Hand Gesture Recognition Using Kinect

Yi Li
Computer Engineering & Computer Science
University of Louisville
Louisville, KY 40214, USA
yi.li@louisville.edu

***Abstract*—Hand gesture recognition (HGR) is an important research topic because some situations require silent communication with sign languages. Computational HGR systems assist silent communication, and help people learn a sign language. In this article, a novel method for contact-less HGR using Microsoft Kinect for Xbox is described, and a real-time HGR system is implemented. The system is able to detect the presence of gestures, to identify fingers, and to recognize the meanings of nine gestures in a pre-defined Popular Gesture scenario. The accuracy of the HGR system is from 84% to 99% with single-hand gestures, and from 90% to 100% if both hands perform the same gesture at the same time. Because the depth sensor of Kinect is an infrared camera, the lighting conditions, signers' skin colors and clothing, and background have little impact on the performance of this system. The accuracy and the robustness make this system a versatile component that can be integrated in a variety of applications in daily life.**

***Keywords*—Human-computer interaction, hand gesture recognition, Kinect, finger identification**

I. Introduction

There is always a need to communicate using sign languages, such as chatting with speech and hearing challenged people. Additionally, there are situations when silent communication is preferred; for example, during an operation, a surgeon may gesture to the nurse for assistance. It is hard for most people who are not familiar with a sign language to communicate without an interpreter. Thus, software that transcribes symbols in sign languages into plain text can help with real-time communication, and it may also provide interactive training for people to learn a sign language. Gesture recognition has become an important research field with the current focus on interactive emotion recognition and hand gesture recognition.

Microsoft Kinect provides an inexpensive and easy way for real-time user interaction. The software driver released by Microsoft called Kinect Software Development Kit (SDK) with Application Programming Interfaces (API) give access to raw sensor data streams as well as skeletal tracking. However, there is no hand specific data available for gesture recognition, although it does include information of the joints between hands and arms. Little work has been done for Kinect to detect the details at the level of individual fingers.

In this article, an HGR system using Microsoft Kinect for Xbox has been implemented. The system detects hand gestures made by the user, compares them with the signs in the pre-defined Popular Gesture set, and displays the matched meaning and the corresponding picture on the screen. The system represents a new approach to human-computer interaction that uses nothing but bare hands as the input media.

Depth data is generated and converted from the raw image data of Kinect sensor by an open-source framework called OpenNI (Natural Interaction), with an open-source driver called SensorKinect by PrimeSense, which makes Kinect for Xbox compatible with Microsoft Windows 7. Using the newest version of graphics driver, the system is developed in C# of Microsoft Visual Studio 2010. The accuracy of the HGR system is from 84% to 99% with a single-hand gesture, and from 90% to 100% if both hands perform the same gesture at the same time. The identification of all individual fingers has never been done in the literature with the Kinect platform to the best of my knowledge. Because the depth sensor of Kinect is an infrared camera, the lighting conditions, signers' skin colors and clothing, and background have little impact on the performance of this system. The accuracy and the robustness make this system a versatile component that can be integrated in a variety of applications in daily life.

II. Literature Review

Generally, a sign language consists of three parts: finger-spelling, word level sign vocabulary, and non-manual features. Therefore, HGR forms the basis in translating sign languages. Recent technology for gesture recognition is to incorporate the information of object distances, or depths, normally captured by a 3D camera, or a set of cameras that produce 3D images. In [1], Van den Bergh *et al.* used a Time-of-Flight camera to enhance recognition. Their system could recognize six gestures: open hand, fist, pointing up, L-shape, pointing at the camera, and thumb-up. The RGB-based recognition showed an accuracy of 99.54%, and the depth-based recognition showed an accuracy of 99.07%, while the combination of the two methods showed 99.54%. This suggests that depth-based recognition may be good enough to form the basis of an HGR system especially in hand segmentation, as the combination of two methods did not provide significant improvement to the accuracy.

