Kinect to Architecture

Leroy D'Souza, Isuru Pathirana, Dermott McMeel, Robert Amor
University of Auckland
Private Bag 92019, Auckland, New Zealand
d.mcmeel@auckland.ac.nz, trebor@cs.auckland.ac.nz

***Abstract*—For a client to be able to immerse themselves within an architectural model there needs to be a natural and intuitive interface. A keyboard and mouse based interface can detract from the focus on the space being navigated and therefore a more intuitive approach to such interaction needs to be identified. The Kinect device provides significant recognition ability for whole body gestures and its use was investigated in this project. With a small number of gestures required for navigation an important design criteria is the recognition accuracy and fatigue for prolonged interaction with a model. The paper reports on the Kinect-based navigation system as well as a user trial which identifies the strengths and weaknesses of this approach.**

Keywords-natural user interface; navigation; architecture

I. Introduction

Three-dimensional models play an important role in architectural design since they allow architects to review, communicate and present design proposals to clients. They also give clients an accurate impression of the designed space and its utility. However, navigation of the model using traditional interface devices, such as a keyboard and mouse, inherently limits the naturalness and the speed of interaction with the model.

The objective of this project was to design a Natural User Interface (NUI) that would allow users to navigate through 3D architectural models. This would provide a more interactive environment to conceptualise 3D space. Existing systems such as immersive virtual environments have shown great potential in this area. However, these systems often require users to wear additional devices, thereby compromising the naturalness of the system. The focus of this project is to incorporate the use of hand gestures and body poses to allow users to interact with 3D models.

In recent years many systems have utilised the movement of the human arm, or hand gestures, as a means of interacting with computers. However, several of these systems, such as glove-based devices, compromise convenience by requiring the user to be instrumented with bulky devices [1, 2]. They can also prove to be too expensive for the consumer market [3]. Alternatively, vision-based recognition of hand gestures promises natural and non-obstructive interaction, but can be challenging due to the complexity of the human hand structure and motion.

The recent release of Microsoft's Kinect controller [4] has generated significant interest with its ability to detect human proximity and motion. Consequently, it was decided that the use of the Kinect would be explored for gesture detection. The designed system is a gesture-based interface that can be used by architects to import 3D architectural models, allowing their clients to navigate through them using the Kinect. The system also gives visual feedback to users with the use of an on-screen avatar.

II. Related Work

A. Gesture Recognition

3D gesture recognition in virtual environments has been an important research topic over the years. Researchers have proposed many different methods for resolving this issue [5]. However, most systems use either motion or video sensor-based recognition.

Motion sensor-based gesture recognition utilizes data obtained from devices such as, acceleration measurements from accelerometers and angular velocity from gyroscope sensors [1, 6]. This approach has the advantage of being unaffected by environmental factors such as lighting, but can prove to be cumbersome for the user.

Video sensor-based gesture recognition uses the images from one or more video sensors [7, 8]. It then uses complex image processing algorithms to obtain gesture information. This method is relatively inexpensive, but is mainly used for indoor activities due to its dependency on the level of lighting. The Microsoft Kinect uses the video sensor-based approach. The Kinect has an advantage over traditional video sensor-based systems because it uses infrared light to measure proximity using time-of-flight information. Giving it a depth accuracy to within a few centimeters and allowing it work in low-light environments [9]. The depth data collected by the Kinect has been used to control quadrocopters [10] and perform finger tracking [11]. The Kinect also has a microphone array and a RGB camera, however the focus of the project is on its depth sensors.

B. Natural User Interfaces

The term "Natural User Interface" or NUI describes an interface that focuses on human abilities such as touch, vision, voice, motion and other higher cognitive functions [12]. The main goal of a NUI is to ensure that the system conforms to the users' needs rather than the user having to conform to the system. This essentially means that a successful natural user interface can be described as one where the user takes very little time to transition from a novice to an expert. This is a significant step in the area of Human Computer Interaction because the use of NUIs will avoid current restrictions present in complex interaction with digital objects, such as the one being faced in the field of architecture.

