PAPER • **OPEN ACCESS**

Hand gesture recognition on python and opencv

To cite this article: Ahmad Puad Ismail *et al* 2021 *IOP Conf. Ser.: Mater. Sci. Eng.* **1045** 012043

View the article online for updates and enhancements.

Hand gesture recognition on python and opencv

Ahmad Puad Ismail[1], **Farah Athirah Abd Aziz**[1], **Nazirah Mohamat Kasim**[1] **and Kamarulazhar Daud**[1]

[1] Faculty of Electrical Engineering, Universiti Teknologi MARA (UiTM), Cawangan Permatang Pauh, Pulau Pinang, Malaysia

Abstract. Hand gesture recognition is one of the system that can detect the gesture of hand in a real time video. The gesture of hand is classify within a certain area of interest. In this study, designing of the hand gesture recognition is one of the complicated job that involves two major problem. Firstly is the detection of hand. Another problem is to create the sign that is suitable to be used for one hand in a time. This project concentrates on how a system could detect, recognize and interpret the hand gesture recognition through computer vision with the challenging factors which variability in pose, orientation, location and scale. To perform well for developing this project, different types of gestures such as numbers and sign languages need to be created in this system. The image taken from the realtime video is analysed via Haar-cascaded Classifier to detect the gesture of hand before the image processing is done or in the other word to detect the appearance of hand in a frame. In this project, the detection of hand will be done using the theories of Region of Interest (ROI) via Python programming. The explanation of the results will be focused on the simulation part since the different for the hardware implementation is the source code to read the real-time input video. The developing of hand gesture recognition using Python and OpenCV can be implemented by applying the theories of hand segmentation and the hand detection system which use the Haar-cascade classifier.

Keywords-hand gesture, human computer interaction (HCI), contour, convex hull, convexity defects, gesture recognition, python, openCV.

1. Introduction

In a day-to-day life, hand gesture recognition is one of the system that can detect the gesture of hand in a real time video. The gesture of hand is classify within a certain area of interest. Designing a system for hand gesture recognition is one of the goal of achieving the objectives of this project. The task of recognizing hand gestures is one of the main and important issues in computer vision. With the latest advances in information and media technology, human computer interaction (HCI) systems that involve hand processing tasks such as hand detection and hand gesture recognition.

The first step in any hand processing system is to detect and locate the hand in the real-time video from the webcam. The detection of hand is challenging because of variation in pose, orientation, location and scale. Also, different intensity of light in the room adds to the variability. In the process of detection of hand, according to Mohamed [1], hand gesture recognition generally involves multiple levels such as image acquisition, pre-processing, feature extraction and gesture recognition. Image acquisition involve capturing image in the video frame by frame using a webcam. The captured images go through the image pre-processing process which involves color filtering, smoothing and thresholding. Feature extraction is a method that extracting features of the hand image such as hand contours while gesture recognition is a method to recognize hand gesture by extracting the features.

In this study, designing of the hand gesture recognition is one of the complicated job that involves two major problem. Firstly is the detection of hand. User hand is detected by using webcam in real-time video. The problem would be the unstable brightness, noise, poor resolution and contrast. The detected hand in the video are recognized to identify the gestures. At this stage, the process involves are the segmentation and edge detection. According to Karishma [2], with various information of image like color, hand posture and shape based (shape of hand) in a real time would affect the recognition of gestures. Another problem is to create the sign that is suitable to be used for one hand in a time. The extraction of hand need to be followed to determine each number and sign used. According to Amiraj [3], the extraction of hand involves the contour and convexity defects. Convexity defect gives an issue on how to calculate the depth of the defects. Some of the defects have far greater depth than others so to identify the depth would include some equations.
The list of objectives that will need to be achieve for this project: (1) to establish a complete system for detecting, recognizing and interpreting hand gesture recognition through computer vision using Python and OpenCV, and (2) to create the numbers and sign languages of hand gesture shown in the system that will meets the name of the project.

