Human Computer Interaction Using Face and Gesture Recognition

Yo-Jen Tu, Chung-Chieh Kao, Huei-Yung Lin
Department of Electrical Engineering, National Chung Cheng University, Chia-Yi 621, Taiwan, R.O.C.

***Abstract*— In this paper, we present a face and gesture recognition based human-computer interaction (HCI) system using a single video camera. Different from the conventional communication methods between users and machines, we combine head pose and hand gesture to control the equipment. We can identify the position of the eyes and mouth, and use the facial center to estimate the pose of the head. Two new methods are presented in this paper: automatic gesture area segmentation and orientation normalization of the hand gesture. It is not mandatory for the user to keep gestures in upright position, the system segments and normalizes the gestures automatically. The experiment shows this method is very accurate with gesture recognition rate of 93.6%. The user can control multiple devices, including robots simultaneously through a wireless network.**

I. INTRODUCTION

In recent years, the field of computer vision progressed rapidly and the efforts have been made to apply research results in the real-world scenarios. When implementing research findings, hardware cost becomes an important issue. In this paper, we use only a video camera and a PC to develop a face and gesture based human-computer interaction (HCI) system.

We target this HCI system implementation towards robot tour guidance, recreational, home and health-care applications. In museums, we hope to replace the traditional keyboard and mouse setup with our system – a robot tour guidance system. The robot will detect which exhibitions the visitors are interested in and introduce them on-the-fly. This will not only make exhibitions more interesting, but also reduce the tour guidance personnel training costs for the museums. For recreational usage, users can substitute wired controllers with hand gestures, enjoy the hands-free control of electronic devices. In household uses, we can combine head movement with simple hand gestures to control air-conditioners, lighting, and other home appliances. It may also be used to aid patients in all kinds of situations when their body mobility is limited.

The proposed HCI system not only can detect facial features in head-tilted situations, but also can recognize hand gestures correctly anywhere in the whole image. It is also robust to busy backgrounds and different clothing situations, extracting hand regions and detecting hand gestures efficiently using a trained neural network system. Using the system setup in the experiments made in this paper, we then implement the HCI system in two different real-life scenarios. First, control two camera's shutters to take snapshots with pre-defined hand gestures. Second, give commands wirelessly to trigger the head movement of the robot. Fig. 1 shows the system diagram of the proposed HCI system.

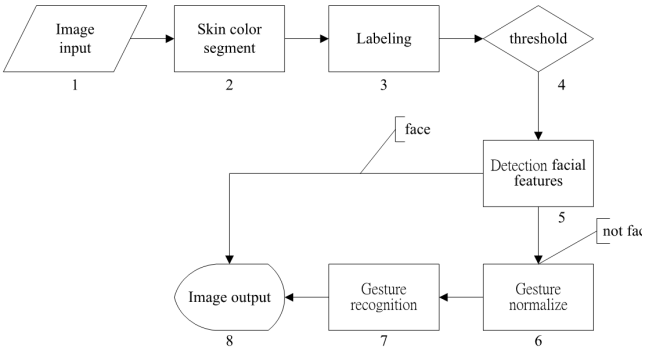

Fig. 1: System diagram of the proposed system

II. RELATED WORK

We focus on face detection, head position estimation and hand gesture recognition. In face detection related research, Rowlet et al. [1] presented an efficient neural network-based face detection system. Unlike some works which are limited to detecting upright, frontal faces, this system is able to detect the faces at any degree of rotation in the image plane. The system employs multiple networks; the first is a "router" network, which processes each input window to determine its orientation. This information is then used to prepare the window for one or more "detector" networks. Lee et al. [2] proposed a new facial feature detection approach based on the local image region and direct pixel-intensity distributions. The above two researches only allow one degree of freedom, which is not suitable for the estimation of head tilt positions.

In the head angle estimation related research, Chen et al. [3] presented a robust approach to estimate the 3D pose of human heads using a single image. Their method only makes use of the information about the skin and hair region of the heads. Yamada et al. [4] proposed a real-time head pose estimation system using a new image matching technique. The system consists of a training stage, in which the subspace dictionaries for classifying the head poses are computed using template matching and the factorization method, followed by a recognition stage, in which the head poses are estimated using the subspace method.

In the hand gesture recognition related research, Wachs et al. [5] suggested a methodology using a neighborhood-search algorithm for tuning the system parameters. They addressed the problem of simultaneous calibration of the parameters of the processing/fuzzy C-means (FCM) components of a hand-gesture recognition system. This system is limited if it is implemented as a part of HCI systems because it is not capable of detecting the hand gesture locations in the image

automatically– the user must restrict the hand gestures in a certain area. Kim et al. [6] analyzed the hand gestures with four processes: detecting the hand in bimanual movements, splitting of a meaningful gesture region from an image stream, extracting features and recognizing the gesture. The user needs to wear long sleeve clothing, exposing only palms and hands in order to allow the hand gesture recognition to function correctly and properly.

