Hand Gesture Recognition Using Computer Vision

Pavithra Kannan, Rohan Pradip Shah, Thulasi Srinivasan, Uzma Parveen S

Department of Electronics and Communication
M.S Ramaiah Institute Of Technology
Visvesaraya Technological University
Bangalore 560054
India
rps2689@gmail.com

Abstract

The proposed algorithm describes a vision-based approach to hand gesture recognition, which deals with real-time tracking of the hand with simultaneous recognition of the gesture being made. Accurate tracking of the hand is done using Haar-like features, while gesture recognition is done by computing orientation histograms of video frames. The system is trained using a well-defined vocabulary of hand gestures, and the algorithm generates the current position of the hand and the gesture being made. The AdaBoost algorithm is used to boost the process of Haar-training. The use of orientation histograms provides translational invariance and also makes it adaptable to variations in illumination. The algorithm has been demonstrated on a real-time simplified drawing application that uses hand gestures as a means of human-computer interaction (HCI). The algorithm and its application have been implemented using an ARM-based processor, in order to demonstrate its effectiveness on embedded devices.

Keywords: AdaBoost algorithm, Haar- like features, Hand Gesture, Human- computer interaction, Orientattion Histogram.

1. Introduction

In this paper, we describe a method of implementing, on computer, one of the most sought after means of communication: Gestures. HumanComputer interaction, over recent times, has been given an extremely diversified approach, with Computer Vision being one of the forerunners in the field. In everyday life, human activities deal with the interaction with machines. With gesture recognition, these complicated interfaces can be simplified to create a more user-friendly environment. Integrating Computers with Hand Gestures makes it easier to control commands. Devices such as controllers for games, security systems and television remotes require the user to either push a button or touch a screen for their functioning. Over the centuries, these devices have always been improved and enhanced by advancements in technology, making them easier to operate from the users point of view. In the past decade, a lot of devices that were previously button-operated have been replaced successfully with touch-technology, in which the user generates a command by simply touching a screen. Today, we recognize a definite possibility of reducing these human-machine interactions to a simple hand gesture. With a little imagination, hand gesture recognition can have sundry applications in todays world. We have implemented a real-time algorithm that is completely based upon image processing, without requiring sensors at the users end other than a camera. Other ways of gesture recognition make the use of either an accelerometer or a thermal sensor. In the proposed methodology, the user makes a real- time dynamic gesture with his/her hand in the viewing space of the camera, and this gesture is recognized and used as an input mechanism for some control operation. In this particular implementation, the user may hold his/her palm either facing upwards or sideways and move his/her hand anywhere in the viewing space of the camera. The algorithm processes the video stream in order to return the position of the hand and the gesture made at every instantaneous frame of the live input. Thus, depending on which gesture is made and where it is made, any appropriate operation could be performed with gestures as the sole base of the input mechanism. This idea could have a lot of practical real- life solutions, apart from being present in television controls and big computers. It has been demonstrated during the course of this implementation that this idea could even extend to embedded systems. This implies that even cellular phones and other small electronic gadgets could be operated using our proposed methodology.

