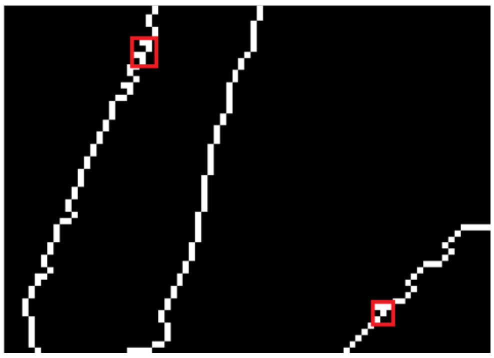
This is an excerpt of the paper *A Low Complex hand segment and recognition method for human-wearable device-interaction using priority-based scan-line stereo matching*, it's about the low complex hand gesture recognition algorithm. URL of the paper: http://imkaywu.com/2013/12/05/Paper.html

The algorithm of hand gesture recognition uses the relative position of each finger with respect to the reference point for recognition. The algorithm has 3 steps: 1). Detect the rough direction of hand and transform the contour points into *time-series curve*; 2). Locate the accurate positions of fingertips and fingerroots (two points located at the root of each finger) which are used to obtain finger directions and hand direction; 3). Recognize hand gestures using Euclidean distance metric.

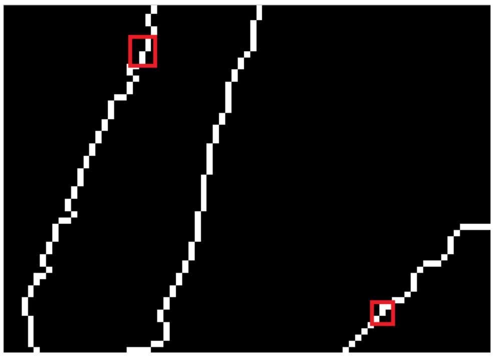
1. Hand direction detection and time-series curve

1.1. Contour points

To find the direction of the hand, the algorithm first detects the contour points. Because the contour is not always smooth after the segmentation, this may render a problem as is shown in figure 1: the points connecting algorithm can run into a dead-end when it runs into the marked points. So it's essential to eliminate the points that have two neighboring background points of the same direction in the first place.



(a). the top-left points should be eliminated



(b). result of the refined contour detection

Figure 1. Contour detection

As for the direction detection, the algorithm uses the relative position between P_i and P_{i+3} ($P_i, P_{i+3} \in P_C$, P_C is the consecutive points in the contour). Therefore there are 16 different directions and 16 direction angles ($\theta_i,\ i \in \{1,2\ldots,16\}$), see figure 2.

$$\mathrm{k}_i = \tan(\theta_i). \tag{1}$$

The algorithm counts the number for each direction N_{d_i} ($N_{d_i} \in N_d, i \in \{1,2\ldots,16\}$), and assigns an index I_{d_i} ($I_{d_i} \in I_d,\ i \in \{1,2\ldots,16\}$) to each direction using the algorithm below.

Algorithm 1: Assign an index for each direction.

```
1.  $N_d = N_d / \sum_{i=1}^{16} N_{d_i}$;
2.  $N_{d_{I_{max}}} = max(N_d)$;
3.  for $i = 1{:}16$
4.          $I_{d_i} = i - I_{max}$;
5.  endfor
```

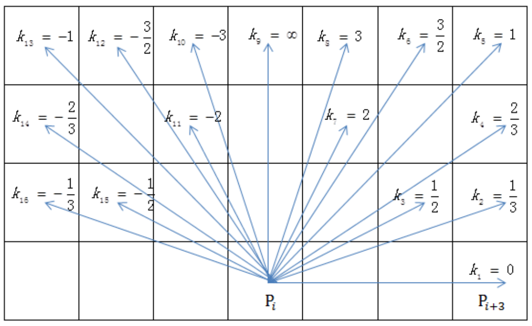


Figure 2. 16 types of direction of neighboring points (each square represents a pixel)

