Scientia Iranica B (2015) 22(4), 1501–1509

Sharif University of Technology
Scientia Iranica
Transactions B: Mechanical Engineering
www.scientiairanica.com

Implementation of instant numeral recognition in space trajectory

J.-S. Sheu* **and K.-C. Wang**

Department of Computer Science, National Taipei University of Education, Taipei, 10671, Taiwan.

Received 22 May 2014; received in revised form 16 July 2014; accepted 4 October 2014

KEYWORDS
Digit recognition;
Feature extraction;
Motion-sensing.

Abstract. This study implemented the hand gesture digit recognition function by using the Kinect motion sensor. This interactive motion-sensing technology was employed to track, record, and convert the trajectories of handwritten numerals in space into corresponding shapes of digits on the plane. Subsequently, the feature matching method was used to achieve digit recognition. In the recognition process, first binarization and Hilditch thinning algorithm were performed on the region for recognition, and the segmentation algorithm proposed by G.E.M.D.C. Bandara etc. was used to perform segmentation of the region after thinning. This paper proposes the improvement of the conditions of the arc type of the discriminant to achieve feature extraction and enhance the digit recognition rate.

© 2015 Sharif University of Technology. All rights reserved.

1. Introduction

The rapid advancement of technology continues to make daily life in modern society more comfortable. Whether in research or in entertainment, the use of computer tools and human-machine interfaces have become the norm. In sci-fi films and television shows, electronic tools and interfaces that can be manipulated by physical actions or gestures are often featured. Such an achievement in reality would be of considerable help to humankind.

The body sensing device allows us to manipulate computer-related devices and interfaces more effectively. Moreover, this device is combined with skeletal tracking and detection technology to process the screen image and show the position of skeleton nodes. These skeleton nodes can be detected and recorded in real-time. Gesture recognition can manipulate the functions of the human-machine interface and digit recognition.

The purpose of this study is to combine body-sensing controllers to achieve digit recognition systems with real-time detection and the recording of dynamic positioning.

2. Literature review

Kinect sensors have proven effective in web interface manipulation. The NITE and OpenNI interface designs [1-3] can collect and transfer information from the Kinect sensor, as well as control data flow, including skeleton data. The functions of the design are described below:

1. Image update: Forces OpenNI to receive a new framework from Kinect, which is then transferred to the process of NITE.
2. Skeleton update: Contains skeleton data of all the skeleton nodes; if a valid detected skeleton is in action, all data on the joint coordinate positions will be passed to the main program.
3. Session management: Forces the session management to handle current hand nodes and updates

*. *Corresponding author. Tel.: +886 2-27321104; Fax: +886 2-27375457*
E-mail address: jiashing@tea.ntue.edu.tw (J.-S. Sheu)

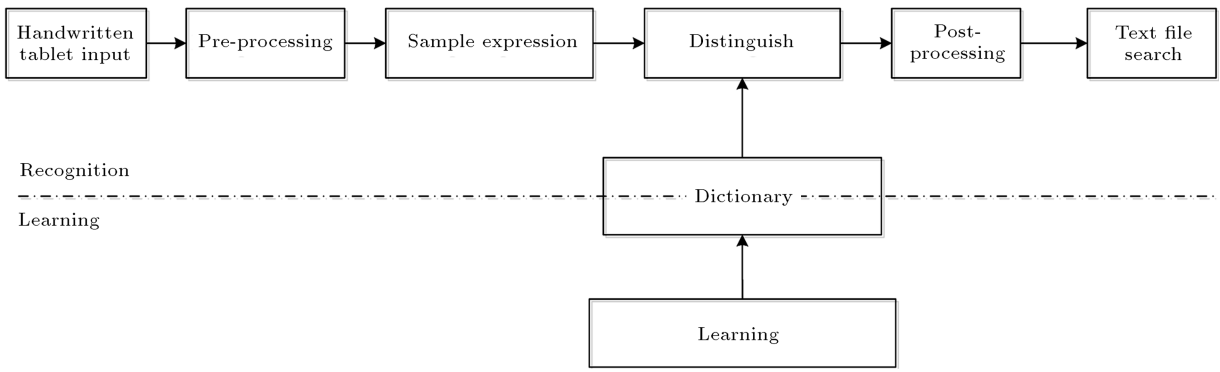

Figure 1. The principle framework of connection character recognition.

the location of all nodes in response to a function call.

4. Fist sensing: The hand is determined as either a fist or a palm, based on the shape of its portion, which is achieved by calculating the protruding area of the hand portion and comparing it with the area in a normal state.

To develop a Kinect with Google Earth functions, called Kioogle, the body sensing principle is employed, which involves the use of different gestures and postures as a pass, interpreting Kinect's skeleton node messages, and controlling the desired functions [2]. The map control mode contains several features, which are discussed in the remaining items on this list.

