ATLANTIS PRESS

Advances in Engineering Research, volume 104

7th International Conference on Mechatronics, Control and Materials (ICMCM 2016)

Gloved Finger Recognition by Curve Matching

Guangqiang Shu[1, 2, a], Tao Lin[1, b]

[1]School of Computer Science, Sichuan University, Chengdu 610065, China

[2]School of Computer Science, Sichuan Normal University, Chengdu 610068, China

[a]254625466@qq.com, [b]28227265@qq.com

Keywords: gloved finger recognition, contour extraction, curve fitting, curvature integration, similarity assessment, curve matching

Abstract. Finger recognition is an important basis of natural human-computer interaction, but existing researches are mainly on bare-hand finger recognition. Due to the fact that people need to wear gloves because of cold weather, working environment, etc, this paper presents a method to recognize gloved fingers by curve matching. This method firstly extracts contour curve from image edge. After fitting the contour curve, it calculates curvature integration of the fitted curve and then assesses similarity between the curvature integration and finger template's. Finally, it recognizes fingers and positions fingertips according to the assessment. And the finger template which is used to match contour curve can be chosen at random without training. The experiment shows that this method is effective in recognizing fingers and positioning fingertips under the condition of scale variations, rotation variations and affine transformations, whether the subjects wear gloves or not.

1. Introduction

Finger recognition technology is the basis of natural human-computer interaction applications, such as gesture recognition, virtual reality, smart games, identity authentication, etc [1], but existing researches are mainly on bare-hand finger recognition. Due to the fact that people need to wear gloves because of cold weather, working environment, etc, the realization of gloved finger recognition is worth being studied, and it has a good application prospect.

Finger recognition can be realized by various ways. Based on whether the subjects wear devices or not, and based on whether they use multiple sensors or not, the current finger recognition technologies fall into three types. The first type of recognition method needs to wear hardware devices, such as various data gloves, to get finger data. This type of method usually has good accuracy and real-time performance. The biggest problem is its inconvenient use. Based on the first type of method, the second type overcomes the inconvenience, but it still needs multiple sensors to support. For instance, by using the Kinect camera with infrared depth information and by combining it with human skin color model, we can extract the palm and fingers from image [2]. Besides, one can use stereo camera composed of two infrared cameras or three ordinary cameras to get the depth information of image [3], and then positions palm and recognizes fingers. This type of recognition method uses multiple sensors, which is difficult for the deployment of finger recognition system, especially in the portable devices. The third type of recognition method uses only a camera in hardware to obtain image, and then completes finger recognition according to marks, skin colors, shapes and other information. Because of low requirements of hardware, this method based on monocular vision can be easily deployed widely. The method presented in this paper belongs to the third type.

The easiest way to achieve the third type of recognition method is to paste special color markers on the fingers and then find the corresponding color lump from the camera images. Obviously, this method is not only inconvenient in application, but also is easily interfered by ambient light. Similarly, much literature shows that skin colors are used to position palms and recognize fingers [4]. Because of drastic changes of skin colors among different races and the influence of luminous environment on camera imaging, the finger recognition method based on skin colors has a poor

recognition effect in many situations. Besides, in case that people wear gloves, this method will be entirely ineffective. In order not to use color information, S. Kim et al use the Active Shape Models algorithm to detect fingers [5]. While this method needs to train finger set and mark its features to obtain the search template. In the searching process, it needs initially positioning. And the edge of the target fingers finally obtained is not accurate.

In order to recognize gloved fingers by only using monocular camera, we present a method to recognize fingers and position fingertips by matching the image contour curve with the curve of finger template. This method is characterized by not using of skin color information, no need of training for finger template, and no need for initial positioning in the searching process. The specific recognition process is: firstly, calculating image edge and extracting contour curve; secondly, using polynomial approximation to fit the contour curve and calculating the invariant features of curvature integration; thirdly, assessing the similarity between the invariant features of the contour curve and the finger template's, and then recognizing fingers and positioning fingertips according to the assessment.

