to the user and enable them to know that their hand is being accurately visualised and also enable them to know if any other areas of the display are being modified or affecting the final result.

The box shall be a bright blue colour in order to be sharply distinguished from the glove.

As the background of the feed shall be a plain light colour, it is important for the video to not be filmed in a blue room as the box would be difficult to notice.

The hand wearing a black glove shall have the fingertips to the centre of the palm drawn over, and also have the points in between the fingers shown as well.

The centre of the palm shall have a dot, representing the core from which all calculations can be done. This will enable the wearer of the glove to visualise where the computer believes the centre of their palm is and make adjustments to the way in which they have angled their hand accordingly.

From the centre of the palm shall be lines going toward the fingertips. They shall represent the fingers that the computer is able to distinguish. The lines shall lead to a dot in the centre of the fingertip. From this point to the centre, each finger can be represented as a vector, and the angles between each finger can determine the gesture that is being made.

The thenear space shall also be plotted, and from such, there can be an accurate visualisation of all of the fingers. In order to work out which is the thenear space the areas that

Along the bottom of the display shall be text which will display the gesture that the user is doing.

Gestures to Be Implemented

The fist

When there are no fingers to display, the majority of the overlay shall disappear. The centre of the palm shall still be visible but as there are no fingers to display all that shall be shown shall be the bounding box and the area around the hand.

The Wave (open palm)

When all five fingers are out the gestures will be seen as a wave. All of the fingers shall have lines drawn towards the centre.

The Point

When there is only one finger out, the output shall be that the user is pointing. This shall be the case with any singular finger.

The Peace Sign

When there are two fingers side to side, the output shall state that the user is showing the peace symbol.

The Rock Sign

This shall encompass two different gestures that overlap. When the user has the thumb, index and little finger, or when the user has just the index and little finger out, this shall be seen as the rock sign.

There shall be no gesture for when the user has four fingers out. This is due to the fact that there are not many popular four fingered gestures outside of sign language. (J. FENLON, A. SCHEMBRI ET ALL)

Design steps


The way in which this shall be done is first through reading in the live camera feed. As the program shall be made in Java. The OpenCV libraries shall be used due to the large amount of image processing elements that it contains.

Then, the image shall be eroded and dilated. This is to remove excess noise in the image and make it easier to manipulate and find large contours.

The image colour scope will then be changed to HSB values in order to extract the glove from the rest of the background. If there were many objects of a similar colour to the black glove, then it would be advisable to change the colour to one that is not common in the area of use. As the background in the testing room was white, this allowed for a much easier image to visualise and to filter out of the image. The edges were very easily detected also, and so not much modification or a small range was needed in order to extract the glove. If a human hand was used, and the HSV values were implemented instead, then there would still have been a further issue with separating the hand from the arm.

The convexity defects will be found as the fingertips shall be necessary in order to calculate the gestures. At the same time the centre of the palm shall be calculated and the areas in between the fingers. This will enable a clear vector image of the hand in real time.

The fingers then will be calculated, their lengths and angles between shall be used in order to determine the gesture.



The gestures of the hand shall be programmed with clear parameters that shall be triggered when the points in the hand, defects, amount, and angles fulfil the conditions.

The text shall then update along the bottom of the display. It shall change with each movement of the hand, and when a condition is not met, shall either remain blank, or state to “Make a gesture”.

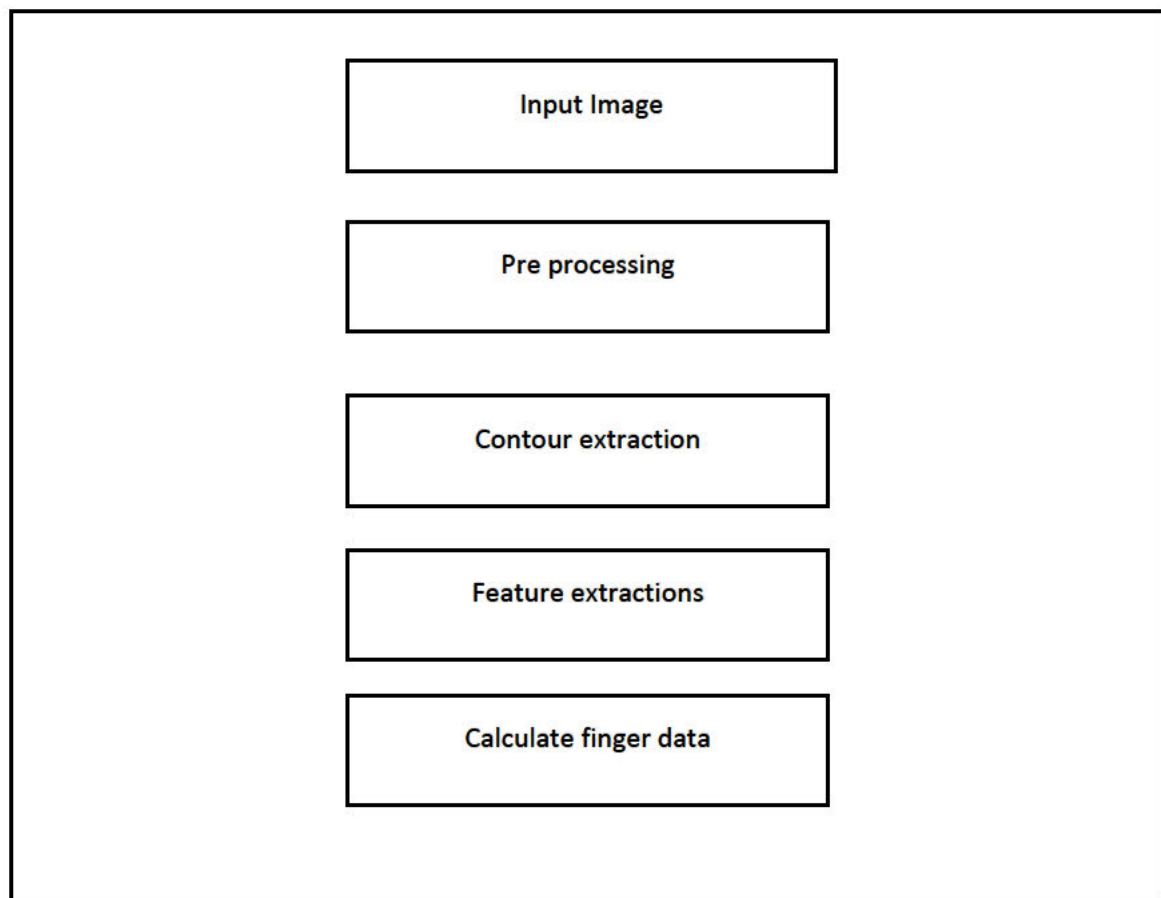
There shall be no buttons and such the program shall not have values that the user can modify, nor add.

Methodology

The methodology used was Kanban. This was due to the continuous updates and amendments that were needed to be made to the code. The use of this methodology enabled a very high response to change, and so when a particular change did not work, the ability to rectify that issue was there. The time spent on redundant features was also minimised, as the methodology was optimised for smaller groups, or individuals.

Implementation

Order of processing



First the OpenCV library was installed into the Java SDK.


The program frame was determined. As the camera was almost 1080p, a frame of 960 x 768 was set in order to not stretch the image quality.

The window name was set so that the user would be able to identify the program amongst others that were open in the system.

The camera was then opened and the feed went into a buffer. This enabled the camera frames to be modified as images.

The HSV (Hue, Saturation and Value) filter was then applied to the image. This was in order to filter out all of the areas of the image that did not satisfy the condition. This left the black glove in the image. Whilst the rest of the image was visible, the only area in the values to be modified was the one that satisfied the condition.

A skin detection method was implemented to ensure that the skin did not interfere with the results. This was particularly important if the skin of the user was dark. This is to avoid potential overlap where the arm would be seen as an extra finger. It is also following the method that has been shown effective (S.Perna) in prior testing.



The values were (50, 255, and 35). As the glove was black the hue needed to be low, however it was of note that the variations in the shade of black due to being an item of clothing allowed for slight change in hue between different black gloves. The saturation was high as the colour was fully saturated, and not pale. The value was set to 35 as the variations of the light in the room meant that the glove would never be black in the daylight, and in absolute darkness no image would be able to be taken.

The convexity defects were then found. This was done by getting the contour of the image, which was the edges of the glove, and then finding the peaks in order to make out the fingertips. The convex defects were found by taking the smallest values from the contour. This gave an accurate image of a hand, or whatever shape the hand was in. A condition that had to be fulfilled was that there had to be no more than 5 fingers. This is to avoid a system overload if there were too many dark areas in the room that the program believed were areas of the glove and further convex points.

As the hand shape was now fully simulated, the centre was then able to be calculated. The fingers were also able to be calculated, along with which was which. Due to the fact the thumb is the shortest, it would be closest to the centre. This enabled the knowledge by number, if there were five fingers, of which was which. This, however, would not do the same for cases where there were less and so had limited applicability.

The fingers were drawn onto the hand along with the centre, and convex points.

A bounding box of the hand was also drawn. This was calculated around the convex hull. The shape of the box changes in accordance to the size of the hand. This made it very easy to notice when the fingers were not being accurately portrayed and also when the angle of the hand in relation to the camera was not optimal. This is because as the user shifted the degree of their hand, the box became bigger or smaller.

The colours were chosen according to how well they would stand out on the black glove. Having bright fingers enabled them to be vibrant and clear to see.

