International Journal of Multimedia and Ubiquitous Engineering
Vol.8, No.6 (2013), pp.339-348
http://dx.doi.org/10.14257/ijmue.2013.8.6.33

Real-Time Hand Gesture-Based Interaction with Objects in 3D Virtual Environments

Jong-Oh Kim[1], Mihye Kim[2] and Kwan-Hee Yoo[1*]

[1]*Department of Digital Informatics and Convergence, Chungbuk National University*
410 Seongbongro, Heungukgu, Cheongju, Chungbuk, South Korea
[2]*Department of Computer Science Education, Catholic University of Daegu*
13-13 Hayangro, Hayangeup, Gyeongsangsi, Gyeongsnagbukdo, Korea

khyoo@chungbuk.ac.kr, mihyekim@cu.ac.kr

Abstract

This paper introduces a hand-gesture-based user interface that enables users to control virtual objects in three-dimensional (3D) virtual environments. Users are able to manipulate objects by moving, rotating, scaling, and selecting their hands in the real world. A gesture-based virtual simulation system for vehicle maintenance is developed, incorporating several object-manipulation features (such as opening a car door, sitting in a seat, replacing tires, and disassembling the engine) to evaluate the effectiveness of this protocol. Our results indicate that the proposed hand-gesture-based user interface could potentially be employed in place of a mouse or keyboard to interact with objects in 3D virtual environments.

Keywords: *Hand gesture, User interface, Gesture recognition, Virtual objects*

1. Introduction

Owing to continuing advances in 3-dimensional (3D) technologies, a broad array of virtual reality applications based on 3D virtual environments (VEs) have been implemented in various areas. A typical user interface for interacting with objects in such a 3D VE system utilizes a mouse or a keyboard. The Wii is also popular with gamers, and employs an ordinal user interface to support more realistic human–machine interaction. More recently, hand-gesture-based interfaces [1-7] have emerged as a promising interaction technique due to their natural properties. In the real world, humans generally manipulate objects by moving their hands or fingers. Accordingly, it is desirable to develop a more intuitive and natural user interface by imitating real-world interactions in a VE. A considerable amount of research on hand-gesture interfaces has been carried out over the past several decades. Nevertheless, gesture-recognition techniques are still in their infancy and remain the subject of ongoing research [8]. Additionally, most systems use webcam-like devices to capture a sequence of hand-gesture images, and thus accuracy and performance are compromised by their inadequate capacity for real-time image processing [9]. Furthermore, a given hand-gesture-recognition system is not readily applied to other areas, as these systems are usually limited to a specific application environment, domain, and even human cultural background [10].

* A corresponding author

ISSN: 1975-0080 IJMUE

This paper proposes a more flexible hand-gesture-based user interface that enables users to control virtual objects in 3D VEs. Users are able to manipulate objects by moving, rotating, scaling, and selecting their hands in the real world, and to generate callback and delegate functions to control hand-gesture events. We utilize Microsoft Kinect [11] to capture hand movements in real time. The idea behind this approach was partially introduced in [12] and [13], and it was applied to control virtual cameras in virtual spaces [9]. In this paper, we clarify the technique and generalize the target objects from virtual cameras to all types of objects. Furthermore, we develop a gesture-based virtual simulation system for vehicle maintenance to evaluate the effectiveness of the proposed protocol. The car maintenance system is implemented with several object-manipulation features, such as opening a car door, selecting a seat and sitting in it, replacing tires, and disassembling the engine. The results are very promising for object control in virtual spaces.

The remainder of the paper is organized as follows. Section 2 reviews the existing literature on hand-gesture-based user interfaces in 3D VEs. Section 3 describes the proposed hand-gesture interface for controlling objects in VEs. Section 4 introduces the hand-gesture-based virtual simulation system for vehicle maintenance and its object manipulation features. Section 5 concludes the paper with some directions for future research.

