Hindawi
Security and Communication Networks
Volume 2018, Article ID 4879496, 11 pages
https://doi.org/10.1155/2018/4879496

Research Article

A High-Security and Smart Interaction System Based on Hand Gesture Recognition for Internet of Things

Jun Xu, Xiong Zhang, and Meng Zhou

Display Center, School of Electronic Science and Engineering, Southeast University, Sipailou 2#, Nanjing, Jiangsu 210096, China

Correspondence should be addressed to Xiong Zhang; zhangxiongseu@163.com

Received 28 February 2018; Revised 12 April 2018; Accepted 9 May 2018; Published 6 June 2018

Academic Editor: Ilsun You

In this work, we propose a vision-based hand gesture recognition system to provide a high-security and smart node in the application layer of Internet of Things. The system can be installed in any terminal device with a monocular camera and interact with users by recognizing pointing gestures in the captured images. The interaction information is determined by a straight line from the user's eye to the tip of the index finger, which achieves real-time and authentic data communication. The system mainly contains two modules. The first module is an edge repair-based hand subpart segmentation algorithm which combines pictorial structures and edge information to extract hand regions from complex backgrounds. Second, the position which the user focuses on is located by an adaptive method of pointing gesture estimation, which adjusts the offsets between the target position and the calculated position due to lack of depth information.

1. Introduction

Internet of Things (IoT) which connects objects around us together to the Internet becomes one of the most important information technologies nowadays. One of the issues of IoT is how to make the object-object interaction and human-object interaction smart and safe. Data collected from users may include sensitive or private information of their daily activities; hence security protection and privacy preserving are vital for the development of IoT. According to the principle of information generation and the technical architecture, IoT consists of three layers, namely, perception [1, 2], network protocol [3, 4], and application layers [5, 6]. The perception layer is composed of various sensors and gateways, such as temperature sensor, two-dimensional barcode, camera, and Global Positioning System (GPS). The perception layer recognizes objects and collects information with Radio Frequency Identification (RFID) and Wireless Sensor Networks (WSNs); therefore, the security issue for the perception layer is essentially on node capture attacks of RFID and WSNs [7, 8]. The network protocol layer is the central pivot of IoT which transmits and processes the information from the perception layer. As one of the key technologies in the network layer, wireless mobile communication networks employ traditional encryption and authentication techniques to improve the security and privacy of IoT [9, 10]. The application layer realizes the intelligent services by connecting IoT and users, such as city management, intelligent transportation, telemedicine, and smart home. The application layer also faces the challenges of security threat. For example, the attackers can capture the unattended equipment and send data to the application system by identity impersonation and data modification. Therefore, this study is aimed at developing a secure and authentic interaction system based on computer vision for smart home in the application layer of IoT. As shown in Figure 1, our interaction system can control the cursor of a computer or a smart TV by recognizing users' pointing gestures. The cursor is located immediately according to the line from eye to fingertip when the user points to the target. Since the interaction data is produced only when the user's eye and fingertip are detected and the pointing direction is recognized, the system can effectively avoid unauthorized access of illegal users, which is more secure than the traditional point-touch devices.

Many studies have been conducted to recognize pointing gestures based on computer vision. When human interacts with an object by pointing gesture, the pointing direction is estimated by two points on the camera perspective line.

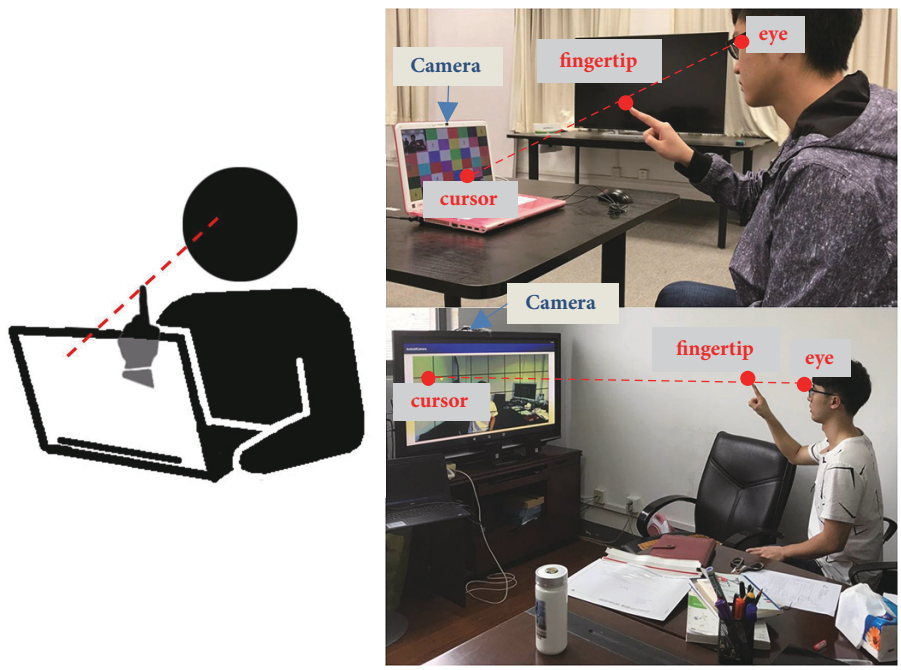

Figure 1: Applications of our interaction system based on pointing gesture recognition.

According to the different input cameras, pointing gesture recognition technologies can be classified into 3D methods [11–13] and 2D methods [14–16]. 3D methods rely on specific input devices, such as stereo camera or Kinect, while 2D methods use less expensive and more easily available cameras, with reduced computation cost for 2D information. However, pointing gesture recognition based on 2D methods is still challenging. First, since 2D information of the hand provides relatively weak and ambiguous characteristics, the discrimination of bare hand from background is easily influenced by the large variability of hand appearances and the presence of other skin-like objects. Some studies have been conducted to deal with this problem [17–19]. For example, Li and Wachs [17] proposed a weighted elastic graph matching method to detect and recognize ten hand postures under complex backgrounds. The average recognition accuracy reached 97%; however the performance was unclear when hand shape distorts. In [18], hand regions were extracted based on the Bayesian model of visual attention by combining shape, texture, and color cues. The long run time, 2.65 s, of this method clearly made it unsuitable for real-time application. Gonzalez et al. [19] proposed a hand segmentation method based on pixel color and edge orientation to deal with the overlap of hand and face, whereas the segmentation results were affected by thin lines under the chin or over the collar. The second challenge is to estimate the pointing position determined by intersection between the interaction plane and the pointing vector due to lack of depth information. In order to solve this problem, some interaction systems based on 2D pointing gesture methods only utilize information of pointing direction instead of pointing position [14, 15], and some other systems require users to make coordinate calibration before operating [16]. In this paper, we propose an edge repair-based hand subpart segmentation algorithm, which accurately and effectively segments the palm and finger regions from the background by using 2D information. Furthermore, on the basis of the hand segmentation algorithm, an adaptive method of pointing direction estimation is developed which can adjust the eye-fingertip line during operation.

2. System Model

We develop a vision-based interaction system using pointing gesture recognition as a node in the application layer of IoT. When the user points to the screen, a straight line from the eye to the fingertip determines the position on the screen where the cursor should locate. Figure 2 illustrates the flow chart of our system. First, 2D images are captured by a normal camera. Second, the user's eye is detected by the AdaBoost classifier based on Haar-like features [20]. Third, hand region is segmented from the complex backgrounds using the edge repair-based hand subpart segmentation algorithm. Fingertip of the index finger is detected by combining convex hull and convexity defect features [21]. If both the eye and the hand are located, an adaptive method of pointing direction estimation is proposed to obtain the pointing position according to the eye-fingertip line. Finally, the cursor is moved to the pointing position. Consequently, the system is secure and authentic because it can only be activated when the eye and hand of the user are both detected.

