Eye-hand Hybrid Gesture Recognition System for Human Machine Interface

N. R. Raajan, R. Krishna Kumar, S. Raghuraman, N. Ganapathy Sundaram, T. Vignesh
Department of ECE, SEEE, SASTRA University
Thanjavur, Tamil Nadu, India
nrraajan@ece.sastra.edu, krishaliaskk@gmail.com, raghuraman1291@gmail.com, n.ganapathy91@gmail.com, vignesh.ykciv@gmail.com.

Abstract— **Gesture Recognition has become a way for computers to recognise and understand human body language. They bridge the gap between machines and human beings and make the primitive interfaces like keyboards and mice redundant. This paper suggests a hybrid gesture recognition system for computer interface and wireless robot control. The real-time eye-hand gesture recognition system can be used for computer drawing, navigating cursors and simulating mouse clicks, playing games, controlling a wireless robot with commands and more. The robot illustrated in this paper is controlled by RF module. Playing a PING-PONG game has also been demonstrated using the gestures. The Haar cascade classifiers and template matching are used to detect eye gestures and convex hull for finding the defects and counting the number of fingers in the given region.**

Keyword – hybrid gesture, machine interaction, real-time vision system, eye gesture recognition.

I. INTRODUCTION

A real-time online gesture recognition system promises a great deal of things. They are going to act as an interface between users and the computers in the near future. These systems can make the interaction between differently abled people and the computers, more effective. Lots of research work has been done in the past decade on eye tracking and hand gesture recognition systems. The main aim of such a system is to recognise the gestures made by users, with their hands, eyes and perform the corresponding tasks, as expected, unambiguously. Various image processing tools and algorithms are available to accomplish this challenge. This paper harnesses the power of OpenCV libraries which has better performance in terms of speed for efficient functioning of real-time gesture recognition task.

II. RELATED WORKS

Many invasive techniques are available for eye tracking and hand gesture recognition. After the development of technology for image and video processing, better algorithms and image processing tools paved the way for non-invasive techniques. Even then coloured gloves, permanent markers, materials embedded in cornea are used for recognising gestures based on hands and eyes. Our technique gives a robust result without employing any of the above invasive methods. Adaptive thresholding is a common technique employed along with hough circles to detect an open or closed eye. Our method has high reliability when compared to this method. Other methods include complex valued artificial neural networks and complex wavelet transforms which cannot be used for real-time applications due to speed limitations. Biological measurement techniques like Electro-oculogram(EOG) are available for eye blink and gaze detection which again is an invasive technique. The Infrared-illumination technique uses infrared light to make pupil white and use image subtraction techniques to detect its movement. Our method uses just web cam with optimum illumination to detect eye gestures like closed eye, left gaze, etc. For hand gestures, statistical models are available for classification like hidden Markov model. Our approach is to have a reliable, faster recognition system for real time applications.

III. HAND REGION EXTRACTION

A. Background Subtraction

Background subtraction is one of the most fundamental image processing operations. Before performing background subtraction, a background model needs to be prepared. The background scenes in an unconstrained environment often contain complicated moving objects such as curtains fluttering, fans turning, trees waving in the wind, etc. Light intensity might also vary in such scenes depending on door-window positions and weather conditions. So the normal averaging background method would not be suitable. A good method to face this is to develop a time-series model for each pixel or a group of pixels to deal with the temporal fluctuations well.

To obtain a performance fairly close to that of adaptive background subtraction, we form a codebook to represent significant slowly varying states in the background. We compare the present value of a pixel with the previous values. If this value is near the previous value, it is considered as a small variation on that color. If the present value is not close to the previous value, then it starts a new code element and links it with the pixel. We

choose HSV colour space since it has a separate axis aligned with brightness. This separate axis helps us because background variation in most cases is not along the colour axis, but along the brightness axis.

The following diagram clearly gives an example of the code method. As we can see, a codebook can be considered to be made up of boxes.