2. The Proposed Methodology

This paper proposes an algorithm for the real- time dynamic recognition of hand gestures for HCI. This real- time algorithm is based on two phases of implementation- the detection phase and the recognition phase. In the detection phase, Haar- like features are used to detect certain limited postures of the human hand. Once the hand is detected, the algorithm enters the recognition phase, in which the instantaneous gesture of the hand is calculated using normalized orientation histograms. There is a rigorous training phase before real- time operations can start, which includes generating a multi- staged cascade of Haar features. This involves the creation of an extensive database of positive images as well as negative images. Since Haar-like feature is a weak classifier, it is well integrated with an iterative learning algorithm called Adaboost. This constructs a strong classifier using only a training set and a weak learning algorithm, resulting in the generation of a cascade of Haar classifiers. This cascade is the entity that defines what object is to be recognized, which in our case is the human hand. During this training, care should be taken to include as many different postures of the human hand as possible, in order for the detection to not be too stringent. Also, in real time implementation, detection of hand gestures relies on various factors such as complex backgrounds, intensity variations and different zooming conditions. All these different possible variations, along with noise, must be accounted for while the database is being created. Haar- training is essential for the detection phase of the algorithm. Apart from this, a separately trained hand gesture set must also be incorporated for instantaneous recognition of the hand gesture made, after it has been detected. This set includes a limited number of hand gestures, well- distinguished from one another, which are required to be recognized in the real- time operation of the system. Normalized orientation histograms are created for each image of this training set. In the detection phase, using the cascade that was initially created, for each frame of input, a sub-window containing each Haar-like feature scans the entire image horizontally and vertically, pixel by pixel. The object will be detected if the value of the Haar-like feature is above a certain threshold. The sub-window initially has a smaller size and its width and height increase with iterations by a pre- defined scale factor. This ensures that the hand is detected independent of its position and size relative to the input frame. In the recognition phase, a region of interest containing the detected hand is processed for a gesture match. This is done by creating a normalized histogram for each detection of each successive input frame, followed by comparison of the histogram made with those of the trained set of hand gestures. The algorithm has been implemented in real-time, and also has been implemented using an ARM- based Beagle board, so that it may serve as a working prototype for this mechanism of hand gesture recognition on embedded systems.

3. Haar- Like Features:

Haar- like features are digital image features used in object detection algorithms. They owe their name to their intuitive similarity with Haar wavelets. The main purpose of Haar-like features is to meet the real-time requirements, while providing speed, accuracy and robustness against different backgrounds and lighting conditions. The idea behind Haar-like features is to recognize objects in an image based on the value of simple features, instead of using raw pixel values directly. There are two motivations for the employment of the Haar-like features rather than raw pixel values. The first reason is that compared with raw pixels, the Haar-like features can efficiently reduce/increase the in-class/out-ofclass variability, thus making classification easier. A Haar-like feature in simple terms is a template of multiple connected black and white rectangles as shown in Fig.1 below

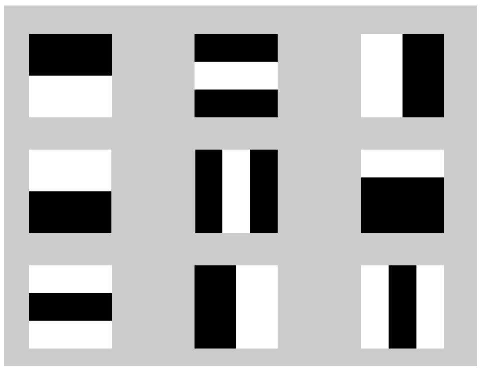


Fig.1. Haar- like features

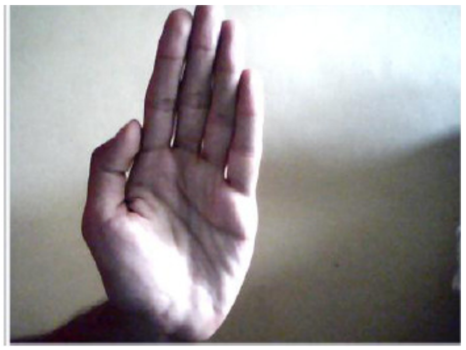
The value of a Haar-like feature is the difference between the sums of the pixels values within the black and white rectangular regions:
The concept of integral image is used to compute a rich set of Haar-like features. Compared with other approaches, which must operate on multiple image scales, the integral image can achieve true scale invariance by eliminating the need to compute a multi-scale image pyramid, and significantly reduces the image processing time. The simplest Haar-like feature is a two-rectangle feature. The value of this is calculated as the difference between the sum of the pixels within the two rectangles. This will indicate characteristics like edges or borders between light and dark regions. Three-rectangle features indicate for instance a dark line or a dark thin area lying between light regions, depending on the size of the middle rectangle. Four-rectangle features compute the difference between diagonal pairs of rectangles, and so on. Using the integral image for computing the sum will allow a two-rectangle feature to be calculated with six references to the integral image, a three-rectangle feature with eight references, a four-rectangle feature with nine references etc.