The rest of this paper is as follows. The second part calculates image edge and presents a method to extract image contour curve. The third part fits the contour curve and calculates curvature integration of the fitted curve; the fourth part assesses similarity between the invariant features of the contour curve and the finger template's, and then matches the two curves. The fifth part finishes recognition experiments and analyses the results. The last part concludes the whole paper and gives the future research plan.

2. Contour extraction

Image contour curve is extracted from image edge. Many operators can be used to calculate image edge. The classic operators of edge detection include Robert operator, Sobel operator, Prewitt operator, etc. These operators are sensitive to noise. The real images in application are usually with complex edges and uneven illumination. Thus using these operators to detect edges has a poor effect. Problems, such as edge blur, weak edge loss, discontinuity of the overall edge, occur frequently. On the basis of summarizing these detection methods, Canny obtains strong and weak edge arrays by setting high and low thresholds. Then he connects the weak edge to the strong edge. By doing so, he gets an edge detection operator with good connectivity [6]. Recently, C. Topal et al propose an edge drawing algorithm which connects extreme anchors of image edge. This algorithm obtains a more continuous edge which is single pixel wide [7]. However, the edge drawing algorithm is easy to be influenced by local extreme point of image edge. Compared with Canny operator, this algorithm obtains an edge which is not better in keeping image contour. Thus, we should use the classic Canny operator in image edge detection.

The image edge obtained by Canny operator covers many details, such as image texture, color blocks, etc. These details will interfere with image contour curve extraction. For computers, it is difficult to find image contour from complicated edge lines. Inspired by that object can continue to move because of its inertia, we try to go on the original curve changing tendency in the searching process, in the hope of getting the correct contour curve. The specific steps are as following. We firstly determine contour curve changing direction according to current and previous contour points. Then, we search for the edge point which is most close to the changing direction around current point and make it as next new contour point. The new contour point and the original current contour points form a new changing direction. After that, we keep searching along the new contour changing direction until we reach image border or until we can't find edge points. Thus, we get one contour curve of the image.

For above process to work, we encode eight neighboring pixels around current contour point. We encode the right-most pixel of the contour point as number 0. After counterclockwise rotating, these pixels are in turn encoded as number 1, 2, 3, 4, 5, 6, and 7. These encodings show neighboring eight directions around the current contour point. And we calculate contour changing tendency according to these encodings. Meanwhile, so as to calculate the next contour points' coordinates according to

the current contour points' coordinates, we need to record the differences between these pixel points' coordinates and current contour points' coordinates. They are in turn: (0,1), (-1,1), (-1,0), (-1.-1), (0,-1), (1,-1), (1,0), (1,1). The whole encoding diagram is shown in Table 1. In the program code, these differences will be put into an array, and the array subscripts correspond to their encodings. Here, we call the array encoding array.

Table 1 The encoding diagram of current contour point's neighboring pixel points.

3:(-1,-1)	2:(-1,0)	1:(-1,1)
4:(0,-1)		0:(0,1)
5:(1,-1)	6:(1,0)	7:(1,1)

Here's the extraction progress of image contour curve. The first step is to calculate the differences between previous contour point's coordinates and current contour point's, and then match it with the encoding array. We thence obtain the matched subscript, which is the encoding of the previous contour point. The positions of this encoding and current contour point's form the line indicating the contour curve changing tendency. Then, we search the next contour point on this line first. At this time, we simply add 4 to this encoding value and obtain a new encoding of the point to be searched. If there is an edge point in this position, we will make it the next contour point and modify it to non-edge point so as to avoid being repeatedly scanned. If there is no edge point in this position, we will respectively add 3 to and subtract 3 from the encoding value of the previous contour point and get two new encodings. They are used to search for the edge points in the direction which is a bit off the line formed by both the previous and current contour points. If there is still no edge point, we need respectively add 2 to and subtract 2 from the encoding value of previous contour point until the value to add and subtract becomes 1. If the edge point failed to find, then the current contour point is the end point of this contour curve. For example, if the encoding value of the previous contour point is 4, then the encoding value we get in the searching process in turn are 0, 7, 1, 6, 2, 5 and 3.