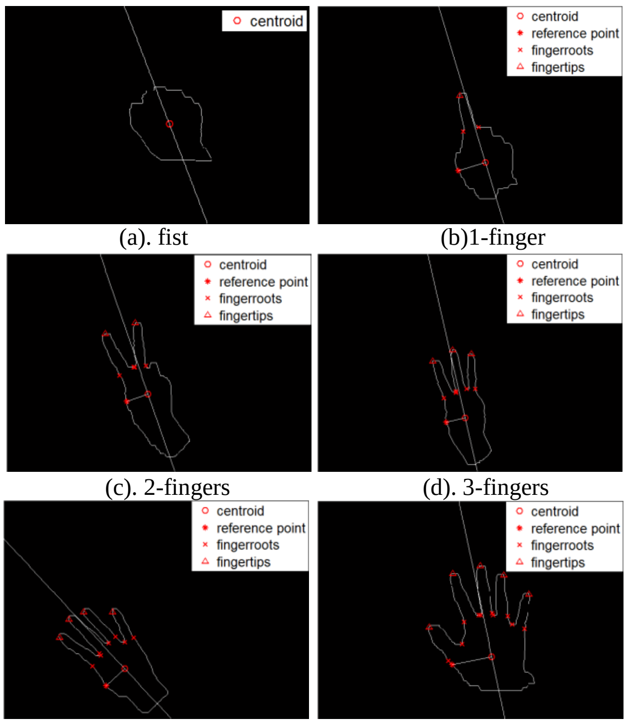
The direction index of the hand I_{hand} is obtained by equation (2):

$$I_{hand} = \sum_{i=1}^{i=16} (N_{d_i} * I_{d_i}). \tag{2}$$

I_{hand} is between two index values I_m and I_n ($m < n, m, n \in \{1,2,\ldots,16\}$), which is used to calculate the hand angle θ_{hand} using (2).

$$\theta_{hand} = \theta_{I_m + I_{max}} + \left(\theta_{I_n + I_{max}} - \theta_{I_m + I_{max}}\right) * (I_{hand} - I_m) \tag{3}$$

This direction of hand is relatively rough, because not all points in the contour contribute to the direction of the hand, thus a refined way of using finger directions to represent hand direction is employed in Section 2.4 for optimal recognition. Figure 3 shows graphs with the hand contour and direction detected.



(a). fist (b)1-finger

(c). 2-fingers (d). 3-fingers

(e). 4-fingers (f). 5-fingers

Figure 3. Hand centroid and direction

1.2. Time-series curve

Once the direction is detected, the algorithm transforms the contour points into a *time-series curve* [1, 2, 3], which serves to extract hand features. Such a shape representation has been successfully used for the classification and clustering of shapes. The *time-series curve* records the relative distance between each contour point to the centroid. The centroid o is found using Distance Transform [5, 6]. The horizontal axis denotes the degree between each contour point and the reference point relative to the centroid, normalized by 360°. The vertical axis denotes the Euclidean distance between the contour points $(x_{contour}, y_{contour})$, $((x_{contour}, y_{contour}) \in P_C)$ and the centroid (x_{center}, y_{center}), normalized by the radius (R) of the maximal inscribed circle.

$$\theta_c = \frac{arctan\,(\frac{y_{contour}-y_{center}}{x_{contour}-x_{center}})}{360^{\circ}}, \quad (4)$$

$$d_c = \frac{\sqrt{(x_{contour}-x_{center})^2+(y_{contour}-y_{center})^2}}{R}. \quad (5)$$

The reference point $r\ (x_r, y_r)$ is perpendicular to the hand direction and is obtained using (6):

$$\frac{y_r-y_{center}}{x_r-x_{center}} = \tan(\theta_{hand} + 90^{\circ}). \quad (6)$$

The *time-series curve* is used to detect the fingertips and fingervalleys of the hand, see figure 5, the locally highest points are detected as fingertips and locally lowest as fingervalleys using the algorithm below.