4. Adaboost

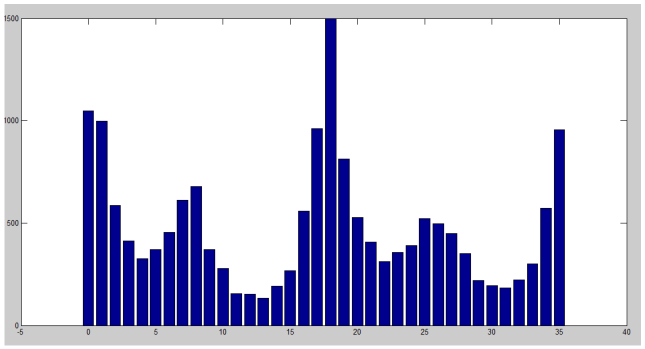
Although, it is not difficult to find a Haar-like feature that has slightly better accuracy than random guessing, a single Haar-like feature is certainly not enough to identify the object with a high accuracy. For this reason, the AdaBoost learning algorithm is used to boost the classification performance of a simple learning algorithm. Boosting is a supervised machine learning algorithm to improve the overall accuracy stage by stage based on a series of weak classifiers. The AdaBoost algorithm first chooses the feature that classifies more data correctly. In the next step, the data is re-weighted to increase the importance of misclassified samples. This process continues and at each stage the weight of each weak learner among the other learners is determined. The AdaBoost learning algorithm starts with uniform distribution of weights over each trained hand gesture sample. A weak classifier is obtained from the weak learning algorithm. The weights are increased for those hand gesture sample images that were misclassified previously. This process continues by the addition of new weak classifiers to the linear combination till the accuracy standards are met. Finally a strong classifier is obtained as a linear combination of the selected weak classifiers.

5. Orientation Histogram

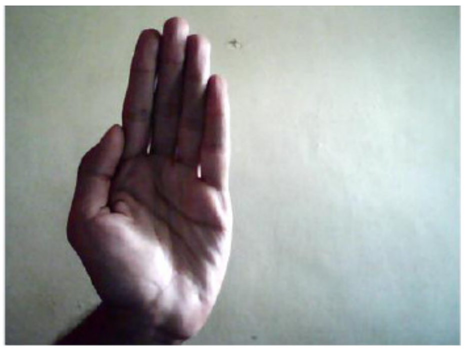
For the purpose of static gesture recognition, we compute orientation histograms for each frame that is being processed. From William T Freeman [1], in an orientation histogram, each bin is classified according to the intensity gradients of adjacent pixels in a digital image. To eradicate the effects of small or redundant gradients, we set a threshold level, below which the contrast between those particular pixels is not good enough for processing. Firstly, there is a training sequence in which a preferred set of gestures is inculcated into the system. The orientation histograms of each of these trained gesticulations are computed. In order to find the local gradients, the intensity values of each pair of adjacent pixels are subtracted in the horizontal as well as vertical directions, and then the inverse tangent of the ratio of the latter to the former is found. Each of these angle values in the matrix are partitioned into four quarters (-180¡angle¡180), based on pixel signs, provided the contrast exceeds the fixed threshold. The magnitude and orientation are calculated by accessing corresponding pixels in the gradient matrices using the formulae: Magnitude = (1) = tan-1(dy/dx) (2) where dx , dy are corresponding pixels of the horizontal and vertical gradient matrices) determines the contrast. The gradients above the contrast threshold qualify into the histogram processing block, in which each gradient is placed into its corresponding bin. This is how an orientation histogram is constructed. The histograms of each of the training input gestures are stored in memory. Next there is a block of logic that continuously monitors a live stream of input video through the camera. There is a frame by frame processing of the live stream. An orientation histogram of each frame is constructed, and compared with those of the trained gestures. This comparison is done by calculating the Euclidean distance between the two histograms being compared. The Euclidean distance is calculated using the formula: ED = (a0-b0)2 +(a1-b1)2+.. (3) Ideally, the Euclidean distance for a match is zero. Thus, with a small margin for error, different gestures can be distinguished from one another. Once a match has been detected, the live gesture has to be held in place for 30 frames for a particular gesture to be officially identified. Once a gesture is ”identified”, processing enters the stage of motion determination.



