ELEKTROPIKA: Int. J. of Electrical, Electronic Engineering and Technology, Vol. 2, 29–46, 2012

A Vision-Based Human Hand Gesture Recognition Interface for Image Browsing

Lih Yang Chan[1], **Bee Ee Khoo**[2]

School of Electrical and Electronic Engineering,
Universiti Sains Malaysia, Engineering Campus,
14300 Nibong Tebal, Pulau Pinang, Malaysia

E-mail: [1]clihyang@gmail.com; [2]beekhoo@eng.usm.my

Abstract—We present the implementation of hand gesture recognition user interface for image browsing application. The hand-gesture interface enables users to view and manipulate image with hand gestures and contact free from input device in real time. The lower level of the approach implements the posture recognition with Viola-Jones object detection method that utilizes Haar-like features and the Adaboost learning algorithm. With this algorithm, real-time performance and high recognition accuracy up to 94% detection rate can be obtained. The users hand gesture interpretation yield average of 89% successful input command in a series of evaluation. To further enhance the speed of hand detection in real-time application, an idea to reduce the area of search window by incorporating skin color segmentation is proposed in this project. A reduction of 19% of processing time is achieved with the proposed method, comparing to the processing time without skin color segmentation. The system is tested for robustness and responsiveness. The results show satisfactory performance for the system to work in real-time.

Keywords: Real Time, Viola-Jones, skin color segmentation

1. INTRODUCTION

The rapid growth of computerization has made human-computer interaction essential in daily life. However, keyboards and mice as principle method of human-computer interaction today hinder expansion of the computer's abilities to serve humans in many aspects of life. Moreover, the devices are easily contaminated with bacteria in public computer such as hospital [5] and spreading infection from one to others. The study done by Schultz, Gill, Zubairi and Huber [9] shows that 95% of keyboards in clinical areas are contaminated with harmful microorganism. The contactless vision based gesture recognition input is one of the solutions since users do not need to touch or hold any device with this method. Meanwhile, it is a free and natural way of Human Computer Interaction (HCI). But, it is challenging to promote the growth of hand gesture-based input application. First of all, the vision-based gesture recognition implementation cost has to be comparably equal or lower than normal input devices cost like keyboard, in order to encourage the gesture-based input application deployment. Besides, the vision-based hand gesture's recognition accuracy is lower and varying under different environment, compared to devices based input which is consistent over different condition [10].

Noticeably, many works have been done to make vision-based gesture recognition system feasible. But, the better usability of recognition system always comes with higher cost of deployment. For example, Yen-Ting Chen, Kuo-Tsung Tseng [11] and Hongo, H, et al [12] employ multiple camera to achieve desirable accuracy but indirectly increases the cost of implementation when more hardware and processing power are required. On the other hand, there are numerous works that successfully implement the gesture recognition HCI at lower cost but usability is compromised where the users are bounded to certain limitation that defeats the purpose of hand gesture recognition HCI. For instance, [13] employs fast feature extraction with single camera but requires users to rigidly aligned hand to camera. Yuanxin Zhu *et al.* [14] gesture recognition works only for users who wear long sleeve shirts.

Therefore, a HCI system needs to be developed to integrate suitable hand gesture recognition techniques together to achieve better usability and lower cost at the same time. Hence, in order to promote vision-based hand gesture recognition, this paper's main motivation is to develop a gesture HCI system that balance between usability and cost. We present a hand gesture recognition HCI integrated image browsing application, which works in real time at personal computer using low cost webcam. Compared to previous similar works, this system does not need initialization to locate user's hand because of its hand localization feature. This paper is also a proof of concept of accelerating hand localization by narrowing search area with color segmentation. This paper is organized as follows. Section 2 discusses recent related works in vision based hand gesture recognition. The training process of classifier is revealed in Section 3. Then, in section 4, the color segmentation process is explained. The system architecture is reviewed in Section 5. The system is evaluated for its performance under various conditions in Section 6. The result of evaluation is being discussed in Section 7. Finally, this paper is ended with conclusion in Section 8.

2. RELATED WORKS

There are many HCI systems implemented based on the vision based hand recognition method. For instance, Kolsh and Turk [2-3] successfully implemented the real time hand gesture recognition using Viola-jones object detection method which uses Haar-like feature for wearable computing and virtual environments. The system, named as HandVu successfully demonstrated the feasibility of vision based gesture recognition HCI and "deviceless" interaction capability. Wachs, Helman and Edan [6] implemented hand gesture recognition input method using webcam at PC which enables doctors to take control of mouse cursor in Windows environment with hand gesture. This system is accomplished by using Haar-like feature and Fuzzy c-means classifier. However, these projects approach requires manual initialization, where users need to place their hand in a designated window to kick start the hand tracking. Recently, Qing Chen *et al.* [8] use similar methods to solve the problem of real-time vision-based hand gesture. His idea is to divide the recognition problem into two levels according to the hierarchical property of hand gestures. The lower level of the approach implements the posture detection with a statistical method based on Haar-like features and the Adaboost learning algorithm. The postures that are detected by the lower level are converted into a sequence of terminal strings for high level gesture classification based on probability. His experimental result shows average of 95.75% detection rate for four static postures recognition based on an experiment with 100 sample hand images for each posture. Other than that, the hand detection in "Self-correcting tracking for articulated objects" [15] proposed by Baris Caglar and Niels provides robust methodology to detect hand through edges segmentation with curve and parallel bars detection. However, this approach requires high resolution camera and is not feasible for relatively low resolution camera like webcam.

3. CLASSIFIER TRAINING

To recognize a hand gesture, a classifier is required to differentiate the class of the input data. In this paper, the classifier uses Viola-Jones object detection method. The important elements in Viola-Jones method are appearance based Haar-like features and Adaboost learning algorithm. This method is suitable for real time implementation because Haar-like feature extraction involved only simple mathematic manipulation. Adaboost is proven to be strong and robust in training classifier [4]. As part of the Viola-Jones detection method, the integral image is implemented to speed up

Haar-like feature extraction. Firstly, the image acquired from webcam is converted from RGB to grey scale before integral image is constructed based on the Equation 1:

$$ii(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y') \tag{1}$$

Where ii is the integral image array, i is the grey scale image array, x, y, x', y' are referring to xy coordinate in the array of the image. There are four types of Haar features are used for classifier training as shown in Figure 1. The value of feature is the difference sum of all pixels in grey to white area. During implementation, the feature value is calculated based on the integral image

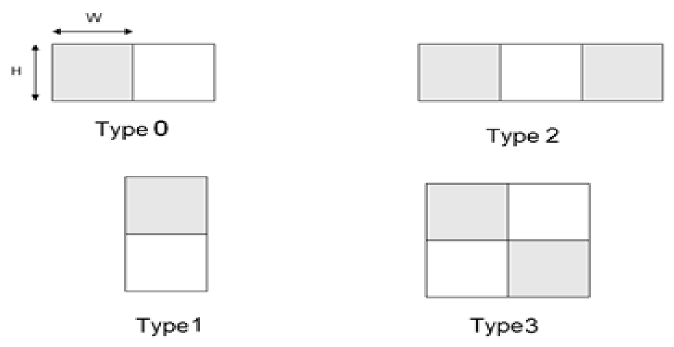

Figure 1: Four types of feature implemented in the classifier

constructed earlier. The computation of the feature involved only simple addition and subtraction.

