2014 International Joint Conference on Neural Networks (IJCNN)
July 6-11, 2014, Beijing, China

Real-Time Hand Gesture Recognition with Kinect for Playing Racing Video Games

Yanmin Zhu [1] and Bo Yuan [2]
Intelligent Computing Lab, Division of Informatics
Graduate School at Shenzhen, Tsinghua University
Shenzhen 518055, P. R. China
[1] zhuym111@gmail.com; [2] yuanb@sz.tsinghua.edu.cn

***Abstract*—This paper presents a Kinect based hand gesture recognition system that can effectively recognize both one-hand and two-hand gestures. It is robust against the disturbance of complex background and objects such as the faces and hands of other people by exploiting the depth information and carefully choosing the region of interest (ROI) in the process of tracking. The recognition module is implemented using template matching and other light weight techniques to reduce the computational complexity. In the experiments, this system is tested on real world tasks from controlling the slide show in *PowerPoint* to playing the highly intense racing video game *Need for Speed*. The practical performance confirms that our system is both effective in terms of robustness and versatility and efficient for various real-time applications.**

Keywords—hand gesture recognition; Kinect; video games

I. INTRODUCTION

Hand gesture recognition takes a significant position in the field of human-computer interaction (HCI). Both academic and industrial communities have shown growing interests in this area [1, 2]. With the development of multimedia and machine intelligence techniques, hand gesture recognition has been applied to a wide range of interactive systems [3], such as video games [4], virtual reality [5], sign language recognition [6], surgical system [7] and robot control [8].

Compared with glove-based hand gesture recognition that needs extra devices to be worn on hands [9, 10], vision based hand gesture recognition is more appealing to users due to its natural and intuitive operation. However, vision based hand gesture recognition is confronted with many tough challenges such as cluttered background and illumination change. For example, complex background with regions similar to skin color can interfere with the segmentation of hand while the change in illumination can alter the appearance of hand due to the influence of shadows. Furthermore, the articulated structure of hand may result in special challenges such as deformation and self-occlusion. After all, many applications have a high requirement on the efficiency and it is difficult to find a solution that can robustly recognize hand gestures in uncontrolled circumstances in a real-time manner. The reason is that most traditional detection, tracking and recognition techniques with high precision are computationally intensive and cannot satisfy this demanding constraint.

Fortunately, the introduction of RGB-D sensors such as Kinect and ASUS Xtion in recent years has significantly simplified the process of vision based object recognition, especially the segmentation phase. These sensors provide remarkable improvement when dealing with complex backgrounds and the depth data can also provide more information about the object and help improve the performance of recognition. Due to their unique benefits, RGB-D sensors are becoming more and more popular in computer vision and pattern recognition.

Among all existing commodity RGB-D sensors, Kinect by Microsoft is the most popular one. Since the first release of Kinect in late 2010 in combination with Xbox 360 (the Windows version was released in 2012), a large number of interesting applications have been developed. For hand pose recognition, Ren et al. built a Kinect based hand gesture recognition system [11], which can recognize hand poses and was used for performing arithmetic computation and playing the rock-paper-scissors game [12]. Their system required the hand to be at the forefront and the user to wear a black belt on the gesturing hand's wrist for hand segmentation purpose. Its recognition time was around 0.5 second as reported. Using both the color and depth information that Kinect provides, Oikonomidis et al. [13] built a 3D model of hand to recover the full articulation structure and used the particle swarm optimization algorithm to find the parameters of hand model. The system's frame rate was 15 FPS when implemented using parallel computing on GPU. The authors later applied this approach to recognizing two strongly interacting hands [14] and the frame rate dropped to 4 FPS. Keskin et al. [15] extracted the skeleton structure of hand from depth images using random decision forest and used this approach to recognize ASL digitals and achieved a recognition rate of 99.9%.

Most hand pose recognition algorithms cannot meet the real-time requirement as detailed information of the hand needs to be extracted, resulting in very high computational complexity. The hand also has numerous appearances due to its articulated structure and different viewpoints and large training datasets are necessary.