The gesture was then calculated through trigonometry.

```

// Calculate the distance between the fingertips and centre of palm
public double calculateDistance(Point P1, Point P2) {
    return Math.sqrt(((P1.x - P2.x) * (P1.x - P2.x)) + ((P1.y - P2.y) *
(P1.y - P2.y)));
}

//Calculate the angle between fingers
public double calculateAngle(Point P1, Point P2, Point P3) {
    Point vertice1 = new Point();
    Point vertice2 = new Point();
    double angle = 0;

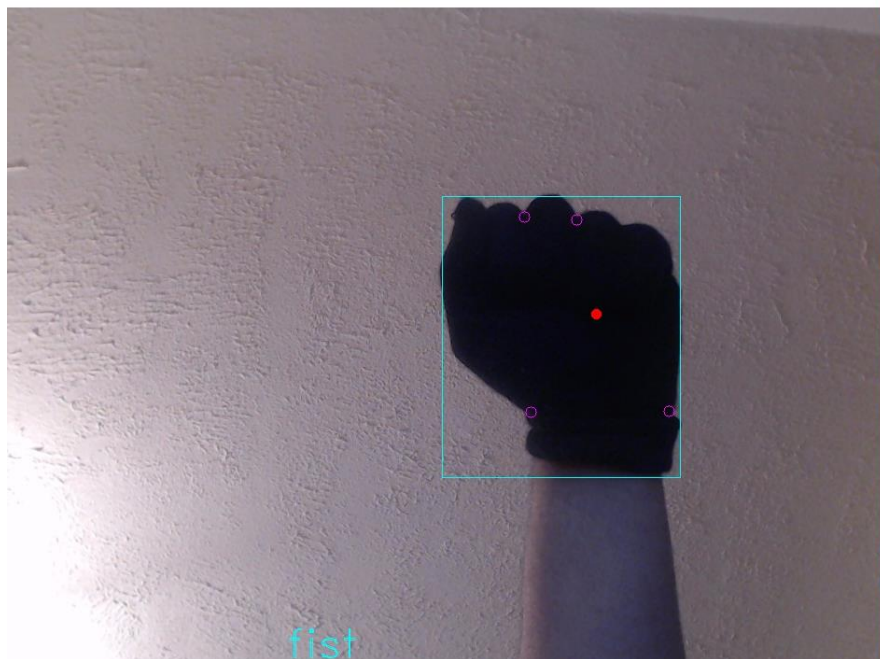
    vertice1.x = P3.x - P1.x;
    vertice1.y = P3.y - P1.y;
    vertice2.x = P3.x - P2.x;
    vertice2.y = P3.y - P2.y;

    double dotProduct = (vertice1.x * vertice2.x) + (vertice1.y *
vertice2.y);
    double length1 = Math.sqrt((vertice1.x * vertice1.x) + (vertice1.y *
vertice1.y));
    double length2 = Math.sqrt((vertice2.x * vertice2.x) + (vertice2.y *
vertice2.y));
    double angleAcos = Math.acos(dotProduct / (length1 * length2));
    angle = angleAcos * 180 / Math.PI;

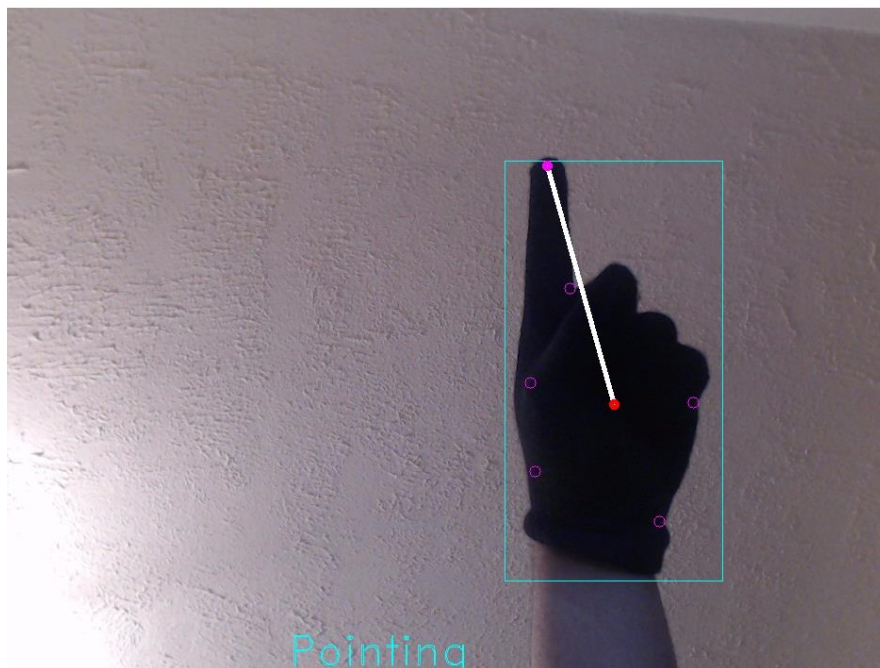
    return angle;
}

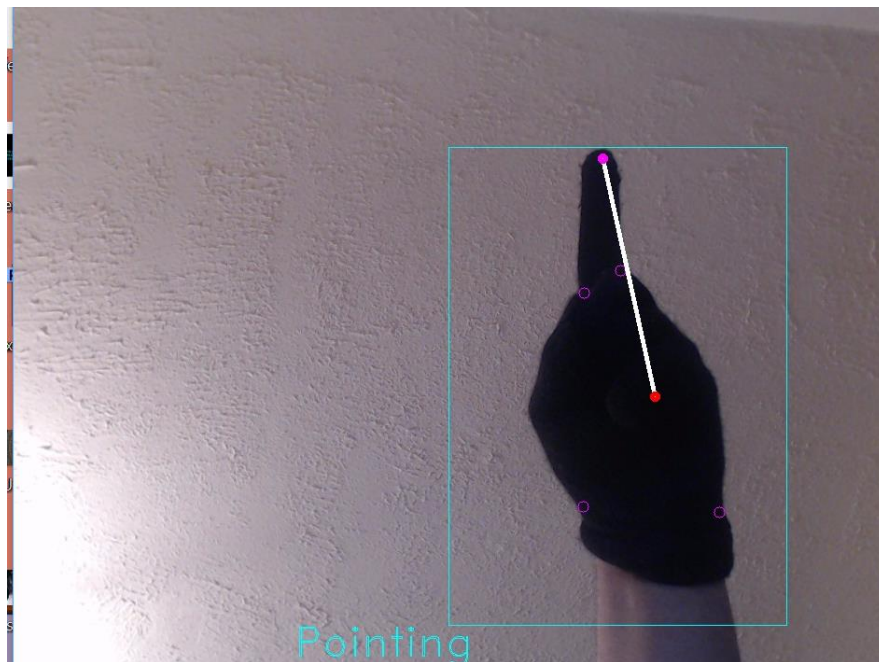
```

The “Fist” conditions were satisfied when there was a black glove with no fingers. The size of the fist did not matter, as long as there were no visible fingers being shown.

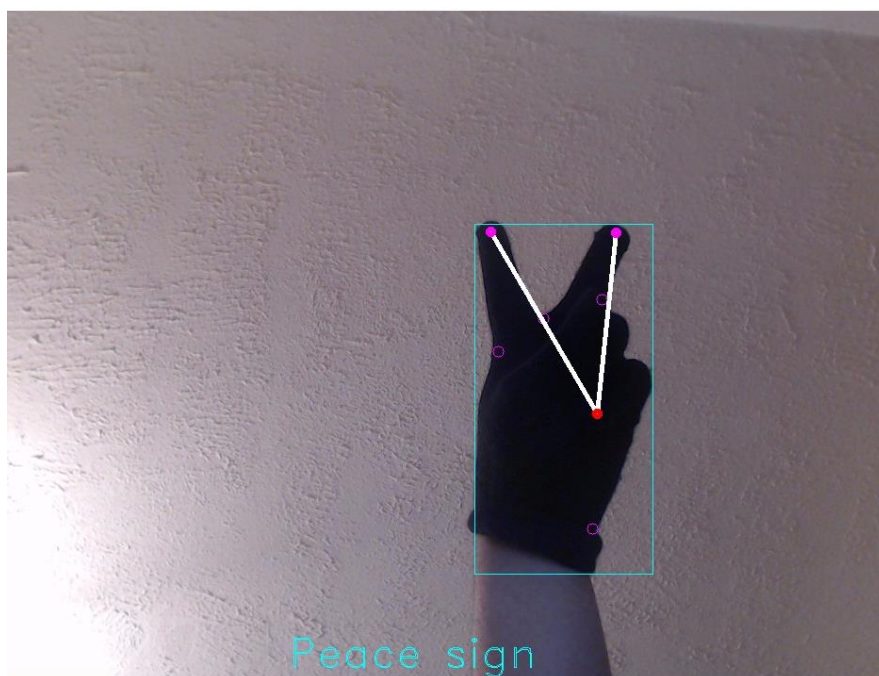


Secondly, was the “Pointing” gesture. This condition had to be satisfied when any finger was being pointed, and so had a simple condition, once the length of a finger was long enough, or the fist was no longer just a fist and had a digit rising from it, the condition was satisfied.



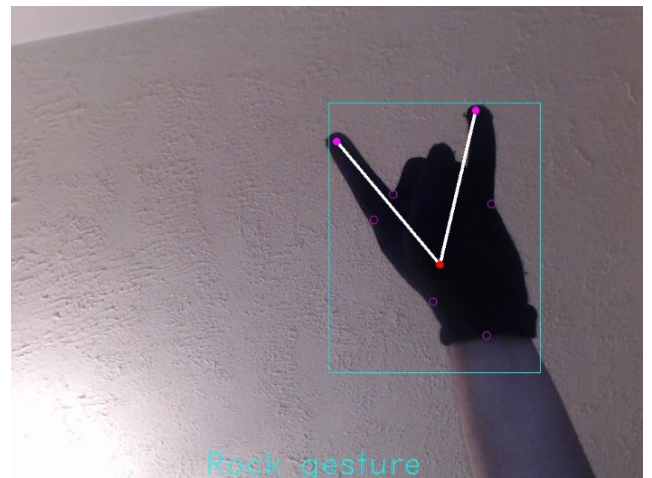
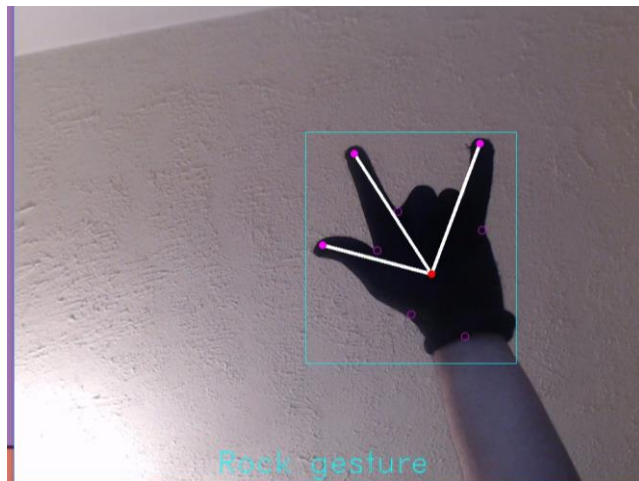


The third was the “Peace sign”, as this is two fingers side by side, it was necessary for there to be two fingers require, and the angle of the two not too large for the condition to be satisfied by singers further apart. This was tested through slowly lowering the acceptable angle between the fingers until and accurate representation of the peace sign was achieved. It shall be discussed in the testing and evaluation section however, that depending on the user and the thickness of the glove, the angle cannot be entirely accurate.

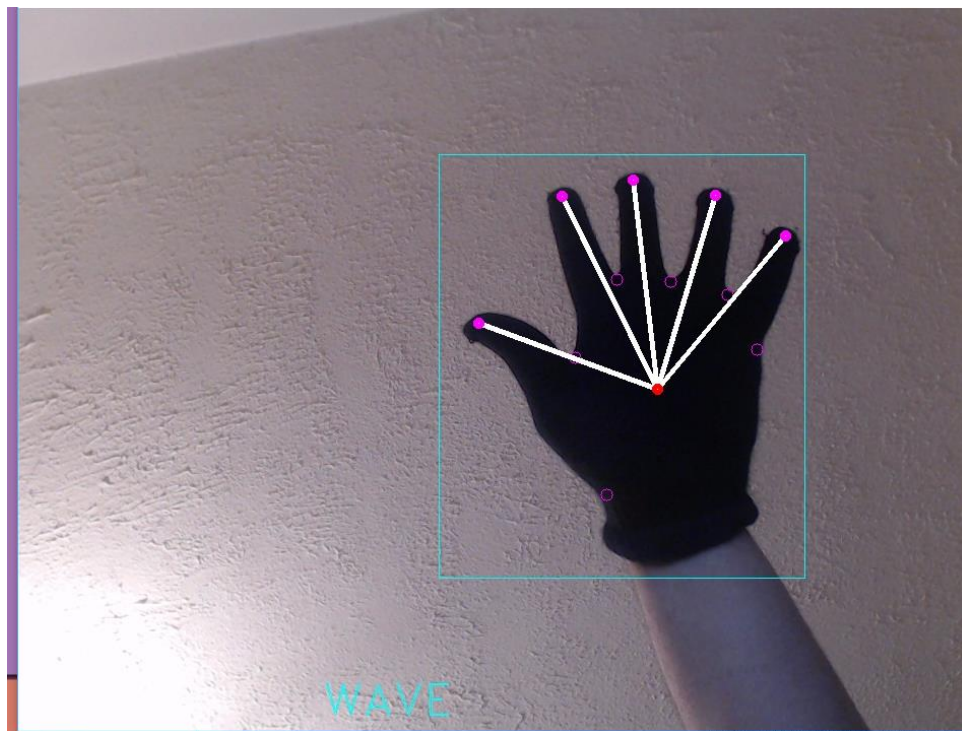


The fourth, the “Rock sign”. As there were two different ways of doing this gesture, multiple conditions had to be set. The first was for when there were three fingers, the angles of the first and third had to be between 60 and 90 degrees in order for the gesture to be accurate. The other condition was that when there were two fingers, the angle had to be greater than 45 degrees but

less than 90. This was due to the fact that the distance between the index and fourth finger would be much larger than that between the first two, and so did not overlap with the peace sign gesture.



The fifth was the “Wave”. When all five fingers were clearly showing the software was able to detect the gesture. This gesture was very simple to create, as the only conditions were that the five fingers were visible.



The names of the gestures were then output onto the display when the conditions were satisfied.

Justification pieces of code

I am using OpenCV the majority of my coding, due to a lot of the work needed having similar or the necessary functions already in OpenCV; it enables me to have the focus primarily on the linking of the code and the steps that are necessary to be taken, saving a lot of time that would otherwise be spent coding for the camera input.

The main tool I am using I background subtraction (A. MORDVINTSEV AND A. REVISION) This enables me to eliminate areas of the video that are not moving and just have the parts that are. Due to this it will be possible to focus on the hands and the face. I will be using the background subtractorMOG method.

The camera used shall be a Logitech 1080p webcam. This will be the only camera I use during the project as I would like the testing to be stable. Due to cameras having different sensors and image quality, the results for the readings of skin colour will be different.

A project in a similar vein was created previously, which involved using gestures to manipulate the mouse functions on a computer (S. PERNA). This was an indication of the further features that could be implemented.

