International Journal of Multimedia and Ubiquitous Engineering
Vol.10, No.4 (2015), pp.413-428
http://dx.doi.org/10.14257/ijmue.2015.10.4.39

An Integrated Approach of Real-time Hand Gesture Recognition Based on Feature Points

Yingying She, Yunzhe Jia, Ting Gu, Qun He and Qingqiang Wu*

Software School, Xiamen University, Xiamen, China
**Corresponding author: wuqq@xmu.edu.cn*

Abstract

Hand Gesture recognition systems enable people to interact with digital systems naturally. Due to the spread of body motion capture device, depth information is available for getting more delicate and effective gesture recognition results. However, due to the limitation of devices such as Microsoft Kinect, it is still very difficult to obtain hand gesture information in rela-time. This paper proposes an integrated approach of real-time hand gesture recognition based on feature points. It explains our solutions for hand segmentation and feature points abstraction based on real-time motion captured images. Having been tested with a series of applications, our method is proved to be robust and effective, and suitable for further extension in real-time hand gesture recognition systems.

Keywords: *HCI, Hand gesture recognition, Feature points*

1. Introduction

Hand gesture recognition is an important topic in HCI (Human Computer Interaction). Since the hand-based interaction is the natural way of human machine interactions, it has a very wide range of application scenarios. The traditional approach of processing hand gesture interaction is mainly based on RGB optical images. However, the limitation of lacking depth information reduces the precision of gesture recognition. With the popularity of motion capture devices such as the Microsoft Kinect, the cost of depth information capturing has been greatly reduced. Although Microsoft provides the library called Kinect SDK to support application development using the Kinect, it treats the whole hand as one single point. In this paper, we present our study on the process of hand gesture recognition using Microsoft Kinect. This approach combines skeletal-based and morphological-based gesture recognition methodologies, abstracts hand regions from captured images, and positions feature points in order to support the tracking and recognition of hands in real-time.

2. Related Work

There are many approaches associated with the accuracy of hand gesture recognition. Zhou Ren [1] focused on building a robust part-based hand gesture recognition system using the Kinect sensor. In that system, Finger-Earth Mover's Distance (FEMD) was used to measure the dissimilarity between hand shapes in order to abstract the hand shape. HiaathamHasan [2] explored a multi-layer neural network-based approach to recognize the hand gestures. Javier Molina [3] used static and dynamic models to get a real-time user independent hand gesture. Radhikacentre [4] presented a computer vision method to recognize the hand gesture from the image captured by a webcam. Wang C [14] presented a hand gesture recognition algorithm by using the depth and skeleton from Kinect.

ISSN: 1975-0080 IJMUE

Since the insufficient hands motion capture data is delivered by the Microsoft KinectSDK, two major issues are the keys to success in hand gesture recognition: One is the hand region segmentation, and another is hand feature points positioning.

In the processing of hand region segmentation, most techniques are relied on color spaces to identify hands. Dan Xu [5] proposed a method which can locate hands simultaneously in real-time by using skin-color detection and K-means in conjunction with stereoscopic depth information, and using a YCbCr color space filter to locate hand regions. However, this method has usage limitations due to the premise of this method, the entire human body should be captured and the whole skeleton should be tracked at the same time. The histogram color-based image threshold can be used to detect skin on the human body, and use the GMM model to segment human hand regions [6]. Zhong Yang et al introduced the HSV color space skin filter [8]. Antonis A. Argyro *et al.* [9] used a Bayesian classifier which is boot strapped with a small set of training data to detect skin-colored objects [10-11]. No matter what color spaces are used, the color-based methods are sensitive to noises. In addition, the morphological filtering technique can be used to effectively remove background and object noise from binary images [7]. In this paper, we improve the techniques of eliminating noise interference by combining color and depth information in the hand segmentation process.

Hand feature points provide the basic hand gesture tracking and recognition information. In terms of positioning feature points, traditional approaches mainly focus on looking for the center of a maximum inscribed circle [12]. Jagdish L Raheja *et al.* located the palm center by using distance transformation. However, the result of this method is not stable as we illustrate in the experimental result (section 4.2). After analyzing images captured by Kinect, we extend the idea presented [11] by positioning feature points based on morphological operations. The Graham Scan is used in [10] to find the convex hull of a hand including the fingertips, but this process is not always precise due to the capture device factors and dynamic hand movements. Raheja, J. L. *et al.* assumes that the fingertip is the point with minimum depth in each finger point cluster [13]. However, the limitation of this method is it cannot precisely conduct fingertips when they are blending. In this paper, we propose a depth image based algorithm to solve the recognition problems on blending fingers. In the experimental section (section 4.3), we provide the comparison of a traditional binary image based algorithm and our algorithm.

3. Methodology

Hand region segmentation and feature point extraction plays an important role in the hand gesture recognition process. As seen in figure 1, there are six steps in the hand gesture detection pipeline. These 6 steps have been embed into a hand gesture recognition system. In this paper, we explain a detailed implementation of step 1 to 5 in the following sections.

3.1. Background Separation, Hand Region Extraction and Blob Clustering

The purpose of background separation is to extract the foreground target region and eliminate background interference from the original images. The background interference refers to noise points that are not in the hand region. In our project, in order to segment hand regions, we assume that hands are the closet objects to the Kinect. Based on the OTSU threshold method, we extract the targeted hand regions from depth images [15]. However, since the depth value of hands and nearby obstacles is similar, the OTSU method classifies