2.1. Methods of Eye Detection and Hand Segmentation. In our system, the face position is first coarsely determined by background subtraction and skin color detection. Then connected regions with large areas are extracted. In order to verify face from these regions, an ellipse model is employed

to select the approximately elliptical candidates because of the face shape [22]. For each candidate, the AdaBoost algorithm based on Haar-like features is employed for eye detection. The AdaBoost classifier is trained by the positive samples of eye images and the negative samples of all kinds of background images without human eyes. A result of eye detection is shown in Figure 2.

In order to segment hand regions efficiently, we propose an edge repair-based hand subpart segmentation algorithm which includes four procedures as illustrated in Figure 3.

Firstly, a hierarchical chamfer matching algorithm (HCMA) [23] is used to locate the whole hand region in the binary image produced by combining skin color detection and background subtraction. As shown in Figure 4, the chamfer distance image of the hand is searched for the optimal position which matches the hand template from the previous frame. After the distance image is traversed, the optimal position is figured out by calculating the minimum edge distance Ed.

$$\mathrm{Ed} = \frac{1}{3}\sqrt{\frac{1}{N}\sum_{i=0}^{N-1} v_i^2}, \tag{1}$$

where v_i is the pixel value that the template hits and N is the number of the contour pixels in the template. In order to accelerate the matching, a pyramid structure is built by halving the resolution of the distance image gradually. At the top level of the pyramid structure, a grid of positions is chosen to start the matching. Each position and its neighborhood are computed for Ed. If a smaller edge distance is found, the template is moved to the new position (X, Y) in the distance image of level n by

$$\begin{aligned} X &= \frac{X_r + 2^n - 1}{2^n} \\ Y &= \frac{Y_r + 2^n - 1}{2^n}, \end{aligned} \tag{2}$$

where

$$\begin{aligned} X_r &= c_x + \cos\theta \cdot s_x \cdot x - \sin\theta \cdot s_y \cdot y \\ Y_r &= c_y + \sin\theta \cdot s_x \cdot x + \cos\theta \cdot s_y \cdot y \end{aligned} \tag{3}$$

where (x, y) are the coordinates of the points in the template and (c_x, c_y), (s_x, s_y), and θ are translation, scaling, and rotation parameters, respectively. For each start position, the position with local minima is obtained and then used as the start position at the next level $n - 1$. When all the levels are traversed, the optimal position is finally identified, which is illustrated by the blue rectangle in Figure 4(b).

Secondly, the located hand region is analyzed to detect the palm and fingers separately by combining pictorial structures and Histogram of Oriented Gradient (HOG) features [24]. Figure 5(a) shows the hand model based on pictorial structures, which includes a root part of the palm and five-finger parts. Thus the hand configuration is denoted by $L = \{l_0, l_1, l_2, \ldots, l_n\}$ where the subscript 0 corresponds to the palm part and the subscripts $1 \sim n$ correspond to the finger parts.

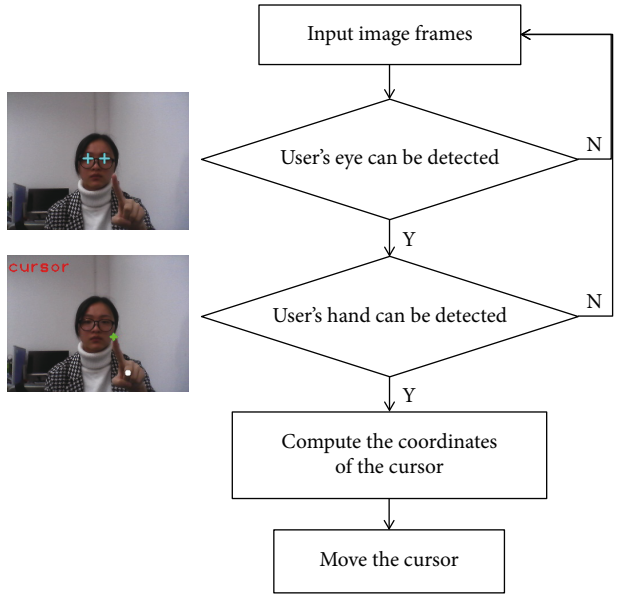

Figure 2: Flow chart of our proposed system.

Given an image I and a set of hand model parameters $\boldsymbol{\theta} = \{u_i, c_{ij}\}$, the maximum a posteriori (MAP) probability of L is represented by

$$\begin{aligned} P(L \mid I, \theta) &\propto P(I \mid l_0, u_0) \\ &\cdot \prod_{i=1}^{n} \left(P(I \mid l_i, u_i) \cdot P(l_i, l_0 \mid c_{i0})\right), \end{aligned} \tag{4}$$

where u_i is the appearance parameter of part i and c_{ij} is the connection parameter between part i and part j. Due to the different characteristics of the palm and finger in the model, two support vector machine (SVM) classifiers are employed to detect the subparts of the hand. On the one hand, the classifier for the palm part is trained by HOG features. On the other hand, the input feature vector of the classifier for the finger part considers both HOG features and the spatial relationship between the finger and palm. Let the state of the finger part i be l_i (x_i, y_i, θ_i). Assuming (x_j, y_j) and (x_0, y_0) are the coordinates of the joint and the palm center, the relative position (x_i, y_i) is computed by

$$\begin{aligned} x_i &= \frac{\left(x_j - x_0\right)}{w_0} \\ y_i &= \frac{\left(y_j - y_0\right)}{w_0}, \end{aligned} \tag{5}$$

where w_0 is the size of the palm part. θ_i is the absolute part orientation as shown in Figure 5(a). Figure 5(b) shows the detection result of the image in Figure 4.

Thirdly, since the blurry border of the hand and face probably leads to an incompletely connected hand silhouette, we propose an edge repair method to recover the contour of each subpart. The edge image of each subpart is extracted to

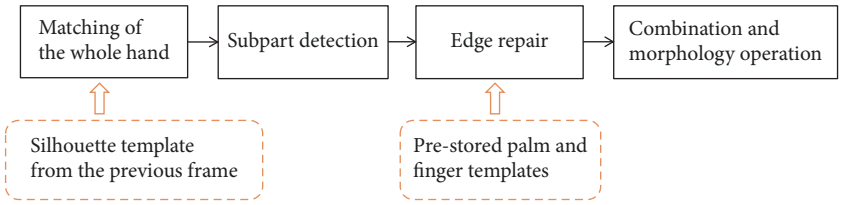

Figure 3: Architecture of the edge repair-based hand subpart segmentation algorithm.

