JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 33, 329-345 (2017)

DOI:10.1688/JISE.2017.33.2.4

A Virtual TouchPad for Smartphone with Depth Camera*

WEI-SHENG WONG[1], SHIN-CHUNG HSU[1] AND CHUNG-LIN HUANG[2,+]
[1]*Department of Electrical Engineering*
National Tsing Hua University
Hsinchu, 30013 Taiwan
[2]*Department of M-Commerce and Multimedia Applications*
Asia University
Taichung, 41354 Taiwan
E-mail: d9761817@oz.nthu.edu.tw; clhuang@asia.edu.tw

This paper proposes a virtual touchpad human computer interaction (*HCI*) based on the fingertip hand gesture recognition (*FHGR*) method for smartphone with depth camera. The virtual touchpad *HCI* requires less computing power and memory resource than the conventional hand gesture recognition (*HGR*) methods. It consists of three modules: (1) open hand identification, (2) active fingertip detection and tracking, and (3) fingertip hand gesture recognition (*FHGR*). First, the handshape region is segmented and divided into three component regions: arm, palm, and fingers. Second, we locate the fingertips and then detect the active fingertips. Finally, the trajectory of every active fingertip is analyzed for *FHGR*. The two major contributions of this paper are (1) development of a real-time *FHGR* method with high recognition accuracy and (2) implementation of a virtual touchpad *HCI* for the smart phones.

***Keywords*:** virtual touchpad, human-computer interaction (HCI), fingertip hand gesture recognition (FHGR), on-click hand gesture, on-press hand gesture

1. INTRODUCTION

Real-time hand gesture recognition (*HGR*) methods have been applied for many human-computer interaction (*HCI*) systems. The *HGR* methods often operate in complicate environment with cluttered background, moving objects, and changing illumination. Skin color modeling [1, 2] has been used to segment the hand region for *HGR*. A mixture of Gaussian model combined with histogram-based color model is effective for hand region extraction. Mohr *et al.* [3] present an algorithm to segment the homogeneous skin-color regions based on skin-color modeling under certain constrains. Lamberti *et al.* [5] propose a real-time color-glove-based *HGR* method by using a vector quantization classifier. Priyal *et al*. [6] use the anthropometric measure for the *HGR* system which is robust to similarity transformation and perspective distortion. Color-based segmentation has been combined with trackers [7-9] to generate better object segmentation results. Optical flow modeling [10] and color-texture analysis [11] are also developed to obtain better hand-region segmentation with high computation complexity.

The color image analysis algorithms of *HGR* are not reliable. Using depth image much simplifies the image analysis of *HGR*. Human hand is the closest object to the smartphone which can be segmented by the thresholding process. However, not only the

Received February 18, 2016; revised April 16, 2016; accepted June 2, 2016.
Communicated by Wen-Liang Hwang.
* This research was financially supported by MOST of Taiwan under the research grant No. MOST 103-2221-E-468-006-MY2.
+ Corresponding author.

hand region but also wrist and arm will be segmented. Chen *et al.* [12] propose a model-based hand tracking method using high quality depth map and large training data set. Xie *et al.*[13] use skin color filter to obtain better silhouette in the depth image. Then they analyze the distance from the contour to palm center to find the fingertips. The other hand segmentation methods [13, 14] divide the wrist and hand by analyzing a boundary curve. Schlegel *et al*. [15] build a vision-based interaction system named *AirTouch* to track multiple fingertips in 3D space, but the user need to wear a glove with marks.

Using depth camera makes the *HGR* more resilient to complex scenes. Many *HGR* methods [20-25, 27, 30] rely on Kinect® depth camera. The depth images are used for foreground object extraction based on the depth profile analysis. Zhou *et al.* [25] present a *HGR* system using Finger-Earth Mover's Distance to measure the dissimilarity between hand shapes. It matches only the fingers not the hand. Kellogg *et al.* [26] introduce so-called *AllSee* that can operate in hand-held devices such as smartphones and tablets. Lee *et al.* [28] propose hand and face recognition *HCI* to control smart TV. Wu *et al.* [29] propose a view-invariant *HGR* which maps a set of point-to-point distances description to a high-dimensional kernel space by using kernel principal component analysis. Then, they apply nonparametric discriminant analysis to extract the view-invariant shape features. Zabri Abu Bakar *et al*. [30] simplify the *HGR* by finger counting. They use Kinect camera and apply the K-curvature algorithm to detect the fingertips. Jiang *et al.* [31] propose a discriminating features extraction for *HGR* of which the features in color and depth images are explored. They reduce the depth and binary pattern features using deflation orthogonal discriminant analysis to enhance the discriminative ability.

Similar to [28], we propose a real-time fingertip hand gesture recognition (*FHGR*) method and develop the so-called virtual touchpad *HCI* for smartphones. The conventional touchpad often causes inconvenience because human hand may occlude the small screen of the hand-held device. This inspires us to develop a virtual touchpad *HCI* with fast response and less memory space requirement. We implement the virtual touchpad *HCI* on HTC One (M8) with dual camera as shown in Fig. 1.

Fig. 1. The HCI interface and the experiment environment.

2. SYSTEM ARICHETUE

Due to the complexity of illumination variation and background, we propose a *FHGR* method for the virtual touchpad *HCI* system using the depth images. The system consists of three different states: *idle* state, *ready* state, and *active* state. The initial state is the *idle* state. The system in idle state will keep probing the appearance of open-hand gesture. Once an open hand gesture is identified, it enters the *ready* state. In *ready* state, it will locate and then track the fingertips to identify the active fingertips. Based on the

trajectory of the active fingertips, it may identify the *on-click* hand gesture and enter the *active* state. The system in *active* state will recognize the hand gesture and then trigger the corresponding App. In the meantime, it either goes back to the *idle* state if there is no *on-click* hand gesture. If another *on-click* hand gesture is identified, it remains in the *active* state and continues identifying the following hand gesture.

Our system consists of three modules: *open hand identification*, *fingertip detection and tracking*, and *FHGR*. The flow diagram of the system is shown in Fig. 2. First, it extracts the foreground region and identifies an open hand. Second, it detects the fingertips and tracks the movement of the fingertips. Third, it analyzes the trajectory of the fingertips to recognize the fingertip hand gesture. The computation limitation of smartphones or hand-held devices makes the system implementation nontrivial. The average total computation time of the three modules is within 30*ms* per frame. The location of the active fingertip is displayed by a cursor on the screen. Once the hand gesture is recognized, it will trigger the corresponding *HCI APP*.