C. Navigation of Virtual Environments

Virtual environments have proved efficient for training in three-dimensional spaces. It is therefore not surprising to find architects using virtual environments to intuitively conceptualise 3D space. Traditionally, virtual environments have required users to familiarise themselves with the complex techniques and equipment required to fully experience the 3D space. These systems can at times prove to be counter-intuitive, as it can take a long time for the user to get accustomed to the system.

Arch-Explore was a similar project designed to provide a natural interface to navigate 3D models using immersive environments [13]. The system allowed users to walk through a given miniature architectural model with the use of a Head Mounted Display (HMD) or a complex immersive environment. The downside to this technique was the fact that users had to wear additional equipment.

III. A GESTURE-BASED DESIGN NAVIGATOR

The main goal of this work was to create a navigation system which non-expert users (e.g., clients of an architect) could utilise to explore a 3D model of a building and its surrounding site. To achieve this we looked for approaches that could emulate the feeling of walking through the 3D model and that would support gestures to drive the navigation. An important aspect of the project was to design navigational interactions which would appear natural for a user. Since naturalness is subjective and unique to each user, research was conducted to find out what factors determined naturalness. It was found to comprise of four major components: accuracy; ease of use; memorability of the gesture; and if the actions were fatiguing.

To enable navigation of a 3D model the core feature of the program is to allow unconstrained movement through the model. So gestures would be required to allow users to perform a forward and backward motion and a way to turn or pan their view. Since most architectural models would be multi-storey structures, the system must also implement gestures for moving up and down through different storeys.

For successful navigation of a 3D environment some form of visual feedback must be provided back to the user to ensure they know how their actions are being translated into the virtual world. This can be done with the use of an avatar as is typically deployed in the majority of first-person interactive games.

As a complete system the program should also provide the user with an option to choose the model that they wish to navigate from a catalogue of designed buildings. Ideally this selection should also happen with the use of gestures to maintain the experience of immersion.

The support of standard 3D model file formats is key to the long-term success of the program, both for loading models into the system and visualising the model in real-time. This would ensure that architects are not inconvenienced by having to convert their designs to a special format solely for the use of this platform, thereby making it both versatile and scalable.

IV. THE KINECT AS A GESTURE DETECTOR

To achieve a natural user interface hand gestures and body poses were incorporated into the program to allow users to experience the architectural space in a 3D model. Gestures were also used to interface with the other aspects of the program such as the start screen and the model selection. As previously mentioned, the Kinect sensor is used to collect depth data in order to perform gesture detection.

A. Gesture Detection

The Microsoft Kinect has been chosen as the video sensor for the system. It was released by Microsoft for the Xbox 360 video game platform in 2010. The Kinect is used to receive hand motion and gesture input from the user.

The Kinect sensor consists of a RGB camera, depth sensor and multi-array microphone, which provide capabilities such as, full-body 3D motion capture, facial recognition and voice recognition. The depth sensor consists of an infrared laser projector combined with a monochrome complementary metal-oxide semiconductor (CMOS) sensor, which captures video data in 3D. The sensing range of the depth sensor can be adjusted.

The depth sensor first illuminates the scene in an infrared depth pattern, which is invisible to the human eye (see Figure 1 for an example of an illuminated scene). The CMOS image sensor reads back the depth pattern, which is now distorted due to the various objects in the scene. This distortion is processed by a System on Chip (SoC) connected to the CMOS image sensor, producing a depth image of the scene. The data is then transmitted to the computer via Universal Serial Bus (USB) interface. Highlights of this method are that it is accurate to within a few centimeters and can work in low-light conditions, as may be found inside a room [9].

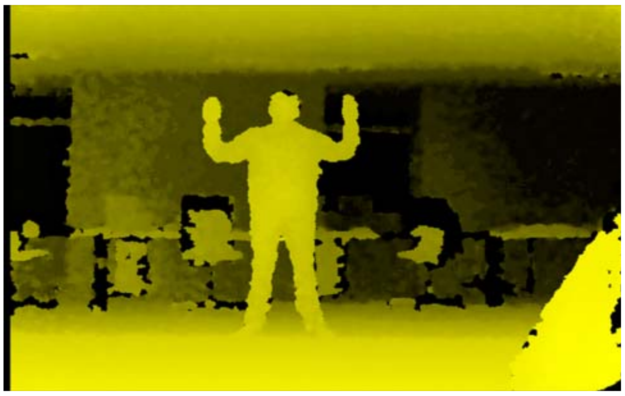

Figure 1. An infrared depth pattern.