III. Proposed Recognition System

In this paper, we develop a human-computer interaction (HCI) system using a single video camera to capture images. For facial recognition and detection, first of all we label the areas of an image using skin colors, which act as candidates for the face and hand. Second, connected components are discovered from these image areas. Third, we set a threshold for the connected components to filter out noise, eliminating the areas which are too small to be candidates for the face or hand. For the remaining succeeded candidates, we declare search areas for the eyes and mouth. We search for the eyes using the black and white color feature characteristic. We discover the mouth using the distinct redder color tone of the lips compared to facial skin. Finally, after retrieving the eyes and mouth, we use a simple isosceles triangle geometric shape to find a best match, and output the resulting triangle model thus finishing detection of the face.

For hand gestures, we eliminate the arm and elbow first, then search for the long-axis of the hand, normalize the hand gesture to a certain fixed angle prepping it for gesture recognition. We input the normalized hand result to a trained and weighted neural network, thus showing the recognition result and accuracy. Last of all, we combine the results of both facial and hand recognition, and show them on the screen for us to verify the correctness of the detection and recognition results.

IV. Face Detection

Details on face detection implementation are discussed in this section. The first step is skin color detection, which is an important preprocessing stage for the next step. Second, we define a facial search area and range, this helps reduce search time and error rate greatly, this is critical since our goal is to implement the algorithm in a real-time system. Finally we detect the eyes and mouth, also estimate head tilted position simultaneously.

A. Skin Color Detection

Face detection is a popular research topic with many applications. Various techniques using different color spaces were developed previously based on the skin color with the color constancy constraint. We analyzed a few skin color detection methods, grouped them into categories based on which color space is used to accomplish skin color detection and selected the most suitable one to implement in our HCI system. Color spaces such as RGB, YC_bC_r, hue-saturation-value (HSV), and NCC r-g color spaces each have their pros and cons.

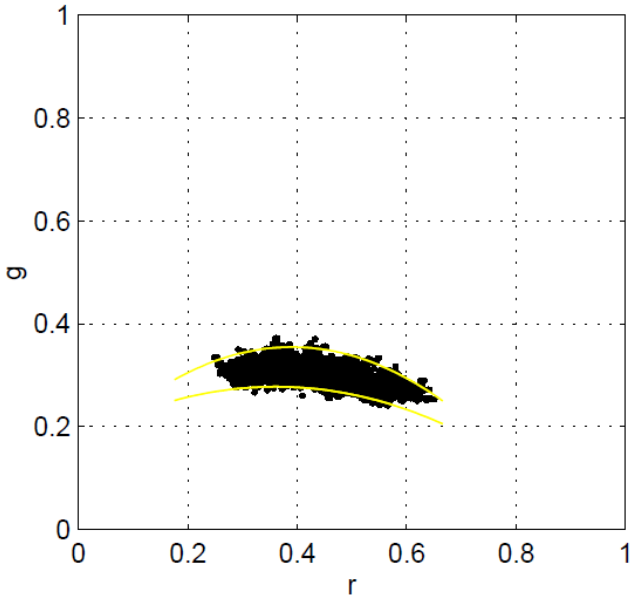

Fig. 2 Skin color locus in NCC color space

The RGB color space requires short calculation time because images captured from the video camera can be used directly without any transformation steps. Singh et al. [8] used RGB color space as a part of the proposed skin color detection algorithm. The output using only RGB color channel fails when there are some more skin regions like legs, arms, etc.

Because brightness intensities exist in each of the three color channels, skin color detection accuracy is easily affected by the surrounding lighting condition. YC_bC_r and HSV color spaces separate light intensity from color, which solves one of the main issue when using RGB color space. Hsu et al. [9] adopted YC_bC_r color space for face detection since it is widely used in video compression standards. The HSV model is used mainly in computer graphics and is considered by many to be more intuitive to use, closer to how an artist actually mixes colors. Garcia et al. [13] applied HSV and YC_bC_r color space models for skin color clustering and filtering.

The normalized color coordinates NCC r-g color space combines the advantages of the above color spaces, which reduces the illumination brightness dependence.

The NCC r-g color space also maintains a low overhead for the color space transformation algorithm. Soriano et al. [7] proposed a simple membership function for the skin locus, and used a pair of quadratic functions to define the upper (1) and lower (2) bound of the cluster. Fig. 2 shows the r-g skin histogram for multiple skin types of the face images captured from a web camera. The x-axis represents r = (R/(R+G+B)), and the y-axis represents g = (G/(R+G+B)). R, G and B represent the original video or image's RED, GREEN, and BLUE color channels in the RGB color space.

