A CAVE BASED 3D IMMERSIVE INTERACTIVE CITY WITH GESTURE INTERFACE

Ziyang Zhang
Ryerson University
350 Victoria Street
Toronto, Ontario
. Canada M5B 2K3
zhangzyster@gmail.com

Tim McInerney
Ryerson University
350 Victoria Street
Toronto, Ontario
Canada M5B 2K3
tmcinern@ryerson.ca

Ning Zhang
Ryerson University
350 Victoria Street
Toronto, Ontario
Canada M5B 2K3
ning.zhang@gmail.com

Ling Guan
Ryerson University
350 Victoria Street
Toronto, Ontario
Canada M5B 2K3
lguan@ee.ryerson.ca

ABSTRACT

3D city models have greatly changed the way we interact with geographic information. However, both the visualization and interaction are limited on conventional 2D displays. This paper presents a system that visualizes 3D cities and supports gesture interaction in a fully immersive Cave Automatic Virtual Environment (CAVE). The proposed system utilizes gestures to control navigation, selection, object manipulation, and menu functions. These functions form the basis for most Virtual Reality (VR) applications, including 3D city model applications such as urban planning, virtual tourism, etc. The use of user gestures provides a more natural interaction in fully immersive VR environments. We also propose the use of pattern recognition methods, specifically a Hidden Markov Model, to facilitate real time dynamic gesture recognition and demonstrate its use for virtual menu control. As a secondary contribution, we present a novel selection method for selecting multiple objects in the VR environment. An informal user study provides evidence that users prefer the natural gestural interface over conventional 3D input device interactions, such as the use of input device buttons. The study also shows a strong preference for the 3D immersive city visualization over conventional 2D displays.

Keywords

3D city, Cave Automatic Virtual Environment (CAVE), interaction design, gesture recognition, Hidden Markov Model (HMM)

1 INTRODUCTION

Three dimensional (3D) city models are now widely used in geographic applications such as Google Earth, to facilitate map exploration, urban planning, virtual tourism, and for many other purposes. The use of 3D urban data results in a more realistic visual experience for a user and has also greatly changed the way users interact with these applications. For example, in traditional 2D map-based applications, users can only move the map in 2 directions. Applications using 3D data, on the other hand, enable users to control the viewing position and direction in free space, generating unique and often insightful viewpoints.

However, despite the fact that various 3D navigation/manipulation methods have been developed for a mouse and keyboard or multi-touch screens, there are still limitations when interacting with 3D objects using these 2D input devices. Simply controlling the viewpoint involves 6 Degrees of Freedom (DOF), let alone adjusting a virtual lens or manipulating virtual objects. Learning the mapping from 2D input actions to 3D manipulations is arduous and has hindered novice and expert users alike from fully harnessing the power of interactive 3D VR applications.

The last two decades have witnessed the development of increasingly wider screens and more immersive Virtual Reality (VR) systems. Immersive systems, such as a head mounted display (HMD), a Cave Automatic Virtual Environment (CAVE) [Cru92a] and various multi-screen or curved-screen systems [Maj13a], have a distinct advantage over traditional 2D displays for many applications. Furthermore, with the development of various tracking technologies, movements of the user's body can be fed into 3D applications as inputs, opening up exciting new possibilities for Human Computer Interaction (HCI) by significantly adding to the feeling of "presence" in the virtual scene. The design of 3D user interfaces (3DUI) has been studied for decades since the beginning of VR [Bow01a]. Interactions in a 3D virtual environment usually fall into several categories: navigation, selection, manipulation, and system control, and various interaction techniques have been proposed for each category. Bowman *et al.* [Bow06a] argued in 2006 that after the 1990s' invention of basic 3D interaction techniques, research in 3DUI should focus on more application and task specific techniques and adapt to the ongoing trend of new large area tracking and display technologies. As a result, there has been considerable and recent research work that attempts to bring VR and 3D user interfaces together to create more effective applications, such as 3D medical data visualization and various areas of design, such as mechanical design, building and architecture design [Kan12a], and interior design [Nan13a].