B. *Gesture Recognition*

The raw data from the Kinect is obtained via the SensorKinect driver. This raw data is processed to extract the skeletal and gesture data of the user. The OpenNI library [14] is used to interface between SensorKinect and the NITE framework [14], which is used to perform skeletal tracking and gesture detection. The Gesture Manager accesses this skeletal data via OpenNI APIs. OpenNI was chosen as the desired toolkit for this purpose because it provides an abstraction layer allowing rapid and easy development of a natural user interface.

For skeletal tracking, the NITE framework requires the user to first perform a calibration pose, as shown in Figure 1. The user's body proportions are then mapped onto this skeletal model, which consists of 15 different points, to achieve skeletal tracking. NITE uses skeletal tracking to help facilitate gesture detection. For example, to detect a push gesture the motion of the hand point in relation to the torso point would be tracked. When the relative distance between the two points has reached a given limit, a successful push gesture event can be triggered. However NITE is unable to perform finger detection which inherently limits the amount of hand gestures that can be used in this approach. The interaction of the various sub-systems is shown in Figure 2.

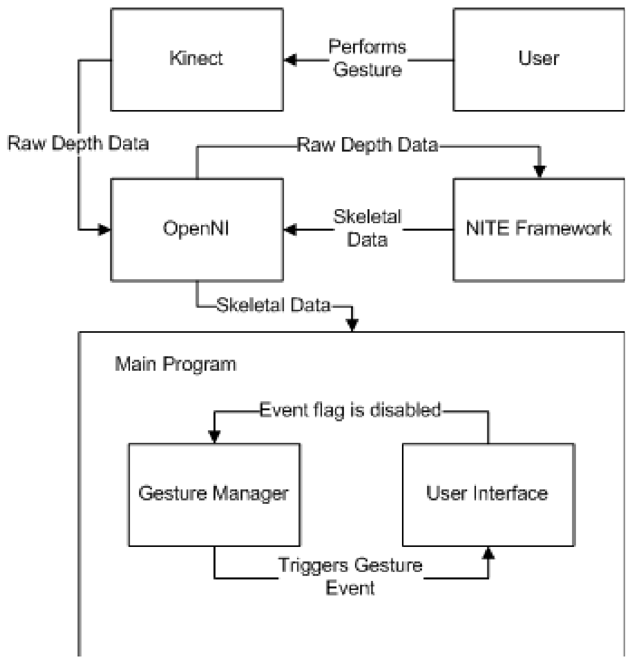

Figure 2. System architecture.

V. DESIGNING GESTURES FOR NAVIGATION

To meet the requirements for natural gestures we aimed for a minimal set of gestures that covered the core requirements of the system for both navigation and control functions. This enabled the choice of gestures which were easily recognisable as distinct by the recognition toolkits, and which mapped to movements that the users perceived as matching the desired functionality. Aiming for a minimal set also helped the memorability aspects with few poses to be learnt by a user. The gestures selected and the range of movement required to trigger a gesture were developed in an iterative manner with a small number of test subjects. For example, in the initial set of gestures the step forward and backward poses were not contemplated, and were incorporated after suggestions by the test subjects. The final set of gestures for the system are shown in Table I.

TABLE I. GESTURES SUPPORTED.

Gesture	**Functionality**
Point forward	Move forward
Step forward	Move forward
Step backward	Move backward
Point up	Move to an upper storey
Point down	Move to a lower storey
Point left/right	Pan camera left/right
Hold-up pose	Calibrate user to system
Push hand forward	Select
Cross arms	Return to starting position
Cross arms (hold for 2 seconds)	Return to model selection

When the user performs a given gesture, it is processed to see if it is one of the gestures implemented in the system. If it is then the gesture is added to a First-In-First-Out (FIFO) buffer. This enables the User Interface to accurately respond to a given sequence of gestures. To detect each of the gestures, both the orientations and positions of the user's hands and feet relative to their torso is tracked. Each gesture requires the user's hand or feet to be in a certain position to be detected.