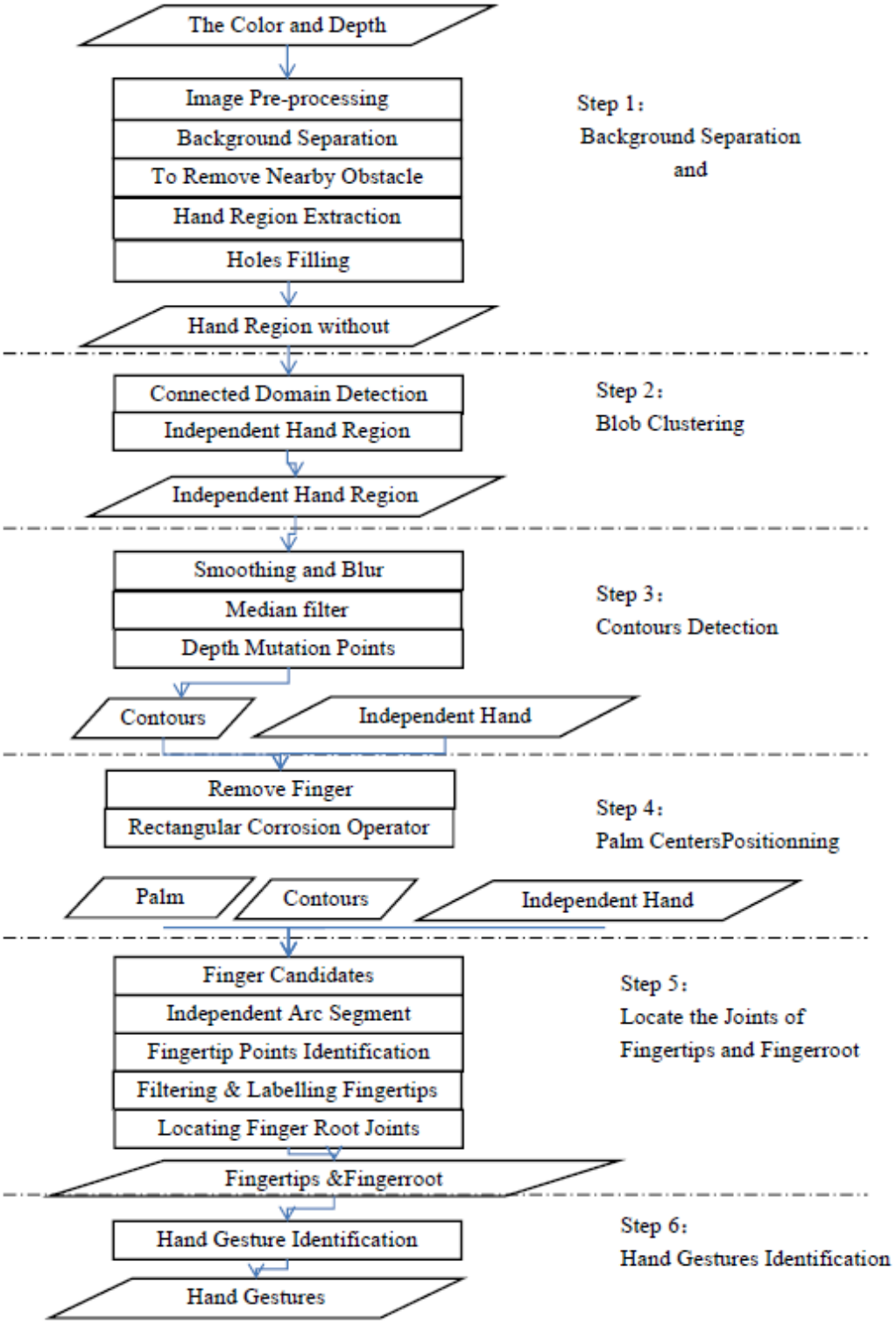


Figure 1. The Procedure of Hand Gesture Detection

these noises as the foreground targeted hand region, there are further actions to be made on skin-color filtering to separate hand regions and other obstacle noises. In addition, we fill in holes in hand regions based on the connected domain partition in the hands region.

After hand region extraction, we get the region for all hands. We have to separate each hand region in order to have accurate hand gesture recognition. We propose a clustering algorithm based on DBSCAN [16] (Density-Based Spatial Clustering of Applications

with Noise) to do clustering. A connected domain is an independent hand region, and it is an independent cluster. We use depth value and the hand points in hands region to obtain clusters.

There are holes in the hand region. In the depth images that we get from Kinect, points in these holes have a similar depth as points in the hand region. A connected domain is an independent hand region, and it is an independent cluster. In the hand region M, it contains points and realted depth value. In the hand region clustering algorithm, for a point p in M, q is the neighbor point of p, if the difference in depth value of p and q is less than ε, then we consider there is an edge between p and q. If in the depth image, there exists a path from p to q, then we consider *p_depth* and *q_depth* to belong to the same connected domain.

Based on the scattered feature of skin-color obstacles, we consider the ratio of pixel numbers in an obstacle connected domain and total pixel numbers in connected domains are related less. In order to eliminate obstacles skin-color and noise, we set the ratio of skin-color pixels and the total number of the pixels threshold to η, the minimum point number as *MinPts*. For a connected domain *Connected_i*, if the ratio of *Connected_i.skinPts* and *Connected_i.size* is large than η, and *Connected_i.size* is large than *MinPt*, then we consider the connected domain is a hand region.

Algorithm: Hand Region Clustering

Input: hand points set M, depth map Depth, threshold η, ε
Output: clusters C, each cluster contains points belonging to one hand with depth value

```
i = 1;
mark all points in Depth as unvisited;
mark skin-color points in M as unvisited;
do
    randomly pick one unvisited point p' in M;
    mark p' as visited;
    create new cluster newC_i;
    find p, which is corresponding point of p'in Depth;
    add p to newC_i;
    count = 1;
    Stack.push(p);
    while Stack is not empty
    s = Stack.pop();
        for each point t in 8- neighborhood of s in Depth
            if t is unvisited && abs( t.depth-s.depth)<ε
                mark t as visited in Depth;
                add t to newC_i;
                Stack.push(t);
                find t', which is corresponding point of t in M;
                if t belongs to skin-color points
                        mark t as visited in M
                        count++;
                end if
            end if
        end for
  end while
  if newC_i.size>MinPts &&  (count/newC_i.size)>η
        add newC_i to C;
  else
```


mark points in $newC_i$ as noise;
end if
i = i + 1
until no unvisited skin point in M;
return C;

3.2. Contour Detection

Contour is an important element to express characteristics of gestures. The contour includes the external contour of the whole hand region and the internal contour in hand region. We consider the initial gesture as the open hand status. During the gesture tracking, fingers might bend sometimes. In these cases, we have to use the internal contour to calculate the degree of bending.

