Real-time Finger Tracking for Interaction

Noor Shaker
Damascus University
Faculty of Information Technology
email: noor.shaker@gmail.com

M. Abou Zliekha
Damascus University
Faculty of Information Technology
email: mhd-it@scs-net.org

Abstract

In this work, we describe an approach for human finger motion and gesture detection using two cameras. The target of pointing on a flat monitor or screen is identified using image processing and line intersection. This is accomplished by processing above and side images of the hand.

The system is able to track the finger movement without building the 3D model of the hand.

Coordinates and movement of the finger in a live video feed can be taken to become the coordinates and movement of the mouse pointer for human-computer interaction purpose.

1. Introduction

In the past few years high technology has become more progressed and less expensive.
With the availability of high speed processors and inexpensive webcams, more and more people have become interested in real time applications that involve image processing.

Vision-based hand tracking is an important problem in the field of human-computer interaction, since hand motions and gestures could potentially be used to interact with computers in more natural ways. A number of solutions have been proposed in the current literature, but the problem is still far from being solved since interactive applications require that the hand tracking is performed in real-time.

This work presents the implementation and analysis of a real-time index finger tracking system that can be used for interaction purposes. The system uses two low-cost web cameras one mounted above the work area and facing downward and the other on the side of the working area. In real-time, the system can track the orientation of the index finger without the use of special markers or gloves and without building the 3D model of the hand.

2.Related Work

With the growth of attention about computer vision, the interest in human-computer interaction has increased proportionally.

Human features and monitoring devices were used to achieve Human-computer interaction, but during our research we were interested only in works that involved the use of the index finger of the hand.

Previous work in the visual analysis of gestures may be categorized according to many criteria. These include the modeling used (eg kinematic, statistical, template based, stick figures, blobs), sensor modality (infra-red, visible light, range), tracking space dimensionality (2D or 3D), number of sensors, sensor mobility and sensor placement. Explicit shape models are widely used in 3D tracking, as kinematic constraints and known target types facilitate the quantification and analysis of three dimensional movements [3].

From an interaction perspective, most of the hand tracking work to date has focused on 2D interfaces. In [8], a finger was tracked across a planar region using low-cost web cameras in order to manipulate a traditional graphical interface without a mouse or keyboard.

Similarly, in [9] infrared cameras were used to segment skin regions from background pixels in order to track two hands for interaction on a 2D tabletop display. Their method then used a template matching approach in order to recognize a small set of gestures that could be interpreted as interface commands. However, no precise fingertip position information was obtained using their technique.

Recently, some studies of integrating various cues such as motion, shading and edges were proposed.

3.System Overview

The following sections describe the implementation details of the finger tracking and gesture recognition system, which is primarily based on real time hand tracking presented in [6]. The system can extract the 3D position and 2D orientation for the right hand index finger.

4.System Structure

In order to detect the orientation of the index finger for the hand we need to do some preprocessing operation on each image.

4.1. Background Subtraction

The first phase of the tracking system involves separating potential hand pixels from nonhand pixels.

Since the 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.

