

A CAVE BASED 3D IMMERSIVE INTERACTIVE CITY WITH GESTURE INTERFACE

by

Ziyang Zhang

B. Eng., Zhengzhou University,

China, 2012

A thesis

presented to Ryerson University

in partial fulfillment of the

requirements for the degree of

Master of Applied Science

in the Program of

Electrical and Computer Engineering

Toronto, Ontario, Canada, 2014

©Ziyang Zhang 2014

AUTHOR'S DECLARATION FOR ELECTRONIC SUBMISSION OF A THESIS

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I authorize Ryerson University to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize Ryerson University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

I understand that my thesis may be made electronically available to the public.

A CAVE Based 3D Immersive Interactive City with Gesture Interface

Master of Applied Science 2014

Ziyang Zhang

Electrical and Computer Engineering

Ryerson University

Abstract

This thesis presents a system that visualizes 3D city data and supports gesture interactions in a fully immersive Cave Automatic Virtual Environment (CAVE). To facilitate more natural interactions in this immersive virtual city, novel techniques are proposed for operations such as object selection, object manipulation, navigation and menu control. These operations form a basis of interactions for most Virtual Reality (VR) applications. The proposed techniques are predominantly controlled using gestures. We also propose the use of pattern recognition methods, specifically a Hidden Markov Model, to support real time dynamic gesture recognition and demonstrate its use for menu control in VR applications. Qualitative and quantitative user studies are conducted to evaluate the proposed techniques. The results of the user studies demonstrate that the interaction techniques for object selection and manipulation are measurably better than traditional techniques. The results also show that the proposed gesture based navigation and menu control techniques are preferred by experienced users. These findings can guide future user interface design in immersive environments.

Acknowledgements

I would like to express my sincere thanks to my supervisor Dr. Ling Guan and my co-supervisor Dr. Tim McInerney, for their support and guidance during my Master study, and for their valuable suggestions and inspirations of new research ideas, as well as the help they gave in shaping this thesis.

Many thanks also go to all the members of Ryerson Multimedia Laboratory. Their kind help and the friendly environment in the lab make my two-year study here an enjoyable and memorable time. Thank Mr. Jordan Sparks for his insightful comments on user interface design. Thank Dr. Yifeng He, Dr. Ning Zhang and Mr. Xiaoming Nan for coordinating one of the user studies. Thank all the colleagues and friends who volunteered in the user studies.

Finally I would like to thank my parents. The completion of this work would not be possible without their constant love, support, and faith in me.

Contents

<i>Declaration</i>	i
<i>Abstract</i>	ii
<i>Acknowledgements</i>	iii
<i>List of Tables</i>	vii
<i>List of Figures</i>	ix
<i>List of Appendices</i>	xi
1 Introduction	1
1.1 Background	1
1.2 Purpose and Scope of This Thesis	2
1.3 Contributions	2
1.4 Overview of This Thesis	4
2 Literature Review	5
2.1 VR Systems	5
2.1.1 Tracking in VR systems	6
2.2 3D Urban Data	8
2.3 HCI in VR	9
2.3.1 Input Hardware	9
2.3.2 Interaction Techniques	10
3 Methodology	12
3.1 System Configuration	13
3.1.1 The CAVE	13
3.1.2 Software Configuration	14
3.1.3 Input Devices	14
3.2 HMM Realtime Gesture Recognition	15
3.2.1 Normalization and Quantization	16

3.2.2	Hidden Markov Model and Classification	18
3.3	System Control	20
3.3.1	Menu Control by Button	21
3.3.2	Menu Control by Gestures	22
3.4	Navigation	24
3.4.1	Joystick	24
3.4.2	Posture Triggered Flying	25
3.4.3	Waving Hand	25
3.5	Selection	26
3.5.1	Ray Casting Selection	27
3.5.2	Paint Selection	28
3.5.3	Stretch Selection	28
3.6	Manipulation	28
3.6.1	Wand Manipulation	29
3.6.2	Two-Handed Gestures	30
3.7	Visibility of Hidden Buildings	31
3.7.1	Collapse	32
3.7.2	Semi-Transparent	32
3.7.3	Wireframe	33
3.8	Adding Constraints	33
3.8.1	Map Data Structure	34
3.8.2	Linkage Between Map Data and 3D City Data	36
3.8.3	Path Finding by Dijkstra's Algorithm	37
3.8.4	High Level Interactions Enabled by Map	39
3.9	Summary	40
4	Experiments & Results	42
4.1	Gesture Recognition	43
4.1.1	Gesture Database	43
4.1.2	Gesture Recognition Results	43
4.2	3D City Data	44
4.3	Empirical User Study	44
4.3.1	User Study Design	46
4.3.2	Results on Menu Control	49
4.3.3	Realtime Gesture Recognition Records	50
4.3.4	Results on Navigation	52

