

A Hand Gesture Recognition Based Virtual Touch Wall

P. N. V. S. Gowtham

Abstract—The aim of this paper is to provide with a real time application based on hand gesture recognition. Virtual touch is a concept whereby any normal surface with no internal circuitry and hardware for touch sensitivity, is converted into a touch sensitive surface by using image processing techniques. Using such a technique the paper presents the transformation of a plain white wall into a large touch wall. The virtual touch wall aims at building a gesture sensitive environment for users, to provide them new ways of interaction in their work atmosphere and in their recreational space. The hardware requirements of the system are kept on the minimum, limiting only to a simple USB webcam and a PC. Other alternatives to the PC have also been presented, such as the Beagle Board, which is a mini version of a PC and very inexpensive. The approach is based on simple and fast motion detection trying to eliminate unnecessary regions of interest and a recognition algorithm based on the hand contours, their convexities and a further comparison of hand shapes based on HU moment matching which works on image contours. The paper also presents a way of improving the distorted hand contours by distinguishing between external contours and holes and explicitly filling up holes. The use of a predefined set of contours for each gesture helps in better recognition. A simple state machine is implemented to further improve reliability. A total of 5 gestures have been implemented and tested on the PC and the Beagle Board and the performance compared. The Linux XLIB library and the X display functions have been utilized to control mouse cursor actions and other display properties of the wall.

Index Terms— Gesture Recognition, HU Moments, Hand Tracking, Contours, Xlib, Beagle Board, virtual gesture wall

I. INTRODUCTION

One of the greatest gifts of almost all living things in this world is the ability to communicate through vocals and actions. Human and machine interaction by far has only been through simple means of communication like a mouse, keyboard or switches. Voice recognition and gesture recognition are two powerful and more natural means of communicating with machines as is with human beings.

There are innumerable instances where conventional applications can be replaced with hand gestures[1]-[4]. In many cases special gloves or markers have been used for efficient detection and tracking which constraints the user and other vases demand a very simple and plain background [5]-[7]. In [8] a very real time method of hand gesture recognition based on convexity defects is presented but an analysis based only on convexity defects is dependent on smoothness of background. [9] Implements gesture recognition based on Hausdroff distance but is not real time especially when implemented on dedicated systems such as

Beagle Board. This paper attempts to solve these problems by proposing a real time gesture recognition system with no user constraints and with any kind of background environment. Further the paper presents a practical application of gesture recognition by implementing an interactive gesture based virtual touch wall. Such a system consists of a projector, a projecting surface and a processing unit. Interaction between the user and the projecting surface is based on gestures than on conventional methods such as mice and keyboard. Such a system provides a new dimension for users in their work, gaming and recreational aspects of their lives.

II. SYSTEM ARCHITECTURE AND HARDWARE BLOCKS

Based on hardware level abstraction, the entire system is sub divided into three major hardware modules. They are the Beagle Board module, the Camera Module and the Projection Module.

Fig. 1 shows the physical arrangement of the entire system.

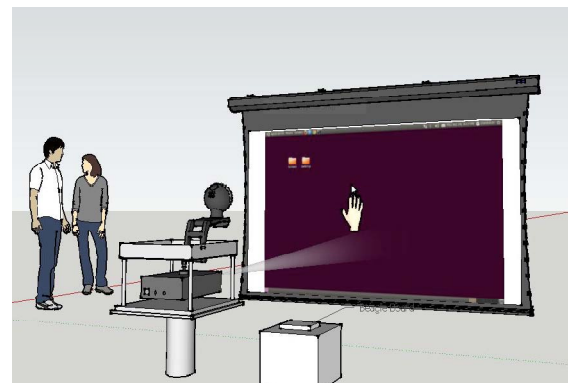


Fig. 1. Physical system setup

A. The Beagle Board Module

The role of this module is to accept frames from the camera module and perform several image processing algorithms on the frame and display its output on the projection module. Beagle board is a low-cost, single board computer developed by Texas Instruments [10], [11]. It is equipped with OMAP-3530 System on chip processor, which includes an ARM Cortex-A8 core. Ubuntu Natty 11.04 is installed on the Board using an external SD card, prior to which the card is set up with an MSDOS type boot partition and the Linux File System partition[12]. The OS which is in the form of a tar file is untarred using the command:

```
tar -xjvf armel-rootfs.tgz
```

This function copies the root file system to the file system partition. The boot files namely MLO, U-boot and Uimage are copied to the Boot partition manually in specific order. The SD card is then placed on the SD/MMC card slot of the board and the user button is used to boot the OS. Fig. 2a shows the Beagle Board and its internal architecture. Opencv

Manuscript received December 20, 2011; revised February 10, 2012.