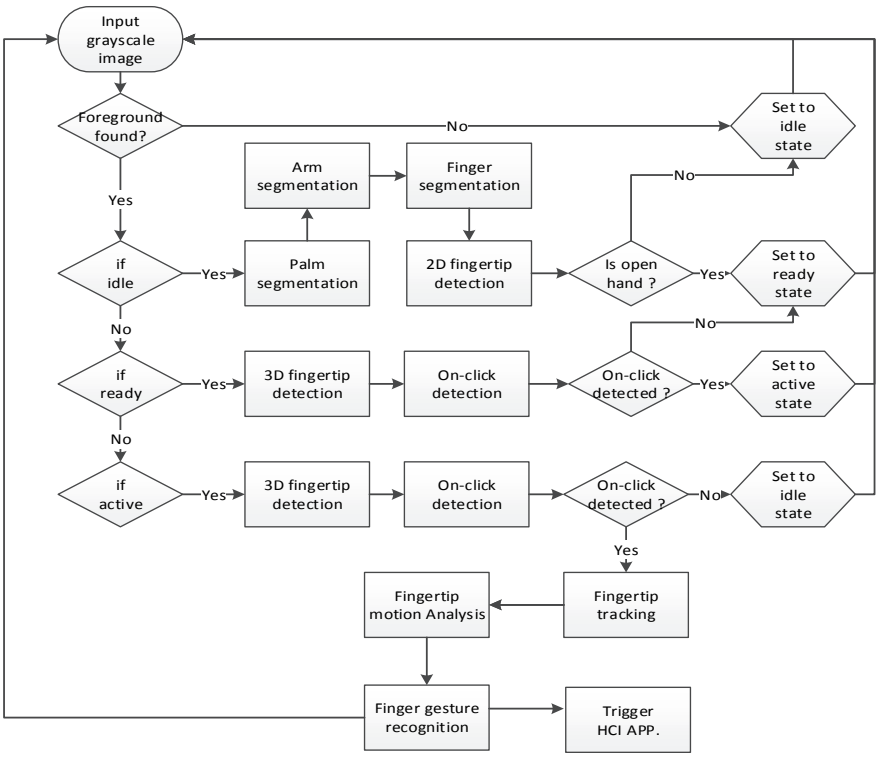


Fig. 2. The overview of system flowchart.

3. HANDSHAPE DETECTION

Our system detects the handshape in the depth images under three assumptions: (1) the handshape is an open hand with five fingers facing toward the camera, (2) no object exists between camera and human hand, and (3) wrist center and arm are lower than the palm.

3.1 The Iterative Thresholding

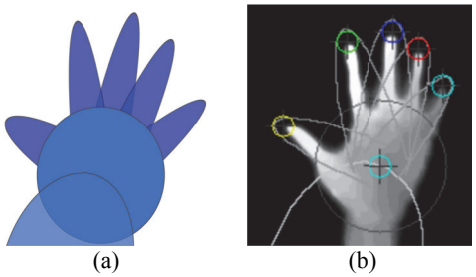
Here, we propose the iterative thresholding process to segment the foreground regions. The large segmented regions are considered as potential handshape regions, whereas, the small segmented regions are considered as noise. Different from Otsu thresholding method, we propose a simple iterative thresholding process to segment the foreground regions. In each depth image, with the maximum depth M_{th}, we choose $th_0 = 0.98M_{th}$ as the initial threshold, and apply the thresholding process to obtain the initial binary map $\mathbf{BW}_0$ which contains invalid hand shape (as shown in Fig. 3 (a)). So, we change the threshold $th_1 = 0.98*th_0$ to obtain an updated binary map $\mathbf{BW}_1$ which is supposed to be larger than $\mathbf{BW}_0$, *i.e.*, $||\mathbf{BW}_1| - |\mathbf{BW}_0|| > \varepsilon$ where ε is a pre-determined small constant. Then, we repeat the thresholding process to obtain another new binary map $\mathbf{BW}_2$ by using the threshold $th_1 = 0.98*th_1$ and have the similar results, *i.e.*, $||\mathbf{BW}_2| - |\mathbf{BW}_1|| > \varepsilon$. The iterative thresholding process repeats until the processing time-out (35*ms*) or no area increment, *i.e.*, $||\mathbf{BW}_{i+1}| - |\mathbf{BW}_i|| < \varepsilon$. The processing time-out indicates that no valid handshape exists. The experimental constant 0.98 is the iteration step size indicating a trade-off between the processing time and the completeness of the segmented region. The iteratively segmented handshapes are shown in Fig. 3.

(a) (b) (c)

Fig. 3. After three iterations (a)-(c), it finds the best cut-off threshold to segment the best handshape.

3.2 Open Hand Component Regions and Fingertip Identification

We assume that the shape of an open hand is different from the other foreground objects. The open hand shape consists of three component regions: palm, arm, and finger regions. If it cannot be decomposed into these component regions, then it is not an open hand. The three component regions can be described by multiple overlapping ellipses as shown in Fig. 4. We extract the individual component region in the open hand by apply



(a) (b)

Fig. 4. (a) We decompose a handshape into palm, arm, fingers which are represented by ellipses; (b) Sketching the boundary on a depth image.

(a) (b) (c) (d)

Fig. 5. (a) The input depth image; (b) The binary image $\boldsymbol{P}$; (c) The distance map $\boldsymbol{P}^{DT}$; (d) The radius of central circle is DT_{max}, the radius of larger circle is $1.5DT_{max}$.

ing the multiple elliptical template matching. From the finger component region, we may identify the location of the fingertip.