5. Horizontal movement: The user must present the palm of one hand. Based on the detected coordinate position of the palm on the screen, the speed variable of the slight movements of the other hand's fist will induce the screen to move.
6. Zooming: The user must present both hands for gesturing. The distance between both hands (i.e., whether close or far) will determine if the image will be enlarged or shrunk.
7. Rotation: The user must present both hands to gesture and move them along the direction of the Kinect's Y-axis (vertical direction) for detection; opposite movements will cause the plane to rotate on the screen.
8. Tilting: The user must present both hands to gesture and move them along the direction of the Kinect's Z-axis (depth direction) for detection; opposite movements will cause the non-plane to rotate on the screen.

This task involves studying how to make the computer capable of literacy. At present, the process of character recognition includes programming text into the computer and running several models and algorithms, followed by the recognition of features. Owing to the difference in input information, character recognition is categorized into connection recognition and offline recognition. Online and offline recognitions are briefly explained in the following.

Compared with offline recognition, connection recognition utilizes time, strokes, and other dynamic information. As shown in Figure 1, the principle framework of connection character recognition is applicable to offline character recognition, as long as it is programmed into the scanner or another image instead of a handwritten tablet.

As shown in Figure 1, text input through a handwritten tablet forms a chronological sequence of point coordinates, which can add pressure in detecting the size sequence. The pre-processing stage includes normalization, binarization, and noise removal. After feature extraction [4], the text must be compared with the feature set containing known texts in the dictionary to achieve the recognition of the words. The post-processing stage of the system involves determining the results of the word recognition process as text outputs [5]. At this stage, linguistic knowledge is used to manipulate the results, such as automatic error correction and semantic error findings, to continuously enhance the recognition rate of the system.

A number of sample expression forms and corresponding dictionary forms exist. Each form can select different features or essential constituent units. Moreover, each feature or basic constituent unit employs different extraction methods. However, these methods are generally classified into two types, namely, the statistical decision method, the syntactic structure method, and the fuzzy mathematics and artificial intelligence methods [6-10].

The object to be processed is a two-dimensional image. The difference between the offline recognition principle and connection recognition lies in the pre-processing stage. In connection recognition, less dynamic information can be used, but the picture must be saved and addressed after the completion of the text. In the pre-processing stage of offline recognition, the original image must be converted into a form acceptable to the recognizer via binarization,

tilt correction, or adjusting the thickness (refinement) of the strokes. Given that general offline recognition will extract features in the picture after pre-processing, when this step is less than ideal, the picture often reverts to a state with an uncorrectable error. Therefore, offline recognition, in practice, will usually add some models or methods, such as artificial neural networks and statistical methods, as well as hidden Markov models and fuzzy logic, to improve the soundness of the pre-processing stage [11,12].

3. Research methods

This research established the system flow chart shown in Figure 2. The first step involves detecting the human skeleton and recording information on the coordinates of the joints in the left and right hands, and the shoulder center. The three-joint coordinate information can be used to design the gestures after the skeleton data and crawl have been successfully detected. The three designed function modes include brush switch, clear screen, and closing the window. The brush switch function controls the right-hand brush, which can draw figures for recognition on the canvas. The clear screen function clears the handwriting on the canvas. The function for closing the window shuts down the system interface window, clears the screen, and closes the window functions to execute the off state of the brush and avoid triggering errors. Finally, the digit recognition system is used for recognizing images.

3.1. Skeleton coordinates crawl

In this paper, the coordinate system used 3D Kinect skeleton data. The coordinate positions of the joints are automatically plotted when the user is detected, and the skeleton data are gathered by the Kinect sensor [13,14]. The experimental process is shown in Figure 3.

3.2. Brush feature

The skeleton coordinate crawl function grabs the coordinate node positions of the right hand, thus creating and updating the coordinates instantly. The drawn line connects the two point coordinates, which achieves the effect of the instant draw stroke, as shown in Figure 4.

3.3. Gesture judgment

As described below, combining the body sensing principle with Kinect enables the system's function mode to use different gestures to pass and explain the skeleton node information, as well as trigger the required functionality of Kinect.

