A Six-degree-of-freedom Virtual Mouse based on Hand Gestures

Xingfeng Wang, Kaihuai Qin
Department of Computer Science and Technology
Tsinghua University
Beijing 100084, P.R.CHINA
qkh-dcs@tsinghua.edu.cn
wangxf04@mails.tsinghua.edu.cn

Abstract—**The virtual mouse is an important research topic in HCI (Human Computer Interaction) technology, and the most popular method is based on hand gestures. In this paper, a six-degree-of-freedom virtual mouse based on hand gestures is presented using an inexpensive USB webcam. The hand tracking and gesture recognition framework includes the motion, skin color and finger information of the hand. It can even provide computer access for the disabled people who cannot bend their fingers at all with severe disabilities. The experiments show that the system has good interactive input functions in real time.**

Keywords-virtual mouse; hand gesture; six-degree-of-freedom; tracking; recognition

I. INTRODUCTION

With the development of computer technologies, people have more demands for HCI devices. The traditional 2D (dimension) mouse with three degrees of freedom (to move along the X-, Y- and Z-axis) can't satisfy their demands. They hope the mouse can control the rotations around the axes besides the movements along the axes. So they can browse the whole 3D entity or virtual scene. Moreover, they hope the interaction can be more natural. The glove-based interaction [1][2][3] is more intuitional, but it is expensive and users have to wear special equipment. Now the vision-based natural hand gesture interaction is popular [4][5][6].

In this paper a six-degree-of-freedom virtual mouse based on hand gestures in complicated background is proposed. The CamShift (Continuously Adaptive MeanShift) algorithm according to the skin color is used to track the hand, and fingers features are used to recognize poses. Both the motion and poses of the hand are used for gesture understanding.

This paper has five sections. In the next section, the hand tracking and centroid calculation of a hand are explained. Section III explains the feature extraction method. In Section IV, the hand gesture understanding is described. We present our results and conclusions in Section V.

II. HAND TRACKING AND CENTROID CALCULATION

The skin color is an apparent and important character of a hand. And it is not sensitive to scale variety and distortion of hand gestures. So the skin color of a hand can be a clue of the hand tracking and segmentation.

According to the computer color theory, one color has many different expressions, so there are many different color systems. The color systems have RGB (red, green, blue), CMY (cyan, magenta, yellow), HSV (hue, saturation, value) and so on. Every color system has its application fields, and the RGB color system is the most popular. For example, the images from a USB webcam are the RGB color system. The HSV color system is transformed from the RGB color system, and it is more appropriate to human vision. In the HSV color system, the hue part is the essence of a color. So we use the hue information of the hand's skin color in this paper.

A. Background difference

Sometimes the background is very complicated. If we can separate the background from the image, the speed and accuracy of the hand tracking will be improved. In this paper the USB webcam is fixed, so the difference between the background image and the hand gesture image can eliminate the influence of the background. If *B(x, y)* is the background image, *P(x, y)* is the hand gesture image, and then the difference *D(x, y)* is

$$D(x,y,i)=\begin{cases}P(x,y,i), & otherwise\\ 0, \left|P(x,y,i)-B(x,y,i)\right|<T1\end{cases}$$

where *D(x,y,i)*, *i=1, 2, 3,* represent R, G and B color part, respectively, and *T1* is the threshold. If the R, G and B parts of a pixel in the hand gesture image have all changed less than *T1*, these values of this pixel are all set as zero. Otherwise, *P(x, y, i)* are the R, G and B values of this pixel.

B. Hand tracking

A moving hand gesture tracking and recognition system need to be real-time, so we must try our best to improve the efficiency of the tracking and recognition. Considering the continuity of the movement, we need not compute the whole image of the video. In this paper CamShift algorithm and the later contour extraction algorithm interact to reduce the calculation scope greatly.

978-0-7695-4031-3/10 $26.00 © 2010 IEEE
DOI 10.1109/iCECE.2010.69

In CamShift algorithm the input image is a probability distribution image. So we should transform the difference image into the back projection image according to the hue histogram of the hand skin. In the back projection image the value of each pixel characterizes probability that the corresponding pixel of the difference image belongs to the hand skin. The bright pixels of the back projection image correspond to the hand skin. In order to fill the holes in the gesture region, we can do a closing operation of the mathematical morphology, as shown in Fig.1.

