2012 12th International Conference on Quality Software

Fingertips Detection Algorithm Based on Skin Colour Filtering and Distance Transformation

Yun Liao[1], Yuxiang Zhou[2], Hua Zhou[1] and Zhihong Liang[1]

1 - Yunnan University Key Laboratory of Software Engineering of Yunnan Province, Kunming 650091, China

2- Imperial College, Exhibition Road, London, SW7 2AZ, United Kingdom

Abstract — **Multi-Fingertip location algorithm is always difficult and hot in Finger-based human-computer interaction systems. There are two major difficulties in this field: 1) obtaining accurate hand binary image; 2) locating fingertips in hand binary image. This article presented a multi-fingertip real-time track and location algorithm based on skin colour and distance transformation. The algorithm consists of the following steps. First, it use elliptical boundary model to detect skin colour of human hand in YCbCr colour spaces. After that, we use distance transformation to filter finger from hand, leaving palm area only. Meanwhile, it uses zero and first moment to calculate center of gravity of palm. Then the initial position of fingertips and finger-roots can be located by pixels of hand edge to the center of gravity of palm. Last, it accurately locates fingertips according to the position relationship between fingertips and finger-roots. Experimental results show fingertips can be located quickly and accurately by the algorithm, which fully meets the requirements of real-time computing tasks.**

Keyword - Skin colour detect; Distance transform; Fingertip location

I. INTRODUCTION

Vision-based human-computer interaction systems are always a research hotspot in recent years [1, 2, and 3]. A finger contains large amount of information during information exchange process, so it plays an important role in the field of the human-computer interaction system. Reference [4] refers there are mainly four applications in finger interaction techniques: sign language recognition, gesture recognition, finger writing and virtual touch. The key point is to obtain finger status information in these applications. It is possible to obtain status information of fingers by web camera with the continuous development of image processing, machine vision, artificial intelligence and other disciplines. However, the state of fingers changes during their movement (up to 27 degrees of freedom [5]), and vision itself is ill-posedness due to human hand is a multi-joint non-rigid object. So is vision-based finger interaction one of the greatest challenging tasks. Hand segmentation and fingertip location are key technologies.

II. RELATED WORK

Reference [6] searches fingertips in binary image based on the following two conditions: 1) Fingertip center is surrounded by a circle which fills the foreground pixels. The diameter of the circle is determined by the width of finger. 2) Define a search rectangle out of the circle which surrounds fingertip. The search rectangle includes two parts, which are no pixel part and foreground pixels part, whereas no pixels part is bigger than foreground pixels part. Reference [7] firstly defines a search area based on the size of palm and the relationship between palm and arm. Then it finds 20 candidates fingertips positions in the search area. Last, it locates accurate fingertips in the 20 candidates' fingertips position. Reference [8] calculates curvature of each pixel of hand contour, and then ensures the pixel as a fingertip if the curvature is bigger than a threshold value. Reference [4] calculates convex hull of hand contour, and then locates fingertip by the convex hull.

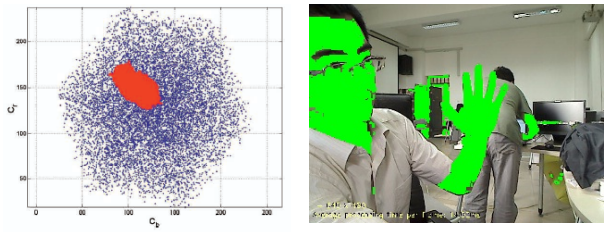
Overall, those algorithms of fingertips location usually have certain requirements on the number of fingers. In addition, there are more constraints on the application environment. So, it is hard to use in real environment. In order to solve the above problems, this article presented a multi-fingertip real-time track and location algorithm in complex environment. The algorithm consists of following steps. First, it use elliptical boundary model to detect skin colour of human hand in YCbCr colour spaces. After that, we use distance transformation to filter finger from hand, leaving palm area only. Meanwhile, it uses zero and first moment to calculate center of gratuity of palm. Then initial location fingertips and finger-roots position can be located by the hand edge of the pixel distance from the center of gravity of palm. Last, it accurately locates fingertips according to the position relationship between fingertip and finger-root. Experimental results show fingertips can be located quickly and accurately by the algorithm, which fully meets the requirements of real-time computing tasks.

