Efficient Fingertip Tracking and Mouse Pointer Control for a Human Mouse

Jiyoung Park and Juneho Yi

School of Information and Communication Engineering
Sungkyunkwan University
Suwon 440-746, Korea
{jiyp, jhyi}@ece.skku.ac.kr

Abstract. This paper discusses the design of a working system that visually recognizes hand gestures for the control of a window based user interface. We present a method for tracking the fingertip of the index finger using a single camera. Our method is based on CAMSHIFT algorithm and it tracks well particular hand poses used in the system in complex backgrounds. We describe how the location of the fingertip is mapped to a location on the monitor, and how it is both necessary and possible to smooth the path of the fingertip location using a physical model of a mouse pointer. Our method is able to track in real time, yet does not absorb a major share of computational resources. The performance of our system shows a great promise that we will be able to use this methodology to control computers in near future.

1 Introduction

Computer vision has a significant role to play in the human-computer interaction devices of the future. There have been reported many research results [1-5, 7, 8] in the literature that try to substitute the currently used devices such as mouse and keyboard with a vision based natural interface. Rehg and Kanade described "DigitEyes"[4], a model-based hand tracking system that uses two cameras to recover the state of a hand. Kjeldsen and Kender suggested a simple visual gesture system to manipulate graphic user interface of a computer [5]. Andrew et al. presented a method for tracking the 3D position of a finger, using a single camera placed several meters away from the user [7]. DigitEyes is based on a 3D model of a human hand and is computationally very expensive. The performance of the other two approaches is not robust to complex background.

This research presents a system that efficiently tracks the fingertip of the index finger for the purpose of application to a human mouse. The main features of the proposed system are as follows. First, adaptive online training of skin color distribution is employed to reliably detect human hand regions. Segmentation of hand regions is robust to noise, and is very accurate because a user-specific lookup table is used. Second, we have proposed a tracking method based on CAMSHIFT algorithm [1] where the method is particularly tuned for tracking hand regions. As can be seen in the experimental results, the tracking performance is robust to cluttered background having colors similar to those of skin regions. Third, a novel method to display the

J.L. Crowley et al. (Eds.): ICVS 2003, LNCS 2626, pp. 88–97, 2003.
© Springer-Verlag Berlin Heidelberg 2003

mouse pointer gives users a natural feeling of controlling a mouse by physically modeling the mouse pointer. As a result, the motion of the mouse pointer on the monitor is very smooth even though actual tracking rate is as low as 5-6 Hz. The problem due to the difference of resolution between the input image from the camera and the monitor has also been resolved. The experimental results show a great promise that the proposed system can be applied to control computers in near feature.

This paper is organized as follows. The following section briefly reviews an overview of our system. Section 3 and 4 present the proposed methods for fingertip detection and the display of the mouse pointer on the monitor, respectively. Experimental results are reported in section 5.

2 System Overview

Let us briefly overview the entire system. Fig. 1 shows a block diagram of the computation in the system. Details of each computation will be presented in later sections.