P. N. V. S. Gowtham is with the Department of Electronics and Communications, International Institute of Information Technology (e-mail: gowtham.peddada@gmail.com).

which is a library of programming functions is used to perform computer vision algorithms and operations on the Beagle Board or any Linux system in general[13]-[15].

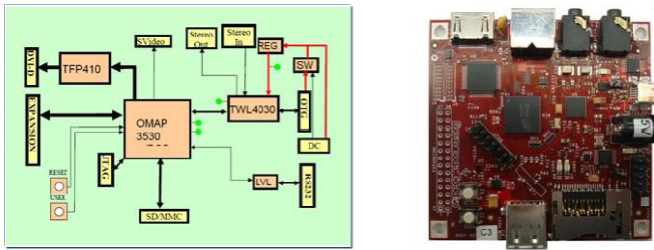


Fig. 2(a). The Beagle Board and its System architecture

B. The Camera Module

The camera provides frames to the Beagle Board in real time. The webcams tested and used are the Logitech C100 and the Logitech C200.

C. The Projection module

The projection module consists of a projector to project the operating system of the Beagle Board and a projecting surface. Any smooth white surface works well with the system.

Fig. 2(b) shows the hardware block diagram of the entire system

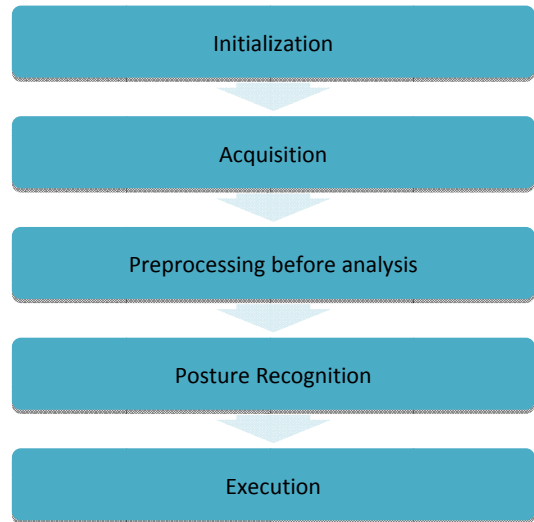


Fig. 3(a). Flow diagram

B. Acquisition

Once the system is begun, every frame from the webcam is acquired and processed. Each frame is a 3 channel RGB frame with each channel having 8 bit depth. The size of each frame is 640X480.

C. Pre Processing Before Analysis

A series of image processing operations are undertaken on each frame captured from the webcam. These operations include motion detection, skin segmentation, threshold, smoothing, contour analysis, hole filling, morphological operations such as dilation, contour cropping and contour approximations.

D. Posture Recognition

Once the user's hand contour has been successfully retrieved, the number of fingers in the contour are counted using convexity defects and then recognition is made by comparison with the contours stored in the startup step. For this purpose of recognition a comparison technique based on HU moments is presented.