1. The brush switch: At program startup, the default state is the off function. The coordinate node of the

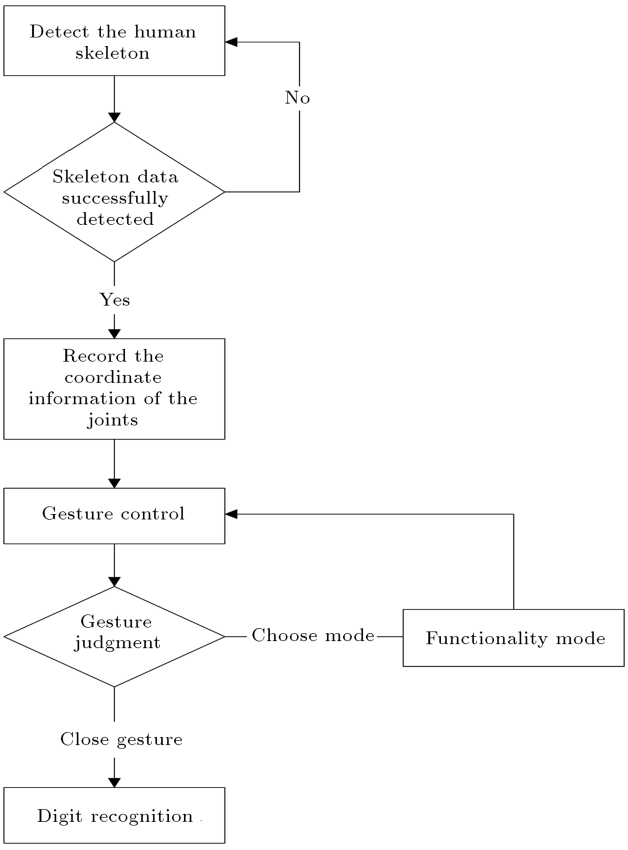

Figure 2. System flow chart.

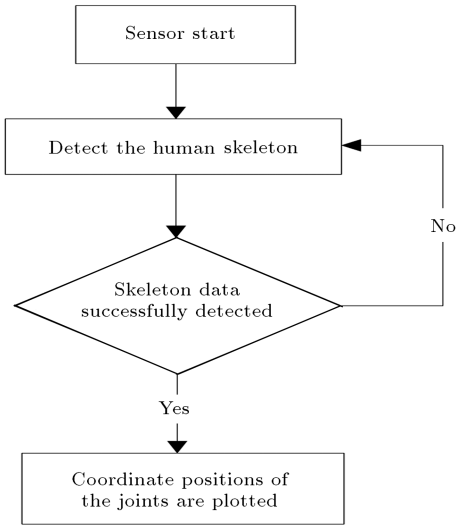

Figure 3. Skeleton coordinates crawl.

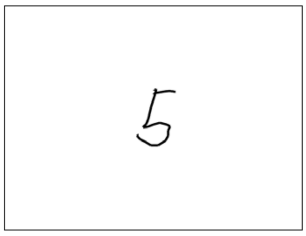

Figure 4. Using brush to draw 5.

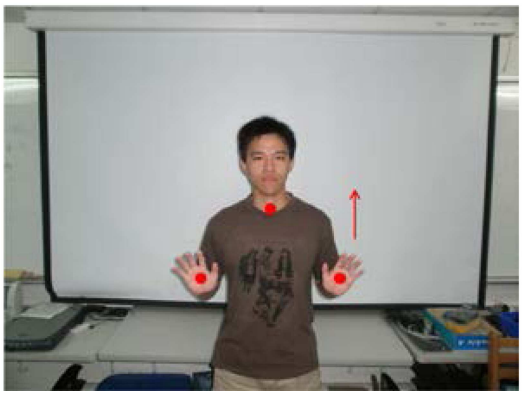

Figure 5. The brush switch gesture.

shoulder center is set as a reference point. Given that most people are used to holding a pen with the right hand, the coordinates of the right palm are set as the brush coordinates. Considering that the right hand is the brush, the left palm is thus set as the brush switch control. The brush switch will be triggered when the left palm exceeds the positive direction of the reference point in Kinect's Y-axis (vertical direction). Using this gesture once more will switch to the opposite state, as shown in Figure 5.

2. Clear screen: The state of the brush mode must be off for the user to start in order to prevent the accidental triggering of this feature which will clear the handwriting. The shoulder center coordinate node is the reference point. The clear screen function will be triggered when the right palm exceeds the positive direction of the reference point in the Kinect X-axis (horizontal direction) by some distance, as shown in Figure 6.
3. Close skeleton detection: The state of the brush mode must be off for the user to start, to prevent the accidental triggering of this feature to clear the handwriting. This feature requires both hands to be presented to make hand gestures. The close skeleton detection function will be triggered when the position of the left and right palms in the Kinect X-axis coordinate (horizontal direction) and Y-axis coordinate (vertical direction) are close to each other, as shown in Figure 7.

Figure 6. Clear screen gesture.

Figure 7. Close skeleton detection gesture.

3.4. Gesture control description

The coordinate positions of three joint points, namely, shoulder center, right palm, and left palm, are used to achieve the action design of gestures. The operation instructions of a variety of modes are introduced below:

1. The brush switch: This feature is at off state by default, and the brush function can be turned on or off when the left hand of the user is raised above shoulder height. To turn the switch on or off, the left hand is lowered below shoulder height and lifted once again, as shown in Figure 8.
2. Clear screen: This function can only be performed when the brush function is at off state, and can be triggered when the right palm of the user is raised some distance above the right side of the shoulder center. The operation is similar to pushing the right hand outward, as shown in Figure 9.