This study comprises on how to implement a complete system that can detect, recognizing and interpreting the hand by using Python and OpenCV in any intensity of light, pose or orientation of hand. In order to accomplish this, a real-time gesture based system is developed. In the proposal stage, designing a system that could detect the hand through the contour of the hand. The contour of the hand refers to the curve for a two variables for the function along which that function has a contact value that is not changed. Besides, to detect the appearance of hand in a frame of the real-time video, I will be using the Haar-cascade Classifier to track the appearance of hand before the image-processing is done. The result of the appearance of hand that the system could detect will be validated for analysis. For hardware implementation section, a wired web camera is used thus the hand gesture recognition can be done and implemented.

For this project, the prototype itself has its own limitation in terms of hardware implementation. First of all, the video capturing device used for this project can only configure up to 640 by 480 pixel resolutions. This might be the disadvantage of getting the hardware to properly work for detecting the hand. To solve this, the web camera must be in a fixed position up to 30 inches (75 cm) to help capture the video more efficiently. Another limitation to this project is the variation in plane and pose. Variations can be classified as rotation, translation and scaling of camera pose [4], [5]. The direction of the web camera is important to obtain a good video without any shadowing effects and the intensity of light in a room is enough. According to Ray [6], by selecting the correct light intensity and direction, the shadows can be reduced to a minimum.

2. Methodology
Hand gesture recognition project is done using Python programming language and OpenCV [7], [8] as library. Python programming language produces simple and easy system code to understand. Also, Python package used here is Numpy [9], [10]. The image that is captured using web camera will be processed in a region called as Region of Interest (ROI) where act as a region of wanted area while ignoring the outside region, called background [11].

2.1. Hand Segmentation Methods
According to Nobuyuki Otsu [12], segmentation is done to convert gray scale image into binary image so that only two object in image which one is hand and other is background. Otsu notes again, this algorithm is used for segmentation purpose and gray scale images are converted into binary image consisting hand or background. A very good segmentation is needed to select an adequate threshold of gray level for extract hand from background for example there is no part of hand should have background

and background also should not have any part of hand. In general, the selection of an appropriate segmentation algorithm depends largely on the type of images and the application areas. The Otsu segmentation algorithm was tested and found to give good segmentation results for the hand gesture. Otsu algorithm is nonparametric and unsupervised method of automatic threshold selection. The basic theory of segmentation of hand is by using the equation as follow:

$$p_i = \frac{n_i}{MN}, \qquad p_i \geq 0, \quad \sum_{i=1}^{L} p_i = 1 \tag{1}$$

where the pixels of a given picture be represented in L gray levels$[1, 2, 3, \dots \dots \dots \dots \dots, L]$ in a digital image of size $M \times N$, and number of pixels at level i is denoted by ni and the total number of pixels in the image is $MN = n1 + n2 + n3 + n4 \dots \dots \dots \dots .. nL$. The normalized histogram has components of p. The formula above was adapted from Rafael [13]. To apply the formula (1), the image of each frame that has been captured from real-time video are considered as pixels and being compared to each other of the frames. As each frames contains equal value of colour in a pixel, the pixel is then considered as eliminated or removed. These applied to other pixels within each frame, and each frames are compared with respect to time. As any of the pixels between two frames do not match with each other, the pixel at the first frame retain its contain. Otsu's thresholding method involves iterating through all the possible threshold values and the measurement of spread for the pixel level for each side of the threshold is calculate for example pixels that either fall in foreground or background. The aim is to find the threshold value where the sum of foreground and background spreads is at its minimum. Figure 1 shows how the hand image is obtained from the real-time video in a gray scale filter and then the image is converted for Otsu's algorithm that showing the thresholded image.

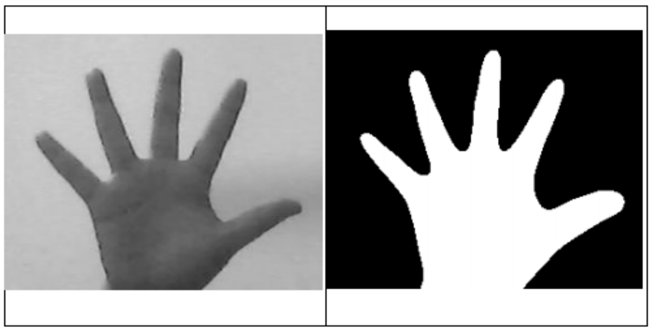
Figure 1. Comparison between gray scale image and thresholded image

The limitation of the Otsu's thresholding method is that as I previously mentioned, this algorithm only consider a bimodal image where bimodal image is an image whose histogram has two peaks. For that image, Otsu's does take a value in the middle of those peaks as threshold value. Then, it automatically calculates a threshold value from image histogram for bimodal image. This only work when a frame capturing a frame from video and converting the RGB format into gray scale filter.