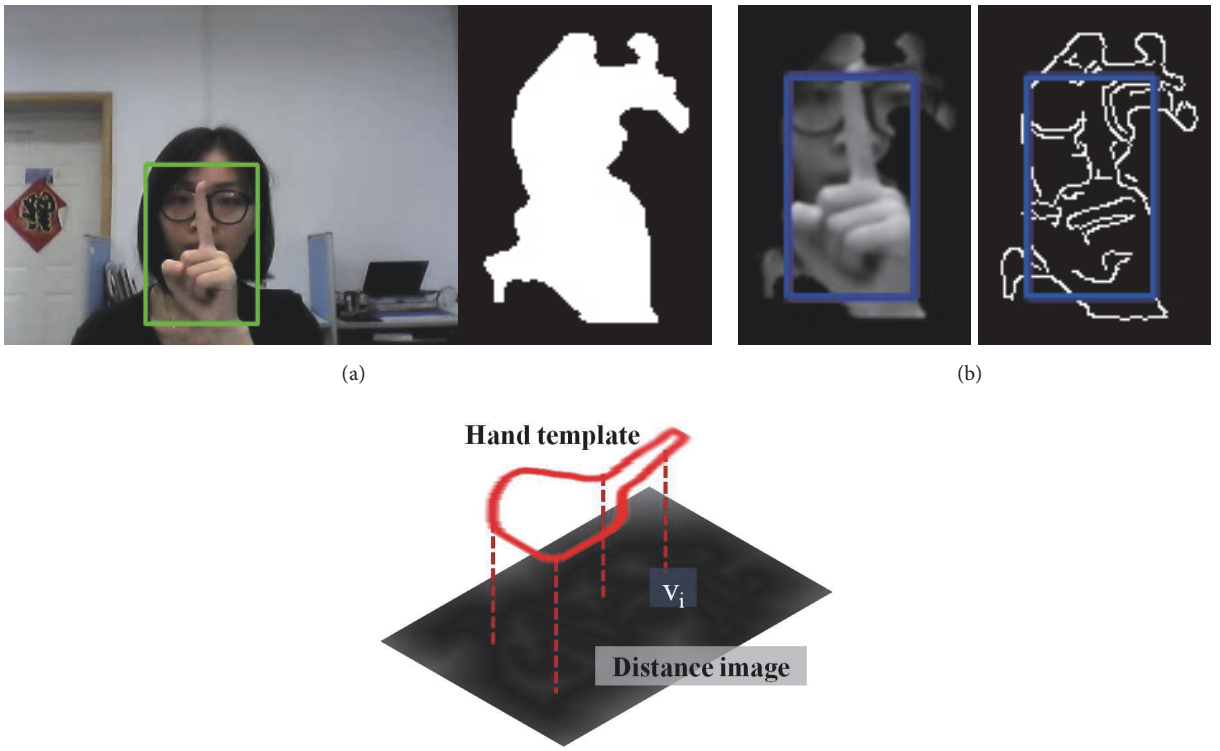

Figure 4: Template matching of the whole hand by HCMA. (a) Hand segmentation by skin color detection and background subtraction. (b) Edge extraction by Canny edge detector. (c) The edge distance is computed by the pixel values v_i, which the hand template hits.

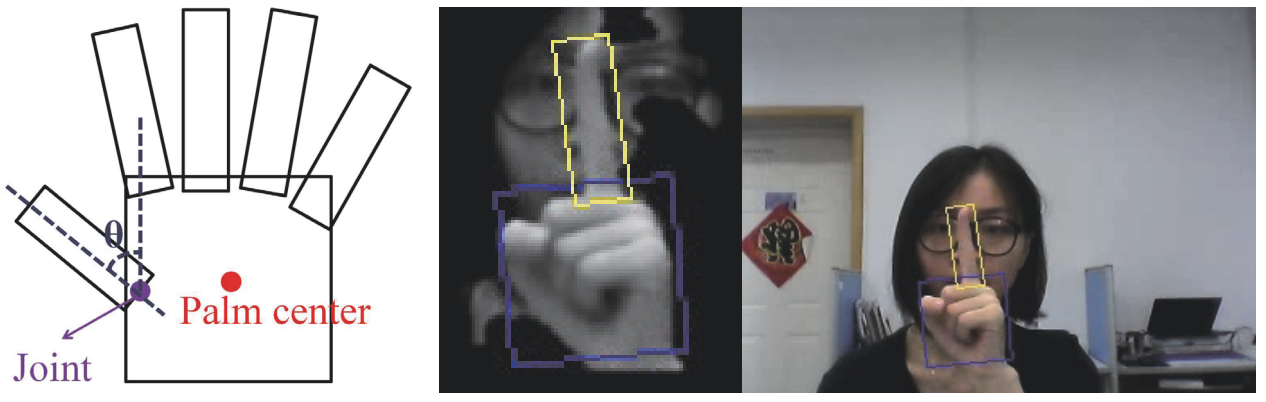

Figure 5: Palm and finger detection based on pictorial structure and HOG features. (a) Hand modeling. (b) Detection result.

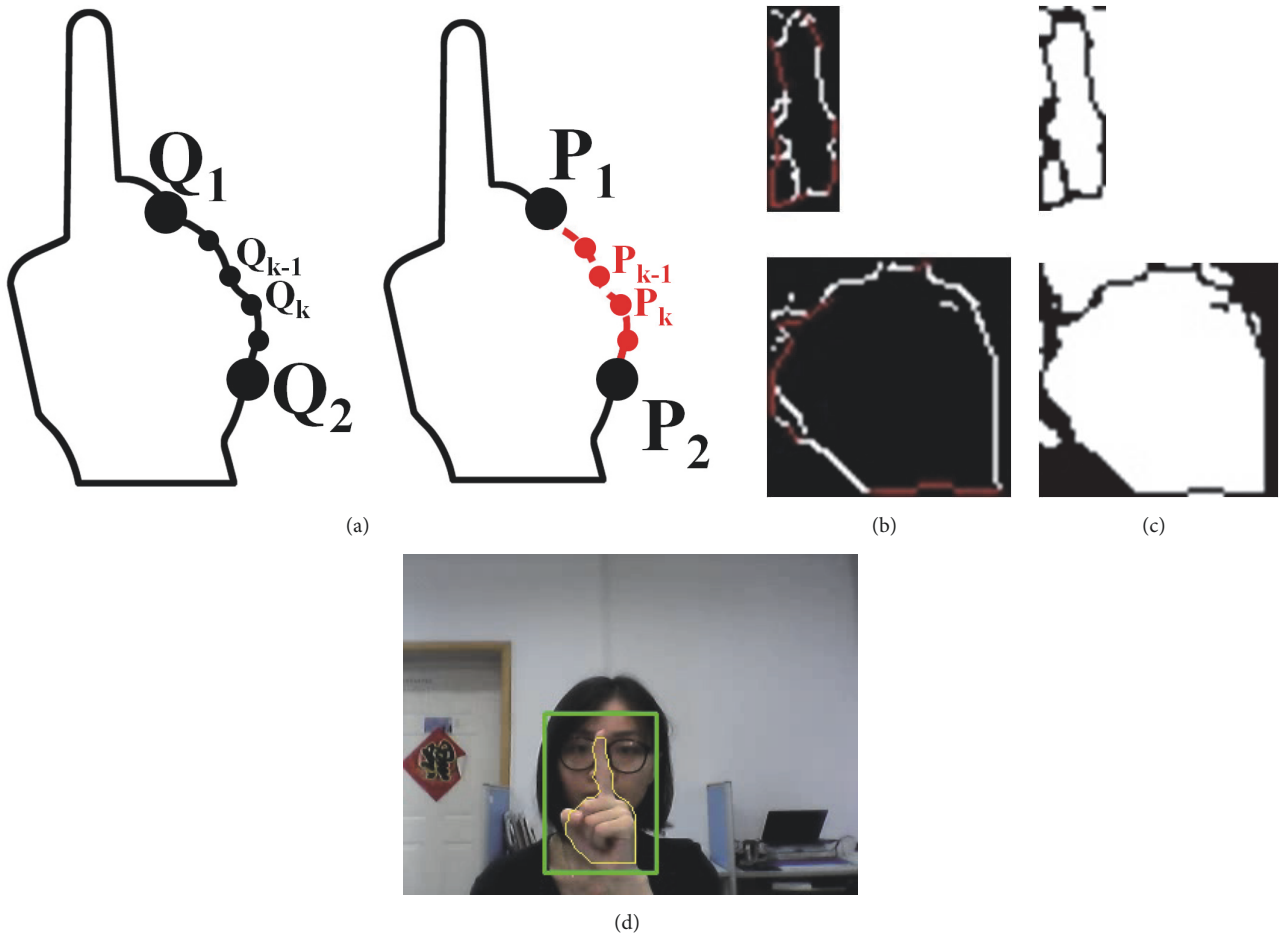

FIGURE 6: Edge repair. (a) Breakpoint connection. (b) Edge repair results. (c) Refined segmentation using edge repair results. (d) Combination of all subparts.

