A Combined Method of Skin-and Depth-based Hand Gesture Recognition

Tukhtaev Sokhib[1] and Taeg Keun Whangbo[2]
[1]Department of IT Convergence Engineering, Gachon University, Korea
[2]Department of Computer Science, Gachon University, Korea

Abstract: *Kinect is a promising acquisition device that provides useful information on a scene through color and depth data. There has been a keen interest in utilizing Kinect in many computer vision areas such as gesture recognition. Given the advantages that Kinect provides, hand gesture recognition can be deployed efficiently with minor drawbacks. This paper proposes a simple and yet efficient way of hand gesture recognition via segmenting a hand region from both color and depth data acquired by Kinect v1. The Inception model of the image recognition system is used to check the reliability of the proposed method. Experimental results are derived from a sample dataset of Microsoft Kinect hand acquisitions. Under the appropriate conditions, it is possible to achieve high accuracy in close to real time.*

Keywords: *Gesture recognition, Microsoft Kinect, inception model, depth.*

Received September 21, 2017; accepted September 23, 2018
https://doi.org/10.34028/iajit/17/1/16

1. Introduction

It is widely recognized today that people can use their hands to control devices without actually touching them. Hands were themselves the very first means of communication; thus, it is natural and intuitive to use hands while interacting with a gesture-driven system. Being a non-verbal way of conveyance, hands are much more common form of communication such as speech. Some difficulties of hand gesture recognition have been resolved via solutions involving computer vision [23]. Another way that hand gestures can be understood by a computer is through the data-glove approach. Parvini *et al*. [19] proposed a gesture recognition system that utilizes bio-mechanical characteristics where the range of motion of each section of the hand participating in a sign, relative to non-participating sections, is a user-independent characteristic of that sign. They used CyberGlove as a sensory device. Similarly, Deyou [4] proposed a gesture recognition approach which uses a dataglove. However, gloves must be worn, which can become cumbersome and cause sweaty hands that adversely affect personal hygiene [10]. Therefore, a system that could interact with humans without using haptic devices such as controllers, gloves, or world-grounded devices would be highly valuable. Even though there have been methods previously proposed for gesture recognition, none of them is perfect [16, 19, 23], and as a result, scholars and researchers have been motivated to learn more about hand gestures. Recent investigations show machine learning methods to be a promising tool when a large enough dataset can be supplied [3], and for hand gesture recognition tasks Kinect has shown to be favorable [16]. The demand for a large dataset is undeniable, it is because typical models of the hands have many Degrees Of Freedom (DoF). Several hand gesture recognition methods have been offered by computer vision researchers over the past decade, most of which are based on images derived from consumer depth cameras [1, 6, 21].

The structure of this paper is as follows: Section 2 discusses other related research works on gesture recognition via depth cameras. Section 3 provides in-depth information concerning the proposed method for hand gesture recognition and Section 4 reveals the experimental results obtained from the Inception classification model. Section 5 highlights some limitations of the proposed method. Finally, section 6 completes the manuscript with conclusions and future directions.

2. Literature Review

Since the introduction of depth cameras there have been many research investigations regarding hand gesture recognition. Our method relies upon color and depth images derived from Kinect, so going forward; we will limit the discussion to the most relevant depth camera-based prior works.

A distance metric called Finger-Earth Mover's Distance (FEMD) has been used for a gesture variance [22]. FEMD is capable of differentiating hand gestures using a time series curve of the fingers only, not including the whole hand. Another vision based gesture recognition method was proposed by Vaezi and Nekouie [29]. They used a 2D image and 3D model for their estimation to avoid computational complexity induced by reconstruction and inverse kinematics. One deficiency of their work is its requirement of color markers to locate the exact angles of fingers with respect to the other parts of a hand in a 2D color image. The work by Dominio *et al*. [5] is also relevant to our study. They extracted hand region from depth and color data as well and introduced two new relevant features: the distance from the palm center to the finger, and the curvature of the hand shape. The method we propose is similar to that of [5]. It is mainly intended for sedentary activities that do not require substantial changes of the scene. Under such conditions, we extract the closest points from the depth data. Other detailed information is provided in the upcoming chapter.