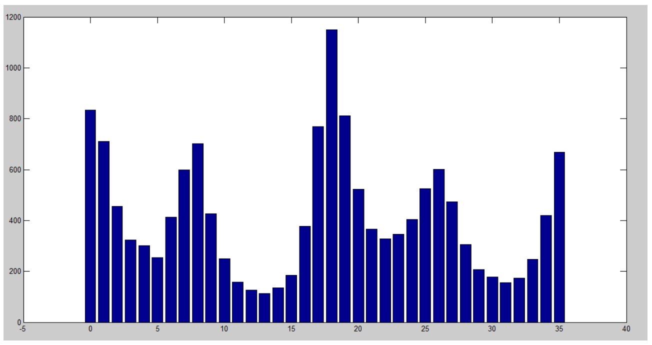
(a)



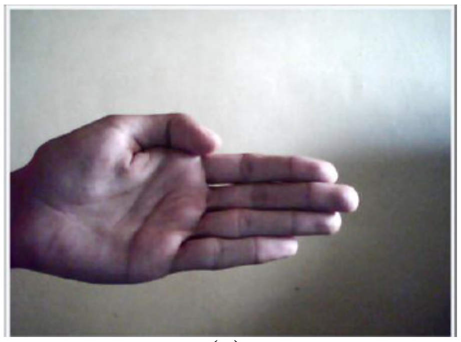
(b)



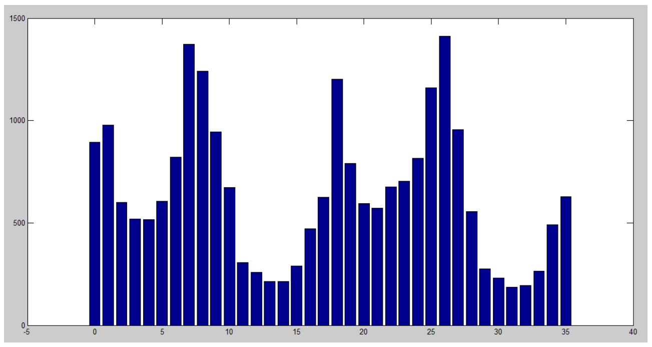
(c)



(d)



(e)



(f)

Fig. 2 (a) and (b) represent an open palm hand gesture and its corresponding orientation histogram. (c) and (d) represent a similar hand gesture and its corresponding orientation histogram.(e) and (f) represent a distinctly different hand gesture and its corresponding orientation histogram.