The instances consist of feature from different location in the image by varying the X and Y based on the image size. To determine optimized H and W, three ranges of H and W is nominated which are 25%, 50% and 75% of the window size. Such value is selected because of the detection is operating under low resolution webcam where the biggest window size is only 84 $\times$ 84 pixels. The difference between sizes of 12.5% to 25% is only 10 pixels, and hence 12.5% is redundant. The Viola-Jones object detector [1-2] is trained with samples of positive and negative images in grey scale. The positive images are images with 4 types of hand postures, while the negative images consist of other type of hand postures.

To prepare the data, for training the samples images of the four posture to be recognized are collected from 5 individuals as shown in Figure 2. The four static gestures which are used to give commands are "Open", "Close", "L" and "V". Figure 3 and 4 show the sample of positive and negative images of Open posture respectively. The classifier training is done using GML Adaboost toolbox in Matlab platform based on Adaboost algorithm. The four types of postures are trained separately to yield an independent classifier at the end result. In the first round of training, the weight of each training sample set is set equal. In order to select a weak learner from the pool of instances, errors of each instance are calculated as sum of weight of misclassified sample set at a nominated threshold. Then the instance with lowest error is selected as the weak learner for the particular training round. The threshold of weak learner is set by referring to the lowest error in the distribution as shown in Figure 5. The misclassified sample set under the initial training round is given higher weight at the next round of training. Therefore, the error is reduced during next training round. Figure 6 shows the error rate of four feature types is reduced over each round of training for "Open "posture. However, feature type 2 is exceptional as it is a weak feature that hardly differentiates positive and negative data. After 200 rounds of training, those three feature type error are stabilized at nearly 6%. After the training is completed, the threshold and the weak learners are saved along with its weight in hard drive in order to be loaded during classification in the system.

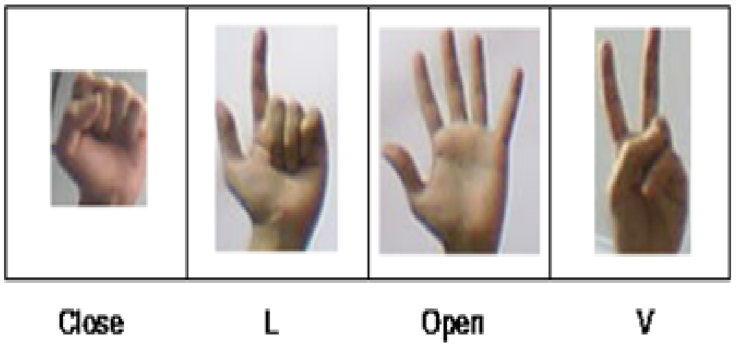

Figure 2: Four types of gesture which is recognized by the system

Each learner utilizes different instance which is defined during Adaboost training. The learner outputs either 1or 0 result by comparing to its threshold. Then, the output of learner is multiplied with its weight and all the product of multiplication is summed up. The number of learner, n is depending on the number of training iteration, t during Adaboost training. Putting the classifier in equation as shown in Equation 2 [7]:

$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right) \tag{2}$$

After training, the learner performance is evaluated and limits are optimized with Receiver Operating Characteristic (ROC) curve. The learner is tested with 100 of positive and negative sample images respectively as control. The ROC curve is plotted by varying the classifier limit to collect the detection rate over false positive percentage with the 200 sample images. Detection rate is the percentage of positive image detected over the total positive image, while the false positive percentage is the percentage of false classification of negative image as positive one. The total error percentage is calculated as the number of falsely classified image either positive or negative over total number of tested image. In "Open" posture training, the learner of feature type 1 and 3 are the best among the four where the detection rate at 97% with false positive rate at 8% as shown in Figure 7. In "V" posture training, the learners have 93% detection rate in average at 10% false positive rate as shown in Figure 8. In "L" posture training, the learners have 98% detection rate in average at 5% false positive rate as shown in Figure 9. As "Close" posture has less instances per feature type, all four type of features are trained together into single learner. The combined learner achieve 97% detection rate at 6% false positive rate as shown in Figure 10. The four types of learners are cascaded into a single classifier after limit is optimized. Then it is tested with the 200 control samples and results yield are as shown in Table 1.

As the final outcome, the hand posture detector consists of four classifiers that cascades in a row where each classifier processes different types of features. The flow starts with Integral image construction from the image acquired from the webcam in real time. Then features type 0 is extracted from different location in the integral image by searching in the window of skin color region (during initialization state) or region around last detected hand gesture (during tracking). The results of classification are then passed to the next type of feature extraction. However, this time, the feature extraction does not sweep through the whole integral image. Instead, it is only extracted from the location of the previous positive detection. For example, there is data set from 160 locations which are fed into the type 0 classifier. If 30 out of 160 data sets are classified as positive detection by type 0 classifier, then, the next kind, i.e. type 1 classifier only extracts and processes the data from those 30 locations. Therefore, data processing time is reduced at subsequent

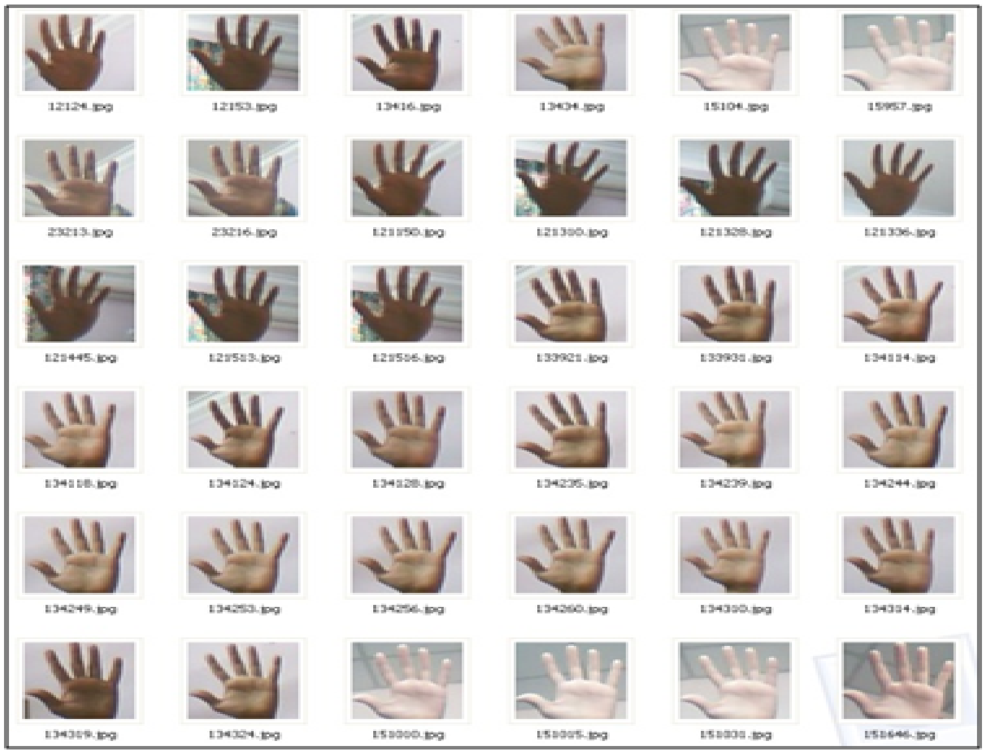

Figure 3: Sample of "Open" gesture positive image

Table 1: Detection rate and error of each posture cascaded classifier after limit tuning

Posture	Detection Rate%	False Positive %	Total error %
Open	96.8	4.8	4
Close	96	3	3
L	96.45	0	1.5
V	91.41	1.8	4.7

cascaded classifier by avoiding the processing of the 130 negative data. The classification process is repeated for the rest of the three feature types. The data set that finally passes all four feature classifications is taken as successful detection. The overall classification flow is shown in Figure 11.