Some attempts successfully adapted machine learning methods to gesture recognition using Kinect's skeletal data. Bhattacharya *et al.* [2] applied Support Vector Machines and Decision Trees to recognize aircraft gestures used in military. Another promising method for hand gesture recognition is to create a model of the hand with all possible DoFs included and match the model with the current shape from Kinect. Similar work has been conducted by Microsoft Research group in [25]. The hand region of interest is derived from depth input stream coming directly from Kinect and sent to the reinitializer to assume a distribution over poses and the "golden energy", which compares the output from the reinitializer with the actual shape of the hand. The output with least error is derived from the golden energy and shown as a best matched result in real-time application. This work [25] is outstanding in comparison with other similar approaches reported in [19, 21], but lacks the ability to render a good match when practically implemented with other physical things like pens or balls. The rich information of Kinect is not only applicable for hand or face recognition but also can be used for full body pose recognition as claimed in [26]. A 3D range camera is also used to recognize hand gestures in real time. Li and Jarvis [13] created a system to identify hand gestures from a 3D range camera. They captured a depth data with a range camera to segment the hand and locate the hand in 3D space. The system has some limitations when the user holds his hand still such that the forearm and hand are the same distance from the camera. In this situation, it is difficult to segment only a hand distinguished from the forearm, and therefore a color web camera is required to segment the hand using color information. Pavlovic *et al*. [20] surveyed several approaches to modeling, analysis, and recognition of hand gestures for visual interpretation. Hand gesture and arm motion have been considered to be promising candidates for indexing visual commands to control the computer. Specific to our hand gestures, there is a work done by Trigo and Pellegrino in [28] which analyzed the use of invariant moments, k-curvature features and template matching to classify hand gestures for HCI and proposed shape descriptors. Kinect enables the manipulation of windows and gaming applications too. Guan-Feng He *et al*. [8] proposed a vision-based HCI system in which users can control windows and games such as the popular Angry Birds. Madani *et al.* [15] presents another approach where gesture signals from Wiimote device are classified to deal with numeric gestures.

Human Robot Interaction (HRI) has also become a hot topic and many works have been claimed at creating robotic devices that can mimic human actions. Gestures are crucial in HRI for robots to follow what humans do or respond to their gestural commands [7, 27]. Apart from hand gesture recognition, palm center and finger recognition have been researched over the past several years. Kinect provides full body skeleton tracking, which is especially useful when combined with depth data to detect finger position.

Skin color detection has been found to have a great impact on detecting bare hands even under fairly changing backgrounds [14]. It is difficult to have a single method for skin color detection tasks, as human skin color tone is cardinally different for people from different places. Since an Red, Green and Blue (RGB) color space is not a good option for skin color detection as explained in [24], there are a variety of other color spaces that can be used for this specific task. Through the years researchers have found the YCbCr color space to be pertinent compared to its counterparts, and Shaik *et al*. [24] presented a comparative study of skin color detection on Hue, Saturation and Value (HSV) and Luminance, Chroma-blue, Chroma-red (YCbCr). Inspired from [24], we chose the YCbCr color space to help detect human skin color in color images.

3. Proposed Method

The method we propose uses color images and depth images captured via Kinect. However, with Kinect, it is not possible to capture scene information of very near or very far distances [11]. To achieve good results, the user must stand facing the Kinect at a relevant range, that is, 0.8 m to 5.5 m. At distances closer than 0.8 m, Kinect records a scene with a lot of noise and flickering. Similarly, at distances further than 5.5m, the depth image varies substantially in brightness. As stated above, there are more than 20 DoFs for a single hand to be accurately modeled. In this paper, we used only images, and a limited number of gestures were included. On the one hand, in VR-based gaming applications or in other fields like health care, or entertainment where hand gestures are used, there is not much necessity for multiple variations of hand shapes. On the other hand, it would be hard to memorize and mimic each gesture for different tasks, even if the system could include many gestures. Having considered the previous aspects, a dataset of hand gestures [16] was selected to experimentally test our method. This dataset contains gestures performed by 14 different people, each performing 10 different gestures repeated 10 times each, for a total of 1400 gestures. We did not use all the available data though; there was an additional built-in dataset-assembled with a Leap Motion device that comes with Kinect. Figure 1 pictorially explains the overall procedure of our approach.

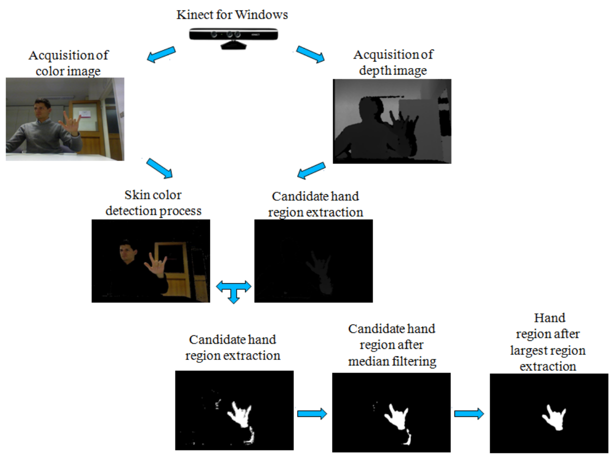

Kinect for Windows
Acquisition of color image
Acquisition of depth image
Skin color detection process
Candidate hand region extraction
Candidate hand region extraction
Candidate hand region after median filtering
Hand region after largest region extraction

Figure 1. Overall schematic of the proposed method demonstrated by pictures.