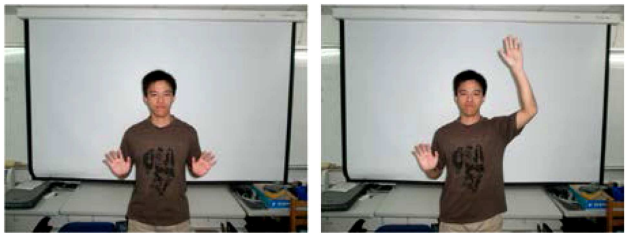

Figure 8. Procedure for the brush switch.

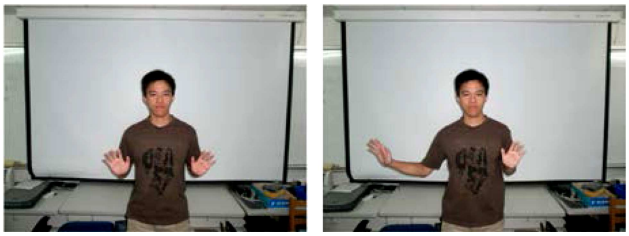

Figure 9. Procedure for clear screen.

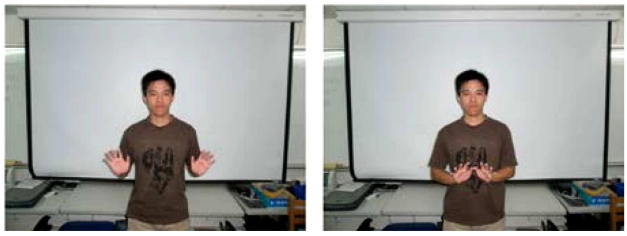

Figure 10. Procedure for closing the window.

3. Closing the window: This function can only be performed when the brush function is at off state. The function, which requires both hands to gesture, can be triggered when the user's hands are close to each other within a certain distance. The operation is similar to putting hands close together, as shown in Figure 10.

3.5. Digit recognition

3.5.1. Segmentation algorithm

Two rules have to be satisfied in determining the main starting point. The column-based search is in the refinement part. First, find only one neighboring point in the refinement part as the main starting point, that is the so-called endpoint. For example, the main starting point of the letter B is found, as shown in Figure 11(i), and the three points of the point are labeled "a" "b" and "c." Second, the first rule cannot be used to judge or search text such as "o" or "0." Given that all points have two adjacent points, that is, no endpoint is used, only the second rule can be used as the basis for judging. This rule states that the highest column-based point will be searched as the main starting point if only one neighboring point cannot be found. Following this rule, the letter "o" can be searched for the main starting point "a", as shown in Figure 11(ii).

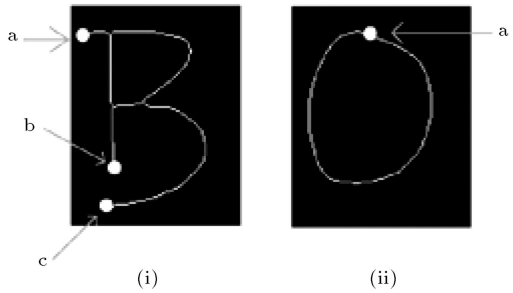

Figure 11. Marked main starting point.

In this example, B is taken as the text skeleton, as shown in Figure 12. This text skeleton search starts from the main starting point "a". At the junction point of the current visit, when a visit is made to branch K_1, two neighbors will be unvisited. As such, the direction of the current visit is first towards the right. Subsequently, N_1 will become the next point to visit. Another point, N_2, in the K_1 branch is the secondary starting point of the K_1 branch; as such, N_2 will be the starting point of another path in this skeleton. Therefore, N_2 will be placed in a secondary starting point in the queue, and will then be treated as the starting point of the visit. In each branch of the text skeleton, zero or more points will be placed in a secondary starting point in the queue.

Each partition will be generated in the search path during the tracking process. At the junction of each skeleton, a secondary starting point will be produced and added into another queue. If a point is found with no points to visit around it, then it is called the end point of the search. When the end point is reached, the entire text skeleton is searched from the secondary starting point until all unvisited points are visited. All visited paths are recorded, and only those unvisited points are visited. Therefore, the same path can only be visited once. After tracking, these paths will be cut into meaningful segmentation.

The search rule for searching unvisited neighboring points from an arbitrary direction involves beginning in a clockwise order from the "top" direction of the point (i, j) currently visited, as shown in Table 1. If the "top" does not meet the unvisited conditions, the "top right" direction will be found next. The rule then sequentially finds "top", "top right", "right", "lower right", "lower", "lower left", "left" and "upper left".