2.2. Track Gesture Of Hand Using Haar-Cascade Classifier

In order for the program to continuously detect the hand at all times, especially when the hand is in static or not moving around, the program needs to learn the hand image. This is where the machine learning comes in. In this project, the machine learning technique was performed to track the gesture of hand before the image processing is done. As suggested by Sander [14], the suitable machine learning technique in term of image processing technique that is much available to use is called as cascade classifier. According to Viola [15], the cascaded classifier are actually the program features for the machine to know the exact object that will need to classify in each frames of images. One example that will be use in this project is Haar-cascaded type. Viola [15] again suggested that Haar-cascaded classifier was the simple classifier for machine learner to detect any object.

2.3. Region of Interest (ROI)
Region of Interest, or commonly abbreviated as ROI, is a region that will be considered as the region of wanted area while ignoring the outside region, called background. As proposed by Sander [14], to detect the availability of hand recognition, the detected region must be located the same as the region of interest. In other words, as the classifier tracks the wanted object, in this case the bounding rectangle around the hand contour, the area of hand must overlap with the region of interest. By using the conditional method of overlapping region, the system can identify whether the region selected, in this case the rectangle around the hand contour, is overlap with other active region, in this case the hand itself.

By developing this theoretical part of ROI to the simulation and hardware implementation, the process of detecting the appearance of hand will be much simplified compared to the other method like in [16]. Figure 2 shows the criteria for ROI to overlap that can be considered as overlapping ROIs. These criteria are needed to make sure that the program can identify the detection of hand gesture.

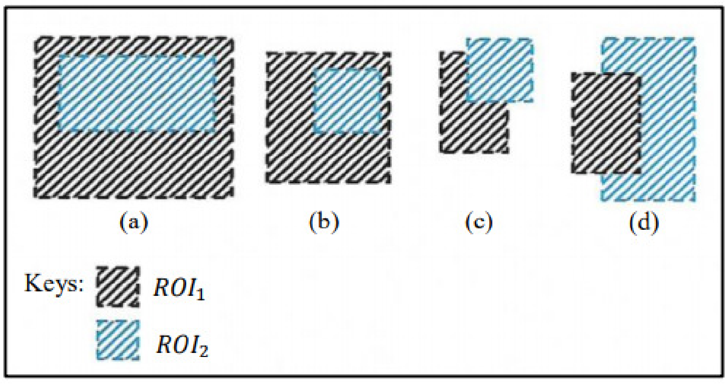

Figure 2. Overlapping Region of Interest (ROI) [16]

For this project the Region of Interest (ROI) is used to see the gesture of hand because the implementation for the ROI is suitable to detect the overlapping two regions on it. Besides, the system will detect only two outputs for each hand gesture recognition. This can be explained in Table 1 as follow:

Table 1. Application of Region of Interest (ROI).

Region of Interest (ROI)	Region implemented for
ROI_1	An active region, in this case the hand itself.
ROI_2	The rectangle sub window on screen. This region is static or not movable outside the frame of the realtime video and it highlights the area that involves.

2.4. Implementation on Simulation and Hardware prototype
The detection of hand gesture was calculated through the space consumption within the area between the convex hull and contour of hand. Convex hull is apply to find the edge of user hand by forming descriptor in which the smallest convex set contains the edges. In this project, the OpenCV library files that is suitable for any of image processing techniques are used.

