Hand Tracking and Hand Gesture Recognition for Human Computer Interaction

Dejan Chandra Gope
dejan.gope23@gmail.com
Department of Computer Science and Engineering
Dhaka University of Engineering and Technology (DUET)
Gazipur-1700, Dhaka, Bangladesh.

ABSTRACT: Hand tracking and hand gesture recognition is an important problem in the field of human-computer interaction. A number of solutions have been proposed in the current literature, but the problem is still far from being solved since the hand exhibits significant amounts of articulation and self-occlusion that cause difficulties with existing algorithms. To further exasperate these problems, interactive applications require that the hand tracking perform in real-time. The current ubiquity of webcams offers an opportunity to create computer vision systems which can enable novel new methods for human-computer interaction. To that end, we present a system which allows the user to control the operating system cursor in a hands-free way by gesturing in mid-air. Our system leverages OpenCV and the X windowing system to track the index finger and thumb of a user using a webcam. The user can motion in the direction she/he wishes the cursor to move, and can execute mouse operations by "pointing" toward the camera and breaking a "plane of interaction". Our goal with this work is to demonstrate a proof-of-concept system for enabling a new method of controlling and interfacing with a computer using commodity webcams.

Keywords: Hand Gesture Recognition, Hand Pointing, Pointing Accuracy, Computer Vision, Fingertip Interaction, Human Computer Interaction(HCI), Hand-posture Recognition, Hand Gesture Pose Estimation.

1 INTRODUCTION

Vision-based hand gesture recognition is an active area of research in human-computer interaction (HCI), as direct use of hands is a natural means for humans to communicate with each other and more recently, with devices in intelligent environments. The trend in HCI is moving towards real-time hand gesture recognition and tracking for use in interacting with video games, remote-less control of television sets, and interacting with other similar environments. Given the ubiquity of mobile devices such as smart phones and notebooks with embedded cameras, a hand gesture recognition system can serve as an important way of using these camera-enabled devices to interact more intuitively than traditional interfaces. This paper presents the implementation and analysis of a vision-based static hand pose estimation technique. The system uses a single low-resolution camera (640x480, 1067x800 - mobile phone and notebook web cameras) under uniform lighting conditions, and estimates the 2D orientation or the pose of the hand gesture. In interactive applications such as volume control of a music player, a 2D navigation game, multimedia browsing, the orientation of a single pointing gesture or a directed palm can be used to manipulate the various levels of output.

2 RELATED WORK

Hand gesture recognition and tracking has been an important and active area of research in the field of HCI, and sign language recognition. The use of glove-based devices to measure

hand location and shape, especially for virtual reality, has been actively studied. In spite of achieving high accuracy and speed in measuring hand postures, this approach is not suitable for certain applications due to the restricted hand motion caused by the attached cables. Computer vision techniques measure hand postures and locations from a distance, providing for unrestricted movement. Numerous approaches have been explored by the vision community to extract human skin regions either by background subtraction or skin-color segmentation. Methods based on background subtraction are not feasible when applied to image with complex Vision-Based Hand Gesture Pose Estimation for Mobile Devices backgrounds or real-world scenarios where the user wants to use the application on-the-go. Once the image regions are identified by the system, the image regions can be analyzed to estimate the hand posture. Specifically, for finger gesture recognition and tracking, a common approach is to extract hand regions and then locate the fingertip to determine the pose orientation. In a 3D pointing interface using image processing is presented to estimate the pose of a pointing finger gesture. This system however, suffers from various drawbacks in real-world scenarios due to the use of a fixed threshold for image binarization and the use of pre-determined finger length and thickness values. Also, low-cost web cameras and infrared cameras have been used for finger detection and tracking. In finger detection is performed by fitting a cone to rounded features, and in a template matching approach is used to recognize a small set of gestures.

3 SYSTEM FLOW

The system order is shown in Figure 1. Initially, when we get an image from the camera, we convert the color space RGB to YCbCr. Then, we define a range of colors as 'skin color' and convert these pixels to white; all other pixels are converted to black. Then, we compute the centroid of the dorsal region of the hand. After we identify the hand, we find the circle that best fit this region and multiply the radius of this circle by some value to obtain the maximum extent of a 'non-finger region'. From the binary image of the hand, we get vertices of the convex hull of each finger. From the vertex and center distance, we obtain the positions of the active fingers. Then by extending any one vertex, we control the mouse movement.