6. Implementation:

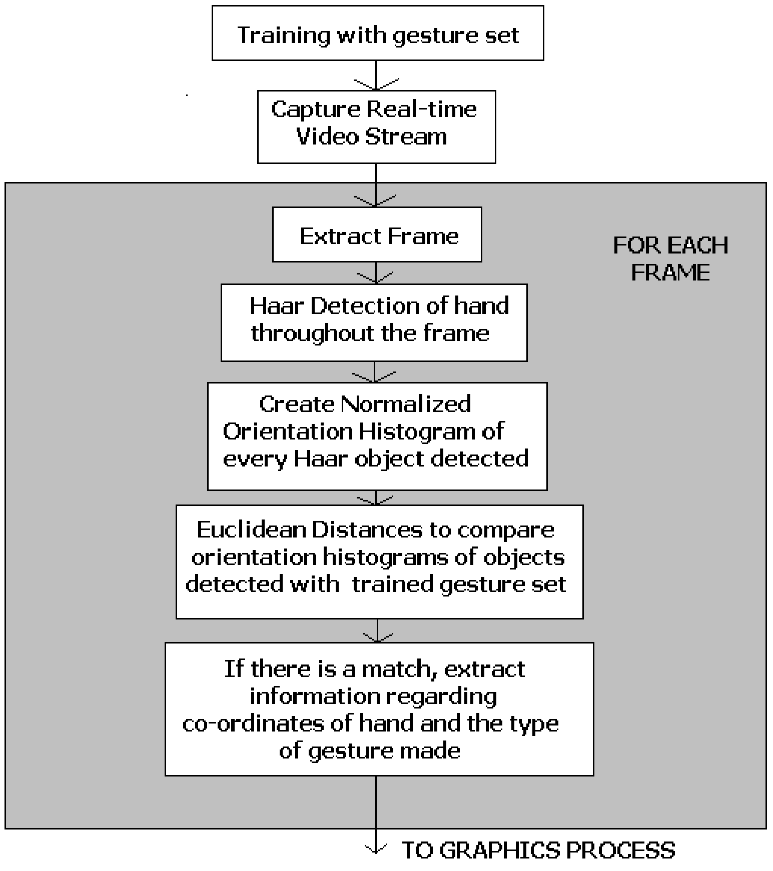

Training with gesture set
Capture Real-time Video Stream
Extract Frame
FOR EACH FRAME
Haar Detection of hand throughout the frame
Create Normalized Orientation Histogram of every Haar object detected
Euclidean Distances to compare orientation histograms of objects detected with trained gesture set
If there is a match, extract information regarding co-ordinates of hand and the type of gesture made
TO GRAPHICS PROCESS

Fig. 3 Block Diagram: Vision Algorithm

The proposed algorithm consists of a combination of Haar-like features and normalized orientation histograms, to achieve dynamic real-time recognition of hand gestures. As mentioned earlier, Haar-like features are used for detection of hand within the viewing space of the camera and orientation histograms are used for recognition of static gestures instantaneously. The flow of the algorithm is shown in the block diagram above. The initial step is to train the gesture set. A limited number of distinguishable gestures are trained into the system. Normalized orientation histograms of these gestures are created, in order for them to be compared to those of the gestures made in real-time. In this particular implementation, the training set consists of palm facing upwards and sideways. Next, the live input video stream is captured through the camera. Thus any potential gesture or movement made in front of the camera from this point onwards will be processed for recognition.

Each individual frame of the video stream will be subject to the various phases of the algorithm. The frame rate is. As mentioned above, each frame is individually processed and hence needs to be extracted from the input video stream. The first phase of processing, Haar detection, is then carried out by using the cascade that was made during the training phase. A sub-window consisting of the Haar-like features is iteratively scanned through the image from left to right and top to bottom. This ensures that the detection of the hand would be independent of the relative position of the hand in the image space. After each iteration, the size of the sub-window is increased in accordance with a scaling factor as defined by the algorithm. This ensures that the detection occurs independent of the relative depth of the hand with respect to the camera. In the cascade of Haar classifiers, the first stage would be the weakest of classifiers, and the strength of the classifiers in each stage progressively increases. Each stage is represented as a binary node wherein the output may be either a pass or a reject. Thus only those image features are detected which pass all the stages of the cascade, for any particular input frame. In each frame, for every hand detected by the previous stage, the region of interest (ROI) of that particular detection is passed over to another segment of the algorithm that does histogram computation. By the term region of interest we mean the region of the image that encompasses the detection that was made by the Haar-like feature cascade, i.e. the region encompassing the hand. This region is processed so as to compute its normalized orientation histogram. By normalizing the orientation histograms, we are accounting for the differing sizes of the ROI of the objects detected, since the orientation histograms of the detected and trained hands can only be compared if they are of the same dimensions. This is a key feature in the working of the mechanism. The normalized orientation histogram of the ROI of the hand detected is then compared to that of each of the trained hand gesture images. This comparison is done using the concept of Euclidean distances. An appropriate threshold is set for the Euclidean distances. This threshold depends on various factors such as noise and operating conditions which may vary from system to system. For any particular gesture from the trained set, if the Euclidean distance value is within the threshold defined, it is designated as a match. If the above process yields a match, the positional co-ordinates of the hand in the image space, in terms of pixels, is extracted, along with the type of gesture that was recognized. The block diagram shows an example in which this information is passed on to a graphics application after being extracted.