However, the visualization of, and interaction with, massive 3D city data in a VR system has not been fully studied. One important reason of this lack of research may be attributed to the difficulty of acquiring 3D city models. This problem is being resolved by the advancement in semi-automatic and automatic methods of generating 3D city models, from both the computer vision and photogrammetry communities [Fru03a][Zhu09a]. Commercial applications like Google Earth have gathered an enormous amount of 3D city data through the contribution of 3D modelers. With a long term goal of creating a fully immersive interactive 3D city planning system, the work described in this paper focuses on interaction design in a virtual city scenario. Effective, natural interaction is an integral part of this goal. As part of our interaction work, we propose a novel paint-to-select technique for multiple objects selection in order to simplify and make more efficient the subsequent manipulation of multiple virtual buildings. In order to achieve natural interaction within the system, the interface is predominantly controlled by user gestures, from simple direct manipulation gestures for scene navigation and object selection, to more complete gestures (e.g. a circular hand motion) for menu control and other specialized purposes. We apply a Hidden Markov Model (HMM) to recognize these more complete 3D dynamic gestures. To the best of our knowledge, this is among the first systems that use a 3D signal based HMM to assist in dynamic gesture recognition in a VR environment.

We evaluate our system with a user study, both from the gesture recognition side and the user experience side. The use of the advanced HMM pattern recognition algorithm leads to a good recognition result of the system control gestures. Based on evidence from the user study, users prefer our natural interaction interface and immersive 3D city over traditional visualization on 2D screens and traditional input device-based interactions.

2 RELATED WORK

Immersive VR systems have been used to facilitate design in various application areas. Kang *et al.* developed middleware that connects a CAVE-like 3-screen immersive VR system with Autodesk Navisworks, a popular building design and simulation software under the Building Information Model (BIM) standard. The feeling of "presence" greatly improved the preview of a building being designed, which in turn helps the designer and planner make better decisions. Nan *et al.* proposed a virtual design system for interior design in a CAVE which uses 2-finger gestures to rotate, translate and scale virtual objects. This manipulation technique resembles multi-touch gestures in today's smartphones and tablets to manipulate images.

Visualization of 3D city data has been used as a decision support tool for urban planners. Isaacs *et al.* [Isa11a] developed a 3D virtual city application using stereo screens to visualize an urban sustainability simulation. However their work focus on the visualization side and lacks the ability to interact with the virtual city using a natural 3D interface.

HMM has been used widely in the classification of stochastic signals such as human voice and hand gesture. Schlomer *et al.* [Sch08a] proposed a HMM based gesture interface for media browsing, which recognizes 5 gestures from the acceleration sensor data outputted from a Wii controller. Rigoll *et al.* [Rig97a] developed a real time vision based HMM gesture recognition system, which detects a hand from camera images and uses the 2D hand location as input to the classifier. Their system achieved approximately 90% accuracy for 24 gestures that involve intense hand movement.

The system described in this paper is designed for a novel interactive virtual city application implemented in the fully immersive CAVE environment. It utilizes some of the interaction techniques developed by previous researchers [Bow01a][Nan13a], such as the point selection and gesture controlled object rotation. A gesture-driven interaction framework is proposed, and is supported by an HMM gesture recognition system, which recognizes 3D trajectories of hand movements in real time. In addition, a novel 3D "paint-to-select" technique is proposed and implemented.

3 SYSTEM OVERVIEW

As illustrated in Fig. 1, our CAVE consists of 4 wall size displays, each driven by a stereo projector and a workstation PC. These workstations, together with a server node, form a graphics cluster that runs the application. Optical tracking cameras emit infrared light and capture the light reflected by markers worn on the user's head and hands. A tracking server collects these captured images and outputs the location and orientation of marker sets to the user application. To ensure real time response, a separate server continuously monitors the tracking result for gesture recognition, and sends out a trigger signal once a predefined gesture is performed.