2. Related Work

Much research has examined hand-gesture-based interaction with objects in virtual spaces. In the early days of this research, mechanical data gloves such as the CyberGlove were used to obtain information on hand motions [2, 4]. In this approach, users had to wear burdensome, uncomfortable data gloves that limited the natural movements of their hands. To overcome this drawback, vision-based approaches are driven by gesture recognition, using a video camera or webcam instead of a wearable device. One such approach is DesIRe [1], which enables users to create, move, rotate, scale, and select objects in VEs by using predefined hand gestures by means of an infrared LED. Dardas and Alhaj [2] implemented a technique that uses hand gestures to generate commands for interacting with objects in games, combining Bag-of-Features and Support Vector Machine (SVM). In a real-time approach, Fang et al. [3] developed a more accurate gesture recognition method based on fast hand tracking, hand segmentation, and multi-scale feature extraction. In a similar way, Rautary and Agrawal [4] implemented a user-friendly hand-gesture-based interface for interacting with objects in virtual spaces, using a webcam.

More recently, recognition-based approaches using Microsoft Kinect [11] have attracted greater interest than techniques that employ general 2D cameras (such as webcams, video cameras, or digital cameras) to obtain a sequence of hand motions because of Kinect's superior real-time image-processing features, which enable greater accuracy and enhanced performance [9]. Slambekova [5] developed a more natural technique for controlling objects in a virtual world, using Kinect to combine gaze direction with hand gestures. Several university research groups have used Kinect to acquire information on hand motions. The MXR research lab at the University of Southern California has uploaded a video that demonstrates avatar actions (such as shaking hands, bowing, hugging, and flying) and virtual camera movements in a VE controlled by real-world hand, arm, and body motions using Kinect and Open Natural Interaction [6]. The MIT media lab has implemented DepthJS, a web browser extension

that allows Kinect to interact with any web page in Safari or Chrome [7]. In the present research, we develop a more flexible and accurate method for steering objects in virtual spaces based on Microsoft Kinect. We used callback and delegate functions to process hand-gesture events and applied depth calibration and depth cueing to the images captured from Kinect.

3. Hand Gesture-based User Interface System

3.1. System Architecture

In this paper, we propose a hand-gesture-based user interface system that enables users to control objects in 3D virtual environments. This system was developed by extending the gesture-based control method of Kim *et al.*, [9] from virtual cameras to all types of objects. To obtain faster and more accurate hand movements in real time, we employ Microsoft Kinect, which allows skeleton images of one or two persons to be tracked, as a motion-sensing input device. Kinect provides a software development kit (SDK) designed for Xbox 360 video game consoles and Windows-based PCs [11]. Position information on the left and right hands can be obtained via Kinect. A position in 3D space is represented by three Cartesian coordinates (x, y, depth) at each joint of the skeleton. An acquired position is first filtered to suppress image noise and then reprocessed with depth calibration and depth cueing to reduce image flickering. Kinect supports some filtering features, but the resulting object boundaries remain very unstable and flickering [14]. After all filtering processes have been carried out, the system will recognize hand gestures and will then control objects in a virtual space according to these gestures. Figure 1 shows the architecture of the proposed system for controlling objects in VEs [9, 13].

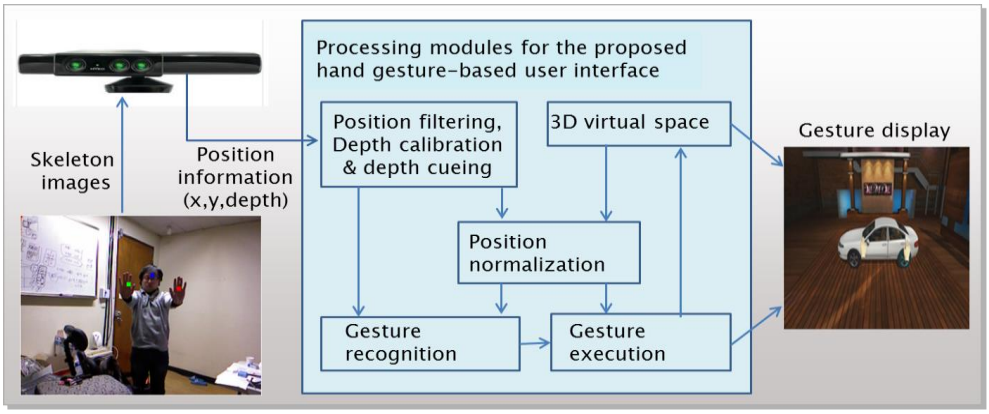

Figure 1. Architecture of the proposed system for controlling objects in virtual spaces

3.2. Gesture Recognition and Execution

As shown in Figure 2, the proposed hand-gesture-based user interface system is composed of two processes: gesture recognition and gesture execution.