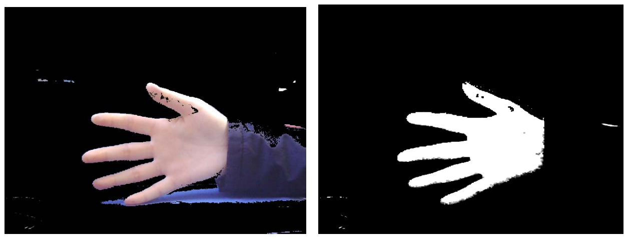

(a) (b)

Figure 1. (a) The difference image. (b) The back projection image after a closing operation.

CamShift algorithm is the MeanShift algorithm of a sequence of continuous images. The MeanShift algorithm has four steps: (1) Choose an initial search window; (2) Calculate the centroid of the window; (3) Move the center of the window to the centroid; (4) Repeat steps (2) and (3) until the moving distance is less than the threshold. The centroid *(x_m, y_m)* is calculated as:

$$M_{00} = \sum_x \sum_y I(x,y)\,, \quad M_{10} = \sum_x \sum_y xI(x,y)\,,$$

$$M_{01} = \sum_x \sum_y yI(x,y)\,, \;\; x_m = M_{10} / M_{00}\,, \;\; y_m = M_{01} / M_{00}$$

where *$I(x, y)$* is the pixel value of the position *(x, y)* in the back projection image. The ranges of *x* and *y* are the size of the search window.

The CamShift algorithm has four steps: (1) Choose an initial search window *W1*; (2) Run the MeanShift algorithm; (3) Resize the search window according to the result of Step (2), and get a new window *W2*; (4) Use *W2* as the initial search window for the next video frame and repeat the algorithm. The details can see the Intel's computer vision library OpenCV [7].

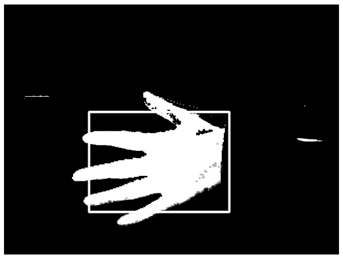


Figure 2. The back projection image with the window *W2*.

This paper doesn't directly use the *W2* as the initial search window for the next frame. We change *W2* a little and use it as the calculation scope of a contour extraction algorithm. Then the bounding rectangle of the hand gesture contour is used as the initial search window for the next frame. The human hand is irregular, so *W2* just includes the main body of the hand gesture, as shown in Fig.2. Furthermore, the bounding rectangle includes the whole gesture, and it is more suitable for continuous tracking.

III. FEATURE EXTRACTION

In this paper the gesture understanding depends on the position of the gesture and the state of fingers. The position of the gesture is the center of the window *W2* after the CamShift algorithm. And the state of fingers needs to be extracted. After computing the k-curvature of the contour points to extract the finger tips and finger roots, we part the finger tips from the finger roots, and mark the thumb and the little finger.

A. Contour extraction

Contour extraction doesn't calculate the whole image. We find the gesture contour in the window *W3*. *W3* has the same center with the window *W2*, but it is larger than *W2*. Here we set the width and height of *W3* to be 1.5 multiples of the width and height of *W2* to ensure that the whole gesture is in *W3*. And the calculation scope decreases greatly.

Because of the influence of the background and illumination, there are some points and connected regions in the back projection image, but they are very small as compared with the hand gesture. We can eliminate these noises when extracting the contour.

$$contour(i) = \begin{cases} gesture, & Count(i) > T2 \\ noise, & Count(i) \le T2 \end{cases}$$

where *contour(i)* is the extracted contour, *Count(i)* is the number of pixels on the contour, and *T2* is the threshold. After the contour extraction, the bounding rectangle *W4*of the gesture is set to be the initial search window of the CamShift algorithm for the next video frame. If the hand moves too quickly, *W4* can be enlarged to be the initial search window of the CamShift algorithm for the next video frame.