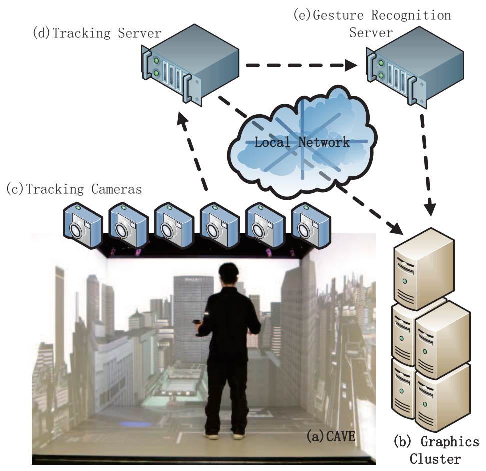


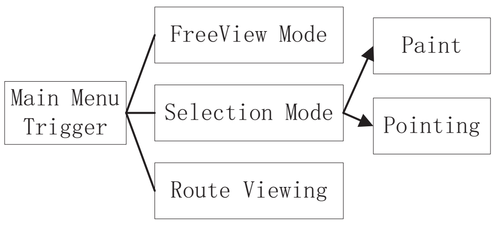
Figure 1: Overview of the implementation in CAVE.

We have constructed our virtual city application on top of Open Scene Graph (OSG), a popular cross platform visualization toolkit that supports a wide range of 3D data formats. A highly configurable VR toolkit, VR Juggler, is used to direct the visualization into the CAVE screens. The use of VR Juggler supports the portability to various other display systems, as well as a traditional desktop monitor. Fig. 1(a) shows a user experiencing the virtual city in the CAVE.

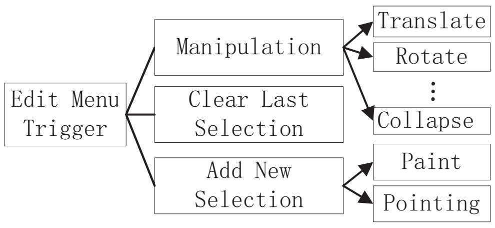
4 SYSTEM FUNCTIONS

Considering that the most significant advantage of the CAVE system is its immersive visualization capability, we focus the design of our user interface on controlling the viewing of the virtual scene, while also adding some ability to modify parts of the scene. In particular, in a 3D virtual city, users typically want to navigate freely around the city, or view the city scene along an interactively defined route (i.e. navigate along the route). In addition, city planners may want to select and modify/manipulate single buildings or several buildings. The user may also want to view an area from a specific view point. However, if there are large buildings in between occluding the target area, the user may want to “collapse” the buildings so that they are no longer visible.

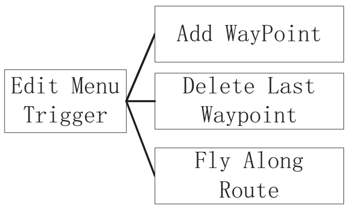
We have therefore defined 3 basic interaction modes that support the navigation and selection/manipulation system functions: FreeView, Selection, and RouteView. In order to keep the interactions as simple and natural as possible, we organize all the system functions into a menu structure. The interaction with the menu will be introduced in section 5.1. The current mode can be switched in the menu. There is also an Edit menu for Selection and RouteView modes, which controls the behavior of the corresponding mode. The menu structure is illustrated in Fig. 2. Details about interactions in this framework will be given in section 5.



(a) Main Menu



(b) Edit Menu in Selection mode



(c) Edit Menu in Route Viewing mode

Figure 2: Menu structure.

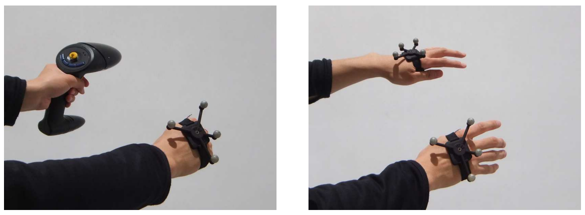
5 GESTURE-BASED INTERACTION

One basic purpose of Virtual Reality is to have an experience that is as real as possible. Therefore, it is important to minimize the distractions from other parts of the system, such as input. Supported by the tracking ability in the CAVE, we have developed a predominantly gesture based interface in an effort to make the interaction as natural as possible. However, although using gestures as commands can be effective in 3D environments, if the number of commands becomes large, it becomes more difficult for users to remember the different gestures. Speech interaction has also become more popular recently. However, speech has not been used to fully control complex systems. One disadvantage of voice commands is privacy issues. The result is the use of gesture and voice commands is often limited to helping navigate through a menu. Other tools, such as a tablet, have also been used to help control VR applications [Med13a] but may distract from the idea of natural interaction.