Although the curve obtained by this method might not have been the real image contour curve, this finger recognition method won't recognize these non-finger contour curves as fingers in the recognition process. For example, we completely can't rely on skin colors to recognize the multi-color gloves shown as Fig.1 (a). Thus, we firstly use Canny operator to calculate its edge, and the edge obtained is as shown in Fig.1 (b), which includes the lines of fingers and decorative pattern, etc. Then, we extract the contour from the edge and obtain the contour curve of palm, as shown in Fig.1 (c). This contour curve completely meets the demand of finger recognition. To the rest lines of Fig.1 (b), we can't obtained finger curve from them.

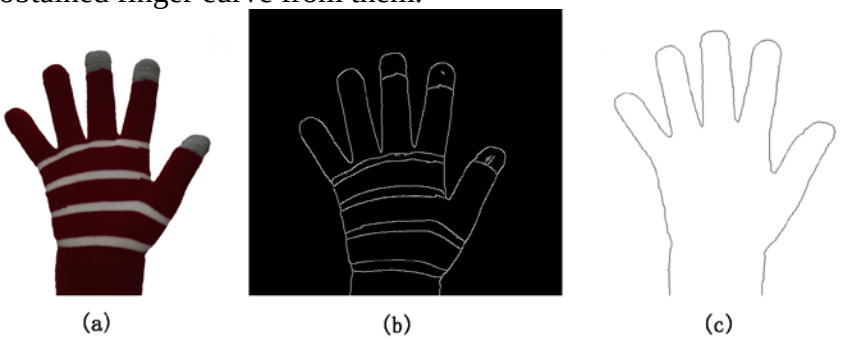

Fig.1 The extraction progress of contour curve (a) a multi-colors glove; (b) the edge image of the multi-colors glove; (c) the extracted contour curve.

3. Curvature integration

After extracting contour curve from image edge, we need to do parametric fitting for it. Interpolation and approximation are two methods for parametric curve fitting. The curve obtained by interpolation method goes through every contour point, while the approximation method only describes the curve shape. The latter is less affected by few interference points. Because of the probably existing serrated lines of image edge, we use approximation method to fit image contour

curve so as not to be affected. For each contour curve, we express it parametric Eq. (1), as shown in the following:

$$\begin{cases} x = \varphi(s) \\ y = \psi(s) \end{cases} \tag{1}$$

Here, *(x, y)*is the point in the contour curve, where s is the curve length, $\varphi(s)$and $\psi(s)$ are fitting functions.

We use polynomial approximation method to fit contour curve. The smaller the polynomial order is, the less smooth fitting curve we get. And the bigger the polynomial order is, the more fierce oscillation the fitting curve produce. The experiment shows that fitted curve obtained by using 20 order can do well in approximating the shape of the original contour curve. For the oscillation segments at both ends of the fitted curve, we just remove them.

Due to the fact that the polynomial approximation method is less affected by local interference points, subtle changes of finger template don't obviously affect the fitted curve. Thus, we can optionally select a finger as a template without any training. This advantage greatly facilitates the design and application of finger recognition system.

After getting the fitted curve of image contour, so as to find the line segments that match the finger template curve in various scales and all kinds of rotation directions, we need to calculate invariant features of curves. Scale Invariant Feature Transform algorithm and Speed Up Robust Features algorithm are two most-famous algorithms to calculate invariant features. However, because the finger contour curve is simple, these two algorithms can obtain few features from it, which is not enough for the needed amount to normally recognize finger. Besides, in case the glove is multi-color glove, features obtained by these two algorithms can't indicate fingers under normal condition. Owning to that curvature integration is invariant to scale and rotation [8], here we take advantage of these invariant features of curvature integration to do curve matching. Curvature *k(s)* in length s of parametric curve can be calculated by Eq. (2):

$$k(s) = \frac{\left|\varphi'(s)\psi''(s) - \varphi''(s)\psi'(s)\right|}{(\varphi'(s)^2 - \psi'(s)^2)^{3/2}} \tag{2}$$

Where ' represents first derivative, and '' represents second derivative.