Algorithm 2: Detection of fingertips and fingervalleys.

```
1.  for $i = 1: length(d_c)$
2.    if($d_c^i < d_c^{min}$)
3.        $d_c^{min} = d_c^i$;
4.        $d_c^{max} = d_c^i$;
5.        if($d_c^{min} < d_c^{fingervalley_{temp}}$)
6.          $fingervalley_{temp} = i$;
7.        endif
8.    elseif($d_c^i < d_c^{max}$)
9.        $fingervalley_j = fingervalley_{temp}$;
10.       $firstTime = \sim firstTime$;
11.       $j = j + 1$;
12.       $d_c^{min} = d_c^i$;
13.       $d_c^{max} = d_c^i$;
14.   elseif($d_c^i > d_c^{max}$)
15.       if($firstTime$)
16.         $fingervalley_{temp} = i - 1$;
17.         $firstTime = \sim firstTime$;
18.       endif
19.       $d_c^{max} = d_c^i$;
20.   endif
21. endfor
```

Line 5-7 is used to change the index of fingervalley from fingervalley1 to fingervalley2 which makes the detection of fingervalleys more accurate, see figure 4. The initial value of the variable $firstTime$ is *true*, it's necessary because it can remain the value of $fingervalley_{temp}$ unchanged while $d_i > d_{max} = true$.

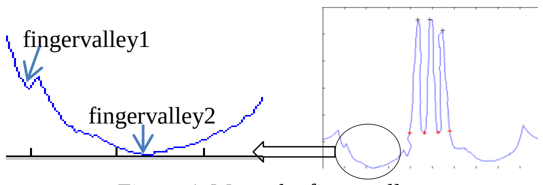
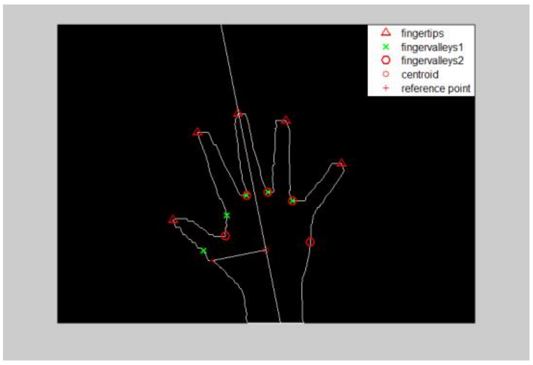
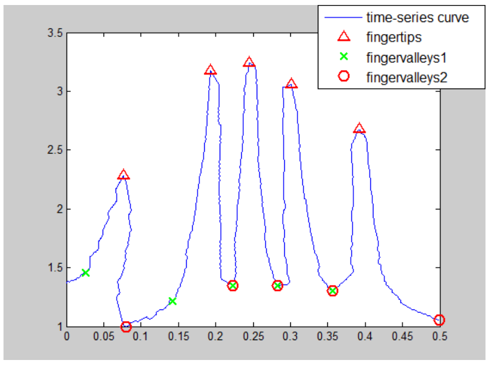


Figure 4. Move the fingervalley

The detected positions of potential fingertips and fingervalleys are shown in figure 6. The algorithm further eliminates the invalid fingertips and fingervalleys using Algorithm 3, the result of algorithm is shown in Figure 5 (b).



(a). Original hand contour with fingertips and fingervalleys detected



(b). Time-series curve with fingertips and fingervalleys detected

Figure 5. Original hand gesture and the time-series curve

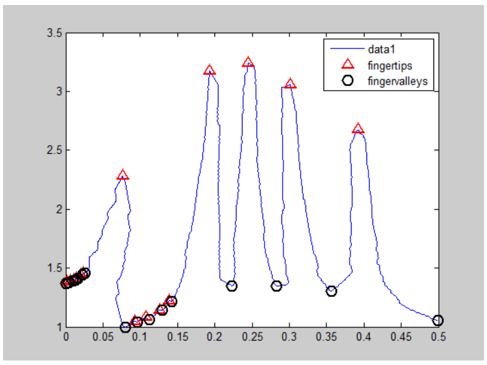


Figure 6. Fingertips & fingervalleys (Initial detection)

```
Algorithm 3: Eliminate invalid fingertips and fingervalleys
1.  for each fingertip $ft_i$
2.      The current positions of fingervalleys are $fv_c$;
3.      while(true)
4.          if($d_c^{ft_i} - d_c^{fv_c} >$ threshold)
5.              Record the positions of fingertips& fingnervalleys;
6.          endif
7.          Calculate $func_{opt}(ft_i, fv_c)$ for $fv_p$;
8.          $func_{opt}(ft_i, fv_{opt}) = max(func_{opt})$;
9.          if($fv_c == fv_{opt}$)
10.             break;
11.         elseif($fv_c$ is changable)
12.             Update fingervalleys' positions $fv_c$
13.         else
14.             break;
15.         endif
16.     endwhile
17. endfor
```