4. COLOR SEGMENTATION

The purpose of color segmentation is to narrow down the search area to reduce the processing time instead of performing high precision color classification. The skin color segmentation is implemented by first converting the image into HSI color space from RGB. Then, each pixel is determined whether

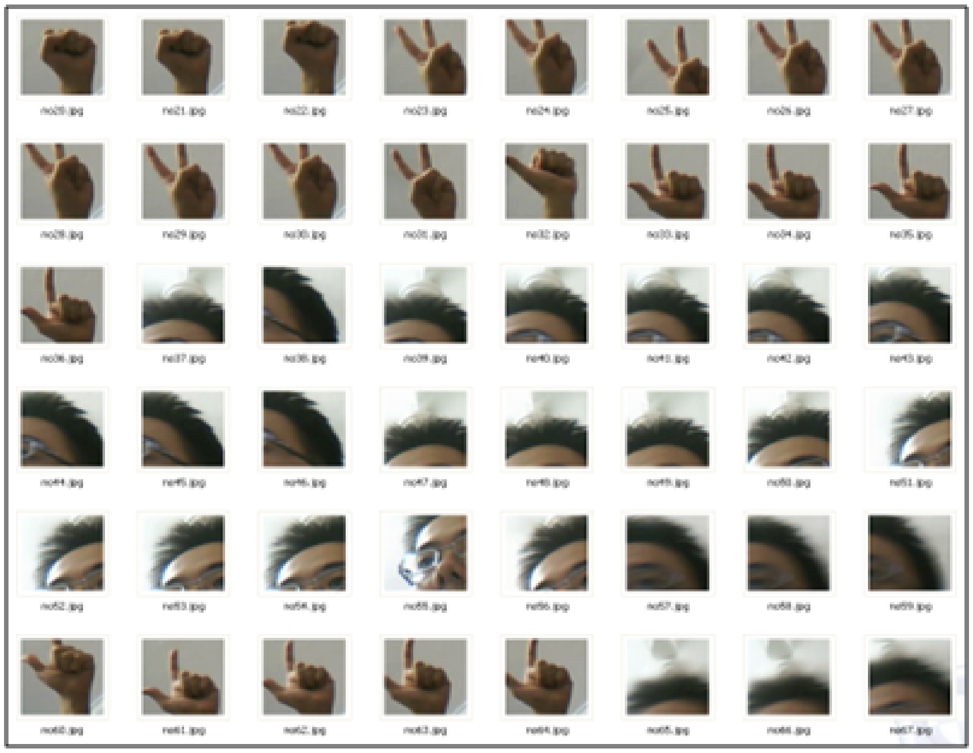

Figure 4: Negative image samples for "Open" posture training consists of some other hand postures and body parts

it is a skin color by referring to Look-up table with H and S value. The look-up table (LUT) is basically a two-dimensional skin color histogram that each dimension represents Hue and Saturation respectively in HSI color space. The histogram is constructed based on 2 megapixels of blob skin color sample of five individuals collected from various distance of hand between cameras, ranging from 75cm to 125cm. The segmented image is morphologically processed with dilation using 48x48 masks for noise removal. The found skin color region is always shifted to upper left by 40 pixels to produce the search window to match the reference point of classification.

Table 2: Processing time comparison between two methods

Process	With Skin Segmentation	Without Skin Segmentation
Acquire image	0.016	0.016
Color segmentation	0.25	0
Integral image	0.172	0.172
Classification	0.375	0.812
Total time (seconds)	0.813	1

The segmentation processing gain is evaluated with CPU with AMD Athlon 1.83GHz. The processing time required by CPU to execute the detection is recorded using "Tic Toc" functions

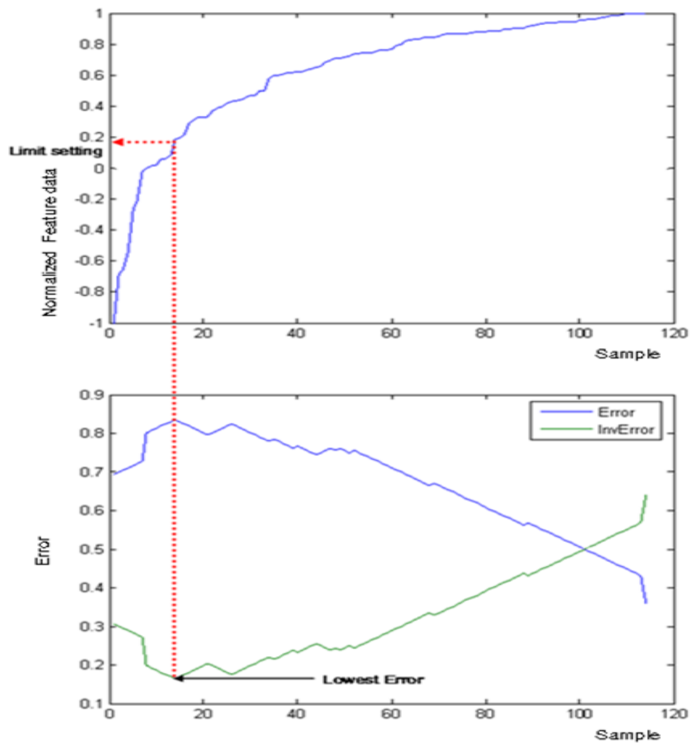

Figure 5: Example of threshold setting for weak learner based on the point of lowest error

in MATLAB 7.0 and it is shown in Table 2. The classification time with skin color segmentation varies depending on the size of search window which is constructed based on the amount of skin color appeared in the image. Therefore, the average of 10 readings is collected from sample images with skin color covering nearly 30% of whole image. From the processing time comparison, it shows that there is 19% of time reduction with skin color segmentation search window.

5. SYSTEM SPECIFICATION

The application is built on Matlab 7.0 GUIDE under Microsoft Windows XP operating system environment in PC. GUIDE is MATLAB graphical users interface (GUI) development environment which provides tools to lay out GUI. The PC is connected to a 320 × 240 pixels webcam through USB interface. At this camera resolution, user is expected to be giving command at a distance within one meter from the webcam using his/her left hand.

There are four windows in the application as shown in Figure 12. Each window has it functionality, namely the image viewer, control panel, Image thumbnail preview and live webcam view. Image viewer is the window displaying the image under viewing by user. Control panel contains button to initialize webcam driver in MATLAB, loads the classifier limits from training database, activates and deactivates the system timer. The image thumbnail preview window shows the previous and next picture preview.