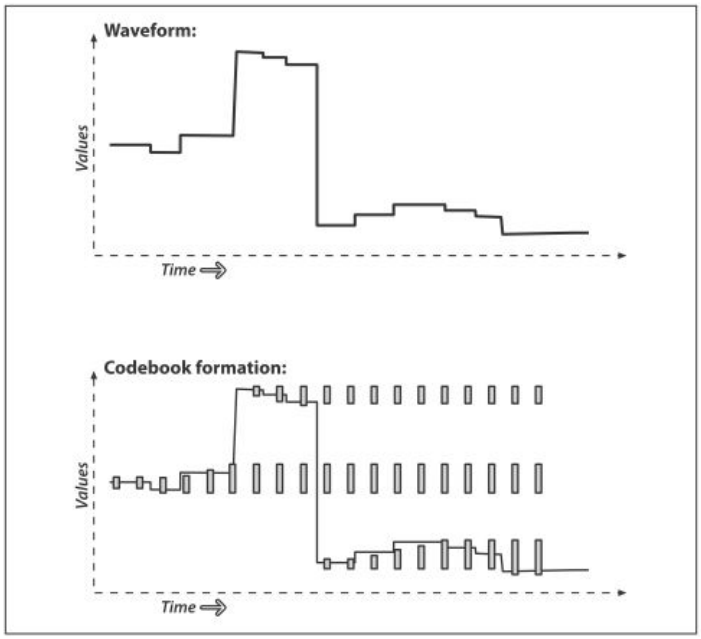


Fig. 1. Codebook Method example

Initially we take nearly 300 samples for the background model. Each pixel in the background is maintained with a codebook (structure) to make note of its variations. If the pixel changes during the sampling period, the codebook expands to include the value. In the above picture, each box is a code element. Thus each pixel will have codebook with different number of code elements.

Each code elememt has two thresholds (max and min) for each axis in the HSV colour space. These thresholds will enlarge (max increasing, min decreasing) if the new samples from the background fall within the thresholds used for learning (learnHigh and learnLow). Otherwise if the samples fall outside the thresholds (max, min) and (learnHigh, learnLow), then a new code element will be created. This is how we create the background model. In the background difference mode, we define acceptance thresholds maxMod and minMod. By use of these threshold values, we can prevent creating a new code element if a pixel value is close enough to a max or a min boundary.

The pictures portray the background and the extracted foreground using codebook method

Fig. 2. Foreground with background

Fig. 3. Foreground extracted separately

B. Finding largest contour area

We have assumed than the only foreground moving object is hand. Therefore after background subtraction, we will get a noisy binary image with the extracted hand region. To remove the noises, we find the contours in the noisy binary image and then take the only contour with the largest area. Contours are the outlines of the

Binary Linked Objects present in the image. This gives us a clear binary image with the hand region only and with less noise. The picture with noiseless hand region is given below.

Fig. 3 and Fig. 4 Unwanted portion in hand region

However as we can see in the picture above, there are some objects attached with the hand due to sudden unpredictable change in camera position or illumination. To reduce this, we use the above binary image as mask in the original frame. In the masked original frame, we extract the hand region using HSV colour space. The Hue, Saturation and Brightness (V) values for skin colour are found to be from 0 to 28, 8 to 140 and 0 to 255 respectively.

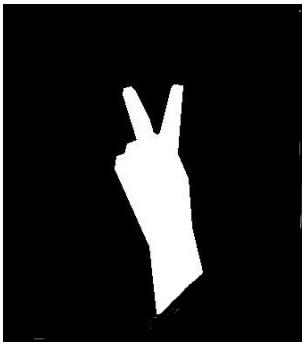


Fig. 5. Masked hand region Fig. 6. Final binary hand image

IV. Hand Gesture Recognition

In our method, the number of fingers shown in the frame is considered as a gesture. Given the binary image of the hand region, the number of fingers shown is calculated by means of convex hull. A convex hull of a set of M points in space is the smallest convex set that includes all these M points. The convex hull is found by means of Sklansky's algorithm [7]. Then we find the defects inside the convex hull and take note of the defects' start and end points. The start points approximately give the position of the tip of the fingers. The following figure gives an explanation of convex hull and defects.

Fig. 7. Convex hull

The red line represents the convex hull. The black lines with arrow marks represent the defects. The blue points are the start of the defects and the red points are the end of the defects or depth point.

Hence the number of start points gives the number of fingers. However due to some unavoidable noise in the image, the start points and end points tend to vary. If this varies frequently, it cannot be used to control a bot.

So we define a rectangular area in the frame. The number of start points inside the rectangle is taken as the number of fingers. Care is taken to define the rectangular area in the portion of the background which is noiseless. The following picture shows our output which gives the number of fingers. The blue rectangle is the defined rectangular area.