Numerous researches have been done with Kinect since its launch in 2010. However, only a few systems were developed for HGR, and only a few gestures were recognized. Yang *et al.* [2] proposed a gesture recognition system using depth information provided by Kinect, and implemented in a media player application. The hand trajectory was examined by a 3D feature vector, and the gesture was recognized by a hidden

978-1-4673-2008-5/12/$31.00 ©2012 IEEE

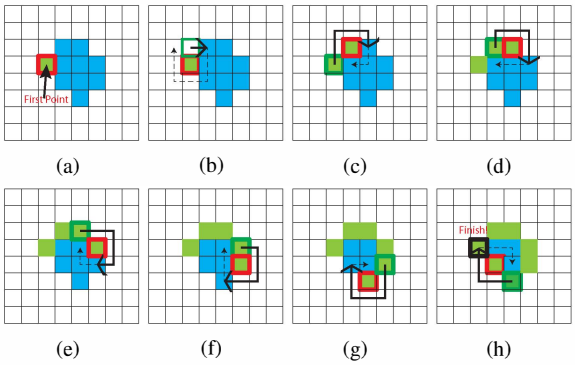


Fig. 1. Contour Tracing. To trace the contour of the blue pixels: Let green pixels represent detected contour pixels. The pixel with red boundary denotes the current contour pixel; the pixel with green boundary denotes the starting pixel of neighborhood checking, namely the last contour pixel except for the starting point. The black arrow shows the clockwise path of neighborhood checking; the dashed arrow shows the path that the algorithm does not need to finish, because the next contour pixel is already detected.

Markov Model (HMM). This system demonstrated the applicability of using Kinect for gesture recognition in a contact-less user interface (UI). The system was not able to find fingertips; thus it was limited to recognize only motion gestures like wave, and move up/down, left/right, and forward/backward. A method to track fingertips and the centers of palms using Kinect was presented by Raheja *et al.* [3]. When fingers were extended, the accuracy of detecting fingertips was nearly 100%, and that of palm centers was around 90%. However this method did not attempt gesture recognition. He *et al.* [4] proposed an approach using depth data provided by Kinect to detect fingertips. After fingertips were found, the mouse clicking motion was recognized and tested on the popular game Angry Bird; that is, it recognized only one gesture.

III. Methods

The system of the present work consists of three main components: Hand Detection, Finger Identification, and Gesture Recognition. This system is built on the Candescent NUI [5] project, which is freely available online. OpenNI framework is used to extract depth data from the Kinect sensor.

A. Hand Detection

The first step is to separate the hands from the background. Depth thresholds are set manually in advance to specify the depth range where gestures must appear to be recognized. The detection range, $zRange$, is between 0.5m and 0.8m; up to two hands in this range can be detected. Pixels with depths outside $zRange$ are ignored in the rest of the gesture detection process. Henceforth, a hand pixel may also be referred to as a point. These hand pixels are projected to a 2D space for subsequent analysis. Distance between two pixels $p_1(x_1, y_1)$ and $p_2(x_2, y_2)$ is defined as:

$$D(p_1, p_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}. \quad (1)$$

The K-means clustering algorithm is applied to partition all the pixels to two groups. K-means clustering is a method to partition n observations into k clusters, $C_1, C_2, \ldots, C_k$; each observation is identified with the nearest mean, $\mu_i(x, y)$, which is calculated as the mean of of points in C_i. K-means clustering minimizes the within-cluster sum of squares:

$$\arg\min_{C} \sum_{i=1}^{k} \sum_{p_j(x,y) \in C_i} \|p_j(x, y) - \mu_i(x, y)\|^2 . \quad (2)$$

In the present system, the value of k is 2. The system runs in real-time, and K-means clustering is continuously applied whenever there is change in the input data source. At the beginning of every iteration, each empty cluster is initialized with a random point located within the $zRange$ as the mean. After K-means converges, the points belonging to each hand are clustered. If the distance between the two centroids of hands is less than a pre-defined value, the two clusters are merged into one.

Next, the convex hulls of hands are computed. The Graham Scan algorithm [6] is used to compute the convex hull of the detected hand clusters. Hand contours are detected by a modified Moore-Neighbor Tracing algorithm. After the K-means clustering, a group of hand points are found and stored. Figure 1 shows an example of implementing the algorithm to detect contour.