3.1. Skin Color Segmentation

Detection of skin color in images is a very popular and useful technique for face and hand gesture recognition. For most image applications, an RGB color model is preferred [9]. The RGB color model is an additive, that is, three light beams are added to form a final color. Though, it is difficult to determine specific color in RGB color model [9]. For this reason, skin color based techniques exploit other color models such as HSV or Luminance In-phase Quadrature (YIQ). We selected the YCbCr color space over all other available color spaces because color and brightness are split nicely, allowing us to perform computations on individual channels. In the YCbCr color space the luminance information is stored separately in the Y component and, the chrominance information is stored in Cb and Cr in blue and red, respectively. However, there is a transformation to easily convert from the RGB color model to the YCbCr color model:

$$\begin{aligned} Y' &= 0.299 * R' + 0.587 * G' + 0.114 * B' \\ Cb &= -0.169 * R' - 0.331 * G' + 0.500 * B' \\ Cr &= 0.500 * R' - 0.419 * G' - 0.081 * B' \end{aligned} \tag{1}$$

As such, it is possible to derive each channel separately and detect skin color values from components.

A color image is acquired from Kinect; at the same time, a depth image is also acquired, and both are sent to the preprocessing stage where the resolution of the color image is down-sampled to fit the resolution of the depth map. After converting the color image from the RGB to the YCbCr color space, we are able to detect skin color intensities. For this to be accomplished, we first extract the two main components: Cb-chrominance-blue and Cr-chrominance-red from the image in the YCbCr space. The threshold value for extracting skin color intensity values is determined based on trial of likely skin colors in the dataset chosen. In the case of our experimental dataset, intensity values less than 78 and greater than 126 in the Cb channel and intensity values less than 134 and greater than 173 in the Cr channel are found to be most likely skin color intensity values derived after experimenting with many combinations. Pixel intensities that are not considered to be skin color are converted to black, which helps when implementing the coordinate matching method. Figure 2 shows sample images from the dataset to give a clear understanding of the process.

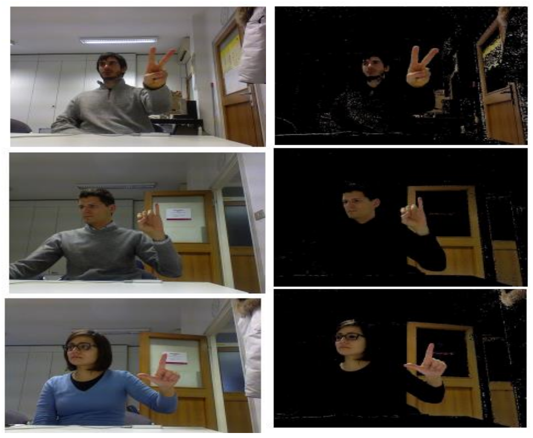


Figure 2. Samples derived from skin color extraction.

3.2. Depth Image Processing

Next, the depth data that was captured at the same time as the color image is processed. To work with depth data from Kinect we need to have a clear understanding of the nature of it. The depth stream of Kinect is made up of pixels that contain distance information from the camera plane to the nearest object. The distance is measured in millimeters. This feature is very useful when there is a need for removing a background scene from image. The depth image comes in three different resolutions: 640x480 (the default), 320x240, and 80x60. Reasonably, the largest resolution is what we need because we are going to acquire the highest quality possible. In our case, we acquire a depth image with the largest resolution and extract a candidate hand region from it. There are several ways of extracting a candidate hand region from the depth image. One of them is to enable skeletal tracking and identify the human shape. When skeletal tracking is enabled, the Kinect runtime processes the depth data to segment the player from the map and removes background pixel intensities. Furthermore, it is natural and intuitive for the users who control the Kinect-driven system with hand gestures to raise their hands in front of their torso and move their hands a bit forward in order to clearly represent the gesture. There should be nothing that occludes the user or Kinect, because it will obviously block the gesture. Since a gesture of the identified human is made in front of the camera, the closest intensity values in the near vicinity can be derived by using a threshold value. For our dataset of hand gestures, the threshold value is chosen to be an intensity value greater than 15 and less than 23. These exact numbers are chosen after examining the frequency of pixels values of the depth image. In normal cases, the user's body covers 8-10 neighboring pixel intensity values; we arrived at this conclusion after observing the histogram distributions of several depth images. Figure 3 presents some of the histograms of those depth images in one plot.