Here, we apply distance transform (*DT*) [4] to the segmented hand region to find the palm center and the fingertips. We use the chessboard distance metrics because of its less computation complexity. After the depth thresholding process (with threshold θ_{depth}), the segmented hand region is represented by a set of points with sufficient depth value as $\boldsymbol{P} = \{\boldsymbol{p} | z_{\boldsymbol{p}} \geq \theta_{depth}\}$. Then, we apply the distance transform (*DT*) on $\boldsymbol{P}$ to generate the distance map $\boldsymbol{P}^{DT}$ as shown in Fig. 5 (c). The point in $\boldsymbol{P}^{DT}$ with the maximum distance value (*i.e.*, DT_{max}) is identified as the hand center $\boldsymbol{p}_{hc}$. The larger circle centered at $\boldsymbol{p}_{hc}$ with radius $R=DT_{max}$ is shown in Fig. 5 (d). Finally, we have the palm region $\boldsymbol{P}_{palm}$ as

$$\boldsymbol{P}_{palm} = \{\boldsymbol{p} | \, \|\boldsymbol{p} - \boldsymbol{p}_{hc}\| \leq R_{palm}, \forall \boldsymbol{p} \in \boldsymbol{P}\}, \tag{2}$$

where $R_{palm} = 1.5 \times DT_{max}$, and $\boldsymbol{P}_{palm} \subset \boldsymbol{P}$. Similarly, we may apply *DT* on $\boldsymbol{P}_{palm}$ to obtain the distance map $\boldsymbol{P}_{palm}^{DT}$ in which the point with the maximal distance is treated as the palm center $\boldsymbol{p}_{pc}$. The palm center $\boldsymbol{p}_{pc}$ is very close to the hand center $\boldsymbol{p}_{hc}$. We apply the circular template with radius R_{palm} on the hand region (centered at $\boldsymbol{p}_{pc}$) to remove the palm region from the hand region.

Hand region $\boldsymbol{P}$ can be enclosed by the most compact rectangle with height *h*, width *w*, orientation θ, and centroid $\boldsymbol{p}_c$. The orientation θ of region $\boldsymbol{P}$ can be found by applying Principal Component Analysis method. So, we have the ellipse template for $\boldsymbol{P}$ with major axis $a=h$, minor axis $b=w$, orientation θ, centroid $\boldsymbol{p}_c$. Based on the presumption that the arm is located below the palm, we may segment the arm region $\boldsymbol{P}_{\text{arm}}$ by applying elliptical template matching on the lower part of $\boldsymbol{P}$. To find the fingertips, we need to obtain the finger component regions by subtracting $\boldsymbol{P}_{palm}$ and $\boldsymbol{P}_{arm}$ from $\boldsymbol{P}$ as $\boldsymbol{P}_{finger}=\boldsymbol{P}-\boldsymbol{P}_{palm}-\boldsymbol{P}_{arm}$. The distance map of hand region $\boldsymbol{P}^{DT}$ is shown in Fig. 6 (a), whereas the distance map of finger regions $\boldsymbol{P}_{finger}^{DT}$ are shown in Fig. 6 (b) which is obtained by setting the depth of non-finger pixels (*i.e.*, $\boldsymbol{p} \notin \boldsymbol{P}_{finger}$) to zero. In $\boldsymbol{P}_{finger}^{DT}$, we identify the location of the local maxima as the fingertips as $\boldsymbol{f}_i$, $1 \leq i \leq 5$. Fig. 6 (c) shows three fingertips and the palm center.

(a) (b) (c)

Fig. 6. (a) The $\boldsymbol{P}^{DT}$ distance map; (b) The $\boldsymbol{P}_{finger}^{DT}$ distance map; (c) The fingertips and the palm center.

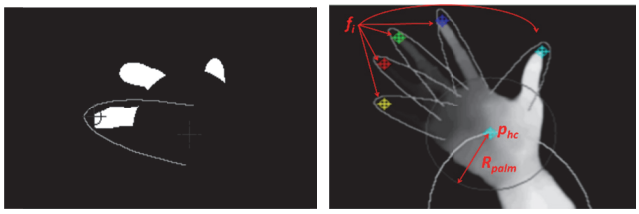
3.3 Open Hand Verification

Based on the fingertips, we may find a more compact finger elliptical templates for open hand verification process to verify whether the extracted handshape is an open hand. We assume the width of the fingers in the range of $(2/7–1/2)R_{palm}$ where $R_{palm}=DT_{max}$. Here, we apply the elliptical template matching on the finger regions as shown in Fig. 7 (a). The boundary of i_{th} finger ellipse template is defined as

$$x_i(t) < x(\boldsymbol{p}_i^M) + a_i\cos(t)\cos(\theta_i) - b_i\sin(t)\sin(\theta_i)$$

$$y_i(t) < x(\boldsymbol{p}_i^M) + a_i\cos(t)\sin(\theta_i) - b_i\sin(t)\cos(\theta_i)$$

where $(\boldsymbol{p}_i^M) = (\boldsymbol{f}_i + \boldsymbol{p}_{h\_c})/2$ is the centroid, $a_i = |\boldsymbol{f}_i + \boldsymbol{p}_{h\_c}|/0.9$, $b_i = R_{palm}/3$, $t = [\pi/2, 3\pi/4]$, and θ_i is the till angle of the i_{th} ellipse template. The major radius a_i and the minor radius b_i are experimentally determined so that the ellipse template is slightly longer and wider than the finger component region and compactly covers the finger component region as shown in Fig. 7 (b).



(a) (b)

Fig. 7. (a) The boundary of the finger-enclosing-ellipse; (b) The elliptical templates.

We further decompose the three component regions of the hand region $\boldsymbol{P}$. As shown in Fig. 7 (b), we apply the overlapping palm template ($\boldsymbol{P}_{palm}$), arm template ($\boldsymbol{P}_{arm}$), and finger templates (*i.e.*, $\boldsymbol{P}_{ell}$) on $\boldsymbol{P}$ to remove the palm, arm and finger component regions from $\boldsymbol{P}$. If the area of the remaining residual region of $\boldsymbol{P}$ is very small, *i.e.*, $|\boldsymbol{P}–\boldsymbol{P}_{palm}–\boldsymbol{P}_{arm} –\boldsymbol{P}_{ell}| < \varepsilon$, then $\boldsymbol{P}$ is decomposed successfully and can be considered as an open hand. The threshold ε is a relative small tolerance which is experimentally defined as $\varepsilon = 0.05|\boldsymbol{P}|$. To verify $\boldsymbol{P}$ as an open hand, we define the following four criteria: (1) $|\boldsymbol{P}–\boldsymbol{P}_{palm}–\boldsymbol{P}_{\text{arm}} –\boldsymbol{P}_{ell}| < \varepsilon$; (2) the $2D$ distance from all fingertips to palm center is smaller than $3R_{palm}$; (3) the angle of each finger to arm is larger than 90°, and (4) the angle between two neighboring fingers is larger than 15°.