And we define curvature integration $K(s_1:s_2)$ of a curve from point s_1 to s_2 as:

$$K(s_1 : s_2) = \int_{s_1}^{s_2} k(s)ds \tag{3}$$

Thus, we can express the curvature integration *K(0:l)* of a curve with the length of *l* as:

$$K(0 : l) = \int_0^l k(s)ds \tag{4}$$

Given any scale factor $\alpha > 0$, we can get the curvature integration $\bar{K}(0 : \alpha l)$ of a new curve $\bar{s}$ with a length of αl, as shown below:

$$\bar{K}(0 : \alpha l) = \int_0^{\alpha l} \bar{k}(\bar{s})d\bar{s} = \int_0^{\alpha l} \frac{1}{\alpha} k(s)\alpha ds = \int_0^l k(s)ds \tag{5}$$

From Eq. (4) and Eq. (5), we know curvature integration is invariant to translation and rotation.

Assume the line segment $(k_1 : k_2)$ of curve Γ matches with the line segment $(\bar{k}_1 : \bar{k}_2)$ of curve $\bar{\Gamma}$, we can get:

$$\Gamma(k + \Delta k) - \Gamma(k_1) = \alpha(\bar{\Gamma}(\bar{k} + \Delta k) - \bar{\Gamma}(\bar{k}_1)) \tag{6}$$

$$\Gamma(k - \Delta k) - \Gamma(k_1) = \alpha(\bar{\Gamma}(\bar{k} - \Delta k) - \bar{\Gamma}(\bar{k}_1)) \tag{7}$$

Eq. （6）- Eq.（7）, dividing both side of the equation by $2\Delta k$, and taking limit of them with $\Delta k \to 0$, we can get below equation:

$$\lim_{\Delta k \to 0} \frac{\Gamma(k + \Delta k) - \Gamma(k - \Delta k)}{2\Delta k} = \lim_{\Delta k \to 0} \frac{\bar{\Gamma}(\bar{k} + \Delta k) - \bar{\Gamma}(\bar{k} - \Delta k)}{2\Delta k} \tag{8}$$

That is,

$$\dot{\Gamma}(k) = \alpha\dot{\bar{\Gamma}}(\bar{k}) \tag{9}$$

Here $\dot{\Gamma} = ds / d\int k$, $\dot{\Gamma}$ removes the influence of initial point position of the matching line segment in the curve. In the process of curve matching, we directly use $\dot{\Gamma}$ value to match.

To cut down the amount of calculation, we sample $\dot{\Gamma}$ value Because of oscillation in the beginning and ending of the fitted curve obtained by polynomial approximation, we remove respectively 5% of oscillation segment in two ends of the finger template fitted curve. Then we average the curvature integration value to eight parts and calculate the corresponding $\dot{\Gamma}$ value of each part. Here, we call curvature integration value of each part as unit curvature integration. And we use the unit curvature integration to sample all the curvature integration of image contour curve and calculate the corresponding $\dot{\Gamma}$ value of each curvature integration part.

As regards with the multi-color gloved palm contour curve shown in Fig.1 (c), we use 20 order polynomial approximation to fit the parametric contour curve, as shown in Fig.2 (a). Its curvature is shown in Fig.2 (b), and its curvature integration is shown in Fig.2 (c). Fig.2 (d) shows the sampled data of $\dot{\Gamma}$ value. In this picture, s represents the curve length, and k represents curvature, while $\int k$ represents curvature integration.