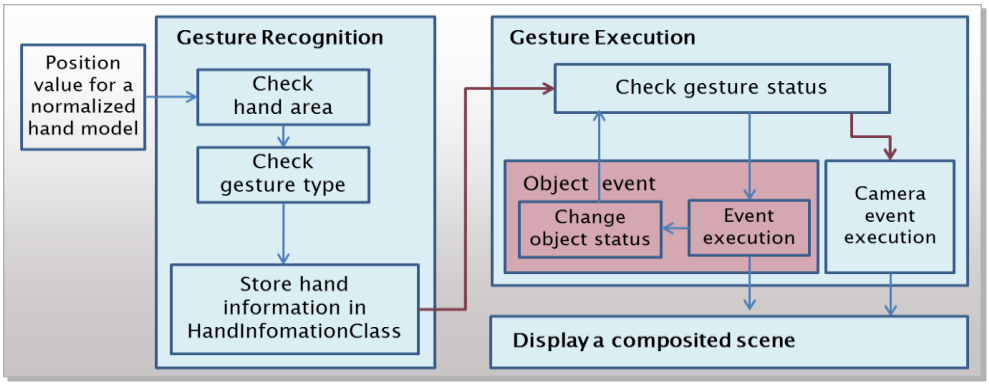

Figure 2. Overall configuration of the previously proposed method [2]

3.2.2. Gesture Recognition: The gesture-recognition process is divided into two phases: hand-area detection and gesture-type detection. As shown in Figure 3, the hand-area detection phase is processed in terms of three areas determined by the position values of the user's hands: the gesture-inactivation, gesture-activation, and object-selection areas. Before processing gesture recognition, depth calibration and cueing operations are applied to a sequence of positions, along with position filtering, to obtain more accurate gestures. In the depth-calibration process, a sequence of user arm positions is calibrated by $|A_{max} - A_{min}|$, where A_{max} is the average user arm length in a 5-second capture in which the arm is most extended, and A_{min} is the average user arm length in a 5-second capture in which the arm is most contracted. In other words, the position information for the user arm is defined as the absolute value of the difference between two values, A_{max} and A_{min}. Depth cueing is then carried out to smooth the hand coordinates using a medium filter. These operations provide the hand position value by which a gesture is detected.

As Figure 3 shows, a hand gesture in the inactivation area is ignored. In other words, if the position value of a user hand is less than or equal to 0.15, the gesture will be ignored. If the position value is greater than 0.15 and less than 0.8, the hand-recognition system will be activated to recognize the gesture. For more accurate gesture recognition, a standby time of 3 seconds is specified when a user hand satisfies $0.15 < |A_{max} - A_{min}|$. We refer to this boundary as a valid area. Accordingly, 3 seconds after a user hand enters the activated area, the recognition system prompts the input mode that allows the gesture to be recognized. If the depth value is greater than or equal to 0.8, a virtual object corresponding to the position of the user hand will be selected (picked) 3 seconds after the hand enters the object-selection area. Finally, information on the hand position (x, y, length), gesture type, and selected object is stored in the hand information class *HandInformationClass*.