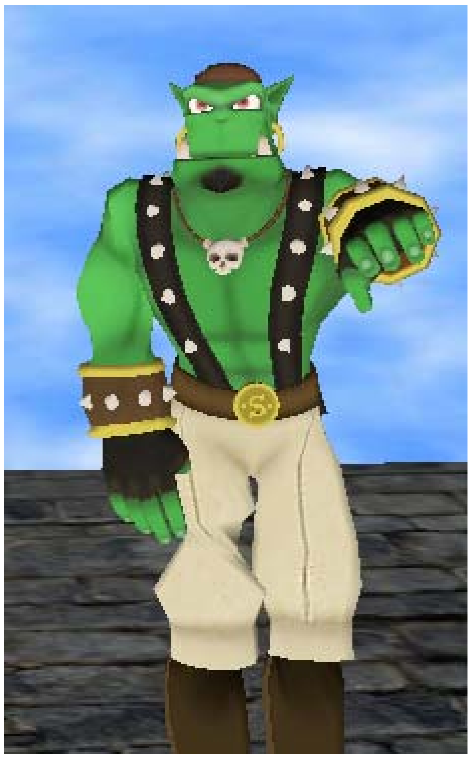

Figure 3. Gesture for moving forward.

The pointing forward gesture (see Figure 3) is detected by measuring the relative distance between the right hand point and the torso in the Z direction i.e. directly in front of the user. This is then compared to a minimum value to trigger a pointing forward gesture. Similarly, the pointing left/right and up/down gestures are detected by measuring the location of the hand point with respect to the torso point in the Z and X (sideways) directions, and Z and Y (up or down) directions respectively. An additional constraint was placed on the down gesture that the user's hand must be placed some distance below and away from the torso. This was done to ensure that the pointing down gesture was not accidentally triggered when the user was in the rest position (with his/her hands to their side). The cross gesture is detected by checking the positions of the hand, elbows and their angular orientation. Finally the push gesture detection is provided by the NITE framework and is detected by sensing a continuous push motion towards the Kinect. The body poses consist of either stepping forward by placing the right foot ahead of the left foot or by stepping backward by placing the right foot behind the left foot. The relative distance between the left and the right feet in the Z direction is then measured and if this is greater than a minimum value, the gesture is triggered.

Initially, the minimum detection values were set to fixed values. However, upon preliminary testing it was found that due to variation in users' heights, the system had trouble detecting some of the gestures. It was then decided that to improve the system's capabilities, the minimum values needed to be dynamically generated. Hence, a preliminary study of 6 users was conducted. The study involved measuring the participants' heights using the Kinect by calculating the difference between the head point and the feet points. They were then asked to perform all the different gestures implemented in the system to the extent which they felt comfortable, for example, pointing left till a point where he/she did not feel any additional strain or discomfort. The ratios of the relative X and Y distances of their hands/feet from their torso to their height were measured for each gesture. These ratios were then averaged to find an overall ratio for each gesture. The gesture detection incorporated these ratios and the height of the current user to dynamically tailor the hand and feet positions for each user.

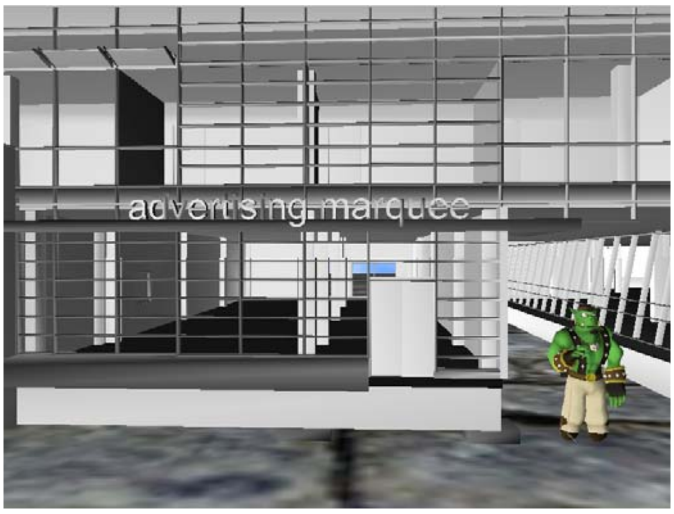

Figure 4. Navigating a 3D model.