4. FINGERTIP HAND GESTURE RECOGNITION

Here, we locate the active fingertips and then identify *on-click* and *on-press* hand gestures. By analyzing the movement of the active fingertips, the system will differentiate the *on-click* from *on-press* hand gesture. If there is no translation movement of the fingertip, then it is an *on-click* hand gesture else it is an on-press hand gesture. By identifying the number of active fingertips and the trajectory of each fingertip, we can further

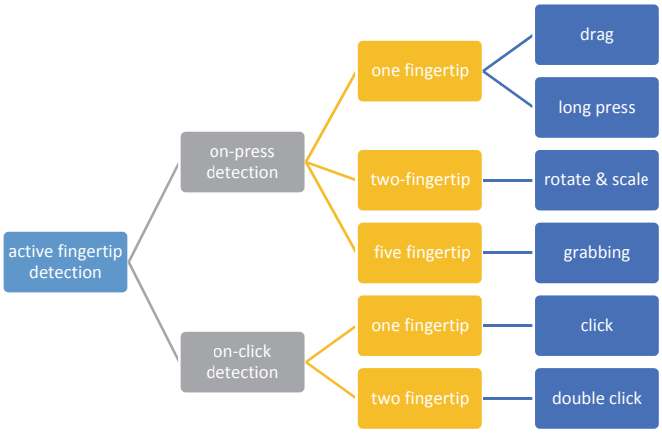


Fig. 8. The category of finger hand gestures for the virtual touchpad *HCI* system.

categorize the *on-click* hand gestures and the on-press hand gestures. The former can be categorized as *single-click* fingertip hand gesture and *double-click* fingertip hand gesture, whereas the latter can be classified as *single-fingertip* hand gesture, *double-fingertip hand gesture*, *and five-fingertip hand gesture*. These hand gestures can be further categorized as *long-press*, *drag*, *rotate*, *rescale*, and *grabbing* as shown in Fig. 8.

4.1 Active Fingertip Detection

By analyzing the distance map and the depth map of the handshape region, we may find the active fingertip. We generate the 3*D distance map* of the handshape region by adding the 2*D* distance map (obtained from *distance transform*) and the depth map. The 3*D* distance is defined as the quadratic summation of the 2*D* distance and the depth value. Here, we may search for the local maximum in the 3*D distance map* to locate the active fingertips. We analyze the 3*D* distance variations of these fingertips to find the 3*D* distance threshold d_{th} for active fingertip detection. The active fingertip determination is based on the comparison of the 3*D* distance of the fingertip with 3*D* distance threshold d_{th} which will be mentioned in section 4.3.

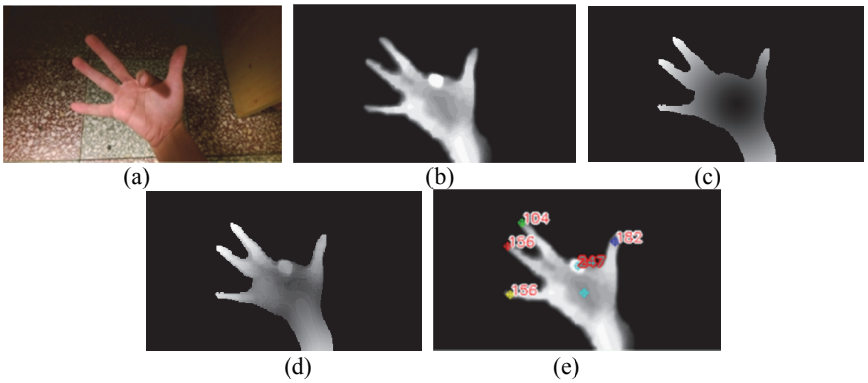


Fig. 9. (a) The color image; (b) The depth map; (c) The 2D distance map; (d) The 3D distance map; (e) The active fingertip is labeled with the corresponding 3D distance value.

4.2 Active Fingertip Tracking

To track the active fingertips between two continuous frames, we analyze the permutation $\boldsymbol{M}(t) = \{m_i | i=1, \ldots, 5\} = (m_1, m_2, m_3, m_4, m_5)$ of the many-to-many mapping between the fingertips in two frames $f(t-1)$ and $f(t)$. The index m_i indicates that the i_{th} active fingertip in previous frame $f(t-1)$ is mapped to the $m_{i_{\text{th}}}$ active fingertip in current frame $f(t)$. The i_{th} mapping (*i.e.*, $m_i \in \boldsymbol{M}$) between the active and non-active fingertips is replaced by x. $|\boldsymbol{M}|$ indicates the number of active fingertip mappings. If $|\boldsymbol{M}|<5$, then there exists one or more active/non-active mapping in $\boldsymbol{M}$. For instance, the permutation $\boldsymbol{M}(t) = (3, 2, x, 1, x)$ (*i.e.*, $|\boldsymbol{M}|=3$) shows that the *3rd* and the *5th* mappings are non-active/active mappings and the 1st, 2nd and 4th mappings are active/active mappings. The 1st, 2nd and 4th active fingertips in frame $f(t-1)$ are mapped to the 3rd, 2nd and 1st active fingertip in frame $f(t)$ respectively. For five fingertips, we have 5!=120 different permutations. We find the best permutation $\boldsymbol{M}_{best}$ by minimizing the sum of the distance of all the fingertip mappings in $f(t-1)$ and $f(t)$ as

$$\boldsymbol{M}_{best} = \underset{M}{\text{Argmin}} \sum_{i=1}^{5} \| F_i(t-1) - F_i(t) \| \tag{3}$$

where $F_i(t-1)$ and $F_i(t)$ indicate the **3***D* position i_{th} active fingertip in frames $f(t-1)$ and $f(t)$ respectively. We define $\{F_i(t-1) | i=1, \ldots, 5\} = (\boldsymbol{p}_1, \boldsymbol{p}_2, \boldsymbol{p}_3, \boldsymbol{p}_4, \boldsymbol{p}_5)$, where $\boldsymbol{p}_i$ denotes the i_{th} active fingertip 2*D* coordinate. If the 4th and 5th fingertips are non-active in frame $f(t)$, then we have $\{F_i(t) | i=1,..5\} = (\boldsymbol{p'}_1, \boldsymbol{p'}_2, \boldsymbol{p'}_3, \boldsymbol{p}_x, \boldsymbol{p}_x)$, where $\boldsymbol{p}_x$ indicates the 2*D* coordinate of non-active fingertip. The distance of the active/non-active fingertip mapping is set to *zero*. Based on Eq. (3), we can find the best permutation $\boldsymbol{M}_{\text{best}}$ even though the number of active fingertips are different in frames $f(t-1)$ and $f(t)$.