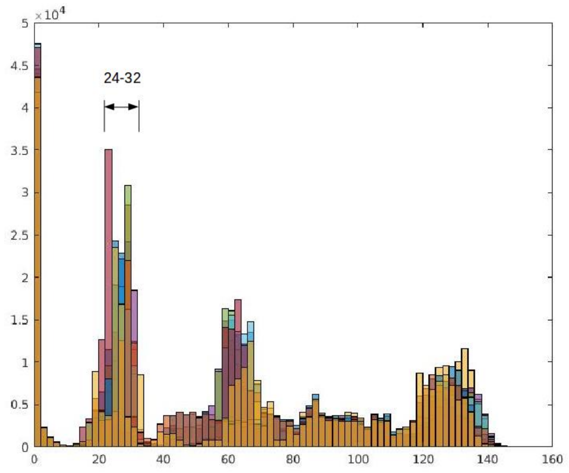
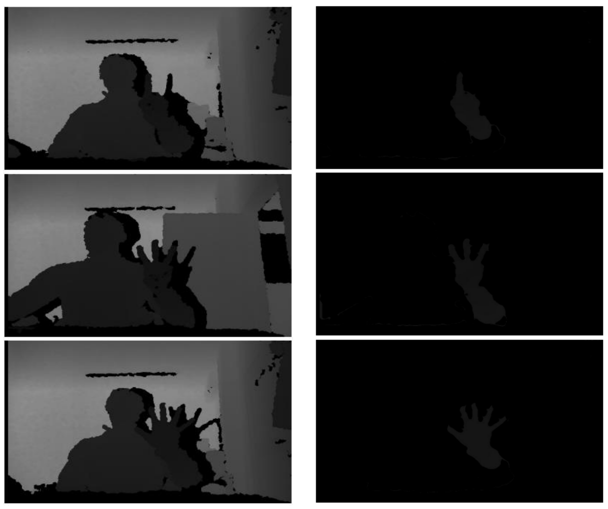


Figure 3. Histogram distributions of 10 different depth images colored in different tones.

As can be seen in Figure 3, the most frequent intensity values range from 24 to 32, and these values are of the human body's. We are not considering the black values since they represent invalid distance information, therefore they can be safely ignored. Users perform gestures a bit closer to the Kinect, meaning that smaller intensity values than 24 and greater than 15 are possible hand candidates. These candidate values sometimes include some noise as well as parts of the forearm. We remove those types of noise using a median filter with a slightly larger threshold, and forearm parts are eliminated by coordinate matching, which is performed using a color image of the same frame. Pixels that are out of this threshold are set to black and will be used for coordinate matching. Figure 4 illustrates output images after the threshold value has been applied to the depth images.



a) Depth images from Kinect. b) Thresholded results.

Figure 4. Samples derived after threshold value is applied.

3.3. Coordinate Matching and Hand Candidate Region Extraction

The coordinate matching stage is not difficult to complete, but it plays significant role in extracting a hand and neighboring hand candidate regions. Firstly, we obtain both skin segmented and thresholded images from the previous two processes given in above sub-chapters. Then a skin color image is converted to a grayscale intensity image because a color image is three dimensional and the depth image to be matched is two dimensional, therefore we need to have both images be of the same dimension. Converting an RGB image to grayscale does not hinder our method because after skin segmentation, non-skin pixel values are converted to black, which means that black remains black while skin-color pixel values assume some value of gray. After all these processes are complete, we compare each single pixel of the thresholded image and gray image, and if the pixel value at the corresponding location of both images is not black, we mark this pixel value as white; if the condition is not met, we mark it as black. The pseudo code of an algorithm is given below for one image pair:

Algorithm 1. RegionExtraction

```
ti=thresh_image
gi=gray_image
rt=number_of_rows_of_ti
ct=number_of_columns_of_ti
vt=pixel_value_of_ti
rg=number_of_rows_of_gi
cg=number_of_columns_of_gi
vg=pixel_value_of_gi
for(i=1 to rt)
  for(j=1 to ct)
  {
    for(k=1 to rg)
    {
      for(l=1 to cg)
      {
        if ti[i,j] && gi[k,l] !=0
        {
                ti[i,j]=255
            }
            else
            {
                ti[i,j]=0
            }
        }
    }
}
return ti
```

With this technique, it is possible to extract a hand and its closest regions more accurately because the forearm parts in the color image are removed by the thresholded image, and other noise in the depth image is removed by the color image. Figure 5 displays an acquired image after using this technique.