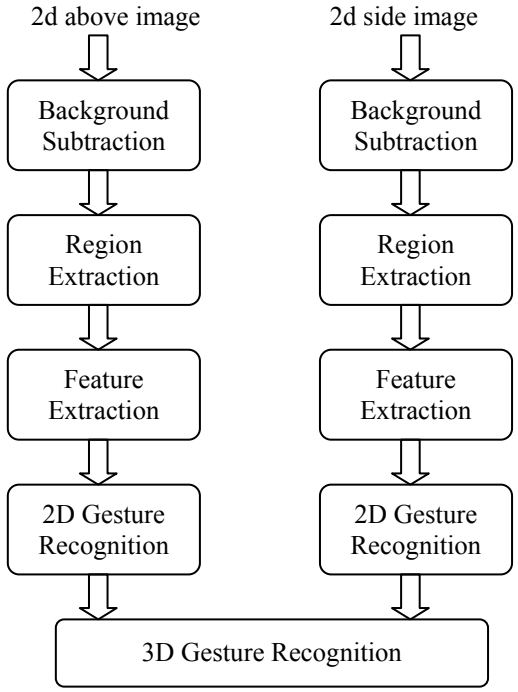


Figure 1: System overview

4.1.1. Grayscale Filter. We convert all captured images into grayscale images.


Figure 2: Grayscale image

4.1.2. Threshold. For each pixel in frame *i*, we compute the foreground mask image If (for each camera) as:

$$If = \begin{cases} 255 \text{ if } |Ii - Ib| > \mu b \\ 0 \quad \text{otherwise} \end{cases}$$

where μb is a fixed threshold to differentiate foreground data from background data. The resulting If image is a binary image.

4.2.Region Extraction

Now that the skin region has been detected, we must determine which regions correspond to the hand. It is possible that small noisy regions will still be present after background subtraction, but we assume that the region corresponding to the hand will be the largest.


Figure 3: Binary image

4.2.1. Blurring. Since our attention will be at the orientation of the index finger and we are not concerning in the exact shape of the hand, we will apply a blurring filter on each captured frame. This process will close small holes and smooth the edge of the hand witch make it easer for incoming operations.


Figure 4: Blurred image

After applying the blurring filter we apply a threshold operation on each frame to get binary images representing the hand region.


Figure 5: Hand's region image

4.2.2. Edge Detection. We apply edge detection technique (Laplacian filter then image subtraction) to get the edged image of the hand

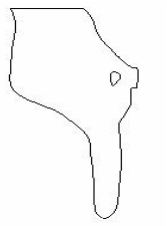


Figure 6: Edged image

4.2.3. Contour. After we get the edged image we can easily get the contour of the hand by scanning the

images and following the edges using Rosenfeld and Kak algorithm for edge detection.

All the following mathematical operations will be done on the contour image.

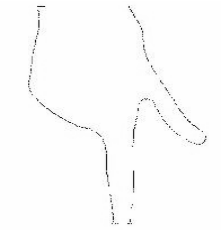


Figure 7: Contour image

4.3.Feature Extraction

4.3.1. Peck Detection. We attempt to find pixels that represent peaks along the contour.

All the previous processing steps are exactly the same for both above and side images, but in this stage we have two strategies

1. Above Image: For these images we will assume that the finger peak will be at the pixel which has the most value on the y axes (vertical axes).

Applying this strategy may yield to more than one pixel have the same y value, the algorithm returns the first one and that may not be the correct peak. Incorrect peak detection may also happen if we get the pixel with the most y value, but it is still may be incorrect because of the pervious processing steps we have done till now.

And since the correct detection of the peak is important for correct gesture recognition we should improve our algorithm, for this reason we will take all the pixels which has the same y value and some of the surrounding pixels on the edge according to their distance of the most y-value and then get the average y value for all these pixels and we assume that the x value is in the middle of the finger.

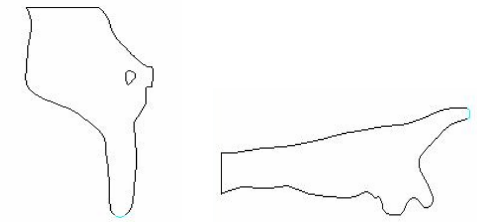


Figure 8: Above and side image peak detection

2. Side Image: We will apply the same strategy for the above images except that we will take the pixels that have the most value on the x axes (horizontal axes). Then we will approximate these pixels to get a better estimation of the peak.

4.2.2D Gesture Recognition

In order to get the 3D orientation of the finger we will divide the problem into two simple sub problems, we will get one image from above of the finger, and another one form the side. Then we calculate the orientation from each image alone, this gives us two orientations: one that represents the horizontal orientation which we get from the above image, and one representing the vertical orientation which we get from the side image.

Combining these two orientations gives us the 3D orientation of the finger.