$$F_1(r) = -1.376r^2 + 1.0743r + 0.1452 \quad (1)$$

$$F_2(r) = -0.776r^2 + 0.5601r + 0.1766 \quad (2)$$

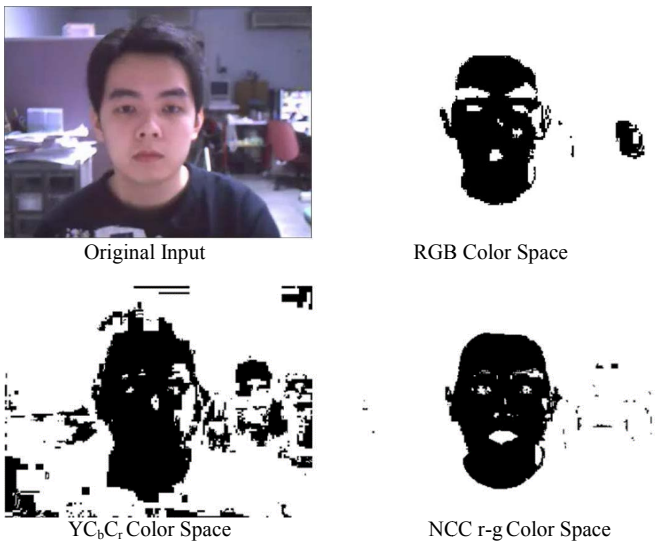

Fig.3: Skin color extraction results comparison

Note however that the white point (r=g=0.33) is within the locus model. We avoid labeling whitish pixels as skin with the rule:

$$w = (r - 0.33)^2 + (g - 0.33)^2 > 0.0004 \qquad (3)$$

Combining the above bounds and rules, the pixels with chromaticity (r,g) are then labeled as skin candidates using the skin color locust constraint Skin(r, g) as follows:

$$\text{Skin} = \begin{cases} 1 & \text{if } F_2(r) < g < F_2(r), w > 0.0004 \\ 0 & \text{otherwise} \end{cases} \qquad (4)$$

In our research we tried a few skin color locus models in different color spaces. In Fig. 3 we show the results of our test. The upper-left image is the original input image. The upper-right image is the output using the RGB color space locus proposed by Singh et al. [8]. Hsu et al. [9] proposed the skin color locus in YC_bC_r as shown in the lower-left image. Finally, the lower-right image uses an NCC r-g color space [7].

Apparently we can see the NCC r-g color space gives the best result. This combined skin locus and color space keeps the calculation costs low, also copes well with the skin color change due to varying lighting conditions. After this analysis we decide to use the NCC r-g color space skin locus model in our HCI system.

B. Face Detection

We introduce the specific steps towards the face detection. We first define the search range limit, and then search for the eyes and mouth. Finally we match the facial features from the search results.

When implementing a robust real-time face detection system, reducing computational costs is a critical issue. In normal situations, a person's facial features will locate in fixed relative positions. Thus, we can use this characteristic to define the search windows and search range when searching for them.

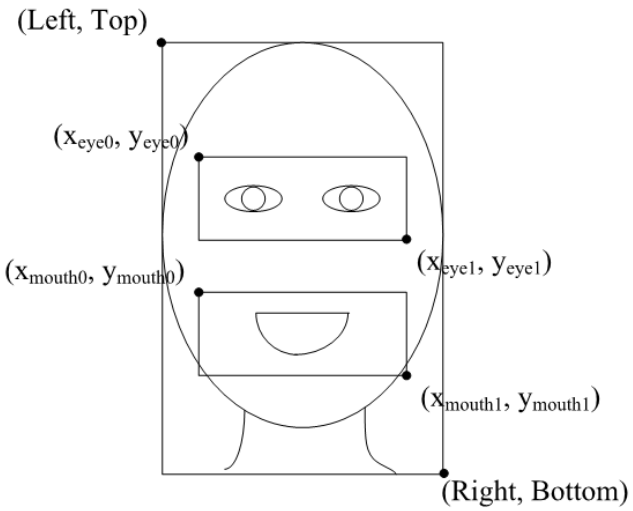

Fig.4: Search windows for eyes and mouth

Fig. 4 shows the diagram of the search windows for the eyes and mouth.

The search window for the eyes is defined by (5) (6):

$$\begin{aligned} x_{eye0} &= \text{Left} + \frac{\text{Right} - Left}{10} \\ y_{eye0} &= \text{Top} + \frac{\text{Bottom} - Top}{5} \end{aligned} \qquad (5)$$

$$\begin{aligned} x_{eye1} &= \text{Right} - \frac{\text{Right} - Left}{10} \\ y_{eye1} &= \text{Top} + \frac{\text{Bottom} - Top}{2} \end{aligned} \qquad (6)$$

The search window for the mouth defined by (7) (8):

$$\begin{aligned} x_{mouth0} &= \text{Left} + \frac{\text{Right} - Left}{5} \\ y_{mouth0} &= \text{Top} + \frac{\text{Bottom} - Top}{2} \end{aligned} \qquad (7)$$

$$\begin{aligned} x_{mouth1} &= \text{Right} - \frac{\text{Right} - Left}{5} \\ y_{mouth1} &= \text{Top} + \frac{3(\text{Bottom} - Top)}{4} \end{aligned} \qquad (8)$$

The eyes tend to have a darker tone and have a more distinct characteristic than other facial features. If we express this characteristic in color space matters, the three channels in the RGB color space tend to have similar intensity values. We can distinct the eyes with other facial features using this observation. To extract eyes from other areas of the face, (9) expresses this feature:

$$|R - G| + |G - B| + |B - R| < T_{eye} \qquad (9)$$

The RGB channel intensities have values ranging from 0~255. After performing multiple evaluations, we conclude that setting T_{eye} to 100 is a good threshold for extracting the eyes. We can locate the eyes in an image after using equation (9) as an evaluation function. The eyebrows sometimes will be extracted from the image, so we require some kind of mechanism to solve this problem.