Example Code of Image filtration

```
public Mat
colourFilterHSV(int hue, int
saturation, int value, int h1,
int s1, int v1, Mat image) {
    Mat changedImage =
new Mat();
    if (image != null) {
        Core.inRange(
            image,
            new
Scalar(hue, saturation, value),
            new
Scalar(h1, s1, v1),
            changedImage
        );
    } else {
        System.out.println("Image
error");
    }
    return changedImage;
}
```

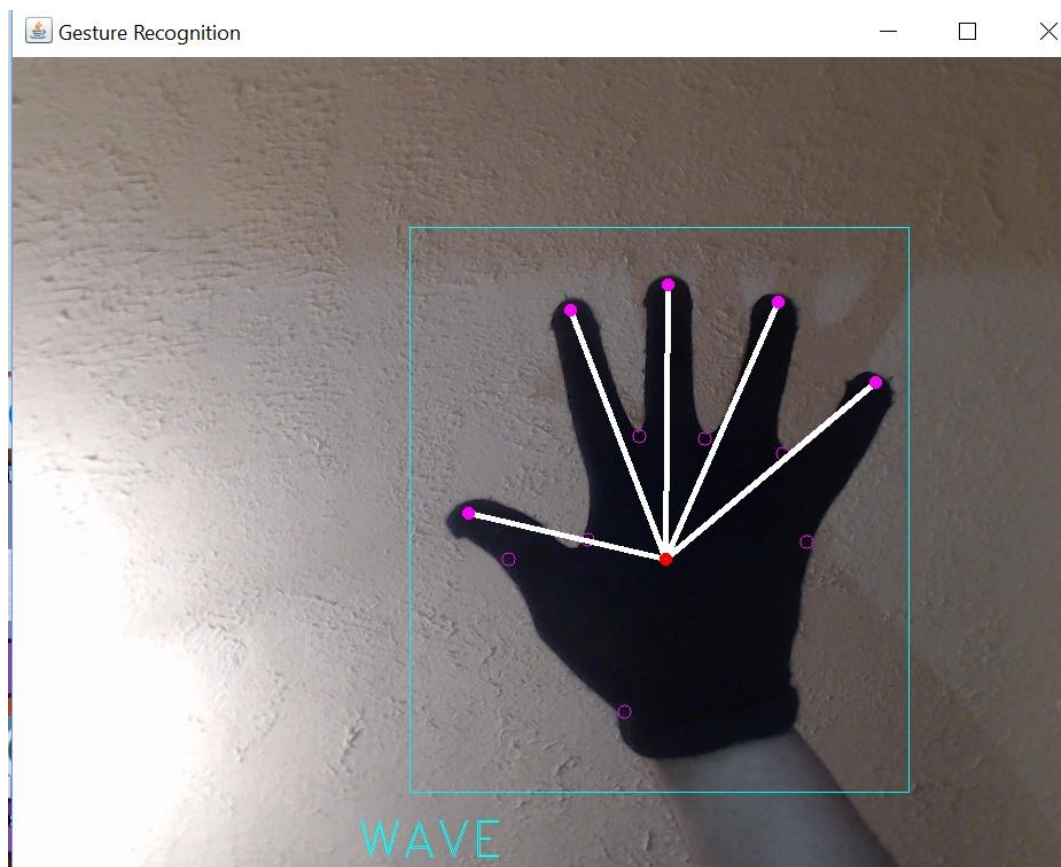
Testing and evaluation

There were many areas that needed tweaking after the code was completed. They have been broken into the main areas below.

Due to the nature of the code, testing was hard to quantify. As the results were in runtime on the display, the human eye was unable to track the difference in small variations of time it took to recognise each gesture. The main method of testing was to notice how frequently the software miscalculated the gesture. The test developed in order to evaluate the program is in the Accuracy section below.

Processing issues:

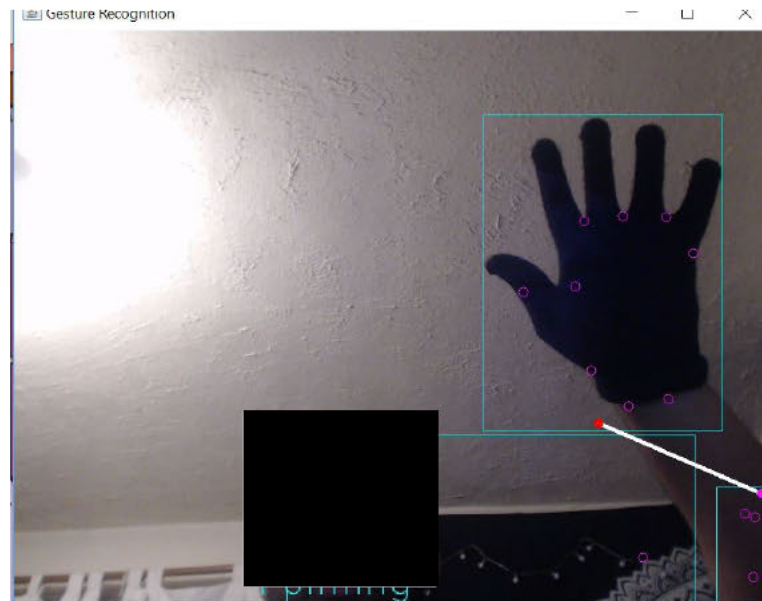
Light variations



Throughout the day and night, the light intensity in the test room changed. This affected the results of how accurate the software was able to guess the gestures. As the room became darker, the glove started to blend more with the background. Additional filtering in the HSV area allowed for the skin tone not to blend in also. The difference between natural light and artificial light was also made apparent in the tests. This was due to the fact that the natural light has a uniformity and came through as an external source, whereas the artificial light had a yellow hue and was in the room.

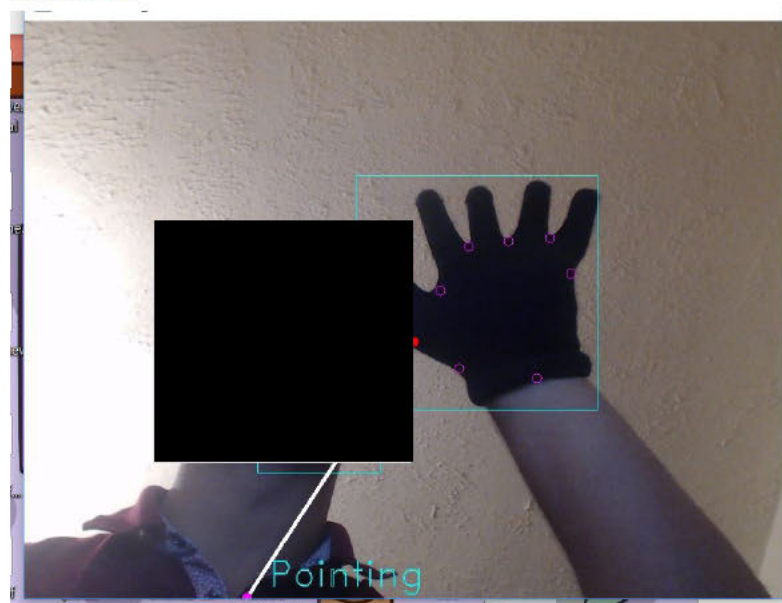
There was a drastic increase in the speed at which the program guessed the gesture when a light source was added underneath the hand.

Hair colour



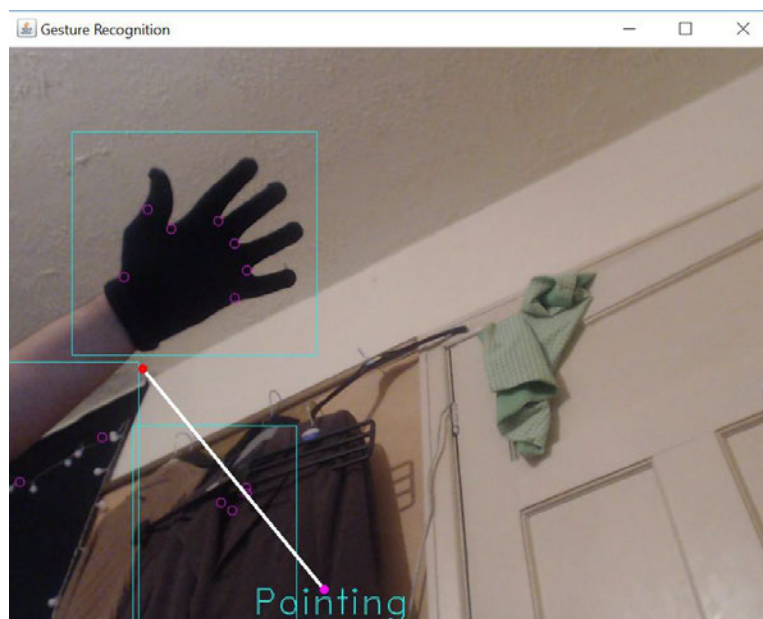
The darker the hair, the harder it was to filter out of the threshold area. For blonde or ginger individuals, their hair colour did not pose much of a problem during test. However when the user had brown to black hair, the system behaved in ways that were anticipated. The hair was seen as an extra vector and the area of the hand was calculated with that included. This made the hand and gestures very inaccurate.

Facial hair



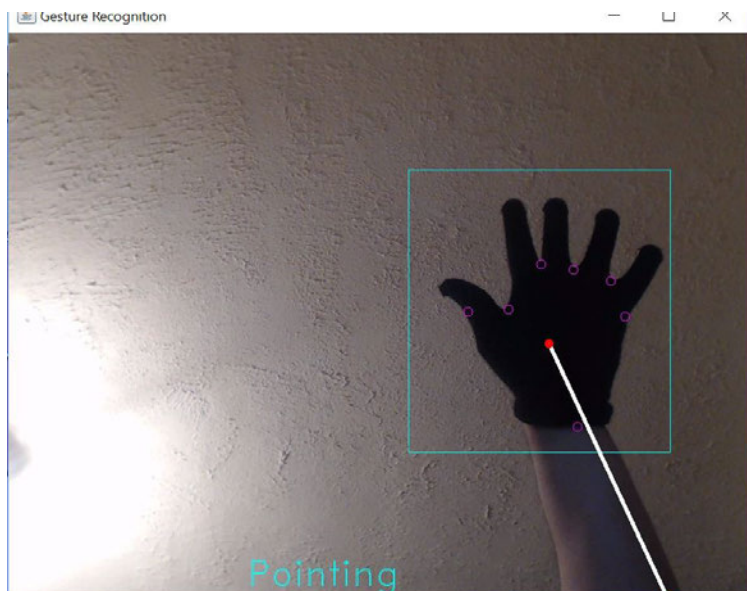
This was a very similar issue to the Hair colour, mentioned above. However, when the user had facial hair, this was often a lot closer to the hand also, and showed up even more prominently on the display. The solution to this was to have just the hand in the frame. This was something to consider in the place of users with beards that were long, as they would affect the programs ability to perform.

Other objects in room



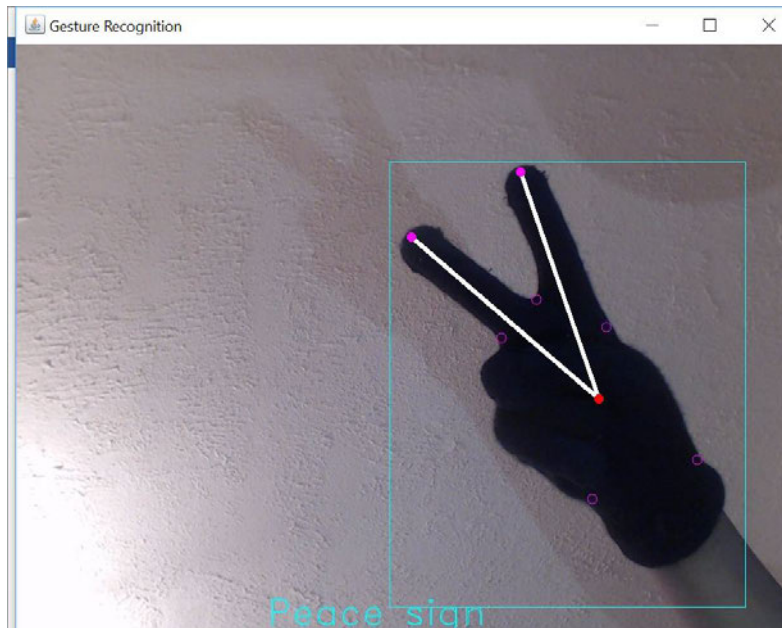
When there were objects in the room of colours outside the threshold, they did not affect the image. When a black object was introduced to the frame it was immediately included in the threshold and such, became a variable in the modifications. In the above picture, a black hanger was introduced into the testing area. Whilst the other objects, aside from the black tapestry, are not notices, the hanger is encapsulated in its own bounding box.

Skin tone



When the skin tone was dark, in certain lighting the skin would fall into the range of the threshold. This was particularly the case when the light source was above the hand and close to the camera; this caused one side of the video feed to go quite dark. The issue still remained on lighter skin tones

when the light source was too close to the image. The test verified that a uniform brightness would be idea for the video.