detect where the contour breaks. An edge point is determined as a breakpoint if one or two adjacent points among its eight-neighborhood are edge points. For each breakpoint P_i, a contour point Q_i in the prestored template is found, which is the closest to P_i based on Euclidean distance. As shown in Figure 6(a), in order to connect the adjacent breakpoints P_1P_2 depending on the template, Q_1Q_2 is divided into several subsegments by $\{Q_k \mid k = 1, \ldots, m\}$ and a set of $\{P_k \mid k = 1, \ldots, m\}$ is generated by $P_k = P_{k-1} + Q_k - Q_{k-1}$. Then, P_{k-1} and P_k are connected by the Catmull–Rom interpolation method [25]. Figure 6(b) shows the result of edge repair where the connections of breakpoints are illustrated in red.

Finally, the repaired edge images of all the subparts are used to extract the refined hand pixels from the coarse binary image as shown in Figure 6(c). All images of hand subparts are combined to generate the whole hand region in Figure 6(d).

2.2. Cursor Positioning. When both the eye and the fingertip are detected, the position on the screen where the user points to can be figured out by the line extending forward from the dominant eye to the fingertip. Since the depth information of the eyes and fingertips is unavailable and the pointing ways of different users are individually different, the offset exists between the target position and the calculated position. Thus, we propose an adaptive method of pointing direction estimation which can adjust the eye-fingertip line through a learning process.

As shown in Figure 7, a three-dimensional coordinate system with the camera position as the origin is established. The intersection point (x_i, y_i) of the screen and the eye-fingertip line can be calculated through similar triangle theory as

$$x_i = x_{E\_real} - \left(\frac{x_{E\_real} - x_{F\_real}}{z_{E\_real} - z_{F\_real}} \right) \cdot z_{E\_real}$$
$$y_i = y_{E\_real} - \left(\frac{y_{E\_real} - y_{F\_real}}{z_{E\_real} - z_{F\_real}} \right) \cdot z_{E\_real}, \quad (6)$$

where $(x_{E\_real}, y_{E\_real}, z_{E\_real})$ and $(x_{F\_real}, y_{F\_real}, z_{F\_real})$ represent the eye's coordinates and the fingertip's coordinates in the 3D coordinate system, respectively. Since the relationship of those coordinates in 3D coordinate system is similar to that

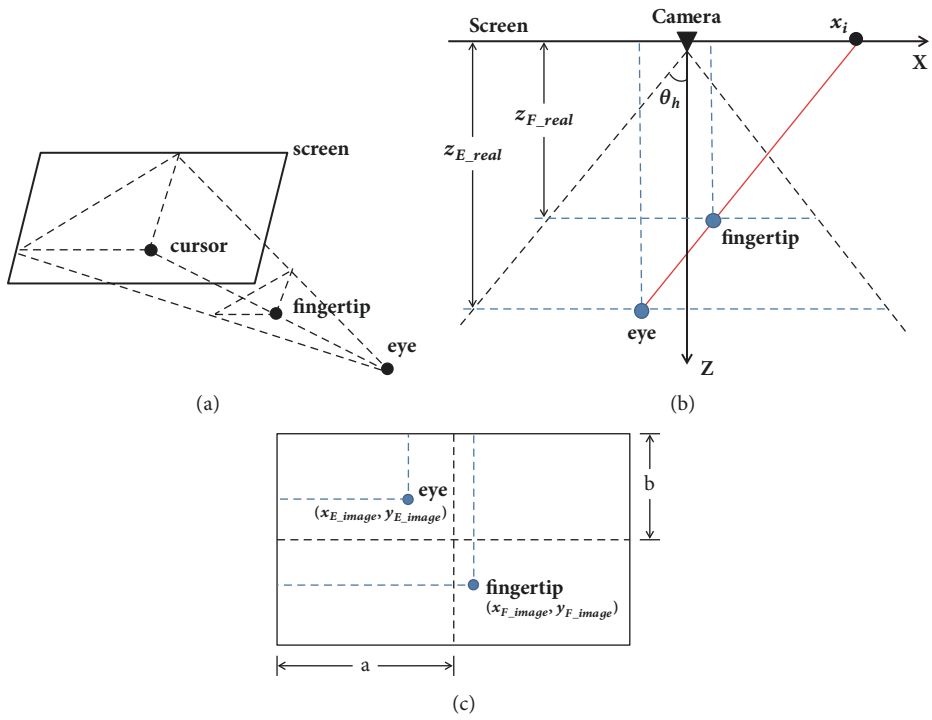

FIGURE 7: Cursor positioning based on the eye-fingertip line. (a) 3D coordinate system. (b) X-Z plane of (a). (c) The captured image with the resolution of 2a × 2b.

in the captured image (Figure 7(c)), $(x_{E\_real}, y_{E\_real}, z_{E\_real})$ can be computed by

$$x_{E\_real} = z_{E\_real} \cdot \tan\theta_h \cdot \frac{x_{E\_image} - a}{a}$$
$$y_{E\_real} = z_{E\_real} \cdot \tan\theta_v \cdot \frac{y_{E\_image} - b}{b}, \quad (7)$$

where $(x_{E\_image}, y_{E\_image})$ are the coordinates of the human eye in the image, θ_h and θ_v are the camera's angles of view in the horizontal and vertical direction, and (a, b) is half of the spatial resolution of the image. $(x_{F\_real}, y_{F\_real}, z_{F\_real})$ can be figured out by a similar equation to (7).

Then the cursor's coordinates $(x_{\text{cursor}}, y_{\text{cursor}})$ are computed by transforming (x_i, y_i) into the screen coordinates in pixels as

$$x_{\text{cursor}} = \left(x_i + \frac{1}{2}W\right) \cdot \frac{x_{\text{res}}}{W}$$
$$y_{\text{cursor}} = y_i \cdot \frac{y_{\text{res}}}{H}, \quad (8)$$

where $(x_{\text{res}}, y_{\text{res}})$ is the spatial resolution of the screen and W and H are the width and height of the screen.