Fig.5: Extraction results for the eyes and mouth

We propose a new method to locate the mouth. Using the previous analyzed skin color information, we dynamically adjust the mouth-extraction threshold values. The specific steps are as follows:

1. From the previous skin color detected pixels of the image, we calculate the average of R and G color channel intensities. This is done in (10). (n indicates skin color total pixel count in a certain webcam frame)

$$R_{avg} = \frac{1}{n}\sum_{i=1}^{n} R_i \quad , \quad G_{avg} = \frac{1}{n}\sum_{i=1}^{n} G_i \qquad (10)$$

2. People's mouth color tend to have a redder tone compared to the rest of the skin, meaning the R/G channel intensity ratio is higher than all other detected skin pixels. We express this in (11).

$$1.2 \times \frac{R_{avg}}{G_{avg}} < \frac{R_{mouth}}{G_{mouth}} < 1.5 \times \frac{R_{avg}}{G_{avg}} \qquad (11)$$

Fig. 5 shows the facial extraction results. Although some unwanted small areas are extracted using the above method, we can locate the exact position of the mouth by retrieving the largest extracted redder tone skin area and calculating the center of gravity of that specific area.

When we extract the eyes from other facial features, sometimes eyebrows are extracted accidently at the same time. We define three sets of rules to avoid this problem. Using accurate mouth position data acquired from the previous step, information in two of these rules, we use the rules below to find the best match for the eyes while eliminating the eyebrows.

1. Width between eyes are limited by the width of the face

$$\frac{width}{4} < D_{eye} < \frac{3width}{4} \qquad (12)$$

$$D_{eye} = \sqrt{x_{eyeR} - x_{eyeL}^2 + y_{eyeR} - y_{eyeL}^2} \qquad (13)$$

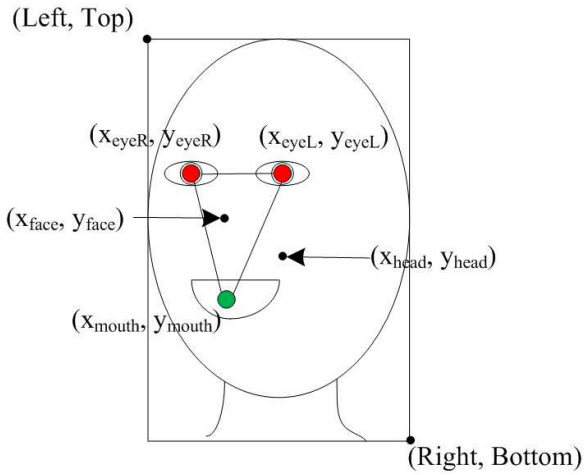

Fig.6: Diagram showing centroids of head and face

2. Distance between the eyes and mouth are also limited by the width of the face

$$\frac{width}{3} < D_{mouth} < \frac{3width}{4} \qquad (14)$$

$$D_{mouth} = \sqrt{(x_{mouth} - \frac{x_{eyeR}+x_{eyeL}}{2})^2 + (y_{mouth} - \frac{y_{eyeR}+y_{eyeL}}{2})^2} \qquad (15)$$

3. The mouth must lie between the eyes

$$x_{eyeR} < x_{mouth} < x_{eyeL} \qquad (16)$$

Using the above rules, we eliminate eyebrows and other unwanted areas, retrieving the correct eye positions.

In our system, the goal is to control the hardware by mixing the eye movement and hand gestures. We now discuss how to recover orientation (pose) of a moving head. Ohayon et al. [11] proposed a method which required the construction of a head model using 3D points. Although the accuracy is very high, it can only process 4 frames per second due high computational costs. Instead, we propose a low complexity method suitable for real-time systems. First we calculate the centroid of the triangle connecting eyes and the mouth. We then compare it with the detected head's centroid. By doing so, we can estimate the tilt position of the head. In Fig. 6, (x_{head}, y_{head}) is the centroid of the detected head, and (x_{face}, y_{face}) is the centroid of the triangle formed by connecting the eyes and mouth.

$$x_{face} = \frac{x_{eyeR} + x_{eyeL} + x_{mouth}}{3} \qquad (17)$$

$$y_{face} = \frac{y_{eyeR} + y_{eyeL} + y_{mouth}}{3}$$

We calculate the projection distances to the x and y-axis of points (x_{head}, y_{head}) and (x_{face}, y_{face})

$$D_x = x_{head} - x_{face}$$
$$D_y = y_{head} - y_{face} \qquad (18)$$

Using the projection distances from (18), we can estimate the tilt position of a person's head. For example, if D_x is a positive value, it means the head tilts to the right, and the head tilts to the left if the value is negative. D_y indicates if the person tilts his/her head upward or downward.

V. Hand Gesture Recognition

For hand gesture recognition, we apply a particular type of neural network model, which is known as a "feed-forward back-propagation neural network" [12]. This neural network is applied for hand gesture recognition in our system. This neural model is easy to understand, and can be easily implemented in image processing tasks.