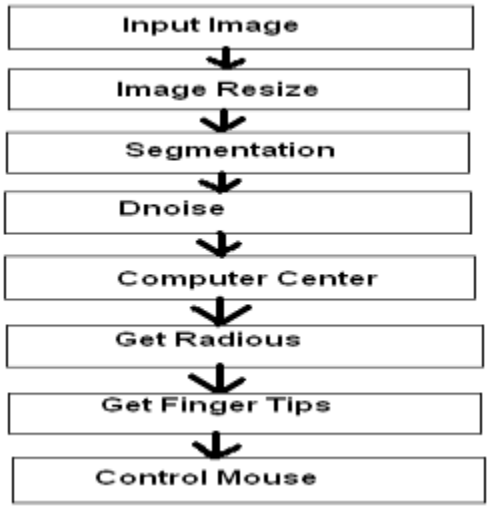
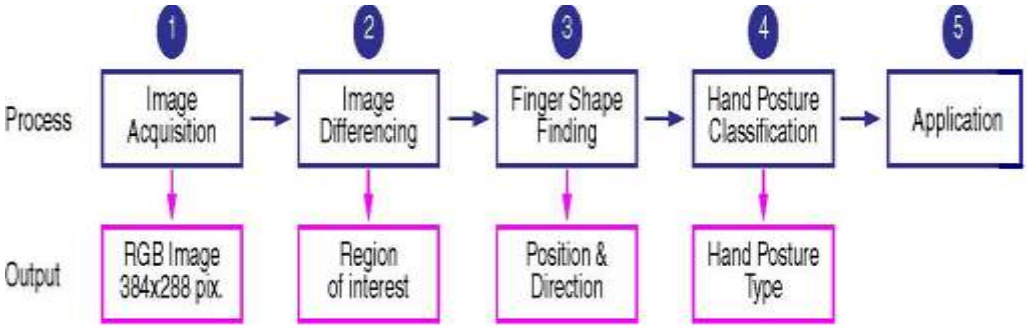


Figure-1: An overview of our hand gesture recognition and mouse control system. Input images converted to binary image to separate hand from background. Center of hand and computed calculated radius of hand found. Finger tips points using Convex Hull algorithm. Mouse controlled using hand gesture.

4 HAND AND FINGER DETECTION

Human hand and finger detection is an important role in human-computer interface. They usually function through keyboard, mouse, digitizer, touch panel, and so on. There are many related works of fingertrip interaction with large screen display which utilize free hand/finger tracking: interface in VR environment , television control and finger pointer. In this paper, we describe barehanded interaction between human and large screen display for Hand and Finger Detection. Barehanded means that no device and no wires are attached to the user, who controls the computer directly with the movements of his/her hand. We identify three essential services for fingertip interaction with large screen display detection, identification



and tracking. We will briefly present the three services and describe how they are used by our envisaged applications.

Figure-2: The Hand Posture Recognition and Finger Finding Process.

4.1 Hand Tracking

This section describes the implementation details of the hand tracking system, which is primarily based on the single hand tracker. The system can extract the 3D position and 2D orientation of the index finger for each hand, and when present the pose of the thumb as well. In interactive applications a single pointing gesture could then be used for selection operations while both the thumb and index finger could be used together for pinching gestures in order to grasp and manipulate virtual objects.

4.2 Background Substraction

The first phase of the tracking system involves separating potential hand pixels from non-hand pixels. Before segmentation occurs, we first convolve all captured images with a 5x5 Gaussian filter and then scale this filtered image by one half in each dimension in order to reduce noisy pixel data. All subsequent image processing operations are then performed on this scaled and filtered image. Since the stereo cameras are mounted above a non-moving workspace, a simple background subtraction scheme is used to segment any potential foreground hand information from the non-changing background scene.

4.3 Skin Segmentation

Although the background subtraction scheme described about works fairly well in segmenting foreground data from the non-changing background, it will still allow objects such as shirt sleeves, coffee mugs, or other desktop items that are placed into the workspace to be detected as potential hands. In order to deal with such situations and add some more flexibility to the system, a skin pixel detector has been implemented to further filter the foreground data. As a pre-processing step, for each camera a small number of snapshots are taken of various hands

with a range of different skin-tones and poses. Then using an image editing program each of the captured images is manually segmented into a binary mask where white pixels represent skin areas and black pixels represent non-skin areas. This set of captured images and associated skin masks is then used as the training set for a histogram-based skin classifier.

4.3 Region Extraction

Now that the skin regions have been detected, we must determine which regions correspond to the left and right hands. It is possible that small noisy regions will still be present after background subtraction and skin segmentation, but we assume that the regions corresponding to the hands will be the largest. Thus we first extract the contours of all the detected skin regions using binary image processing operations and connected component analysis.