It can be seen that the cursor's position is closely related to the distance between the fingertip and the screen z_{F_real} and the distance between the eye and the screen z_{E_real}. However, the two distances cannot be accurately obtained from our monocular vision-based system. The inaccurate distances will lead to an offset between the calculated position and the target position of the cursor. Hence, we propose a method to

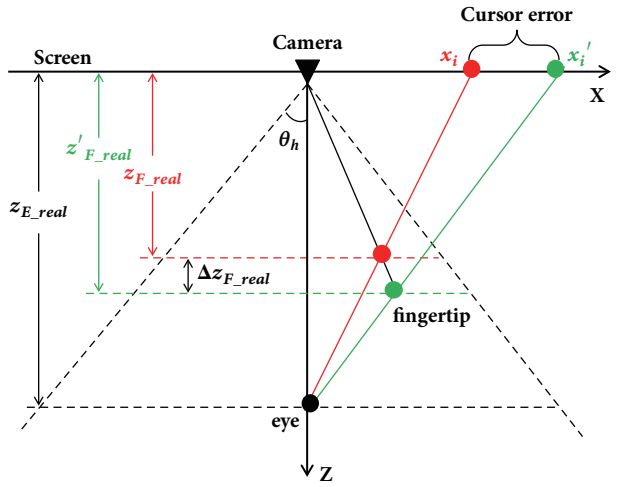

FIGURE 8: Error analysis model.

adjust the cursor's position through a learning process. Firstly, z_{E_real} is estimated according to the face's area and z_{F_real} is initialized to z_{E_real} minus 30 based on users' habits. Secondly, when the cursor is not located at the desired position, the user is allowed to alter the cursor's position by moving the fingertip slightly. The cursor's coordinate is adjusted from $(x_{\text{cursor}}, y_{\text{cursor}})$ to $(x'_{\text{cursor}}, y'_{\text{cursor}})$ by

$$x'_{\text{cursor}} = s_x \cdot d_x + x_{\text{cursor}}$$
$$y'_{\text{cursor}} = s_y \cdot d_y + y_{\text{cursor}}, \quad (9)$$

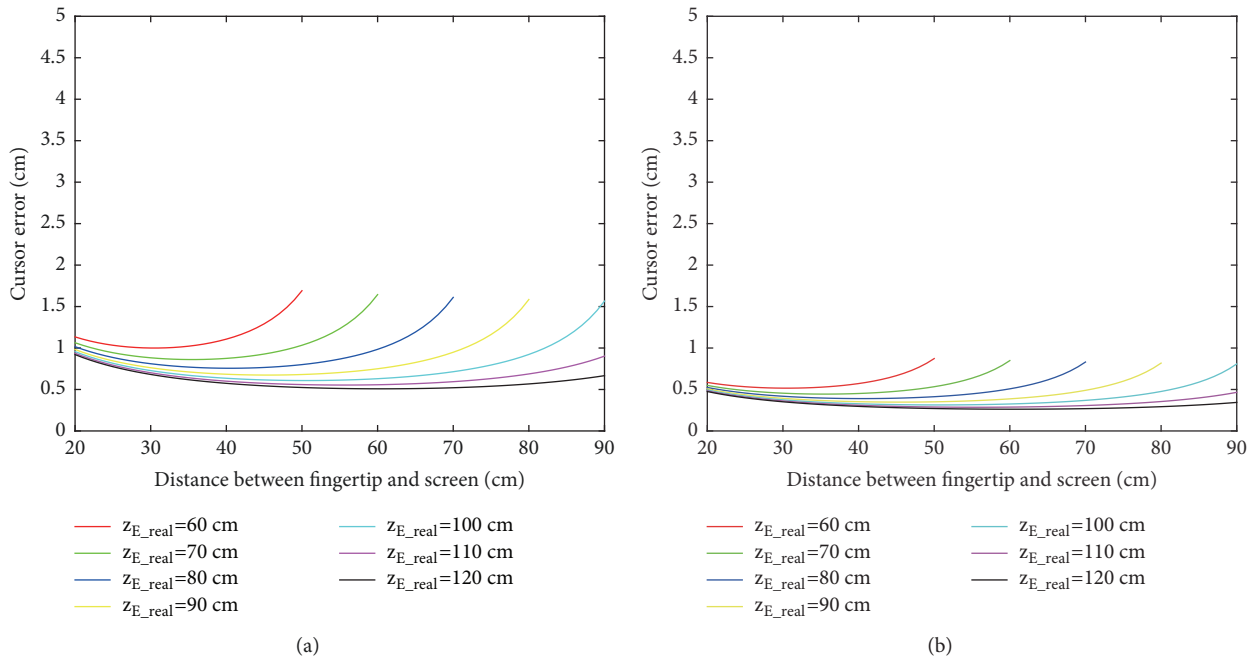

FIGURE 9: Cursor errors with different z'_{F_real} and z_{E_real} (a) when $x'_i = 15.5$ cm and (b) when $x'_i = 8$ cm.

where (d_x, d_y) represents the moving distance of the fingertip in the successive frames and (s_x, s_y) is the multiple coefficient. Our system monitors the fingertip's movement and records $(x'_{\text{cursor}}, y'_{\text{cursor}})$ when the fingertip moves a short distance after a pause. Then z_{F_real} can be adjusted by (10) from $z^{(n)}_{F\_real}$ to $z^{(n+1)}_{F\_real}$.

$$\begin{aligned} z^{(n+1)}_{F\_real} &= (1-\alpha)\, z^{(n)}_{F\_real} + \alpha \cdot z'_{F\_real} \\ &= (1-\alpha)\, z^{(n)}_{F\_real} + \alpha \cdot \frac{1}{2}\left(z^{x}_{F\_real} + z^{y}_{F\_real}\right), \end{aligned} \tag{10}$$

where α is the update rate. $z^x_{F\_real}$ and $z^y_{F\_real}$ are estimated based on x'_{cursor} and y'_{cursor}, respectively, according to the derivation of (6)–(9). After the adjusting process works several times, z_{F_real} will approach the real value.

2.3. Feasibility Verification of Our System. In order to verify whether our cursor positioning system is feasible or not, we perform error analysis to evaluate how much the cursor error depends on the eye-hand position in the direction of z-axis and the eye-hand position in the image. Because of the lack of depth information, the real locations of the fingertip and the eye in the direction of z-axis cannot be obtained, represented by z'_{F_real} and z'_{E_real}. Hence the estimated values of z_{F_real} and z_{E_real} are used in our pointing direction estimation method. Generally, the offsets of Δz_{F_real} and Δz_{E_real} exist between the real values and the estimated values, which lead to the errors of the cursor position. Similarly, assuming only X-Z plane is considered; Δx_{F_image} and Δx_{E_image} possibly exist due to the detection deviations of the fingertip's and eye's x-coordinates in the image. Note that the adjusting process of the cursor position is not activated here in order to analyze the error.

As shown in Figure 8, it is assumed that Δz_{F_real} exists and the eye is fixed on z-axis; the cursor's coordinate x_i is computed by

$$x_i = z_{E\_real} \cdot \tan\theta_h \cdot \frac{x_{F\_image} - a}{a} \cdot \frac{z_{F\_real}}{z_{E\_real} - z_{F\_real}}. \tag{11}$$

Then the cursor error Δx_i is calculated by subtracting the deviation value x_i from the real value x'_i as

$$\begin{aligned} \Delta x_i &= z_{E\_real} \cdot \tan\theta_h \cdot \frac{x_{F\_image} - a}{a} \\ &\quad \cdot \left(\frac{z'_{F\_real}}{z_{E\_real} - z'_{F\_real}} - \frac{z'_{F\_real} - \Delta z_{F\_real}}{z_{E\_real} - z'_{F\_real} + \Delta z_{F\_real}} \right), \end{aligned} \tag{12}$$

where $\Delta z_{F\_real} = z'_{F\_real} - z_{F\_real}$. $(x_{F\_image} - a)/a$ can be computed by using similar triangle principle in Figure 8. Therefore,

$$\begin{aligned} \Delta x_i = x'_i &- \frac{\left(z_{E\_real} - z'_{F\_real}\right) \cdot x'_i}{z'_{F\_real}} \\ &\cdot \frac{z'_{F\_real} - \Delta z_{F\_real}}{z_{E\_real} - z'_{F\_real} + \Delta z_{F\_real}}. \end{aligned} \tag{13}$$