4. Similarity assessment

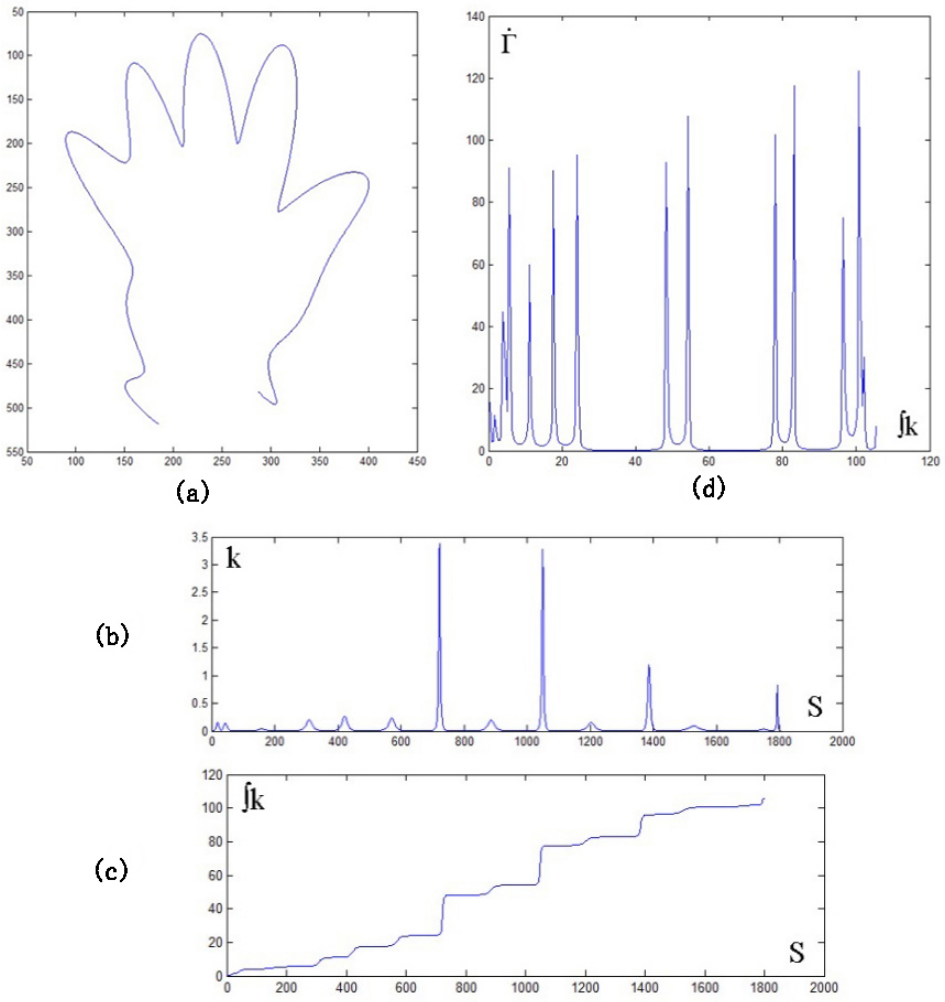

Fig.2 The calculating process of contour curve curvature integration (a) the fitting curve of image contour; (b) the curve curvature; (c) the curvature integration; (d) the sampled data of $\dot{\Gamma}$ value.

After working out the sampled data of image contour curve features and finger template curve features, we use the sampled data of finger template to do sliding matching with the sampled data of image contour curve. The match window's length equals to the length of the finger template sampled

data. Each time the match window slides, we calculate a similarity assessment value. This value reflects the level of similarity between the corresponding lines in image contour curve and finger template curve.

From Eq. (9), we can see that scale factor is one of the influential factors when we use $\dot{\Gamma}$ value to do curve matching. We use the normalized cross-correlation method to assess the similarity between two sampled data [9]. This method is not affected by scale factor. The equation to calculate the normalized cross-correlation of two curves is as below:

$$v(u) = \frac{\sum_{i\in\Omega}[f(i)-\overline{f}][t(i-u)-\overline{t}]}{\sqrt{\sum_{i\in\Omega}[f(i)-\overline{f}]^2[t(i-u)-\overline{t}]^2}} \tag{10}$$

Here, t represents the finger template curve, which slides on the image contour curve f like a window. While u represents the offset between finger curve and contour curve. $\overline{t}$ is the average value of the whole finger template curve. $\overline{f}$ is the average value of the sliding window on the contour curve. The value range of *v(u)* is between -1 and 1. The bigger the *v(u)* value is, the better the matching is. And the corresponding line segment is more likely a finger.

Based on experiments, we set 0.76 as the threshold to assess the similarity of fingers. If the value is lower than this threshold, the line segment in the corresponding matching window does not belong to finger. Once the value is higher than 0.76, we can use *u* value to position the specific locations of the starting and ending of the segment line in the image contour curve which matches the finger template curve. Because of the symmetry of finger template curve, we can easily find the corresponding fingertip position in the image contour curve.