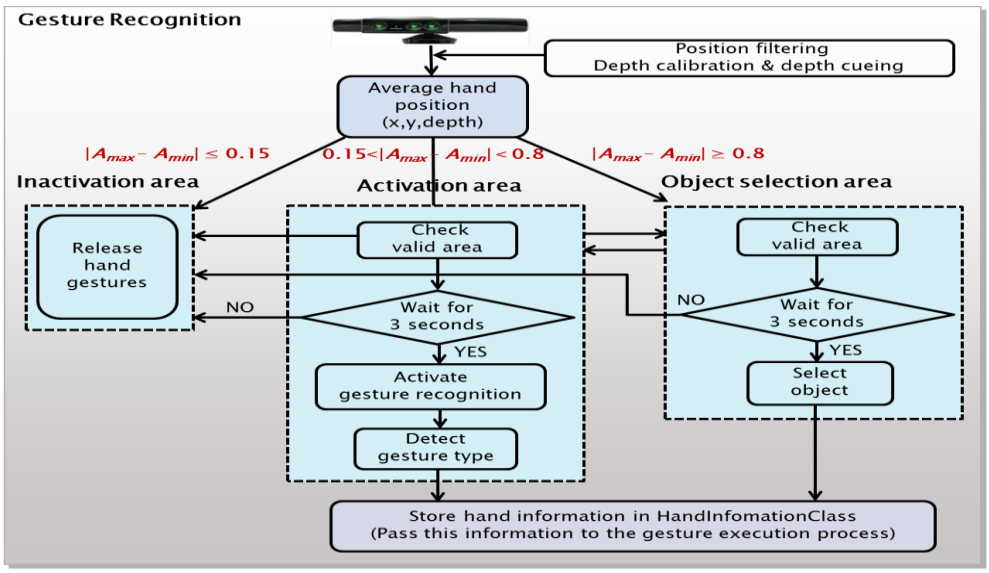

Figure 3. Gesture recognition procedures

3.2.2. Gesture Execution: After the information is stored in *HandInformationClass*, the selected virtual object can be controlled in the virtual space. The proposed object-control system includes 27 gesture events for controlling objects in a virtual space created from the possible combinations of gesture statuses of the left and right hands (none, picking, and four movement vectors: left, right, up, and down). In a manner similar to the technique used in [9], 16 of the 27 gestures are created when both hands are independently moved according to the four movement vectors (4 x 4), and eight gestures are created when only one hand is moved while the other hand is fixed. Of the remaining three gestures, one is generated by no action, and the other two by left-hand and right-hand picking motions. In this paper, eight of the twenty-seven gestures are defined as shown in Figure 4.

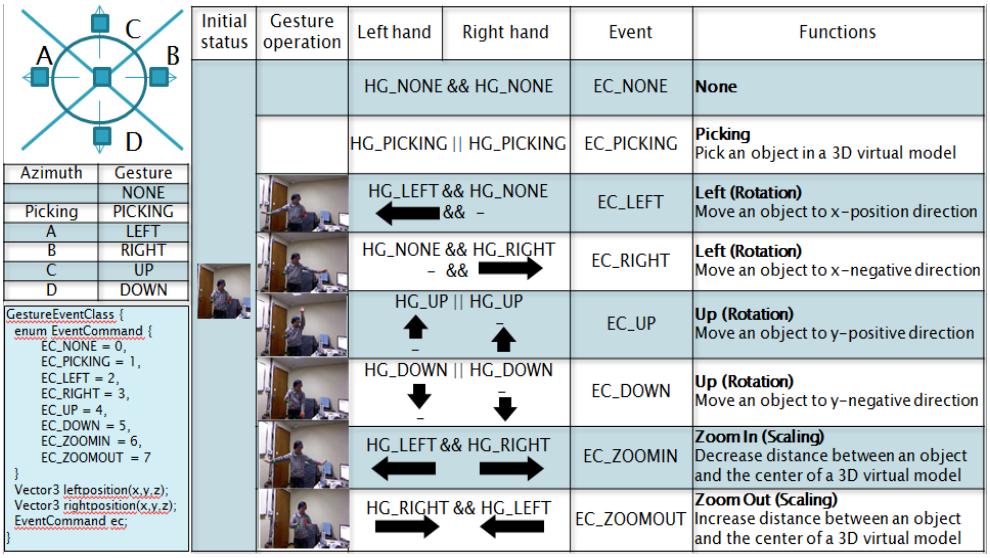

Figure 4. Eight gesture events for controlling objects in VEs

A detected gesture event is stored in the event class *GestureEventClass*, as shown in the left and lower parts of Figure 4. If the detected hand motion is one of the object-control gestures defined in our system, the gesture event corresponding to the recognized hand motion will be applied to the object in the virtual space by the event executor (*EventExecutor*). In other words, the object in the virtual space will be manipulated according to the corresponding gesture-event function. If a hand motion is not recognized (i.e., in the case of a general hand motion), the gesture will be ignored.