With traditional techniques, one must understand the inputs of the algorithms, and the outputs for correct implementation. For a neural network solution, you do not have to know these details at all. In neural network systems, you simply show the relation of the output associated with the given input. With an adequate amount of training, the network will mimic the function that you are demonstrating. With a neural network, it is possible to apply some inputs irrelevant to the solution. During the training process, the network will learn to ignore any inputs that do not contribute to the output. If some critical inputs are left out in the training process, the network will fail to result in a correct solution.

A. Back-propagation Neural Network

A back-propagation neural network is a kind of feed-forward network. In a 3-layer feed-forward network (Fig. 7), the information moves in only one direction, forward, from the input layer (X_1, X_2, X_3, X_4 ...), through the hidden layer (H_1, H_2, H_3, H_4 ...), and to the output layer (Y_1, Y_2, Y_3 ...).

Below are specific settings we used in our neural network.

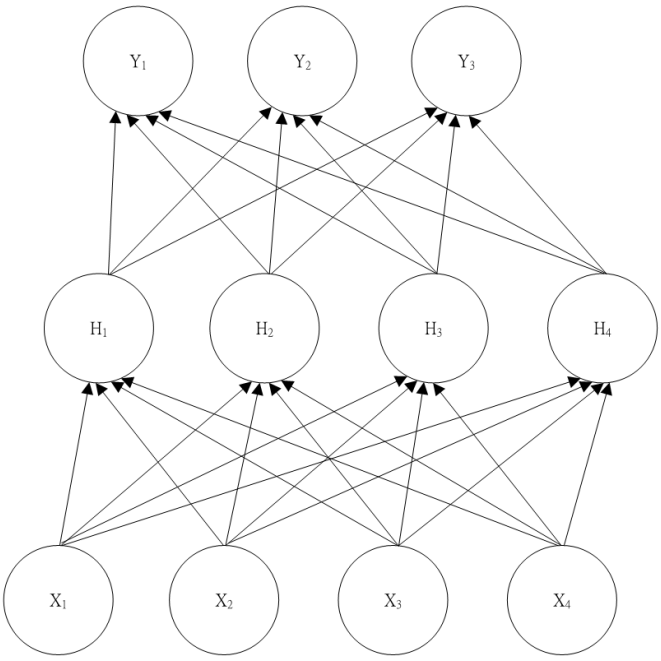

Fig.7: Diagram of a back propagation back-propagation neural network

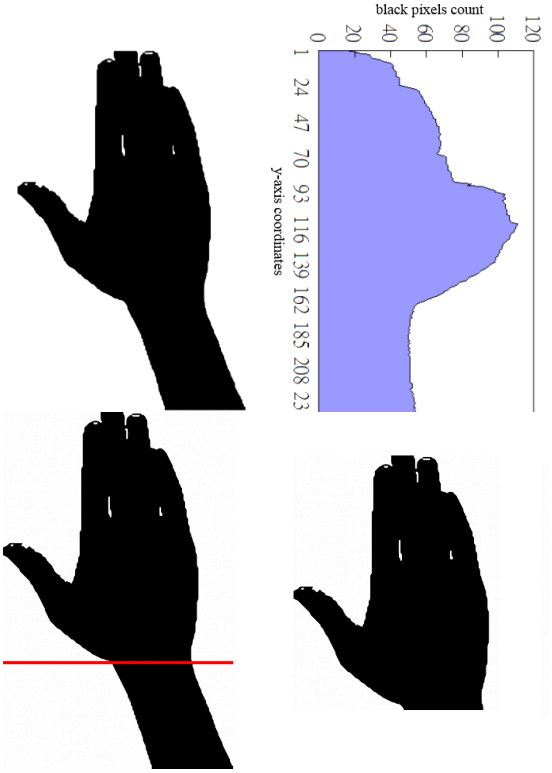

Fig.8: Exclude arm for higher hand gesture recognition accuracy

1. Network layer: We use the basic 3-layer neural network model for our real-time system. Although increasing layers in a neural network system will have a higher accuracy rate, it will also increase the complexity of the network. Adding layers will increase the learning and recalling time in the training process, which is not suitable for real-time systems. To increase accuracy, we add more neurons in the hidden layer instead.
2. Hidden layer neurons: A common approach to decide how many neurons are located in this layer is to double the number of neurons in the input layer. We give 16 neurons in the input layer and 30 neurons in the hidden layer.
3. Learning rate: Since learning is usually done in post-process, it will not affect the real-time performance when used. We choose a learning rate of 0.01 to maintain the system's stability.

B. Hand Gesture Segmentation and Recognition

We use the skin color to locate the hand's position. If a person wears a long sleeve shirt, the captured image area will be the hand gesture candidate. But if one wears short sleeve clothing, we would need to exclude the whole arm and preserve the hand section only.

From observations, we find the width constant from the elbow to the wrist. Using this characteristic, we can eliminate the arm and capture only parts of the hand needed for hand gesture recognition. In Fig. 8, we first convert the hand image into a binary image, and then project the image onto the y-axis (x-axis indicates black pixel count of the same y-axis value).

We can clearly see there is a large difference in terms of the change of the black pixel count from the wrist to the hand. We then normalize the hand into the upright position, and prep it for hand gesture recognition training the neural network system to output the same result for both images.