Assuming the width of the screen is 31 cm, Figure 9 shows the errors of the cursor with different z'_{F_real} and z_{E_real} when the user points to the edge of the screen and a quarter of the screen. It is indicated that the cursor error becomes larger when the user points to the position closer to the screen edge. The cursor error increases as z_{E_real} decreases and as the hand

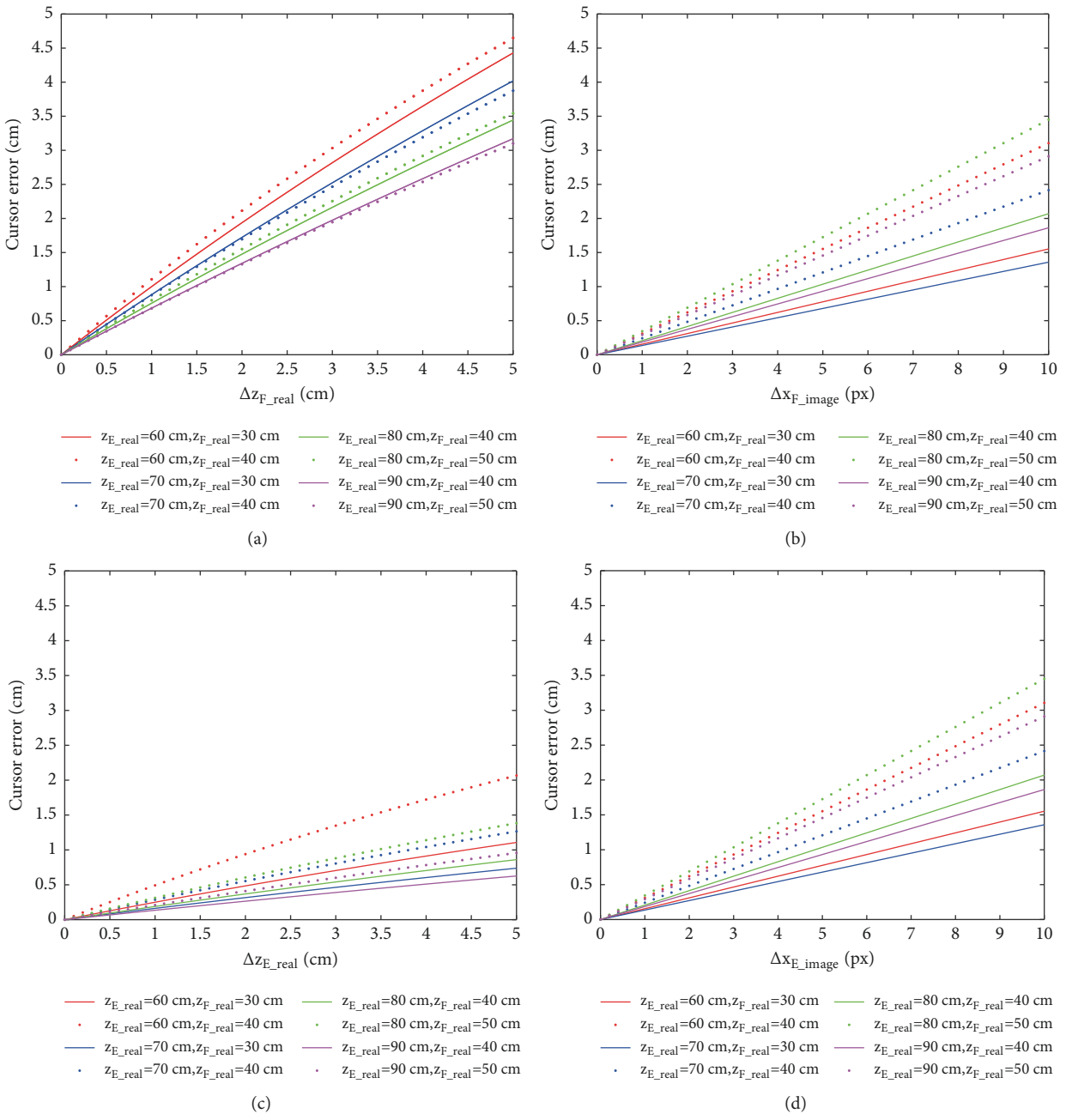

Figure 10: Maximum cursor errors effected by the four offsets of Δz_{F_real}, Δz_{E_real}, Δx_{F_image}, and Δx_{E_image}.

is closer to the head. Moreover, the cursor error is acceptable when operating the laptop in a close distance less than 100 cm and the error will be the smallest when the hand locates at the midpoint of the screen and the eye.

Figure 10 illustrates how much the cursor error depends on the four offsets of Δz_{F_real}, Δz_{E_real}, Δx_{F_image}, and Δx_{E_image} using the similar derivation of (13), where the two distances from the screen denoted by z_{E_real} and z_{F_real} are set according to users' operation habits. Not that the user is assumed to point to the edge of the screen, which causes the maximum error among the entire screen. As shown in Figure 10, the maximum cursor errors fall below 3 cm and 1.5 cm when the offsets of eye and hand positions on z-axis are less than 3 cm, and they fall below 3 cm when the offsets of eye and hand positions in the image are less than 9 pixels. The precision is acceptable for the block-level positioning and the following experiments will prove the attainability when the adjusting process is activated.

3. Experimental Results

Our proposed system was evaluated on a laptop with a 2D camera and the resolutions of the screen and the captured image were 1366 $*$ 768 and 320 $*$ 240, respectively. Besides,