4.3.2. Finger's Line. We will represent the orientation of the finger as a line.

To get this line we should have two points, one of them we have already has which is the fingers peak.

For simplicity we will assume that the figure has an average length and width.

We have two cases:

1. Above Image: We take the two pixels (p1,p2) that position at the middle of the finger by taking the middle value on the y axes of the finger contour, then the point will be positioned at

$$p(x,y) = ((p1(x) + p2(x))/2, p1(y))$$

Now we have two points, we can calculate the equation of the finger line as:

$$y - y1 = tg\Theta(x - x1)$$

Where $(x1, y1)$ is the position of p1, and $tg\Theta$ is the tilt of the finger from a vertical axes

$$tg\Theta = (y - y1)/(x - x1)$$

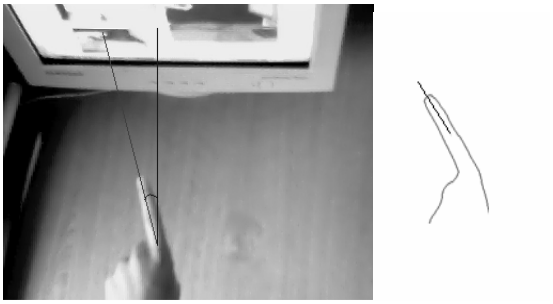


Figure 9: Above image orientation

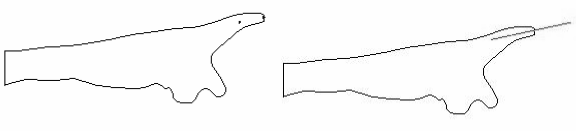
2. Side Image: We can apply the same technique we use for the above image to get the line that represents the orientation of the finger in the side image, the difference is in the points we choose to get the line equation:

The first point is the peak and the second point is taking from two points (p1,p2) that position at the middle of the finger on the x axes

$$p(x,y) = (p1(x), (p1(y) + p2(y))/2)$$

And the line equation will be the same.

$$y - y1 = tg\Theta(x - x1)$$



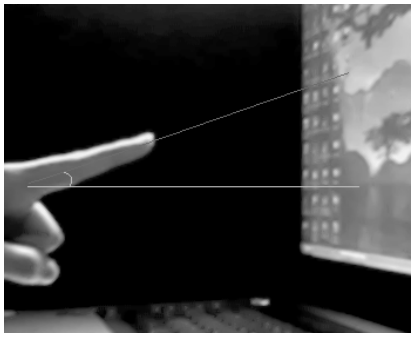


Figure 10: Side image orientation

4.3.3. Finger's 2D Orientation Detection.

After we get the finger line from both images it becomes simple to get the orientation for the finger in each image, then extracting the 3D orientation.

The orientation from both images is the same as the tilt of the finger line.

Since the line equation we get is:

$$y - y1 = tg\Theta(x - x1)$$

The orientation of the line will be: $tg\Theta$

4.4.3D Gesture Recognition

The final step is to extract the 3D orientation from two 2D orientations.

The above image gives the orientation on the x-axes, while the side image gives the y-axes orientation. Combining these two orientations we can conclude the 3D orientation.

4.5.Screen Intersection

After we get the 3D orientation of the finger we should get its intersection with the screen. The position of the intersection point depends on the finger's orientation and the distance between the hand and the screen.

Since we have the line equation that represents the finger's orientation, we can calculate the intersection as follows:

From the triangles (abc),(bde) in figure (11) we get :

$$tg\Theta = (d(y) - b(y))/(d(x) - b(x))$$

$$tilt = tg\Theta$$

$$\Theta = a\tan(tilt)$$

$$ac = bc * \sin\Theta$$

We can apply this technique to both above and side images the same way.

4.6.Finger Tracking

The tracking process is established through applying the above mentioned operations on each captured frame and calculating the angle for each finger's line in each frame, then combining the resulting 2D orientations to get the 3D one.