978-1-4799-1484-5/14/$31.00 ©2014 IEEE

There are also many studies on recognizing dynamic hand gesture using Kinect. Tang [16] proposed an approach that can recognize the gestures of "grasp" and "drop" with over 90% accuracy. Liang [17] conducted gesture recognition using depth images and compared the proposed method with traditional methods using 2D images and showed that depth information improved performance notably. There are also some attempts by using Kinect skeleton tracking SDK for hand tracking. However, it takes a few seconds for Kinect to find the skeleton and the precision is not reliable enough.

This paper presents a hand gesture recognition system, which can track the movements of both hands and recognize both dynamic gestures and poses of hands. With the function of one-hand gesture recognition, users can have full control of the slide show in *PowerPoint* without using keyboard and mouse. For two-hand gesture recognition, users can smoothly control the popular racing video game *Need for Speed*, which is very demanding in response time.

The prominent features that are different from existing hand gesture recognition applications are: i) our system can deal with strongly cluttered background such as the presence of multiple people and there is no need for users to wear marking devices; ii) our approach has comparatively low computational complexity with fast response speed and its practical performance is good enough to be used with high end video games.

Section II presents Kinect and the experiment environment. The main process of our approach is detailed in Section III. The two applications and experiment results are shown in Section IV. Section V concludes this paper and indicates the directions for future work.

II. KINECT AND EXPERIMENT ENVIRONMENT

A. Kinect

As Fig. 1 shows, Kinect consists of an RGB camera, an IR emitter, an IR depth sensor, a microphone array and a tilt. The RGB camera can capture three-channel data in a 1280 × 960 resolution at 12 FPS or a 640 × 480 resolution at 30 FPS. The IR emitter emits infrared light beams and the IR depth sensor reads the IR beams reflected back to the sensor. The reflected beams are converted into depth information measuring the distance between an object and the sensor. The depth map's resolution can be 320 × 240 or 640 × 480, both at 30 FPS.

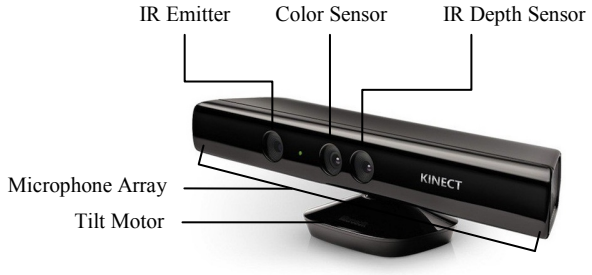

Fig. 1. The components of Kinect

The angular field of view is 57 degrees horizontally and 43 degrees vertically and Kinect can be tilted up or down by 27 degrees. Under the normal model, the valid operating distance of Kinect is 0.8m~4m. Under the near model, the valid operating distance is 0.4m~3m.

B. Experiment Environment

In our experiments, we used both the RGB video stream and the depth map produced by the Kinect sensor with a resolution of 640 × 480 at 30 FPS. The computer system was a standard PC with Intel Core i5-2320 CPU at 3.0 GHz and 8G RAM. The OS was 64-bit Windows 7 and our system was developed using Microsoft Visual C++ and OpenCV 2.4.5. This specification represents a typical computing environment nowadays. For the applications, we used *Microsoft Office PowerPoint 2010* and *Need for Speed: Most Wanted* (2012).

III. HAND GESTURE RECOGNITION TECHNOLOGIES

Fig. 2 shows the overall procedure of the proposed hand gesture recognition system. One-hand gesture recognition starts from locating the hand using color image and depth data and then tracks its movement to get its trajectory, which is used to recognize the gesture. For two-hand gesture recognition, we first locate and track one hand as in one-hand gesture recognition. Then using the shape and location information of the identified hand, we locate the other hand and perform two-hand gesture recognition.