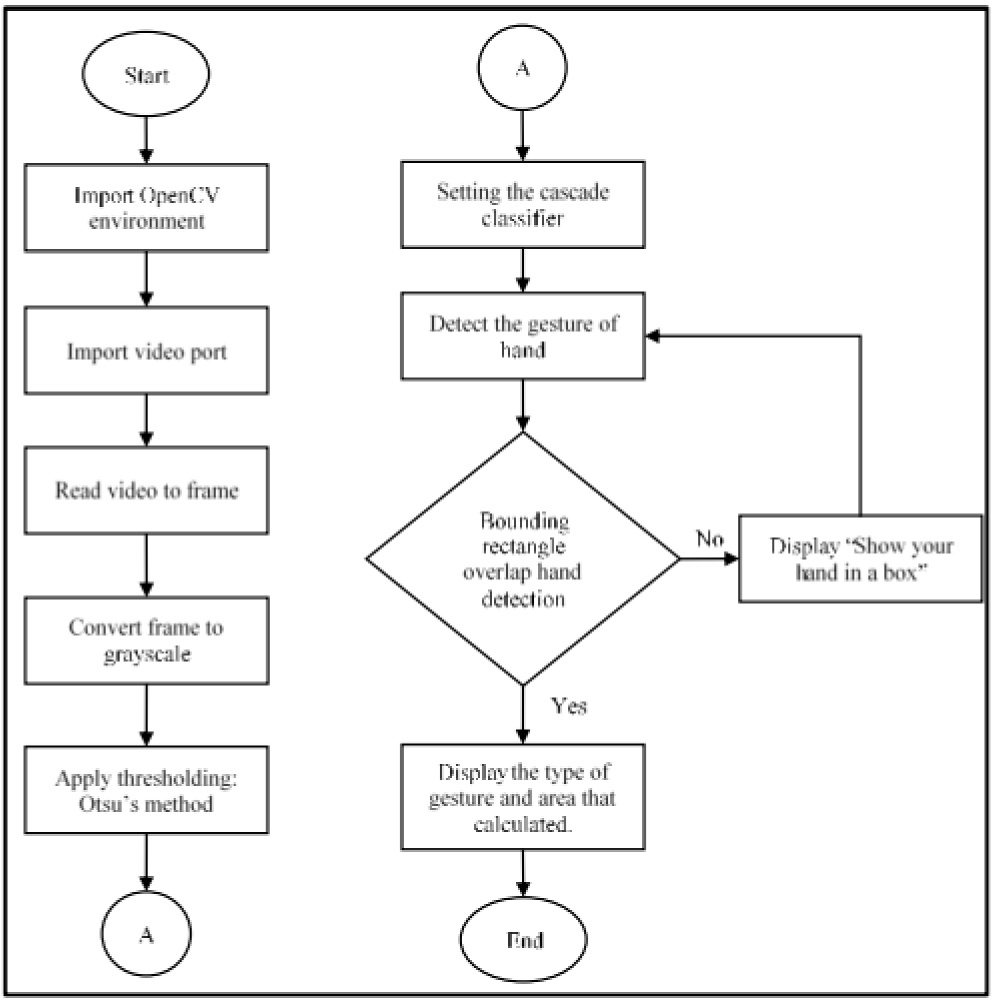
Start
Import OpenCV environment
Import video port
Read video to frame
Convert frame to grayscale
Apply thresholding: Otsu's method
A
A
Setting the cascade classifier
Detect the gesture of hand
Bounding rectangle overlap hand detection
No
Display 'Show your hand in a box"
Yes
Display the type of gesture and area that calculated.
End

Figure 3. Flowchart of Python file for hand gesture recognition

By referring to the flowchart in Figure 3, generating a Python file for both simulating and hardware prototype can be implemented. The main program action to detect the appearance and the gesture of hand is by using the theory of Region of Interest (ROI). For programs to track the rectangle around the hand contour, the program required to identify whether the selected area of interest, referring to the case of the project, the rectangle will overlap with the detection of hand. Uncertainty it is overlap, it is considered as showing the type of gestures such as number or sign and the area within the area of region of interest. Uncertainty there is not overlap (outside the region of interest), the program can be considered as "Show your hand in a box". To plan the region of interest for this project, a group of coordinate will be recognized to present the area of bounding rectangle around the hand contour. For this project, there is an area of bounding rectangle, therefore there is a region of interest to be implemented with each type of gestures. After generate the Python main file, now the simulation begins where the detection of hand will be tested thus recognizing the type of gestures shown based on the selected area of interest. To run the Python file, the OpenCV environment must be link and run through the command window.