Where $\text{func}_{\text{opt}} = \frac{\left(d_c^{ft_i} - d_c^{fv_p}\right)}{\max\left(d_c^{fv_p}\right)} - 2.5 * \frac{\left|\theta_c^{ft_i} - \theta_c^{fv_p}\right|}{\max\left(\theta_c^{fv_p}\right)}$. fv_p is the potential positions of fingervalleys. For the left fingervalley, $fv_p = \{fv_{1st}, \dots, fv_c\}$, for the right fingervalley, $fv_p = \{fv_c, \dots, fv_{end}\}$.

This is sufficient in most cases, but in others, the result may look like the table 1. It's impossible that one fingervalley belongs to several fingertips, so it's necessary to choose one between these candidate fingertips and eliminate others. The chosen fingertip should maximize Eq. (7).

$$\mathrm{d}_c^{ft} - \frac{\mathrm{d}_c^{fv1} + \mathrm{d}_c^{fv2}}{2}. \tag{7}$$

The information of fingertips and fingervalleys is sufficient to discriminate the number of fingers, yet not accurate enough to recognize gestures due to the relative inaccuracy of the hand direction and reference point. Therefore, a refined way of hand direction detection is proposed next.

Fingertip	5		6		7	
Fingervalley	3	8	4	8	4	8

Table 1. An example of false detection of fingertips

2. Positions of fingertips and fingerroots localization

The algorithm further detects what we call fingerroots (points located at the root of fingers, see figure 3) of each finger using the following procedure:

```
Algorithm 4: Detection of the fingerroots.
1.  while(true)
2.      Detect the direction of finger $\theta_f^i$;
3.      Find the fingerroots $\mathrm{fr}^i$;
4.      Calculate the external rectangular area of finger $A_{Rec}$;
5.      Calculate the area of the finger $A_f$;
6.      $R_A = A_f / A_{Rec}$;
7.      if($R_A > R_{threshold}$)
8.          break;
9.      endif
10. endwhile
```

2.1. Finger direction

The finger direction θ_f^i is calculated using fingertip ft (x_{ft}, y_{ft}) and fingerroots fr_1^i (x_{fr1}^i, y_{fr1}^i) and fr_2^i (x_{fr2}^i, y_{fr2}^i) by equations (8)-(10).

$$\theta_f^i = \frac{\theta_{f1}^i + \theta_{f2}^i}{2}, \tag{8}$$

$$\theta_{f1}^i = arctan(\frac{y_{ft} - y_{fr1}^i}{x_{ft} - x_{fr1}^i}), \tag{9}$$

$$\theta_{f2}^i = arctan(\frac{y_{ft} - y_{fr3}^i}{x_{ft} - x_{fr3}^i}). \tag{10}$$

Where $\theta_{f1}^i, \theta_{f2}^i$ is respectively the direction of the fingertip and the fingerroot.

2.2. Fingerroots detection

After the detection of the finger direction, the algorithm below finds the temporary fingerroots $(x_{fr_{temp}}, y_{fr_{temp}})$ which are perpendicular to the finger direction using (11), (12).

$$\frac{y_{fr_{temp1}} - y_{fr3}^i}{x_{fr_{temp1}} - x_{fr3}^i} = \tan\left(\theta_f^i + 90^{\circ}\right), \tag{11}$$

$$\frac{y_{fr_{temp2}} - y_{fr1}^i}{x_{fr_{temp2}} - x_{fr1}^i} = \tan\left(\theta_f^i - 90^{\circ}\right). \tag{12}$$

Algorithm 5: Detection of the temporary fingerroots.

```
1.  if ($fr_{temp1} \in \{fr_1^i : ft\}$)
2.          $fr_1^{i+1} = fr_{temp1}$;
3.          $fr_2^{i+1} = fr_2^i$;
4.  elseif($fr_{temp2} \in \{ft : fr_2^i\}$)
5.          $fr_1^{i+1} = fr_1^i$;
6.          $fr_1^{i+1} = fr_{temp2}$;
7.  else
8.          $fr_1^{i+1} = fr_1^i + 1$;
9.          $fr_1^{i+1} = fr_2^i + 1$;
10. end
```

The fingerroots and fingertips are used to calculate the area of the finger (A_f) and the external rectangular area (A_{Rec}). This process will be terminated until the area ratio $(R_A = A_f / A_{Rec})$ surpasses a certain threshold.