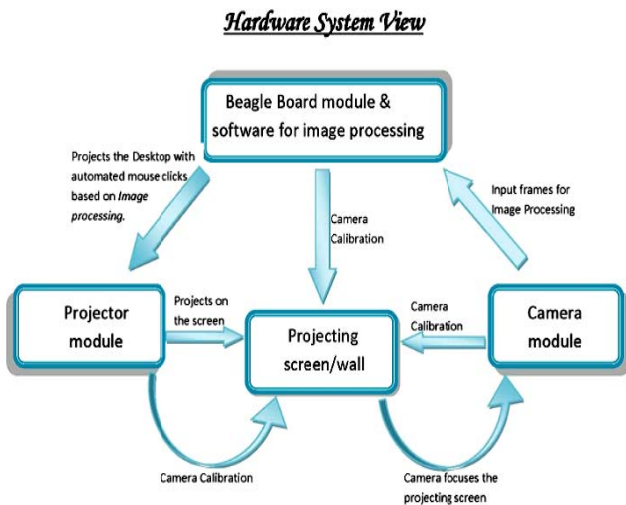


Fig. 2(b). The hardware block diagram of the entire system

III. WORK FLOW

There are five stages in the working of the system as shown in the Flow diagram in Fig. 3.a. The five stages are Initialization, Acquisition, Pre-Processing before analysis, Posture Recognition and Execution. These five stages occur in succession as long the system is active.

A. Initialization

A set of recognizable postures are stored in the system memory at the startup of the system [1]. These postures correspond to the contours of different hand gestures. They are very essential in the later stages of Hand Recognition.

Functional relationship view

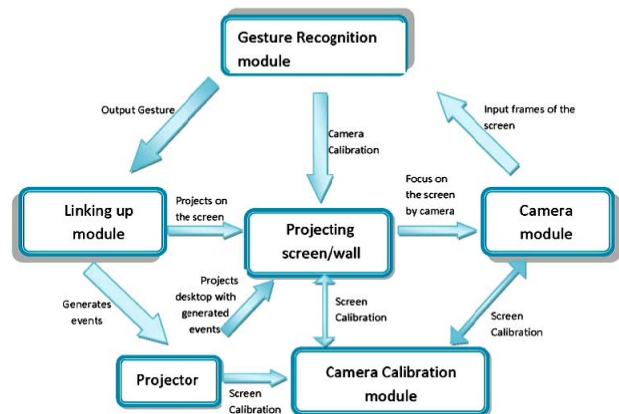


Fig. 3(b). Functional relationship between the three software modules

E. Execution

Once the gesture has been recognized, the corresponding action is executed. Several actions based on mouse events have been presented in the later sections.

These five stages are shared by three modules on the software level. They are the Gesture recognition module, the Linking module and the Camera Calibration module. Fig. 3b shows the functional relationship between the three software modules.

IV. GESTURE RECOGNITION MODULE

The steps involved in recognizing hand gestures are as shown in Fig. 4

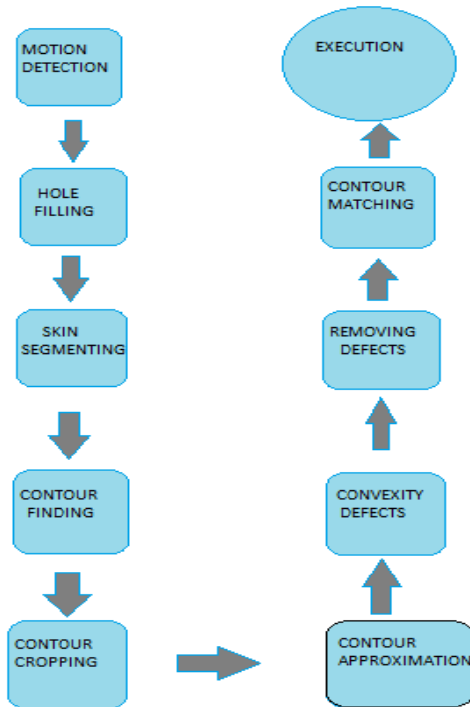


Fig. 4. Flow chart of the algorithm for gesture recognition

A. Motion Detection

Every frame from webcam is converted into a gray scale image and a pixel wise subtraction is performed between the current frame and previously acquired frame. The resulting output is then smoothed using a Gaussian filter to remove noisy pixels. A threshold is now applied on smoothed image thereby changing it into a binary image with maximum value 255 and minimum value 0. 255 corresponds to pure white and 0 corresponds to pure black. The requirement of a fixed background is eliminated by using motion detection. The background subtraction image would immediately adapt to new background with an error only in the first frame after change. Change in background should not be continuous in time domain, but can be discrete at random time intervals.