B. Finger Identification

Let $\mathbf{N}(a)$ be the eight-point neighborhood of a pixel a. Let p denote the current contour pixel. Let q denote the starting pixel of current neighborhood checking. Let $\mathbf{C}$ denote the set of detected contour points, which is initialized to be the empty set.

Contour Tracing Algorithm:

1) From top to bottom, and left to right, scan all pixels on the screen until a pixel s being a hand point is found, which is set as the starting point.
2) Set the current contour pixel p to be s. Set the starting pixel of neighborhood checking q to be the point to the immediately north of s.
3) Insert p into $\mathbf{C}$, and calculate the neighborhood $\mathbf{N}(p)$.

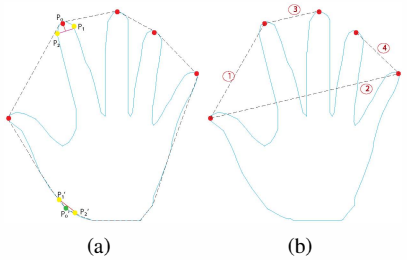
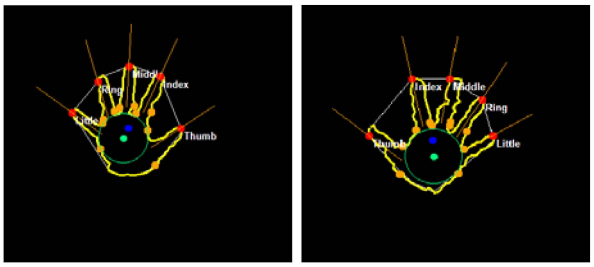


Fig. 2. 2(a) Fingertip Detection. The red dots are real fingertips, and the green dot is not a fingertip. The yellow dots are used to check for three-point alignment. The distance between p_0 and the line made by p_1 and p_2 is apparently larger than that of p'_0 and the line made by p'_1 and p'_2. Thus a threshold distance could be used to differentiate fingertip points and non-fingertip points.
2(b) Finger Identification.



(a) (b)

Fig. 3. Finger Identification Results.

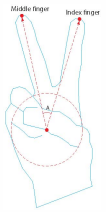


Fig. 4. Recognition Example. First the system determines that the number of fingers in this gesture is two. Then the finger names of "index finger" and "middle finger" are found. The classifier in the third layer calculates the angle of two finger vectors are within a range of 30 degree to 60 degree. Thus the gesture is recognized as "Victory" in the Popular Gesture scenario, while in the Numbers scenario it is recognized as the number "Two" through a similar process.

4) Start from q, go around the neighborhood $\mathbf{N}(p)$ in the clockwise direction until a hand pixel r is found.
5) Set q to be p, and p to be new contour pixel r. Then repeat Step 3 until the starting point s is reached again, or the number of detected points exceeds a maximum value.

After the hand contours are detected, the centers of the palms are calculated as the centers of the inscribing circles of the hand contours. The positions of palm centers are used to calculate the directions of fingers, which are described in the next section. The centers of the inscribing circles are much more stable than the centroids of hand clusters, because the latter vary a lot when opening/closing hands or bending fingers.

Fingertips are detected by checking the three-point alignment relationship. Candidate points are the points that are on both the convex hull and the hand contour. The method is shown in Figure 2(a).

Three-point Alignment Algorithm:

1) Let $\mathbf{C}$ be the set of all candidate points that are on both the convex hull and the hand contour.
2) For each point p_0 in $\mathbf{C}$, take the two points p_1 and p_2 in the two opposite directions along the contour from p_0.
3) Find the midpoint of p_1 and p_2 and calculate the distance between this midpoint and p_0. If the distance is larger than a specific value, the three points are not aligned, i.e., far from being collinear. Then this p_0 is identified as a fingertip. Otherwise go back to Step 2 and check the next candidate point.