There are six commands in this application: Next image, previous image, region selection, zoom, pan and contrast increment. The commands are mainly static hand gesture and combined with a little temporal hand gesture for region selection box navigation and panning. The four static gestures which are used to give commands are "Open", "Close", "L" and "V" as shown in Figure 2. Every gesture is kick started with initialization by showing "Open" posture to allow the system

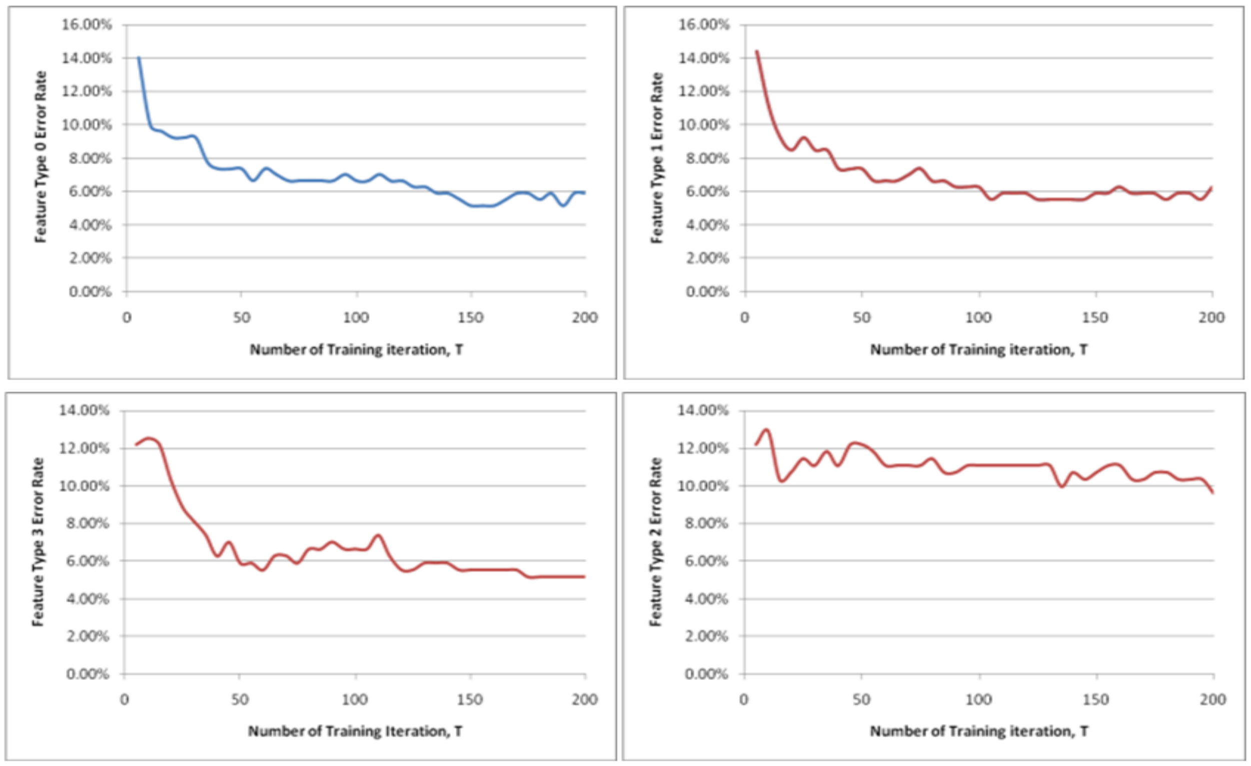

Figure 6: Feature error rate are improved over each training round except Feature Type 2

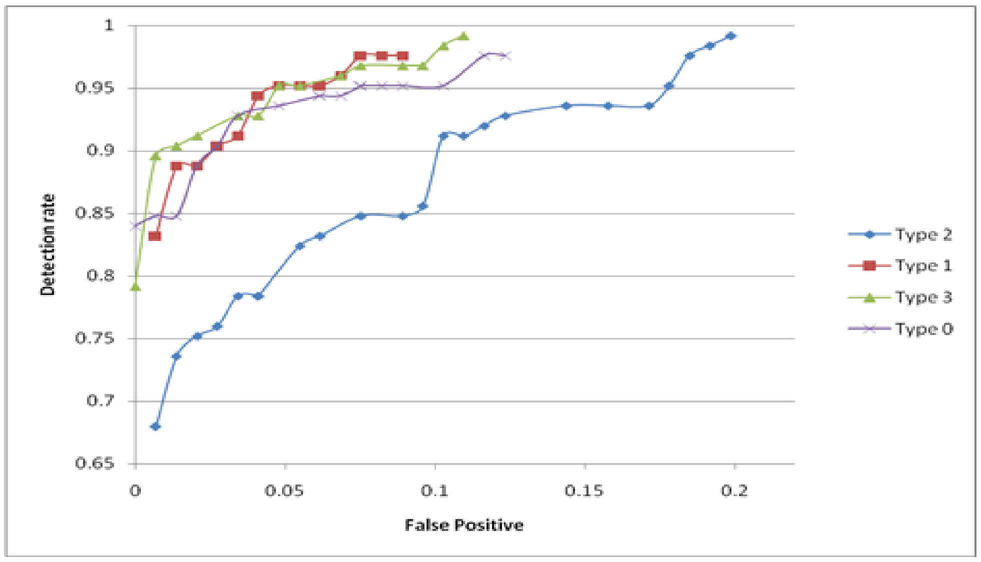

Figure 7: "Open" posture learner ROC curve

locates the user hand with the view of webcam. Whenever the system located the hand position, which usually takes less than a second, users can start to give commands with the hand gestures.