A simple threshold of 10 is experimented to work well [16].
 If $image(i,j) > 10$; then $image(i,j)=255$;
 Else $image(i,j)=0$;
 Fig.s 4.1.a, 4.1.b show original frame and threshold I mage respectively.



Fig. 4.1a. Original Frame



Fig. 4.1b. Threshold image



Fig. 4.2a. Dilation output



Fig. 4.2b. Combination output

B. Hole Filling and Skin Segmentation

Output of motion detection does not yield the full area of the object under motion. As seen in Fig. 4.1b, there are many black regions or holes with in white boundaries in the motion detection image. For this reason, a hole filling operation is performed. Dilation reduces number of holes, but hand contour will still not be devoid of holes. Fig. 4.2a shows Dilation output. At this stage multiple dilations are not performed as this increases size of unnecessary regions in the image.

A logical AND is performed between the dilated image and the original frame. This new image is now converted into YCrCb color space and an upper and lower bound are set to color segment it. This process is not accurate, but will suffice because most of the image is eliminated in the motion detection image leaving apart only very small areas which move.

The lower limit is (0,113,67) and the upper limit is (255,173,127). Combination of background subtraction and skin segmentation is shown in Fig. 4.2b which is the final output.

C. Finding appropriate Contours

A contour analysis gives groups of pixels that have same label due to their connectivity in binary image. After such an analysis the image is treated as a set of contours and not as a collection of discrete pixels. After contour analysis individual contours with a minimum size of 4000 pixels are isolated.

The largest contour is the one with highest probability of being a hand. Once this contour has been selected, hole filling operation is performed on the particular contour. There are two types of contours namely exterior contours and holes. Holes associated with the largest contour are identified in the image and are filled white. This is a contour specific hole filling unlike the dilation performed in previous steps. Fig. 4.3a and 4.3b show the output contours before and after contour hole filling.



Fig. 4.3a. Before Contour filling

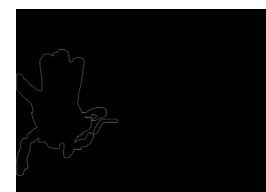


Fig. 4.3b. After Contour Filling

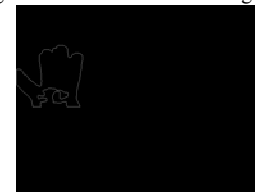


Fig. 4.4a. Upper Half Contour



Fig. 4.4b. Polygon Approximation

D. Contour Cropping and Contour Approximations

The most useful information of the lies in its fingers .Hence the hand contour is cropped to the upper portion, leaving only the orientation of the fingers to analyze .Fig. 4.4a shows the upper half contour.

This output contour is still uneven in shape with very small distortions. Hence a polygon approximation of the contour is done to approximate the upper part of the hand as a polygon .This yields a contour derived of straight line segments making it easier in finding convexities .The result of polygon approximation is shown in Fig. 4.4b.

E. Finding Convexity Defects

Convexity defects are valley points. In particular, convexity defects are sequences of contour points between two consecutive contour vertices on the contour Hull. Fig. 4.5a , 4.5b , 4.5c , 4.5d , 4.5e shows the convexity defects in a palm , open thumb , V shaped fingers , a fist , and in three fingers .These convexity defects are very useful in providing information about the location and state of the fingers . In the figures below, the pure white markers are the convexity defects or the valley points and the dark markers are the start and end points respectively.

