Remote Mouse Control Using Fingertip Tracking Technique

Phuong Dat Le and Vinh Hao Nguyen

Faculty of Electrical and Electronic Engineering, Ho Chi Minh City University of Technology, Ho Chi Minh City, Vietnam
lephuongdat1988@gmail.com, vinhhao@hcmut.edu.vn

Abstract. This paper proposes a real-time fingertip tracking technique using a web camera to enable users to remotely control their computer mouse by their own bare hands. The hand region is firstly extracted by background subtraction and filtered by the morphological opening operations and blob labeling. Then, convex hull and convexity defect are used to count the fingers and detect the coordinates of the fingertip. Next, the coordinates of the fingertip is mapped to the screen coordinates and smoothed by the Moving Average. Finally, the events corresponding to the detected fingers are sent to the computer system to control the mouse. Experimental results show that the proposed technique can successfully count the finger with the accuracy of 98.3% and work well in real time.

1 Introduction

Along with the development of technologies, computer systems have become vital to the world we live in. Most computer systems require more and more interaction. Thus, it is necessary to have types of interaction which are natural and easy to use.

Human computer interaction (HCI) can be described as an interaction to convey information from users to computer systems. Traditional interaction that relies on the mouse and keyboard is still the most familiar HCI. However, these devices are inconvenient and unnatural. Combining computer vision and HCI, it is possible to create an advanced input device which is an attractive alternative to these cumbersome interface devices. The use of hand gestures via computer vision is one of the most interesting parts of HCI. Using their hands as an input device can help people communicate with computers in a more intuitive and natural way.

There have been many attempts to recognize hand gestures. Most of them are based on hand segmentation because it can decrease the amount of image information. Colored gloves [1], [2] and markers [3] are used to detect user actions. However, these methods are inconvenient and too complex to use. Another popular method is using skin color [4]-[8]. Skin color detection is a very challenging task as the skin color in an image is sensitive to various factors such as light condition, background color, motion, and ethnicity. Edge in [9], [10] has been used for detection, but the limitation of this method is that the hand can easily be extracted only from uniform background images. References [11], [12] propose background subtraction for hand

I. Zelinka et al. (eds.), *AETA 2013: Recent Advances in Electrical Engineering and Related Sciences*, Lecture Notes in Electrical Engineering 282,
DOI: 10.1007/978-3-642-41968-3_47,

segmentation. They archived quite good results; though they need the background to be unchanging and different from skin color. To overcome the limitations of the methods above, some studies use special cameras such as infrared cameras [13] or kinect [14], [15]. They can detect the hand from the input image even in complex backgrounds and under different lighting conditions, yet they are not very practical, owing to the expense of the devices.

Real time fingertip detection and tracking in a 2-D plane can be applied in computer system as a visual mouse [16]-[18]. Inspired by those systems, the main focus of the paper is finding an efficient method to detect the fingertips and control the mouse pointer, utilizing a bare hand and a simple web camera with a frame rate of approximately 30 frames per second. This paper also suggests a robust user interface which is intuitive and easy to use. In each frame, hand regions are first detected by background subtraction and filtered by the morphological opening operations and blob labeling. Next, the forearm is removed by using distance transform. Then, fingers and fingertips are detected with convex hulls and convexity defects. After that, the coordinates of the fingertip is mapped to the screen coordinates and smoothed by the Moving Average. Finally, the event which is determined by the number of fingers, and the coordinates are sent to the computer to control the mouse.

2 Hand Segmentation and Finger Detection

2.1 Hand Segmentation

Mouse movement is based on finger trajectory, as well as hand motion. Background subtraction is one of the best methods to obtain the moving areas. Therefore, it is suitable for the proposed technique. Average background is used to achieve the better segmentation. Equation (1) shows how to obtain the average background in the initial phase.

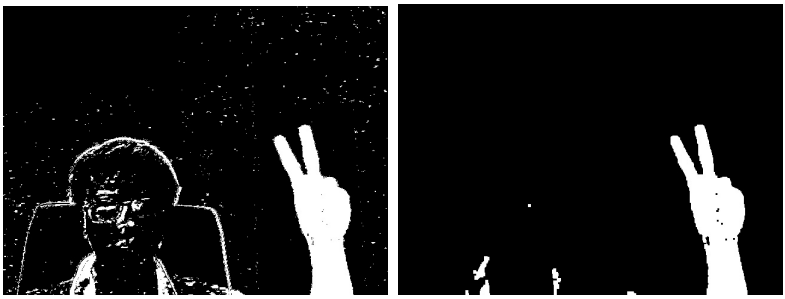
$$\overline{I_{bg}} = \frac{1}{N} \times \sum_{k=1}^{N} I^k \tag{1}$$

where I represents the value of color at a specific position, N is the number of frames which are used to calculate the average background.