7. Results and future work

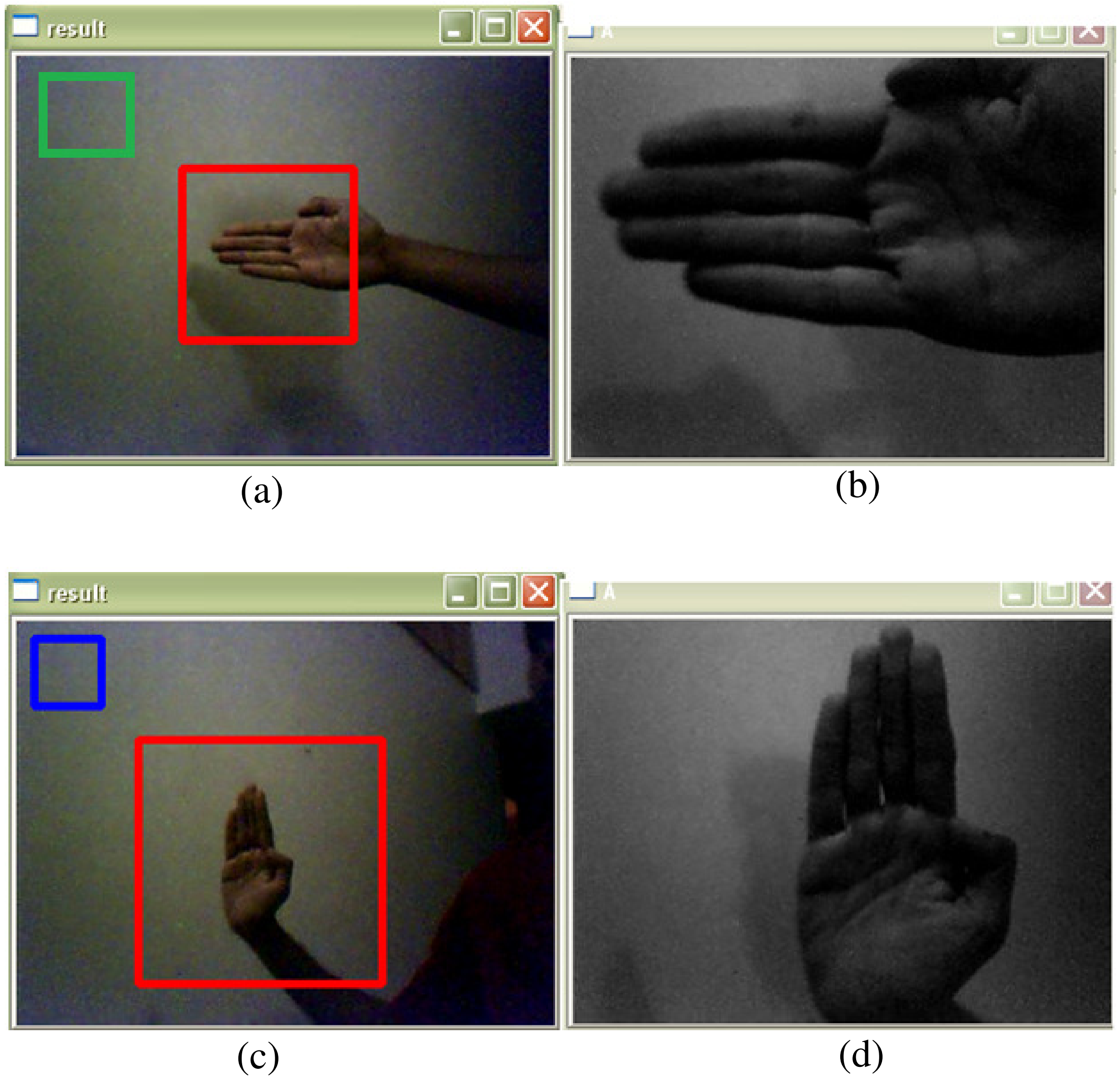


Fig .1 (a) and (b) represent an open palm hand gesture and its corresponding orientation histogram. (c) and (d) represent a similar hand gesture and its corresponding orientation histogram.(e) and (f) represent a distinctly different hand gesture and its corresponding orientation histogram.