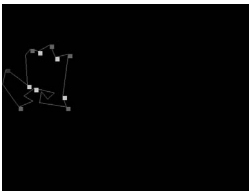


Fig. 4.5a. Palm

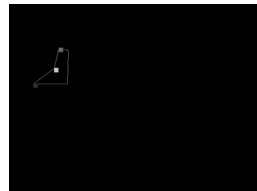


Fig. 4.5b. Thumb

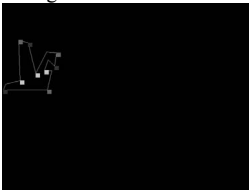


Fig. 4.5c. V Shaped Fingers

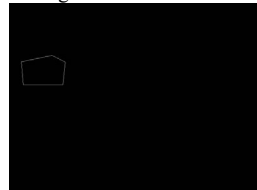


Fig. 4.5d. Fist

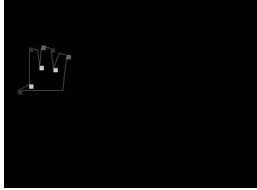


Fig. 4.5e. Three Fingers

F. Eliminating Convexity Defects

As shown in Fig. 4.6a not all the convexity defects provide useful information .Some defects are formed especially due to incomplete and distorted contours. Most of the defects which are on the underside of the contour are not related to the fingers. Every defect has a start and an end point .The start and the end points correspond to the coordinates where the valley begins and ends .For a defect to qualify as a defect caused by fingers, a condition is set that the Y-coordinate or the height of the defect is greater than the heights of both the start and end points by a minimum value of 20 units.

This condition does not cover the defects at the hand sides which have the height of the end point less than the height of the defect and height of the start point greater than the height of the defect .Such defects are generally the first and last defects in the list of defects which are sorted in increasing

order of their X-coordinates. Such defects are identified, but not removed .They are marked as special defects, as they are required at later stages.

A minimum bounding rectangle is a rectangle of minimum area which bounds the contour. The rectangle and coordinates of its vertices are calculated. In case the first defect starts with a special defect, then the horizontal distance from the special defect to its next defect (in the updated list of defects which has some unnecessary defects removed) is measured. In case the first defect is not a special defect , then the horizontal distance between the nearest vertical edge of the bounding rectangle and the defect is measured .After that, the distance between every two consecutive defects is measured .That is (defect2.xcoordinate - defect1.xcoordinate) , (defect3.xcoordinate - defect2.xcoordinate) and so on are measured until the last defect is reached .If the last defect is a not a special defect , then distance between last defect and nearest vertical side of bounding rectangle is measured .If it is a special defect , then it is ignored . Based on measured distances, a specific range is provided to approximate the number of fingers .In this case, with a distance greater than 75, number of fingers is approximated as 3 .With distance greater than 50 number of fingers approximated is 2 and a distance greater than 15 one finger is approximated .These distances are found to be nearly the same except for very small and very large hand contours. This is confirmed in later stages.

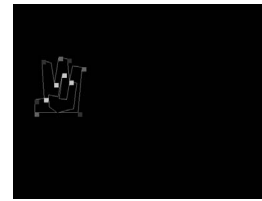


Fig. 4.6a. Unnecessary Convexity defects at the bottom

G. Contour Matching

After the finger classification in step 4.8 , the classified images are further confirmed using HU moment matching as follows .A Set of contours for each of the five gestures are stored in the memory at start up . Each set which contains four to seven contours differ from each other significantly when compared using HU moments .Depending on the number of fingers calculated, the contour polygon is compared to one set among the five sets of contours stored at startup. That is if two fingers were found out, it is first compared with the V shaped set of contours. If 5 fingers are found, then it is first compared with the Palm set of contours. Hu moments are said to be invariant to scale, translation and rotation .They are :

$$\begin{aligned}
 h1 &= (\eta_{(2,0)} + \eta_{(0,2)}) \\
 h2 &= (\eta_{(2,0)} - \eta_{(0,2)})^2 + 4(\eta_{(1,1)})^2 \\
 h3 &= (\eta_{(3,0)} - \eta_{(1,2)})^2 + (3\eta_{(2,1)} - \eta_{(0,3)})^2 \\
 h4 &= (\eta_{(3,0)} + \eta_{(1,2)})^2 + (\eta_{(2,1)} + \eta_{(0,3)})^2 \\
 h5 &= (\eta_{(3,0)} - 3\eta_{(1,2)})(\eta_{(3,0)} \\
 &\quad + \eta_{(1,2)})(\eta_{(3,0)} + \eta_{(1,2)})^2 \\
 &\quad - 3(\eta_{(2,1)} + \eta_{(0,3)})^2
 \end{aligned}$$