B. Finger tips and finger roots extraction

We extract finger tips and finger roots according to the k-curvature of the contour point. And the k-curvature is the cosine of the angle between the two vectors [*C(i),C(i-k)*] and [*C(i),C(i+k)*].

$$cur = \frac{[C(i),C(i-k)]\bullet[C(i),C(i+k)]}{\left|[C(i),C(i-k)]\right|\left|[C(i),C(i+k)]\right|}$$

where $k = \sqrt{S_{w4}} / 15$, *C(i)* is the *i*th point on the gesture contour, *cur* is the k-curvature of *C(i)*, and S_{w4} is the area of the bounding rectangle *W4*. The value of *k* is associated with the area of the hand gesture to make the gesture recognition more adaptive and accurate. If *cur>0*, the corresponding point is supposed to be a potential feature point. At the top and bottom of each finger the potential feature points range densely. We set the potential feature points as a group in which the distance between two adjacent points is less than *2k*. The middle point of each group is called feature point, and it is a potential finger tip or finger root. In order to speed up, we can calculate the k-curvature every ten points on the contour.

If the hand opens out, there are five finger tips and four finger roots. But sometimes the feature points at the wrist could be extracted. We call them interference points. The interference points are usually less than nine. And these interference points are far from the finger tips and finger roots. We can mark the interference points according to these two characteristics. Choosing a threshold *T3* to make sure there are two pairs of two adjacent points between which the distance is more than *T3*. One pair is the thumb and its adjacent interference point. The other pair is the little finger and its adjacent interference point. Here *T3* is multiples of *k* in the k-curvature algorithm to make *T3* adapt to the area of the hand gesture. Then we calculate the midpoints *M1* and *M2* of the both pairs. The line between *M1* and *M2* divides the hand gesture into two parts. The part that has less feature points is marked, and the feature points of it are interference points. The feature points of the other part are the finger tips and finger roots.

C. Distinguishing finger tips from finger roots

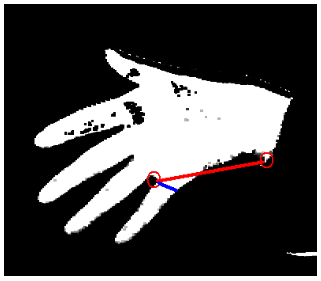


Figure 3.A example for the little finger.

In the back projection, every pixel value of the background is zero, and every pixel value of the hand gesture is nonzero. For each feature point excluding the interference point, if the nonzero points are more than the zero points on the line between the left adjacent feature point and the right adjacent feature point, the feature point is the finger tip; otherwise, it is the finger root. But sometimes the thumb or the little finger is different from our expectation. For example, as the red line shown in Fig.3. If we can calculate on the blue line, then the result will be right. So we improve the algorithm as follows:

(1) Calculate the distance between the *i*th feature point (excluding the interference point) and the (*i-1*)th feature point (including the interference point) *l1* and the distance between the *i*th feature point and the (*i+1*)th feature point (including the interference point) *l2*. If the *i*th is the last feature point, then the (*i+1*)th is the first feature point. If the *i*th is the first feature point, then the (*i-1*)th is the last feature point.

(2) If $l1 > T3$, then $d = index_{i+1} - index_i$ .Where $index_{i+1}$ and $index_i$ are the indexes of the (*i+1*)th and *i*th feature points on the gesture contour. After finding the index of ($index_i - d$)th point (not feature point) on the gesture contour, we calculate the numbers of nonzero points and zero points on the line between the point and the (*i+1*)th feature point in the back projection image. Then we can judge the *i*th feature point is the finger tip or the finger root.

(3) If $l2 > T3$, then $d = index_i - index_{i-1}$ .Where $index_i$ and $index_{i-1}$ are the indexes of the *i*th and (*i-1*)th feature points on the gesture contour. After finding the index of ($index_i + d$)th point (not feature point) on the gesture contour, we calculate the numbers of nonzero points and zero points on the line between the point and the (*i-1*)th feature point in the back projection image to judge the *i*th feature point.