In our gesture based interface, both hands of the user are tracked. As shown in Fig. 3(a), one hand is tracked by a set of markers worn on the back of the hand, the other is tracked by the internal markers of a remote controller (which is usually called *wand* in VR systems). The wand is used instead of another marker set (as in

Fig. 3(b)) because it has buttons and a joystick on it. As mentioned above, based on our experience, users are often unable to remember all the gestures in a solely gesture controlled environment. Thus, we believe a better solution is to use as few gestures as possible, and complement them with wand buttons. Wand buttons are familiar to users from their use of a computer mouse. To make the interaction simple and easy, the number of buttons used should also be small. In this work, only 2 buttons and 3 dynamic gestures are used to control all functions.

To clarify the terminology used in this paper, a *dynamic gesture* here is defined as a movement pattern of a body part, such as a circular hand motion. Specifically, we use the trajectory of the user's tracked hand position as the input to dynamic gesture recognition. A *static gesture*, on the other hand, is defined as a still posture. These dynamic and static gestures are commonly referred to as *offline* gestures and need to be processed and recognized before the corresponding command is triggered. However, when manipulating a virtual object, the current location of the user's body can directly be used to make changes, similar to those described in [Nan13a] and similar to the multi-touch control of an image on a tablet or smartphone. These simple direct manipulation gestures are commonly referred to as *online* gestures.



(a) Hand and wand (b) 2 hands

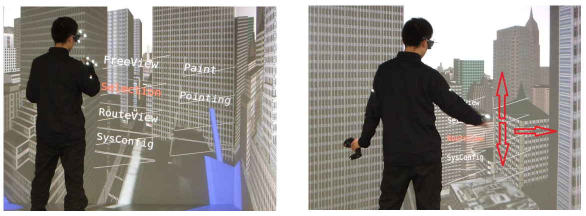
Figure 3: Input devices for gesture interaction.

5.1 Menu Control Interaction

The Main menu and Edit menu can be triggered from 2 different dynamic gestures, which will be introduced in section 6. After the menu is triggered, a list of menu items is shown floating in front of the user.

Two kinds of menu navigation techniques are tested in our system. The first is a touch-based technique, in which the user uses the virtual representation of his/her finger to "touch" a menu item in order to select it. Fig. 4(a) shows a picture of a user touching a menu item.

The second one is a sliding based technique where the user's hand moves up or down to highlight different menu items. After the target item has been highlighted, the user performs a dynamic gesture (moving the hand straight across to the right - see section 6) to select the



(a) Touch menu (b) Sliding Menu

Figure 4: Two kinds of menu interaction.

item. Fig. 4(b) illustrates how the sliding interaction is performed.

5.2 Freeview Navigation

The freeview navigation mode supports scaling, rotating and moving interactions in the 3D virtual city. This set of basic navigation interactions can be performed at any time in the application, even in selection mode and route view mode. The reasoning behind this design decision is that 3D city data is usually very large and the user may not be able to find the area of interest when in selection/routeview mode. In this case, a scaling down, moving and scaling up interaction sequence (i.e. pan and zoom of the entire scene) is required. This interaction sequence resembles the Worlds in Miniature (WIM) technique [Sto95a][Bow06a] in VR navigation, which manipulates a small version of much larger scene to navigate in the large scene.

In order to have access to the freeview navigation functions from anywhere in the application, a wand button is reserved to trigger the freeview navigation mode. Triggering by a dynamic gesture is also an option. However, gesture triggering is less robust than a button trigger. Based on our experience, users become more frustrated upon failing to trigger a navigation than upon failing to trigger a menu, since navigation is performed far more often in a virtual city application. After triggering the freeview navigation, online gesture control takes place. Similar to manipulating images on multi-touch screens, the positions of both hands are used to make changes to the scene, as shown in Fig. 5. Suppose $\mathbf{P}_{L0}$ and $\mathbf{P}_{R0}$ denote the position of left and right hands at triggering time t_0, respectively; $\mathbf{P}_L$ and $\mathbf{P}_R$ represent the corresponding position at current time t.