Figure 5 shows the structure for executing gesture events to control objects in a virtual space [13]. The class *EventExecutor* employs registered object-control functions to control objects in a virtual space, by analyzing gesture events in *GestureEventClass*. If a virtual object can be controlled by hand gestures, we set the class *EventExecutor* as a member of the object instance, create a callback function to be executed on the gesture event for the object, and link the generated function to a corresponding gesture delegate function in *EventExecutor* based on gesture type. Then, when a gesture event occurs, the system executes the function *executeCommand*(*GestureEvent* ge), to which the gesture event is passed as a *GestureEvent* instance, and executes the generated callback function via the corresponding delegate function in *EventExecutor*. Accordingly, a real person can interact with 3D virtual objects in VEs, just as he/she does with real-world objects.

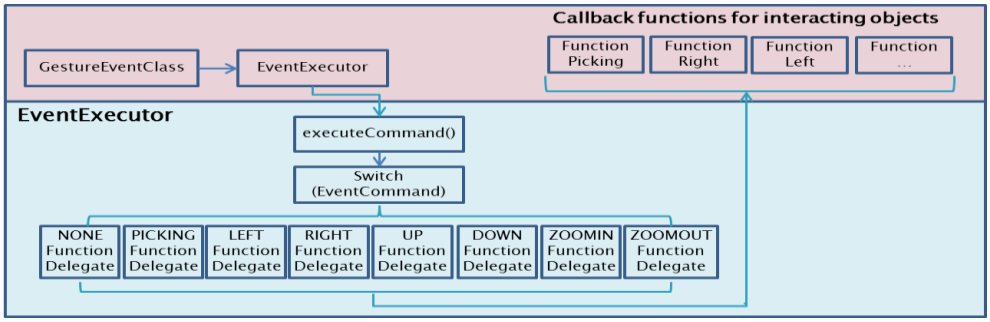

Figure 5. Execution structure of gesture events for object control

4. Hand Gesture-based Virtual Vehicle Simulation System

To evaluate the effectiveness of the proposed protocol, we have developed a number of applications, including a gesture-based gallery system [13], a gesture-based virtual simulation system, and a gesture-based virtual reality system. In this paper, we present a gesture-based virtual simulation system for vehicle maintenance. This system incorporates a number of object-manipulation features, such as opening a car door, selecting a seat and sitting in it, replacing tires, and disassembling the engine. It was written in C# using Microsoft Visual Studio 2008, Kinect, and Unity 3D and was implemented on a Windows 7 platform with an Intel Core i5-2500 3.30 GHz Quad-Core processor.

Figure 6 shows the simulated replacement of a flat tire on a virtual car with a spare tire from the virtual trunk. Figure 6 (a) shows the initial 3D virtual space in which the

virtual car is located, together with flesh-colored virtual left and right hands for controlling the virtual object (car). When a user hand is located in the inactivation area (i.e., $|A_{max} - A_{min}| \leq 0.15$) or no gesture is made, the virtual hand remains flesh colored, as shown in Fig. 6 (a). If a user hand enters the activation area (i.e., $0.15<|A_{max} - A_{min}|<0.8$), the virtual hand begins to turn red, as shown in Figure 6 (b). After 3 seconds, the virtual hand is completely red and is surrounded by four arrow keys, indicating that the gesture recognition system has begun to recognize the hand gesture, as shown in the right-hand part of Figure 6 (c). If a user hand enters the object-selection area (i.e., $|A_{max} - A_{min}| \geq 0.8$), the virtual hand begins to turn blue. If the hand is completely blue and surrounded by four arrow keys, as shown in the left-hand part of Figure 6 (c), the virtual object at the hand position is selected and can be manipulated via real-world hand gestures. To replace the flat tire, we first remove the flat tire from the virtual car, open the virtual trunk, extract the spare tire, and set it on the ground. In Figure 6 (d), both blue hands hold the spare tire and push it closer to the virtual car. The tire is then set upright with the left hand and attached to the appropriate wheel on the virtual car. Thus, a virtual flat tire on a virtual car in a virtual space is replaced with a virtual spare tire from the virtual trunk.