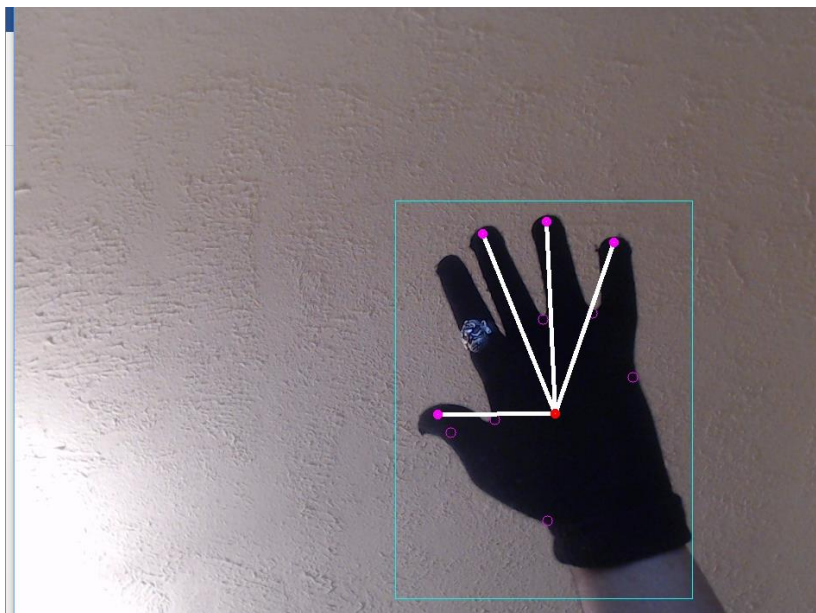


The software, when tested on a user with a slightly darker skin tone, performed reasonable well. There was not much variation in how accurate the results were compared to the previous user. This was to be expected, due to the variation in skin colour not being much at all.

When a light was shone underneath the hand, the same change in results happened. The gestures were recognised much more clearly and had very few wrong instances.

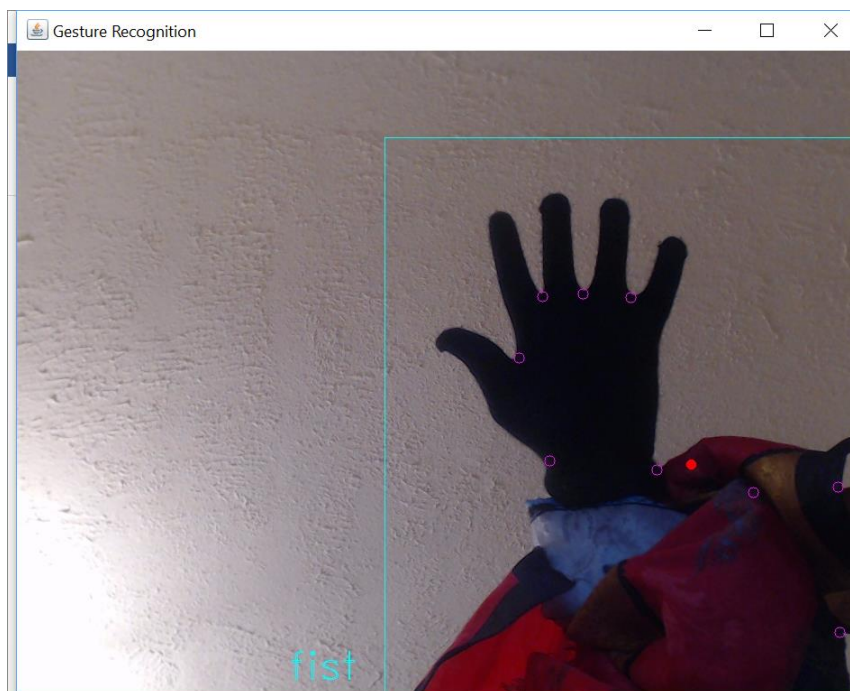
Clothing

The clothing worn had a very large effect on the results. This was not limited to jewellery, sleeve length, and clothing colour.



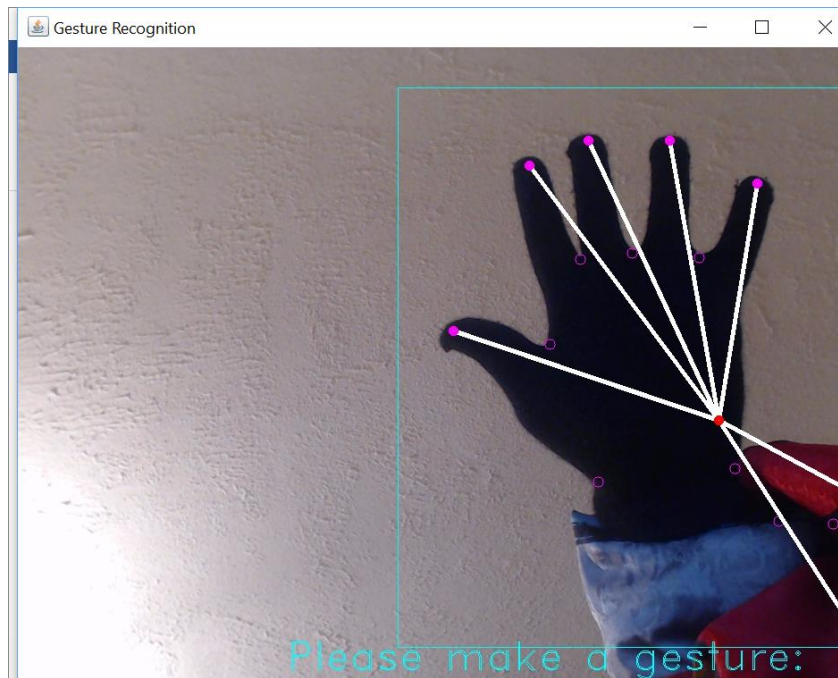
For jewellery, a silver ring was worn the index finger. This caused the finger to no longer be factored into the results. This was an interesting result, due to the fact that the black of the glove was still in the image. The program ignored the black of the rest of the finger and only processed the other fingers.

Upon further inspection, this was shown to be because the values around the finger were brighter, the finger and the surrounding area had been taken out of the threshold range. The camera and the automatic adjustments contributed to this, as when the ring was not being worn the entire image was slightly darker. The image became lighter and the focus shifted onto the ring, shifting the brightness values.



The next test was a brightly coloured item of clothing on the arm of the user. Due to the wide range of colours in the object, the camera was constantly changing the brightness and such the digits were being recognised occasionally. However, there was a lot of black in the object, and the program struggled to recognise what was the hand and what was not. This led to the bounding box including the whole arm, and such the calculations for the hand were not accurate due to the centre being halfway down the forearm.

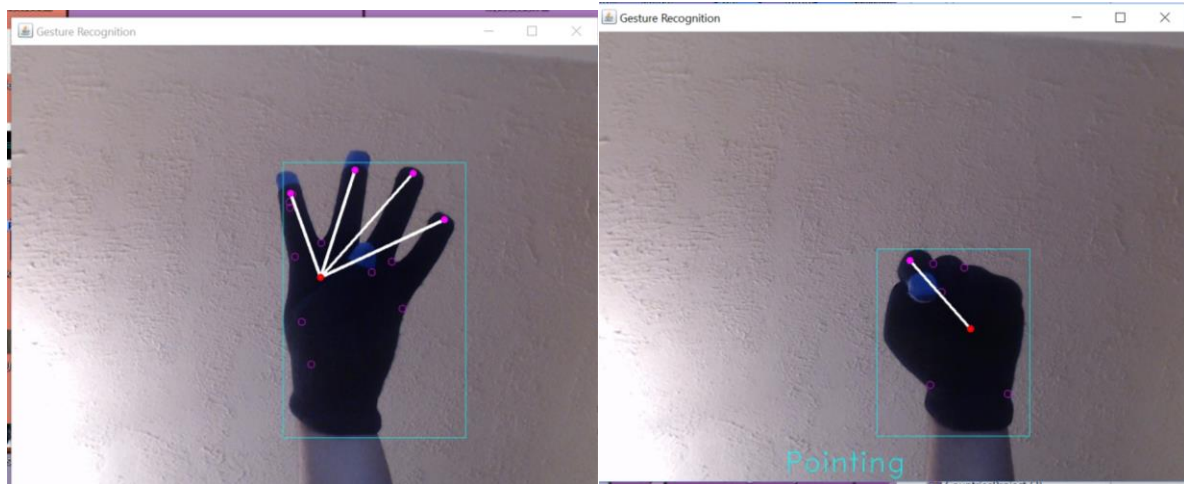
The output was varied, in that there were two very frequent occurrences. The first being that no fingers were detected, and nothing shown. (As in the above image). The other was that the software kept on discovering more and more finger to add. This was unexpected as there should have been no more than five fingers.



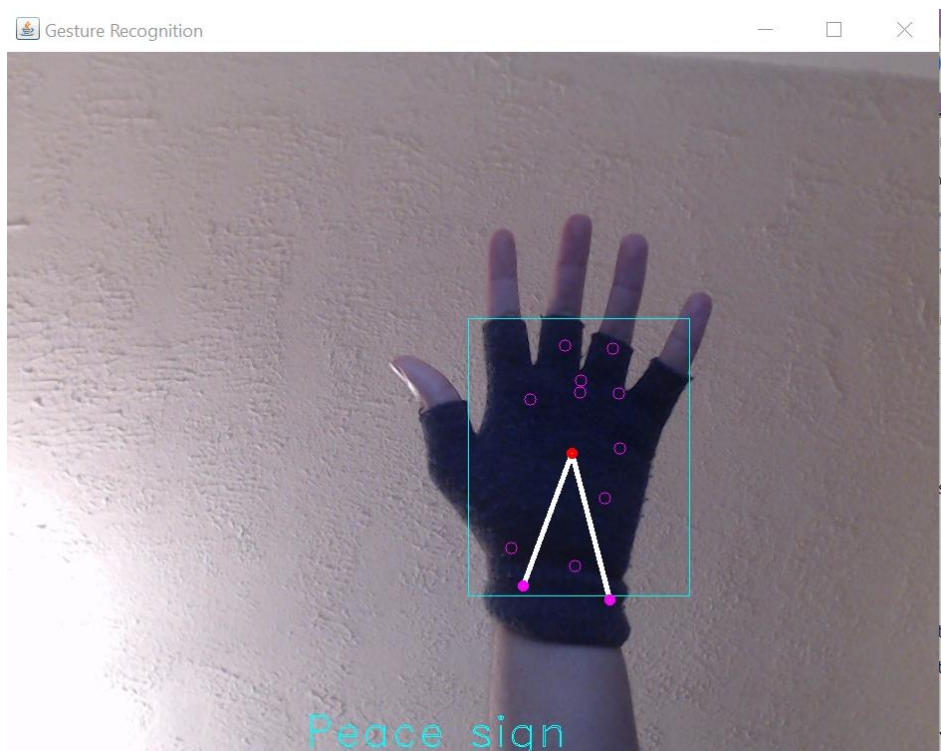
Finally the length of the glove was tested. This was in order to offset the centre of the hand. As the glove length became longer, the distance between the fingers needed to become wider in order to satisfy the equations for some gestures (The “Rock” and “Peace” sign). This meant that for the most effective processing, the glove needed to resemble the shape of the hand as closely as possible, and the end of the glove, where it connected to the wrist, need to be as close to the palm as possible in order to guarantee the most accurate results.

Glove colour

When testing different coloured gloves with different thresholds it was found that the black glove was the best. After testing with a Red, grey and blue set of gloves, the black glove was able to be in the threshold with the least amount of additional processing and widest parameter values. As the brightness in the room decreased the grey glove yielded similar results to the black one, and that was because the darkness enabled the colours of the grey glove to become closer to the black glove.



Glove Type



The software was tested with gloves that had different variations. The first test was on a glove that had the fingers removed. This was in order to test the thresholding was working and that the skin colour was not being included in the threshold.

The results were as expected, the fingertips that were detected were not those of the actual hand, but those of the tip of the gloves. This test was able to show that the thresholding was effective in filtering just the glove and not the skin colour.