- **Rotation** The rotation of the virtual city can be represented by the rotation from vector $\mathbf{v}_0$ to $\mathbf{v}$, where

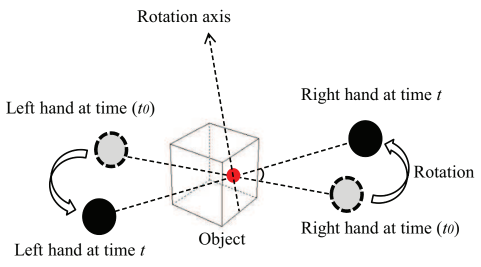
$$\begin{aligned} \mathbf{v}_0 &= \mathbf{P}_{R0} - \mathbf{P}_{L0} \\ \mathbf{v} &= \mathbf{P}_R - \mathbf{P}_L \end{aligned} \tag{1}$$

- **Scaling** Scaling factor of the virtual city is given by:

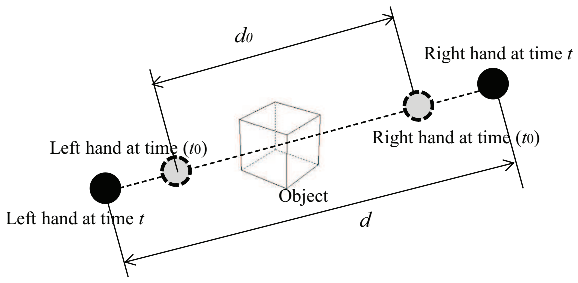
$$s = \frac{|\mathbf{v}|}{|\mathbf{v}_0|} \tag{2}$$

- **Moving** The translation can be represented by:

$$\mathbf{t} = \frac{\mathbf{P}_R + \mathbf{P}_L}{2} - \frac{\mathbf{P}_{R0} + \mathbf{P}_{L0}}{2} \tag{3}$$



(a) Rotation



(b) Scaling

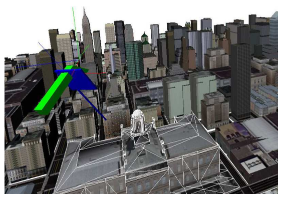
Figure 5: Illustration of rotation and scaling.

Rotation, scaling and moving can be performed at the same time, resulting in smoother and faster scene navigation. The user presses the wand button to trigger a freeview navigation, holds the button until the desired target area is reached, and releases the button to finalize the viewpoint change.

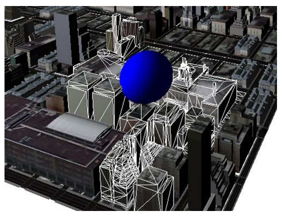
5.3 Object Selection and Manipulation

Two selection techniques are implemented in selection mode. One is the classic point-to-select metaphor (i.e. ray-casting selection) [Bow06a]. When the selection triggering wand button is pressed, a ray is cast from the viewer location, in the user pointed direction. The first object intersected is selected. This technique is useful when a single building close to the user needs to be selected.

However, pointing selection is often not efficient for selecting multiple objects. In order to address this problem, we propose the 3D paint-to-select technique. The concept of "painting", popular in 2D drawing and photo editing software, is familiar and simple. In the 3D CAVE environment, the paint "brush" can be represented with a 3D object of arbitrary shape, although for most applications a simple shape such as a sphere or a cube is used. All scene objects that intersect the 3D brush are selected between the triggering of the selection mode with the wand button, the movement of the hands controlling the position of the brush, and the final release of the button. In our implementation, a simple sphere brush is used, as shown in Fig. 6(b). Screen shots of both pointing selection and paint selection are given in Fig. 6. Finally, the user can use the edit menu to add new objects to the selected set, and manipulate the selected objects in various ways. The rotation, scaling and translation manipulation is done in the same way as in section 5.2.



(a) Point-to-select



(b) paint-to-select

Figure 6: Two kinds of selection techniques.