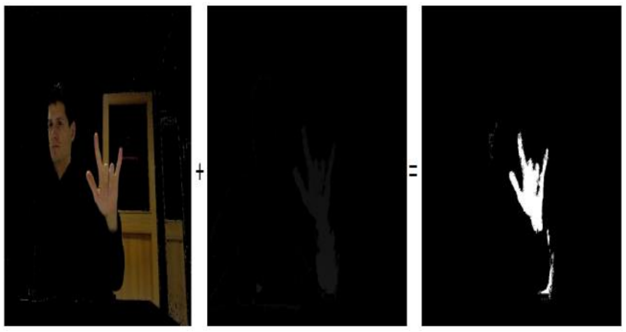
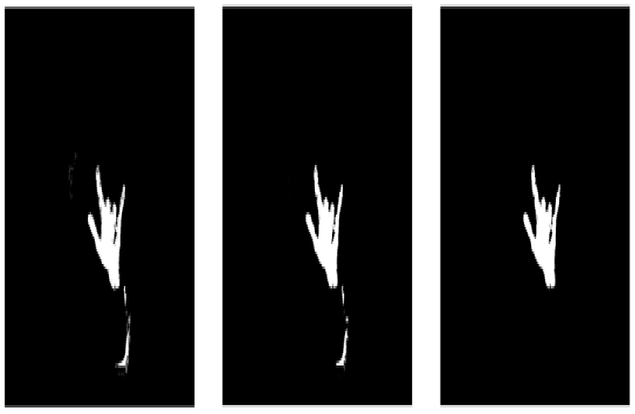


Figure 5. A result after coordinate matching is applied.

The results after coordinate matching are good but still not sufficient. Therefore, we further improve the image using preprocessing tools like a 3x3 median filter, where the largest region is extracted in the image using an object extraction method. Median filtering not only removes salt and pepper noise but also smooths the image content. This helps recover the hand image. Figure 6 depicts the median filtering process.



a) An original image. b) An image after median filtering. c) The final result.

Figure 6. Preprocessing results.

4. Experiment and Results

In this chapter, we assess the performance of our method utilizing Inception-an image recognition system. We used the results derived from preprocessing stage as the input image. The dataset of binary images was created by applying our method to the existing dataset of Microsoft Kinect hand acquisitions. The dataset consists of 1200 images of 10 different gestures. The gestures are based on American Sign Language (ASL) as used in [12]. Figure 7 depicts some example images of the dataset.

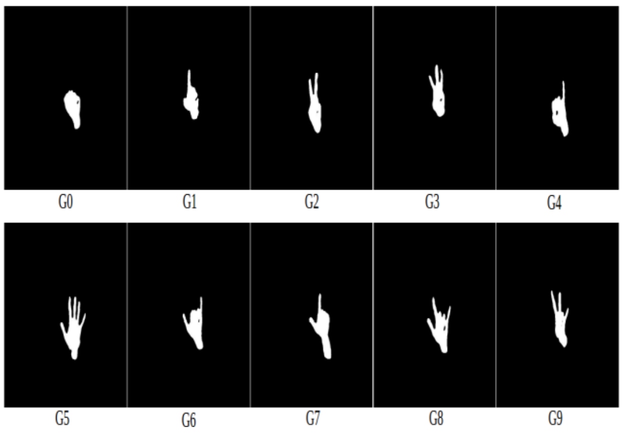


Figure 7. Dataset samples originated from Microsoft Kinect hand acquisitions [16].

We performed a process referred to as inductive transfer, which means applying learnings from previous training sessions to a new training session. Inception is a large image classification model with millions of parameters that can differentiate many images. For our purpose we only used the last layer of the network and adapted it for gesture classification. We classified our gesture images using a laptop with an Advanced Micro Devices (AMD) A10 processor, Gallium 0.4 on AMD Aruba, 8GB RAM on an Ubuntu OS. Therefore, we took advantage of a virtual software container-docker on our machine for implementing our training without actually installing the necessary dependencies. Tensorflow is an open source software platform released by Google to build and train deep learning models; we employed this tool to train our network and show the robustness of our method. We used 10 gestures, and the results are measured by values ranging between 0 and 1. The closer the value is to 1 the more accurate the model is. From Table 1 to Table 10, we present the accuracy results of our method for 10 gestures with 15 examples each. Gesture types that did not have any probability or had only minor probability were excluded from the table for space efficiency.

Table 1. The accuracy results for G0[1].

G\№	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
G0	**0.99**	**0.99**	**0.93**	**0.98**	**0.99**	**0.97**	**0.97**	**0.96**	**0.97**	**0.99**	**0.99**	**0.99**	**0.94**	**0.99**	**0.99**
G1						0.02	0.01								
G4			0.05						0.02				0.03		

Table 2. The accuracy results for G1.

G\№	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
G0														0.05	
G1	**0.87**	**0.88**	**0.87**	**0.91**	**0.95**	**0.84**	**0.96**	**0.84**	**0.89**	**0.89**	**0.86**	**0.87**	**0.96**	**0.86**	**0.90**
G2				0.04				0.13					0.02		
G3											0.05				
G4			0.05		0.02	0.03	0.01		0.04			0.11			0.04
G7	0.05	0.05								0.05					

Table 3. The accuracy results for G2.