After fingertips are detected, their direction vectors are easy to calculate by subtracting the position of the palm center $P_c(x_c, y_c)$ (which is calculated in Section III-A): $\vec{V} = (x - x_c, y - y_c)$. The vectors are pointing from the palm center to fingers.

Finger names are determined according to their relative distances, which require the user to open palm and to extend fingers. Four procedures to identify the names of all fingers are shown in Figure 2(b). The identification of all individual fingers has never been done in the literature with the Kinect platform to the best of my knowledge.

Finger Identification Algorithm:

1) The first and easiest step is to identify the thumb and the index finger, since the distance between them is the largest among all neighboring fingers.
2) The little finger is identified as the farthest finger away from the thumb, meanwhile the middle finger is identified as the closest one to the index finger.
3) The remaining one is the ring finger.

The process of detecting hands and identifying fingers are performed every time when the input data source changes. If the same object still exists in the next frame with some transformation compared to the previous frame, all properties of this object is mapped to the old frame. The results of finger identification are shown in Figure 3.

C. Gesture Recognition

Once the fingers are identified with their names, we are ready for gesture recognition. Gestures are passed through three layers of classifiers: finger counting, finger name collecting, and vector matching. The angle A between two vectors $\vec{V_1}(x_1, y_1)$ and $\vec{V_2}(x_2, y_2)$ is calculated as:

$$A = \arccos \frac{\vec{V_1} \cdot \vec{V_2}}{\left\|\vec{V_1}\right\| \left\|\vec{V_2}\right\|}. \tag{3}$$

Three Layers of Classifiers:

1) Finger counting classifier: Gestures are first classified by the number of extended fingers, and then sent to the corresponding second layer of classifiers.
2) Finger name collecting: Gestures are further classified by the names of the extended fingers. If the combination of the extended fingers in one gesture is unique among all gestures, the recognition process terminates, and the meaning of the gesture is displayed; otherwise the gesture is sent to the corresponding third layer of classifiers.
3) Vector matching: The direction vectors of all extended fingers are measured, and all pairwise angles between extended fingers are calculated. The meaning of the gesture is now classified according to these angles. Then the meaning of the gesture is assigned and displayed on screen.

An example of recognizing "Victory" is illustrated in Figure 4.

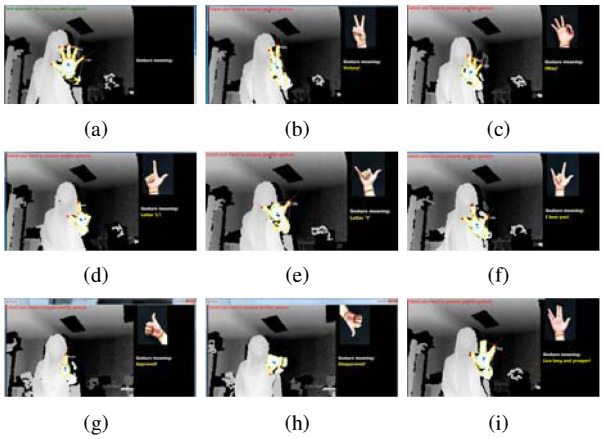


Fig. 5. Recognition Results. 5(a) is the "Start Gesture" for the purpose of calibration; 5(b) is "Victory"; 5(c) is "Okay"; 5(d) is the letter "L"; 5(e) is the letter "Y"; 5(f) is "I love you"; 5(g) is "Thumb-up"; 5(h) is "Thumb-down" ; 5(i) is the *Star Trek* gesture, which means "Live long and prosper".

TABLE I
HGR ACCURACY

Gestures	Start	Thumb-up	Thumb-down	L	Y
Accuracy(%)	99	89.5	88.25	91	90.5
Gestures	Okay	Victory	Star Trek	I ♡ U	
Accuracy(%)	84.75	89.25	84	85.25	

The Popular Gesture scenario consists of nine gestures: "Start Gesture" (open hand), "Thumb-up", "Thumb-down", "Victory", "Okay", "I love you", the letter "L", the letter "Y", and "Live long and prosper". The HGR results of Popular Gesture are shown in Figure 5.