Then, the same procedures applied for the hardware prototype. For this prototype, USB video web camera will be (a), (b), (c), and (d) Keys: $ROI1$ $ROI2$ used as Nayana [17]. The source code in the Python core need to be modified in the video capture section to change the camera from the laptop web camera to the USB Video web camera. The procedures of detecting and analysing the gesture of hand using the USB video web camera remains the same during the simulation part. Notice that OpenCV environment is used once again to detect the hand via Haar-cascade classifier.

3. Results And Discussion

The explanation of the results will be focused on the simulation part since the different for the hardware implementation is the source code of the real-time input video. These results are analysed throughout the project scope. At first, the result of Haar-cascade classifier will be explain in detail to show how the hand is detected before image processing. Then, the simulation part begins where the area of interest that has been calculated. The Python file is actually the command file to run the specified program onto the Python command script. Table 1 shows the details of the generated Haar-cascade classifier.

Table 2. Explanations of Haar-cascade classifier file

Points	Explanations
(1)	This section covers the type of classifier to be used in this project. For this case, it will use the OpenCV and Haar-cascade classifier methods.
(2)	This is the part where the window size plays a role. The window size means the size of the specified hand. By default, it is fixed to 24 by 24 pixels.
(3)	The variation of the threshold and window detection are set. Generally, this will be based on the type of camera that will be used to detect hands and recognize gestures.
(4)	Each of the posture, about 10 times of repetition to analysing the cascade classifier.

In this project, the total data collected around 10 times of repetition with different scales for each posture to determine that the bounding rectangle could detect the hand gesture or not. Figure 4 shows how convex hull is apply to find the edge of user hand.

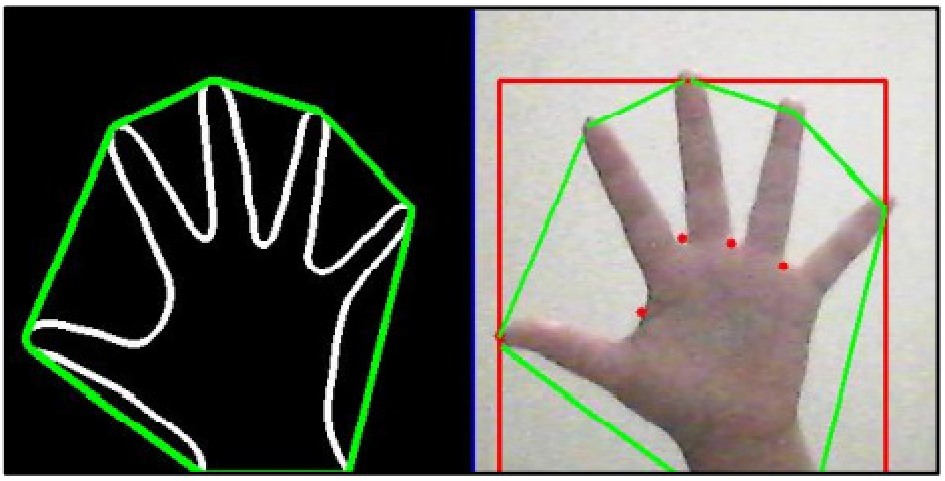

Figure 4. Left figure shows the background (in black), the hand-shape region (in black), the hand contour (in white) and the convex hull (in green). On the right, convexity defects (in red).

Initially, convex hull compares with hand contours to detect convexity defects where defects are the point where both are different. Firstly, vertices of the convex hull are calculated where they change direction. Then, the hand contour segment between consecutive pairs of consecutive vertices of convex hull and finds that pixels in each segment maximizes the distance from the convex hull. This maximum distance is called the defect depth where the dots are called convexity defects.

The proposed hand gesture recognition requires a webcam to capture the user's gesture, preferably with clear background. The test is done under normal lighting, the system can recognize and track the user's hand movement. The segmentation process is able to detect user's movement and count the contours. Each signal differentiates using different ROI. Figure 5, Figure 6 and Figure 7 shows the output when execution of Python file.