4.3 On-click Hand Gesture Detection

By analyzing the 3*D* positions of the fingertips and palm center, we detect the *on-click* hand gesture detection. For *single on-click gesture* detection, we choose the fingertip closest to camera (which has the maximal distance value) as the only active fingertip. For *double on-click hand gesture* detection, we propose a 3D distance threshold d_{th} *f*or active fingertip detecting. We may differentiate the active fingertip based on the threshold d_{th}. The *double on-click hand gesture* detection is a classification problem. The hand center is considered as the cluster center in the 3*D* distance map, and the active fingertips are the pixels with local maximum distance.

To distinguish the active fingertips, we define the average distance of the fingertips and hand center as the threshold d_{th}, *i.e.*, $d_{\text{th}} = \sum_{i=1}^{n}(d_i+d_{hc})/n+1$, where d_i is the 3D distance of the i_{th} fingertip and d_{hc} is 3D distance of the hand center. However, it may not find the active fingertips because the thumb is shorter than the other four fingers. Using the average value may cause the thumb miss-identified as an active fingertip. So, we define the weighted $AVG = \sum_{i=1}^{n}(d_i+d_{hc}\times n)/2n$. The weighted AVG is more effective for detecting the active fingertips. We add a standard deviation (σ) to the weighted AVG as

$$d_{\text{th}} = \frac{\sum_{i=1}^{n} d_i + d_{hc} \times n}{2n} + \sigma \tag{4}$$

where n is the number of active fingertips, d_i is 3D distance of the i_{th} fingertip and d_{hc} is the 3D distance the hand center, and σ is the standard deviation of the 3D distance of these active fingertips. The threshold d_{th} can be used to differentiate the click fingertips from the non-click fingertips more accurately.

4.4 On-Press Hand Gesture Detection

Based on the movement the fingertips, we may differentiate the on-press hand gesture from the on-click hand gesture. There are three different on-press hand gestures: single-fingertip, two-fingertips, and five-fingertips hand gestures.

A. Single-Fingertip On-Press HGR

An *on-press* fingertip hand gesture can be divided into *long-press* hand gesture or *dragging* hand gesture based on the fingertip movement. If the position of the active fingertip changes for more than 10 pixels, then it is a valid fingertip movement which indicates a *dragging* hand gesture, else it is a *long-press* hand gesture.

B. Two-Fingertip On-Press HGR

The *two-fingertip on-press* hand gesture is treated as *rescaling* or *rotation* hand gesture. Based on the trajectory of the fingertips, the system detects *two-fingertip on-press* hand gesture. Then, it analyzes the variation of the line segment connecting these two active fingertips. It calculates both of the *scaling ratio* and the *rotating angle* based on the length and orientation variations of the line segment in two different frames. For the i_{th} frame, l_i. is the line segment of two fingertips and θ_i is the tilt angle of l_i. The angle of rotation $\Delta\theta_i$ is computed as $(\theta_i - \theta_1)$ for the rotation matrix $\boldsymbol{R}(\Delta\theta_i)$ and the ratio r_i = length(l_i)/length(l_1) is computed for the rescaling matrix $\boldsymbol{E}(r_i)$.

C. Five-Fingertip On-Press HGR

The *five-fingertip on-press* hand gesture is a *grabbing* hand gesture. To detect the grab hand gesture, we calculate the average length of the five line segments from five fingertips to the palm center. If the average length keeps decreasing in five or more consecutive frames, then it is identified as a grabbing gesture. The grabbing gesture is similar to the iPad's five finger pinch command to terminate the current App. We verify the accuracy and reliability of the proposed scheme through simulation and performance analysis.

5. EXPERIMENTAL RESULTS

Our system is developed on HTC One (m8) equipped with Dual Lens API and OpenCV4Android 2.4.8 on Android SDK 4.4.2. Our experiment consists of fingertip detection, handshape detection, fingertip tracking, on-click detection, on-press detection, and fingertip tracking.

5.1 Fingertip Detection

The experimental results of fingertip localization are shown in Fig. 10. We can use

the distance or relative position between fingers and palm center to check the reliability of the result. The fingertip detection can reach almost 100% accuracy with 100 testing images of open hand taken from different scenes. Although it may not be able to locate the bending and curved fingers, but the accuracy of finding stretched fingers meets the requirement for implementing the open-hand detection algorithm.

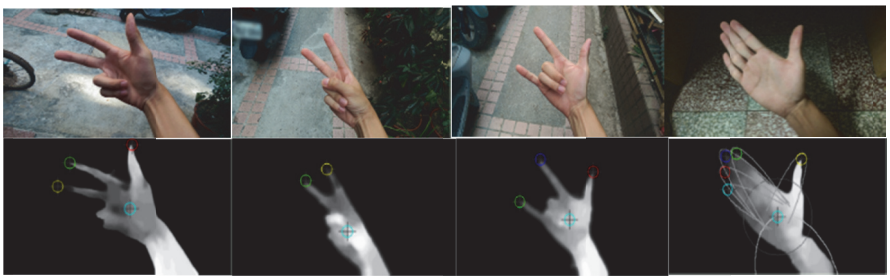
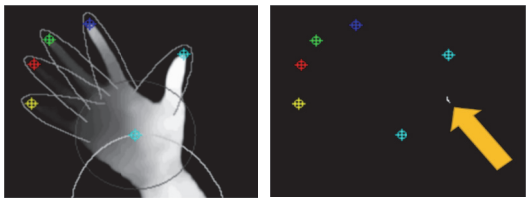


Fig. 10. The fingertip localization (processing time < 15ms).