The algorithm implemented successfully detects a hand posture as well as recognizes a hand gesture in real- time. In the implementation of the algorithm, detection of the horizontal hand posture is indicated by the appearance of a blue rectangular box as shown in Fig.4 and the vertical hand posture results in the appearance of a green rectangular box as shown in Fig.5. The movement of the hand from the horizontal gesture to the vertical gesture is indicated by the transition from a blue to a green rectangular box. The transition occurs when the orientation of the hand is in the range of 40 to 50. We can assertively say that the error is 5. The performance of the algorithm is at its best when the hand is positioned at the centre of the camera viewing space. There is a gradual decrease in performance when the hand is moved towards the boundaries. In order to achieve robustness to varying conditions of operation such as illumination, posture, zooming conditions, skin colour and size of the hand, it is crucial to incorporate all possible variations during the training phase of the system. For this purpose, we have compiled an extensive database of hand gestures. The training set of our detection phase comprised of over 1000 positive image samples and 2000 negative image samples. In contrast, the training set of the recognition phase must contain gestures that are clearly distinguishable from one another. Hand gesture recognition has innumerable applications. In our particular implementation, we have interfaced the computer vision algorithm with a graphics application- a drawing application in which one gesture represents the marker, and the other represents the eraser. The algorithm has been implemented on an ARM based BeagleBoard , that demonstrates the Texas Instruments OMAP 3530 system-on-a-chip which includes an ARM Cortex- A8, TMS320C64x+ DSP for accelerated video and audio decoding , and a Graphics processing unit to provide accelerated 2D and 3D graphics. The BeagleBoards powerful performance and multi- functionality enables its use as an effective HCI mechanism for different Linux- based application on an embedded platform.

References

[1] Alexander Kuranov, Rainer Lienhart, and Vadim Pisarevsky, ”An Empirical Analysis of Boosting Algorithms for Rapid Objects With an Extended Set of Haar-like Features”, *Intel Technical Report MRL-TR*, July, 2002.

[2] Andre L C Barczak and Farhad Dadgostar, Real-time Hand Tracking Using a Set of Cooperative Classifiers and Haar-Like Features,*Research Letters in the Information and Mathematical Sciences*, ISSN 1175-2777, Vol. 7, pp 29-42, 2005.

[3] R K McConnell, ”Method of and apparatus for pattern recognition”, U. S. Patent No.4,567,610, January, 1986.

[4] Messom, C.H. and Barczak, A.L.C., ”Fast and Efficient Rotated Haar-like Features Using Rotated Integral Images”, *Australian Conference on Robotics and Automation*, pp. 1-6, 2006

[5] Rainer Lienhart and Jochen Maydt, ”An extended set of Haar-like features for rapid object detection”, ICIP02, pp. I: 900-903, 2002

[6] Paul Viola, Michael Jones, ”Rapid object detection using a boosted cascade of simple-features”, *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 511518 (2001)

[7] Qing Chen, Francois Malric, Yi Zhang, Muhammad Abid, Albino Cordeiro, Emil M. Petriu and Nicolas D. Georganas, ”Interacting with digital signage using hand ges-

tures", *International Conference on Image Analysis and Recognition*, LNCS 5627 , pp.347-358, Halifax, Canada, July 2009.

[8] Qing Chen, Nicolas D Georganas and Emil M Petriu, "Real-time vision-based hand gesture recognition using Haar-like features", *IEEE Instrumentation and Measurement Technology Conference*, Warsaw, Poland, May 2007.

[9] William T Freeman and Michal Roth, "Orientation Histograms for Hand Gesture Recognition", *IEEE Intl. Wkshp. on Automatic Face and Gesture Recognition*, Zurich, June, 1995.