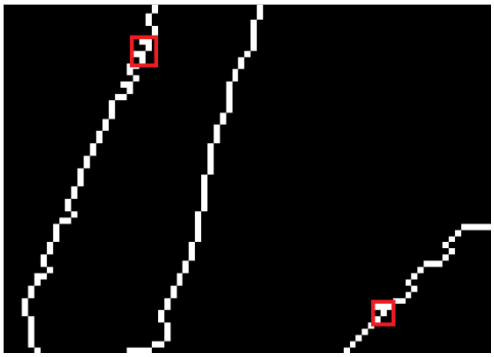
The algorithm of hand gesture recognition uses the relative position of each finger with respect to the reference point for recognition. The algorithm has 3 steps: 1). Detect the rough direction of hand and transform the contour points into *time-series curve*; 2). Locate the accurate positions of fingertips 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.

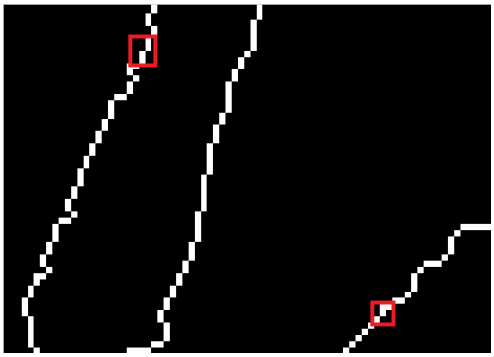
## 1. Hand direction detection and time-series curve

### 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 ($\theta_i,\ i \in \{1,2 \dots,16\}$), see figure 2.

$$k_i = \tan(\theta_i). \quad (1)$$

The algorithm counts the number for each direction $N_{d_i}$ ($N_{d_i} \in N_d, i \in \{1,2 \dots,16\}$), and assigns an index $I_{d_i}$ ($I_{d_i} \in I_d,\ i \in \{1,2 \dots,16\}$) 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. $\quad I_{d_i} = i - I_{max}$;
5. end

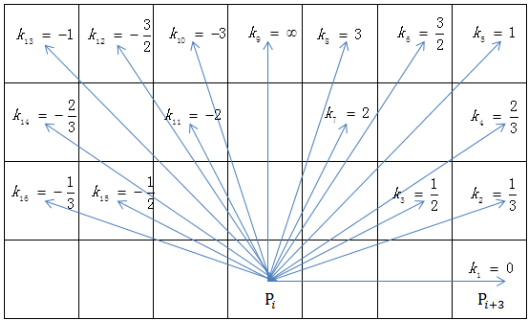



Figure 2. 16 types of direction of neighboring points (each square represents a pixel)

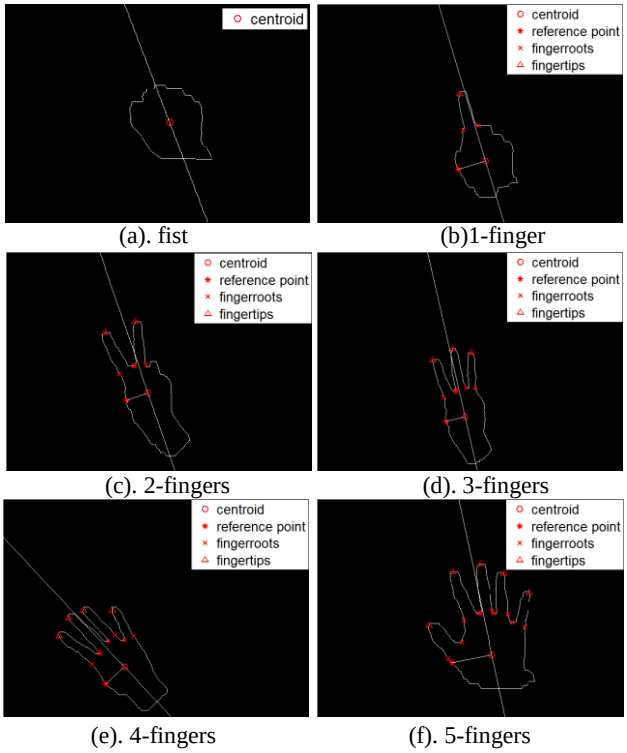
The direction index of the hand $I_{hand}$ is obtained by equation (2):

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

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

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

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 $360^{\circ}$. 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.