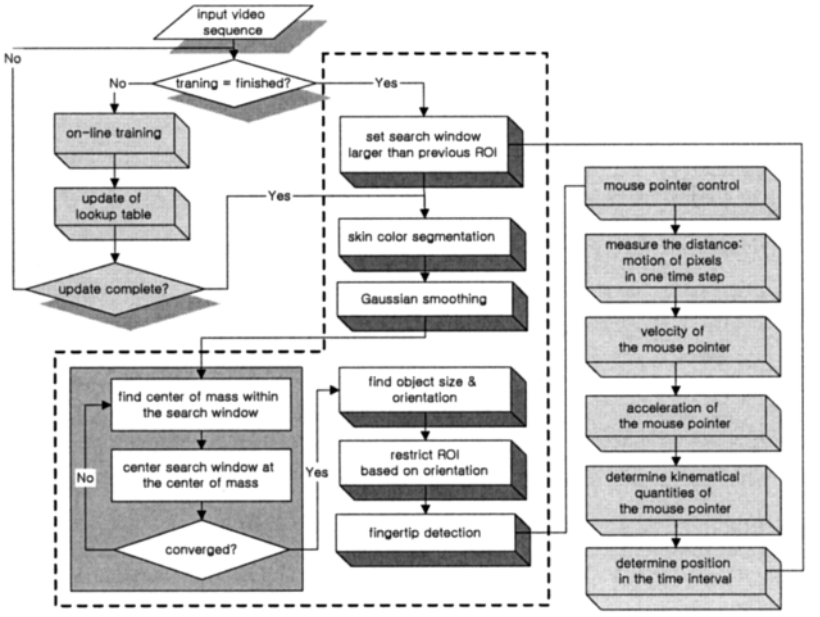


Fig. 1. System overview

The system is divided into three parts: adaptive on-line training of skin colors for hand region segmentation, fingertip tracking, and mouse pointer display. The first part is on-line training of skin colors robust to changes of illumination and hand poses, and complex backgrounds. The system learns the probability distribution of skin colors of the current user. The second part is fingertip tracking (dashed region in Fig. 1). Hand

regions are segmented in the input image sequence using the learned skin color lookup table computed in the first part. After segmentation of the hand regions, we compute the location of the fingertip. To rapidly detect the fingertip location, we have employed a region of interest (ROI) using the information about the size and the orientation of a hand region. The last part is for the control of the mouse pointer. The mapping between camera coordinates and monitor coordinates of the fingertip location is computed. In addition, smoothing of the mouse pointer path is performed so that the user can have natural feeling as if he or she were using a real mouse.

3 Fingertip Detection

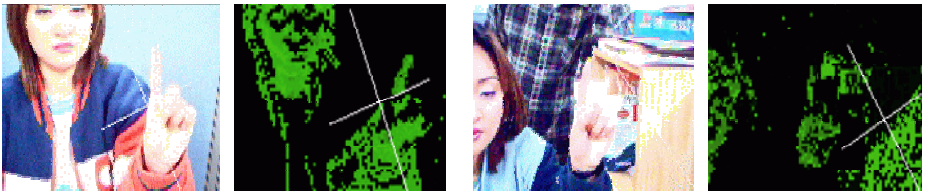
3.1 Adaptive Online Training of Skin Colors

We use an on-line training method as can be seen in Fig. 2. The user is asked to place his or her hand so that the rectangular box can stay in the hand. The method gets the system to learn variations of skin colors due to changes of local illumination and hand poses. The advantage of this method is that the accurate distribution of the current user's skin colors is obtained.

Fig. 2. The on-line training procedure to learn variations of skin colors due to changes of local illumination and hand poses.

There has been a research report [1] claming that, "Humans are all the same color (hue)". However, when skin color is detected by only hue information, we find it difficult to find skin regions of the user because it is affected by the colors of the environment having values of H (hue) similar to skin regions. Fig. 3 shows a couple of examples when the CAMSHIFT algorithm is applied to track a hand region. The intersection of the two lines represents the center of the hand region detected. However, the result of skin color segmentation is wrong because of (a) the red sleeve of the jacket and (b) the color of the bookshelf.

We have employed H (hue) and S (saturation) values of HSV color space for learning the probability distribution of skin colors. A 2D look up table is computed using H and S as the result of on-line training of skin colors. The entries of this table are the probability density of hue and saturation values.



(a) (b)

Fig. 3. Examples of wrong segmentation results of CAMSHIFT algorithm when only hue (H) value is used. (a) the effect of the red sleeve of the jacket, (b) the effect of the color of the bookshelf

3. 2 Detection and Tracking of Hand Regions

Determining the location and size of an ROI. The computation time is saved a lot by confining necessary computation to ROI. The hand region is tracked in the current frame within the ROI that was computed in the previous frame. In the ROI of the current frame, the hand region is detected but the centers of the ROI and the hand region detected do not accord because of the hand motion. The center of the hand region within the current ROI becomes the center of ROI to be used in the next frame.

Noise around the hand region may cause the propagation of errors in detection of the hand region. When there is noise around the hand region, the location (i.e. center) of ROI to be used in the next frame is not computed correctly and this has an effect on detection of the hand region. We can reduce the effect due to noise by not considering pixels in the skin region of which probability density is less than some threshold value. After thresholding, we obtain a binary image, $B(x,y)$, in ROI. $B(x,y)$=1 for pixels belonging to hand region. However, in some cases, part of the hand region is not detected due to this thresholding operation. The size of an ROI is determined by considering the size and the orientation of the hand region. The center coordinates (x_c, y_c) of the hand region can be simply computed using the 0^{th}, 1^{st} order moments [6] as follows.