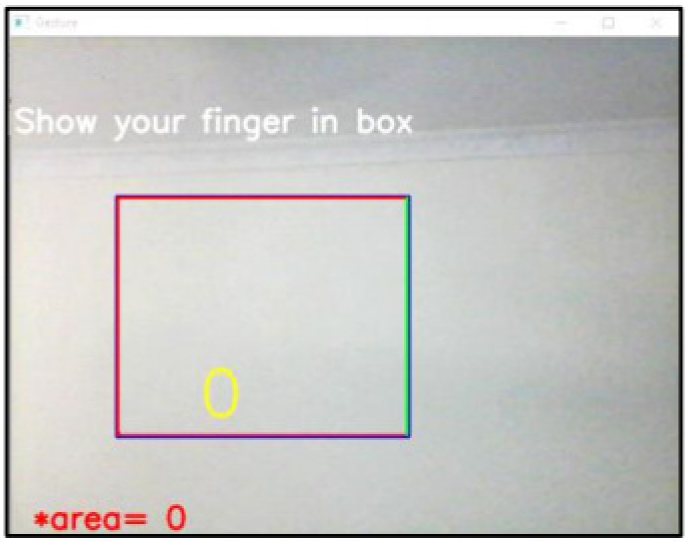

Figure 5. Image captured by web camera

Figure 6. Bounding box without any gesture shown

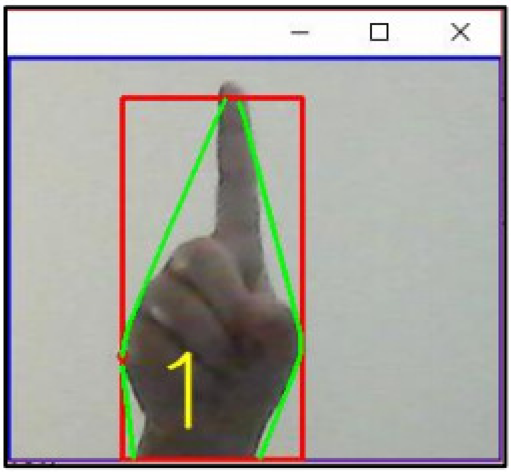

Figure 7. Recognition of hand showing number 1

All of the numbers and signs were tested until number 5. The experimental results are tabulated in Table 3.

Table 3. Analysis table for Region of Interest

Events	Area in Region of Interest, ROI		
	Area of Hull, H	Area of Contour, C	Area = H - C
0	62879	62879	0
1	13569	10876	2693
Good	13045	10821	2224
2	20646	13744	6902
Gun	20216	13804	6412
3	24362	17113	7249
OK	25468	16922	8546
4	25781	15395	10386
Rawr	20395	14148	6246
5	28383	16822	11561
Stop	33220	19047	14173

To evaluate the area of recognition proposed, an area by Region of Interest is taken and calculated in pixels value. Based on observation, most of area of number gestures have larger area compared to gestures recognition of sign.

Table 4. Analysis table for gesture recognition

Events	Performance of Haar-cascade Classifier		
	Hits	Missed	False
1	10	0	0
Good	10	0	0
2	10	0	0
Gun	10	0	0
3	10	0	0
OK	10	0	0
4	10	0	0
Rawr	10	0	0
5	10	0	0
Stop	10	0	0

Whilst in Table 4, the total data collected around 100 samples with different scales for each posture. Each of the posture with different scale were captured with 10 times to analyse the gesture. Haar-cascade Classifier determine whether the box in the frame could detect the hand gesture or not. Based on all of the different sign, the box in the frame were recognizing all of the sign.

However, there are some limitations of the program, i.e. the lighting environment changes are still influential in the segmentation process, in particular the process of background removal. Also, the user's hand and the webcam must be in a fixed position so the webcam will capture the image with the exact area to identify the gesture.

4. Conclusion

As a conclusion based on the result of the project, it can be seen that developing the hand gesture recognition using Python and OpenCV can be implemented by applying the theories of hand segmentation and the hand detection system which use the Haar-cascade classifier. To summarize it, this system has accomplished several objective in this project: (1) manage to establish a complete system for detecting, recognizing and interpreting hand gesture recognition through computer vision using Python and OpenCV, and (2) able to create the numbers and sign languages of hand gesture shown in the system that will meets the name of the project.