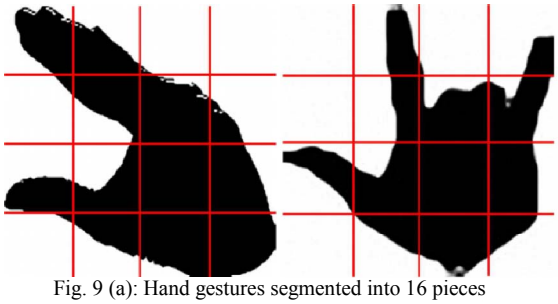

Fig. 9 (a): Hand gestures segmented into 16 pieces

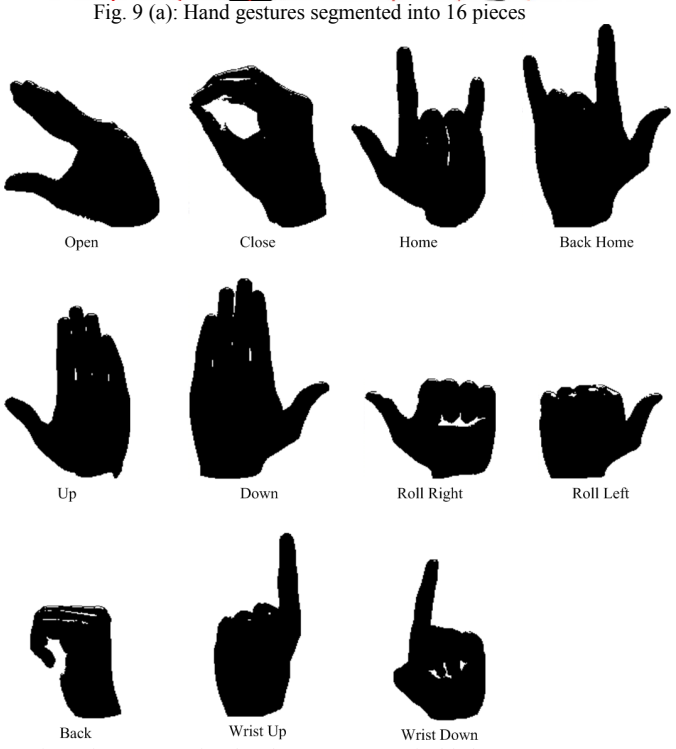

Fig. 9 (b): Names of 11 hand gestures recognizable by our system

Fig. 9 (c): Image examples used in the neural network training process

After the above steps, we are ready for hand gesture recognition. Wagne et al. [14] proposed gesture recognition interface is referenced for this step - segmenting the hand image into 16 pieces (Fig. 9(a)), each having a 10x10, 100 pixels. We record how many black pixels are in each piece, and normalize the pixel count from 0~100 to 0~1, providing the inputs for the neural network. In the hand gesture normalization process, some gestures will show different results after this step. For example, a "Roll Right" gesture could be normalized to a thumb down or a thumb up position. This can be solved in the neural network training process, allowing the output of both these images to have the same output results.

We define and name 11 hand gestures (Fig. 9(b)), and give the input of each hand gesture with 10 images for the training process of the neural network. Fig. 9(c) shows some examples of the images used to train the neural network. For consistency, all images used for training are first segmented, and normalized into the upright position beforehand.

VI. RESULTS

We split our results into two sections for discussion, one part showing the results of our system onscreen and the other part of experiments controlling the hardware with our system.

A. Facial detection and hand gesture recognition

Using Microsoft Visual C++ 6 and OpenCV library for reading and outputting image data, we find our setup capable of processing a 320 x 240 resolution image in 0.06 seconds, meaning the possibility of achieving the real-time calculation with 15 frames per second.

Fig. 10 shows the results of face detection. We capture the position of facial features (the eyes and mouth) in multiple head tilted situations. We circle and connect each of them, showing an inverted triangle on the screen. The bounding box marking the face area is achieved by using data retrieved from our facial feature detection step. The system is also capable of locating facial features even if the person is wearing glasses. We tested our algorithm using a 300 frame standard video clip "akiyo". We detected facial features correctly in 278 frames, showing an accuracy rate of 92.7%.