III. SKIN COLOUR DETECTION

Skin colour detection is the process of pickup pixels corresponding to the human skin in image. In past years, researchers proposed many algorithms in skin colour detection field, based on numerous data such as colour, texture and multi-information fusion. Among these, the algorithm base on colour is widely used in various systems because it is simple, high real-time and robust when the shape of target is changed. Reference [9] assumes that chromaticity component has no relationship with luminance component in skin colour. It extracts 2000 skin colour images and 4000 non skin colour images. Then it transforms these colours into 6 difference colour space which are r-g, $CIE-u^*v^*$, $CIE-a^*b^*$, $CIE-xy$, $1-Q$ and $Cb-Cr$. The reference indicates the distribution of skin colour in these space is similar to an elliptical distribution. Based on above theories, it proposes elliptical boundary model of skin colour detection. Figure 1 (a) is the project of skin colour in Cb-Cr colour space.

1550-6002/12 $26.00 © 2012 IEEE
DOI 10.1109/QSIC.2012.62

IEEE computer society



(a) (b)

Figure 1. Skin colour detection in Cb-Cr colour space

With the elliptical boundary model in Cb-Cr colour space, pixel is considered as skin colour if values satisfy equation (1) and (2).

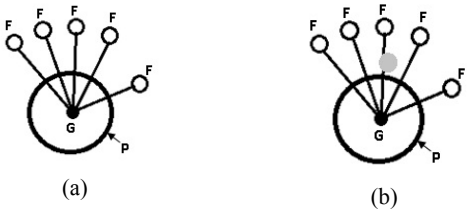
$$\frac{(x - e_{cx})^2}{a^2} + \frac{(y - e_{cy})^2}{b^2} \le 1 \quad (1)$$

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} C_b - c_x \\ C_y - c_y \end{bmatrix} \quad (2)$$

In the above equations, $c_x = 109.38$, $c_y = 152.02$, $\theta = 2.53$ (radian), $e_{cx} = 1.6$, $e_{cy} = 2.41$, $a = 25.39$, $b = 14.03$. Figure 1 (b) is the result of skin colour detection whit this algorithm.

IV. HAND MODEL AND EXTRACTION OF PALM CENTER OF GRAVITY

Hand model is very important for finger based interaction of human-computer interaction systems. The choice of hand model is very closely related with the tasks to be processed. The hand model can be very complex, such as reference [10] creates a complex 3D hand model as a main way to exchange information with computer. The hand model can be also very simple. For example, reference [11] creates a hand model with histogram of the image gradient direction. This article presented a simplified 2D hand model, which is shown in Figure 2(a).



(a) (b)

Figure 2. 2D hand model

The model includes palm P, palm center of gravity G and fingertips F. Fingers only have 3 statuses in this model, which are existence, not existence and fingertip position, so it greatly reduces the computational complexity. We can locate fingertips based on the relationship of position between fingertips and center of gravity. So the key point to calculate fingertips position with this model is to find palm center of gravity.

The usual method to find palm center of gravity is calculating hand center of gravity with all fingers, and using hand center of gravity approximate palm center of gravity. This method will find an incorrect palm center of gravity, as shown in Figure 2(b).

In order to find accurate hand center of gravity, it is very important to filter fingers from hand, because fingers will influence the calculation of center of gravity. This article presented a palm center of gravity location algorithm based on distance transformation. Distance transformation of image is defined as a new image, in which pixels value is set as a distance to the nearest zero pixel in the input image. There are two methods to process distance transformation, called approximate template method and Euclidean distance method, which are based on how to calculate distance.

The approximate template method is presented in reference [12]. The basic idea is to use a template to calculate distance from a pixel to the nearest zero pixel in image. The pixels value in template is the approximation of distance to template center (Figure 3).