After Blob clustering, for every independent hand region, we have to do contour detection. In general, the contour of a hand region that we get from the previous step is zigzag. We have to do an anti-aliased process in order to smooth the contour. The traditional linear filtering algorithm is simple and easy to implement, and the result is good. However, due to the usage of a neighborhood domain average algorithm, it might have the result of edge blur, and affect the subsequent processes. Also, a low-pass filter can effectively remove the high frequency noise in the image. It is very good at removing isolated noises. However, it also removes the high frequency part of the target, and reduces the image clarity. In our project, we use the median filtering algorithm to remove the isolated noise in the image, while maintaining the target edge information.

In order to obtain the complete hand region contour while a finger and palm are overlapping, such as bending fingers, the calculation should also count in the depth value. We set a threshold μ, when the difference of two pixels depth is large than μ, we consider these two points depth mutation points. In order to identify depth mutation points, we define a threshold η. If in a point's 8-neighborhood region, the number of depth mutation points is larger than η, then this point is a depth mutation point.

If there is a point p with value 1 in the M, and q is a point in p's 8-neighborhood region, we have the following calculation,

$$\mathrm{f(p,q)} = \begin{cases} 1 & |p.dept - q.dept| \gg \mu \\ 0 & |p.dept - q.dept| < \mu \end{cases}$$

$$\mathrm{count(p)} = \sum_{|p-q| \le 1} f(p,q)$$

$$\mathrm{Contour} = \{p | \mathrm{count(p)} \ge \eta, \mathrm{M(p)} = 1\}$$

Here is the detail algorithm for finding the hand contour:

Algorithm: Finding the Hand Contour

Input: hand binary image M with depth value, threshold μ, η
Output: hand contour *contour*

set depth of all non-hand points in M to -1;
for each hand point p in M
  count = 0;
  for each point q in 8- neighborhood of p
    if abs(q.depth - p.depth) < μ
    count++;
  end for
  if count > η

add *p* to *contour*;
end for

3.3. Palm Center Positioning

For each hand, we have to find its palm center as one of the feature points. In our project, the palm center positioning algorithm is based on morphology. We adopt the morphology corrosion approach from [7], and use the morphological corrosion operater to process the hand region. The idea is to remove fingers form the hand, and calculate the palm centre based on the leaving pixels. The region of the corrosion operator in the hand area is related to the total number of pixels in the hand. In this paper, we use the rectangular corrosion operator to calculate the palm center. The edge of corrosion operator is

$$a = \sqrt{\frac{M.size}{\varepsilon}} \ . (\varepsilon = 40)$$

The idea is to use the operator a to process the hand region binary image M, and obtain the corrosion result M_1. Then, the center of M_1 is the palm center.

3.4. Fingertip Positioning

3.4.1 Fingertip Candidate

Fingertips have the largest curvature in the finger. So, the way to locate the fingertip is by looking for the point with the largest curvature in the finger. Since it is unable to get the hand contour curve equation, We couldn't get accurate curvature calculations for a point in a finger. We use the geometric feature of points in the finger to get the fingertip candidates.

- First, we set a thresholdθ, a point p_t and vectors $\overrightarrow{p_t p_{t-m}}$ and $\overrightarrow{p_t p_{t+m}}$ in the contour. If the angle γ between these vectors is less thanθ, then the pointp_t is the candidate of a fingertip. Usually, the candidate point set is an arc in the contour.

$$Candidate = \{p_t | p_t \in Contour, < \overrightarrow{p_t p_{t-m}}, \overrightarrow{p_t p_{t+m}} >< \theta\}$$

- If we take two pints p_s、p_t（$s > t$）from candidates,$s - t > \tau$and $\{p_{s+1}, p_{s+2}, \dots, p_{t-1}\} \cap Candidate = \emptyset$, then p_s and $p_t$belong to different arcs. The value of τ is in proportion to the total pixel number of the contour, $\tau = \frac{Contour.size}{\varepsilon}$（$\varepsilon = 40$）.
- After getting the candidate set, we have to do clustering, and obtain independent arc segments. After clustering, if independent fingertip arc C_i , and $p_{i,j}$ which is the jth point in the arc C_i, then candidate fingertips are $Candidate = C_1 \cup C_2 \cup \dots \cup C_n$. In Figure 2, candidate sets are denoted in yellow arcs.

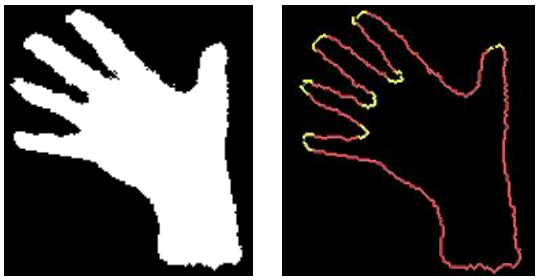


Figure 2. Candidate set

The arc C_i （i=*1,2，…，n*）is a symmetrical arc in genreal, and the fingertip point is the mid-point in the arc. However, since the resolution and dynamic interaction, arcs in captured images are not always symmetrical. In order to get the accurate fingertip coordinate values, we calculate the angle between mid-point and its neighbor points which have a distance *m* within the arc. In total, there are *2m+1* neighbor points corresponding to *2m+1* angles. We choose the point with the smallest angle as the fingertip candidate.

$$D_i = \left\{ p_{i,j} \middle| p_{i,j} \in C_i, \frac{C_i.\text{size}}{2} - m < j < \frac{C_i.\text{size}}{2} + m \right\}$$

$$\text{FingerCandidate}_i = \underset{p_t \in D_i}{\operatorname{argmin}} < \overrightarrow{p_t p_{t-m}}、 \overrightarrow{p_t p_{t+m}} >$$