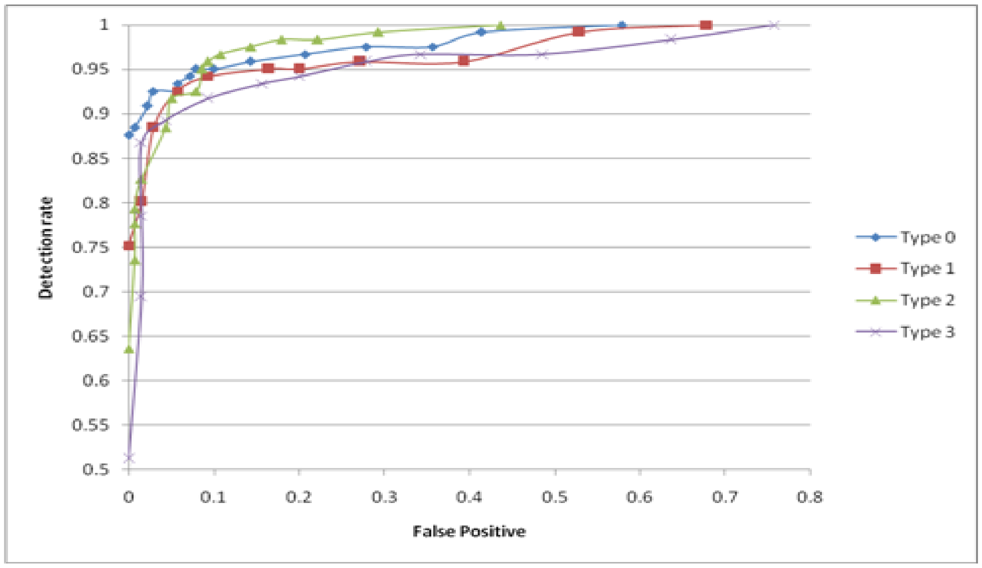

Figure 8: "V" posture learner ROC curve

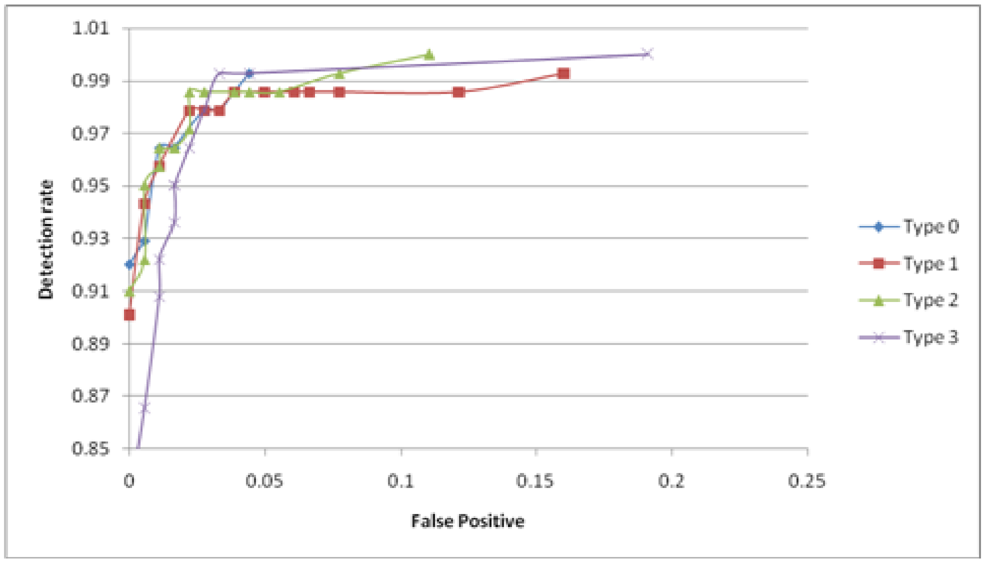

Figure 9: "L" posture learner ROC curve

5.1. System Work Flow

The system flow consists of five states which are Initialize, System Start, Region selection, Browse, and Image Zoomed. Actions or commands are executed during transition between those states when gestures are detected. The overall state machine work flow is illustrated in Figure 13.

State 0: When the clock is activated, the system is started in state 0. In this state, the system is searching for "open palm" gesture in skin color region. Whenever an "open palm" is detected, the coordinate is recorded and then proceeds to state 1.

State 1: Two types of gesture are expected in state 1, which are "Open" and "Close" palm. The detector

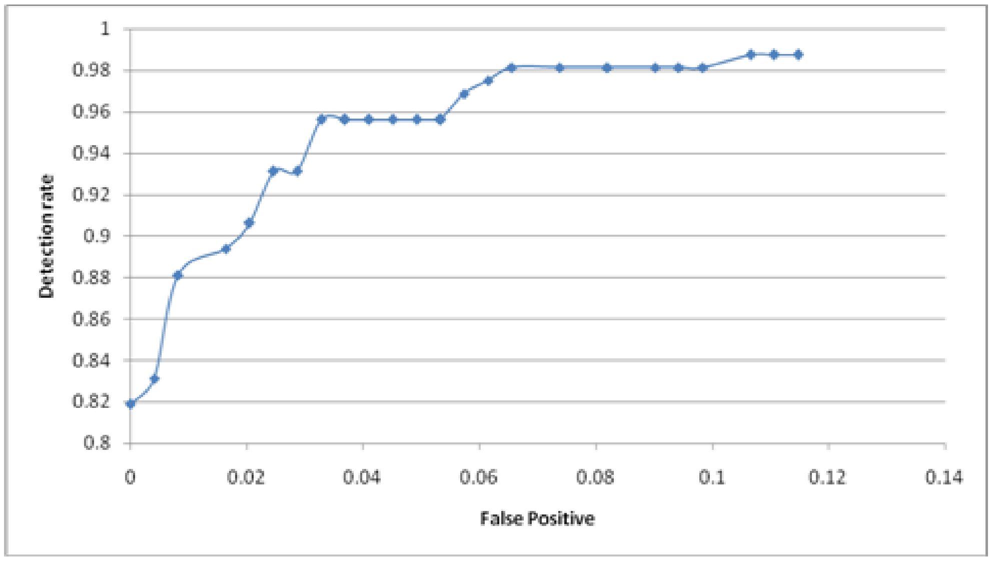

Figure 10: "Close" posture learner ROC curve

is searching for the gesture within an 84×84 window. The location of searching window is the XY coordinate from state 0 where "open" palm is found. If "open" palm is detected for 3 times consecutively, the system will enter into State 3. If "Close" palm gesture is detected instead, a red color region selection box will appear in the display image before the system enters into State 2.

State 2: In this state, the system is expecting the user to move the region selection box using "Close" gesture to the area where he/she wants to zoom in. While moving the selection box, the system stays in State 2 and the red selection box location controlled by user's "close" hand gesture movement. So, the user is able to navigate the selection box to the region where he/she wants and zoom in the image by "Open palm" gesture.

State 3: In browsing mode, the user can scroll through the images using 2 gestures - "Open" for next image and "V" for previous image. A thumbnail preview is available to ease the browsing.

State 4: The image is zoomed in by this state. "L" gesture is to view the image in higher contrast and "Open" is to have the image back at normal contrast. In this state, the user is allowed to pan through the image using "Close" gesture.

When there is no gesture detected for 5 clock cycles continuously, the system will exit its current state. Since no action is taken during state exit, the image under viewing will remain displayed and this allows the user to rest his hand. Basically, there are three important components: State controller, Classifier and action execution, to make the system state machine flow to work. When the clock is activated, the loop as shown in Figure 14 keeps running until it is deactivated. The clock is the MATLAB timer object that runs a pre-programmed procedure or callbacks continuously. First, the State controller checks the current state and determines which kind of posture classification needed.

For example, the state controller selects the "Open" palm posture detection only during initialization state. Then, the classifier processes the image acquired from webcam and return the result of classification and position of the detection. The result of classifier can be either "Open" posture detected, or "Close" posture detected or so on, depending on the number of detection selected by State controller according to the state status. The action executor reacts accordingly to the classification result and position. For example, when "close" posture is detected at state 1,