5.2 Open-Hand Detection

Fig. 11 shows the rendering of the ellipses to cover the fingers, arm, and palm regions. An open hand should be enclosed inside the union of these ellipses. We identify an open hand by the following three constrains: (1) There are five fingertips; (2) The amount of residual points is less than 2% of the hand area as shown in Fig. 11 (b); (3) The open-hand verification process is valid. In the experiments, we test 200 frames with various angles between fingers and arm as shown in Fig. 12. Our method avoids false alarms as shown in Fig. 13 by verifying the above three constraints. We test 200 testing images, the average accuracy is 84%, 96%, 98% for tolerance ε=1%, 3%, 5% respectively.



(a) (b)

Fig. 11. (a) The ellipse masks; (b) the residual points after ellipse mask removal.

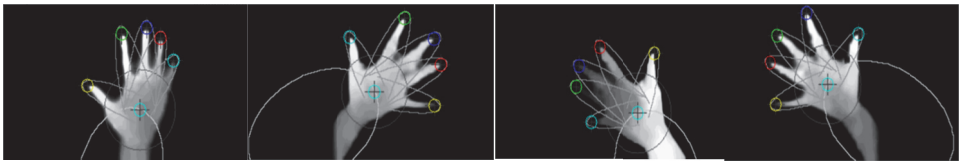


Fig. 12. The open-hand detection.

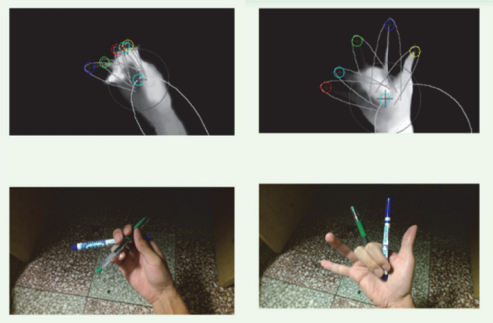


Fig. 13. False positives are avoided.

5.3 Fingertip Tracking

The fingertip tracking starts when an on-press gesture is detected. The tracking process provides the crucial information for the hand gesture identification. The tracking results are plotted on the color circles as shown in Fig. 14. This tracking algorithm is tested by 10 testing video sequences, each video sequence contains over 15 frames. It shows no error when the number of fingertips between two frames are consistent with 93.6% accuracy. There are 63 mismatches and 975 matches.

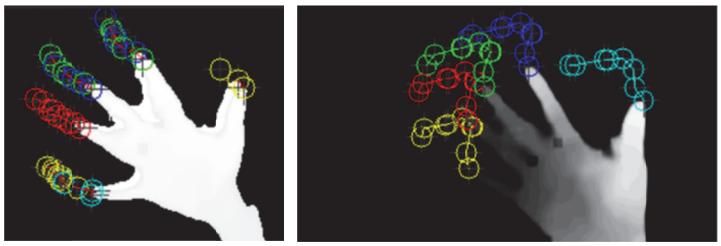


Fig. 14. The tracker tracks the movement of the thumb and other fingers.

5.4 On-click Hand Gesture Detection

Fig. 15 shows two different examples, and the fingertips are labeled with the corresponding depth value. The grayscale hand images on the top of Fig. 15 are the contrast enhanced images, and the labeled numbers are the original depth value of the designate positions. The fingertip is tracked frame by frame. If the movement is identified, it becomes an active fingertip. If an active fingertip exists for a short time interval (in four or less consecutive frames) with depth variation, then an on-click gesture is identified. The number of active fingertips in an on-click gesture may be more than one.

5.5 On-Press Hand Gesture Detection

The position of each active fingertip will be tracked and analyzed for *on-press* hand gesture identification. The difference between the *on-click* and *on-press* gestures is that

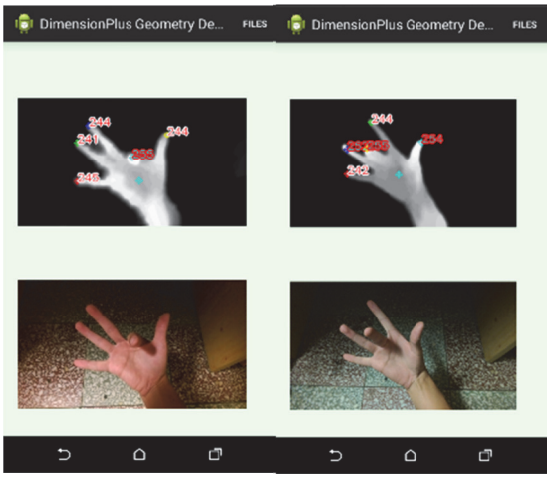


Fig. 15. The color image and the enhanced depth image labeled by the original depth value of each fingertips, highlighted text representing on-clicked point.

the latter makes translational fingertip movement. By analyzing the fingertip trajectory, we may identify the hand gestures as *single, two,* and *five fingertips op-press* hand gesture.

5.6 HGR Evaluation

To evaluate our *FHGR* system, we propose two modules: *fingertip hand gesture recognition* and *gesture parameter calculation*. The former analyzes the active fingertips to recognize the hand gesture, whereas the latter calculates the gesture parameters based on the trajectory of the fingertips. We analyze two different kinds of errors: *fingertip hand gesture recognition error* and *precision error*. The latter measures the location errors of the detected fingertips. Here, we use a tripod to fix the smartphone with depth camera and take 100 frames to examine the position accuracy. The precision error in terms of RMSE is 0.94 (pixel). The accuracy of fingertips detection is important for the trajectory analysis. There are three categories of hand gesture with 1, 2, and 5 active fingertips. Here, we prepare 100 testing images for each category and check the number of the identified active fingertips. The number of active fingertips for each category is listed in Table 1.

Table 1. The number of detected active fingertips for 3 testing cases.

Input\Result	**1**	**2**	**3**	**4**	**5**
1 (total 100)	91	8	1	0	0
2 (total 100)	0	92	7	1	0
5 (total 100)	0	1	6	11	82

A. Single-Fingertip HGR

Based on the fingertip tracking, we have the relative movement of the active fingertip for hand gesture recognition. We define three different types of the *single-fingertip*

hand gesture, *i.e.*, *on-click*, *long-press*, *drag*. A long fingertip movement specifies a *drag* hand gesture, whereas a short fingertip movement indicates an *on-click* or a *long-press* hand gesture. *On-click* is a short tapping hand gesture, whereas, *long-press* is a long tapping hand gesture. Based on this pre-assumption, we can differentiate three different types of *single-fingertip* hand gesture based on the following criteria:

If (#consecutive frames <5) then "*on-click*".
Else if (movement > 10 pixel) then " *drag*".
Else if (movement ≤ 10 pixel) then "long-press".