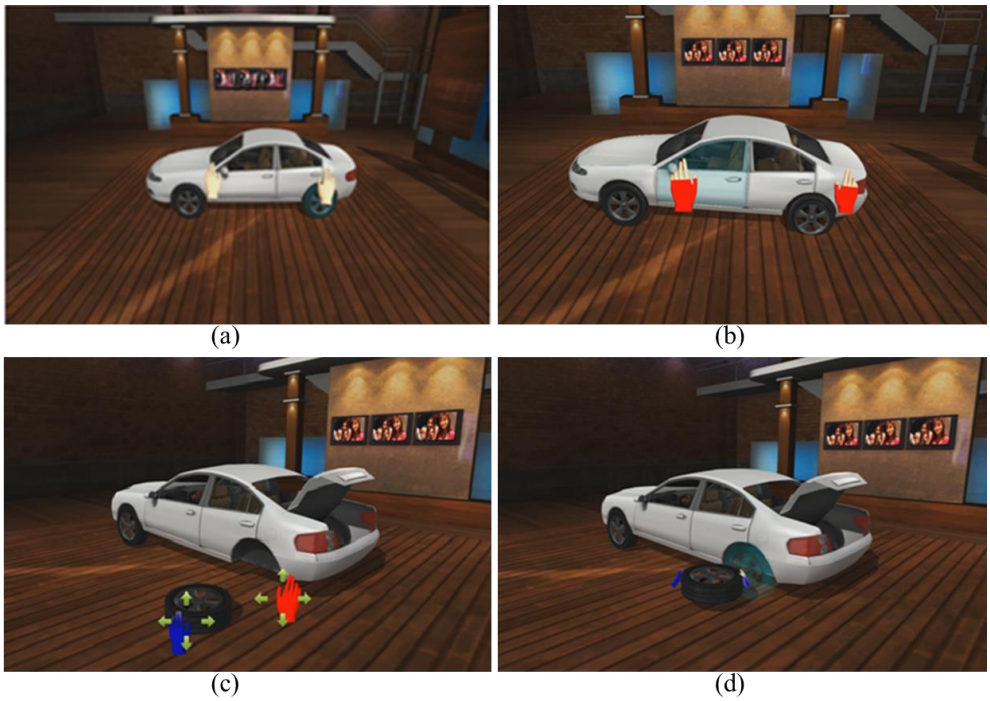

Figure 6. Replacing a tire on a virtual car in a virtual space

Figure 7 shows examples of opening a door on the virtual car and selecting a seat (a) and controlling the audio system while sitting in the seat (b). Figure 8 shows examples of simulated engine disassembly on a virtual car.

(a) (b)

Figure 7. Opening a car door and selecting a seat (a), and controlling the audio system while sitting in the seat (b)

(a) (b) (c)

Figure 8. Disassembling the engine of a virtual car

5. Conclusion

This paper proposed a hand-gesture-based user interface system that enables users to control objects in 3D virtual environments using Microsoft Kinect. The objective was to develop a more accurate and flexible gesture-based virtual object control system by applying a number of filtering operations to a sequence of hand-gesture images captured from Kinect and processing hand gesture events with callback and delegate functions. As an application of the proposed approach, we developed a gesture-based virtual simulation system for vehicle maintenance and demonstrated a number of object-manipulation features, such as opening a car door, selecting a seat and sitting in it, holding a tire with both hands, moving and replacing the tire, controlling the audio system, and disassembling the engine. Our experimental results indicate that the proposed hand-gesture-based user interface could potentially be employed in place of a mouse or keyboard to interact with objects in 3D virtual environments, providing ease of use and natural object control in virtual spaces.

This paper explored a number of gesture-recognition and execution features for more natural and accurate hand-gesture-based interaction with objects in 3D VEs. Nevertheless, the study is ongoing, and more advanced gesture-based control will be investigated in future research by extending the technique from gestures based on linear movements to more general kinds of gestures and by enabling more diversified gesture operations with combinations and sequences of multiple gestures.

Acknowledgements

This research was supported by National Research Foundation of Korea (NRF) grant funded by the ICT standardization program of the Ministry of Knowledge Economy (MKE) and by the Ministry of Education(MOE) and National Research Foundation of Korea(NRF) through the Human Resource Training Projection for Regional Innovation.