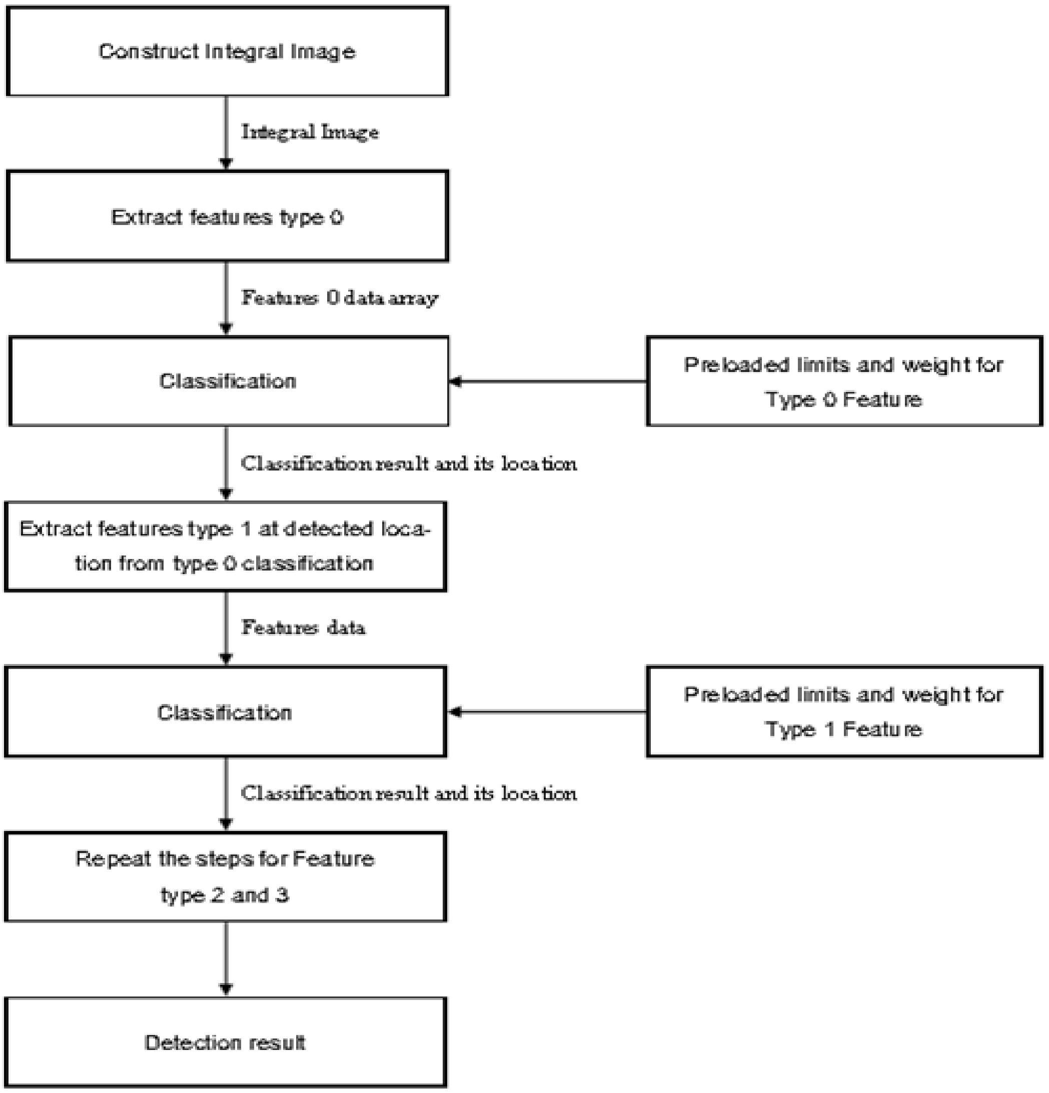
Construct Integral Image
Integral Image
Extract features type 0
Features 0 data array
Classification
Preloaded limits and weight for Type 0 Feature
Classification result and its location
Extract features type 1 at detected location from type 0 classification
Features data
Classification
Preloaded limits and weight for Type 1 Feature
Classification result and its location
Repeat the steps for Feature type 2 and 3
Detection result

Figure 11: Classification work flow that cascades each type of feature classification

the action executor displays the red region selection box based on the position of the detection. The action executor's task also include reading images from hard drive, updating the image viewer display as well as performing simple tasks such as zooming and contrast incrementing. After the action is executed, the State Controller updates the state accordingly. Then, the loop is continued over again if the system clock is not deactivated. Figure 14 shows the loop and components linkage in high level.

6. EXPERIMENT AND DISCUSSION

To determine the usability of the application, the completed system is tested for robustness and responsiveness using Intel Core2 Duo 3GHz CPU with a 320×240 pixels webcam. In first experiment three different users attempt to perform all six commands with 50 trials each. This experiment's

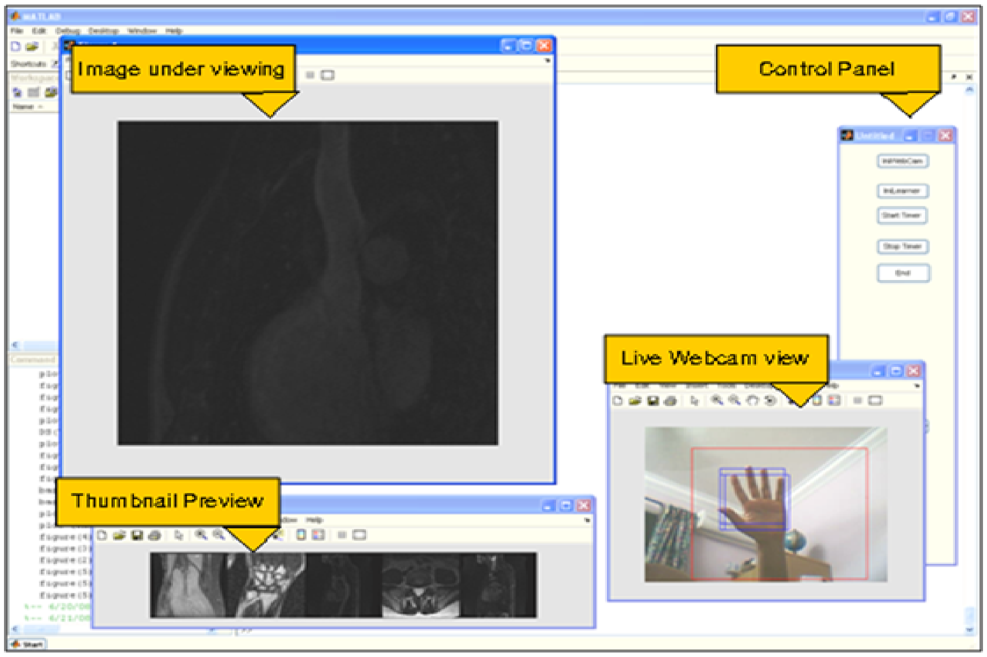

Figure 12: Layout of the Image Browsing application

objective is to examine the system's robustness towards individual hand variation. The experiment is conducted under fluorescent lighting with a plain background. The users 1 and 2 have their hand image samples in the training set while user 3 is a totally new user. Table 3 shows the result of the experiments. The average successful percentage is the average of all the actions combined.

Table 3: The result of experiment for user variation evaluation

Action	User 1	User 2	User 3
Next Picture	50	49	49
Previous Picture	49	49	47
Region selection	44	43	41
Zoom	48	46	44
Pan	44	43	41
Increase contrast	47	46	44
Average Successful %	94%	92%	88.67%

6.1. Background Robustness

The second experiment is to determine the performance of the recognition system at complex background versus plain background. The blue boxes are the location where the hand is detected. Similar with the previous experiments, the user tries to perform all six commands and the number of successful attempts are recorded. Figure 15 shows the complex and plain background used for