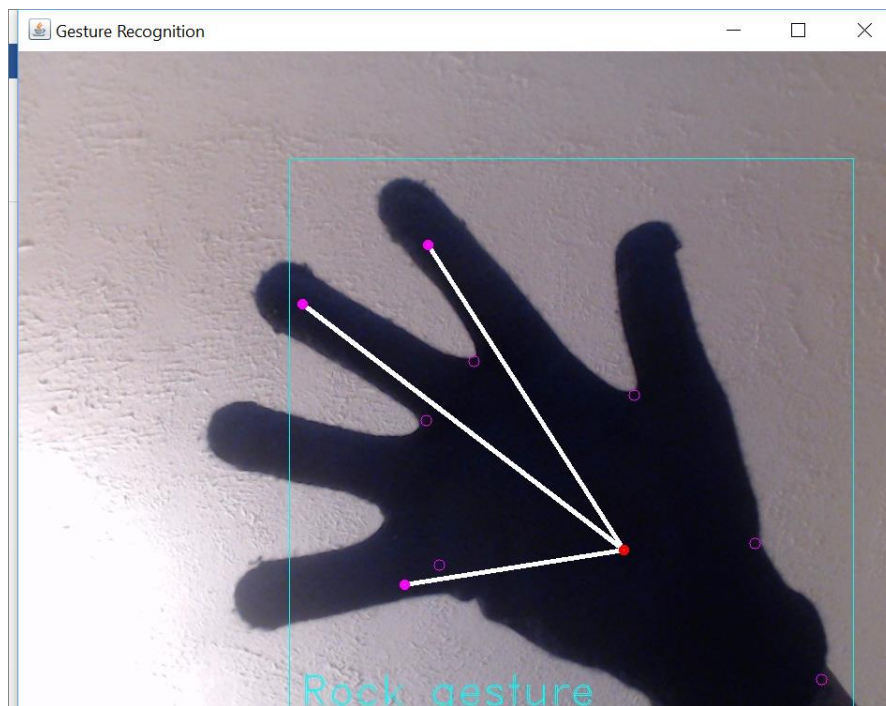
The second test was with a glove that had blue fingertips. The expected result was that the tips of the fingers would not be recognised and the overlay of the hand would only reach the point where

the tips of the fingers turned blue. This was demonstrated in the results, and the accompanying picture validates this. The test was an important factor in determining the HSV values were being threshold appropriately for the borders of the threshold. The darker the colours were, the closer they were to being within the threshold range, it was important to be able to ensure that dark shades of colours that weren't black were not being recognised.

The third was a test on a glove that had the exact same design of the glove with the blue fingertips, but the fingertips on this were white. The expected results were the same as the previous glove. The fingertips should end at the white part of the glove. This was also demonstrated in the results, with the hand overlay stopping at the white area of the glove.

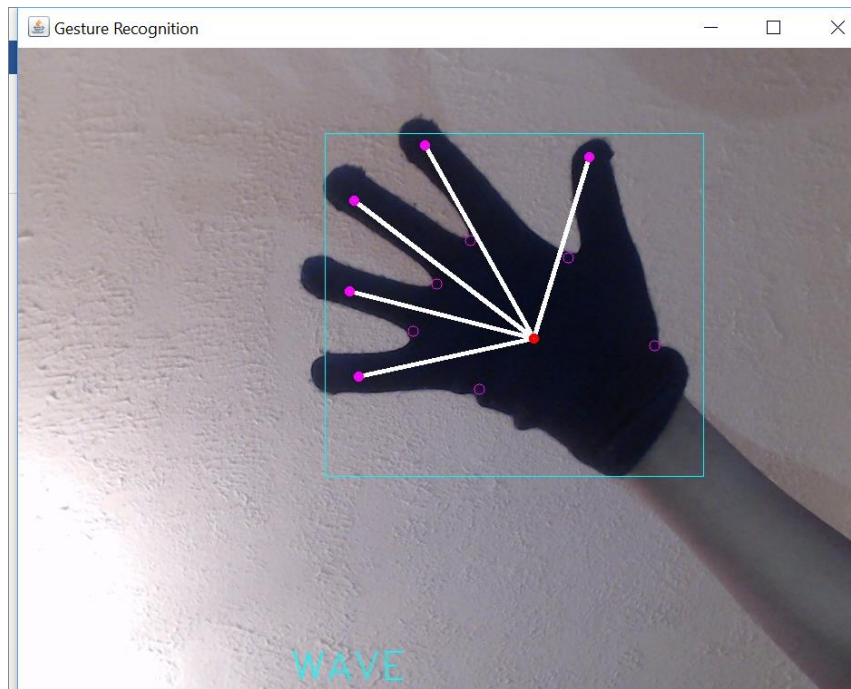
This demonstrated the results on the other end of the scale, with the areas further away from the threshold not being included in the processed image. This test was supportive in that there were no outlier values being included.

User with a different hand size



A very interesting result appeared when testing the program with the same glove on a user with a different hand size. The user had a smaller hand and such the glove appeared thicker, resulting in the angles between the fingers appearing smaller. The results were not as accurate as when the user had a larger hand.

There was a frequent error when the palms were open. The rectification of this issue was to change the settings of the dilations and erosions as they made the shape of the hand appear bigger or smaller, respectively.



Once amended the accuracy became much better, and there were rarely misclassifications of the hand and how many fingers were being held up.

Other areas of testing

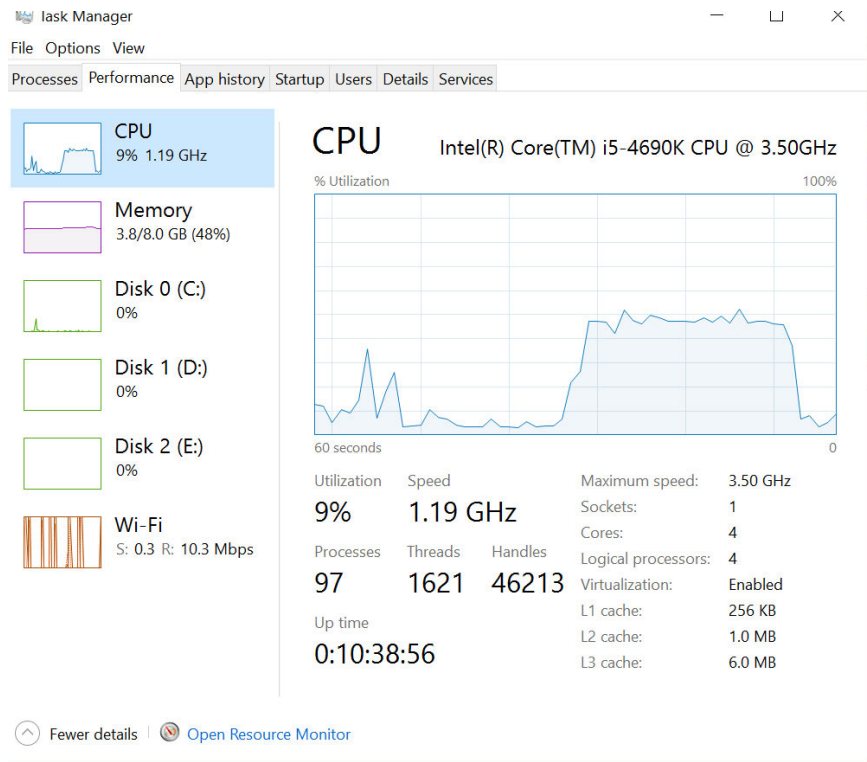
Processor usage

As the program was being tested, the additional processing caused the CPU usage to get much higher.

Table 1: Processor usage

Processing done	CPU Usage (%)
None	12
Threshold image	15
Convert to HSV	16
Get Convexity Defects	18
Final Program	27

In the above table, the usage rose with each added filter. Once the program was completed, it required around 15% of the processing power from the CPU.



The above results were taken after the program had been fully coded. The long period of time where the CPU usage is high was when the program was open. The program seemed to use over 40% of the processing power continuously. It is with noting that there were not particular spikes in the usage when the program was reading particular gestures, nor even when the hand was not in the frame at all.

These results led to the belief that there could be a lot more gestures added and processing performed that would not limit the system, and that the camera itself was a very determining factor in how much processing power was used.

Memory usage

Due to the fact the images were being buffered and not stored to disk, the memory did not get much higher, when running the program the memory rose from 2.98GB usage to 3.03GB in usage and did not get higher.

This further added to the conclusion that a lot more gestures could be added and also supported the idea that the program could run on a mobile device.

Accuracy

The test created was to hold the gesture out for ten seconds. Then to move the gesture from all four corners of the display and count how many times the software guessed the wrong gesture, or lost one of the fingers.

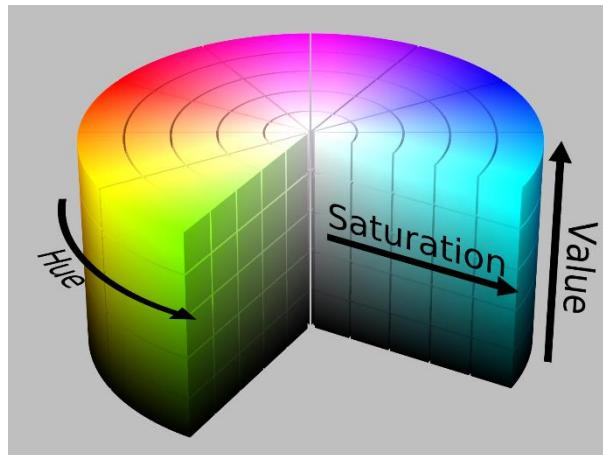
Table 2: Test: Holding the glove in the same position for ten seconds

Variable Changed			Gesture: Times wrong		
	Fist	Point	Peace	Rock Sign	Wave
Glove Colour: black (No changes)	0	0	5	2	1
Natural light only	1	0	7	3	1
Artificial light only	0	0	5	3	1
Fingerless glove	0	4	9	15	7
Blue tipped gloves	0	2	6	4	1
Black sleeved shirt	8	10	20	14	5
Face in frame	3	5	8	8	7
Black object in frame	7	7	15	12	10
Red object in frame	1	0	6	1	0
Green Object in frame	0	1	5	2	0
Blue Object in frame	2	2	3	1	1

Code rectifications

After assessing all of the results from the testing the following changes were made:

Adjustments to the HSV Values:



The Hue Saturation and Value were very specific to the skin colour, by allowing them a greater range, it allowed for people of more varied skin tones to be able to use the program. [21] This did, however, increase the colours that would not be able to be classified as the glove, particularly in very dark skin.

Limit to the angle between fingers:

As there are only five fingers on (most) hands, the angle that between each should be no more than 60, with the exception of the thumb due to its extra flexibility. Due to this the limitation was added onto the angles between fingers. This would prevent the program from having many fingers that did not exist due to finding extra convex points.

Adjustment to the BRG Values:

The BGR values were adjusted for the same reason as the HSV values. The area in which the different skin tones were contained was too specific to one user. The variables needed to be widened in order to encapsulate a larger set of users.

Background decided:

The background had to be as clear and light as possible. For this reason it was decided that the program was best ran on a white or cream background, and the user should only show their hand if they have dark hair.

There was a large black tapestry hanging in the test room which had to be removed due to that being the biggest contour found when running the program. The rest of the tests ran much more efficiently and with less error once that change was implemented. Any noise that could be removed from the background was done so also. This included objects of all colours, however the testing did confirm that it was only necessary for very dark objects and black objects.

Erosion and Dilation values amended:

The image was eroded and dilated according to just one had before the initial tests. However once discovering that by having the initial values decided on a relatively large hand prevented users with smaller hands or thicker gloves from using the program properly and average erosion and dilation amount was decided. This was done through testing different sized faux hands with gloves in order to find the values that contained the most gloves. Exceptionally small gloves, such as those of a baby, are able to be processed, however the glove has to be much closer to the camera.

Adjust light levels:

The light levels had a much larger effect on the results than expected. This brought forth the idea of a constant brightness in the room. This was very helpful in maintaining a fair test when checking for other errors, though in practice, it was a very unrealistic variable to keep constant. This is as with each hour of the day the light levels change if using natural light, and if the user in a room with a window, and they are using an artificial light source, they will still find that the small amount of natural light will affect the programs ability to threshold the image.

The colour scheme is also affected by the light, as most incandescent light emits a yellower glow than natural light. Users with LED lighting would experience a whiter light. [22]

Conclusion

In conclusion, I found that the system was effective in guessing the gestures that were being performed. The amount of processing needed to calculate a few hand gestures does show signs that expanding the system to the whole body, or many more gestures would not be a very processing intensive program.

There were unexpected points that arose, such as the different that hair colour, or facial hair made to the results. This is something that would definitely need to be improved upon over time as the users of the program should not be expected to have to make any drastic changes.

The time taken to create the program and cost to do so for a team would not be a large amount. This means that many teams developing software would be able to add features like these to their systems. The cost would not be large due to the fact that OpenCV is a free library.

The effect that this program could have in the current world could be for a number recognition system, that the user only needs to hold up the number of fingers, or a basic gesture based language. The program in its current state is not able to encapsulate the entirety of the British sign Language due to the overlap that many of the gestures have, however, with the correct glove colouring and thresholding, the program could translate some sign language.

There are many applications for this program, and the testing has shown that there are many potential avenues that can be explored that did not seem relevant at first.