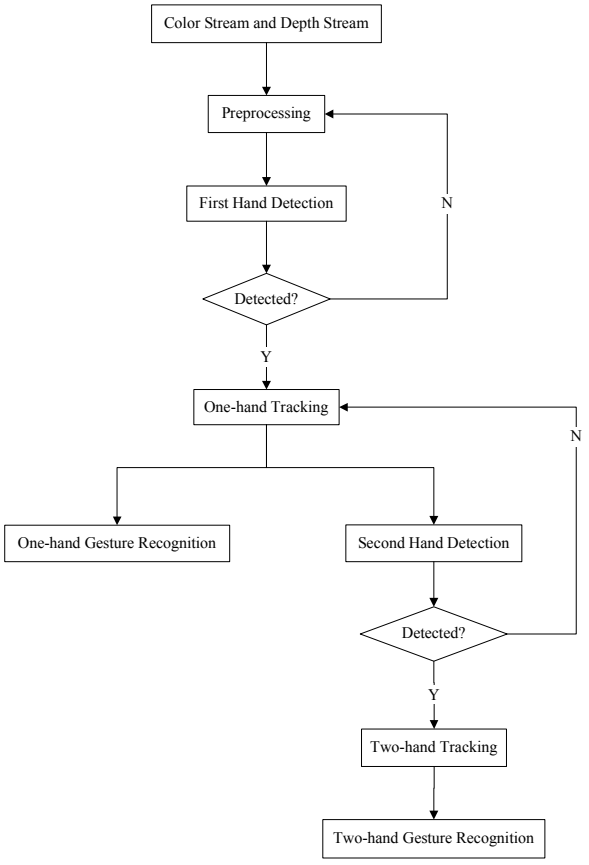

Fig. 2. Flowchart of hand gesture recognition: one-hand gesture recognition (left branch) and two-hand gesture recognition (right branch)

A. Preprocessing

Poor lighting condition can have a severe influence on skin color detection in RGB images. Although Kinect is able to perform automatic white balance, the result is not satisfactory. To gain more robustness against illumination uncertainty, the system conducts color balance before detecting skin color.

There are many color balancing algorithms: Scale by Max, Gray World, White World (Perfect Reflector) etc. As Kinect can only work normally in indoor environment, the most likely situation confronted is insufficient illumination and colored lighting. We compared empirically several color balancing approaches and found that Gray World had relative better performance. The Gray World method assumes that the red, green and blue channels of a given image should have the same mean value. So, we adjusted the mean value of each channel to be the same as that of the green channel since human eyes are most sensitive to this region of the spectrum.

$$r_i = \frac{avg(g)}{avg(r)} r_i,\ g_i = g_i,\ b_i = \frac{avg(g)}{avg(b)} b_i \qquad (1)$$

In (1), r_i, g_i, b_i are the red, green and blue components of pixel i in the image respectively while $avg(r)$, $avg(g)$ and $avg(b)$ are the mean values of red, green and blue channels. The effect of color balancing is shown in Fig. 3 where color balancing can help detect skin more precisely and eliminate the disturbance of some skin-like color.

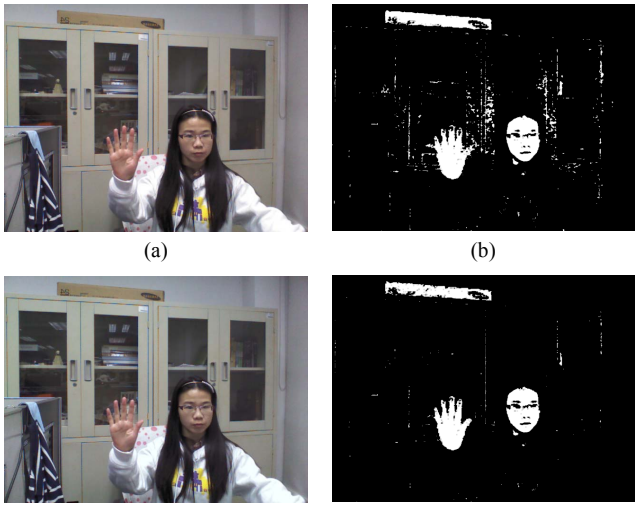

Fig. 3. The effect of color balancing: original image (a); skin detection result of original image (b); balanced image (c); skin detection result of balanced image (d)