4.5	4	3.5	3	3.5	4	4.5
4	3	2.5	2	2.5	3	4
3.5	2.5	1.5	1	1.5	2.5	3.5
3	2	1	0	1	2	3
3.5	2.5	1.5	1	1.5	2.5	3.5
4	3	2.5	2	2.5	3	4
4.5	4	3.5	3	3.5	4	4.5

Figure 3. Approximate template

(a) (b)

Figure 4. Distance transformation

Figure 4 is the transformation results with approximate template method. Figure (a) is the original binary image, and Figure (b) is the gray-scale image which is the result of transformation from (a). From (b) we can find that pixels which are farther away from the hand contour have brighter colour. Fingers are darker than palm internal, because fingers are more slender than palm. So, we can filter fingers by setting the value of pixel as zero if the pixel value is smaller than a threshold. Figure 5 is the result.

(a) (b)

Figure 5. Finger filter result

In Figure 5, (a) is the hand binary image; (b) is the filtered result image. We can calculate palm center of gravity with zero and first moment based on (b). $MM_{\infty} = \sum_i \sum_j I(i,j)$ The formulas are following:

$$MM_{\infty} = \sum_i \sum_j I(i,j) \tag{3}$$

$$MM_{10} = \sum_i \sum_j iI(i,j) \tag{4}$$

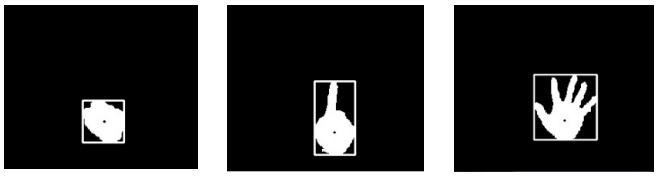
$$MM_{01} = \sum_i \sum_j jI(i,j) \tag{5}$$

$$i_c = \frac{MM_{10}}{MM_{00}}, j_c = \frac{MM_{01}}{MM_{00}} \tag{6}$$

I(i, j) is the value of pixel which coordinates ***(i, j)***, (i_c, j_c) is the coordinate of palm center of gravity. The algorithm to locate palm center of gravity is listed below:

TABLE I. LOCATION PALM CENTER OF GRAVITY BASED ON DISTANCE TRANSFORMATION ALGORITHM

Algorithm (1)
Step1. Get binary image of hand by equation (1) and (2).
Step2. Transform binary image with approximate template method, then get distance transformed result as Figure 4 (b).
Step3. Binarysation the distance transformed result to filter fingers as Figure 5(b).
Step4. Calculation palm center of gravity with formulas (3) to (6)



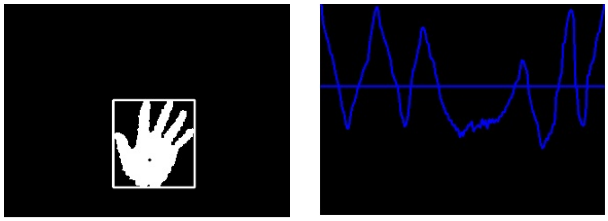
(a) (b) (c)

Figure 6. Location palm center of gravity

Figure 6 is the execution result of the algorithm above. The white areas in Figure 6 are hands and the black nodes in white areas are palm center of gravity. From result above, we can see the position of center of gravity is accurate regardless of the number of fingers.

V. FINGERTIPS LOCATION BASED ON PALM CENTER OF GRAVITY

First, we observe 2D hand model in Figure 2. Fingertips points F have a significant feature, which is different from that of the other points in hand binary image, is the distance from fingertips point to center of gravity is relatively longer than its surrounding pixels. If it uses the distance from a pixel on hand contour to center of gravity as axis Y and uses the index of pixel on hand contour as axis X in Cartesian coordinate system, the pixel index and its distance to center of gravity will form a hand contour distance image, as shown in Figure 7:



(a) (b)

Figure 7. Hand contour distance image

Figure 7 (a) is the binary image of hand, and Figure 7 (b) is the distance image which is formed with the distance from polygon vertices of hand contour to center of gravity and the index of polygon vertices on hand contour. Because the hand contour is closed, the right half wave crest at left border is joined with the left half wave crest at right border in Figure 7 (b). We can find 5 waves crest in Figure 7 (b), the points of which crest may be fingertips, and the wave trough may be finger roots. So, we can locate fingertips by finding the pixels which are wave crests in Figure 7 (b).