$$\begin{aligned}
 & + (3\eta_{(2,1)} - \eta_{(0,3)})(\eta_{(2,1)} + \eta_{(0,3)}) \left(3(\eta_{(3,0)} + \eta_{(1,2)})^2 \right. \\
 & \quad \left. - (\eta_{(2,1)} + \eta_{(0,3)})^2 \right) \\
 h6 = & (\eta_{(2,0)} - \eta_{(0,2)}) \left((\eta_{(3,0)} + \eta_{(1,2)})^2 \right. \\
 & \quad \left. - (\eta_{(2,1)} + \eta_{(0,3)})^2 \right) \\
 & + (4\eta_{(1,1)})(\eta_{(3,0)} \\
 & + \eta_{(1,2)}) \left((\eta_{(2,1)} + \eta_{(0,3)})^2 \right) \\
 h7 = & (3\eta_{(2,1)} - \eta_{(0,3)})(\eta_{(2,1)} \\
 & + \eta_{(0,3)}) \left((3\eta_{(3,0)} + \eta_{(1,2)})^2 \right. \\
 & \quad \left. - (\eta_{(2,1)} + \eta_{(0,3)})^2 \right) \\
 & - (\eta_{(3,0)} - \eta_{(1,2)})(\eta_{(2,1)} \\
 & + \eta_{(0,3)}) \left(3(\eta_{(3,0)} + \eta_{(1,2)})^2 \right. \\
 & \quad \left. - (\eta_{(2,1)} + \eta_{(0,3)})^2 \right)
 \end{aligned}$$

where $\eta_{(p,q)}$ are the normalized moments.

The measure of similarity is given by

$$S(A, B) = \sum_{i=1}^7 \left| \frac{1}{m_i^A} - \frac{1}{m_i^B} \right|$$

where

$$m_i^A = \text{sign}(h_i^A) \log|h_i^A| \text{ and } m_i^B = \text{sign}(h_i^B) \log|h_i^B|$$

where h_i^A and h_i^B are the Hu moments of Contours A and B respectively.

The closer the result $S(A, B)$ is to zero, the greater is the resemblance between the two contours. The posture is confirmed to be a recognized gesture only if at least 2 of the following three conditions are satisfied.

The minimum value of the match from all the contours in the set is less than 0.2.

Out of the all the contours in the set, at least three contours yield a match less than 0.3.

The average of the best three matches is less than 0.35.

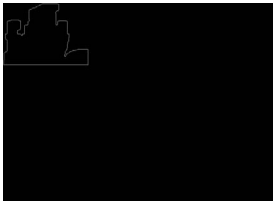


Fig. 4.7a: template1



Fig. 4.7b: template2

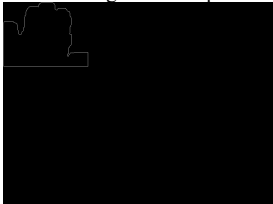


Fig. 4.7c: template3

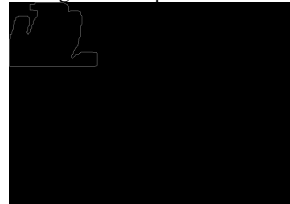


Fig. 4.7d: template4

If at least two conditions are satisfied, then output is confirmed. Else output is matched with next most probable contour set. If k be the number of fingers detected in the contour polygon then next probable contour sets are the sets having $k+1$ and $k-1$ fingers. In order to improve accuracy, the output is declared as a recognized gesture only after processing 5 continuous frames in which at least 3 frames

give the same recognized output. In cases where there is equal weightage for multiple gestures, for example 2-palm, 2-thumb, and 1 finger, the output is declared to be the previous output. Fig. 4.7a, b, c, d show the set of palm templates.

H. Experiments

Experiments on a set of five people with varying skin tone have resulted in the following outcomes. The following table illustrates the accuracies of the output of the gesture recognition module

Gesture Sequence	Accuracies (percentage)
Fist	95
Thumb	90
V shaped Fingers	88.23
Palm	85.4
Three Consecutive Fingers	89

The beagle board on running the same algorithm is measured to be 3.2 times slower than a 2.24 GHz Intel Dual Core Processor while the Beagle Board XM runs on a 1 GHz OMAP processor.

V. LINKING MODULE

This module is the follow up of gesture recognition. Once a gesture is recognized, several actions are executed on the projecting screen. These events are related to the pointer movement and mouse click events of the Beagle Board mouse.