Table 1. Search rule for searching unvisited neighboring points.

Upper left	Top	Top right
Left	(i, j)	Right
Lower left	Lower	Lower right

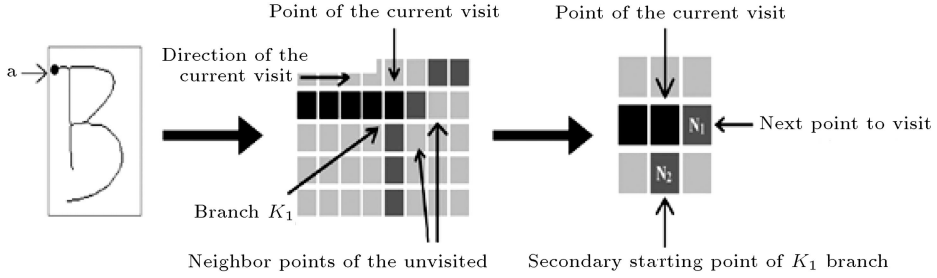

Figure 12. Direction of the visit of text skeleton.

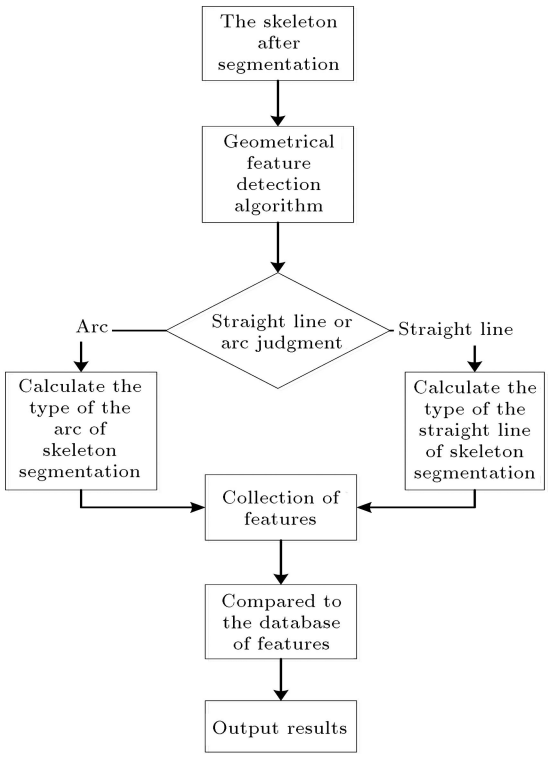

Figure 13. Recognition flow chart.

3.5.2. Recognition

As shown in Figure 13, the skeleton after segmentation will undergo the detection algorithm for geometrical features to calculate the value of the straight line or the arc of skeleton segmentation. What follows is the respective calculation of the type of skeleton segmentation, and the features are then collected and compared with the database of feature output results [5].

1. Geometrical feature detection algorithm [7,9]: After segmentation, the segment is judged to be a straight line or an arc. All the sub-divisions must undergo this determination before recognition. This method utilizes the least squares difference of the line segment. The distance of each small segment, as well as its radian, is calculated.

   The calculation formula is given by:

$$\mathrm{mSTR} = \mu_{\mathrm{STRAIGHTNESS}}^{\mathrm{seg}(n)} = \left[\frac{d_{p_0p_n}^{\mathrm{seg}(n)}}{\sum_{k=0}^{N-1} d_{p_kp_{k+1}}^{\mathrm{seg}(n)}}\right], \quad (1)$$

$$\mathrm{mARC} = \mu_{\mathrm{ARC-NESS}}^{\mathrm{seg}(n)} = \left[1 - \frac{d_{p_0p_n}^{\mathrm{seg}(n)}}{\sum_{k=0}^{N-1} d_{p_kp_{k+1}}^{\mathrm{seg}(n)}}\right]. \quad (2)$$

   In Eqs. (1) and (2), $d_{p_kp_{k+1}}^{\mathrm{seg}(n)}$ express the n-th segmentation of point k and point $k+1$, respectively, and the number of points in each segmentation is N. Subsequently, whether the segment is a straight line or an arc is determined, which is achieved via a comparison to a threshold value. For example, when the threshold value of $\mu_{\mathrm{STRAIGHTNESS}}^{\mathrm{seg}(n)}$ [8] is greater than 0.6, this segment can be considered a straight line. When the threshold value of $\mu_{\mathrm{ARC-NESS}}^{\mathrm{seg}(n)}$ is greater than 0.6, this segment can be considered an arc. However, the threshold sum of $\mu_{\mathrm{STRAIGHTNESS}}^{\mathrm{seg}(n)}$ and $\mu_{\mathrm{ARC-NESS}}^{\mathrm{seg}(n)}$ cannot exceed the selected threshold value. All segmented threshold values must satisfy the following equation:

$$\mu_{\mathrm{ARC-NESS}}^{\mathrm{seg}(n)} + \mu_{\mathrm{STRAIGHTNESS}}^{\mathrm{seg}(n)} = 1. \quad (3)$$

   In the computation of this algorithm, calculating the two values of $\mu_{\mathrm{ARC-NESS}}^{\mathrm{seg}(n)}$ and $\mu_{\mathrm{STRAIGHTNESS}}^{\mathrm{seg}(n)}$ is important in order to determine whether the segment is a straight line or an arc.