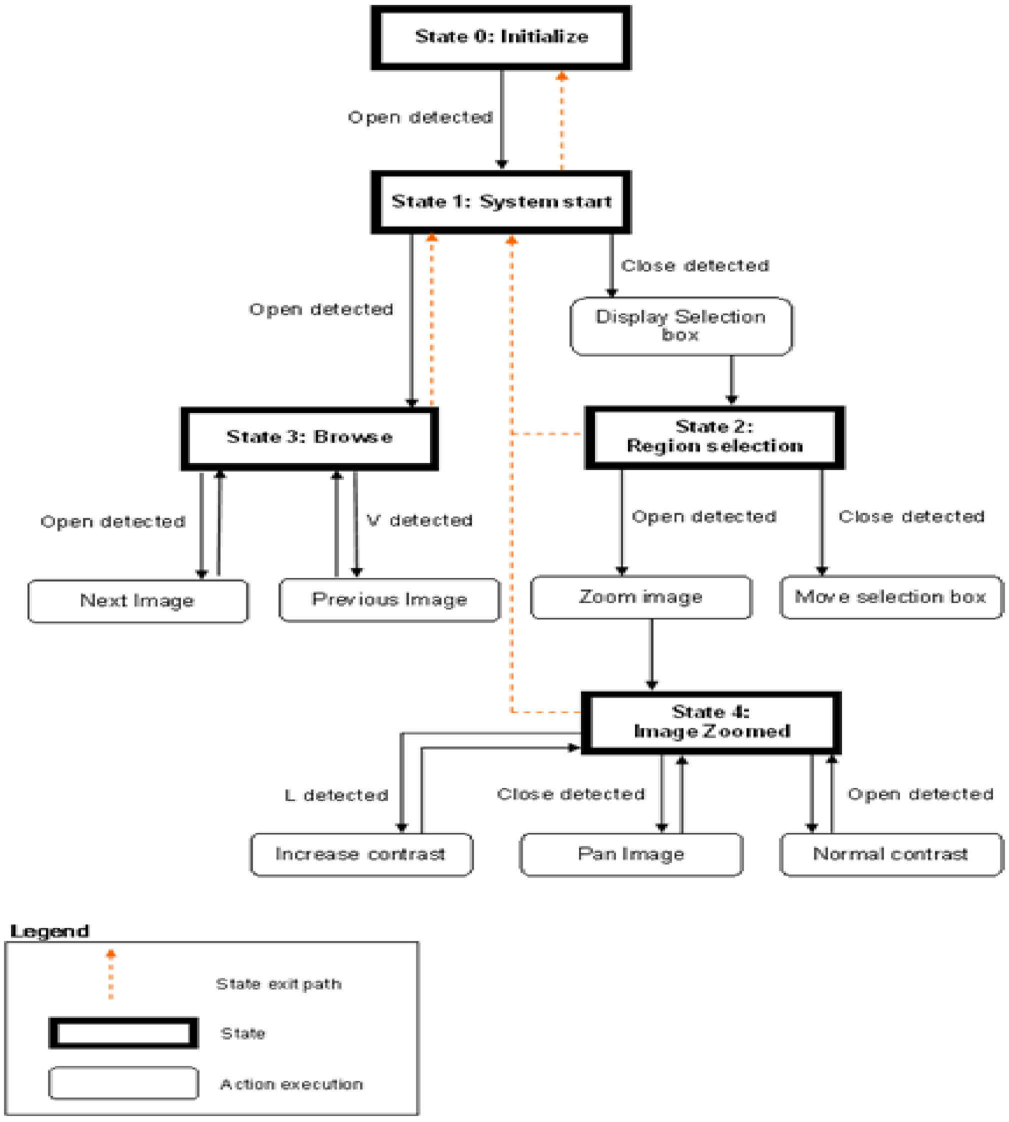

Figure 13: The state machine diagram

experiment. The complex background contains various objects, while the plain background is just a simple ceiling. The results of this experiment are tabulated in Table 4.

6.2. Lighting Influence

The third experiment is to determine the lighting influence on detection rate of each posture under plain background. The three lighting condition are natural lighting, indoor fluoresces lighting and mixed lighting of these two. The Indoor fluorescent lighting is conducted at night with no other sources of light from the outdoor. The natural lighting experiment is conducted under the conditions whereby the indoor lighting is turned off, allowing sunlight from outside to enter. The mixed lighting is the same except that the indoor lighting is turned on. Each posture is tested 40 times under the three conditions and the result is tabulated in Table 5.

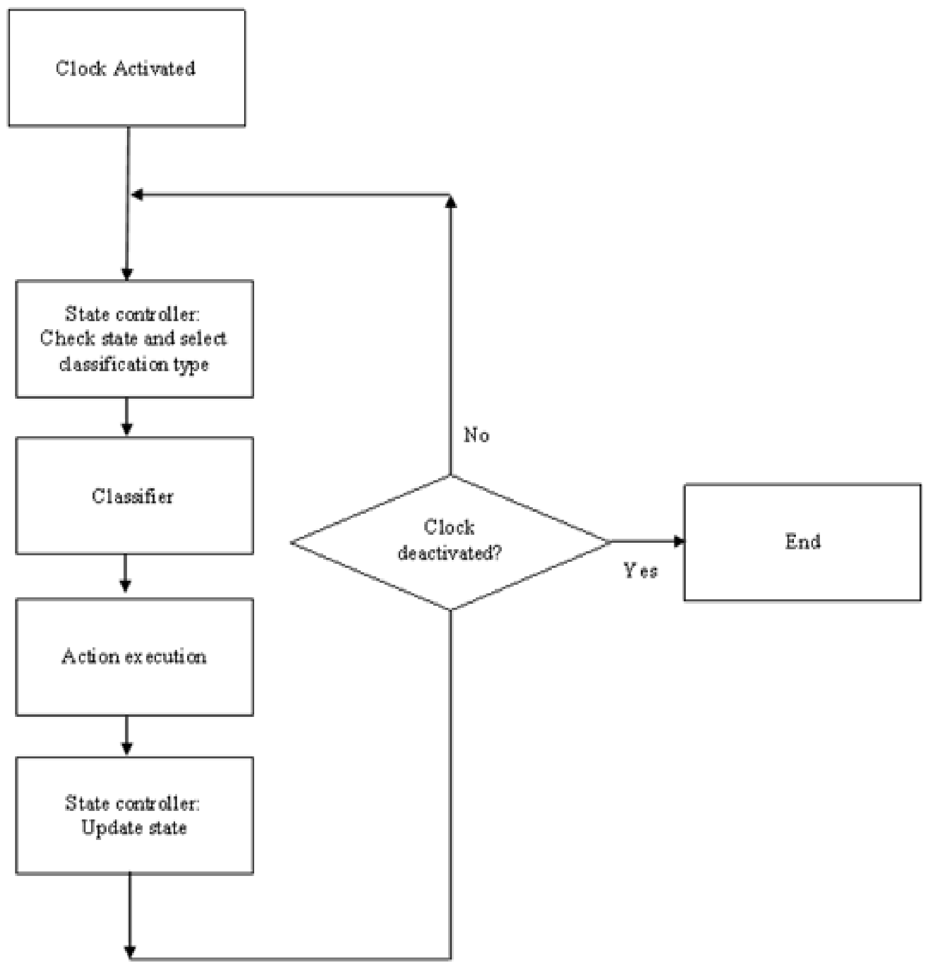
Clock Activated
State controller: Check state and select classification type
Classifier
Action execution
State controller: Update state
No
Clock deactivated?
Yes
End

Figure 14: The high level work flow chart that illustrates the linkage of three main components realizes the state machine

6.3. Hand Orientation

To examine the robustness towards tilted hand images, the detector is tested under three situations where the user hands are intentionally tilted. Each posture is tested 40 times and the results are recorded in Table 6. In order to evaluate the responsiveness of the system, the processing time is recorded using TIC TOC function in MATLAB. Since the number of posture classification varies in each state, the processing time requirement is different as well. So, the average of each state is calculated to represent the speed of processing in each clock cycle as tabulated in Table 7.