2.3. Finger and external rectangular area

The area of finger is calculated by counting pixel numbers in the finger region.

As for the external rectangular area, the temporary finger direction is always perpendicular to the line of the fingerroots, so the height as well as the largest width is needed to calculate A_{Rec}.

$$Height = \sqrt{\left(x_{ft} - x_{vp}\right)^2 + (y_{ft} - y_{vp})^2}, \tag{13}$$

$$Width =$$

$$\max\left(\sqrt{\left(x_{contour1} - x_{vp}\right)^2 + \left(y_{contour1} - y_{vp}\right)^2}\right) + \max\left(\sqrt{\left(x_{contour2} - x_{vp}\right)^2 + \left(y_{contour2} - y_{vp}\right)^2}\right). \tag{14}$$

In (13), (x_{ft}, y_{ft}) is the coordinate of the fingertip, (x_{vp}, y_{vp}) is the vertical point between the fingertip and the line of the fingerroots. In (14), $(x_{contour1}, y_{contour1}) \in S_{C_1}$, S_{C_1} is the contour point set between fingerroot1 fr_1^i and fingertip ft and $(x_{contour2}, y_{contour2}) \in S_{C_2}$, S_{C_2} is the contour point set between fingertip ft and fingerroot2 fr_2^i. (x_{vp}, y_{vp}) is the vertical point between contour point and the finger direction.

2.4. Accurate Hand direction

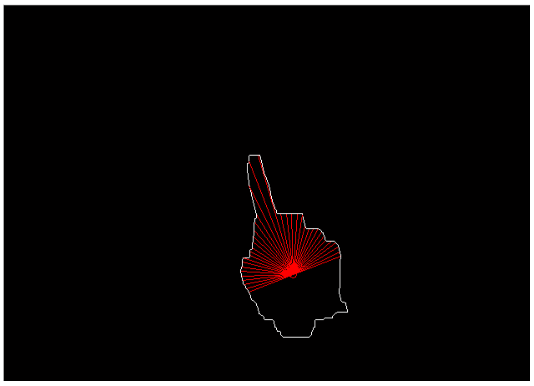
It's intuitive to assume that the directions of the fingers can represent the direction of the hand. So the refined hand direction is defined as follows:

$$\theta_{hand} = \frac{\sum_{i=1}^{i=|\theta_f|} \theta_{f_i}}{|\theta_f|}. \tag{15}$$

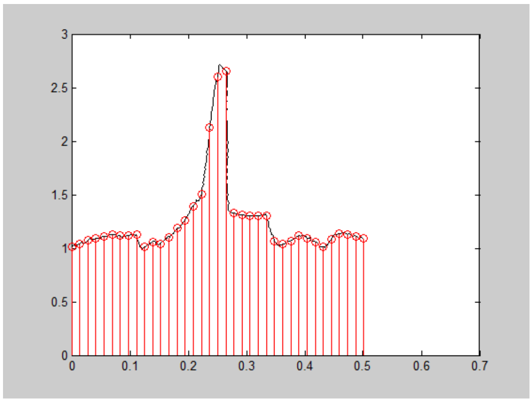
Where $|\theta_f|$ is the length of vector θ_f.

3. Recognition

For the recognition phase, the algorithm first converts the hand contour into a time-series curve with the newly detected hand direction, and then takes samples in the *time-series curve* to form a distance vector which is used for recognition. See figure 7.



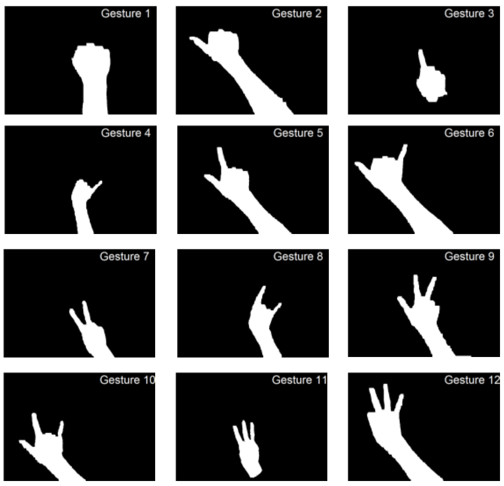
(a). Distance sample vector in hand gesture



(b). Distance sample vector in time-series curve

Figure 7. Distance sample vector

We defined 14 types of hand gestures that can be achieved with no significant by ordinary human hand. See figure 8.