References & Bibliography

- [1] REPRESENTING AND RECOGNIZING HUMAN MOTION: FROM MOTION TEMPLATES TO MOVEMENT CATEGORIES (TUTORIAL NOTES: DIGITAL HUMAN MODELING WORKSHOP, IROS 2001) JAMES W. DAVIS DEPT. OF COMPUTER AND INFORMATION SCIENCE, CENTER FOR COGNITIVE SCIENCE 583 DREESE LAB, 2015 NEIL AVENUE OHIO STATE UNIVERSITY COLUMBUS, OH 43210
- [2] COMPUTER VISION-BASED HUMAN MOTION CAPTURE - A SURVEY TECHNICAL REPORT LIA 99-02 BY: THOMAS B. MOESLUND 7 (NO DATE)
- [3] GIESE, M.A., POGGIO, T.: NEURAL MECHANISMS FOR THE RECOGNITION OF BIOLOGICAL MOVEMENTS. NATURE REVIEWS NEUROSCIENCE 4, 179–192 (2003)
- [4] GINEVRA CASTELLANO, SANTIAGO D. VILLALBA, AND ANTONIO CAMURRI. RECOGNISING HUMAN EMOTIONS FROM BODY MOVEMENT AND GESTURE DYNAMICS. INFOMUS LAB, DIST, UNIVERSITY OF GENOA 2 MLG, SCHOOL OF COMPUTER SCIENCE AND INFORMATICS, UNIVERSITY COLLEGE DUBLIN, 72 (2007)
- [5] HMDB: A LARGE VIDEO DATABASE FOR HUMAN MOTION RECOGNITION - IEEE INTERNATIONAL CONFERENCE ON COMPUTER VISION 2556- 2557 (2011)
- [6] [HTTP://SIMENA86.GITHUB.IO/BLOG/2013/08/12/HAND-TRACKING-AND-RECOGNITION-WITH-OPENCV/](http://simena86.github.io/blog/2013/08/12/hand-tracking-and-recognition-with-opencv/) SIMEN ANDRESEN
- [7] COMPARING THE PERFORMANCE OF L*A*B* AND HSV COLOR SPACES WITH RESPECT TO COLOR IMAGE SEGMENTATION DIBYA JYOTI BORA¹ , ANIL KUMAR GUPTA² , FAYAZ AHMAD KHAN³
- [8] [HTTP://WWW.TMROYAL.COM/A-HIGH-LEVEL-DESCRIPTION-OF-TWO-FINGERTIP-TRACKING-TECHNIQUES-K-CURVATURE-AND-CONVEXITY-DEFECTS.HTML](http://www.tmroyal.com/a-high-level-description-of-two-fingertip-tracking-techniques-k-curvature-and-convexity-defects.html) THOMAS ROYAL
- [9] HANSUNG87 (2011) FINGER DRAWING - WITH OPENCV
[HTTPS://WWW.YOUTUBE.COM/WATCH?V=Z43_HCM74rU](https://www.youtube.com/watch?v=Z43_HCM74rU) (ACCESSED: 10 SEPTEMBER 2016)
- [10] D. MICHEL, I. OIKONOMIDIS, A.A. ARGYROS, “SCALE INVARIANT AND DEFORMATION TOLERANT PARTIAL SHAPE MATCHING”, IN IMAGE AND VISION COMPUTING JOURNAL, ELSEVIER.
[HTTP://CREAT-TABU.BLOGSPOT.CO.UK/2013/08/OPENCV-PYTHON-HAND-GESTURE-RECONITION.HTML](http://creat-tabu.blogspot.co.uk/2013/08/opencv-python-hand-gesture-reconition.html)
- [11] ALEXANDER MORDVINTSEV AND ABID K. REVISION (2013) [HTTP://DOCS.OPENCV.ORG/TRUNK/DB/D5C/TUTORIAL PY BG SUBTRACTION.HTML](http://docs.opencv.org/trunk/db/d5c/tutorial_py_bg_subtraction.html)
- [12] HAND GESTURE SEGMENTATION METHOD BASED ON YCbCr COLOR SPACE AND K-MEANS CLUSTERING ZHANG QIU-YU, LU JUN-CHI, ZHANG MO-YI, DUAN HONG-XIANG AND LV LU SCHOOL OF COMPUTER AND COMMUNICATION, LANZHOU UNIVERSITY OF TECHNOLOGY, LANZHOU, 730050, CHINA
- [13] BRADSKI, G. DR. DOBB’S JOURNAL OF SOFTWARE TOOLS (200)
- [14] HAND TRACKING AND GESTURE RECOGNITION SYSTEM FOR HUMAN-COMPUTER INTERACTION USING LOW-COST HARDWARE HUI-SHYONG YEO, BYUNG-GOOK LEE, HYOTAEK LIM
- [15] VARIATION IN HAND SHAPE AND ORIENTATION IN BRITISH SIGN LANGUAGE: THE CASE OF THE ‘1’ HAND CONFIGURATION JORDAN FENLONA, ADAM SCHEMBRIB, RAMAS RENTELISA, KEARSY CORMIERA, A DEAFNESS, COGNITION & LANGUAGE RESEARCH CENTRE, UNIVERSITY COLLEGE LONDON, UK B LA TROBE UNIVERSITY, MELBOURNE, AUSTRALIA

[16] [HTTP://DOCS.OPENCV.ORG/2.4/DOC/TUTORIALS/IMGPROC/GAUSIAN_MEDIAN_BLUR_BILATERAL_FILTER/GAUSIAN_MEDIAN_BLUR_BILATERAL_FILTER.HTML](http://docs.opencv.org/2.4/doc/tutorials/imgproc/gaussian_median_blur_bilateral_filter/gaussian_median_blur_bilateral_filter.html)

[17] STEFANO PERNA (2013) - HAND GESTURE RECOGNITION SYSTEM AVAILABLE AT: [HTTPS://WWW.YOUTUBE.COM/WATCH?V=32_CRODWDLE](https://www.youtube.com/watch?v=32_CRODWDLE) (ACCESSED: 24 NOVEMBER 2016)

[18] EVALUATION OF RGB AND HSV MODELS IN HUMAN FACES DETECTION MARIAN SEDLAČEK

[19] FACULTY OF INFORMATICS AND INFORMATION TECHNOLOGIES, SLOVAK UNIVERSITY OF TECHNOLOGY, BRATISLAVA/ SLOVAKIA

[20] STEPHENS, N. AND BOLANDER, A., "FACTORS IN THE PERCEPTION OF BRIGHTNESS FOR LED AND INCANDESCENT LAMPS," SAE TECHNICAL PAPER 2005-01-0866, 2005, DOI:10.4271/2005-01-0866.

APPENDICES

```
IMPORT JAVAX.SWING.JFRAME;  
IMPORT JAVAX.SWING.JLABEL;  
IMPORT JAVAX.SWING.JPANEL;  
IMPORT ORG.OPENCV.CORE.CORE;  
IMPORT ORG.OPENCV.CORE.MAT;  
IMPORT ORG.OPENCV.CORE.POINT;  
IMPORT ORG.OPENCV.HIGHGUI.HIGHGUI;  
IMPORT ORG.OPENCV.HIGHGUI.VIDEOCAPTURE;  
  
IMPORT JAVA.AWT.AWTException;  
IMPORT JAVA.UTIL.LINKEDLIST;  
IMPORT JAVA.UTIL.LIST;
```

```
PUBLIC CLASS PROJECTFINAL EXTENDS JPANEL {
```

```
    //CREATE WINDOW FOR CAMERA
```

```
    JFrame frame = new JFrame("GESTURE RECOGNITION");
```

```
    JLabel label = new JLabel();
```

```

STATIC STRING STRING = "";

STATIC BOOLEAN CLOSED = FALSE; //

    PRIVATE STATIC FINAL LONG SERIALVERSIONUID = 1L;

PUBLIC STATIC VOID MAIN(STRING[] ARGS)

    THROWS INTERRUPTEDEXCEPTION, AWTEXCEPTION {

SYSTEM.LOADLIBRARY(CORE.NATIVE_LIBRARY_NAME);

PROJECTFINAL VID = NEW PROJECTFINAL();

VIDEOCAPTURE CAMERA = NEW VIDEOCAPTURE(0);

// SET DISPLAY RESOLUTION

CAMERA.SET(HIGHGUI.CV_CAP_PROP_FRAME_HEIGHT, 768);

CAMERA.SET(HIGHGUI.CV_CAP_PROP_FRAME_WIDTH, 1024);

NEW VIDEO().SETFRAME(CAMERA, VID);


MAT MIMAGE = NEW MAT();

MAT MIMAGECHANGE = NEW MAT();

POINT FINGER = NEW POINT();

POINT CENTRE = NEW POINT();

LIST<POINT> BUFFER = NEW LINKEDLIST<POINT>();

LIST<POINT> FINGERS = NEW LINKEDLIST<POINT>();

LIST<POINT> FINGERSBUFFER = NEW LINKEDLIST<POINT>();

WHILE (!CLOSED) {

    IF (!CAMERA.ISOPENED() && !CLOSED) { //MAKE SURE THE CAMERA IS ON

        SYSTEM.OUT.PRINTLN("NO CAMERA");

    } ELSE {

        LIST<POINT> DEFECTS = NEW LINKEDLIST<POINT>();

        IF (!CLOSED) {

            CAMERA.RETRIEVE(MIMAGE);

```

```
MIMAGECHANGE = NEW IMAGEPROCESSING().MORPHOLOGICALFILTER(
```

```
3,
```

```
7,
```

```
//FILTER OUT WHAT ISN'T THE GLOVE
```

```
NEW IMAGEPROCESSING().COLOURFILTERHSV(
```

```
0,
```

```
0,
```

```
0,
```

```
50, //DARK GLOVE SO LOW HUE
```

```
255, //BLACK GLOVE, FULLY SATURATED
```

```
35, //SLIGHT LEVELS TO GREY AREAS OF GLOVE
```

```
MIMAGE
```

```
)
```

```
);
```

```
DEFECTS = NEW HANDCALCULATIONS().CONVEXDEFECTS( //INNER PARTS OF HAND
```

```
MIMAGE,
```

```
NEW HANDCALCULATIONS().GETCONTOUR(
```

```
MIMAGE,
```

```
MIMAGECHANGE,
```

```
FALSE,
```

```
FALSE,
```

```
450, VID
```

```
),
```

```
FALSE,
```

```
5, VID
```

```
);
```

```
IF (BUFFER.SIZE() < 1) { //CHECK THAT THERE ARE FINGERS
```

```
    BUFFER.ADD(NEW HANDCALCULATIONS().CENTREOFPALM(MIMAGE, DEFECTS, VID));
```

```
} ELSE {
```

```

CENTRE = NEW HANDCALCULATIONS().CENTREOFPALM(MIMAGE, DEFECTS, VID);
}

FINGERS = NEW HANDCALCULATIONS().FINGERS(//GET FINGERS
    MIMAGE,
    NEW HANDCALCULATIONS().GETCONTOURS(MIMAGECHANGE, 200, NEW IMAGEPROCESSING()),
    CENTRE, VID
);

IF (FINGERS.SIZE() == 1 && FINGERSBUFFER.SIZE() < 5) { //MAXIMUM OF FIVE FINGERS ALLOWED
    FINGERSBUFFER.ADD(FINGERS.GET(0));
    FINGER = FINGERS.GET(0);
} ELSE {
    IF (FINGERS.SIZE() == 1) {
        FINGER = FINGERS.GET(0);
    }
}

NEW HANDCALCULATIONS().DRAWFINGERSONTOCENTREOFPALM( //DRAW FINGERS
    MIMAGE,
    CENTRE,
    FINGER,
    FINGERS
);

NEW HANDCALCULATIONS().GESTURES( //DRAWS GESTURES
    FINGERS,
    FINGER,
    CENTRE,

```



```
        MIMAGE, VID
    );

    NEW VIDEO().TOLABEL(MIMAGE, VID); //SHOWS CAMERA FEED
}
}
}
}
}

IMPORT ORG.OPENCV.CORE.CORE;
IMPORT ORG.OPENCV.CORE.MAT;
IMPORT ORG.OPENCV.CORE.SCALAR;
IMPORT ORG.OPENCV.CORE.SIZE;
IMPORT ORG.OPENCV.IMGPROC.IMGPROC;

PUBLIC CLASS IMAGEPROCESSING {
    //ALL IMAGE FILTERING AND MODIFICATIONS

    PUBLIC MAT COLOURFILTERHSV(INT HUE, INT SATURATION, INT VALUE, INT H1, INT S1, INT V1, MAT IMAGE)
{
    MAT CHANGEDIMAGE = NEW MAT();
    IF (IMAGE != NULL) {
        CORE.INRANGE(
            IMAGE,
            NEW SCALAR(HUE, SATURATION, VALUE),
```

```

        NEW SCALAR(h1, s1, v1),

        CHANGEDIMAGE

    );
} ELSE {

    SYSTEM.OUT.PRINTLN("IMAGE ERROR");

}

RETURN CHANGEDIMAGE;
}