$$x_c = \frac{M_{10}}{M_{00}}, \quad y_c = \frac{M_{01}}{M_{00}} \tag{1}$$

$$M_{00} = \sum_x \sum_y B(x,y), \quad M_{10} = \sum_x \sum_y xB(x,y), \quad M_{01} = \sum_x \sum_y yB(x,y)$$

The 2D orientation of the detected hand region is also easy to obtain by using the 2^{nd} order moments. The 2^{nd} order moments are computed as follows.

$$M_{20} = \sum_x \sum_y x^2 B(x,y), \quad M_{02} = \sum_x \sum_y y^2 B(x,y) \tag{2}$$

Then the object orientation (major axis) is;

$$\theta = \frac{\arctan\left(b,(a-c)\right)}{2} \tag{3}$$

where $a = \frac{M_{20}}{M_{00}} - x_c^2,\ b = 2\left(\frac{M_{11}}{M_{00}} - x_c y_c\right),\ c = \frac{M_{02}}{M_{00}} - y_c^2 \cdot$

The size of an ROI can be determined by the orientation, θ, of the hand region by using the moments computed and by scaling the horizontal and vertical lengths of the ROI as in equation (4). The horizontal and vertical lengths for the ROI are denoted by R_x and R_y, respectively.

$$\begin{gathered} R_x = s_x\sqrt{M_{00}},\ R_y = s_y\sqrt{M_{00}}, \\ \text{where } s_x = \cos|\theta| + 1,\ \ s_y = \sin|\theta| + 1,\ \ \max(s_x, s_y) + 0.1 \end{gathered} \tag{4}$$

The scaling factors, s_x and s_y, are determined from the statistical knowledge that the length ratio of the elongated axis vs. its perpendicular axis of a human hand does not exceed 2.1:1.

Hand tracking. The algorithm for tracking a hand region is based on the CAMSHIFT algorithm. The main difference between the CAMSHIFT algorithm and our algorithm is the way to determine the ROI. Fig. 4 shows an example of three successive updates of the search window when the CAMSHIFT algorithm is applied to track a hand region. The search window tends to include noise regions next to the previous search window that have the colors similar to skin regions. We can see that the size of the search window has grown and that the center of the detected hand region is not correctly computed.

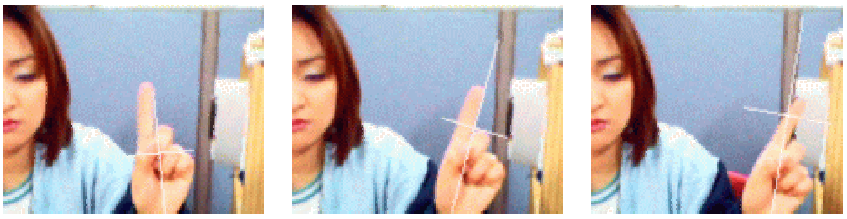


Fig. 4. Growing the search window in the CAMSHIFT algorithm

In contrast to CAMSHIFT, computes the ROI in a particular way that is tuned for a human hand while the CAMSHIFT is not. Therefore, the tracking performance of our algorithm is tweaked, especially when there are colors in the background similar to those of skin region. A block diagram of the algorithm is shown in Fig. 5.

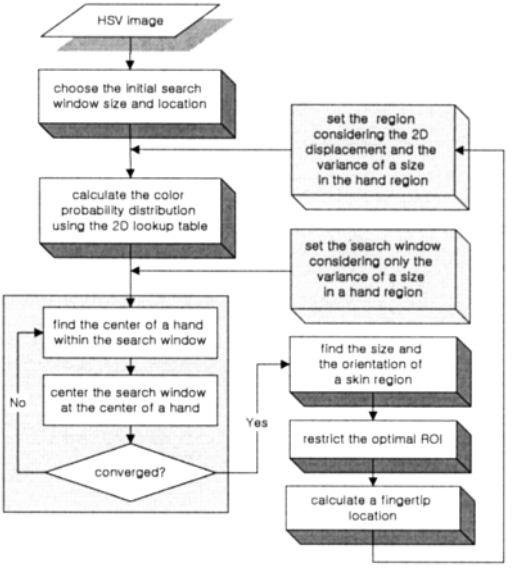


Fig. 5. A block diagram of hand tracking

3. 3 Computation of the Fingertip Location

If there is no noise, skin regions are detected perfectly. In this case, to compute the location of the fingertip, we can simply use the coordinates of boundary pixels [7] or can compute the center coordinates of the first knuckle of the index finger. However, detection of skin region cannot be perfect due to noise. We have used a way robust to noise for the detection of the fingertip location. We assume that the fingertip location lies in the first knuckle of the index finger. The length of the elongated axis of the hand region can be simply computed using equation (5) [8]. The end point of l is determined as the location of the fingertip. Refer to Fig. 6.

$$l = \sqrt{\frac{(a+c)+\sqrt{b^2+(a-c)^2}}{2}} \tag{5}$$

where a, b and c are the same as in equation (3).