VI. User Trial

A user study was conducted to test how intuitive and natural it was for users to navigate 3D architectural models using the designed program. The user study consisted of 7 participants, with all participants being students from the University of Auckland. Six participants had used gesture-based interfaces previously in the context of gaming. The study involved first allowing the participants to become familiar with the interface and try out all the different gestures available. They were then instructed to complete a timed test. To prevent bias, subjects that took part in any preliminary testing were deemed to be unsuitable for this user study.

The test setup required users to collect 8 hoops located at various points in the model (see Figure 5 for an example setup). The time taken to collect all the hoops was recorded. The average response time i.e. the difference between the time when a user saw a given hoop to the time the user collected the hoop was also recorded. To ensure fairness the location of the hoops was randomised for each test, however the relative distance between each hoop was kept the same. The test was then repeated using a keyboard, with the functionality of the system being mapped onto keyboard keys. The times using both the Kinect and the keyboard was analysed.

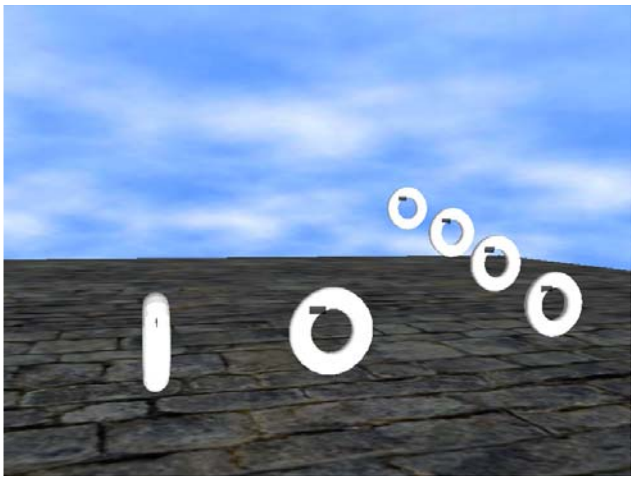

Figure 5. User trial scenario.

The users were then asked to complete a questionnaire about the systems which consisted of two main sections. The first section evaluated each of the gestures in 5 aspects namely, accuracy, ease of use, level of fatigue experienced, how memorable the gestures were and how responsive the system was to a given gesture. This was to determine how natural and intuitive users found the system. The second section required them to complete an evaluation with regards to the system as a whole. Both these sections were quantitative making use of a 5- point Likert Scale, with 5 representing either Excellent or Strongly Agree and 1 either Poor or Strongly Disagree. The participants were given the opportunity to provide qualitative feedback about the feature(s) or gesture(s) they preferred the most and the one they found most difficult to use. For a given gesture, all the scores for a given factor, for example accuracy, were added together and the sum was divided by the maximum total score possible. This was used to calculate a percentage.

In Figures 6 and 7 one can see that the push gesture was rated lowest in terms of accuracy, ease of use, memorability and responsiveness. During the course of the project it was found that in-built gestures, such as Swipe or Push which were provided by NITE, could not be successfully integrated into the program. A possible solution is to design a method to detect these gestures manually. In terms of ease of use the cross gesture had the highest rating and the up gesture was the most accurate.

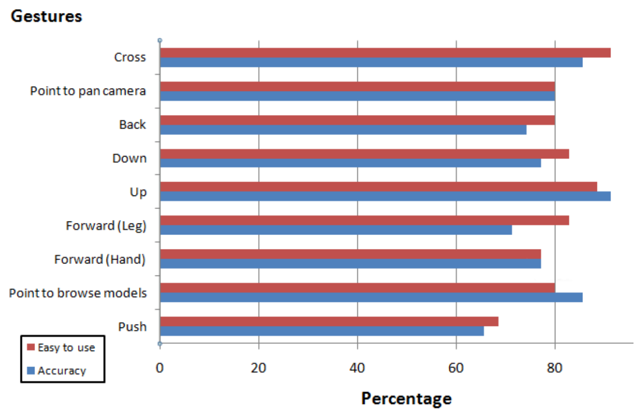

Figure 6. Analysis of ease of use and accuracy of gestures.