(4) Otherwise, we calculate the numbers of nonzero points and zero points on the line between the (*i-1*)th feature point and the (*i+1*)th feature point in the back projection image, and judge the *i*th feature point is the finger tip or the finger root.

(5) Repeat the above four steps until all the feature points (excluding interference points) are distinguished.

D. Marking the thumb and the little finger

After distinguishing the finger tips, we know that the thumb and the little finger are adjacent. According to this characteristic we can distinguish the thumb and the little finger from other finger tips. If not both of the left and right adjacent feature points (excluding the interference points) of one finger tip are finger roots, the finger tip is the thumb or the little finger.

Thus, we obtain the position and the numbers of finger tips and finger roots, and distinguish the thumb and the little finger of the hand gesture.

IV. GESTURE UNDERSTANDING

There are six degrees of freedom in the 3D space. They are the respective movement along the X-axis, Y-axis and Z-axis, and the respective rotation around the X-axis, Y-axis and Z-axis. So this paper has six virtual degrees of freedom according to the gestures.

If *P(x, y)* is the centroid of the initial gesture, *PR(xr, yr)* is the centroid of the gesture in the previous frame, *PN(xn, yn)* is the centroid of the gesture in the current frame, *Num* is the number of feature points(including the interference points), in the conditions of $|PN - P| > T4$ and $|PN - PR| > T5$ we understand the gestures as follows:

(1) When $Num < 7$, for example, the fingers all hold together. If $xn - x > T6$, then the virtual movement right along the X-axis happens. If $x - xn > T6$, then the virtual movement left along the X-axis happens. If $yn - y > T6$, then the virtual movement up along the Y-axis happens. If $y - yn > T6$, then the virtual movement down along the Y-axis happens.

(2) When $Num \geq 7$, we distinguish the thumb and the little finger, and calculate the distance between the thumb and the little finger *l*. In the condition of $l \geq T7$, if $xn - x > T6$, the virtual rotation clockwise around the X-axis happens, if $x - xn > T6$, the virtual rotation anticlockwise around the X-axis happens, if $yn - y > T6$, the virtual rotation clockwise around the Y-axis happens, if $y - yn > T6$, the virtual rotation anticlockwise around the Y-axis happens.

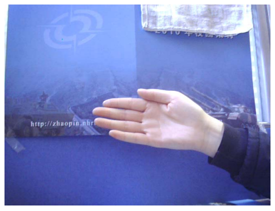
(3) When $Num \geq 7$, we calculate the distance between the thumb and the little finger *l*. In the condition of $l < T7$, if $xn - x > T6$, the virtual movement forward along the Z-axis happens, if $x - xn > T6$, the virtual movement backward along the Z-axis happens, if $yn - y > T6$, the virtual rotation clockwise around the Z-axis happens, if $y - yn > T6$, the virtual rotation anticlockwise around the Z-axis happens.

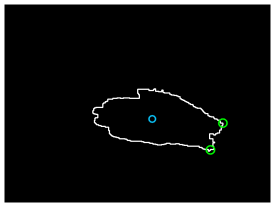
(4) The other gestures are invalid.

In the above discussion *T4*, *T5*, *T6* and *T7* are experiential values. And the steps of each movement and each rotation are set according to the practical demand.

V. RESULTS AND CONCLUTIONS

We have a teapot rendered with OpenGL, and control the motions of the teapot in the six degrees of freedom with the understood gestures, as shown in the Fig.4, 5, 6 and 7. Fig.4 is the initial state. In Fig.5(b) the green circles indicate the feature points, and they are less than seven. We needn't distinguish the finger tips and finger roots. In Fig.6(b) and Fig.7(b), the blue, green and red circles denote the interference points, finger roots and finger tips, respectively, and the red concentric double circles denote the thumbs and the little fingers. All the center circles are the centroids of the gestures.