There are two sub functionalities of the module

A. Pointer Movement via Hand Tracking

In the gesture recognition module, a contour analysis on each of the input frames resulted in identification of eligible hand contours in each of the frames. These hand contours were further verified using HU Moments. From the hand contour, the coordinates of the center of the hand are found out. This is done by assuming a bounding rectangle around the hand contour and finding out the midpoint of the bounding rectangle. This is the location of the hand.

A mapping function is assigned which maps the coordinates of the midpoint of the bounding rectangle to the screen size. Using this function, every point in the input frame is mapped to a point on the projecting screen. With the help of this mapping function, the desired location of the mouse pointer on the projecting screen is retrieved. Once the location of the hand changes between two successive frames, the location of the mouse pointer on the projecting screen should also change accordingly. This is achieved using the Linux X11 library. The X11 library is a low level C library for writing X clients. X clients are programs written in Linux. They are similar to any other shell scripting code snippets. These snippets use the X display as their interface. Xlib can draw simple solids, receive events, write text in various fonts and also do primitive inter process communication. The Xwarp pointer function which is an inbuilt Linux function is used to move the pointer to the desired location in the X Display. The function moves the

pointer by setting offsets relative to the current position of the pointer.

The following figures illustrate Pointer Movement via Hand Tracking. Consider two discreet time intervals t_1 and t_2 ($t_1 < t_2$). At each time instant the movement of the hand is tracked which is updated in the position of the mouse pointer.

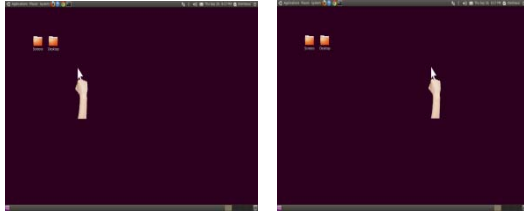


Fig. 5.1a. At time t_1 Fig. 5.1b. At time t_2

B. Mouse Clicks Via Gesture Recognition and Permissible States

Depending on the output gesture recognized, several states and functionalities have been defined for the entire system. Three gestures have been selected and assigned particular tasks. The fist, the palm and the V shaped Fingers. To achieve these functionalities, the Xlib is used again. There are three basic types of pointer events in the X-Library. They are Button press, Button release, Motion Notify. The motion Notify event is generated by the system whenever the pointer is moved and when the pointer motion begins and ends in a window. The Button press event is used to signify the left and right clicks and the Button release event is used to signify the left and right click releases. The different functions used to perform the above functionalities are XCloseDisplay, XFlush, XWarpPointer. The table below shows the permissible states for an event such as mouse clicks to occur. The four permissible states in the system are listed in the table below.

Gesture Sequence	Action
Fist -> Palm -> Fist	Left Click
Fist -> V Shaped Fingers -> Fist	Right Click
Fist	No Action, Only Hand Tracking
Fist -> Palm (Extended period) -> Fist	Double Click

The following figures show the single click, Right Click and the Double Click.



Fig. 5.2a. Single Click



Fig. 5.2b. Right Click

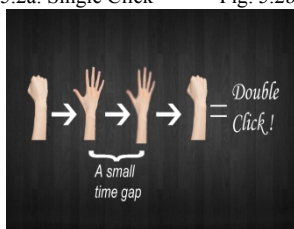


Fig. 5.2c. Double Click

VI. CAMERA CALIBRATION MODULE

The camera calibration module is pivotal for proper functioning of the entire system.

Camera calibration is a necessary step in 3D computer vision in order to extract metric information from 2D images. Improper calibration in the proposed system could mean that the pointer would not move in accordance to the movement of the hand as the distance covered by the hand and the distance covered by the pointer are not the same. Hence the accuracy of the entire system would falter with improper calibration techniques. Using a very simple camera calibration technique it is possible to generate a separate parameter file for the specific camera(s) that is being used. The procedure is as follows:

- A checkerboard pattern is displayed on the screen and camera is focused at the grid to capture a series of images.
- Positions of pattern corners are known with respect to a co-ordinate system. (The Resolution of Desktop screen can be used as a measure).
- Images Corners are obtained from the captured Images.
- Obtain Equations that describe imaging and contain internal parameters of camera.