It is interesting to note that in Figure 7, pointing left/right to browse through different models was more memorable than using the same gesture to pan the camera. This indicates that the naturalness of the gesture is not simply dictated by the action itself, but rather the context that it is used in. The average accuracy across all the gestures was 79%, the average ease of use was 81%, the average responsiveness was 86% and the average memorability was 89%.

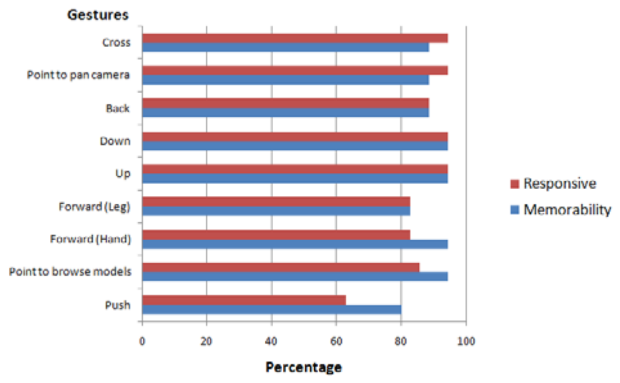

Figure 7. Analysis of responsiveness and memorability of gestures.

The users also rated the gestures on how fatiguing each of the gestures were. The higher the percentage the more fatiguing the gesture. As can be seen in Figure 8, the push and cross gestures caused the least amount of fatigue and the pointing left/right was the most tiring. Most of the users stated that after 8-10 minutes of continual usage they were slightly fatigued. This is not surprising as currently navigation involves continual movement of the right arm.

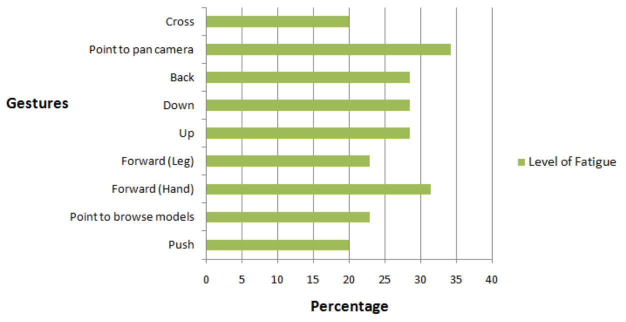

Figure 8. Fatigue level of gestures.

The time taken to collect the hoops is shown in Table II. The average response time with the keyboard was found to be 1.7 seconds and on the Kinect was 4.1 seconds. Both the times to collect the hoops and the average response times are higher on the Kinect than the keyboard. This is to be expected since all the users had far greater prior experience using keyboard interfaces. However, it should be noted that the main aim of the project was not to deploy a Kinect based interface that was faster, but rather to design a more natural interface for users.

TABLE II. TIME TO TRAVERSE THE TRIAL SCENARIO.

Time Taken	**Kinect**	**Keyboard**
Minimum	57s	40s
Average	81s	60s
Maximum	125s	115s

In terms of qualitative feedback, most of the users felt the interface could be improved by incorporating the use of the left hand in gestures. Although the program currently doesn't support this, it can easily be done and was not implemented earlier due to it being deemed a low priority feature. The use of both arms simultaneously for navigation was not implemented, as an assumption was made that requiring both hands to control the system would greatly fatigue the user. Users also found it difficult to have precise control over movements and thought it would be useful to vary the movement speed based on the position of their arms.

VII. CONCLUSIONS AND FUTURE WORK

In this project we successfully designed a natural user interface implementing a range of gestures for navigation of a 3D model. These gestures were able to be recognised by a Microsoft Kinect, coupled with the NITE framework, and support all of the major navigation features necessary to experience a 3D scene. Testing of the system showed that while it might be slower to perform movements, the main characteristics of a natural interface such as accuracy of recognition, ease of use, and memorability of gestures were well met. The use of existing recognition frameworks introduced some limitations include the requirement of performing the calibration pose at the start of the program and the lack of support for finger tracking. This forced us to implement gestures that required fully body or limb movements. These movements, as indicated in the user study, fatigued the user after 8-10 minutes of continual usage. The

Kinect also had difficulty detecting people with jackets or other forms of loose clothing. Rendering a 3D model in real-time is a very resource intensive process, which may be problematic for very large building models, though it can be improved with the use of higher powered computers.