B. First Hand Detection

Since it is natural to put hands forward when one wants to perform gestures, we define that when a user puts forward his/her hand and the distance between hand and body is over a threshold, it is a sign of starting to make gestures. First, the system applies an elliptical boundary model [18] to the RGB image (Fig. 4(a)) to extract possible skin parts (Fig. 4(b)). A threshold *minArea* is used to remove regions that are too small to be hand regions and skin-like objects too far from the sensor are also eliminated (Fig. 4(c)). The result is shown in Fig. 4(d).

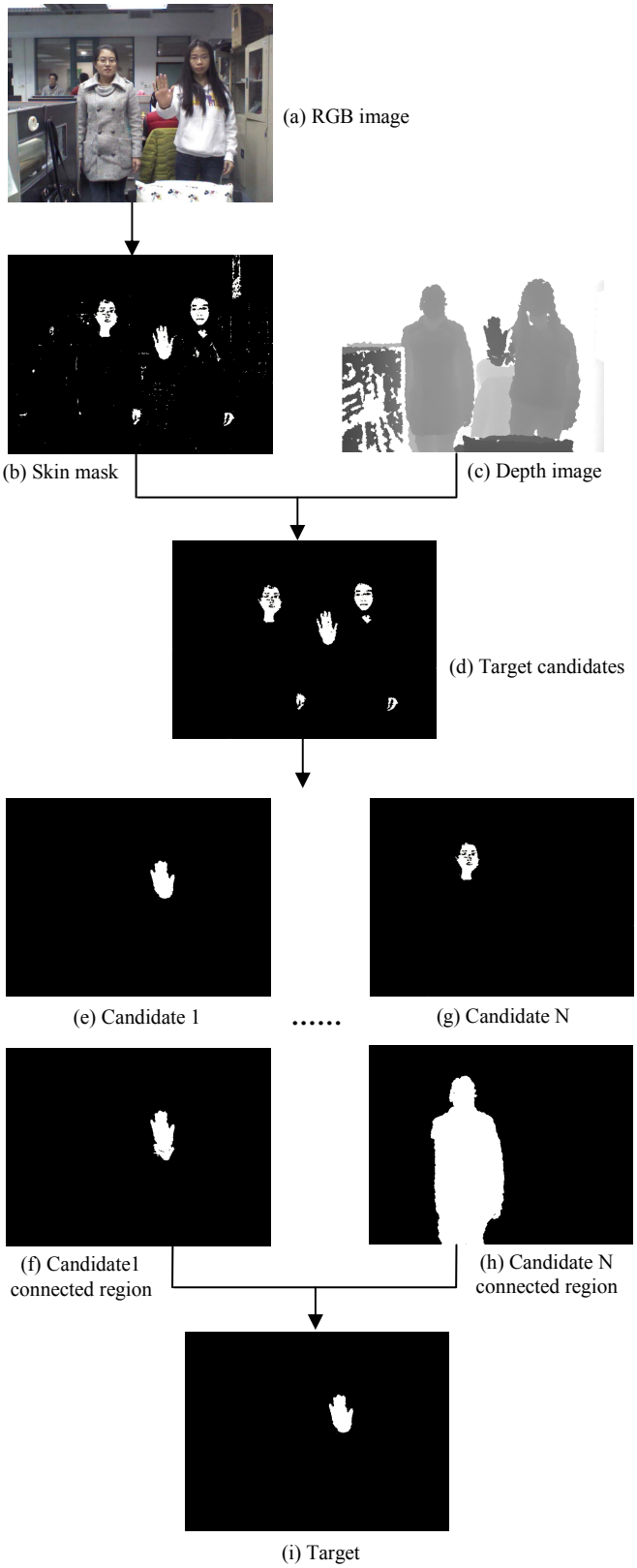

Fig. 4. The process of first hand detection

Next, the system determines whether one of the skin blocks contains the target hand following three steps:

1) Locate target candidates. Suppose the minimum depth of the i^{th} skin block is $minDepth_i$. A depth filter is applied to extract the sub-region with depth in the range of [$minDepth_i$, $minDepth_i$ + *thresh1*] (See Fig. 4(e) and Fig. 4 (g) for some examples). This operation can be used to segment hand from arm, which may have similar color information but different depth information.