7. RESULTS AND DISCUSSION

The user variation experiment shows that users without training samples has lower detection rate compared to the other two users having their hand samples in the training data. The detection rate at cluttered background suffered 9% drops compared to plain background. Compared to

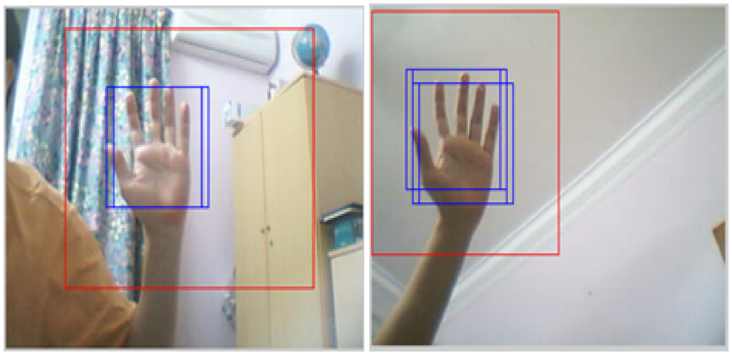

Figure 15: Complex and plain background

Table 4: Result of background experiment

Action	Complex Background	Plain Background
Next Picture	47	50
Previous Picture	41	49
Region selection	38	44
Zoom	43	48
Pan	35	44
Increase contrast	44	47
Average Successful %	82.67%	94%

Table 5: Result of lighting experiment

Posture	Natural lighting	Indoor fluoresces	Mixed lighting
Open	90%	95%	90.00%
Close	90%	92.50%	92.50%
L	95%	92.50%	95.00%
V	95%	95%	95.00%
Average	93%	94%	93.13%

similar studies by Juan Wachs [5] which has an average of 97% of successful action execution, this application has 5% lower successful rate due to the dynamic gesture like region selection and image panning as the main contributor. However, the Juan Wachs's application is more successful as the users are required to wear long sleeve and there is limited segmented webcam viewing area, whereby users need to place the hand in dedicated areas for some action. The recent work done by Qing Chen in 2008 [8] using similar methods has published an average 95.75% detection rate for four

Table 6: Result of Angle Robustness Experiment

Posture	Straight	Tilted	Seriously tilted
Open	95%	27.50%	0%
Close	92.50%	22.50%	5%
L	92.50%	15%	0%
V	95%	20%	0%
Average	94%	21%	1%

Table 7: Processing time of each state

Action	Number of posture	Processing time (seconds)
State 0	1	0.220
State 1	2	0.188
State 2	2	0.188
State 3	2	0.188
State 4	3	0.344
Average		0.2254

static postures recognition based on an experiment with 100 sample hand images for each posture. The classifier in the study for this paper has a similar detection rate at 94% if only the static posture is counted. So, the detection rate of this application is considered acceptable as it is above 90% under optimal situation, while having 88% for new users and 82% under complex background.

In the experiment for hand orientation evaluation, obviously, the hand orientation experiment shows that the hand detector is not robust to cater for tilted posture where the detection rate is 0% when the hand posture is "seriously tilted". This is due to the nature of the appearance-based recognition technique as the system is trying to recognize the object that "looks similar" to the training sample. Similar problem is also observed in the work by Juan Wachs [5] where hands at different angle are artificially generated during training but it cannot address the problem well enough, yielding an average of 48% detection rate. For the lighting influence experiment, there is no significant difference observed from the result. This may due to several reasons as the samples under these conditions have been included in training and the webcam camera driver has auto white balance correction.

The processing time for each cycle clocked at 0.23 seconds in average, which is equivalent to 5 frames per second including the time to update the live webcam view window. The time consumed by State 0 is higher even though for only one posture classification because the system needs to localize the hand in this state. At this speed, this application is considered running in real time where any lagging is hardly observed even when there is other software like antivirus running in the background.

8. CONCLUSION

The system is considered functional in real time as targeted as its detection rate and responsiveness are satisfactory. The accuracy of Viola-Jones method for static postures detection yields over 90% of detection rate with lower than 5% false positive rates. In the system level command input, the successful actions by users are up to 94% under optimal condition. The skin color segmentation helps to narrow down the search window by reducing 19% of the processing time. Overall processing time per frame is 0.22 seconds in average. However, when the number of posture classification increases, the processing time increases as well. So, there is a bottleneck where the application will slow down significantly if the number of postures for classification is increased. Under such situations where numerous postures are needed, the possible solution is to implement the application as a fully compiled Windows Application through C coding, without going through the MATLAB platform.

ACKNOWLEDGMENT

The authors acknowledge the research grant provided by Universiti Sains Malaysia, Penang that has resulted in this article.

REFERENCES

1. Viola, P. and Jones, M. 2001. Rapid object detection using a boosted cascade of simple features. *IEEE Conf. on Computer Vision and Pattern Recognition*, December, Vol 1, 511-518.
2. Klsch, M., Turk, M., Hollerer, T. and Chainey, J. 2004. Vision based hand gesture interfaces for wearable computing and virtual environments. University Of California. Available at `http://www.movesinstitute.org/~kolsch/HandVu/HandVu.html`
3. Kolsch, M. and Turk, M. 2004. Analysis of rotational robustness of hand detection with a Viola-Jones Detector. *IAPR International Conference on Pattern Recognition.*
4. Wachs, J., Stern, H. and Edan, Y. 2005. A real-time hand gesture system based on evolutionary search. *Genetic and Evolutionary Computation Conference, GECCO.*
5. Robert, Y.F. and Schapire, E. 1999. A short introduction to Boosting. *Journal of Japanese Society for Artificial Intelligence*, 14(5):771-780.
6. Chen, Q., Georganas, N.D. and Petriu, E.M. 2007. Real-time Vision-based Hand Gesture Recognition Using Haar-like Features. *Instrumentation and Measurement Technology Conference - IMTC 2007*, Warsaw, Poland, May 1-3.
7. Chen, Q., Georganas, N.D. and Petriu, E.M. 2008. Hand Gesture Recognition Using Haar-Like Features and a Stochastic Context-Free Grammar. *IEEE Trans. on Instrumentation and Measurement*, 57(8).
8. Schultz, M., Gill, J., Zubairi, S., Huber, R. and Gordin, F. 2003. Bacterial contamination of computer keyboards in a teaching hospital. *Infect Control Hosp Epidemiol*, 24:302-3.
9. Moeslund, T.B. and Norgard, L. 2003. A Brief of Hand Gestures used in wearable human computer interface, Computer Vision and Media Technology Lab., Aalborg, `http://vision.auc.dk`
10. Hongo, H., Ohya, M., Yasumoto, M., Niwa, Y., and Yamamoto, K. 2000. Focus of attention for face and hand gesture recognition using multiple cameras. *Proc. of 4th IEEE Int. Conf. on Automatic Face and Gesture Recognition*, 28-30 March, pp. 156-161.
11. Chen, Y-T. and Tseng K-T. 2007 Multiple-angle Hand Gesture Recognition by Fusing SVM Classifiers. *IEEE Int. Conf. on Automation Science and Engineering, CASE 2007*, 22-25 Sept., pp. 527-530.
12. Gupta, L. and Ma, S. 2001. Gesture-based interaction and communication: automated

classification of hand gesture contours. *IEEE Trans on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 31(1):114-120.

13. Zhu, Y., Ren, H, Xu, G. and Lin, X. 2000. Toward real-time human-computer interaction with continuous dynamic hand gestures. *Proc. of 4th IEEE Int. Conf. on Automatic Face and Gesture Recognition*, 28-30 March, pp. 544-549.
14. Caglar, M.B. and Lobo, N.V. 2006. Self-correcting tracking for articulated objects. *Proc. of the 7th International Conference on Automatic Face and Gesture Recognition (FGR'06)*, pp. 609-616.