6 HMM GESTURE RECOGNITION FOR MENU CONTROL

Because of its ability to handle stochastic signals, a Hidden Markov Model (HMM) has been widely used in speech recognition and gesture recognition, and has achieved good results [Rab89a]. A HMM can be fully represented by a parameter set λ:

$$\lambda = (A, B, \pi) \tag{4}$$

where A stands for the transition probability matrix, B represents the observation probability matrix, and π is the initial state probability. Given an existing observation sequence O, a maximum likelihood estimation techniques such as the *Baum-Welch method* can be used to find a model parameter λ that maximizes $P(O|\lambda)$. Given a model parameter, the probability of a sequence belonging to that model, $P(O|\lambda)$, can be found via a *Forward-Backward Procedure*, which is then used to make a recognition decision.

In the CAVE, an optical tracking system constantly updates the 6DOF position and orientation of markers on the user's head and hand. However, due to the user's random standing position and the random direction the user is facing, both position and speed information need to be normalized into the user's local coordinate system before any recognition can be performed. We use the user's head as the reference coordinates. Then the normalized hand position $\mathbf{P}$ is given by:

$$\mathbf{P} = \mathbf{P}_h * \mathbf{M} - \mathbf{P}_r, \tag{5}$$

where $\mathbf{P}_h = (x_h, y_h, z_h)$ is the position of user's hand in global coordinates as returned by the tracking system, and $\mathbf{M}$ is a rotation matrix with angle θ_y rotation around the vertical (in our case, y) axis. The angle θ_y can be found by:

$$\theta_y = \arctan \frac{\sin\psi \sin\phi + \cos\phi \cos\psi \sin\theta}{\cos\psi \cos\theta}, \tag{6}$$

where (ϕ, θ, ψ) and $\mathbf{P}_r = (x_r, y_r, z_r)$ are the orientation Euler angle and position of the reference coordinates (in this case, the user's head), respectively. We use the rotation around the vertical axis only in order to eliminate the influence of the user lowering or shaking his/her head (which are rotations around horizontal axes). The speed of the user's hand is then calculated on a frame by frame basis. The speed data is quantized using k-means clustering, recorded in a memory buffer, and then fed into the trained HMMs. Each model will output the probability of the current sequence belonging to that model. These probabilities are then selected by a threshold and the class with the highest probability is outputted as the recognition result. Fig. 7 shows the framework of the gesture recognition system.

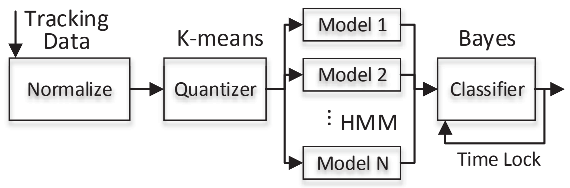


Figure 7: Gesture Recognizor scheme.

In a real time implementation, the recognizer often outputs the same class during a period when the performance of a gesture is about to end. Though correct, we still want just one trigger signal for each gesture. To resolve this problem, a lock mechanism is implemented to keep the recognizer from outputting the same class within a period which is roughly half the length of the gesture duration.

7 EXPERIMENT RESULTS

7.1 Isolated Gesture Recognition Test

In our prototype virtual city application, there are currently only 3 gestures that need to be recognized (2 gestures for triggering the main menu and edit menu, and 1 gesture for the sliding selection of menu items). However, to test performance and leave room for updates to the application and the use of gesture recognition in other VR applications, 7 gestures are defined for testing (Fig. 8).

Based on the contribution of 5 participants, a database containing 875 samples was collected. Among the participants there are 4 males and 1 female, and each performed 25 samples for each gesture. The database was then used to test the gesture recognition framework. Ten samples are randomly selected for each gesture as training data, while the remaining 115 samples are used as testing data. Under this testing scheme, we have an average recognition rate of 96.8 %. The confusion matrix is shown in Fig. 9.

In Fig. 9, each column represents the recognition rate of the corresponding gesture and the rate of incorrect

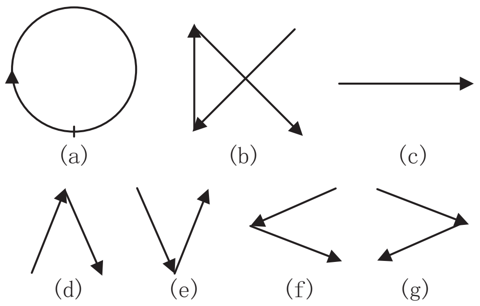


Figure 8: Definition of controlling gestures: (a) Circle; (b) Crossing; (c) Sliding; (d)-(g) Arrows (reserved for future development).