The captured images from the camera are processed to obtain the location of the corners. This can be done using an openCV code which uses color segmentation to know the position of square. This process is repeated for all the four corners through which the borders of the screen are obtained.

Then, two horizontally aligned squares are recognized to obtain the distance between them. This can also be done using the above technique of color segmentation. The same process is repeated for two vertically aligned squares to obtain the distance between them. The results now obtained from the images are according to the calibration settings of the camera. Having known the desktop resolution, the distances on the desktop are mapped to the distances obtained from the images. A linear function is obtained which describes the relation between the distances. According to these results and the linear function, the calibration settings of the camera are changed to minimize the calibration error. A good calibration technique reduces the error to 1 pixel.

VII. CONCLUSION AND FUTURE WORK

An approach based on hand contour convexities and Hu moment matching has been proposed. A method of having different hand gesture templates at the system start up and finding the best match has been presented. A practical application of an interactive touch wall has been presented. The implementation on the Beagle Board shows that the system is suitable in real time even with an embedded device of lower processing power compared to the PC. Any simple USB camera can be connected with the PC or the Beagle Board with no limitations. Future work includes trying to further improve the accuracy of the work and study and compare other techniques on Hand Gesture Recognition which have been proposed so far.

REFERENCES

- [1] A. S. Pentland, *Computer Vision For Human-Machine Interaction*; eds. Cambridge University Press, ch1, pp.3-21, 1998
- [2] T. F. William and D. W. Craig, "Television Control by Hand Gestures," in *IEEE Intl. Workshop on Automatic Face and Gesture Recognition, Zurich*, June 1995.
- [3] S. Ju, M. Black, S. Minneman, D. Kimber, "Analysis of Gesture and Action in Technical Talks for Video Indexing," in *IEEE Conf. on Computer Vision and Pattern Recognition, CVPR 97*
- [4] A. VanDam, "Post-WIMP user interfaces," in *Communications of the ACM*, vol 40, pp. 63-67, February 1997
- [5] A. Bobick and A. Wilson, "A state-based technique for the summarization and recognition of gesture," in *Proc. IEEE Fifth Int. Conf. on Computer Vision, Cambridge*, pp. 382-388, 1995.
- [6] E. Hunter, J. Schlenzig, and R. Jain. "Posture Estimation in Reduced-Model Gesture Input Systems," in *Proc. Int'l Workshop Automatic Face and Gesture Recognition*, pp. 296-301, 1995.
- [7] C. Maggioni. Gesturecomputer. "New Ways of Operating a Computer," in *Proc. Int'l Workshop Automatic Face and Gesture Recognition*, 1995.
- [8] C. Manresa , J. Varona , R. Mas, and F. J. Perales "Real -Time Hand Tracking and Gesture Recognition For Human -Computer Interaction," in *Electronic Letters on Computer Vision and Image Analysis* 0(0), pp.1-7, 2000.
- [9] E. Sanchez-Nielsen, L. Anton-Canalis, and M. Hernandez-Tejera "Hand Gesture Recognition For Human-Machine Interaction," *Journal of WSCG*, vol. 12, no. 1-3 , ISSN 1213-6972.
- [10] Wikipedia(2011). Beagle Board. Wikipedia.org Available :<http://en.wikipedia.org/wiki/BeagleBoard>
- [11] elinux.org Available :<http://elinux.org/BeagleBoardBeginners>
- [12] Ubuntu. elinux.org Available :<http://elinux.org/BeagleBoardUbuntu>
- [13] Wikipedia(2011) Opencv Wikipedia.org Available:<http://en.wikipedia.org/wiki/Opencv>
- [14] G. Bradski and A. Kaehler, "*Learning OpenCV*", Sep. 2008.
- [15] Willowgarage Opencv Available :<http://opencv.willowgarage.com/wiki/>
- [16] S. Malik, Real-time Hand Tracking and Finger Tracking for Interaction, Project Report, Dept Computer Science ,University of Toronto