G\№	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
G1	0.01	0.08											0.04		
G2	**0.97**	**0.83**	**0.86**	**0.90**	**0.99**	**0.95**	**0.95**	**0.99**	**0.91**	**0.98**	**0.90**	**0.89**	**0.94**	**0.86**	**0.93**
G3				0.03		0.02								0.10	0.03
G7												0.06			
G8			0.052				0.03								
G9									0.04		0.06				

Table 4. The accuracy results for G3.

G\№	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
G0								0.05							
G2									0.04	0.11					
G3	**0.98**	**0.96**	**0.92**	**0.93**	**0.93**	**0.93**	**0.97**	**0.91**	**0.90**	**0.87**	**0.95**	**0.97**	**0.94**	**0.97**	**0.97**
G9		0.02	0.06	0.03	0.03	0.02	0.01				0.03	0.01	0.02	0.02	

Table 5. The accuracy results for G4.

G\№	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
G0														0.04	0.07
G1	0.06			0.06	0.03	0.03	0.05	0.03		0.06		0.05			
G4	**0.91**	**0.89**	**0.86**	**0.90**	**0.90**	**0.93**	**0.92**	**0.95**	**0.92**	**0.88**	**0.82**	**0.87**	**0.89**	**0.91**	**0.92**
G6		0.08											0.05		
G7			0.07						0.09		0.11				

Table 6. The accuracy results for G5.

G\№	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
G5	**0.99**	**0.95**	**0.99**	**0.95**	**0.99**	**0.98**	**0.99**	**0.99**	**0.93**	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**	**0.90**
G8				0.04					0.03						
G9		0.02													0.08

Table 7. The accuracy results for G6.

G\№	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
G4								0.04							
G5												0.08			
G6	**0.97**	**0.99**	**0.82**	**0.99**	**0.96**	**0.98**	**0.94**	**0.86**	**0.94**	**0.92**	**0.89**	**0.89**	**0.96**	**0.96**	**0.94**
G7							0.04			0.07			0.02		
G8			0.17		0.02	0.01					0.09				0.04
G9														0.03	

Table 8. The accuracy results for G7.

G\№	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
G1						0.10									
G4											0.06				0.02
G7	**0.97**	**0.89**	**0.97**	**0.85**	**0.99**	**0.89**	**0.99**	**0.98**	**0.90**	**0.96**	**0.92**	**0.99**	**0.91**	**0.98**	**0.95**
G8			0.02	0.06					0.05	0.03			0.07		
G9		0.04													

Table 9. The accuracy results for G8.

G\№	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
G2										0.02					
G3				0.04											
G6						0.04		0.06						0.04	
G7	0.03						0.06		0.03		0.02		0.03		
G8	**0.96**	**0.95**	**0.99**	**0.91**	**0.97**	**0.94**	**0.92**	**0.92**	**0.93**	**0.94**	**0.96**	**0.88**	**0.93**	**0.92**	**0.98**
G9		0.03										0.07			

Table 10. The accuracy results for G9.

G\№	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
G2			0.02			0.05	0.05	0.06		0.05			0.05		0.03
G3		0.05		0.11										0.01	
G4					0.05										
G5											0.01				
G8	0.06											0.05			
G9	**0.92**	**0.92**	**0.93**	**0.86**	**0.93**	**0.91**	**0.90**	**0.91**	**0.99**	**0.93**	**0.97**	**0.90**	**0.93**	**0.97**	**0.92**

The leftmost column defines a gesture type according to Figure 7. The bold row represents the observed gesture type. This applies to Tables 1-10. As seen in the tables, the algorithm works satisfactorily. We tallied the average percentage of each gesture and created a confusion matrix out of 15 test images for each gesture. The accuracy results in Tables 1-10 were converted to integer values and the contributions that did not reach 1% were removed from the table. All the gestures had a 90% recognition rate at least. With this we can effectively verify the robustness of our algorithm. Table 11 presents an average percentage for each gesture.

Table 11. Average results for 10 gestures with 15 examples each.

G0	**98**				1					
G1		**90**	1		5			1		
G2		1	**93**	1						
G3			1	**94**						2
G4	1	3			**90**		1	2		
G5						**98**			1	1
G6							**94**	4	2	
G7		1			1			**95**	2	
G8							1	1	**94**	1
G9			2	1					1	**93**
	G0	**G1**	**G2**	**G3**	**G4**	**G5**	**G6**	**G7**	**G8**	**G9**

Moreover, we successfully integrated the Kinect device and the proposed algorithm as a standalone system to calculate the timings in real time situations. The algorithm was able to detect the gesture in 0.1-0.3 seconds and most existing gesture-based applications could make use of the proposed method in real time.