The difference frame is obtained using the difference of the average background and the current frame. If the result is greater than a threshold (th), it is defined as the foreground.

$$I_{fg} = \begin{cases} 1, & \left| I^k - \overline{I_{bg}} \right| > th \\ 0, & else \end{cases} \tag{2}$$

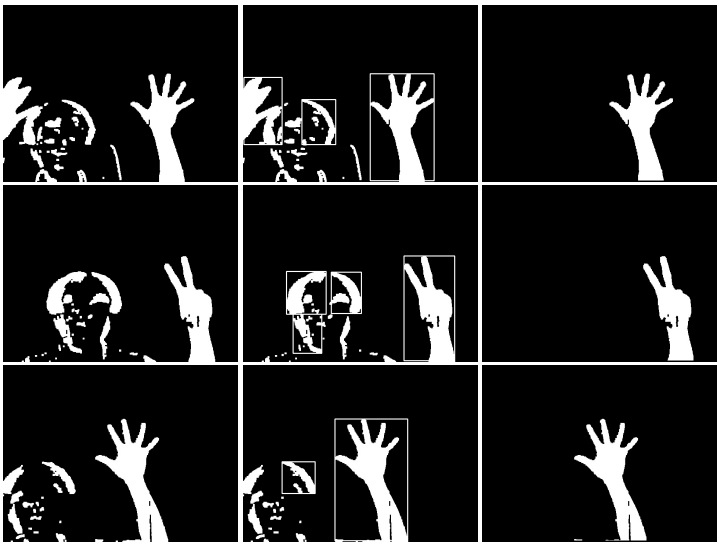
The web cameras always produce a varying degree of noise. In the experiment, the noise of the web camera image can be removed by the morphological opening operation. This operation is performed by using image erosion and image dilation. Erosion trims down the difference frame, and thus the speckles are eroded to nothing while the larger regions that contain visually significant content are not affected. Dilation expands the frame pixels to get back the original shape of the content.



(a) (b)

Fig. 1. Result of background subtraction. (a) Difference frame, (b) difference frame with the opening operation.

Fig. 1 shows a result of background subtraction with and without the opening operation. To detect the hand regions, blob labeling is used in [11]. This method labels each pixel in the foreground image. These pixels which get the same label are connected with each other and are considered as a blob. Information about the minimum size window surrounding separate blobs is also obtained. Limiting the size of the window and choosing the rightmost blob can help remove the unexpected movement of face and body. The rightmost blob is also extracted and it contains the hand region. Fig. 2 shows some results of the blob labeling method.



(a) (b) (c)

Fig. 2. Some results of blob labeling. (a) Difference frames, (b) blobs with limited size, (c) rightmost blobs.

2.2 Finger Detection

2.2.1 Forearm Filtering

In order to enhance the finger detection, the forearm should be filtered from the hand region. Using the distance transform which was introduced in [19], the research in [4], [12] proposed a very useful way to remove the forearm.

The distance transform of an image is defined as a new image in which every output pixel is set to a value equal to the distance to the nearest zero pixel in the input image. The distance transform of the hand region is computed by using the 5x5 neighborhood fast chamfer distance transform [18] that gives a good approximation to the exact Euclidean distance.

Fig. 3 illustrates the procedure of forearm filtering. The binary image which contains hand region is firstly converted to a distance transform image. The pixel with max value in distance transform image represents the palm center. Then, the offset circumference whose radius is 1.65 times the palm center value is calculated. All pixels outside this circumference and lower than the palm center are assumed to represent the forearm and are removed.

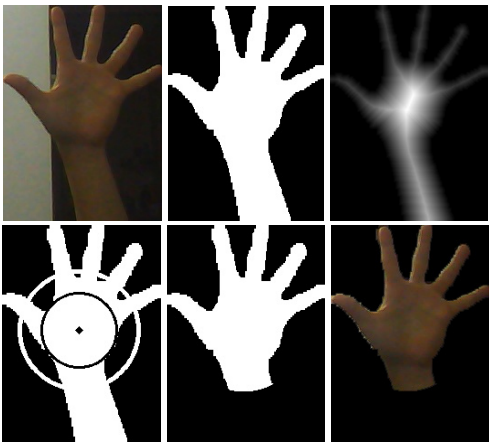


Fig. 3. Forearm filtering

2.2.2 Finger Detection

A useful way to find out the coordinates of the fingertip and the number of fingers is to compute a convex hull for palm region and then compute its convexity defect which is introduced in [20].