A. Trough location algorithm

The trough has characteristics in Figure 7 (b) as follow:

- Trough must be lower than the average distance.
- Extending from the trough to the left and right sides, the wave line will be intersect with a straight line which is formed by average distance between P_1 and P_2 .The projection of this two points at horizontal direction are a and b. The trough must be the lowest in (a, b), shown in Figure 8.

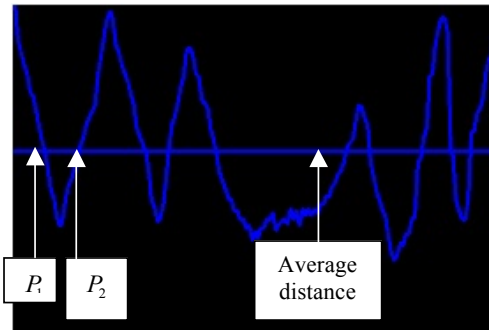


Figure 8.

We can find finger roots based on the above two properties. The algorithm is listed as follow:

TABLE II. THE ALGORITHM OF LOCATION FINGER ROOTS

Algorithm (2)

Suppose:

- '**StartPoint**' is the start point to be searched.
- '**CurrPoint**' is current point processing point.
- '**TroughPoint**' is the trough point which is to be found
- '**AvgDist**' is the average distance.
- Function '**GetNextPoint(p)**' is used to get next point of p on hand contour.
- Function 'GetPointDist(p)' is used to calculate distance from point p to palm center of gravity

Algorithm:

```
BottomPoint = CurrPoint=GetNextPoint(StartPoint)
While CurrPoint <> StartPoint Begin
  If GetPointDistence (CurrPoint) < GetPointDistence (BottomPoint) Begin
    BottomPoint = CurrPoint
  End Else Begin
    If GetPointDistence(BottomPoint) < AvgDist Begin
      find the suspected trough
    End Else If (find the suspected trough) and GetPointDistence(CurrPoint) >= AvgDist Begin
      find the trough
      Break
    End
  End
End
```

B. Wave crest location algorithm

The wave crest has characteristics in Figure 9 (b) as follow:

- Wave crest must be higher than the average distance.
- Extending from the wave crest to the left and right sides, the wave line will be intersect with a straight line which is formed by average distance at P_1 and P_2 .The projects of this two points at horizontal direction is a and b. The wave crest must be the highest in (a, b).

We can find fingertips based on the above two properties. The algorithm is listed as follow:

TABLE III. THE ALGORITHM OF LOCATION FINGERTIPS

Algorithm (3)

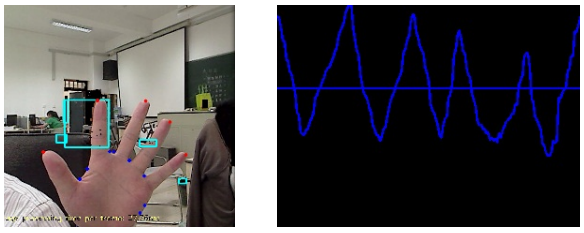
Suppose:

- '**StartPoint**' is the start point to be searched.
- '**CurrPoint**' is current point processing point.
- '**CrestPoint**' is the crest point which is to be found
- '**AvgDist**' is the average distance.
- Function '**GetNextPoint(p)**' is used to get next point of p on hand contour.
- Function 'GetPointDist(p)' is used to calculate distance from point p to palm center of gravity

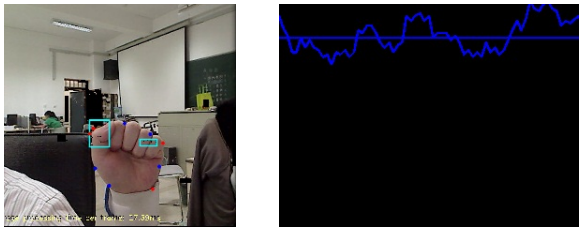
Algorithm:

```
CrestPoint = CurrPoint=GetNextPoint(StartPoint)
While CurrPoint <> StartPoint Begin
  If GetPointDistence (CurrPoint) > GetPointDistence (CrestPoint) Begin
    CrestPoint = CurrPoint
  End Else Begin
    If GetPointDistence(CrestPoint) > AvgDist Begin
      find the suspected crest
    End Else If (find the suspected crest) and GetPointDistence(CurrPoint) <= AvgDist Begin
      find the crest
      Break
    End
  End
End
```

Figure 9 is the result of fingertips location which is using algorithm listed in table Ⅱ and table Ⅲ. The red points are fingertips and the blue points are finger roots in Figure 9.