2) Find the connected region of candidates. Apply another depth filter with range [$minDepth_i$, $minDepth_i$ + *thresh2*] on the depth image to find the i^{th} connected region (regardless of the color information) containing the i^{th} target candidate. This operation can be used to create a connected region consisting of hand and arm (Fig. 4(f)) or face and body (Fig. 4(h)).

3) Determine the target. Compare the area of each candidate with the area of its corresponding connected region. If the ratio is larger than a threshold, we assume that the candidate is the outstretched hand (target). Otherwise, the candidate is eliminated. The reason is that, for example, the face usually only accounts for a small portion of the face-body region compared to the ratio between the outstretched hand and the hand-arm region. As a result, interfering objects such as face and the non-gesturing hand can be effectively removed, as shown in Fig. 4(i).

C. Second Hand Detection

It is very natural for a user to put both hands forward with the same pose before he/she starts to perform two-hand gestures. Consequently, for two-hand gesture recognition, we assume that this is the initial gesture.

As the system has detected the first hand in the previous part (Section III-B), it is assumed that the second hand is on the same depth plane and has similar pose as the first hand. So the depth and shape information can be used to detect the second hand and start tracking both hands.

D. Tracking

Recently, tracking-by-detection has become increasingly popular for object tracking [19, 20]. The idea is to detect the object of interest in each frame and associate the object locations in successive frames to generate the trajectory.

In the process of tracking, according to the hand location and size in the previous frame, a region of interest (ROI) is set for the current frame and the target is only searched in the ROI. This step narrows down the range of search significantly and thus reduces the amount of computation and eliminates the disturbance of distractors outside of ROI. In our system, the ROI is defined as:

$$\begin{aligned} x &: [x_0 - s, x_0 + s], \\ y &: [y_0 - s, y_0 + s], \\ z &: [z_0 - thresh, z_0 + thresh]. \end{aligned} \tag{2}$$

In (2), [x_0, y_0, z_0] is the hand location in the previous frame and s is set to be half of the circumstance of hand's bounding box while *thresh* is a fixed threshold based on the longest distance that hand can move in 1/30 second.

Only the depth images are used for tracking for the sake of efficiency, as the coordinates of depth image and color image are not matched precisely and converting the coordinates of an entire image is a time consuming task.

A practical issue is the coarseness of the depth image. Due to the method of depth information acquisition, Kinect cannot precisely measure the depth of objects with smooth surface such as glass. Furthermore, depth data can be inaccurate at the edge of objects (Fig. 5). These bugs may occur frequently and interrupt the tracking process. Setting ROI can eliminate the influence of some of these flaws. However, those in the vicinity of the target hand can cause real trouble as the detection is based on the nearest object in ROI. In practice, if the hand is missing and the depth of the closest object is significantly different from the hand in the last frame, this object will be removed and a second detection is conducted to increase the robustness of tracking.