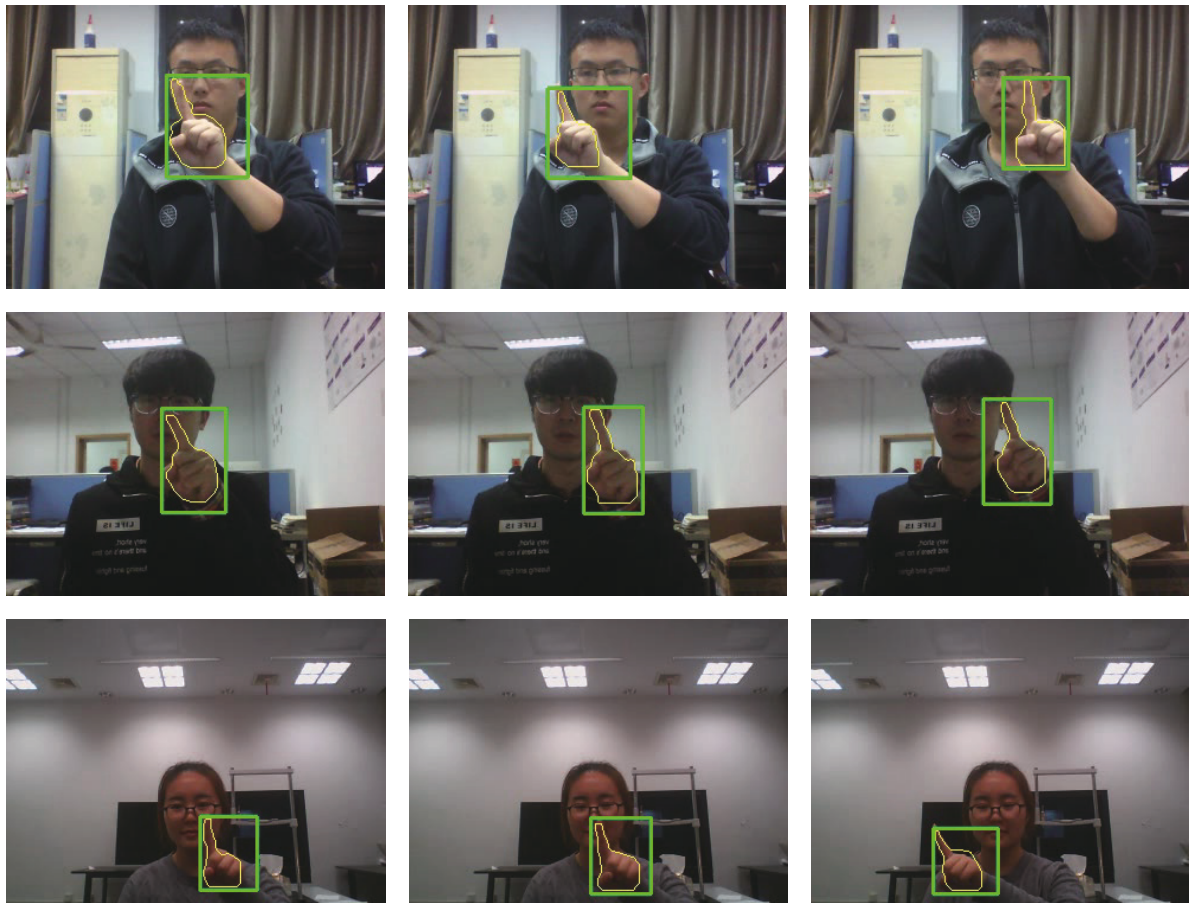

Figure 11: Segmentation results of the edge repair-based hand subpart segmentation algorithm under different situations.

on the basis of the proposed methods mentioned above, the user was not permitted to keep shaking his/her body back and forth which would affect the adjusting process of cursor positioning. Ten subjects were asked to operate the system by pointing gesture.

Firstly, some segmentation results of the edge repair-based hand subpart segmentation algorithm are shown in Figure 11, where yellow lines indicate the hand regions. It demonstrates that our method can extract hand pixels accurately under different complex backgrounds, including human faces with similar color. Moreover, the method is independent of users and robust to various hand appearances.

Then in order to evaluate the accuracy of cursor positioning, the computer screen was divided into several blocks as shown in Figure 12 and the subjects pointed to the four blocks marked with "1" to "4" repeatedly. Each block was pointed 15 times as a set of data.

Figure 12 shows the qualitative experimental results of cursor positioning by different subjects under different backgrounds. In each row, the same subject points to the different positions of the screen and the relative positions between the fingertip and the eye appear different. The cursors highlighted by the black circles are successfully located at the corresponding marked blocks. It is demonstrated that our cursor positioning method is robust to diverse individuals and different situations. Besides, it is also implied that our hand segmentation method works well when hand overlaps the face.

Table 1: Average errors of cursor positioning for the four marked blocks.

Error	d	d'(cm)
Block 1	0.707	2.38
Block 2	0.641	2.16
Block 3	0.680	2.29
Block 4	0.536	1.80
Average	**0.641**	**2.16**

Let the length of the blocks be 1. When the subject points to a block, the cursor error d between the calculated position and the desired position is computed by Euclidean distance. Figure 13 shows the errors of several sets of data. The horizontal axis represents the repeat times when the users point to the blocks. It is proved that the error decreases significantly after several times owing to the adjusting process in the adaptive pointing direction estimation method. Moreover, Table 1 shows the average errors of cursor positioning for the four marked blocks. The errors d' are estimated by $d' = d \times S^{1/2}$, where S represents the area of the block. Because the

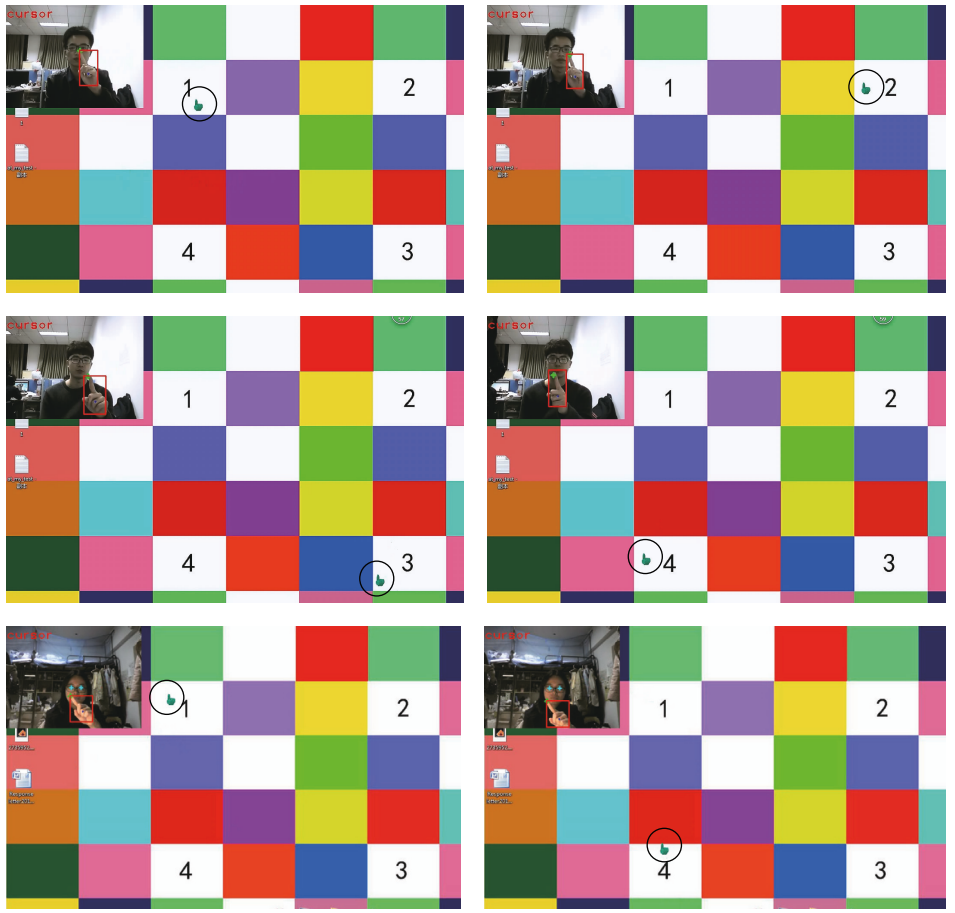

Figure 12: Qualitative experimental results of cursor positioning where black circles highlight the cursor's positions.

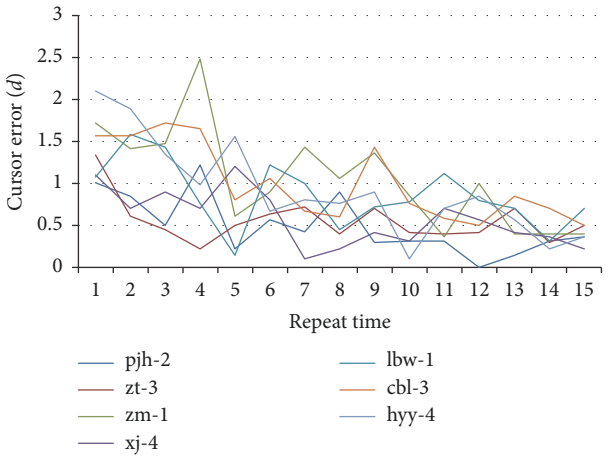

Figure 13: Cursor errors with the repeat times as the horizontal axis.

size of block icon in Windows 8/10 Metro can be set as 3 cm or 6 cm, it can be concluded that the positioning errors are small and acceptable for our application requirement.