2. Defining feature database [10-12]: The feature array is divided into two parts, namely, line portion and arc portion. In each text skeleton, a group of digital arrays will be saved to determine the type of each segment, as shown in Table 2.

4. System environment

Figure 14 shows the system structure. The sensor is Kinect for Xbox and elevated 0.9 meters from the ground. The distance of the user is 1.9 meters. The horizontal viewing angle of Kinect is 57 degrees, centered, approximately, with the sensor 28.5 degrees from the left and right. The vertical viewing angle is 43 degrees, which can be adjusted up or down by 27 degrees, with no adjustment of pitch angle. The relevant skeleton data detected by the sensor will be

Table 2. Feature database.

Feature number	Feature type	Feature shape
0	Horizontal line	
1	Vertical line	
2	Negative slanted line	
3	Positive slanted line	
4	D-like arc	
5	C-like arc	
6	A-like arc	
7	U-like arc	
8	O-like circle	

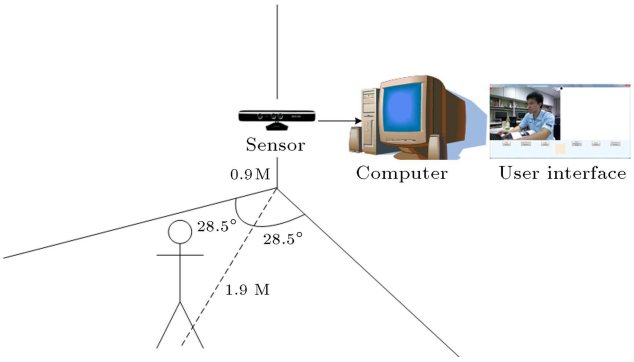

Figure 14. System structure.

Table 3. Kinect SDK opened with 20 joints.

Joint ID	Joint name
0	Hip center
1	Spine
2	Shoulder center
3	Head
4	Shoulder left
5	Elbow left
6	Wrist left
7	Hand left
8	Shoulder right
9	Elbow right
10	Wrist right
11	Hand right
12	Hip left
13	Knee left
14	Ankle left
15	Foot left
16	Hip right
17	Knee right
18	Ankle right
19	Foot right

passed via USB to the computer, which then passes the information to the interface for processing [15-17].

The Kinect sensor can detect and obtain the skeleton streaming information. The Kinect skeleton streaming information will send information from six users back to the data, but only two of six users can be picked for tracking at each time. Each user can be recognized with 48 joints (joints, contact). Currently, Kinect SDK can only be opened with 20 joints, as shown in Table 3.

The Kinect skeleton coordinate system defined three dimensions, namely, X, Y, and Z. As shown in Figure 15, the Z-axis is the direction of the sensor faced, and the unit is meters (m).

Figure 16 shows the control interface of the

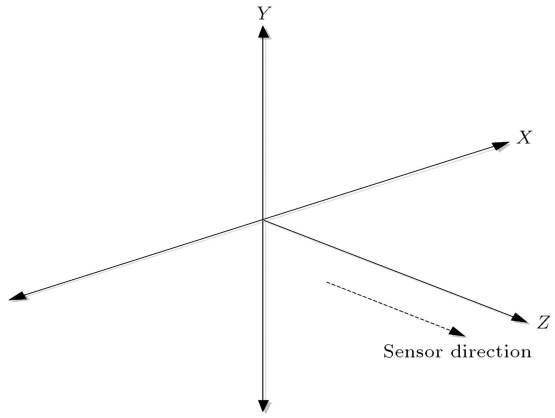

Figure 15. Kinect skeleton coordinate system.

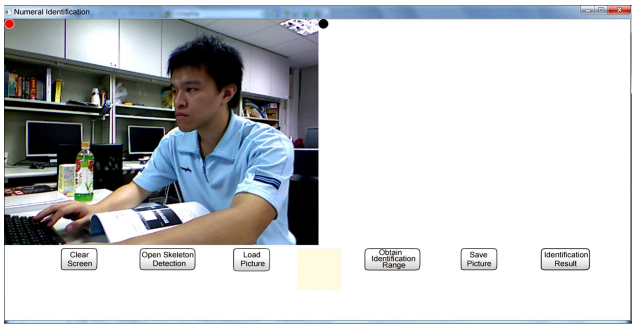

Figure 16. System interface and monitor.

completed development of the system. The control interface is divided into four blocks. As shown in the image block, the first block is the Kinect camera screen, and the second block is the canvas, as shown in the white block part. The third block consists of four buttons, which are for clearing the screen, opening skeleton detection, reading the picture, obtaining the recognition range, and saving the picture, respectively. The fourth block recognizes the results and shows the block, as indicated in the bottom of the middle block in Figure 16. The fourth block contains one white display block and a button for displaying the recognition results.