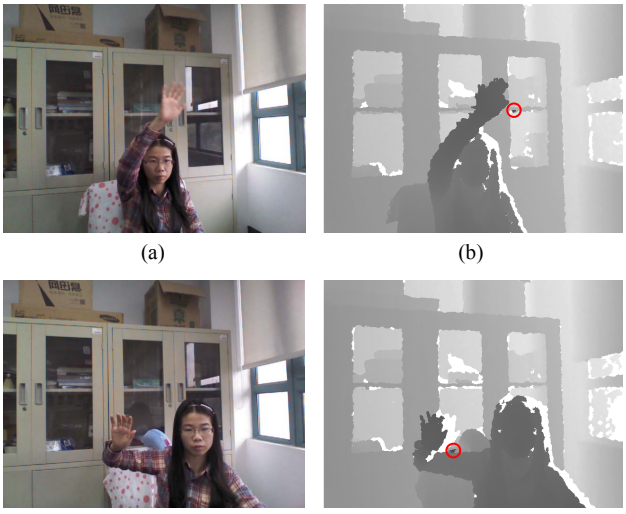

Fig. 5. Examples of the flaws in the depth images produced by Kinect: color images (a) and (c); corresponding depth images with flaws marked by red circles (b) and (d)

E. Palm and Fist Classification

To distinguish palm and fist, we collected a dataset with 500 images of palm and fist and manually segmented them from the background, as shown in Fig. 6

Fig. 6. Segmented samples of palm and fist

Since the system imposes no strict restriction on the user's distance to Kinect or the orientation of the hands, the features of hand need to be invariant to scaling, rotation and translation. Hu's moment invariants [21] are extensively used global features in pattern recognition. These seven values of Hu's moment invariants are computed by normalizing central moments through order three, which are invariant to object scaling, rotation and translation.

Compared with other popular classifiers such as neural networks, decision tree model and naïve Bayes classifier, SVM has better performance when dealing with small datasets, especially in nonlinear and high dimensional situations. Thus, we chose SVM to identify palm and fist.

Due to the limitation of Kinect hardware, the depth data at the edge of object may be quite rough. To better exploit geometrical features, the depth image is smoothed by blurring and binarization using a threshold.

F. Gesture Recognition

To recognize the meaning of gestures based on the trajectory of hands, a challenge is called gesture spotting, which is to separate unintended gestures from meaningful gestures. Traditional dynamic hand gesture recognition approaches include hidden Markov model (HMM), dynamic time warping (DTW) and finite-state machine (FSM).

Since the focus of our system is on efficiency, simple methods are preferred. For directional one-hand gestures such as rightward, leftward, upward, downward and reverse, template matching is employed. First, we extract part of the trajectory with a certain length, for example 0.5 second, which measures the last 15 frames of the complete trajectory. Second, a few key points are sampled from the trajectory. Our system samples three key points: the starting point, the end point and the middle point. Third, their positions relative to the starting point are calculated. Finally, these key points are matched with those on the individual template. In Fig. 7, S is the trajectory of hand; *S(i)* is the i^{th} key points of S; T is the template; *T(i)* is the i^{th} key points on T.

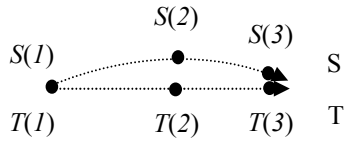

Fig. 7. Matching between hand trajectory and the template

The similarity is measured by the mean square error (MSE) as follows (*N* is the number of sampled key points):

$$MSE = \sum_{i=1}^{N} \left[S(i) - T(i) \right]^2 \tag{3}$$

For circular gestures, there are some differences from the previous approach. First, as drawing a circle takes more time than directional gestures, the last 1 second part of the trajectory (30 frames) is extracted. Second, try to find a closed curve in the trajectory by searching the 30 frames to see whether there is a point coincides with the end point of the trajectory. Third, identify four key points of the closed curve with equal time intervals. Fourth, determine if the closed curve forms a circle by imposing restrictions on the distance between these key points. Finally, according to the sequential relationship of these key points, determine whether the circle is in clockwise or anticlockwise.

After the recognition of a gesture, the system waits for one second before recognizing another gesture. This is to leave a time interval for the user to prepare for the next gesture, which can reduce the recognition error caused by hand reset.

As to two-hand gesture recognition, our system currently focuses on the recognition of poses such as palm and fist including the change of poses and the relative position of the two hands to realize various functions.