Fig. 16 (a) shows an *on-click* hand gesture, and Fig. 16 (b) shows a *drag* hand gesture. To verify the results, we test the algorithm based on 100 testing image sequences for each type of single-fingertip hand gesture. The results are listed in Table 2.

(a) (b)

Fig. 16. (a) The trajectory of on-click hand gesture; (b) The trajectory of drag hand gesture.

Table 2. The confusion matrix of the single-fingertip hand gesture.

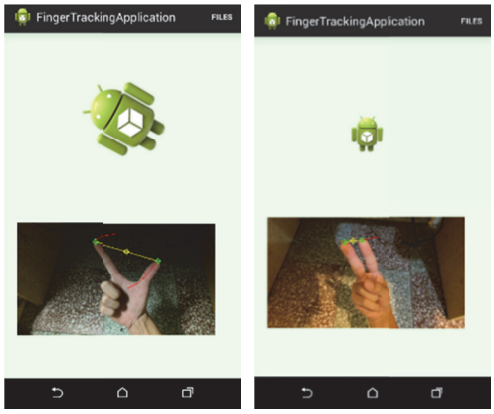
	On-click	Drag	Long-press
On-click	83	11	6
Drag	0	100	0
Long-press	0	5	95

B. Two-Fingertip HGR

There are two different types of two-fingertip hand gesture. The first one is *double-click* hand gesture which consists of two active fingertips performing two *on-click* hand gestures simultaneously. The second one is *scale/rotate* hand gesture which is made to illustrate rotation factor θ and scale factor r by using two active fingertips as shown in Fig. 17. The red curve shows the trajectory of the fingertip, and the yellow line segment connects two active fingertips. Based on the variation of the red curve and the orientation of the yellow line segment, we may have the rotation factor θ and scale factor r. We test 10 testing data sets for both *double-click* and *scale/rotate* hand gestures. Table 3 illustrates the confusion matrix of the experimental results of distinguishing two different two-fingertip (*i.e.*, *double-click* and *scale/rotate*) hand gestures.

Table 3. The confusion matrix of two fingertips hand gesture.

Input \ Results	Double-click	Scale/Rotate
Double-click	42	8
Scale/Rotate	0	50



(a) (b)

Fig. 17. (a) Rotation hand gesture; (b) Rescaling hand gesture.

C. Five-Fingertip HGR

The five-fingertip hand gesture is called the *grabbing* hand gesture as shown in Fig. 18. The yellow line segments represent the distance from the fingertip to the center and the red curves show the trajectory of the active fingertips. It tracks the movement of five active fingertips to detect the *grabbing* hand gesture. If the movement continues over five frames, it will be recognized as a *grabbing* hand gesture. We have some grabbing image sequences as positives and some random fingertip movement as negatives. Our algorithm generates 96 true-positive and 12 false-positive detections out of 100 positive test samples and 150 negative test samples.

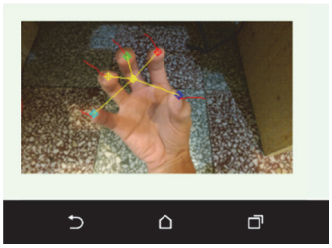


Fig. 18. Grabbing hand gesture.

Table 4. The *grabbing* hand gesture detection.

	# of test samples	# of identification
Positive samples	100	96
Negative samples	150	12

5.7 Processing Time

We test our *FHGR* system by using the images of different resolutions, *i.e.*, 320p, 640p, 1280p. The average processing time of 100 image frames are 13.1, 35.4, 95.3 *ms* respectively. For images of different resolutions (*i.e.*, 640p and 1280p), we have to mul-

tiply the original threshold (defined in section 5.6 for image resolution of 320p) by 2 and 4 respectively so that it can recognize the fingertip hand gestures. This result shows that the real-time depth-based *FHGR* system on a smartphone has been implemented successfully.

6. CONCLUSIONS

This paper proposes a virtual touchpad system for smartphone HCI App. The system includes foreground segmentation, fingertip detection and tracking, open hand gesture detection, arm-palm-finger segmentation, active fingertip detection and tracking. Although the depth image quality obtained from smartphone is not good enough, the experimental results show that the virtual touchpad system is feasible.

REFERENCES

1. P. Kakumanu, S. Makrogiannis, and N. Bourbakis, "A survey of skin-color modeling and detection methods," *Pattern Recognition*, Vol. 40, 2007, pp. 1106-1122.
2. M. J. Jones and J. M. Rehg, "Statistical color models with application to skin detection," *International Journal of Computer Vision*, Vol. 46, 2002, pp. 81-96.
3. D. Mohr and G. Zachmann, "Segmentation of distinct homogeneous color regions in images," *Computer Analysis of Images and Patterns*, Springer, LNCS 4673, 2007, pp. 432-440.
4. R. Fabbri, L. D. F. Costa, J. C. Torelli, and O. M. Bruno**,** "2D Euclidean distance transform algorithms: A comparative survey" *ACM Computing Surveys*, Vol. 40, 2008, pp. 1-44.
5. L. Lamberti and F. Camastra, "Real-time hand gesture recognition using a color glove," in *Proceedings of International Conference on Image Analysis and Processing*, LNCS, Vol. 6978, 2011, pp. 365-373.
6. S. P. Priyal and P. K. Bora, "A robust static hand gesture recognition system using geometry based normalizations and Krawtchouk moments," *Pattern Recognition*, Vol. 46, 2013, pp. 2202-2219.
7. L. Sigal, S. Sclaroff, and V. Athitsos, "Skin color-based video segmentation under time-varying illumination," *IEEE Transactions on PAMI*, Vol. 26, 2004, pp. 862-877.
8. C. Li and K. M. Kitani, "Pixel-level hand detection in ego-centric videos," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 3570-3577.
9. J. Wan, Q. Ruan, G. An, and W. Li, "Hand tracking and segmentation via graph cuts and dynamic model in sign language videos," in *Proceedings of International Conference on Signal Processing*, 2012, pp. 1135-1138.
10. O. Rashid and A. Al-Hamad, "Flow modeling and skin-based Gaussian pruning to recognize gestural actions using HMM," in *Proceedings of International Conference on Pattern Recognition*, 2012, pp. 3488-3491.
11. Y. Deng and B. S. Manjunath, "Unsupervised segmentation of color-texture regions in images and video," *IEEE Transactions on PAMI*, Vol. 23, 2001, pp. 800-810.
12. Q. Chen, X. Sun, Y. Wei, and X. Tang, "Realtime and robust hand tracking from depth," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recog-*