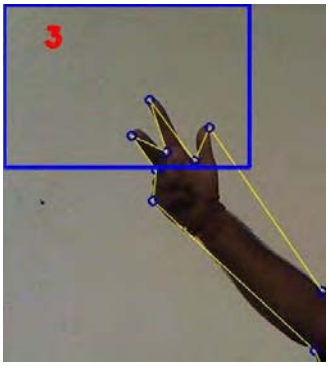


Fig. 8. Output displaying the number of fingers.

V. Face Detection Classifier Training

A. Creating .xml file using HaarTraining for Frontal Face detection

The OpenCV libraries provide the necessary tools to detect a particular region of the images like faces. HaarTraining is a function which can help us create our own classifiers by giving appropriate data. The training phase was about 10 days. There are many databases available for HaarTraining online, which can be used during training phase. We collected database of images for positive samples that contain the region of interests, negative samples with no region of interests and the test samples for the detection phase of our training. About 2000 negative samples and 3000 positive samples were collected for training. The next step was to create samples using these databases which can be achieved using createsamples command with –img (positive samples from database with distortions), -bg (negative samples), and –vec (output images) shell commands. To create training samples without distortion cvCreateTestSamples can be used by –info and –vec shell commands. To create test samples cvCreateTrainingSamplesFromInfo function can be used by shell commands –img, -bg and –info. The HaarTraining can be used by specifying various parameters like number of positive samples, number of negative samples, minhitrate, maxhitrate, weighttrimming, sample' size, number of stages, memory size, vertical symmetry (faces) and mode for Haar like features. The output of HaarTraining is an xml file. The performance can be estimated by using shell commands –data, -w, -h, -info, -ni. The xml file created can detect frontal face and has a performance slightly less than default xml file provided by OpenCV. This was mainly because of inadequate and ad-hoc training data sets.

VI. Eye Detection Classifier Training

A. *Creating .xml file for Eye detection using HaarTraining*

The same procedure as in face detector training was followed except that the database used was different. A new database was created by cropping the frontal face region of every positive sample images obtained online. Again the performance of these classifiers were slightly less when compared to default xml file provided by OpenCV.

VII. Real Time Face And Eye Detection

The .xml files created for the eyes and the faces are loaded into a cvHaarClassifierCascade datatype variable. A linked list for the face region can be created using cvHaarDetectObjects which returns a CvSeq of the face region. The type-casted points of ROI are extracted with cvGetSeqElem which returns a character pointer from the linked list. This ROI of the image can be highlighted with a geometrical shape. The same method is employed in detecting eye ROI except that the image used here is around the eye region of our face.

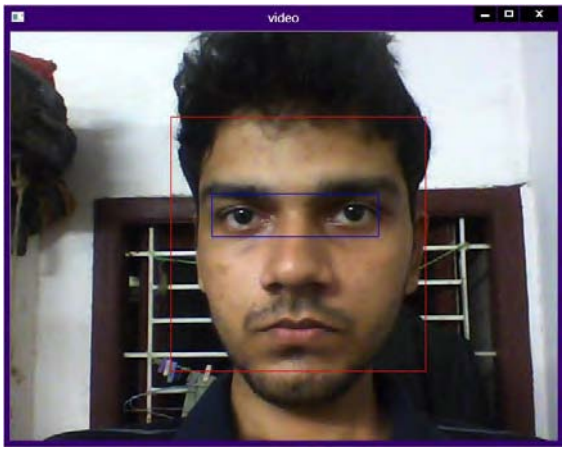


Fig. 9. Face and eye detection

VIII. DYNAMIC TEMPLATE MATCHING

Template matching is the algorithm used here, to find the state of an eye. The method is used dynamically to detect the closed, opened, left or right gazed eye. The template is initially obtained from the user thus takes into account the illumination effects. This template can be used continuously for tracking eye gestures unless there is an appreciable change in the illumination effects. cvMatchTemplate function can be used to do this process in real time. This creates an image which has correlation maps of template with original image. The maximum value of the correlation map image is found using another function cvMinMaxLoc which can be used to determine the state of the eye by simple comparison between the values. Normalized values were obtained by using CV_TM_CCOEFF_NORMED.