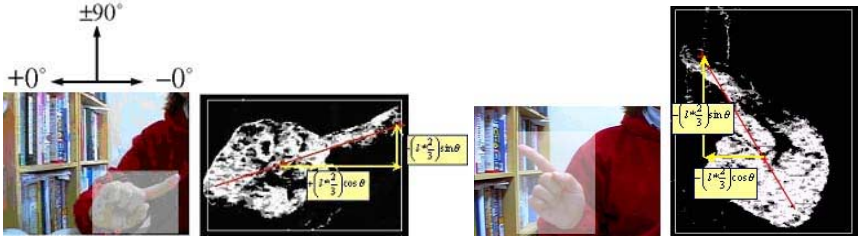


Fig. 6. Computation of the fingertip location in the cases of $\theta < 0$ (left) and $\theta > 0$ (right)

4 Display of a Mouse Pointer on the Monitor

Even if the fingertip is detected correctly, we might not get a natural motion of a mouse pointer due to the following problems. First, because of the limitation on the tracking rate on commonly available PC class hardware, we can merely get discontinuous coordinates of fingertip locations. Accordingly, if these coordinates of the fingertip locations are used for the display on the monitor without any smoothing, the mouse pointer is going to jump around in a very unnatural fashion. Second, if the location of the fingertip in each frame is converted to the monitor coordinates directly, the difference of the resolution between the input image from the camera and the monitor makes it very difficult to position the mouse pointer accurately. Moreover, the jitter can occur even though the hand remains still. A small movement of the fingertip in the input image can cause a significantly large movement of the mouse pointer on the monitor. To circumvent these problems, we describe how the fingertip location in the input image is mapped to a location on the monitor, and how it is both necessary and possible to smooth the trajectory of the mouse pointer by physically modeling of the mouse pointer.

For simplicity, let us describe our method for one-dimensional case. In the i^{th} frame, the difference between the location of the current mouse pointer (X_{i-1}) and the fingertip location (H_i) is converted to an amount of force that accelerates the mouse pointer in an appropriate direction as in equation (6).

$$D_i = H_i - X_{i-1}, \quad F_i = f(D_i) \tag{6}$$

If the displacement of the fingertip location in the input image is smaller than a predetermined threshold, the minimum force, F_{min}, will be set to zero because the mouse pointer should remain still on the monitor. The threshold is set to 1% of the resolution of each axis on the monitor. The mouse pointer is treated as an object having mass and velocity as well as position. Its motion can be described using the second-order differential equation for the position $x(t)$ based on the Newton's law.

$$\frac{d^2x(t)}{dt^2} = \frac{F}{m} \tag{7}$$

where F and m denote the amount of force and the mass of an object, respectively. For the solution of equation (7), we use Euler-Cromer algorithm [9] as follows.

$$x_{n+1} = x_n + v_n \Delta t \tag{8}$$

where Δt is the time interval between successive two frames.

Our observations shows that we need to achieve the tracking rate of more than 20 fps so that a user can perceive a natural movement of the mouse pointer. If a system cannot support this rate, the value of k that satisfies $\Delta t / k = 0.03(s)$ is chosen. That is, locations of the mouse pointer are generated on the monitor during the time between two image frames so that the resulting path of the mouse pointer may be smooth. This one-dimensional method is simply extended to two dimensions, x and y, to be applied to our case.

5 Experimental Results

Fig. 7 shows examples of the tracking results for three lighting conditions: (a) natural and artificial lighting, (b) natural lighting and (c) artificial lighting. The rectangular boxes represent ROI depending on the size and the orientation of the hand region. The location of the fingertip is reliably computed in spite of the background having colors similar to those of skin regions.