A. Future work

This project investigated the utility of 1st generation consumer level products for gesture recognition and it is expected that the ability of these systems will improve dramatically over the next few years. In future the addition of features such as finger tracking would provide users with a much easier and less fatiguing method of navigation. An investigation can be carried out to see if gestures involving both hands would prove to be natural or very fatiguing. To aid movement through a given model a mini-map should be added to the model navigation screen, including a marker showing the user's current location, as is common in many game systems. At the moment the camera view can only be moved forward, backward and panned sideways. This should be expanded to include a tilt feature, with an appropriate gesture being used to perform this. Further improvements to the current system, such as varying the speed of motion based on extent of hand positions or a tutorial mode to familiarise users with the interface can also be investigated. The other capabilities of the Kinect such as the microphone array and RGB camera could also be investigated as ancillary approaches to control the interface using voice commands or to design a customised avatar of the user.

REFERENCES

[1] J.-H. Kim, N. D. Thang, and T.-S. Kim, “3D hand motion tracking and gesture recognition using a data glove,” in Industrial Electronics, 2009. ISIE 2009. IEEE International Symposium on, July 2009, pp. 1013 –1018.

[2] Y. Han, “A low-cost visual motion data glove as an input device to interpret human hand gestures,” Consumer Electronics, IEEE Transactions on, vol. 56, no. 2, pp. 501 –509, May 2010.

[3] B. Takacs, “How and why affordable virtual reality shapes the future of education,” The International Journal of Virtual Reality, 2008, 7 (1): 53, vol. 66, 2008.

[4] (2011) Microsoft xbox website. [Online]. Available: http://www.xbox.com/en-US/kinect

[5] Z. gang Xu and H. lei Zhu, “Vision-based detection of dynamic gesture,” in Test and Measurement, 2009. ICTM ’09. International Conference on, vol. 1, Dec. 2009, pp. 223 –226.

[6] U.-X. Tan, K. Veluvolu, W. T. Latt, C. Y. Shee, C. Riviere, and W. T. Ang, “Estimating displacement of periodic motion with inertial sensors,” Sensors Journal, IEEE, vol. 8, no. 8, pp. 1385 –1388, Aug. 2008.

[7] M. Hasanuzzaman, V. Ampornaramveth, T. Zhang, M. Bhuiyan, Y. Shirai, and H. Ueno, “Real-time vision-based gesture recognition for human robot interaction,” in Robotics and Biomimetics, 2004. ROBIO 2004. IEEE International Conference on, Aug. 2004, pp. 413 –418.

[8] Y. Liu, L. Tang, K. Song, S. Wang, and J. Lin, “A multicolored vision-based gesture interaction system,” in Advanced Computer Theory and Engineering (ICACTE), 2010 3rd International Conference on, vol. 2, Aug. 2010, pp. V2–281 –V2–284.

[9] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake, “Real-Time Human Pose Recognition in Parts from Single Depth Images,” in IEEE Computer Vision and Pattern Recognition (CVPR), June 2011.

[10] J. Stowers, M. Hayes, and A. Bainbridge-Smith, “Altitude control of a quadrotor helicopter using depth map from microsoft kinect sensor,” in Mechatronics (ICM), 2011 IEEE International Conference on, April 2011, pp. 358 –362.

[11] V. Frati and D. Prattichizzo, “Using kinect for hand tracking and rendering in wearable haptics,” in World Haptics Conference (WHC), 2011 IEEE, June 2011, pp. 317 –321.

[12] W. Liu, “Natural user interface- next mainstream product user interface,” in Computer-Aided Industrial Design Conceptual Design (CAIDCD), 2010 IEEE 11th International Conference on, vol. 1, Nov. 2010, pp. 203 –205.

[13] G. Bruder, F. Steinicke, and K. Hinrichs, “Archexplore: A natural user interface for immersive architectural walkthroughs,” in 3D User Interfaces, 2009. 3DUI 2009. IEEE Symposium on, March 2009, pp. 75 –82.

[14] (2011) OpenNI and NITE libraries. [Online]. Available: http://www.openni.org/