5. Limitations

While experimenting in a real time environment, we witnessed that the system occasionally takes more time to identify a gesture and the user needs to hold up his hand for 0.2-0.3 seconds. These shortcomings can be attributed to global illumination effects in a room, and the system needs to be improved in terms of the skin color detection process. According to our recognition conditions, a user can raise their left hand only; however, after implementing mirroring effect of the frame we were able to recognize right-handed gestures as well. Notwithstanding, the recognition rate was reduced by approximately 6%. Furthermore, the system fails to recognize gestures when a user is occluded or shows a gesture that has not been trained before.

6. Conclusions and Future Studies

This article presents a novel approach to gesture recognition for Kinect-based applications that require non-moving and sitting activities. With less number of gestures, this work differs from other existing approaches [16, 17, 18] in that its recognition rate is high and it can be implemented in close to real-time.

There are some theoretical and practical issues that need to be addressed. On the theoretical side, illumination problems and hand segmentation parts must be improved by implementing relatively advanced software. On the practical side, the number of users must be increased up to 3-4 users. As the next step, we aim to include more gesture types of ASL and segment gestures performed with both hands. In the near future, we plan to create a dataset of gestures performed with both hands having more than 10 gestures.

References

[1] Ahn Y., Park Y., Choi K., Park W., Seo H., and Jung K., "TOF Depth Camera 3D Gesture Interaction System," *in Proceedings of International Conference on Advances in Information Technology*, Bangkok, pp. 11-17, 2012.

[2] Bhattacharya S., Czejdo B., and Perez N., "Gesture Classification with Machine Learning Using Kinect Sensor Data," *in Proceedings of 3rd international Conference on Emerging Applications of Information Technology*, Kolkata, pp. 727-31, 2012.

[3] Dean J., Corrado G., Monga R., Chen K., Devin M., Le Q., Mao M., Ranzato M., Senior A, Tucker P., Yang K., and Ng A., "Large Scale Distributed Deep Networks," *in Proceedings of the 25th International Conference on Neural Information Processing Systems*, Lake Tahoe, pp. 1232-1240, 2012.

[4] Deyou X., "A Network Approach for Hand Gesture Recognition in Virtual Reality Driving Training System of SPG," *in Proceedings of the 18th International Conference on Pattern Recognition*, Hong Kong, pp. 519-522, 2006.

[5] Dorninio F., Donadeo M., Marin G., Zanuttigh P., and Cortelazzo G., "Hand Gesture Recognition with Depth Data," *in Proceedings of the 4th ACMI/EEE International Workshop on Analysis and Retrieval of Tracked Events and*

Motion in Imagery Stream, Barcelona, pp. 9-16, 2013.

[6] Elmezain M., Al-Hamadi A., Appenrodt J., and Michaelis B., "A Hidden Markov Model-Based Continuous Gesture Recognition System for Hand Motion Trajectory," *in Proceedings of the 19th International Conference on Pattern Recognition*, Tampa, pp. 1-4, 2008.

[7] Gai S., Jung J., and Yi J., "Mobile Shopping Cart Application Using Kinect," *in Proceedings of the 10th International Conference on Ubiquitous Robots and Ambient Intelligence*, Jeju, pp. 289-291, 2013.

[8] He G., Kang S., Song W., and Jung S., "Real-Time Gesture Recognition Using 3d Depth Camera," *in Proceedings of IEEE 2nd International Conference on Software Engineering and Service Science*, Beijing, pp. 187-190, 2011.

[9] Ibraheem N., Hasan M., Khan R., and Mishra P., "Understanding Color Models: A Review," *ARPN Journal of Science and Technology*, vol. 2, no. 3, pp. 265-275, 2012.

[10] Jerald J., *the VR Book. Human-Centered Design for Virtual Reality*, Association for Computing Machinery and Morgan and Claypool New York, 2016.

[11] Khoshelham K., "Accuracy Analysis of Kinect Depth Data," *In: ISPRS Workshop Laser Scanning*, Calgary, 2011.

[12] Kulkarni V. and Lokhande S., "Appearance based Recognition of American Sign Language Using Gesture Segmentation," *International Journal on Computer Science and Engineering*, vol. 2, no. 3, pp. 560-565, 2010.

[13] Li Z. and Jarvis R., "Real Time Hand Gesture Recognition Using A Range Camera," *in Proceedings of Australasian Conference on Robotics and Automation Sydney*, pp. 529-534, Australia, 2009.

[14] Liu L., Sang N., Yang S., and Huang R., "Real-Time Skin Color Detection under Rapidly Changing Illumination Conditions," *IEEE Transactions on Consumer Electronics*, vol. 57, no. 3, pp. 1295-1302, 2011.