nition, 2014, pp. 1106-1113.
13. Q. Xie, G. Liang, C. Tang, and X. Wu, "A fast and robust fingertips tracking algorithm for vision-based multi-touch interaction," in *Proceedings of IEEE International Conference on Control and Automation*, 2013, pp. 1346-1351.
14. Z. Mo and U. Neumann, "Real-time hand pose recognition using lowresolution depth images," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, Vol. 2, 2006, pp. 1499-1505.
15. D. Schlegel, A. Chen, C. Xiong, J. Delmerico, and J. Corso, "Airtouch: Interacting with computer systems at a distance, in *Proceedings of IEEE Workshop on Application of Computer Vision*, 2011, pp. 1-8.
16. P. F. Whelan and D. Molloy, "Machine vision algorithms in Java: Techniques and implementation," *Springer Science and Business Media*, 2000.
17. N. Oikonomidis, I. Kyriazis, and A. Antonis, *Efficient Model-Based 3D Tracking of Hand Articulations Using Kinect*, BMVA Press, UK, 2011.
18. O. Rashid and A. Al-Hamad, "Flow modeling and skin-based Gaussian pruning to recognize gestural actions using HMM," in *Proceedings of International Conference on Pattern Recognition*, 2012, pp. 3488-3491.
19. Y. Deng and B. S. Manjunath, "Unsupervised segmentation of color-texture regions in images and video," *IEEE Transactions on PAMI*, Vol. 23, 2001, pp. 800-810.
20. K. K. Biswas and S. K. Basu, "Gesture recognition using Microsoft Kinect," in *Proceedings of the 5th International Conference on Automation*, *Robotics and Applications*, 2011, pp. 1-6.
21. Y. Li, "Hand gesture recognition using Kinect," in *Proceedings of IEEE 3rd International Conference on Software Engineering and Service Science*, 2012, pp. 196-199.
22. P. Doliotis, A. Stefan, C. McMurrough, D. Eckhard, and V. Athitsos, "Comparing gesture recognition accuracy using color and depth information," in *Proceedings of 4th International Conference on Pervasive Technologies Related to Assistive Environments*, 2011, pp. 1-7.
23. A. Kurakin, Z. Zhang, and Z. Liu, "A real time system for dynamic hand gesture recognition with a depth sensor," in *Proceedings of the 20th European Signal Processing Conference*, 2012, pp. 1975-1979.
24. D. Wu, F. Zhu, and L. Shao, "One shot learning gesture recognition from RGBD images," in *Proceedings of IEEE Conference on CVPR Workshops*, 2012, pp. 7-12.
25. R. Zhou, J. Yuan, J. Meng, and Z. Zhang, "Robust part-based hand gesture recognition using Kinect sensor," *IEEE Transactions on Multimedia*, Vol. 15, 2013, pp. 1110-1120.
26. B. Kellogg, V. Talla, and S. Gollakota, "Bringing gesture recognition to all devices," in *Proceedings of the 11th USENIX Symposium on Networked Systems Design and Implementation*, 2014, pp. 303-316.
27. K. Lai, J. Konrad, and P. Ishwar, "A gesture-driven computer interface using kinect," in *Proceedings of IEEE Southwest Symposium on Image Analysis and Interpretation*, 2012, pp. 185-188.
28. S.-H. Lee, M.-K. Sohn, D.-J. Kim, and B. Kim, "Smart TV interaction system using face and hand gesture recognition," in *Proceedings of IEEE International Conference on Consumer Electronics*, 2013, pp. 173-174.
29. X. Wu, X. Mao, L. Chen, and Y. Xue, "View-invariant gesture recognition using nonparametric shape descriptor," in *Proceedings of International Conference on Pattern Recognition*, 2014, pp. 544-549.

30. M. Z. A. Bakar, R. Samad, D. Pebrianti, and M. Mustafa,"Finger application using K-curvature method and Kinect sensor in real-time," in *Proceedings of International Symposium on Technology Management and Emerging Technologies*, 2015, pp. 218-222.
31. F. Jiang , C. Wang, Y. Gao, and S. Wu, "Discriminating features learning in hand gesture classification," *IET Computer Vision*, Vol. 9, 2015, pp. 484-491.

Wei-Sheng Wong (翁偉昇) received the B.S. degree and M.S. degree in Electrical Engineering from National Tsing-Hua University, Hsinchu, Taiwan, in June 2012 and September 2014, respectively. His research interests include computer vision, pattern recognition, machine learning. Currently, he is the Director in MoPay Corporation in Taipei, Taiwan.

Shih-Chung Hsu (徐士中) received the B.S. and M.S. degree from National Taiwan University of Science and Technology, Taipei, Taiwan, in 2004 and 2006, respectively. Currently, he is a Ph.D. student in Electrical Engineering Department, National Tsing-Hua University, Hsinchu, Taiwan. His research interests include computer vision, pattern recognition, machine learning.

Chung-Lin Huang (黃仲陵) received his B.S. degree from the National Tsing-Hua University and MS degree from National Taiwan University respectively. He obtained his Ph.D. degree in Electrical Engineering from the University of Florida, Gainesville, FL, USA, in 1987. From 1988 to 2012, he was a Professor in the Electrical Engineering Department, National Tsing-Hua University, Hsinchu, Taiwan. Since August 2012, he has been a Professor in Department of M-Commerce and Multimedia Applications, Asia University, Taichung, Taiwan. He has received numerous paper awards from International and Local Academic Conferences. His research interests are in the area of image processing, computer vision, and visual communication.