5. Experimental results

The state of the Kinect skeleton detection is on when the system is turned on. The user in the operating range can start the operation, as shown in Figure 17.

After the process of recognizing the handwritten numerals, the recognition results will be displayed in the interface. Presentation of the recognition results in the system interface is shown in Figure 18.

The digit recognition ranges from 0 to 9. The brush inputs the numbers from 0 to 9. The feature database includes a set of numbers from handwriting to segmentation, displaying database definitions and results, as shown in Figure 19.

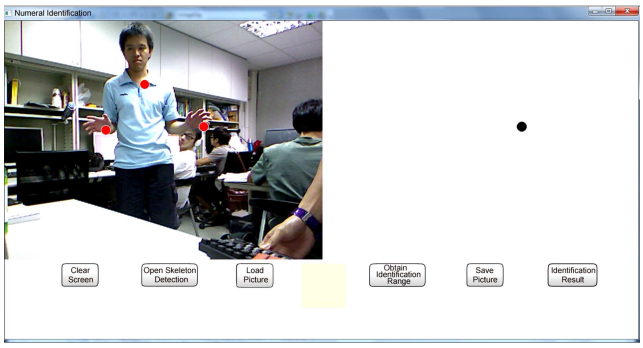

Figure 17. User detection.

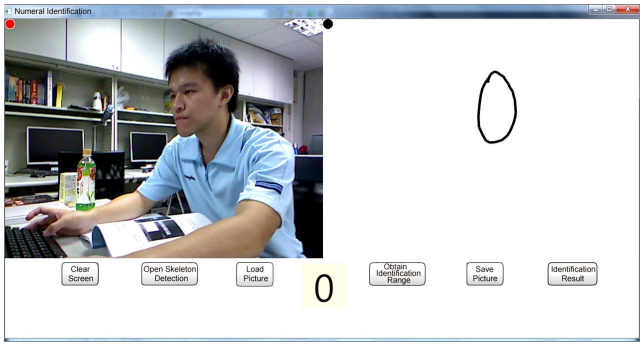

Figure 18. Recognition results presented in system interface.

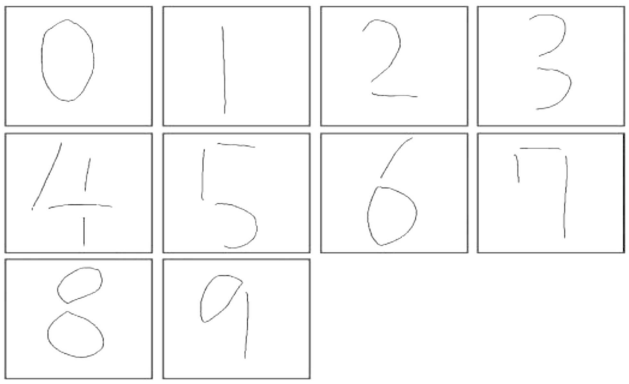

Figure 19. Feature database of numbers 0 to 9.

The handwritten numerals 0 to 9 and a total of 1000 words are the samples. Formula 4 calculates the recognition rate:

$$\text{Recognition rate} = \frac{\text{Recognized}}{\text{Recognized} + \text{Unrecognized}} \times 100\%. \quad (4)$$

After an input of 1000 random handwritten numerals, a total of 65 words can be recognized using the system and method introduced in the present study. The recognition rate of Eq. (4) is calculated to be 65%. The average time of recognition is also shown in Figure 20. The maximum average time is lower than 0.35 seconds.

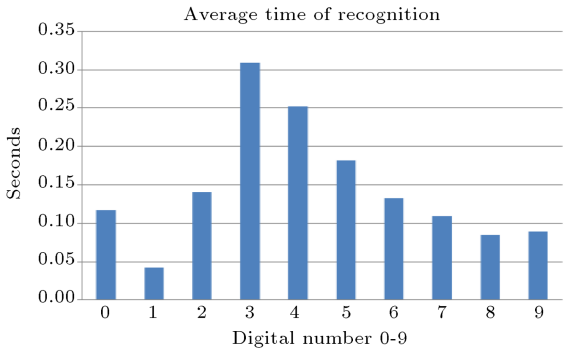

Figure 20. Digital identification average time for numbers 0 to 9.