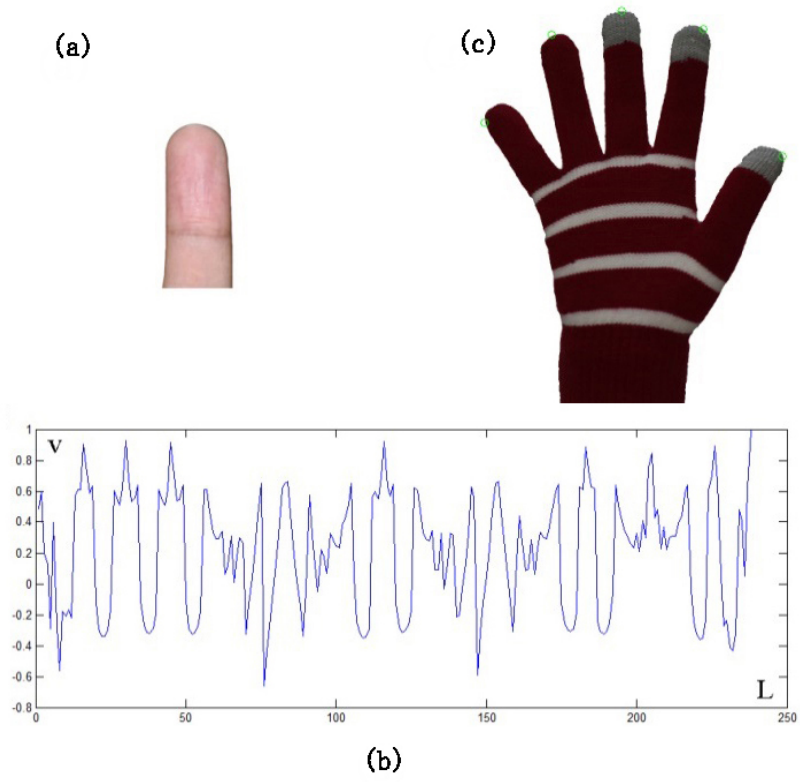

Fig.3 The matching result of finger template and multi-color glove (a) the finger template; (b) the similarity assessment value; (c) the recognition result of multi-color glove.

Due to that the concave curve of two adjacent fingers is similar to the finger curve in shape, we need further judge the similarity assessment point we obtained is a real finger or the concave curve of two adjacent fingers. We use below methods. If the first assessment point equals to the similarity threshold value, we will directly regard it as a finger. If the second assessment point which equals to the threshold value is less four times the width (that is the corresponding widths of 32 unit curvature integrations) of the finger template sampled data than the first assessment point, we think it the concave contour of two adjacent fingers and do not mark it. If there is an assessment point which is

not marked, then we regard its next assessment point meeting the threshold value requirement as finger, irrespective of the distance between it and the previous point. Because the assessment point which is not marked is regarded as the concave contour of two adjacent fingers, the next assessment point meeting the threshold value requirement is supposed to be a finger.

Fig.3 shows the matching result between finger template and multi-color glove. Fig.3 (a) is the finger template selected optionally, its calculating process of feature data is the same with multi-color glove. Fig.3 (b) is the normalized cross-correlation calculating result of the sampled data between multi-color glove contour curve and finger template. Where L represents the number of sliding windows and v represents similarity assessment value. Because of the oscillation of fitted curves, the similarity assessment values of each two match windows in the initiation and terminal can't be used. For the convenience to position the corresponding location of detected fingers in original image contour curve, we don't remove them. Fig.3 (c) is the final recognition result of the multi-color glove. In it, the green circles refer to the detected fingertip positions.

5. Recognition experiment

To verify the effectiveness of the gloved finger recognition method presented in this paper, we use multi-color glove, pure-color glove and bare hand images with different colors to do experiment, as shown in Fig.1 (a), Fig.4 (a) and Fig.4 (b). We firstly resize the three images to 800×800, 400×400 and 200×200 size to do scale change experiments, and use the image with 400×400 size to do follow-up experiments. They are:

(1) Respectively clockwise and counterclockwise rotating the image to 120 degrees and finishing the rotation change recognition experiment.