Here is the algorithm to find fingertip candidates *fingerCandidate*:

Algorithm: Finding Fingertip Candidate
Input: hand contour *contour*, threshold ,
Output: fingertip candidates *fingerCandidate*
1. sort all points in *contour*;
2. for *i* th point in *contour*
   calculate angle γ between$\overrightarrow{p_i p_{t-m}}$、 $\overrightarrow{p_i p_{t+m}}$; (m=10 in our experiment)
   if $\gamma < \theta$
      add p_i to the end of *candidate*
   end for
3. cluster *candidate* into $C_1 \cup C_2 \cup ... \cup C_n$by distance between points in *candidate*;
4. add the point with lowest corresponding in each cluster $C_1, C_2, ..., C_n$ to *fingerCandidate*;
5. return *fingerCandidate*;

3.4.2 Filtering Fingertips

The fingertip candidates include fingertip and finger valleys. We have to filter the finger valleys point in order to get the fingertip points. When a person opens five fingers, the distance between the fingertips and the palm center is longer than the distance from the finger valley to the palm center. So, we design the fingertip algorithm as seen below:

Algorithm: Locating Fingertips and Finger Valleys
Input: finger candidates *fingerCandidate*, palm centre*center*
Output: fingertips points *fingertips*, finger valleys points *fingervalleys*
1. calculate distances between *center* and each point in *fingerCandidate*, storing in *Dis*;
2. calculate *mean*, which is mean of *Dis*;
3. for each p_i in *fingerCandidate*
      if *Dis*[i] < Mean
         add p_i to *fingervalleys*;
      else
         add p_i to *fingertips*;
4. return *fingertips* and *fingervalleys*;

3.4.3. Labeling Fingertips

In order to label fingertips, we assume that all five fingertips have been detected. So, the fingertips size is 5. The fingertips are marked as thumb fingertip (T), index fingertip

(I), middle fingertip (M), ring fingertip (R) and little fingertip (L). When five fingers are open, there are vectors $\overrightarrow{CT}$, $\overrightarrow{CI}$, $\overrightarrow{CM}$, $\overrightarrow{CR}$, $\overrightarrow{CL}$ from the palm centre to the fingertips.

$$\alpha_{<\overrightarrow{CT},\overrightarrow{CT}>} < \alpha_{<\overrightarrow{CT},\overrightarrow{CI}>} < \alpha_{<\overrightarrow{CT},\overrightarrow{CM}>} < \alpha_{<\overrightarrow{CT},\overrightarrow{CR}>} < \alpha_{<\overrightarrow{CT},\overrightarrow{CL}>}$$

Where $\alpha_{<\vec{A},\vec{B}>}$ denotes the intersection angle of $\vec{A},\vec{B}$ The angle between fingertips $\overrightarrow{CT}$ and $\overrightarrow{CM}$, as shown in Figure 3.

The distance between palm center and thumb fingertip and the distance between palm center and little fingertip are smaller than the distances between palm center and the other three fingertips. Therefore, we can get T and L first. Then distinguish them by the assumption that the thumb fingertip is thicker than the little fingertip.

Fingertips $\overrightarrow{CT}$and $\overrightarrow{CM}$

Figure 3. The Angle of $\overrightarrow{CT}$and $\overrightarrow{CM}$

The algorithm of labeling fingertips is described as below:

Algorithm: Labelling Fingertips

Input: fingertips points *fingertips*
Output: labelled fingertips (from thumb to little finger) *fingertips*

1. calculate distances between *center* and each point in *fingertips*, select f_1 and f_2 with lowest and second lowest distance;
2. calculate α、β, former is angle between f_1 and m-*th* ahead and after points in *contour*, using same method in finding fingertip candidate(3.4.1), β is for f_2 as well;
3. if $\alpha > \beta$, label f_1 as T (thumb), otherwise label f_2 as T (thumb);
4. calculate $<\overrightarrow{\text{CenterT}}$、$\overrightarrow{\text{CenterFingertips}_i}>$, where Fingertip s_i are other four fingertips, sorting them in ascend order, then label the fingertips as I (index finger), M (middle finger), R (ring finger), L(little finger) in that order;
5. return labelled *fingertips*;

3.4.4. Locating Finger Root Joints

The finger root joints is defined as the intersection of fingers middle line and the palm, as yellow points in figure 4. The distance between joint and palm center is roughly equal to the distance between the finger valley and the palm center. Define $Ax + B = Y$ is when one of the fingers middle line, *d* is the mean of distances from palm center to five finger valleys. Then the corresponding root $\text{Joints}_i(x_i, y_i)$holds.

$$\text{Joints}_i(x_i, y_i) = \begin{cases} A_i x + B_i = 0 \\ (x - \text{center}.x)^2 + (y - \text{center}.y)^2 = d^2 \end{cases} \qquad i = 1,2,3,4,5$$

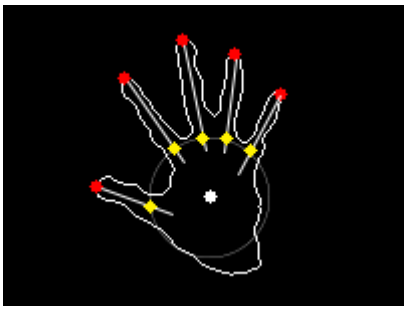


Figure 4. Finger Root Joints

Suppose p_i is a fingertip on the sorted contour, then we choose another n (n is a threshold) pair of points $(p_{i-1}, p_{i+1}), (p_{i-2}, p_{i+2}), \ldots, (p_{i-n}, p_{i+n})$, Then calculate the middle points of all the pairs, $mid_1(t_1, s_1), mid_2(t_2, s_2), \ldots, mid_n(t_n, s_n)$, the middle line of the finger $line_i$ is calculated by

$$line_i = A_i x + B_i \qquad i = 1,2,3,4,5$$

$$\begin{cases} \bar{t} = \frac{1}{n}\sum_{j=1}^{n} t_j \\ \bar{s} = \frac{1}{n}\sum_{j=1}^{n} s_j \\ A_i = \frac{\sum_{j=1}^{n} t_j s_j - n \cdot \bar{t}\bar{s}}{\sum_{j=1}^{n} t_j{}^2 - n \cdot (\bar{t})^2} \\ B_i = \bar{s} - A_i \bar{t} \end{cases} \qquad i = 1,2,3,4,5$$