```

```

PUBLIC MAT MORPHOLOGICALFILTER(INT DILATION, INT EROSION, MAT IMAGE) {

    MAT NEWIMAGE = NEW MAT();

    IMGPROC.DILATE( //DILATE THE IMAGE

        IMAGE,

        NEWIMAGE,

        IMGPROC.GETSTRUCTURINGELEMENT(IMGPROC.MORPH_ELLIPSE, NEW SIZE(DILATION, DILATION))

    );

    IMGPROC.ERODE( //ERODE THE IMAGE

        IMAGE,

        NEWIMAGE,

        IMGPROC.GETSTRUCTURINGELEMENT(IMGPROC.MORPH_ELLIPSE, NEW SIZE(EROSION, EROSION))

    );

    RETURN NEWIMAGE;

}

```

```

PUBLIC MAT SKINDETECTION(MAT ORIGINALIMAGE, PROJECTFINAL PROJECTFINAL) { //STOP SKIN BEING
DETECTED

    MAT MASK = NEW MAT();

    MAT RESULTIMAGE = NEW MAT();

    CORE.INRANGE( //SKIN IS WITHIN THESE VALUES

        ORIGINALIMAGE,

        NEW SCALAR(0, 0, 0),

        NEW SCALAR(50, 29, 29),

```



```
        RESULTIMAGE
    );

    //CONVERT COLOUR TO HSV AS MORE ACCURATE FOR DETECTING SKIN TONE
    IMGPROC.CVTCOLOR(
        ORIGINALIMAGE,
        MASK,
        IMGPROC.COLOR_BGR2HSV);

    FOR (INT I = 0; I < MASK.SIZE().HEIGHT; I++) { //FILTER OUT UNWANTED COLOURS
        FOR (INT J = 0; J < MASK.SIZE().WIDTH; J++) {
            IF (MASK.GET(I, J)[0] < 18 || MASK.GET(I, J)[0] > 149
                && MASK.GET(I, J)[1] > 26 && MASK.GET(I, J)[1] < 220) {
                RESULTIMAGE.PUT(I, J, 255, 255, 255); //SET VALUES OF UNWANTED COLOURS

            } ELSE {
                RESULTIMAGE.PUT(I, J, 0, 0, 0); //SEPARATE OTHER COLOURS
            }
        }
    }

    RETURN RESULTIMAGE;
}

}
```

```
IMPORT JAVA.UTIL.ARRAYLIST;

IMPORT JAVA.UTIL.LINKEDLIST;

IMPORT JAVA.UTIL.LIST;
```



```

IMPORT ORG.OPENCV.CORE.CORE;

IMPORT ORG.OPENCV.CORE.CVTYPE;

IMPORT ORG.OPENCV.CORE.MAT;

IMPORT ORG.OPENCV.CORE.MATOFINT;

IMPORT ORG.OPENCV.CORE.MATOFINT4;

IMPORT ORG.OPENCV.CORE.MATOFPOINT;

IMPORT ORG.OPENCV.CORE.MATOFPOINT2F;

IMPORT ORG.OPENCV.CORE.POINT;

IMPORT ORG.OPENCV.CORE.ROTATEDRECT;

IMPORT ORG.OPENCV.CORE.SCALAR;

IMPORT ORG.OPENCV.IMGPROC.IMGPROC;

IMPORT ORG.OPENCV.UTILS.CONVERTERS;

PUBLIC CLASS HANDCALCULATIONS {

//ALL OF THE CALCULATIONS FOR DRAWING THE OVERLAY

    PUBLIC LIST<MATOFPOINT> GETCONTOUR(MAT ORIGINAL, MAT IMAGE, BOOLEAN DRAW, BOOLEAN
DRAWALL, INT PIXELFILTER, PROJECTFINAL PROJECTFINAL) { //USED FOR CONVEX DEFECTS

        LIST<MATOFPOINT> CONTOURS = NEW LINKEDLIST<MATOFPOINT>(); //LIST OF EACH CONTOUR

        LIST<MATOFPOINT> BIGGESTCONTOURS = NEW LINKEDLIST<MATOFPOINT>(); //LIST OF THE FINGERTIPS

        MAT HIERARCHY = NEW MAT();

        //FIND THE CONTOURS OF THE HAND

        IMGPROC.FINDCONTOURS(

            IMAGE,

            CONTOURS,

            HIERARCHY,

            IMGPROC.RETR_EXTERNAL,

            IMGPROC.CHAIN_APPROX_NONE, //DON'T CHAIN TO THE BORDERS

            NEW POINT(0, 0) //NEW POINT FOR EACH CONTOUR

        );

        FOR (INT I = 0; I < CONTOURS.SIZE(); I++) {

```

```

        IF (CONTOURS.GET(I).SIZE().HEIGHT > PIXELFILTER) {

            BIGGESTCONTOURS.ADD(CONTOURS.GET(I)); //ADD TO FINGERTIPS

        }

    }

    RETURN BIGGESTCONTOURS;

}

```

```

PUBLIC LIST<POINT> GETCONTOURS(MAT IMAGE, INT PIXELFILTER, IMAGEPROCESSING IMAGEPROCESSING) {
//USED FOR THE FINGERS SAME AS ABOVE

```

```

    LIST<MATOFPOINT> CONTOURS = NEW LINKEDLIST<MATOFPOINT>();

    LIST<MATOFPOINT> BIGGESTCONTOURS = NEW LINKEDLIST<MATOFPOINT>();

    LIST<POINT> POINTS = NEW LINKEDLIST<POINT>();

    MAT HIERARCHY = NEW MAT();

```

```

    IMGPROC.FINDCONTOURS( //FIND CONTOURS

        IMAGE,

        CONTOURS,

        HIERARCHY,

        IMGPROC.RETR_EXTERNAL,

        IMGPROC.CHAIN_APPROX_NONE,

        NEW POINT(0, 0)

    );

```

```

    FOR (INT I = 0; I < CONTOURS.SIZE(); I++) {

        IF (CONTOURS.GET(I).SIZE().HEIGHT > PIXELFILTER) {

            BIGGESTCONTOURS.ADD(CONTOURS.GET(I));

        }

    }

    IF (BIGGESTCONTOURS.SIZE() > 0) {

        POINTS = BIGGESTCONTOURS.GET(0).TOLIST();

    }

    RETURN POINTS;

}

```

```

PUBLIC LIST<POINT> CONVEXDEFECTS(MAT IMAGE, LIST<MATOFPOINT> CONTOURS, BOOLEAN DRAW, INT
DEPTHTHRESHOLD, PROJECTFINAL PROJECTFINAL) { //FIND THE SHALLOWS BETWEEN FINGERS

```

```

    LIST<POINT> DEFECTS = NEW LINKEDLIST<POINT>();

```

```

    FOR (INT I = 0; I < CONTOURS.SIZE(); I++) {

```

```

        MATOFINT CONVHULL = NEW MATOFINT();

```

```

        MATOFINT4 CONVEXITYDEFECTS = NEW MATOFINT4(); //VECTOR

```

```

        IMGPROC.CONVEXHULL(CONTOURS.GET(I), CONVHULL); //FIND THE CONVEX HULL OF THE HAND

```

```

        IF (CONVHULL.SIZE().HEIGHT >= 5) { //GET THE SPACE IN BETWEEN THE FIVE FINGERS ONLY WHEN ALL
FIVE FINGERS IDENTIFIED

```

```

            IMGPROC.CONVEXITYDEFECTS(CONTOURS.GET(I), CONVHULL, CONVEXITYDEFECTS);

```

```

            LIST<POINT> POINTS = NEW ARRAYLIST<POINT>();

```

```

            MATOFPOINT2F POINTER = NEW MATOFPOINT2F();

```

```

            CONVERTERS.MAT_TO_VECTOR_POINT(CONTOURS.GET(I), POINTS); //GET THE VECTORS OF THE
CONTOURS

```

```

            POINTER.CREATE((INT) (POINTS.SIZE()), 1, CVTYPE.CV_16U); //POINTS FOR RECTANGLE

```

```

            POINTER.FROMLIST(POINTS);

```

```

            IF (POINTER.HEIGHT() > 10) {

```

```

                ROTATEDRECT ENCAPSULATINGRECTANGLE = IMGPROC.MINAREARECT(POINTER);

```

```

                POINT[] RECTANGLEPOINTS = NEW POINT[4];

```

```

                ENCAPSULATINGRECTANGLE.POINTS(RECTANGLEPOINTS);

```

```

            //BOUNDING RECTANGLE FOR HAND

```

```

            CORE.RECTANGLE(

```



```
        IMAGE,
        ENCAPSULATINGRECTANGLE.BOUNDINGRECT().TL(),
        ENCAPSULATINGRECTANGLE.BOUNDINGRECT().BR(),
        NEW SCALAR(
            250,
            250,
            0
        )
    );
}

INT[] BUFFER = NEW INT[4];

FOR (INT I2 = 0; I2 < CONVEXITYDEFECTS.SIZE().HEIGHT; I2++) { //ADD THE CONVEXITY DEFECTS TO
IMAGE
    CONVEXITYDEFECTS.GET(I2, 0, BUFFER); //GET DEFECTS
    IF (BUFFER[3] / 256 > DEPTHTHRESHOLD) { //CHECK THAT DEFECTS ARE THERE
        IF (POINTS.GET(BUFFER[2]).X > 0 && POINTS.GET(BUFFER[2]).X < 1024 &&
POINTS.GET(BUFFER[2]).Y > 0 && POINTS.GET(BUFFER[2]).Y < 768) {
            DEFECTS.ADD(POINTS.GET(BUFFER[2]));
            CORE.CIRCLE(IMAGE, POINTS.GET(BUFFER[2]), 6, NEW SCALAR(250, 0, 250)); //DRAW THE
DEFECTS
        }
    }
}

}
}

RETURN DEFECTS;
}
```

```

PUBLIC DOUBLE CALCULATEDistance(Point P1, Point P2, ProjectFinal ProjectFinal) { //Distance
BETWEEN THE FINGERS

```

```

    RETURN MATH.SQRT(((P1.X - P2.X) * (P1.X - P2.X)) + ((P1.Y - P2.Y) * (P1.Y - P2.Y)));
}

```

```

PUBLIC DOUBLE CALCULATEAngle(Point P1, Point P2, Point P3, ProjectFinal ProjectFinal) {
//CALCULATE THE ANGLE INBETWEEN THE FINGERS

```

```

    POINT VERTICE1 = NEW POINT();

```

```

    POINT VERTICE2 = NEW POINT();

```

```

    DOUBLE ANGLE = 0;

```

```

    //SET VERTICES

```

```

    VERTICE1.X = P3.X - P1.X;

```

```

    VERTICE1.Y = P3.Y - P1.Y;

```

```

    VERTICE2.X = P3.X - P2.X;

```

```

    VERTICE2.Y = P3.Y - P2.Y;

```

```

    DOUBLE DOTPRODUCT = (VERTICE1.X * VERTICE2.X) + (VERTICE1.Y * VERTICE2.Y); //CALCULATE DOT
PRODUCT

```

```

    DOUBLE LENGTH1 = MATH.SQRT((VERTICE1.X * VERTICE1.X) + (VERTICE1.Y * VERTICE1.Y));

```

```

    DOUBLE LENGTH2 = MATH.SQRT((VERTICE2.X * VERTICE2.X) + (VERTICE2.Y * VERTICE2.Y));

```

```

    DOUBLE ANGLEACOS = MATH.ACOS(DOTPRODUCT / (LENGTH1 * LENGTH2));

```

```

    ANGLE = ANGLEACOS * 180 / MATH.PI;

```

```

    RETURN ANGLE;
}

```

```

PUBLIC POINT CENTREOfPALM(Mat Image, List<Point> DEFECTS, ProjectFinal ProjectFinal) {

```

```

    MATOfPOINT2F PR = NEW MATOfPOINT2F();

```

```

    POINT CENTER = NEW POINT();

```

```

    FLOAT[] RADIUS = NEW FLOAT[1];

```

```

    PR.CREATE((INT) (DEFECTS.SIZE()), 1, CVTYPE.CV_32S);