D. Results

To test this system, four people are asked to perform each of the nine gestures for 100 times: 50 times with the left hand and 50 times with the right hand. The average accuracy of each gesture is calculated and shown in Table I. For "Start Gesture", the accuracy is nearly 100%, and the names of the fingers are perfectly identified. The best case is the letter "L" with an accuracy of 91%, while the worst case is "Live long and prosper" with an accuracy of 84%. Due to the difficulty of the *Star Trek* gesture, only two of the four people are able to perform "Live long and prosper". Furthermore, if the signer uses two hands to do the same gestures at the same time, the accuracy is significantly increased to more than 90% for all nine gestures.

IV. CONCLUSION AND FUTURE WORK

In this article, an HGR system based on Microsoft Kinect for Xbox is introduced. The system is motivated by the importance of real-time communication under specific situations, such as chatting with speech and hearing challenged people. It is capable of working in the dark, and it can be easily transplanted to other applications.

In the developed system, hands are distinguished from the background by the depth information; specifically, the detection range is between 0.5m to 0.8m. The K-means clustering algorithm is used to obtain two clusters of hand pixels; if the distance between the two cluster means is within a threshold, the two clusters are merged, and only one hand is detected. Then the Graham Scan algorithm is used to determine the convex hulls of hands; a contour tracing algorithm is used to detect hand contours. Afterwards, candidate fingertips are collected by calculating the common pixels that are on both the convex hull and the hand contour, from which fingertips are detected by applying the three-point alignment algorithm. Finger names are determined according to their relative distances, and a direction vector is assigned to each finger. Gestures are recognized by passing them through a set of classifiers that contains three layers: finger counting classifier, finger name collecting, and vector matching. A scenario is constructed to be recognized: Popular Gesture, which consists of nine popular gestures. The recognition has accuracy of at least 84% for one hand, while it increases to more than 90% when both hands are doing the same gestures.

The present system provides a new approach for HGR at the level of individual fingers. It opens possibilities for HGR systems to be applied in many practical ways in real life.

Some directions for future work are to further improve the accuracy of recognition, to add more scenarios such as finger-spelling of the alphabet and numerals, and sports referee gestures. Moreover, by incorporating hidden Markov models, the system may allow the recognition of continuous gestures that form words or sentences, and the narrowly defined scenarios may be expanded to discourse contexts. With the recently launched Microsoft Kinect for Windows 7, the sensors are more powerful than the Kinect for Xbox, which makes it more suitable for HGR. It is expected that more applications with excellent recognition results will be developed.

ACKNOWLEDGMENT

The author would like to thank Dr. Ming Ouyang, her adviser, for providing the devices and guiding her all the way throughout this research, which is partially supported by US FAA Grant 11-G-010 to Ming Ouyang.

REFERENCES

[1] M. Van den Bergh and L. Van Gool, "Combining RGB and ToF cameras for real-time 3D hand gesture interaction," in *Applications of Computer Vision (WACV), 2011 IEEE Workshop on*, 2011, pp. 66 –72.

[2] C. Yang, Y. Jang, J. Beh, D. Han, and H. Ko, "Gesture recognition using depth-based hand tracking for contactless controller application," in *Consumer Electronics (ICCE), 2012 IEEE International Conference on*, 2012, pp. 297 –298.

[3] J. L. Raheja, A. Chaudhary, and K. Singal, "Tracking of fingertips and centers of palm using KINECT," *2011 Third International Conference on Computational Intelligence Modelling Simulation*, pp. 248–252, 2011.

[4] G.-F. He, S.-K. Kang, W.-C. Song, and S.-T. Jung, "Real-time gesture recognition using 3D depth camera," in *Software Engineering and Service Science (ICSESS), 2011 IEEE 2nd International Conference on*, 2011, pp. 187 –190.

[5] S. Stegmueller, "Hand and finger tracking with Kinect depth data," 2011, http://candescentnui.codeplex.com.

[6] R. L. Graham, "An efficient algorithm for determining the convex hull of a finite planar set," *Information Processing Letters*, vol. 1, no. 4, pp. 132–133, 1972.