(a)


(b)

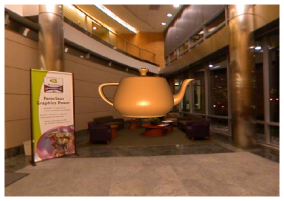
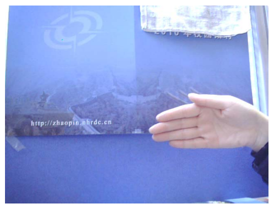
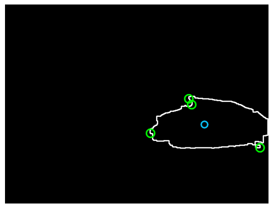

(c)

Figure 4. The initial state.


(a)


(b)

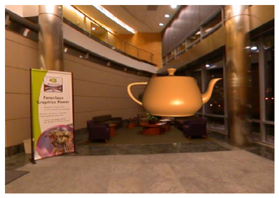
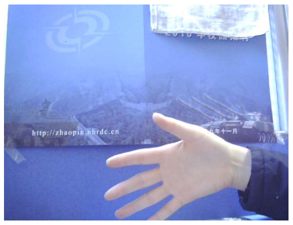
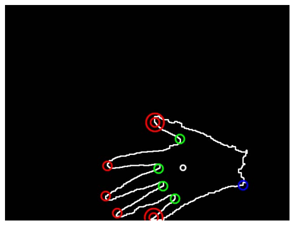

(c)

Figure 5. Move right along the X-axis.

We use an inexpensive USB webcam to capture 640*480 24-color images, and the CPU of PC is 899 MHz. The sampling rate of the USB webcam is 9.3 frames per second. With our algorithm, one frame needs 0.994 seconds averagely, and the recognition rate is about 85%.

To sum up, this paper proposes a six-degree-of-freedom virtual mouse based on hand gestures in complicated background, and experiments show that the virtual mouse has good interactive input functions. The virtual mouse doesn't need difficult gestures, so even the disabled people who cannot bend their fingers at all with severe disabilities can use it. They can put their hands on a desk to just open out or hold together their fingers besides moving their hands.


(a)


(b)

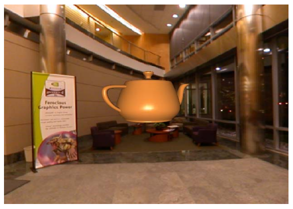
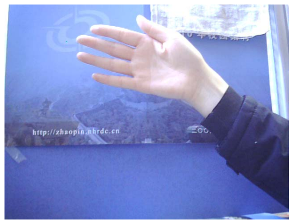
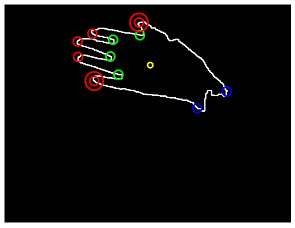

(c)

Figure 6. Rotate anticlockwise around the Y-axis


(a)


(b)

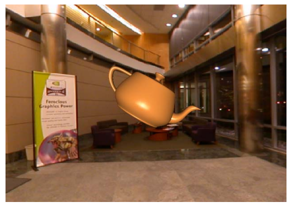

(c)

Figure 7. Rotate clockwise around the Z-axis.

REFERENCES

[1] J. David and Z. David, "A Survey of Glove-based Input," IEEE Computer Graohics & Applications, vol. 4, January 1994, pp.30–39.

[2] Fifth Dimension Technologies, 5DT Data Glove for the Fifth Dimension User's Manual [Z], February 2000

[3] Robert Y. Wang and Jovan Popovic, "Real-Time Hand-Tracking with a color Glove," ACM Transactions on Graphics, vol. 28, no. 3, 2009.

[4] Y. Wu and T.S. Huang, "Hand modeling, analysis and recognition for vision-based human computer interaction," IEEE Signal Processing Magazine, vol. 18, May 2001, pp:51-60.

[5] M.Lee, R. Green and M.Billinghurst, "3D Natural Hand Interaction for AR Applications," Image and Vision Computer New Zealand, 23rd International Conference, 2008

[6] M. Gorman, M. Betke, M. Saltzman, and A. Lahav, "Music Maker —A Camera-based Music Making Tool for Physical Rehabilitation," Boston University Computer Science Technical Report N0.2005-032, 2005.

[7] Intel Corporation, Open Source Computer Vision Library, http://www.intel.com/research/mrl/research/opencv, 2002.