For the future recommendation, this system will include the execution of additional gestures that will allow any users with different skin colour and size of palm to perform more functions easily. The current system only uses the right hand with specific area in ROI to perform gestures. Therefore, the desired enhancement of the technique may possibly using both of user hands to perform different signs with computer operations. Additionally, background subtraction algorithms can be used for more effective performance. Implementing the Graphical User Interface (GUI) will be done in the future where the users know how to translate the gesture from its meaning to the sign or number and vice versa.

5. Acknowledgement

The authors wished to thank Universiti Teknologi MARA (UiTM), Cawangan Pulau Pinang, Malaysia for the instrumentations and experimental facilities provided as well as Faculty of Electrical Engineering UiTM Cawangan Pulau Pinang for all the support given during this research.

6. References

[1] S. M. M. Roomi, R. J. Priya, and H. Jayalakshmi 2010 *Hand Gesture Recognition for Human-Computer Interaction* (J. Comput. Science vol. 6) no. 9 pp. 1002–1007.
[2] S. N. Karishma and V. Lathasree 2014 *Fusion of Skin Color Detection and Background Subtraction for Hand Gesture Segmentation* (International Journal of Engineering Research and Technology) vol. 3 no 1 pp 13–18.
[3] A. Dhawan and V. Honrao 2013 *Implementation of Hand Detection based Techniques for Human Computer Interaction* (International Journal of Computer Applications) vol. 72 no. 17 pp 6–13
[4] C. Von Hardenberg and F. Bérard 2001 *Bare-hand human-computer interaction* (Proceedings of the 2001 workshop on Perceptive user interfaces) pp 1–8.
[5] K. Nickel and R. Stiefelhagen 2007 *Visual recognition of pointing gestures for human–robot interaction* (Image Vis. Comput.) vol. 25 no. 12 pp 1875–1884
[6] R. Lockton 2002 *Hand gesture recognition using computer vision* (4th Year Proj. Rep.) pp 1–69
[7] Itseez 2017 *Open Source Computer Vision Library itseez2015opencv* (Online) Available: https://github.com/itseez/opencv (Accessed: 13-Oct-2017)
[8] Itseez 2017 *The OpenCV Reference Manual itseez2014theopencv 2.4.9.0* (Online) Available: http://opencv.org (Accessed: 13-Oct-2017)
[9] S. van der Walt, S. C. Colbert, and G. Varoquaux 2011 *The NumPy array: a structure for efficient numerical computation* (Comput. Sci. Eng.) vol. 13 no. 2 pp 22–30
[10] Travis E. Oliphant et al 2017 *NumPy Developers numpy 1.13.0rc2*. (Online) Available: https://pypi.python.org/pypi/numpy (Accessed: 13-Oct-2017).
[11] S. Gokturk C. Tomasi B. Girod C. Beaulieu 2001 *Region of Interest Based Medical Image Compression with Application to Colon Ct Images* (International Conference of the IEEE

Engineering in Medicine and Biology Society) vol 23 pp 575-578
[12] N. Otsu 1979 *A Threshold Selection Method from Gray-Level Histograms* (IEEE Trans. Syst. Man. Cybern.) vol. 9 no. 1 pp 62–66
[13] R. C. Gonzalez 2016 *Digital image processing* (Prentice hall)
[14] S. Soo 2014 *Object detection using Haar-cascade Classifier* (Inst. Comput. Sci. Univ. Tartu) vol. 2 no. 3 pp 1–12
[15] P. Viola and M. Jones 2001 *Rapid object detection using a boosted cascade of simple features* (IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognition. CVPR 2001) vol. 1 pp 1511-1518
[16] M. Nahar and M. S. Ali 2014 *An Improved Approach for Digital Image Edge Detection* (Int. J. Recent Dev. Eng. Technology) vol. 2 no. 3
[17] P. B. Nayana and S. Kubakaddi 2014 *Implentation of Hand Gesture Recognition Technique for HCI Using Open CV* (International Journal of Recent Dev) vol. 2 no. 5 pp 17–21