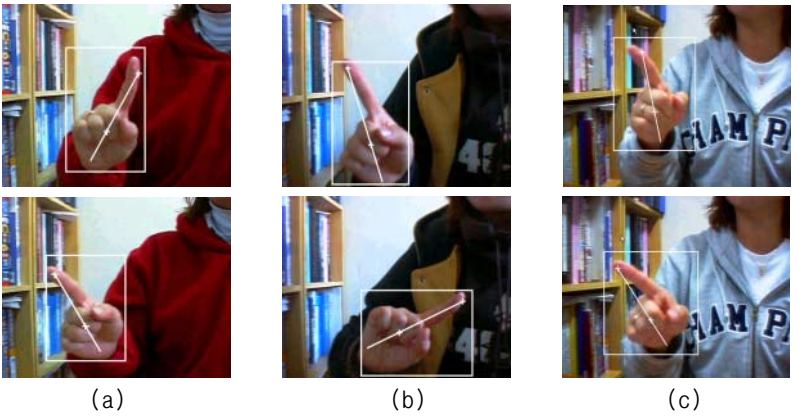


Fig. 7. Tracking results of the fingertip location. (a) natural and artificial lighting, (b) natural lighting, (C) artificial lighting

Table 1 summarizes the experimental results. The accuracy measures the frequency of correct detection of fingertip location.

Table 1. The accuracy of detecting a fingertip location

	# total frames	# detected frames	accuracy (%)
natural and artificial lighting	400	373	93.25
natural lighting	400	368	92
artificial lighting	400	387	96.75
total	1200	1128	94

We can see that the tracking performance is quite robust to lighting conditions. Fig. 8 shows the smoothing result of the mouse pointer path when the tracking rate is 6Hz. With raw tracking results, the monitor coordinates of the mouse pointer are quite discontinuous and not smooth as can be seen in Fig. 8 (a). The path of the mouse pointer is smoothed using the proposed method that physically models the mouse pointer.

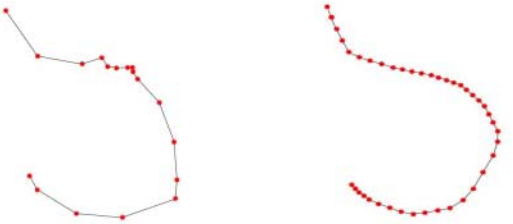


Fig. 8. An example trajectory of the mouse pointer. (a) raw tracking results (6Hz), (b) a smoothed version of the raw tracking results

6 Conclusion

In this paper we have proposed a method to detect and track the fingertip of the index finger for the purpose of application to a human mouse system. The proposed method for fingertip tracking shows a good performance under cluttered background and variations of local illumination due to hand poses by virtue of online training of skin colors and optimal determination of ROI. In addition, a natural motion of the mouse pointer is achieved by smoothing the path of the fingertip location using physical model of the mouse pointer. The experimental results show a great promise that the proposed method can be applied to manipulate computers in near future.

Acknowledgements. This work was supported by grant number R01-1999-000-00339-0 from the Basic Research program of the Korean Science and Engineering Foundation.

References

1. Gary R. Bradski, "Computer Vision Face Tracking For Use in a Perceptual User Interface", *Intel Technology Journal Q2*, 1998.
2. R. Kjeldsen and J. Kender "Interaction with On-Screen Objects using Visual Gesture Recognition" *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 788–793, 1997.
3. C. Jennings, "Robust finger tracking with multiple cameras", *International Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems*, pp. 152–160, 1999.
4. J. Rehg and T. Kanade, "Visual Analysis of High DOF Articulated Object with Application to Hand Tracking", *CMU Tech. Report CMU-CS-95-138*, Carnegie Mellon University, April, 1993.

5. R. Kjeldsen and J. Kender, "Towards the use of Gesture in Traditional User Interfaces", *International Conference on Automatic Face and Gesture Recognition,* pp. 151–156, 1996.
6. B. K. P. Horn, "Robot vision", MIT Press, 1986.
7. Andrew Wu, Mubarak Shah and N. da Vitoria Lobo, " A Virtual 3D Blackboard: 3D Finger Tracking using a Single Camera" *Proceedings of the Fourth International Conference on Atomatic Face and Gesture Reconition*, pp. 536–543, 2000.
8. W. T. Freeman, K. Tanaka, J. Ohta and K. Kyuma, "Computer vision for computer games" *Proceedings of the Second International Conference on Automatic Face and Gesture Recognition*, pp. 100–105, 1996.
9. Harvey Gould and Jan Tobochnik, "An Introduction to Computer Simulation Methods", Addison Wesley Publishing Company, 1996.