Besides, our system can work in real time with an average speed of 131 ms per frame.

4. Conclusion

A security and smart Internet of Things interaction system based on hand gesture recognition is proposed in this work. When a user points to screen, the target position which the user points to is estimated by a straight line from the user's eye to the fingertip. Therefore, the interaction between human and computer should be activated by the coexisting of eye and hand. In our system, we employ a novel hand segmentation algorithm which combines the pictorial structure model, hierarchical chamfer matching algorithm, and curve fitting that segments hand regions accurately and efficiently. Furthermore, we propose an adaptive pointing direction estimation method for cursor calibration. An adjusting process is presented to correct the offsets between the target position and the calculated position arising from diverse individuals and lack of depth information. Experimental results show that our system provides a natural and friendly human-computer interaction and possesses satisfactory and accuracy of cursor positioning under complex backgrounds.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by Perspective Joint Research Project of Jiangsu Province Technology Project (BY2016076-07).

References

[1] Y. Liu, C. Cheng, T. Gu, T. Jiang, and X. Li, "A Lightweight Authenticated Communication Scheme for Smart Grid," *IEEE Sensors Journal*, vol. 16, no. 3, pp. 836–842, 2016.

[2] T. Zhang, L. Yan, and Y. Yang, "Trust evaluation method for clustered wireless sensor networks based on cloud model," *Wireless Networks*, pp. 1–21, 2016.

[3] X. Su, H. F. Yu, W. Kim, C. Choi, and D. Choi, "Interference cancellation for non-orthogonal multiple access used in future wireless mobile networks," *EURASIP Journal on Wireless Communications and Networking*, vol. 2016, no. 1, article no. 231, 2016.

[4] C. R. Panigrahi, J. L. Sarkar, and B. Pati, "Transmission in mobile cloudlet systems with intermittent connectivity in emergency areas," *Digital Communications and Networks*, vol. 4, no. 1, pp. 69–75, 2018.

[5] E. Park, Y. Cho, J. Han, and S. J. Kwon, "Comprehensive Approaches to User Acceptance of Internet of Things in a Smart Home Environment," *IEEE Internet of Things Journal*, vol. 4, no. 6, pp. 2342–2350, 2017.

[6] K. M. A. Alheeti, A. Gruebler, and K. McDonald-Maier, "Using discriminant analysis to detect intrusions in external communication for self-driving vehicles," *Digital Communications and Networks*, vol. 3, no. 3, pp. 180–187, 2017.

[7] J. W. Ho, *Distributed Detection of Node Capture Attacks in Wireless Sensor Networks*, InTech, 2010.

[8] P. Tague and R. Poovendran, "Modeling adaptive node capture attacks in multi-hop wireless networks," *Ad Hoc Networks*, vol. 5, no. 6, pp. 801–814, 2007.

[9] P. Tague, D. Slater, J. Rogers, and R. Poovendran, "Vulnerability of network traffic under node capture attacks using circuit theoretic analysis," in *Proceedings of the INFOCOM 2008: 27th IEEE Communications Society Conference on Computer Communications*, pp. 664–672, usa, April 2008.

[10] M. Conti, R. Di Pietro, L. V. Mancini, and A. Mei, "Mobility and cooperation to thwart node capture attacks in MANETs," *EURASIP Journal on Wireless Communications and Networking*, vol. 2009, Article ID 945943, 2009.

[11] H. Kim, Y. Kim, D. Ko, J. Kim, and E. C. Lee, "Pointing Gesture Interface for Large Display Environments Based on the Kinect Skeleton Model," in *Future Information Technology*, vol. 309 of *Lecture Notes in Electrical Engineering*, pp. 509–514, Springer Berlin Heidelberg, Berlin, Heidelberg, 2014.

[12] Z. Li and R. Jarvis, "Visual interpretation of natural pointing gestures in 3D space for human-robot interaction," in *Proceedings of the 11th International Conference on Control, Automation, Robotics and Vision, ICARCV 2010*, pp. 2513–2518, Singapore, December 2010.

[13] M. J. Reale, S. Canavan, L. Yin, K. Hu, and T. Hung, "A multi-gesture interaction system using a 3-D iris disk model for gaze estimation and an active appearance model for 3-D hand pointing," *IEEE Transactions on Multimedia*, vol. 13, no. 3, pp. 474–486, 2011.

[14] M. Sugi, M. Nikaido, Y. Tamura, J. Ota, and T. Arait, "Development of gesture-based interface for deskwork support system," in *Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2006*, pp. 5171–5176, chn, October 2006.

[15] R. C. Luo, S.-R. Chang, and Y.-P. Yang, "Tracking with pointing gesture recognition for human-robot interaction," in *Proceedings of the 2011 IEEE/SICE International Symposium on System Integration, SII 2011*, pp. 1220–1225, jpn, December 2011.

[16] Z. Černeková, C. Malerczyk, N. Nikolaidis, and I. Pitas, "Single camera pointing gesture recognition for interaction in edutainment applications," in *Proceedings of the 24th Spring Conference on Computer Graphics, SCCG 2008*, pp. 121–126, svk, April 2008.

[17] Y.-T. Li and J. P. Wachs, "Recognizing hand gestures using the weighted elastic graph matching (WEGM) method," *Image and Vision Computing*, vol. 31, no. 9, pp. 649–657, 2013.

[18] P. K. Pisharady, P. Vadakkepat, and A. P. Loh, "Attention based detection and recognition of hand postures against complex backgrounds," *International Journal of Computer Vision*, vol. 101, no. 3, pp. 403–419, 2013.

[19] M. Gonzalez, C. Collet, and R. Dubot, "Head Tracking and Hand Segmentation during Hand over Face Occlusion in Sign Language," in *Trends and Topics in Computer Vision*, vol. 6553 of *Lecture Notes in Computer Science*, pp. 234–243, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[20] P. Viola and M. J. Jones, "Robust Real-Time Object Detection," in *Proceedings of the in International Workshop on Statistical and Computational Theories of Vision – Modeling, Learning, Computing*, 2001.

[21] P. D. Le and V. H. Nguyen, "Remote Mouse Control Using Fingertip Tracking Technique," in *AETA 2013: Recent Advances in Electrical Engineering and Related Sciences*, vol. 282 of *Lecture Notes in Electrical Engineering*, pp. 467–476, Springer Berlin Heidelberg, Berlin, Heidelberg, 2014.

[22] P. Gejguš and M. Šperka, "Face tracking in color video sequences," in *Proceedings of the Spring Conference on Computer Graphics, SCCG 2003 - Conference Proceedings*, pp. 245–249, svk, April 2003.

[23] G. Borgefors, "Hierarchical Chamfer matching: a parametric edge matching algorithm," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 10, no. 6, pp. 849–865, 1988.

[24] H. Zhong, J. P. Wachs, and S. Y. Nof, "A collaborative telerobotics network framework with hand gesture interface and conflict prevention," *International Journal of Production Research*, vol. 51, no. 15, pp. 4443–4463, 2013.

[25] J. Tsao, "Interpolation artifacts in multimodality image registration based on maximization of mutual information," *IEEE Transactions on Medical Imaging*, vol. 22, no. 7, pp. 854–864, 2003.