6. Conclusions

Through the skeleton node detection functionality of Kinect, gestures can be designed to achieve operation control of a man-machine interface, and combined with digit recognition to transform the user's hand joints into a writing brush. This study utilizes the concept of somatosensory interactive technology to enhance the convenience and environmental protection of use. However, the overall system process is insufficiently automated. Further improvements in the future should start from this aspect to design a complete set of gestures that will replace the button function. Moreover, the functionality of word selection or the feature database should be increased to improve the recognition rate.

References

1. References Livingston, M.A., Sebastian, J., Ai, Z. and Decker, J.W. "Performance measurements for the Microsoft Kinect skeleton", *IEEE Virtual Reality Workshops (VR)*, pp. 119-120 (2012).

2. Boulos, M.N.K., Blanchard, B.J., Walker, C., Montero, J., Tripathy, A., and Gutierrez-Osuna, R. "Web GIS in practice X: A Microsoft Kinect natural user interface for Google earth navigation", *International Journal of Health Geographics*, **10**, pp. 45-58 (2011).

3. Frati, V. and Prattichizzo, D. "Using Kinect for hand tracking and rendering in wearable haptics", *IEEE World Haptics Conference (WHC)*, pp. 317-321 (2011).

4. Bandara, G.E.M.D.C., Pathirana, S.D. and Ranawana, R.M. "Use of fuzzy feature descriptions to recognize handwritten alphanumeric characters", *Proc. of 1st Conference on Fuzzy Systems and Knowledge Discovery*, pp. 269-274 (2002).

5. Hilditch, C.J., Meltzer, B. and Michie, D. "Linear skeletons from square cupboards", *in Machine Intelligence*, **IV**, Elsevier, New York, pp. 403-420 (1969).

6. Batuwita, K.B.M.R. and Bandara, G.E.M.D.C. "New segmentation algorithm for individual offline hand-

written character segmentation", *Proc. of 2nd Int. Conf. on Fuzzy Systems and Knowledge Discovery*, Changsha, China, pp. 215-229 (2005).

7. Kim, G. and Kim, S. "Feature selection using genetic algorithms for handwritten character recognition", *Proc. of the Sixth ACM SIGKDD* (2002).
8. Ranawana, R., Palade, V. and Bandara, G.E.M.D.C. "Automatic fuzzy rule base generation for on-line handwritten alphanumeric character recognition", *International Journal of Knowledge-Based and Intelligent Engineering Systems*, **9**(4), pp. 327-339 (2005).
9. Goltsev, A. and Rachkovskij, D. "Combination of the assembly neural network with a perceptron for recognition of handwritten digits arranged in numeral strings", *Pattern Recognition*, **3**, pp. 315-322 (2005).
10. Malaviya, A. and Peters, L. "Handwriting recognition with fuzzy linguistic rules", *Proc. of Third European Congress on Intelligent Techniques and Soft Computing*, Aachen, Germany, pp. 1430-1434 (1995).
11. Chiang, J.H. and Gader, P.D. "Hybrid fuzzy-neural systems in handwritten word recognition", *Proc. of IEEE Trans. Fuzzy Systems*, **5**(4), pp. 497-510 (1997).
12. Malaviya, A. and Peters, L. "Extracting meaningful handwriting features with fuzzy aggregation method", *Proc. of ICDAR'95*, Montreal, Canada, pp. 841-844 (1995).
13. Ekelmann, J. and Butka, B. "Kinect controlled electromechanical skeleton", *Proc. of IEEE, Southeastcon*, pp. 1-5 (2012).
14. Arici, T. "Introduction to programming with Kinect: Understanding hand/arm/head motion and spoken commands", *Signal Processing and Communications Applications Conference (SIU)*, p. 1 (2012).
15. Webb, J. and Ashley, J., *Beginning Kinect Programming with the Microsoft Kinect SDK*, Apress 1st edition (2012).
16. Microsoft, "Kinect for windows SDK", [Online]. http://msdn.microsoft.com/en-us/library/hh855347.aspx.
17. Microsoft, "Kinect for windows development, dpe", (Online). http://msdn.microsoft.com/zh-tw/ hh367958.aspx.

Biographies

Jia-Shing Sheu received MS and PhD degrees from the Department of Electrical Engineering at the National Cheng Kung University, Tainan, Taiwan, in 1995 and 2002, respectively. He is currently Associate Professor in the Department of Computer Science at the National Taipei University of Education, Taipei, Taiwan. His research interests include pattern recognition and image processing, focusing, especially, on real time face recognition, and embedded systems.

Kuo-Chuan Wang received BS and MS degrees from the Department of Computer Science at the National Taipei University of Education, Taipei, Taiwan, in 2012 and 2013, respectively. He has been currently employed as a R&D engineer at the Aurotek Corporation Company, Taipei, Taiwan. His main interests are in the field of motion control cards.