[15] Madani T., Tahir M., Ziauddin S., Raza S., Ahmed M., Khan M., and Ashraf M., "An Accelerometer-Based Approach To Evaluate 3D Unistroke Gestures," *The International Arab Journal of Information Technology*, vol. 12, no. 4, pp. 389- 394, 2015.

[16] Marin G., Dominio F., and Zanuttigh P., "Hand Gesture Recognition with Leap Motion and Kinect Devices," *in Proceedings of IEEE International Conference on Image Processing*, pp. 1565-1569, Paris, 2014.

[17] Nagarajan S. and Subashini T., "Static Hand Gesture Recognition for Sign Language Alphabets using Edge Oriented Histogram and Multi Class SVM," *International Journal of Computer Applications*, vol. 82, no. 4, pp. 28-35, 2013.

[18] Pansare J., Gawande S., and Ingle M., "Real-Time Static Hand Gesture Recognition for American Sign Language (ASL) in Complex Background," *Journal of Signal and Information Processing*, vol. 3, no. 3, pp 364-367, 2012.

[19] Parvini F., McLeod D., Shahabi C., Navai B., Zali B., and Ghandeharizadeh S., "An Approach to Glove-Based Gesture Recognition," *in Proceedings of International Conference on Human-Computer Interaction*, San Diego, pp. 236-245, 2009.

[20] Pavlovic V., Sharma R., and Huang T., "Visual Interpretation of Hand Gestures for Human-Computer Interaction: A Review," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 677-695, 1997.

[21] Ren Z., Meng J., and Yuan J., "Depth Camera Based Hand Gesture Recognition and its Applications in Human- Computer-Interaction," *in Proceedings of 8th International Conference on Information, Communications and Signal Processing*, Singapore, pp. 1-5, 2011.

[22] Ren Z., Yuan J., and Zhang Z., "Robust Hand Gesture Recognition Based on Finger-Earth Movers Distance with a Commodity Depth Camera," *in Proceedings of the 19th International Conference on Multimedea*, Scottsdale, pp. 1093-1096, 2011.

[23] Samantaray A., Nayak S., and Mishra A., "Hand Gesture Recognition using Computer Vision," *International Journal of Scientific and Engineering Research*, vol. 4, no. 6, pp. 1602-1609, 2013.

[24] Shaik K., Ganesan P., Kalist V., Sathish B., and Jenitha M., "Comparative Study of Skin Color Detection and Segmentation in HSV and YCbCr Color Space," *Procedia Computer Science*, vol. 57, 41-48, 2015.

[25] Sharp T., Keskin C., Robertson D., Taylor J., Shotton J., Kim D., Rhemann C., Leichter I., Vinnikov A., Wei Y., Freedman D., Kohli P., Krupka E., Fitzgibbon A., and Izadi S., "Accurate, Robust, and Flexible Real-Time Hand Tracking," *in Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, Seoul, pp. 3633-3642 2015.

[26] Shotton J., Fitzgibbon A., Cook M., Sharp T., Finocchio M., Moore R., Kipman A., and Blake A., "Real-Time Human Pose Recognition in Parts from a Single Depth Image," *in Proceedings of IEEE Conference Computer*

Vision and Pattern Recognition, Providence, pp. 1297-1304, 2011.

[27] Song Y., Gu Y., Wang P., Liu Y., and Li A., "A Kinect based Gesture Recognition Algorithm using GMM and HMM," *in Proceedings of 6th International Conference on Biomedical Engineering and Informatics*, Hangzhou, pp. 750-754, 2013.

[28] Trigo T., Roberto S., and Pellegrino S., "An Analysis of Features for Hand-Gesture Classification," *in Proceedings of 17th International Conference on Systems, Signals and Image Processing*, Rio de Janeiro, pp. 412-415, 2010.

[29] Vaezi M. and Nekouie M., "3d Human Hand Posture Reconstruction Using A Single 2d Image," *International Journal of Human Computer Interaction*, vol. 1, no. 4, pp. 83-94, 2011.

Tuktaev Sokhib received the B.S. and M.S. degrees from Tashkent University of Information Technologies, Tashkent, Uzbekistan in 2015 and Gachon University, in the faculty of IT Convergence Engineering, Korea in 2018. His research interests include Computer Vision, Image Processing, Machine/Deep Learning and Artificial Intelligence.

Taeg Keun Whangbo received the M.S. degree from City University of New York in 1988 and the Ph.D. degree both in Computer Science from Stevens Institute of Technology in 1995. Currently, he is a professor in the Department of Computer Science, Gachon University, Korea. Before he joined the Gachon University, he was the software developer in Q-Systems which is located in New Jersey from 1988 to 1993. He was also the researcher in Samsung Electronics from 2005 to 2007. From 2006 to 2008, he was the president of the Association of Korea Cultural Technology. His research areas include Computer Graphics, HCI and VR/AR.