The whole algorithm is described as below:

Algorithm: Locating Finger joints

Input: fingertips points *fingertips*, palm centre *center*, sorted contour *contour*, finger valleys *fingervalleys*

Output: finger joints *joints*

1.calculate distances between center and each point in *fingervalleys*, then calculate *mean*
of these distances;

2. for each f_i in *fingertips*

get points $mid_1, mid_2, \ldots, mid_n$ (n=30 in out experiment), where mid_j is middle
point of *j-th* ahead and after points of f_i in *contour*;
calculate $line_i$ using Min Square with middle points;
calculate intersection of $line_i$ and $(x - center.x)^2 + (y - center.y)^2 = mean^2$
,
then add it to *joints*;
end for

3. return *joints*;

---

4. Experiments

In these experiment, we test our algorithms in real-time by using Microsoft Kinect captured videos. Figure 5,6,7,8 are captured images from these videos.

4.1. Compare of Depth Image based and Binary Image based Contour Algorithm

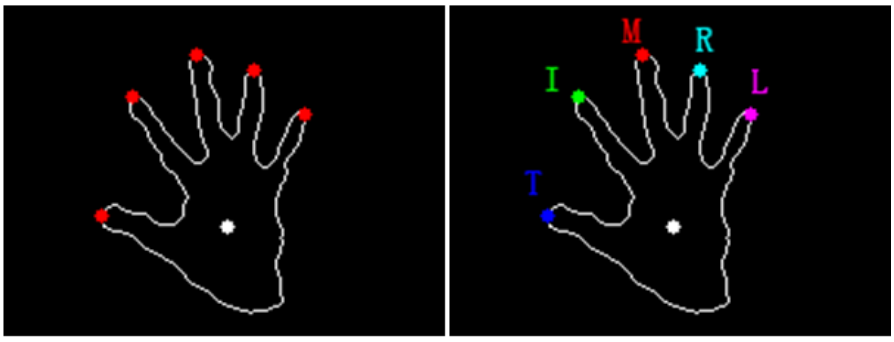
We implement the depth image based algorithm and compare the results with the binary image based algorithm introduce in [17]. In the depth image based algorithm, we set parameter $\mu = 8\text{mm}$, $\eta = 2$. The indicated gestures one, two, three, four and five are selected for the purpose of analysis.

Figure 6 shows the results of the proposed contour algorithm and binary image based contour algorithm. It can be told that the contour generated by the binary image based algorithm is nearly a smooth curve, which is ideal for sorting. Our proposed algorithms can detect the proper contour even when the fingers are bending, but the binary image based algorithm cannot. According to the performance, we choose the binary image based imaging for initial gesture detection and depth image based for further gesture tracking.

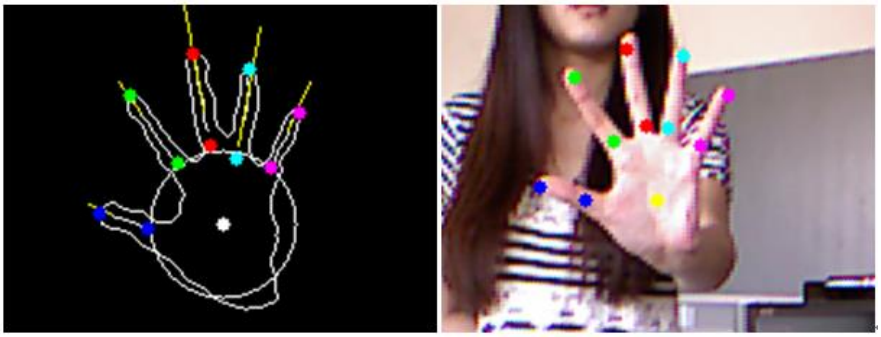
4.2. Comparison of Distance Transform based and Morphological Operation based Palm Center Locating Algorithm

The input of the experiment is the contour (smooth curve) generated by a binary image based contour algorithm. The distance transform based algorithm takes the point with maximum distance value as the palm center, while the morphological based algorithm erodes the finger part, then takes the center point of the left part as the palm center. The indicating five gestures are selected to be captured for analysis. In the morphological operation based algorithm, a (hand.size / 40) × (hand.size / 40) matrix is used as the kernel to perform the operation.

With distance transformations, sometimes more than one point exists with the maximum value. Then the consequent palm center would shock up and down when those points are relatively far away from each other. Due to the deficiency of the hardware, the palm center moves up and down when we hold fingers still. Such problems could be avoided by using a morphological operation based algorithm introduced in section 3.3, and the result of it is good, as shown in Figure 7. In the last column of Figure 7, red point indicates the palm center found by distance transform based algorithm and green points is the result of morphological operation based algorithm