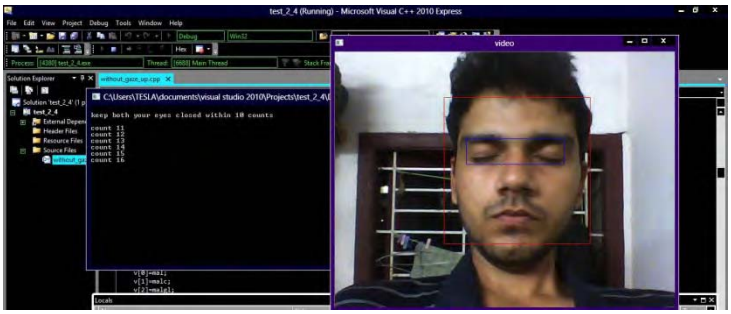


Fig. 10. Closed eye template

Fig. 11. Templates ROI extracted dynamically for different gestures

IX. REAL TIME HYBRID GESTURE DETECTION

The real-time detection of hybrid gestures was a success. The number of false hits was less. The figure illustrates the detection process.

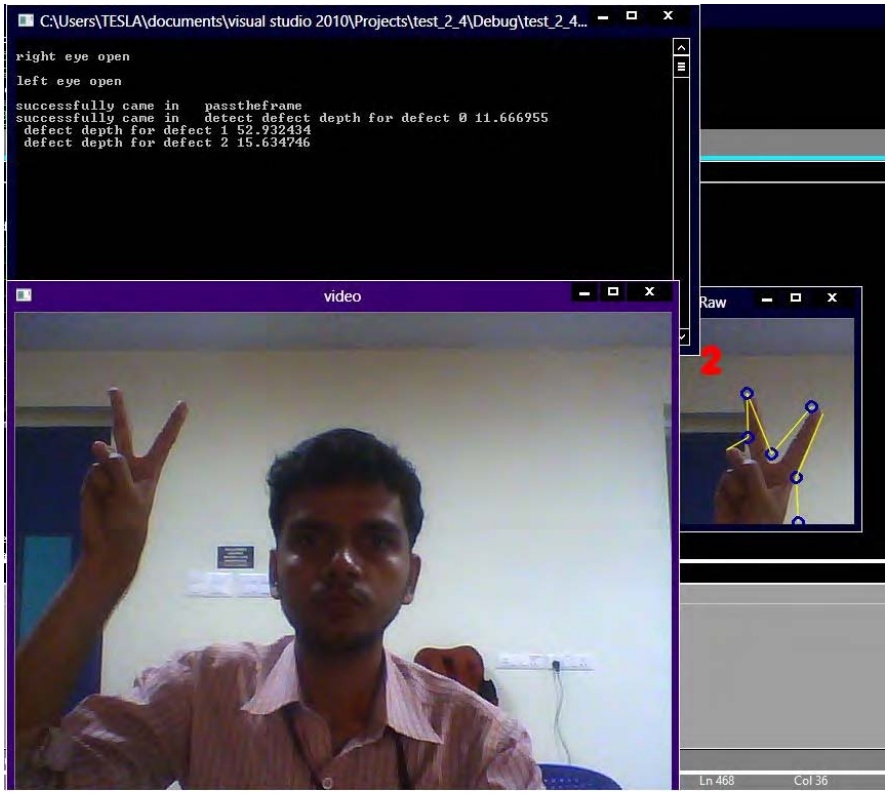


Fig. 12. Hybrid gesture recognition

X. APPLICATIONS

A. *Gesture Recognition robot*

The simplest accomplishment of such a hybrid system is controlling a robot based on hybrid gestures. Serial communication to MSP430G2553 with RF modules helped us execute this process. When the left eye was closed and the number of fingers was equal to two the robot moves in left direction. Many different hybrid gestures were assigned and the task was successfully executed.

B. *Cursor movements based on hybrid gestures*

The next attempt was to move the cursor based on hybrid gestures. The left gaze with one finger moves the cursor to the left whereas with two fingers it does not. This was also successfully accomplished. The following picture illustrates it with pointer trails.