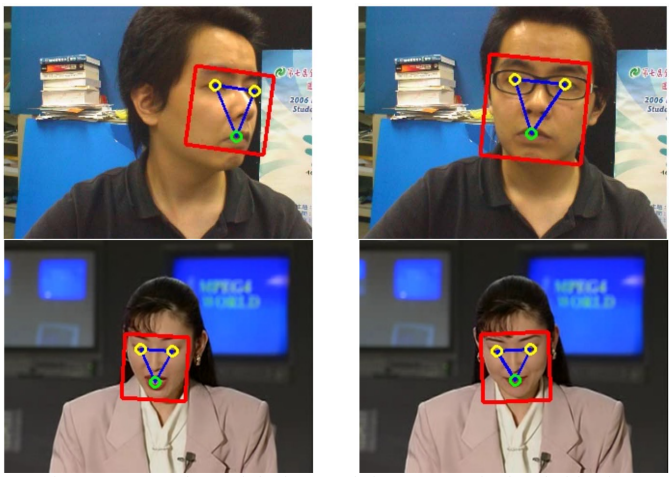

Fig.10: Test results straight from real-time camera feed and video inputs

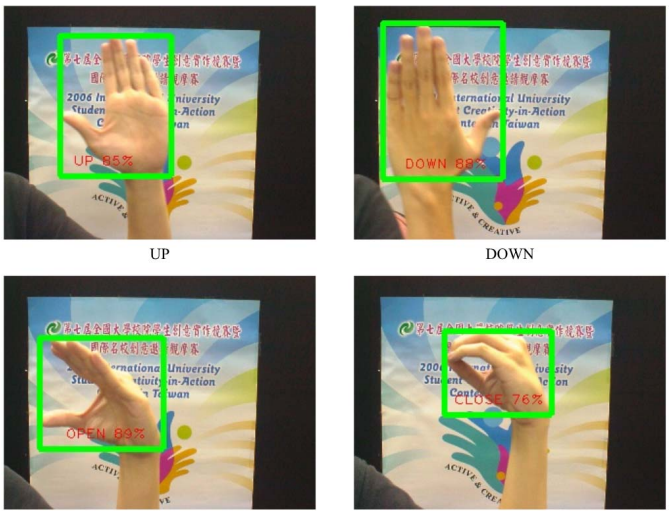

Fig.11: Hand gesture recognition results

B. *Hand Gesture Recognition*

We show some hand recognition result snapshots in Fig. 11. The name of the recognized hand gestures is labeled under the snapshots. Images can be processed in real-time, and it also shows the reliability estimation outputted by the recalling algorithm of the neural network. It is only displayed to evaluate results efficiently. Reliability estimation indicates how similar it is with the data fed to the neuron network at the training step. We place a colorful poster in the background, indicating our system is capable of extracting the hand region correctly even in busy background environments. In some special cases where the hand region weren't extracted correctly, the reliability rate will decrease greatly. However, due to the high tolerance ability of neural networks, our system is still capable of recognizing the hand gesture correctly. Summary of recognition results is shown in Fig. 12.

Gesture Name	Correct Count	Total Count	Accuracy (%)
OPEN	35	35	100
CLOSE	24	29	82.8
HOME	41	42	97.6
BACK HOME	38	38	100
UP	29	30	96.7
DOWN	49	49	100
ROLL RIGHT	23	28	82.1
ROLL LEFT	31	36	86.1
BACK	50	51	98
WRIST UP	37	40	92.5
WRIST DOWN	35	41	85.4
Total	392	419	93.6

Fig.12: Hand recognition result summary

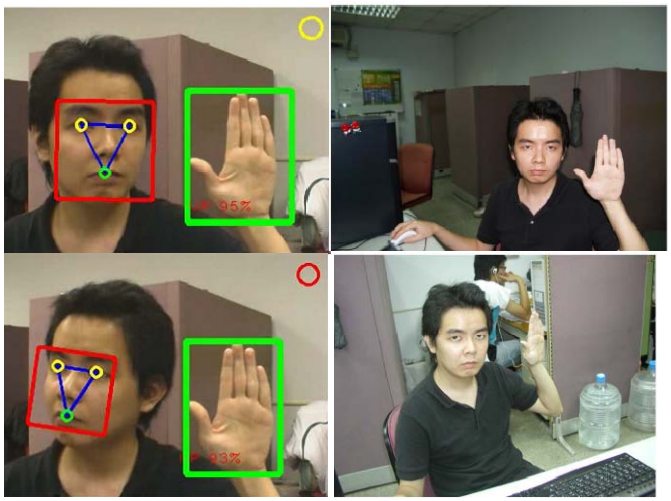

Fig.13: Camera shutter control results

Two images on the left are snapshots taken by video camera in the center, the other two images on the right shows photos taken by the left and right cameras controlled by head tilted position and hand gestures

C. *Camera Shutter Control Experiment*

We setup an experiment (Fig.13) to control the shutter of two cameras placed side by side using tilted positions of the head and hand gestures (one specific video camera was placed in the middle, used to detect all face and hand gestures). When the head is tilted to a certain position and the correct hand gesture is detected, the system will trigger the shutter of the camera the user is looking at (the API for the camera shutter control is provided by the manufacturers). Because there are only two cameras, we only reference projection distance D_x in (18). In our setup, we define the head tilted to the right with $-25 < D_x < -20$, and $20 < D_x < 25$ indicating the head is tilted to the left.

Using this system, a person can easily control many home appliances or any kind of electronic devices only with a few generic hand gestures.