4.4 Feature Extraction

In order to find the fingertips for the thumb and index finger for each hand, we attempt to find pixels that represent peaks along the contour perimeters. The points as either peaks or valleys, we convert the vectors into 3D vectors lying in the xy-plane and then compute the cross product. If the sign of the z component of the cross product is positive then we label the point as a peak, while a negative cross product results in a valley label. Finally, non-maximal suppression is then used to find the strongest peaks and valleys along the perimeter, since we can expect that a sequential set of peaks and valleys will be detected in the neighborhood of the strongest locations.

4.5 Hand Gestures Recognition

Hand gesture recognition system to recognize real time gesture in unconstrained environments. Our gesture alphabet consists in four hand gestures and four hand directions in order to fulfil the application's requirements. The hand gestures correspond to a fully opened hand (with separated fingers), an opened hand with fingers together, a fist and the last gesture appears when the hand is not visible, in part or completely, in the camera's field of view. These gestures are defined as Start, Move, Stop and the No-Hand gesture respectively. Also, when the user is in the Move gesture, he can carry out a Left, Right, Front and Back movements. For the Left and Right movements, the user will rotate his wrist to the left or right. For the Front and Back movements, the hand will get closer to or further of the camera. Finally, the valid hand gesture transitions that the user can carry out are defined.

4.6 Finger Detection

The tracking program allowed the user to define the workspace, move the "mouse" around within the workspace and have the corresponding movement echoed within the application program, and to select screen objects by "clicking" on them. The user's workspace needed to be defined so that accurate relative position could be mapped to the screen. The user set up the workspace at the beginning of the session by moving the forefinger to the upper-left and lower-right hand corners of the workspace, and performing the clicking action. This provided the reference coordinates for the workspace area. After setting up the workspace, the user could move the mouse cursor around the application program by moving their finger within the defined workspace, and select objects by performing the click action. Double-clicking was achieved by repeating the click action within a short time. The tracking system was set up as a server on VxWorks and passed event calls to application programs as they occurred. This is similar to the way a mouse work. Changes in the position of the finger were sent by the server using Xevents to the application programs. When "clicks" and "double-clicks" were detected,

the change in state was also passed to the application. The Finger Tracker interfaces was trialed with programs such as xv and xfig. This showed that the finger could be tracked in real time with cursor movement on the screen providing feedback to the user. This was sufficient to test the robustness of the tracking system.

Figure-3: Finger Tracking

5 MOVING MOUSE CURSOR

We used the index finger as a cursor controller to control mouse cursor. We used two different approaches for moving the mouse cursor. The first method is mapping cursor control. It means that the index finger on a camera screen can position maps to a desktop screen position. In other words, the mouse cursor is placed on the desktop window along with the index finger tips position displayed on the camera screen position. This method has a problem. If the resolution of the desktop window has a higher resolution than the camera resolution, then the cursor position cannot be accurate because when the camera resolution converts to the desktop window resolution we lose intermediate value. We expect the ratio of jumping pixel is up to 4 pixels. The second method is weighted speed cursor control. We get a difference of the finger of the current image and the previous image and compute the distance between the two. Next, we move the mouse cursor if the gap between the two finger images (current and previous frame) is far then the mouse cursor moves fast or, if the gap is close then the cursor moves slow. This algorithm also has a problem. Since we are working with real-time image processing, the client machine should be fast. The image input speed is 15 frames per second and we need image processing time on CPU. It means that some machines which cannot achieve image processing 15 images per sec do not work smoothly because computing the image center and the hand shape takes time. Thus, this algorithm does not work properly. In this paper, we used the first method which uses absolute position of finger tips because it is more accurate than the second method.

5.1 Left Clicking and Double-Clicking

To call system event for left clicking, at least two convex hull vertexes have to be off the palm area which was computed in the previous part. In addition, the x position of one of two vertexes should be lower than the other to restrict detection of other finger tips. When the degree of the index finger and thumb is 70 to 90 degree then we can recognize that the gesture is left clicking. Actually, if the thumb is placed outside the circle of the hand region, then the gesture is left clicking. The double-clicking occurs when the thumb moves 0 to 90 degree and back two times fast.

5.2 Right Clicking

We simply implemented this part using previous gestures. If we make the hand pose left clicking for 3 seconds, then the system calls the right clicking event.

5.3 Scrolling

Scrolling is a very useful task with hand gestures. We also implemented this feature in our system. We assigned a 10% margin as a scroll area on the right side of the desktop screen region and divided by half. If the index finger placed the upper area of the scroll area with clicking pose then the system called the scroll up function or else the system called the scroll down function.