IV. Applications

A. Controlling PowerPoint

We defined eight gestures for controlling *PowerPoint*, corresponding to six most common functions: open file, slide show, next page, previous page, quit slide show and close file, as shown in Table I. We asked five volunteers to perform these gestures to control *PowerPoint* after giving a 5-minute briefing. The recognition rate of each type of gesture is shown in Table II. The average response time was around 20 ms.

Although the result shows that certain gestures such as "reverse" and "circle" are more difficult to recognize than others, the overall recognition rate is already enough for operating slide show in *PowerPoint* and producing good user experience. After all, these two types of gestures are not frequently used in practice.

TABLE I.
GESTURE DEFINITION FOR CONTROLLING POWERPOINT

Gesture	Meaning/Function
rightward	next page
downward	
leftward	previous page
upward	
clockwise circle	slide show
anticlockwise circle	quit slide show
reverse	close file
clench	open file (ENTER)

TABLE II.
RECOGNITION PERFORMANCE FOR CONTROLLING POWERPOINT

Gesture	Recognition Rate (%)
rightward	95
leftward	95
upward	94
downward	93
clockwise circle	88
anticlockwise circle	87
reverse	80
clench	97

B. Controlling Need for Speed

The *Need for Speed* is a series of racing video games that has become popular around the world since 1994. It features intense racing and pursuing in the game and has a very high requirement on the reaction time of players. We used one of the latest versions called *Need for Speed: Most Wanted* to testify the performance of recognition.

The six defined gestures are shown in Table III. These gestures can realize six essential functions, controlling the movement of the car. These gestures can be also used in combination to create more versatile functions. For example, drift can be done by "right hand above left hand" (turn left) + "left palm, right fist" (hand brake).

For this application, the response time is a crucial factor in addition to the recognition rate. In our experiment, with GeForce GTX 480, the frame rate was kept at around 45 FPS with all special effects enabled, ensuring the fluency of the game and good user experience. The average response time of the recognition system was around 20 ms, an acceptable delay that players are unlikely to be aware of. The recognition rate of each type of gesture is shown in Table IV.

TABLE III.
GESTURE DEFINITION FOR PLAYING NEED FOR SPEED

Gesture	Meaning/Function
both palms 	speed up
both fists 	back off
left fist, right palm 	free run
left palm, right fist 	hand brake
left hand above right hand 	turn right
right hand above left hand 	turn left

TABLE IV.
RECOGNITION PERFORMANCE FOR PLAYING NEED FOR SPEED

Gesture	Recognition Rate (%)
both palm	99
both fist	97
left palm, right fist	98
left fist, right palm	98
right hand above left hand	99
left hand above right hand	99

V. CONCLUSION

In this paper, we presented a real-time hand gesture recognition system that can track both one-hand and two-hand movements based on Kinect and verified its performance on controlling *PowerPoint* and the most popular racing video game *Need for Speed*.

We adopted both color information and depth information to segment hand from background. To reduce the influence of illumination change, color balancing was implemented at the preprocessing stage. The detection of the second hand used the shape and depth information of the first hand, which largely simplified the process. To achieve high efficiency, only the depth image was used in tracking and a 3D ROI was set to narrow down the search range. This strategy reduced the computational burden significantly and enhanced the system's anti-disturbance ability by eliminating distractions beyond the ROI range. For pose recognition, Hu's moment invariants were adopted as the features together with SVM to distinguish palm and fist. The trajectory of hand was recognized by template matching for the sake of simplicity.

The most prominent features of our system are its efficiency and robustness, which are two appealing features especially for smart devices with limited processing capacity and less controlled working environment. In the future, we will further enhance the function of our system to handle two-hand trajectory recognition and investigate challenging issues such as hand occlusion.

REFERENCES

[1] S. Mitra and T. Acharya, "Gesture recognition: A survey," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews,* vol. 37, pp. 311-324, 2007.

[2] V. I. Pavlovic, R. Sharma, and T. S. Huang, "Visual interpretation of hand gestures for human-computer interaction: a review," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 19, pp. 677-695, 1997.