D. *Robot Control Experiments*

We design a system where the head of a robot is controlled by hand gestures wirelessly (Fig. 14). Using the system setup

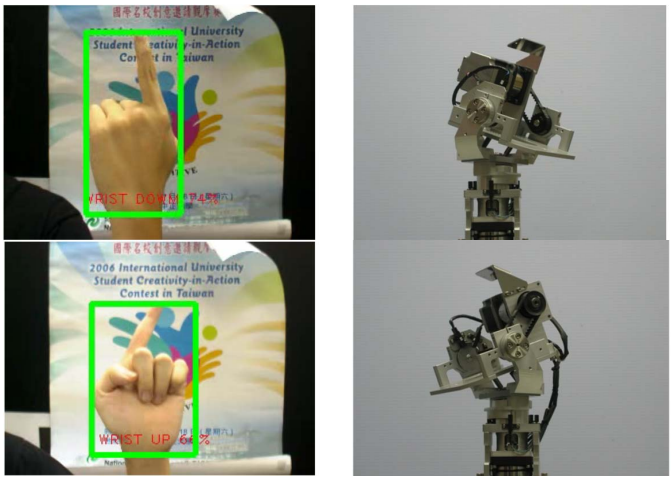

Fig.14: Controlling movement of robot's head wirelessly using hand gestures

in the previous experiments, we then add code to specifically trigger the head movement of the robot. The predefined hand gestures, UP, DOWN, WRIST UP, WRIST DOWN, control the robots head in four directions. The PC with the video camera connected will detect and recognize the hand gestures. Only when correct hand gestures are detected will the PC send signals to the robot wirelessly to make it turn its head.

VII. Conclusion And Future Works

We proposed a human-computer interaction (HCI) system using a PC and a video camera. We control the system using head and hand gestures. The contributions of this work include:

1. The face detection in real-time, and the use of centroids for the face and head to indicate head tilt directions.
2. The extraction of hand from an image without restrictions to clothing the person is wearing, and the recognition of the hand gesture using a trained neural network no matter if the hand is in the upright position or not.
3. Allowing users to control home appliances and hardware using simple hand gestures and face positions.

In future works and studies are listed as below:

1. Although using NCC r-g color space for skin color detection enables tolerance for different lighting situations, it is still vulnerable when surrounding lighting situations change too constantly. In future studies, we will come up with algorithms to overcome this issue.
2. The current proposed system is not capable to detect the face correctly when some facial features are covered up, in future works we wish to conquer this issue.

References

[1] Henry A. Rowley, Shumeet Baluja and Takeo Kanade, "Rotation Invariant Neural Network-Based Face Detection", *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 38-44, 1998

[2] Taigun Lee, Sung-Kee Park and Mignon Park, "Novel Pose-Variant Face Detection Method for Human-Robot Interaction Application", *IAPR Conference on Machine Vision Applications*, pp. 281-284, 2005

[3] Qian Chen, Haiyuan We, Takeshi Fukumoto and Masahiko Yachida, "3D Head Pose Estimation without Feature Tracking", *IEEE International Coference on Automatic Face and Gesture Recognition*, pp. 88-93, 1998

[4] Miki Yamada, Osamu Yamaguchi, Akiko Nakashima, Takeshi Mita and Kazuhiro Fukui, "Head Pose Estimation using Adaptively Scaled Template Matching", *IAPR Conference on Machine Vision Applications*, pp. 285-289, 2005

[5] Juan P. Wachs, Helman Stern, Yael Edan, "Cluster Labeling and Parameter Estimation for the Automated Setup of a Hand-Gesture Recognition System", *IEEE Transactions on Systems, Man and Cybernetics,* pp. 932-944, 2005

[6] Kye Kyung Kim, Keun Chang Kwak and Su Young Chi, "Gesture Analysis for Human-Robot Interaction", *ICACT Advanced Communication Technology,* pp. 1824-1827, 2006

[7] Maricor Soriano, Birgitta Martinkauppi, Sami Huovinen, and Mika Laaksonen, "Using the Skin Locus to Cope With Changing Illumination Conditions In Color-Based Face Tracking", *Proceedings of IEEE Nordic Signal Processing Symposium*, pp. 383-386, 2000

[8] Sanjay Kr. Singh, D.S. Chauhan, Mayank Vatsa, Richa Singh, "A Robust Skin Color Based Face Detection Algorithm", *Tamkang Journal of Science and Engineering*, pp. 227-234, 2003

[9] Rein-Lien Hsu, Mohamed Abdel-Mottaleb, Anil K. Jain, "Face Detection in Color Images", *IEEE Trasactions on Pattern Analysis and Machine Intelligence*, pp. 696-706, 2002

[10] Maricor Soriano, Birgitta Martinkauppi, Sami Huovinen and Mika Laaksonen, "Using The Skin Locus To Cope With Changing Illumination Conditions In Color-Based Face Tracking", *Proceedings of IEEE Nordic Signal Processing Symposium*, pp. 383-386, 2000

[11] Shay Ohayon, Ehud Rivlin, "Robust 3D Head Tracking using Camera Pose Estimation", *Pattern Recognition*, pp. 1063-1066, 2006

[12] Website reference, Pete Mcollum, "An Introduction to Back-Propagation Neural Networks", *http://www.seattlerobotics.org/encoder/nov98/neural.html*

[13] Christophe Garcia and Georgios Tziritas, "Face detection using quatized skin color regions merging and wavelet packet analysis", *IEEE Transactions on Multimedia*, pp 264-277, 1999

[14] Sébastien Wagner, Bram Alefs, Cristina Picus. "Framework for a Portable Gesture Interface", *7th International Conference on Automatic Face and Gesture Recognition,* pp.275-280, 2006