Using a lower limit for the depth of defect, the defects between two fingers are detected. After that, the coordinates of the fingertips are computed based on the start point and end point of these defects. The number of fingers equals the number of these defects plus 1. When the hand is closed or only 1 finger is opened, no defect is detected. In this case, the maximum distance from palm center to the points of convex hull is used to determine whether the hand is closed or not. If this value is less than a

specific value, there is a closed hand. If not, the farthest point on the convex hull is assumed to represent the fingertip of the hand. Fig. 4 shows six cases of the finger counting stage.

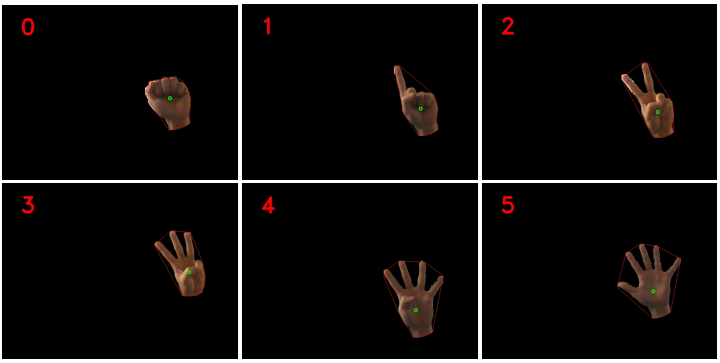


Fig. 4. Six cases of finger counting stage

3 Mouse Control

3.1 Mouse Coordinates

The experiment shows that the mouse pointer is not as smooth and natural as real mouse, even if the coordinates of the fingertip is detected correctly. This issue is caused by the following reasons. First, the coordinates of the fingertip are extracted from each frame; therefore, they are just the discontinuous coordinates of fingertip locations. If we use these coordinates directly, the mouse pointer will jump around in a very unnatural way. Second, the difference of the resolution between the camera and the screen require a coordinate conversion and this makes it difficult to position the mouse pointer accurately. To overcome these difficulties, we suggest using the Moving Average algorithm to average the displacement of the detected fingertip position and use this displacement as the coordinates of mouse pointer on the screen.

Moving Averages are the techniques that can be applied to time series data to help smooth data and filter out the noise. The two most popular types of moving averages are the Simple Moving Average (SMA) and the Exponential Moving Average (EMA). SMA as in (3), is the simplest way and can filter out a lot of noise; however, it also increases the lag and decreases the precision of the data.

$$S(k) = \frac{1}{N} \times \sum_{k=1}^{N} x(k) \tag{3}$$

Here x represents the value of input data, S represents the EMA output value and N is the number of time periods.

This paper proposed EMA which was first suggested in [21], to smooth the fingertip trajectory. EMA reduces the lag by applying more weight to recent data. The EMA for a series data is calculated as follows:

$$E(0) = \frac{1}{N} \times \sum_{k=1}^{N} x(k) \tag{4}$$

$$\alpha = 2 / (N + 1) \tag{5}$$

$$E(t) = E(t-1) + (x(t) - E(t-1)) \times \alpha \tag{6}$$

Here E is the EMA output value, α is the weighting multiplier between 0 and 1.

3.2 Mouse Events

To control the computer mouse, it is necessary to define some conditions to trigger mouse events. Reference [11] proposes a very natural and convenient way to trigger draw function by only one bare hand. In this paper, we improved and systematized those conditions. Let's assume that users only use their right hand, and the hand and the face are not overlap. The events which are determined by the number of the fingers are tied with the corresponding coordinates and sent to the computer via *SendInput* routine to control the mouse.

4 Experimental Results

The proposed technique was implemented in Microsoft Visual Studio 2010 with OpenCV 2.4.5 library. The input images with 24 bit-true color were captured by an embedded camera on the laptop or a common web camera (GENIUS FaceCam 1320) with a resolution of 320x240 pixels. The average processing time per frame was between 30 and 37 milliseconds. This is the same as 30 fps, suggesting that it is fast enough to be used in real time applications.

Compared to some previous studies, our proposed method can provide more precise hand regions while filtering out the forearm. It can recognize 6 cases of finger counting, while [4] and [11] can only recognize 5. The finger counting stage has also been tested with 40 different people with different hand size under varied lighting conditions. Each person was asked to show 6 gestures representing numbers from 0 to 5 in front of the camera. Table 1 shows the experimental test results of finger counting. The average rate of recognition of finger counting is 98.3%. This is a good result when comparing with other studies. The misdetections were observed when the fingers with short lengths were shown or one of them overlap with the others. After the coordinates of the fingertip is detected, it will be converted to screen coordinates to control the mouse pointer. Fig. 5, Fig. 6 and Fig. 7 show the 271 consequent coordinates before and after applying the Moving Averages when the finger was standing still.