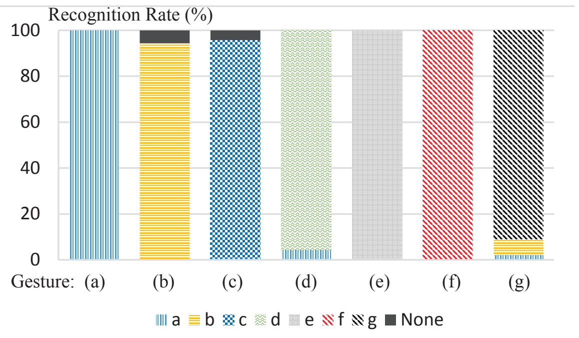


Figure 9: Graphical demonstration of confusion matrix in the isolated recognition test.

recognition as other gestures, while the black color represents recognizing no gestures (i.e. a miss). For example, the figure shows gesture (a) is recognized 100% correctly, while around 95% of gesture (b) is recognized correctly, with a 5% miss rate.

We can see from the result that each of these gestures is well recognized. The crossing (b) and sliding (c) gestures are sometimes not recognized, while the up arrow (d) and right arrow (g) are occasionally mis-classified as other gestures. This may be due to the similarity among gesture definitions. For example, the first half of the right arrow resembles the sliding gesture.

7.2 User Preference Study

To evaluate the user experience of the proposed virtual city system, a comparison interaction set is developed that uses conventional methods, based on wand buttons, for menu triggering and navigation. Since there are a limited number of buttons, 2 of them serve multiple functions that will not be used at the same time. The 3 interaction sets used in the user study are illustrated in Table 1. Note that the 2 selection techniques described in section 5.3 are not compared since they are suitable for different scenarios and are complementary to each other.

6 novice users took part in the study. There is an introduction session first to show a user how to use the system and the interaction techniques. Each of them is

Interaction	(Proposed) Set 1	(Proposed) Set 2	(Compare) Set 3
Main Menu Trigger	Gesture a	Gesture a	Button 3
Edit Menu Trigger	Gesture b	Gesture b	Button 4
Menu Navigation	Sliding	Touch	Button 1,2
Menu Selection	Gesture c	Touch	Button 5
FreeView Navigation Trigger: Button 1			
Selection Trigger: Button 2			
Set Waypoint Trigger: Button 2			

Table 1: Summary of 3 Interaction Sets

	Percentage of users preferring		
	Set 1	Set 2	Set 3
Compare set 1&2	66.67 %	33.36%	
Compare set 1&3	83.36%		16.67%
Compare set 2&3	100%		0%

Table 2: User Preference Study Result

allowed to play around for 5 minutes to become familiar with the interactions. Then they are asked to complete 2 tasks, using each of the 3 interaction sets. The first task is to navigate to a specific location and select 3 buildings in that area. The second task is to select 4 way points along a given street and perform a "Route View" along that street. In these 2 tasks users will perform navigation, selection and menu interaction multiple times. Finally, after finishing the tasks, each user is asked to choose a preference for the comparison of each pair of interaction sets. The result is listed in Table 2.

It is clear that users prefer gesture interactions instead of button interaction when controlling menus. When asked why, the main reasons for this preference include the difficulty in remembering the function of each wand button and the simplicity of the gesture interface. We cannot see a clear preference between the sliding and touch based menu control. However, some users report a tired arm when raising their hand to touch the virtual menu item. Also, users complain about the occasional missing of gesture recognition, which diminishes the gesture interaction experience as a whole.

In addition, users report a great preference to the CAVE based visualization of a 3D city over traditional 2D screens, such as computer monitors and TVs, due to its fully immersive experience and the feeling of presence in the virtual scene.