(2) Based on the consistence of frontal and rear finger contour curve, doing an experiment to recognize fingers from the back of hand.

(2) Due to the fact that the fitted curve obtained by polynomial approximation method is not very sensitive to subtle changes of image contour curve, tilting image at some angles and doing the affine transformation experiment.

Fig.4 The experimental subject of finger recognition (a) a pure-color glove; (b) a bare hand; The green circles show the detected fingertip locations.

This experiments show that the method we present can almost correctly recognize and position fingers in the image. There is a false positive and a false negative only. The reason for false positive is that the false-positive point is too far away from the assessment point meeting the recognition threshold value (bigger than the corresponding data width of 32 unit curvature integrations). The false-negative reason is corresponding finger similarity assessment value is less than 0.76.

In order to observe the finger recognition ability of this method in complex background, we put fingers into common background, as shown in Fig.5. Fig.5 (a) is the recognition result of bare fingers, Fig.5 (b) is the recognition result of gloved fingers in noise background, the green circles show the detected fingertip locations. We can see that after using Canny operator to remove the noise and extract the edge of image, all the fingers are correctly recognized.

(a) (b)

Fig.5 The result of finger recognition (a) the bare fingers; (b) the gloved fingers in noise background.

Compared with the Active Shape Model [5], the method proposed in this paper directly matches the finger template curve on the edge of image contour, does not need initially positioning of finger regions, does not exist the problem of that the edge of recognized finger is not accurate. Experiments show that this method can effectively recognize bare fingers and gloved fingers in image.

6. Conclusion and future work

The gloved finger recognition method based on curve matching presented in this paper is effective in recognizing bare and gloved fingers. This method overcomes the inconvenience of the recognition methods which use data gloves and multi-sensors. And it solves the problem of the recognition method based on skin colors, which can't recognize gloved fingers. Because the background of an image can be very complex, the contour curve obtained by this method may not be correct sometimes. And the concave lines between fingers can't be distinguished directly. All of these problems need to be further studied in the future.

The significance of the gloved finger recognition by curve matching is that, if people wear gloves in natural human-computer interaction, or in case that the ambient light changes drastically, the recognition method depending on skin colors fails completely. At this point, the method presented in this paper is an effective solution.

References

[1] D. Rempel, M. J. Camilleri, and D. L. Lee, The design of hand gestures for human–computer interaction: Lessons from sign language interpreters, International Journal of Human-Computer Studies. 72(2014)728–735.
[2] Fabio Dominio, Mauro Donadeo, Pietro Zanuttigh. Combining multiple depth-based descriptors for hand gesture recognition. Pattern Recognition Letters. 50(2014)101-111.
[3] C. Lyon, Touching the Void: Direct-Touch Interaction for Intangible Displays, Sigchi Conference on Human Factors in Computing Systems. 312(2010)2625-2634.
[4] S. Bilal, R. Akmeliawati, M. J. E. Salami, and A. A. Shafie, Dynamic approach for real-time skin detection, Journal of Real-Time Image Processing. 10(2015)371-385.
[5]S. Kim, Y. Park, K. Lim, H. Lee, S. Kim, and S. Lee, Fingertips Detection and Tracking based on Active Shape Models and an Ellipse, Tencon IEEE Region 10 Conference. 2009, pp. 1-6.
[6]J.Canny, A computational approach to edge detection, IEEE Trans Pattern Analysis and Machine Intelligence. 8(1986)679-698.
[7]C. Topal and C. Akinlar, Edge Drawing: A Combined Real-Time Edge and Segment Detector, Journal of Visual Communication and Image Representation. 23(2012)862-872.
[8]M. Cui, J. Femiani, J. Hu, and P. Wonka, A Razdan. Curve matching for open 2D curves, Pattern Recognition Letters. 30(2009) 1-10.
[9]A. Nakhmani and A. Tannenbaum, A New Distance Measure Based on Generalized Image Normalized Cross-Correlation for Robust Video Tracking and Image Recognition, Pattern Recognition Letters. 34(2013) 315-321.