(a) (b)



(c) (d)

Figure 9. Initial location of fingertips

From Figure 9, we can find that only using distance transformation to location fingertips is inadequate. The algorithm is very likely to locate error position if there is no finger out, just like Figure 9 (c). The error is due to the lack of obvious wave motion. The hand contour is too close to average distance, so a small crest will be located as a fingertip.

Observation Figure 9 (c) and (d) shows the error fingertips points have common specifications, which are the crests are too close to distance average, so we can use this specification to filter error fingertips. We use formula (7) to filter error fingertips in this article.

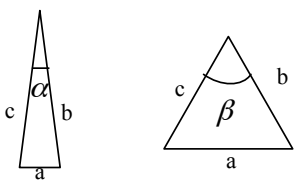
$$\frac{|P_S - P_G|}{D_{avg}} < 0.2 \tag{7}$$

P_S is the coordinate of fingertips, P_G is the coordinate of palm center of gravity, $|P_S - P_G|$ is the distance from P_S to P_G.Figure 10 is the results after using this method. From the figure, we can see that the method filter the error fingertips efficiently. But it also locates an error fingertip in Figure 10 (b).

(a) (b)

Figure 10. Improved fingertips detection

Watching Figure 10 (b), the red point is the error fingertip whereas the two blue points around it are finger roots. Those three points formed a triangle shown in Figure 11:



(a) (b)

Figure 11. Triangle Formatted by fingerroots and a fingertip

The triangle in Figure 11 (a) is formed by a right fingertip and the corresponding finger roots. The triangle of Figure 11 (b) is formed by an error fingertip and the corresponding finger roots. There is a significant difference between the right triangle and error triangle, which is that the angle β in error triangle is bigger than the angle in right triangleα . So we can use the formula (8) to judge whether suspected tips are right fingertips. The suspected tip is a fingertip if the formula (8) is true, or not.

$$\cos\alpha = \frac{b^2 + c^2 - a^2}{2ab} > K \tag{8}$$

α is an angle corresponding to fingertips. b and c are two adjacent borders ofα . a is the opposite border ofα ,.K is a threshold value.

There is another problem in Figure 10. Watching at Figure 10 (b), the suspected fingertip and the corresponding finger roots formed an oblique triangle, in which the distance from fingertip to the roots is different. The left finger root has longer distance than right finger root to fingertip, and the left finger root has obvious errors. This problem is due to the trough may be not finger root. We find that the shorter distance to fingertip, the more accurate finger root is. So we can use the trough as finger root which has shorter distance to fingertip, and find a new point in other finger edge as other side finger root which has the same distance to the fingertip. Through the above steps we can accurately get fingertips and finger roots. The algorithm is listed as follow:

TABLE IV. PRECISE FINGERTIP LOCALISATION

Algorithm (4)

Step1. Using algorithm (1) to locate suspected fingertips.

Step2. Using formula: $\frac{|P_S - P_G|}{D_{avg}} < 0.2$ to filter small crest.

Step3. Calculate finger root position.

1) Calculate the distance from fingertip to both sides of the trough: D_L (The distance to left trough); D_R (The distance to right trough).

2) If $D_L < D_R$ Begin

Let the left trough as left finger root.

Find a new point P on fingertip right. Let the distance from P to fingertip is equal to D_L :($D_p = D_L$).

Point P is the right finger root.

End Else Begin

Let the right trough as right finger root.

Find a new point P on fingertip left. Let the distance from P to fingertip is equal to D_R:($D_p = D_R$).

Point P is the left finger root.