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

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

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 $i = 1{:}\,length(d)$
2. if($d_i < d_{min}$)
3. $d_{min} = d_i$;
4. $d_{max} = d_i$;
5. if($d_{min} - d_{\,fingervalley_{temp}} < d_{threshold1}$)
6. $fingervalley_{temp} = i$;
7. end
8. elseif($d_i < d_{max}$)
9. if($d_{max} - d_{min} > d_{threshold2}$)
10. $fingervalley_j = fingervalley_{temp}$;
11. $firstTime = \sim firstTime$;
12. $j = j + 1$;
13. end
14. $d_{min} = d_i$;
15. $d_{max} = d_i$;
16. elseif($d_i > d_{max}$)
17. if($firstTime$)
18. $fingervalley_{temp} = i - 1$;
19. $firstTime = \sim firstTime$;
20. end
21. $d_{max} = d_i$;
23. end
24. end

The 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 $d_i > d_{max} = true$.

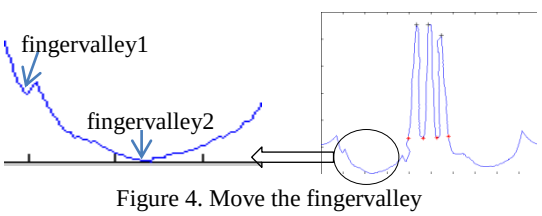
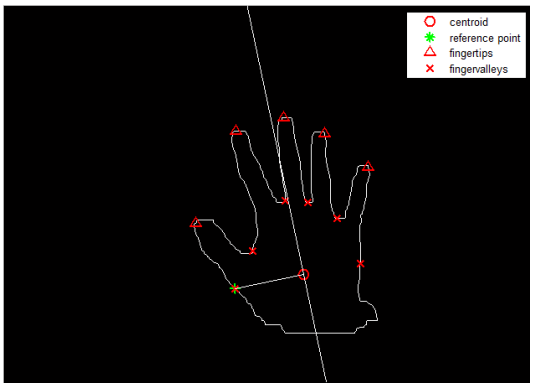
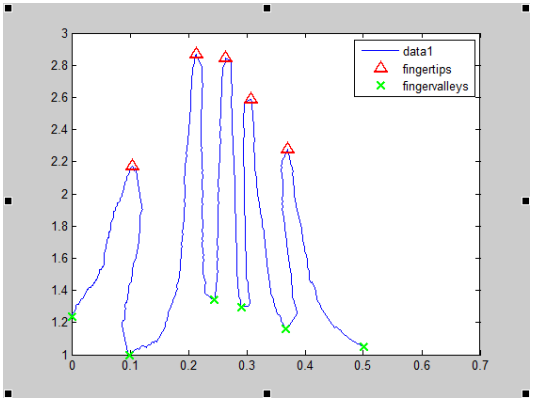



Figure 4. Move the fingervalley




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




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

Figure 5. Original and transformed graph

The information of fingertips and fingervalleys is sufficient in some scenarios which use only the number of fingers to discern gestures, but it's limited due to the relative inaccuracy of the hand direction and reference point. Therefore, a refined way of hand direction detection is proposed next.

## 2. Positions of fingertips and fingerroots localization

The positions of fingertips and fingervalleys are sufficient to locate a finger, yet not accurate enough to recognize gestures. Therefore 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 3**: Detection of the fingerroots.

1. while(true)
2. Detect the direction of finger $\theta_f^i$;
3. Find the fingerroots $fr^i$;
4. Calculate the area of the external rectangle of the 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. end
10. end

### 2.1. Finger direction

The finger direction $\theta_f^i$ 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 (7)-(9).

$$\theta_f^i = \frac{\theta_{f1}^i + \theta_{f2}^i}{2}, \tag{7}$$

$$\theta_{f1}^i = arctan(\frac{y_{ft} - y_{fr1}^i}{x_{ft} - x_{fr1}^i}), \tag{8}$$

$$\theta_{f2}^i = arctan(\frac{y_{ft} - y_{fr3}^i}{x_{ft} - x_{fr3}^i}). \tag{9}$$

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 (9), (10).

$$\frac{y_{fr_{temp1}} - y_{fr3}^i}{x_{fr_{temp1}} - x_{fr3}^i} = \tan\left(\theta_f^i + 90^\circ\right), \tag{10}$$

$$\frac{y_{fr_{temp2}} - y_{fr1}^i}{x_{fr_{temp2}} - x_{fr1}^i} = \tan\left(\theta_f^i - 90^\circ\right). \tag{11}$$