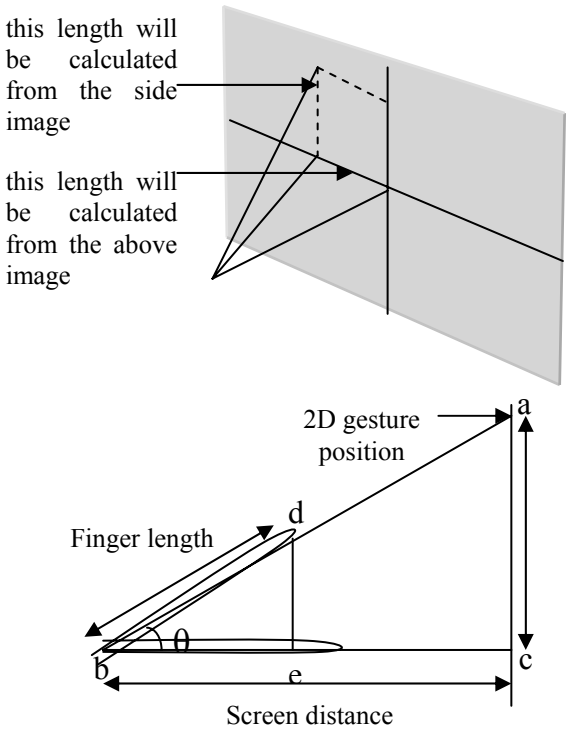


Figure11: 3D orientation with screen intersection

5.Results

The program was tested by different people and the results were very promising; the following conclusions were made based on the experiments that were held

In detection mode the finger's peak was detected accurately when the hand is not rotated more than 45° around the y-axes –the vertical axes- for the above images, and not rotated more than 45° around the x-axes –horizontal axes- for side images.

In tracking mode the results were very robust when the frame rate is 20 fps and higher.

For the detection and tracking to be accurate and robust the lighting conditions must be set so the light is frontal in a way that it will spread evenly on the working area, because side or above light will cause false hand region detection and will effect the tracking process.

6.References

[1] M.S. Lee, D.Weinshall, E. Cohen Solal, A. Colmenarez and D. Lyons, "A computer vision system for on-screen item selection by finger pointing", Computer Vision and Pattern Recognition CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on , 2001.

[2] Jakub Segen and Senthil Kumar, "Shadow Gestures: 3D Hand pose estimation using a single camera", In Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 1999. Vol. 1, pp. 479-485.

[3] Robert Virtue, “3D Shadows: Computer Vision for an Unencumbered Interface”, Proceedings of Human Interface Technologies Conference, 2004.

[4] J. J. Kuch and T. S. Huang. “Virtual Gun: A Vision Based Human Computer Interface Using the Human Hand”. Proc. IAPR Workshop on Machine Vision Applications MVA'94, pp 196-199. Tokyo, December 1994.

[5] Carlo Colombo, Alberto Del Bimbo, and Alessandro Valli, “Visual Capture and Understanding of Hand Pointing Actions in a 3-D Environment”, IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART B: CYBERNETICS, VOL. 33, NO. 4, AUGUST 2003.

[6] Shahzad Malik, “Real-time Hand Tracking and Finger Tracking for Interaction”, CSC2503F Project Report, December 18, 2003.

[7] Paul Withagen, Klamer Schutte and Frans Groen, “LIKELIHOOD-BASED OBJECT DETECTION AND OBJECT TRACKING USING COLOR HISTOGRAMS AND EM”, IEEE ICIP 2002.

[8] Z. Zhang, Y. Wu, Y. Shan, S. Shafer, “Visual panel: Virtual mouse keyboard and 3d controller with an ordinary piece of paper”, In Proceedings of Perceptual User Interfaces, 2001.

[9] Y. Sato, Y. Kobayashi, H. Koike, “Fast tracking of hands and fingertips in infrared images for augmented desk interface”, In Proceedings of IEEE International Conference on Automatic Face and Gesture Recognition (FG), 2000. pp. 462-467.