8 SUMMARY AND DISCUSSION

In this paper, we presented an implementation framework and interaction techniques for a prototype 3D virtual city application. Compared to existing systems, our application focuses on supporting 3D city visualization using a fully immersive interactive VR technique. We proposed a gesture based interface for navigation, selection, object manipulation, and menu interface control. Gesture interaction serves as a novel and more natural interaction mechanism within a VR environment. We also explored the use of pattern recognition methods, specifically a HMM, to recognize (in real time) predefined dynamic gestures, currently used for 3D menu triggering and controlling in the virtual city application. An informal user study shows preference of the gesture based menu interaction over traditional button interactions. Finally, we also presented a novel 3D paint-to-select technique that supports the efficient selection of multiple virtual objects.

In addition, from the user study, users overwhelmingly stated their preference for the fully immersive CAVE-based city visualization over traditional displays, and that 3D interactions using gestures tend to be more natural and simple than interactions controlled using buttons. In the future, we plan to continue improving the virtual city in the CAVE, adding more functionality, such as the display of various information layers, integration of geo-codes, etc., and exploring interaction techniques specifically suitable for these new functions. This added functionality will further our long term goal of using the virtual city system as a platform for urban planning, virtual tourism, and heritage preservation applications, and for many other purposes.

9 REFERENCES

[Bow06a] Bowman, D. A., Chen, J., Wingrave, C. A., et al. New directions in 3d user interfaces. *The International Journal of Virtual Reality 5*, 2 (2006), 3–14.

[Bow01a] Bowman, D. A., Kruijff, E., LaViola Jr, J. J., and Poupyrev, I. An introduction to 3-d user interface design. *Presence: Teleoperators and virtual environments 10*, 1 (2001), 96–108.

[Cru92a] Cruz-Neira, C., Sandin, D. J., DeFanti, T. A., Kenyon, R. V., and Hart, J. C. The cave: audio visual experience automatic virtual environment. *Communications of the ACM 35*, 6 (1992), 64–72.

[Fru03a] Frueh, C., and Zakhor, A. Constructing 3d city models by merging ground-based and airborne views. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on* (2003), vol. 2, IEEE, pp. II–562.

[Isa11a] Isaacs, J. P., Gilmour, D. J., Blackwood, D. J., and Falconer, R. E. Immersive and non immersive 3d virtual city: decision support tool for urban sustainability. *Journal of Information Technology in Construction* (2011).

[Kan12a] Kang, J., Ganapathi, A., and Nseir, H. Computer aided immersive virtual environment for

BIM. In *International Conference on Computing in Civil and Building Engineering* (2012).

[Maj13a] Majumder, A., and Sajadi, B. Large area displays: The changing face of visualization. *Computer 46*, 5 (2013), 26–33.

[Med13a] Medeiros, D., Teixeira, L., Carvalho, F., Santos, I., and Raposo, A. A tablet-based 3d interaction tool for virtual engineering environments. In *Proceedings of the 12th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and Its Applications in Industry* (2013), ACM, pp. 211–218.

[Nan13a] Nan, X., Zhang, Z., Zhang, N., Guo, F., He, Y., and Guan, L. vDesign: Toward image segmentation and composition in cave using finger interactions. In *Signal and Information Processing (ChinaSIP), 2013 IEEE China Summit & International Conference on* (2013), IEEE, pp. 461–465.

[Rab89a] Rabiner, L. R. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE 77*, 2 (1989), 257–286.

[Rig97a] Rigoll, G., Kosmala, A., Eickeler, S. High Performance Real-time Gesture Recognition Using Hidden Markov Models. *International Gesture Workshop* Bielefeld, Germany, September 1997.

[Sch08a] Schlomer, T., Poppinga, B., et. al. Gesture Recognition with a Wii Controller. In*Proceedings of the Second International Conference on Tangible and Embedded Interaction* (TEI08), Feb. 2008.

[Sto95a] Stoakley, R., Conway, M. J., and Pausch, R. Virtual reality on a wim: interactive worlds in miniature. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (1995), pp. 265–272.

[Zhu09a] Zhu, Q., Hu, M., Zhang, Y., and Du, Z. Research and practice in three-dimensional city modeling. *Geo-spatial Information Science 12*, 1 (2009), 18–24.