Positioning fingertips (b) Labelling fingertips



(c) Detecting joints (d) Feature points in original image

Figure 5. Result of Labeling Fingertips and Finger Root Joints

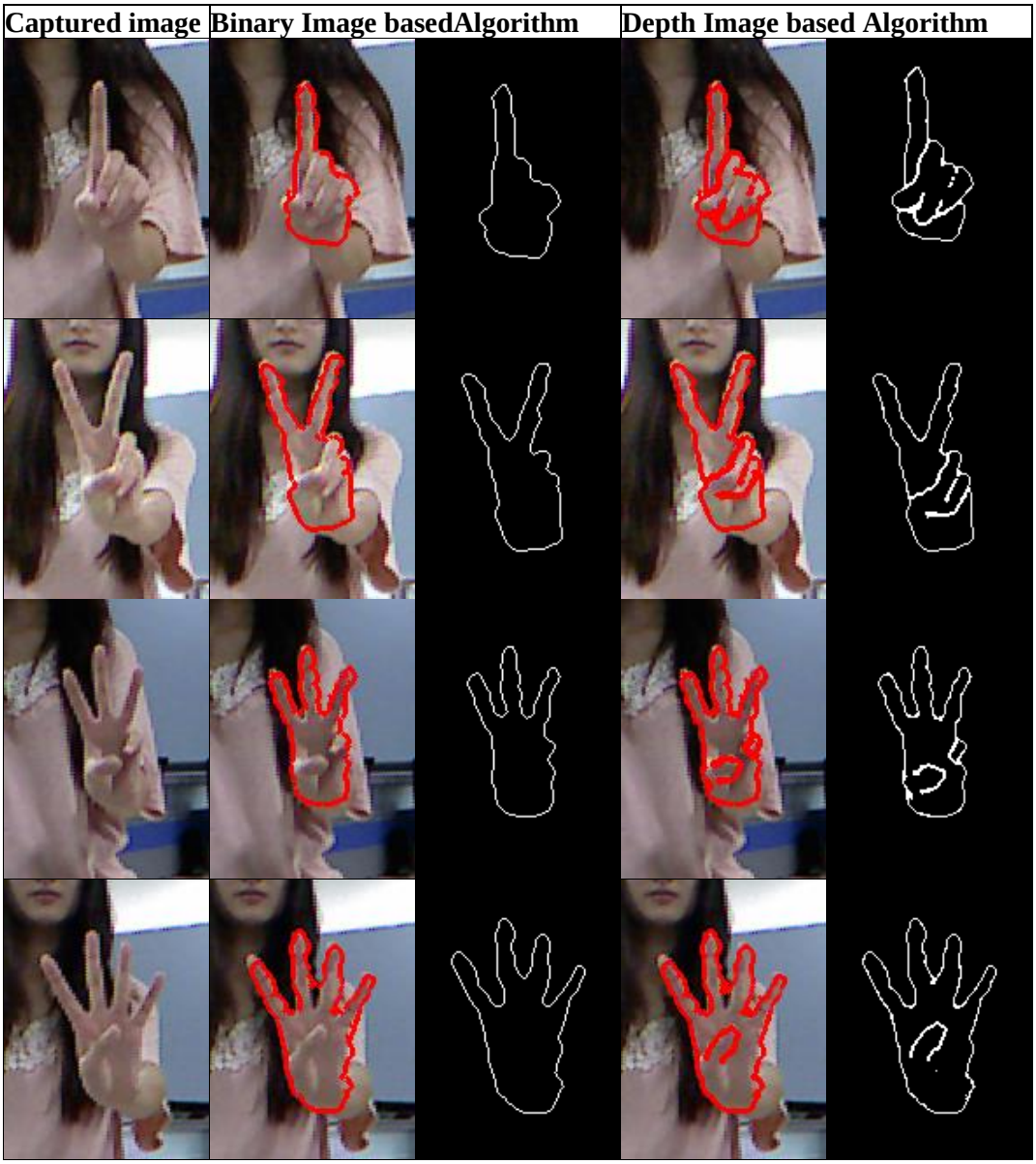


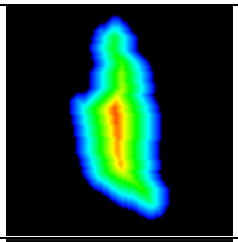
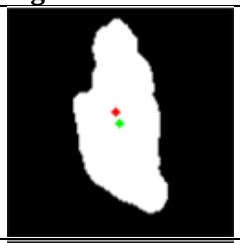
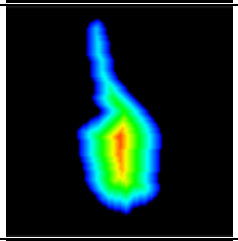
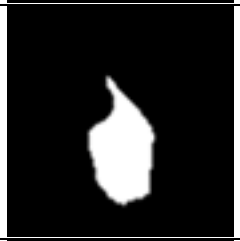
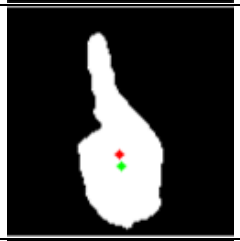
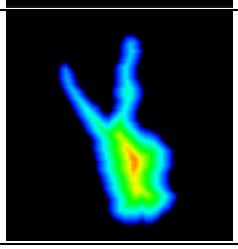
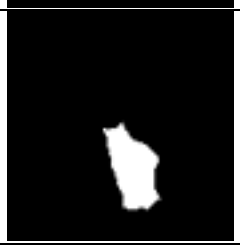
Figure 6. Depth Image based Contour Algorithm vs. Binary Image based Contour Algorithm

4.3. Experiment of Locating and Labeling Fingertips and Joints

Analyzing feature points of hand includes locating fingertips and joints, and labeling them. In this test, we design two test cases. The first test is for locating, and the second one is for labeling fingertips and joints.

In the first test, we detect fingertips in five different gestures; in the second test, labeling is done based on the results of gesture with five fingers open. The parameters m=20 and $\theta = 55°$. Figure 8 denotes the process and results of locating fingertips. The first column is the original captured images from Kinect; the second column is the contour and palm center; the third column shows the fingertips candidate; the last column is the results of fingertips.

The second test denotes the final result of this paper. Figure 5 shows the feature points detected. Based on the result of detecting fingertips, the proposed algorithm finds thumb (blue point in figure 5) first. Then other four fingertips - index finger (green), middle finger (red), ring finger (cyan), little finger (pink) are labeled based on the relative position with thumb. Finger root joints are found with the help of Min Square algorithm in section 3.4.4.