```

```

    PR.FROMLIST(DEFECTS);
}

```



```
IF (PR.SIZE().HEIGHT > 0) { //ENSURE THERE IS A POINT
    IMGPROC.MINENCLOSINGCIRCLE(PR, CENTER, RADIUS);

} ELSE {

}

RETURN CENTER;
}
```

```
PUBLIC LIST<POINT> FINGERS(MAT IMAGE, LIST<POINT> CONTOURPOINTS, POINT CENTER, PROJECTFINAL
PROJECTFINAL) { ///CREATE FINGERS

    INT INTERVAL = 55;

    LIST<POINT> FINGERS = NEW LINKEDLIST<POINT>();

    LIST<POINT> FINGERSPOINTS = NEW LINKEDLIST<POINT>();

    FOR (INT J = 0; J < CONTOURPOINTS.SIZE(); J++) {

        POINT PREVIOUS = NEW POINT();

        POINT VERTEX = NEW POINT();

        POINT AFTER = NEW POINT();

        VERTEX = CONTOURPOINTS.GET(J);

        IF (J - INTERVAL > 0) {

            PREVIOUS = CONTOURPOINTS.GET(J - INTERVAL);

        } ELSE {

            INT A = INTERVAL - J;

            PREVIOUS = CONTOURPOINTS.GET(CONTOURPOINTS.SIZE() - A - 1);

        }

        IF (J + INTERVAL < CONTOURPOINTS.SIZE()) {

            AFTER = CONTOURPOINTS.GET(J + INTERVAL);

        } ELSE {

            INT A = J + INTERVAL - CONTOURPOINTS.SIZE();

            AFTER = CONTOURPOINTS.GET(A);

        }

    }
```



```
POINT v1 = NEW POINT();//SET VERTICES

POINT v2 = NEW POINT();

v1.X = VERTEX.X - AFTER.X;

v1.Y = VERTEX.Y - AFTER.Y;

v2.X = VERTEX.X - PREVIOUS.X;

v2.Y = VERTEX.Y - PREVIOUS.Y;

DOUBLE DOTPRODUCT = (v1.X * v2.X) + (v1.Y * v2.Y);

DOUBLE LENGTH1 = MATH.SQRT((v1.X * v1.X) + (v1.Y * v1.Y));

DOUBLE LENGTH2 = MATH.SQRT((v2.X * v2.X) + (v2.Y * v2.Y));

DOUBLE ANGLE = MATH.ACOS(DOTPRODUCT / (LENGTH1 * LENGTH2));

ANGLE = ANGLE * 180 / MATH.PI;


IF (ANGLE < 60) { //ANGLE BETWEEN FINGERS LESS THAN 60

    DOUBLE CENTREPREC = MATH.SQRT(((PREVIOUS.X - CENTER.X) * (PREVIOUS.X - CENTER.X)) +
    ((PREVIOUS.Y - CENTER.Y) * (PREVIOUS.Y - CENTER.Y)));

    DOUBLE CENTREVERTEX = MATH.SQRT(((VERTEX.X - CENTER.X) * (VERTEX.X - CENTER.X)) + ((VERTEX.Y
    - CENTER.Y) * (VERTEX.Y - CENTER.Y)));

    DOUBLE CENTRENEXT = MATH.SQRT(((AFTER.X - CENTER.X) * (AFTER.X - CENTER.X)) + ((AFTER.Y -
    CENTER.Y) * (AFTER.Y - CENTER.Y)));

    IF (CENTREPREC < CENTREVERTEX && CENTRENEXT < CENTREVERTEX) {

        FINGERSPOINTS.ADD(VERTEX);

    }

}

}

POINT MIDDLE = NEW POINT();//CENTRE OF HAND

MIDDLE.X = 0;

MIDDLE.Y = 0;

INT MID = 0;

BOOLEAN T = FALSE;

IF (FINGERSPOINTS.SIZE() > 0) {
```

```

DOUBLE DIF = MATH.SQRT(((FINGERSPOINTS.GET(0).X - FINGERSPOINTS.GET(FINGERSPOINTS.SIZE() - 1).X) * (FINGERSPOINTS.GET(0).X - FINGERSPOINTS.GET(FINGERSPOINTS.SIZE() - 1).X)) + ((FINGERSPOINTS.GET(0).Y - FINGERSPOINTS.GET(FINGERSPOINTS.SIZE() - 1).Y) * (FINGERSPOINTS.GET(0).Y - FINGERSPOINTS.GET(FINGERSPOINTS.SIZE() - 1).Y))));

IF (DIF <= 20) {

    T = TRUE;

}

}

FOR (INT I = 0; I < FINGERSPOINTS.SIZE() - 1; I++) {

    DOUBLE D = MATH.SQRT(((FINGERSPOINTS.GET(I).X - FINGERSPOINTS.GET(I + 1).X) * (FINGERSPOINTS.GET(I).X - FINGERSPOINTS.GET(I + 1).X)) + ((FINGERSPOINTS.GET(I).Y - FINGERSPOINTS.GET(I + 1).Y) * (FINGERSPOINTS.GET(I).Y - FINGERSPOINTS.GET(I + 1).Y))));

    IF (D > 20 || I + 1 == FINGERSPOINTS.SIZE() - 1) {

        POINT P = NEW POINT();

        P.X = (INT) (MIDDLE.X / MID);

        P.Y = (INT) (MIDDLE.Y / MID);

        FINGERS.ADD(P);

    }

    IF (T && I + 1 == FINGERSPOINTS.SIZE() - 1) {

        POINT ULT = NEW POINT();

        IF (FINGERS.SIZE() > 1) {

            ULT.X = (FINGERS.GET(0).X + FINGERS.GET(FINGERS.SIZE() - 1).X) / 2;

            ULT.Y = (FINGERS.GET(0).Y + FINGERS.GET(FINGERS.SIZE() - 1).Y) / 2;

            FINGERS.SET(0, ULT);

            FINGERS.REMOVE(FINGERS.SIZE() - 1);

        }

    }

    MID = 0;

    MIDDLE.X = 0;

    MIDDLE.Y = 0;

} ELSE {

    MIDDLE.X = (MIDDLE.X + FINGERSPOINTS.GET(I).X);

    MIDDLE.Y = (MIDDLE.Y + FINGERSPOINTS.GET(I).Y);

    MID++;

```




```
    }  
    }  
    RETURN FINGERS;  
}
```

```
PUBLIC VOID DRAWFINGERSONTOCENTREOFPALM(MAT IMAGE, POINT CENTRE, POINT FINGER, LIST<POINT>  
FINGERS) {
```

```
    FOR (INT I = 0; I < FINGERS.SIZE(); I++) {  
        //DRAWS THE LINE BETWEEN EACH FINGER AND THE CENTRE OF THE PALM  
        CORE.LINE(  
            IMAGE,  
            CENTRE,  
            FINGERS.GET(I),  
            NEW SCALAR(255, 255, 255),  
            4  
        );
```

```
        //DRAWS A CIRCLE AT THE FINGERTIPS  
        CORE.CIRCLE(  
            IMAGE,  
            FINGERS.GET(I),  
            3,  
            NEW SCALAR(255, 0, 255),  
            3  
        );
```

```
    }
```

```
        CORE.CIRCLE(  
            IMAGE,
```



```
        CENTRE,
        3,
        NEW SCALAR(0, 0, 255),
        3
    );
}

@SUPPRESSWARNINGS("STATIC-ACCESS")
PUBLIC VOID GESTURES(
    LIST<POINT> FINGERS, POINT FINGER, POINT CENTRE, MAT IMAGE, PROJECTFINAL PROJECTFINAL)
THROWS INTERRUPTEDEXCEPTION {

    IF (CENTRE.X > 30 && CENTRE.Y > 30){

        SWITCH (FINGERS.SIZE()) {
            CASE 0: //NUMBER OF FINGERS
            {
                PROJECTFINAL.STRING = "FIST";
            }
            BREAK;
            CASE 1:
            {
                PROJECTFINAL.STRING = "POINTING";

            }
            BREAK;
            CASE 2:
            {
                DOUBLE ANGLE = NEW HANDCALCULATIONS().CALCULATEANGLE(FINGERS.GET(0), FINGERS.GET(1),
CENTRE, PROJECTFINAL);
```



```
IF (ANGLE <= 40) { //ANGLE BETWEEN TWO FINGERS WILL BE SMALL

    PROJECTFINAL.STRING = "PEACE SIGN";

}

ELSE IF ((ANGLE <= 90) && (ANGLE >= 45)){ //ANGLE FROM SECOND TO LAST FINGER MUCH WIDER

    PROJECTFINAL.STRING = "ROCK GESTURE";

}

ELSE

    { PROJECTFINAL.STRING = "PLEASE MAKE A GESTURE:";

        }

    }

    BREAK;

CASE 3:

    {

        DOUBLE ANGLE = NEW HANDCALCULATIONS().CALCULATEANGLE(FINGERS.GET(0),
FINGERS.GET(2), CENTRE, PROJECTFINAL);

        IF ((ANGLE <= 90) && (ANGLE >= 60)){

            PROJECTFINAL.STRING = "ROCK GESTURE";

        }

    }

    BREAK;

CASE 4:

    PROJECTFINAL.STRING = "";

    BREAK;

CASE 5:

    PROJECTFINAL.STRING = "WAVE";

    BREAK;

DEFAULT:

    PROJECTFINAL.STRING = "PLEASE MAKE A GESTURE:";
```



```
        BREAK;
    }
} ELSE {
    PROJECTFINAL.STRING = "PLEASE MAKE A GESTURE:";
}
CORE.PUTTEXT(
    IMAGE,
    PROJECTFINAL.STRING,
    NEW POINT(300, 700),
    CORE.FONT_HERSHEY_DUPLEX,
    1.5,
    NEW SCALAR(255, 255, 0)
);

}
```



```
}
```

```
IMPORT JAVA.AWT.EVENT.WINDOWADAPTER;
IMPORT JAVA.AWT.EVENT.WINDOWEVENT;
IMPORT JAVA.AWT.IMAGE.BUFFEREDIMAGE;
IMPORT JAVA.IO.BYTEARRAYINPUTSTREAM;
IMPORT JAVA.IO.IOEXCEPTION;
IMPORT JAVA.IO.INPUTSTREAM;
```

```
IMPORT JAVAX.IMAGEIO.IMAGEIO;
IMPORT JAVAX.SWING.IMAGEICON;
```

```
IMPORT ORG.OPENCV.CORE.MAT;
IMPORT ORG.OPENCV.CORE.MATOFBYTE;
IMPORT ORG.OPENCV.HIGHGUI.HIGHGUI;
```

```
IMPORT ORG.OPENCV.HIGHGUI.VIDEOCAPTURE;
```

```
PUBLIC CLASS VIDEO {
```

```
//SET UP THE FRAME FOR VIDEO
```

```
    PUBLIC VOID setFrame(FINAL VIDEOCAPTURE CAMERA, FINAL PROJECTFINAL PROJECTFINAL) {
```

```
        PROJECTFINAL.FRAME.SETSIZE(960, 768); //SET SET OF FRAME
```

```
        PROJECTFINAL.FRAME.SETVISIBLE(TRUE);
```

```
        PROJECTFINAL.FRAME.GETCONTENTPANE().ADD(PROJECTFINAL.LABEL);
```

```
        PROJECTFINAL.FRAME.ADDWINDOWLISTENER(NEW WINDOWADAPTER() {
```

```
            @SuppressWarnings("STATIC-ACCESS")
```

```
                @Override
```

```
                PUBLIC VOID WINDOWCLOSING(WINDOWEVENT E) { //WHEN CLOSING THE WINDOW
```

```
                    PROJECTFINAL.CLOSED = TRUE;
```

```
                    CAMERA.RELEASE(); //TURN OFF CAMERA
```

```
                    E.GETWINDOW().DISPOSE(); //CLOSE THE FRAME
```

```
                }
```

```
            });
```

```
    }
```

```
    PUBLIC VOID TO LABEL(MAT FRAME, PROJECTFINAL PROJECTFINAL) { //SHOWS CAMERA FEED
```

```
        MATOFBYTE BUFFER = NEW MATOFBYTE();
```

```
        HIGHGUI.IMENCODE(".JPG", FRAME, BUFFER); //ENCODE IMAGE TO BUFFER
```

```
        BYTE[] CURRENTFRAME = BUFFER.TOARRAY();
```

```
        INPUTSTREAM STREAM = NEW BYTEARRAYINPUTSTREAM(CURRENTFRAME);
```

```
        TRY { //FOR EACH FRAME
```

```
            BUFFEREDIMAGE THISFRAME = IMAGEIO.READ(STREAM); //READ THE FRAME
```

```
            PROJECTFINAL.LABEL.SETICON(NEW IMAGEICON(THISFRAME));
```

```
        } CATCH (IOEXCEPTION E) {
```

```
            E.PRINTSTACKTRACE();
```

```
        }
```

```
    }
```

██████████

██████████

}