Table 1. The test results of finger counting

Number of fingers inputted	Number of successful counting	Accuracy (%)
0	40	100
1	40	100
2	40	100
3	38	95
4	39	97.5
5	39	97.5

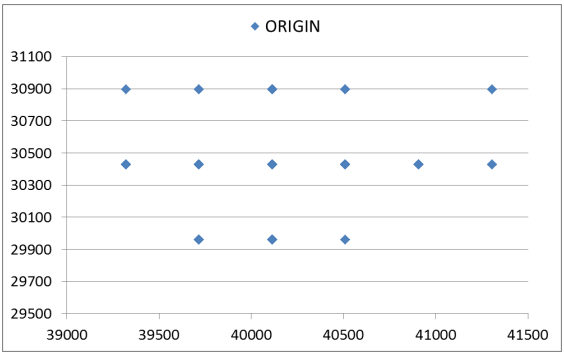


Fig. 5. The original coordinates of the mouse when the finger was standing still

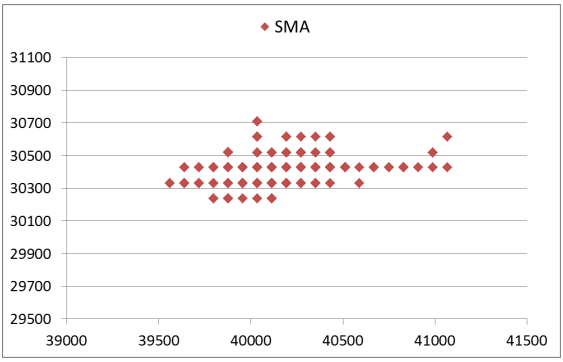


Fig. 6. The SMA coordinates of the mouse when the finger was standing still

In Fig. 5, the coordinates of the mouse jump a long distance around the real coordinates because of the noise from the sources and the conversion. This makes it very difficult to control the mouse. By using the Moving Averages with the time periods N = 5, Fig. 6 and Fig. 7 give much better results for controlling the mouse pointer. We reduced a lot of noise and jitter that appear in the original mouse coordinates.

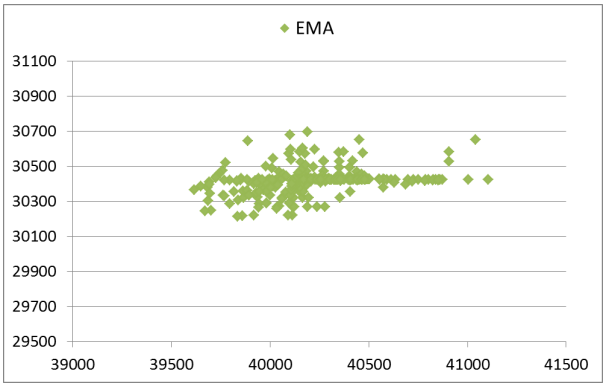


Fig. 7. The EMA coordinates of the mouse when the finger was standing still

The Moving Average also helps smooth the mouse trajectory and makes it become more natural. Fig. 8 shows the comparison of the 25 consequent coordinates when the finger is moving. The EMA coordinates give the best trajectory when reducing the noise and the lag which appear on ORIGIN and SMA trajectories. The sample video about an application of our proposed technique is stored in the following link: https://www.youtube.com/watch?v=r78D1BBgIs0&list=P

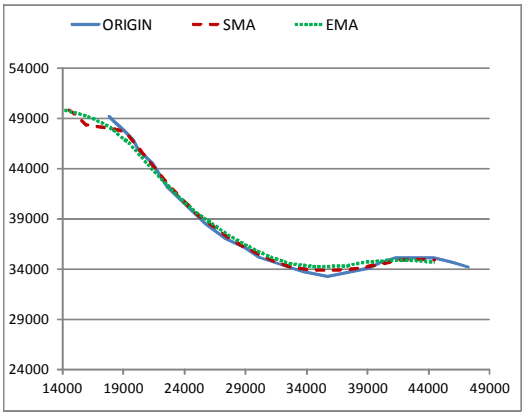


Fig. 8. The coordinates of the mouse when the finger was moving

5 Conclusion

In this paper, we have presented a background model for the real time fingertip tracking technique. Based on natural hand gestures, we have created an interface which is intuitive and easy to use for the users. We also used our proposed technique to interact with the computer such as playing game, surfing the web. Through

experiments, we are able to confirm that our model works with real time speed and high accuracy.