Captured image	Distance transform	Image after erosion	Results of two algorithms
			
			
			

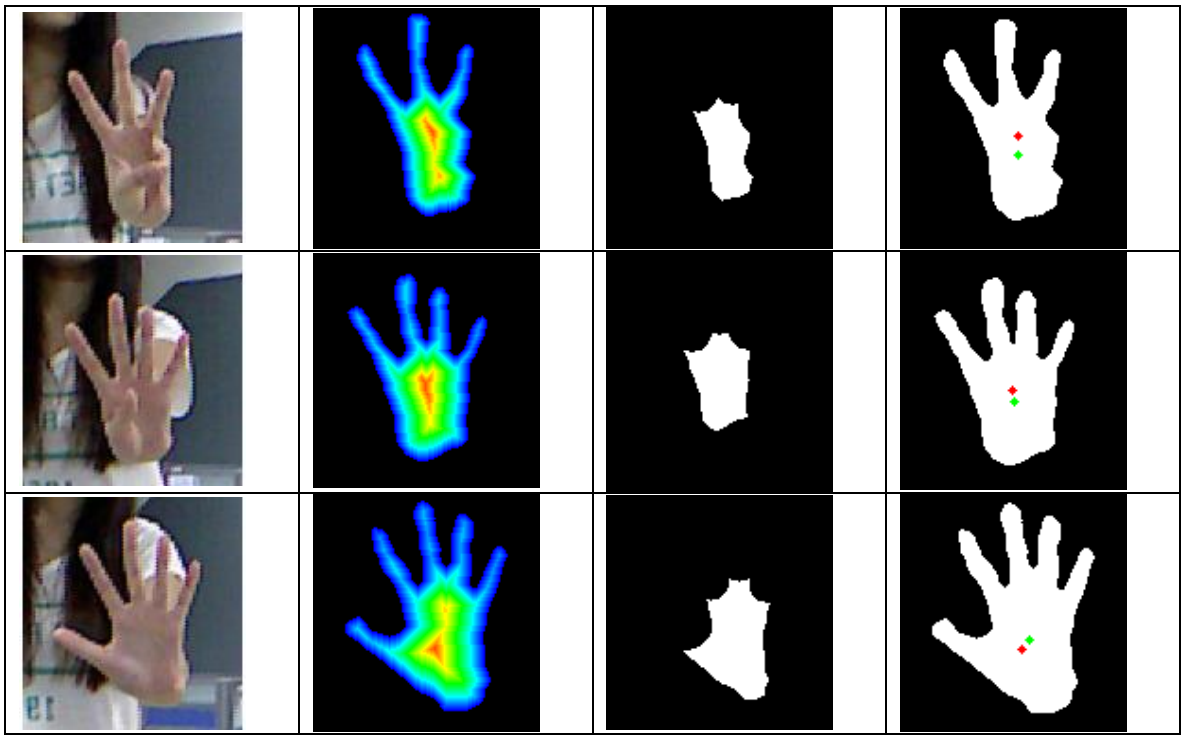
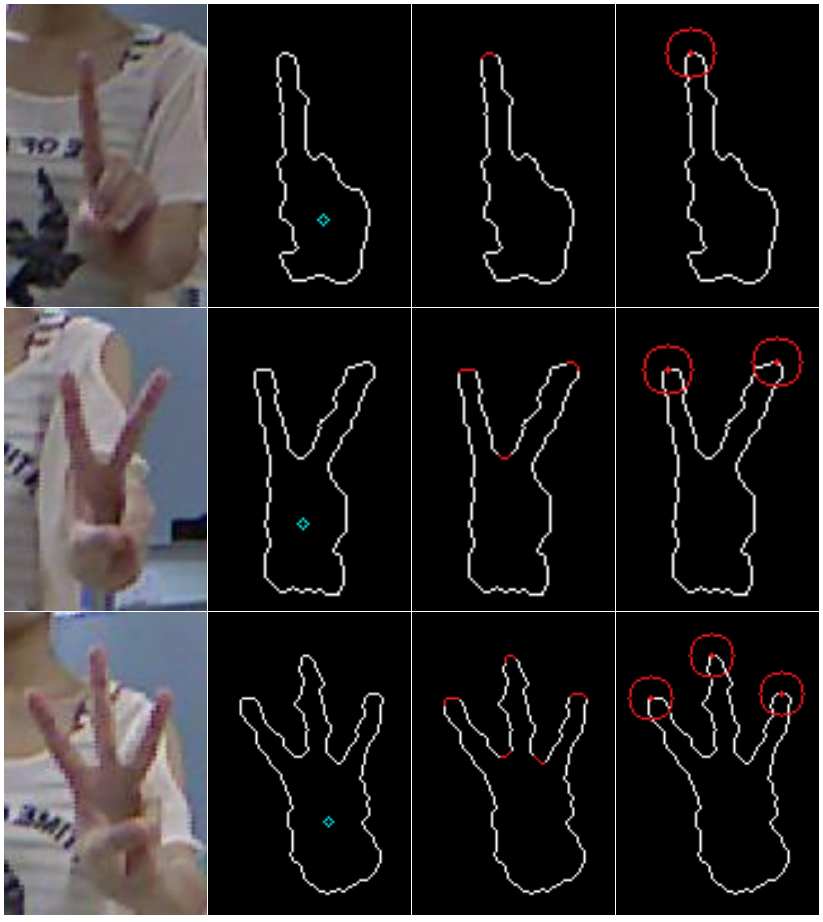


Figure 7. Distance Transform based vs. Morphological Operation based Palm Center Locating Algorithms