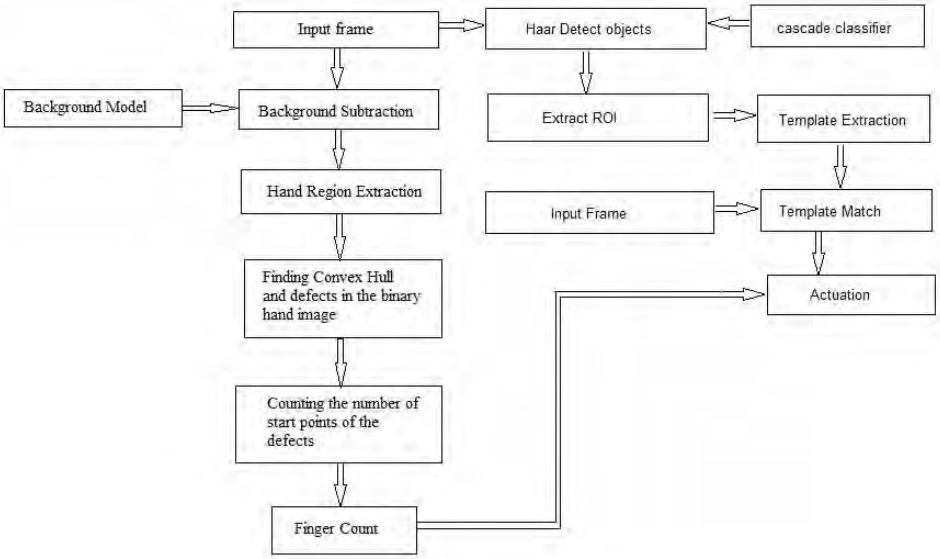
Fig. 13. Mouse pointer moving in downward and left directions

C. *Playing Ping-Pong with gestures*

The system's effectiveness as a HMI can be demonstrated by playing a simple C++ graphics based game created by our team based on Ping-Pong. This was also successfully accomplished with our hybrid gestures. Here we simulated key presses on the keyboard using gestures since we created this game for playing with keyboard, initially.

Fig. 14. Playing Ping-Pong with gesture recognition

XI. FLOWCHART



XII. CONCLUSION

We have proposed an algorithm which is simple and takes less computational time for the hybrid gesture recognition problem. Once the Haar training phase is completed, the system can operate in a standalone fashion with no further modifications. Thus they can act as a promising HMI in near future. The system can also assist differently abled people in interfacing with the computers.

ACKNOWLEDGMENT

We include our special thanks to all the researchers working in field of image processing and computer vision who in some way helped us to attain our goal. Also, we wish to express our gratefulness to the faculties in our university who shared their views and gave suggestions to make our project a success.

REFERENCES

[1] J. Davis and M. Shah "Visual Gesture Recognition", IEEE Proc.-Vis. Image Signal Process., Vol. 141, No.2, April 1994.
[2] Asanterabi Malima, Erol Ozgur, and Miijdat Cetin “A Fast Algorithm for Vision-Based Hand Gesture Recognition for Robot Control”, IEEE International conference on Signal Processing and Communications Applications, 2000.
[3] Sebastian Marcel, Oliver Bernier, Jean Emmanuel Viallet and Daniel Collobert, “ Hand Gesture Recognition using Input – Output Hidden Markov Models”, IEEE Proc. on Automatic Face and Gesture Recognition, pp. 456 - 461 2000.
[4] C.-C. Chang, I.-Y Chen, and Y.-S. Huang, "Hand Pose Recognition Using Curvature Scale Space", IEEE International Conference on Pattern Recognition, 2002..
[5] Qing Chen, Nicolas D. Georganas and Emil M. Petriu, “Hand Gesture Recognition Using Haar-Like Features and a Stochastic Context-Free Grammar”, IEEE Transactions On Instrumentation And Measurement, Vol. 57, No. 8, August 2008.
[6] Nasser H. Dardas and Nicolas D. Georganas, Real-Time Hand Gesture Detection and Recognition Using Bag-of-Features and Support Vector Machine Techniques, IEEE Transactions On Instrumentation And Measurement, Vol. 60, No. 11, November 2011.
[7] J. Sklansky, “Measuring concavity on a rectangular mosaic”, IEEE Transactions On Computers, Vol. C-21, No. 12, December 1972

[8] T.S. Huang and V.I. Pavlovic, “Hand Gesture Modeling,Analysis and Synthesis,” in International Workshop on Automatic Face and Gesture Recognition, pp. 73-79, 1995.
[9] Gary Rost Bradski and Adrian Kaehler, *Learning openCV*, 1st ed., O'Reilly Media Inc., September 2008.
[10] C. L. Lisetti and D. J. Schiano "Automatic classification of single facial images", Pragmatics Cogn., vol. 8, pp.185 -235 2000
[11] V. I. Pavlovic , R. Sharma and T. S. Huang "Visual interpretation of hand gestures for human computer interaction", IEEE Trans. Pattern Anal. Mach. Intell., vol. 19, no. 7, pp.677 -695 1997
[12] J. Davis and M. Shah "Visual gesture recognition", Vis., Image Signal Process., vol. 141, pp.101 -106 1994