The main drawbacks of the proposed technique are that it only works well in an unchanging background, and it has trouble with objects at the same color as the skin. These drawbacks can be overcome by using a gradient or 3-D model. With six cases of finger counting stage and the fingertip coordinates, our model can be extended to handle more events and applied to interact with other smart environments in future works.

References

1. Wang, R.Y., Popovic, J.: Real-Time Hand-Tracking with a Color Glove. ACM Transaction on Graphics 28(3) (2009)
2. Geebelen, G., Maesen, S., Cuypers, T., Bekaert, P.: Real-time hand tracking with a colored glove. In: Proc. 3D Stereo Media (December 2010)
3. Choondal, J.J., Sharavanabhavan, C.: Design and Implementation of a Natural User Interface Using Hand Gesture Recognition Method. International Journal of Innovative Technology and Exploring Engineering 2(4) (March 2013)
4. Lee, B., Chun, J.: Interactive Manipulation of Augmented Objects in Marker-Less AR Using Vision-Based Hand Interaction. In: 2010 Seventh International Conference on Information Technology, pp. 398–403 (2010)
5. Nagarajan, S., Subashini, T.S., Ramalingam, V.: Vision Based Real Time Finger Counter for Hand Gesture Recognition. International Journal of Technology 2(2) (December 2012)
6. Abdallah, M.B., Sessi, A., Kallel., M., Bouhlel, M.S.: Different Techniques of Hand Segmentation in the Real Time. International Journal of Computer Applications & Information Technology 2(I), 45–49 (2013)
7. Raheja, J.L., Das, K., Chaudhary, A.: An Efficient Real Time Method of Fingertip Detection. In: 7th International Conference on Trends in Industrial Measurements and Automation (TIMA 2011), Chennai, India, pp. 447–450 (January 2011)
8. Vivek Veeriah, J., Swaminathan, P.L.: Robust Hand Gesture Recognition Algorithm for Simple Mouse Control. International Journal of Computer and Communication Engineering 2(2), 219–221 (2013)
9. Crampton, S.C., Betke, M.: Counting Fingers in Real Time: A Webcam-Based Human-Computer Interface with Game Applications. In: Proc. Conf. Universal Access Human-Comput. Interaction, pp. 1357–1361 (2003)
10. Ravikiran, J., et al.: Finger Detection for Sign Language Recognition. In: Proc. The International MultiConference of Engineers and Computer Scientists (IMECS 2009), Hong Kong, vol. I (March 2009)
11. Park, H.-S., Jo, K.-H.: Real Time Hand Gesture Recognition for Augmented Screen using Average Background and CAMshift. In: The 19th Korea-Japan Joint Workshop on Frontiers of Computer Vision, pp. 18–21 (2013)
12. Pedro, L.M., Caurin, G.A.P., Belini, V.L., Pechoneri, R.D.: Hand Gesture Recognition for Robot Hand Teleoperation. In: ABCM Symposium Series in Mechatronics, vol. 5, pp. 1065–1074 (2012)
13. Oka, K., Sato, Y., Koike, H.: Real-time Tracking of Multiple Fingertips and Gesture Recognition for Augmented Desk Interface Systems. In: Proc. The Fifth IEEE International Conference on Automatic Face and Gesture Recognition (2002)

14. Oikonomidis, I., Kyriazis, N., Argyros, A.A.: Efficient model-based 3D tracking of hand articulations using Kinect. In: Proc. The 22nd British Machine Vision Conference (2011)
15. Raheja, J.L., Chaudhary, A., Singal, K.: Tracking of Fingertips and Centers of Palm using KINECT. In: Computational Intelligence, Modelling and Simulation, pp. 248–252 (September 2011)
16. Park, J., Yi, J.-H.: Efficient Fingertip Tracking and Mouse Pointer Control for a Human Mouse. In: Crowley, J.L., Piater, J.H., Vincze, M., Paletta, L. (eds.) ICVS 2003. LNCS, vol. 2626, pp. 88–97. Springer, Heidelberg (2003)
17. Argyros, A.A., Lourakis, M.I.A.: Vision-Based Interpretation of Hand Gestures for Remote Control of a Computer Mouse. In: Huang, T.S., Sebe, N., Lew, M., Pavlović, V., Kölsch, M., Galata, A., Kisačanin, B. (eds.) ECCV 2006 Workshop on HCI. LNCS, vol. 3979, pp. 40–51. Springer, Heidelberg (2006)
18. Vivek Veeriah, J., Swaminathan, P.L.: Robust Hand Gesture Recognition Algorithm for Simple Mouse Control. International Journal of Computer and Communication Engineering 2(2) (March 2013)