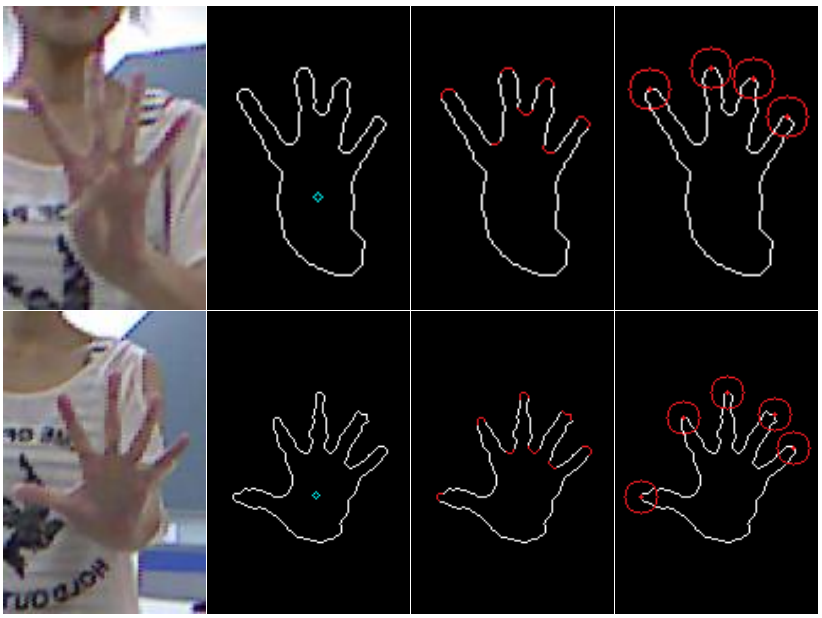


Figure 8. Results of Locating Fingertips

5. Conclusion

Hand gesture recognition technology is an important aspect in NUI (Nature-User-Interaction) study, and can be applied to games, vision enabled robots, from virtual reality to smart home systems.. This paper explains the fundamental hand gesture recognition solution based on feature points. Extends of this work is the study of movement characteristics of hands, and the expression of hand gestures. This approach could be applied in different VR (Virtual Reality) or AR (Augmented Reality) systems, immersive games and other sign language related systems.

Acknowledgements

The authors would like to acknowledge the support of Education game platform based on mobile cloud services (3502Z2014116, principal, technological innovation projects for early-stage companies, Science and Technology Planning of Xiamen, 2014.6-2016.6).

References

[1] Z. Ren, J. Yuan and J. Meng, "Robust part-based hand gesture recognition using kinect sensor", IEEE Transactions on Multimedia, vol. 15, no. 5, **(2013).**
[2] H. Hasan and S. Abdul-Kareem, "Static hand gesture recognition using neural networks", Artificial Intelligence Review, vol. 41, no. 2, **(2014).**
[3] J. Molina, M. Escudero-Viñolo and A. Signoriello, "Real-time user independent hand gesture recognition from time-of-flight camera video using static and dynamic models", Machine vision and applications, vol. 24, no. 1, **(2013).**
[4] R. Bhatt, N. Fernandes and A. Dhage, "Vision Based Hand Gesture Recognition for Human Computer Interaction", International Journal of Engineering Science and Innovative Technology, vol. 2, no. 3, **(2013).**
[5] D. Xu, Y. L. Chen and X. Wu, "Integrated approach of skin-color detection and depth information for hand and face localization", Proceedings of IEEE International Conference on Robotics and Biomimetics, **(2011)** December 7-11; Karon Beach, Phuket.
[6] Y. Ming, Q. Ruan, and A. G. Hauptmann, "Activity recognition from rgb-d camera with 3D local spatio-temporal features"< Proceedings of IEEE International Conference on Multimedia and Expo, **(2012)** July 9-13; Melbourne, Australia.
[7] D. K. Ghosh and S. Ari, "A static hand gesture recognition algorithm using k-mean based radial basis function neural network". Proceedings of the 8th International Conference on Information, Communications and Signal Processing, (**2011**) December 13-16; Singapore.

[8] Z. Yang, Y. Li, W. Chen and Y. Zheng, "Dynamic hand gesture recognition using hidden Markov models", Proceedings of the 7th International Conference on Computer Science & Education, **(2012)** July 14-17; Melbourne, Australia.
[9] A. A. Argyros and M. I. A. Lourakis, "Real-time tracking of multiple skin-colored objects with a possibly moving camera", Proceedings of the 8th European Conference on Computer Vision, **(2004)** May 11-14; Prague, Czech Republic.
[10] Y. Wen, C. Hu, G. Yu, "A robust method of detecting hand gestures using depth sensors", Proceedings of IEEE International Workshop on Haptic Audio Visual Environments and Games, **(2012)** October 8-9; Munich, German.
[11] M. K. Bhuyan, D. R. Neog and M. K. Kar, "Hand pose recognition using geometric features", Proceedings of National Conference on Communications, **(2011)** January 28-30; Bangalore, India.
[12] H. Liang, J. Yuan and D. Thalmann, "3D fingertip and palm tracking in depth image sequences", Proceedings of the 20th ACM international conference on Multimedia, **(2012),** October 29-Movember 2; New York, USA.
[13] C. Wang and S. Chan, "A new hand gesture recognition algorithm based on joint color-depth Superpixel Earth Mover's Distance", Proceedings of the 4th International Workshop on Cognitive Information Processing, **(2014)** May 26-28; Copenhagen, Denmark.
[14] J. L. Raheja, A. Chaudhary and K. Singal, "Tracking of fingertips and centers of palm using Kinect", Proceedings of the 3rd International Conference on Modelling and Simulation, **(2011)** September 20-22; Langkawi, Malaysia.
[15] A. Kurakin, Z. Zhang and Z. Liu, "A real time system for dynamic hand gesture recognition with a depth sensor", Proceedings of the 20th European Signal Processing Conference, **(2012)** August 27-31; Bucharest.
[16] M. Ester, H. P. Kriegel, J. Sander and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise", Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining **(1996)** August 2-4; Oregon, Portland.
[17] S. Suzuki, "Topological structural analysis of digitized binary images by border following", Computer Vision, Graphics, and Image Processing, vol. 30, no. 1**, (1985).**