[3] J. P. Wachs, M. K, H. Stern, and Y. Edan, "Vision-based hand-gesture applications," *Communitions of the ACM,* vol. 54, pp. 60-71, 2011.

[4] H. Kang, C. W. Lee, and K. Jung, "Recognition-based gesture spotting in video games," *Pattern Recognition Letters,* vol. 25, pp. 1701-1714, 2004.

[5] D. Xu, "A neural network approach for hand gesture recognition in virtual reality driving training system of SPG," in *Proceedings of the 18th International Conference on Pattern Recognition*, pp. 519-522, 2006.

[6] D. Uebersax, J. Gall, M. Van den Bergh, and L. Van Gool, "Real-time sign language letter and word recognition from depth data," in

Proceedings of the 2011 IEEE International Conference on Computer Vision Workshops, pp. 383-390, 2011.

[7] L. Gallo, A. P. Placitelli, and M. Ciampi, "Controller-free exploration of medical image data: Experiencing the Kinect," in *Proceedings of the 24th International Symposium on Computer-Based Medical Systems,* pp. 1-6, 2011.

[8] H.-D. Yang, A.-Y. Park and S.-W. Lee, "Gesture spotting and recognition for human–robot interaction," *IEEE Transactions on Robotics,* vol. 23, pp. 256-270, 2007.

[9] L. Dipietro, A. M. Sabatini, and P. Dario, "A survey of glove-based systems and their applications," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews,* vol. 38, pp. 461-482, 2008.

[10] R. Y. Wang and J. Popović, "Real-time hand-tracking with a color glove," *ACM Transactions on Graphics,* vol. 28, no. 3, 2009.

[11] Z. Ren, J. Yuan, and Z. Zhang, "Robust hand gesture recognition based on finger-earth mover's distance with a commodity depth camera," In *Proceedings of the 19th ACM international conference on Multimedia*, pp. 1093-1096, 2011.

[12] Z. Ren, J. Meng, J. Yuan, and Z. Zhang, "Robust hand gesture recognition with kinect sensor," in *Proceedings of the 19th ACM international conference on Multimedia*, pp. 759-760, 2011.

[13] I. Oikonomidis, N. Kyriazis, and A. Argyros, "Efficient model-based 3d tracking of hand articulations using kinect," in *British Machine Vision Conference*, pp. 101.1-101.11, 2011.

[14] I. Oikonomidis, N. Kyriazis, and A. A. Argyros, "Tracking the articulated motion of two strongly interacting hands," in *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition,* pp. 1862-1869, 2012.

[15] C. Keskin, F. Kıraç, Y. E. Kara, and L. Akarun, "Real time hand pose estimation using depth sensors," in *Consumer Depth Cameras for Computer Vision*, pp. 119-137, Springer, 2013.

[16] M. Tang, "Recognizing hand gestures with microsoft's kinect," *Web Site:http://www.stanford.edu/class/ee368/Project_11/Reports/Tang_Hand_Gesture_Recognition. pdf,* 2011.

[17] B. Liang, "Gesture recognition using depth images," in *Proceedings of the 15th ACM on International conference on multimodal interaction*, pp. 353-356, 2013.

[18] J.-Y. Lee and S. I. Yoo, "An elliptical boundary model for skin color detection," in *Proceedings of the 2002 International Conference on Imaging Science, Systems, and Technology*, 2002.

[19] M. D. Breitenstein, F. Reichlin, B. Leibe, E. Koller-Meier, and L. Van Gool, "Robust tracking-by-detection using a detector confidence particle filter," in *Proceedings of the 12th International Conference on Computer Vision*, pp. 1515-1522, 2009.

[20] Z. Kalal, K. Mikolajczyk, and J. Matas, "Tracking-learning-detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 34, pp. 1409-1422, 2012.

[21] M.-K. Hu, "Visual pattern recognition by moment invariants," *IRE Transactions on Information Theory,* vol. 8, pp. 179-187, 1962.