6 EXPERIMENTS AND RESULTS

We tested all mouse tasks such that left click, right click, double-click, dragging, and scrolling on windows. We could not compare with the mouse device because this hand gesture system always shows lower performance than real mouse device. Instead of comparing with a real mouse, we allowed to use this system to four testers to know how it can be adapted easily.

Our method for evaluating performance is that we checked the time that a task, such as clicking, double clicking and scrolling, is done. We designed four experiments to get performance. In the first experiment we placed an icon on the center of desktop window and put the cursor in the top-left corner. We then measured the time in how long it takes to select the icon. In the second experiment the position of the icons is same and we only measured the time to show the drop down menu on the icon. The results are shown below: (time = sec) every tester was new in this system. The clicking task showed similar times with all testers. However, the scrolling result showed the time as unstable. The reason is that the focus of the camera lens works automatically so when the index finger went to the border on the screen then part of the hand colors becomes dim. Eventually, hand segmentation failed because colors could not be on the color range which is in the segmentation part. Thus, we could not get a clear shape of the hand from the video streaming.

7 CONCLUSION AND FUTURE WORK

This project presented a vision-based hand tracking system that does not require any special markers or gloves and can operate in real time on a commodity PC with low-cost cameras. Specifically, the system can track the tip positions of the thumb and index finger for each hand, assuming that a calibrated pair of cameras is viewing the hands from above with the palms facing downward. The motivation for this hand tracker was a desktop-based two-handed interaction system in which a user can select and manipulate 3D geometry in real-time using natural hand motions. The algorithmic details for the hand tracker were presented,

followed by a discussion of the performance and accuracy of the system, as well as a discussion of how the system could be improved in the future.

References

Computer vision based mouse, A. Erdem, E. Yardimci, Y. Atalay, V. Cetin, A. E. Acoustics, Speech, and Signal Processing, 2002. Proceedings. (ICASS).

Virtual mouse vision based interface, Robertson P., Laddaga R., Van Kleek M. 2004.

Hailing Zhou, Lijun Xie, Xuliang Fang, *Visual Mouse: SIFT Detection and PCA Recognition*, cisw, pp.263- 266, 2007 International Conference on Computational Intelligence and Security Workshops (CISW 2007), 2007.

James M. Rehg, Takeo Kanade, DigitEyes: *Vision-Based Hand Tracking for Human-Computer Interaction in, Proc. of the IEEE Workshop on Motion of Non-Rigid and Articulated Objects,* Austin, Texas, November 1994, pages 16-22.

Gary Bradski, Adrian Kaehler, *Learning OpenCV*, O'Reilly Media, 2008.

Asanterabi Malima, Erol Ozgur, and Mujdat Cetin, *A Fast Algorithm for Vision-Based Hand Gesture Recognition for Robot Control.*

J. Serra, *Image Analysis and Mathematical Morphology*, New York: Academic Press, 1983.

K. Homma and E.-I. Takenaka, "*An image processing method for feature extraction of space-occupying lesions,*" journal of Nuclear Medicine 26 (1985): 1472-1477.

Cheng, K. Vronay, D. and Yu, F. "*Interaction Design for the Media PC," in Proceedings of Companion proc*. UIST'04, Santa Fe, New Mexico, ACM Press. 2004, pp. 5-6.

Colombo, C. Bimbo, A. D. and Valli, A. "*Visual Capture and Understanding of Hand Pointing Actions in a 3-D Environment*," IEEE Transactions on Systems, Man, and Cybernetics, 33(4), University of Florence, Italy, pp. 677-686, August 2003.

Dix, A. Finlay, J. Abowd, G. D. and Beale, R. *Human-Computer Interaction*, 3rd ed.Pearson Education Limited, 2004.

Baudel, T. and Beaudouin-Lafon, M. 1993. Charade*: remote control of objects using free-hand gestures.* Commun. ACM 36, 7 (Jul. 1993), 28-35.

Grossman, T. Wigdor, D. and Balakrishnan, R. 2005. *Multi-finger gestural interaction with 3D volumetric displays*. ACM Trans. Graph. 24, 3 (Jul. 2005), 931-931.

Dejan Chandra Gope is a student in the Department of Computer Science and Engineering at the University of Dhaka University of Engineering and Technology, Bangladesh. He received Bachelor of Science in Computer Science and Engineering (B.Sc) from the University of Dhaka University of Engineering and Technology in 2012. He is now studying Master of Science in Computer Science and Engineering (M.Sc) at the same University. His researchinterests lie in studying the role of computer vision and advanced human-computer interaction.