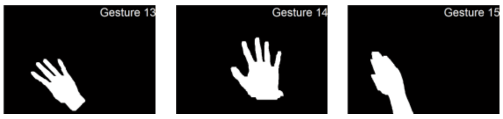


Figure 8. Predefined hand gestures

We use machine learning techniques to learn the feature data of each hand gesture type for the ensuing recognition step.

The algorithm then uses both the number of fingertips and the distance vector for recognition. It first selects the predefined gestures that have the same number of fingertip, then measures the similarity with covariance. The gesture is recognized as the one with which it has the maximal similarity. The result is shown in table 2.

$$covariance_i = \frac{\sum D_i - \overline{D_i} * \sum D_g - \overline{D_g}}{\|D_i\|}. \qquad (16)$$

Where $\mathrm{D_i}$ and $\mathrm{D_g}$ is respectively the distance vector of the ith and the input gesture. Because of the flexibility of the thumb, the recognition results of gestures with thumb in it are not as reliable as others. However, the algorithm is proved to be robust for other gestures and it's robust to orientation changes because the direction and the reference point are relatively fixed using our direction detection algorithm. Besides, it's also robust to scale changes because we rely on the normalized distance.

Table 2. Results of the recognition

Gesture index	1	2	3	4	5	6	7	8	9	10	11	12	13	14
True positive (%)	100	28.6	100	94.7	73.7	93.8	92.7	100	52.4	83.3	100	12.5	100	100
False positive(%)	0	0	2.56	0	0	0	2.01	0	0.63	0	10.6	0	0	0

1	100%													
2		28.6%												
3		71.4%	100%	5.3%		3.125%								
4				94.7%										
5					73.7%									
6						93.75%								
7					26.3%		92.7%					4.2%		
8								100%						
9									52.4%	11.1%				
10										83.3%				
11						3.125%	7.3%			5.6%	100%	83.3%		
12									47.6%			12.5%		
13													100%	
14														100%
	1	2	3	4	5	6	7	8	9	10	11	12	13	14

References

[1] Z. Ren, J. Yuan, and Z. Zhang, Robust hand gesture recognition based on finger-earth movers distance with a commodity depth camera, In Proc. of ACM Multimeida, 2011.

[2] Z. Ren, J. Meng, and J. Yuan, "Depth camera based hand gesture recognition and its applications in human-computer-interaction," in Proc. IEEE Information, Communications and Signal Processing (ICICS' 2011), December 2011, pp. 1 –5.

[3] Z. Ren, J. Meng, J. Yuan, and Z. Zhang, Robust hand gesture recognition with kinect sensor, In Proc. of ACM MM, 2011.

[4] Vladimir I. Pavlovic, Rajeev Sharma, Thomas S. Huang, Visual Interpretation of Hand Gestures for Human-Computer Interaction: A Review, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 19, No. 7, July 1997.

[5] Felzenszwalb, P.F. and Huttenlocher, D.P. 2004. Distance transforms of sampled functions. Cornell Computing and Information ScienceTechnical Report TR2004-1963.J.

[6] K. Grauman and T. Darrell, "Fast Contour Matching Using Approximate Earth Mover's Distance," Proc. IEEE Conf. Computer Vision and Pattern Recognition, pp. I-220-I-227, 2004.

[7] Suarez and R.R. Murphy, "Hand gesture recognition with depth images: a review." IEEE RO-MAN: The 21st IEEE International Symposium on Robot and Human Interactive Communication pp. 411–417, Paris, France, 2012.

P.S. We are currently using Machine Learning algorithms (e. g. Logistic Regression Algorithm) to train the data for the ensuing recognition phase. The result is much more robust and the paper will be changed accordingly. The source code is available in Github: https://github.com/imkaywu/Gesture-Recognition-3. Check my personal blog: http://imkaywu.com/2013/12/05/Paper.html for latest updates.