**Algorithm 4**: 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 $A_{Rec}$.

$$Height = \sqrt{(x_{ft} - x_{vp})^2 + (y_{ft} - y_{vp})^2}, \quad (12)$$

$$Width = \max\left(\sqrt{(x_{contour1} - x_{vp})^2 + (y_{contour1} - y_{vp})^2}\right) + \max\left(\sqrt{(x_{contour2} - x_{vp})^2 + (y_{contour2} - y_{vp})^2}\right). \quad (13)$$

In (12), $(x_{ft}, y_{ft})$ is the coordinate of the fingertip, $(x_{vp}, y_{vp})$ is the vertical point between the fingertip and the line of the fingerroots. In (13), $(x_{contour1}, y_{contour1}) \in S_{C_1}$, $S_{C_1}$ 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:

$$\theta_{hand} = \frac{\sum_{i=1}^{i=|\theta_f|} \theta_{f_i}}{|\theta_f|}. \quad (14)$$

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

## 3. Recognition

For the recognition phase, the algorithm calculates the Euclidean distance between the hand gesture and a hand set ($S_{hand}$). The hand gesture is recognized as a candidate gesture from the hand set with which it has the minimal Euclidean distance.

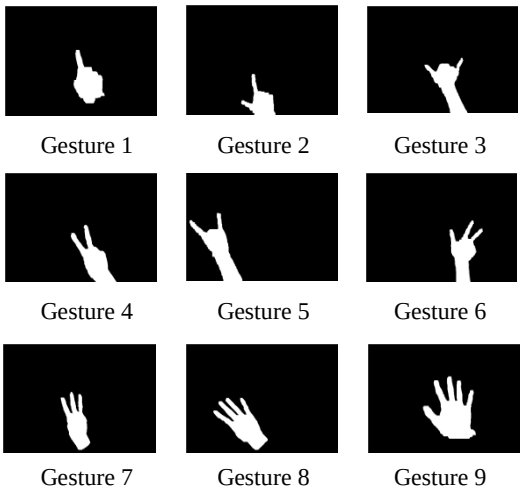
$$d_{hand}^i = \text{Euclidean\_Dist}(fr_{hand}, fr_{S_{hand}^i}), \quad (15)$$

$$d_{hand}^{I_{min}} = \min(d_{hand}). \quad (16)$$

The hand gesture is recognizd as $S_{hand}^{I_{min}}$. Where $fr_{hand}$ is the fingerroots of hand gesture, and $fr_{S_{hand}^i}$ ($S_{hand}^i \in S_{hand}$, i $\in \{1, \ldots, C_5^{length}\}$, length is the finger number of the input gesture) is the fingerroots of a candidate gesture from the hand set. $d_{hand}^i$ ($d_{hand}^i \in d_{hand}$, i $\in \{1, \ldots, C_5^{length}\}$) denotes the Euclidean distance between the hand gesture and each candidate gesture.

We define 9 types of hand gestures as inputs, see figure 6. With this prior knowledge, the performance of the algorithm can be further improved. The hand gestures are collected in uncontrolled environment and the poses for each gesture varies in orientation and scale. The learned data for each fingerroot is in table 1 and the result of the recognition is in table 2.

The reason for choosing Euclidean distance as the recognition metric is its simplicity, efficiency and suitable for hardware implementation. As our test results show, with a class of predefined gestures, this algorithm is robust for real-time application.



Gesture 1 Gesture 2 Gesture 3

Gesture 4 Gesture 5 Gesture 6

Gesture 7 Gesture 8 Gesture 9

Figure 6. Hand gestures

Because of the flexibility of the thumb, the recognition result of gesture 2, 6 is not as reliable as other gestures. 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 change because we rely on the degree of fingerroots relative to the reference point rather than distance of any form for recognition.

| Finger | Thumb | | Forefinger | | Middle finger | | Ring finger | | Little finger | |
|---|---|---|---|---|---|---|---|---|---|---|
| $\theta$ | 0.0447 | 0.0999 | 0.1602 | 0.2273 | 0.2338 | 0.2940 | 0.2840 | 0.3374 | 0.3637 | 0.4156 |

Table 1. Degree between each fingerroot and the reference point

| Gesture index | Gesture 1 | Gesture 2 | Gesture 3 | Gesture 4 | Gesture 5 | Gesture 6 | Gesture 7 | Gesture 8 | Gesture 9 |
|---|---|---|---|---|---|---|---|---|---|
| Accurate rate | 100% | 62.5% | 100% | 100% | 90.9% | 81.25% | 100% | 100% | 100% |

Table 2. Accurate rate of each predefined gesture type

**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.