End

Step4. Filter suspected fingertips by formula:

$$\cos\alpha = \frac{b^2 + c^2 - a^2}{2ab} > K$$

VI. THE EXPERIMENTAL RESULTS AND ANALYSIS

Figure 12 is the result of fingertips location which uses algorithm (4). From Figure 12, we can see that the fingertips location is very accurate and the fault location of fingertips does not exist.

(a) (b) (c)

Figure 12.

VII. CONCLUSION

Accurate location of the fingertips is a challenging task in terms of the difficulties to get hand accurate binary image and fingertips location in hand binary image. This article presented a multi-fingertip real-time track and location algorithm based on skin colour and distance transformation. The algorithm consists of the following steps. First, it use elliptical boundary model to detect skin colour of human hand in YCbCr colour spaces. After that, we use distance transformation to filter finger from hand, leaving palm area only. Meanwhile, it uses zero and first moment to calculate center of gravity of palm. Then the initial position of fingertips and finger-roots can be located by pixels of hand edge to the center of gravity of palm. Last, it accurately locates fingertips according to the position relationship between fingertips and finger-roots. Experimental results show fingertips can be located quickly and accurately by the algorithm, which fully meets the requirements of real-time computing tasks.

ACKNOWLEDGMENT

This project is completed by our group including Yun Liao, Yuxiang Zhou, Prof. Hua Zhou and Prof. Zhihong Liang. Yun Liao proposed and designed the major algorithm, which is implemented and verified by Yuxiang Zhou. In addition, Prof. Hua and Prof. Liang have provided helpful suggestions and well supported this project.

REFERENCES

[1] K. D. Guo, M. M. Zhang, C. Sun, Y. Li and X. Tang, "3D Fingertip Tracking Algorithm Based on Computer Vision", *Journal of Computer Research and Development*, 2010(06): P.1013-1019.

[2] D. U. Klaus and D. Schmalstieg, "Finger Tracking for Interaction in Augmented Environments", *Proc of the 2nd ACM/IEEE Int Symp on Augmented Reality (ISAR'01)*, Washington, DC: IEEE Computer Society, 2001: P.29-30.

[3] Y. L. Li and Y. J. Yu, "Vision-based finger spatial location detecting methods", *Computer Engineering and Design,* 2010(03): P.555-558.

[4] S. B. Xiang, G. D. SU, X. D. Ren, Q. Q. Ji and F. Fang, "Embedded implementation of real-time finger interaction system", *Optics and Precision Engineering*, 2011(08): P.1911-1918

[5] J. M. Rehg and T. Kanadc, "Visual Tracking of high DOF articulated structures: An application to human hand tracking", *In Proc of the 3rd Euro pean Conf. on Computer Vision*, 1994: P.35-46.

[6] C. V. Hardenberg and F. Berard, "Bare-Hand human-computer interaction", *In: Proc. of the ACM workshop on Perceptual User Interface,*. 2001: P.15-17, http://conferences.cs.ucsb.edu/PUI/

[7] K. Oka and Y. Sato. "Real-Time Fingertip Tracking and Gesture Recognition", *Proc of IEEE Computer Graphics and Applications,* 2002: P.2 64-71.

[8] X. Y. Wang, X. W. Zhang and Z. Guo, "An Approach to Tracking Deformable Hand Gesture for Real-Time Interaction", *Journal of Software,* 2007(10): P.2425-2433.

[9] J. Y. Lee and D. I. Yoo, "An elliptical boundary model for skin colour detection", *In Proc. of the International Conference on Imaging, Systems, and Techonlogy*, Las Vegas, USA, 2000: P.579-584

[10] S. P. Jiao, Y. C. Bai and F. F. Zen, "Hand Modeling in Virtual Reality", Journal of Shanghai Jiaotong University, 1998(10).

[11] G. Bradski, B. L. Yeo and M. Minerva, "Gesture for video content navigation", *SPIE 3656 (Proc. of the IS&T/SPIE Conf. on Storage and Retrieval for Image and Video Database VII),* San Jose, California, 1999: P.230-242.

[12] A. Rosenfeld and J. L. Pfaltz, "Digital Picture Processing", 1966: P.471-494.