References

[1] M. A. Kavakli, "Designing in virtual reality (DesIRe): a gesture-based interface", Proceedings of the 2nd international conference on Digital interactive media in entertainment and arts, **(2007)**, pp. 131-136.
[2] N. H. Dardas and M. Alhaj, "Hand Gesture Interaction with a 3D Virtual Environment", The Research Bulletin of Jordan ACM, vol. 2, no. 3, **(2011)**, pp. 86-94.
[3] Y. Fang, K. Wang, J. Cheng and H. Lu, "A Real-Time Hand Gesture Recognition Method", 2007 IEEE International Conference on Multimedia and Expo, **(2007)**, pp. 995-998.
[4] S. S. Rautaray and A. Agrawal, "Real Time Hand Gesture Recognition Systems for Dynamic Applications", International Journal of UbiComp (IJU), vol. 3, no. 1, **(2012)**, pp. 21-31.
[5] T. Phan, "Using Kinect and OpenNI to Embody an Avatar in Second Life", http://www.youtube.com/watch?v=ehTvtkybubM (http://projects.ict.usc.edu/mxr/) (accessed on March **2013)**
[6] D. Slambekova, "Gaze and Gesture Based Object Interaction in Virtual World", Master thesis, Rochester Institute of Technology. Rochester, New York, **(2011)**.
[7] MIT Media Lab, Depthjs, http://depthjs.media.mit.edu/ (accessed on March **2013).**
[8] Wikipedia, Gesture Recognition, http://en.wikipedia.org/wiki/Gesture_recognition (accessed on March **2013)**.
[9] J. O. Kim, C. Park, J. S. Jeong, N. Baek and K. H. Yoo, "A Gesture Based Camera Controlling Method in the 3D Virtual Space", International Journal of Smart Home, vol. 6, no. 4, **(2012)**, pp. 117-126.
[10] J. Wachs, M. Kolsch, H. Stern and Y. Edan, "Vision-Based Hand-Gesture Applications", Communication of the ACM, vol. 54, no. 2, **(2011)**, pp. 60-71.
[11] Microsoft Corporation, Kinect for Windows & Kinect for Windows SDK Beta 1 Programming Guide Version, http://www.microsoft.com/en-us/kinectforwindows/, Kinect for Windows SDK, http://msdn.microsoft.com/en-us/library/hh855347.aspx (accessed on March **2013)**.
[12] J. S. Jeong, C. Park, S. O. Kwon, J. A. Park, S. A. Kwon, M. S. Im, J. J. Han, J. H. Choi and K. H. Yoo, "Hand gesture user interface for controlling 3D objects in 3D virtual space using Kinect", ICCC 2011, **(2011)**, pp. 105-106.
[13] J. O. Kim, J. S. Jeong, J. A. Park, S. A. Kwon and K. H. Yoo, "Gesture User Interface for Controlling Objects in the 3D Virtual Space", Proceeding of 2012 Korean Computer Graphics Society, **(2012)**, pp. 103-104.
[14] M. Camplani and L. Salgado, "Efficient spatio-temporal hole filling strategy for Kinect depth maps", Proceedings of the Three-Dimensional Image Processing and Applications II, vol. 8290, **(2012)**, pp. 82900E 1-10.

Authors

Jong-Oh Kim

He is a M.S. candidate of department of digital informatics and convergence at Chungbuk National University, Korea. He received the B.S. in Computer Education from Chungbuk National University, Korea in 2011. His research interests include computer graphics, e-leanring, 3D character animation, and multimedia.

Mihye Kim

She received her Ph.D. degree in Computer Science and Engineering from New South Wales University, Sydney, Australia in 2003. She is currently an Associate Professor in the Department of Computer Science Education at Catholic University of Daegu, South Korea. Her research interests include knowledge management and retrieval, computer science education, digital textbooks, and cloud computing.

Kwan-Hee Yoo

He is a professor in the Department of Computer Education and in the Department of information and industrial engineering at Chungbuk National University, Korea. He received his B.S. in Computer Science from Chonbuk National University in 1985, and his M.S. and Ph.D. in computer science from KAIST (Korea Advanced Institute of Science and Technology) in 1988 and 1995, respectively. His research interests include computer graphics, 3D character animation, and dental/medical applications.