4.3.5	Results on Selection	53
4.3.6	Results on Visibility	54
4.4	Quantitative Study	55
4.4.1	User Study Design	55
4.4.2	Results for Manipulation	57
4.5	Summary	58
5	Prototype Applications	60
5.1	Urban Plannning	61
5.2	Virtual Touring	61
6	Conclusions and Discussions	63
6.1	Summary of This Thesis	63
6.2	UI Design	64
6.3	Possible Future Work	65
	References	84

List of Tables

2.1	Summary of immersion components for several VR or display systems.	6
3.1	Flystick button assignment for menu control.	21
3.2	Example iterations of Dijkstra’s algorithm.	38
4.1	Interaction techniques for comparison.	46
4.2	3×3 Latin square used for the menu control techniques.	47
4.3	Overall score for menu control techniques.	49
4.4	ANOVA results for overall score for menu control techniques.	50
4.5	Percentage of choices for best menu control technique.	50
4.6	Realtime gesture recognition rate for different groups.	51
4.7	Overall score for navigation techniques.	52
4.8	ANOVA results for overall score of navigation techniques.	52
4.9	Percentage of choices for best navigation technique.	53
4.10	Overall score for single object selection techniques.	53
4.11	ANOVA results for overall score of single object selection techniques.	54
4.12	Overall score for multiple adjacent objects selection techniques.	54
4.13	ANOVA results for overall score of multiple object selection techniques.	54
4.14	Overall score of visibility techniques.	55
4.15	ANOVA results for overall score of visibility techniques.	55
4.16	Completion time (in seconds) recorded for the manipulation test.	58
4.17	ANOVA results for completion time.	58
2.1	Three versions of user study procedure.	70
4.1	Results of user study: menu control.	74
4.2	Results of user study: selection.	75
4.3	Results of user study: navigation.	75
4.4	Results of user study: visibility.	76

4.5	Results of user study: Realtime Gesture Recognition.	76
-----	--	----

List of Figures

2.1	A user exploring a virtual city in a CAVE system.	7
2.2	Optical tracking camera and target.	8
3.1	Overview of the implementation in the CAVE.	13
3.2	A wand input device (Flystick).	14
3.3	Input configurations for gesture interaction.	15
3.4	HMM gesture recognition framework.	16
3.5	CAVE and tracking coordinate system.	17
3.6	Example of Markov chain and HMM.	19
3.7	User interacting with a menu in a CAVE.	21
3.8	A simulation view of the slide menu.	23
3.9	A simulation view of the touch menu.	23
3.10	Demonstration of joystick navigation.	25
3.11	Demonstration of posture triggered flying.	26
3.12	Demonstration of waving hand navigation.	27
3.13	Ray casting selection.	27
3.14	Paint selection.	28
3.15	Stretch selection.	29
3.16	Manipulation by wand.	30
3.17	Illustration of two-handed manipulation.	31
3.18	Manipulation by wand.	32
3.19	The use of animated “collapse” to visualize hidden buildings.	32
3.20	Semi-transparent rendering of occluding building to visualize hidden buildings.	33
3.21	The use of wireframe to visualize hidden buildings.	33
3.22	Class hierarchy of the map data structure.	35
3.23	An example of a XML map file.	36
3.24	Linkage between 3D city and map data.	37

3.25	An example graph to demonstrate Dijkstra's algorithm.	38
4.1	Gesture definition in a recorded database.	43
4.2	Independent recognition result: confusion matrix.	44
4.3	Sample city data.	45
4.4	UI design procedure.	46
4.5	Gesture recognition rate versus user's rating.	51
4.6	Task description for quantitative manipulation test.	56
4.7	Distortion curves collected for the manipulation test.	57
5.1	Structure of prototype applications.	60
5.2	A user viewing demographic information for a selected area.	61
5.3	A user in the middle of a virtual tour.	62

List of Appendices

1	HMM Algorithms	67
2	User Study Procedure	70
3	Questionnaire Used in User Study	72
4	Data Collected in User Study	74
5	Published Papers	77

Chapter 1

Introduction

1.1 Background

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 two 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 (see section 2.1 for a more detailed introduction). Immersive systems, such as a head mounted display (HMD) [56], a Cave Automatic Virtual Environment (CAVE) [18] and various multi-screen or curved-screen systems [37], 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 [10]. 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.*[8] 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 [27], and interior design [43].

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 [22][71]. Commercial applications like Google Earth have gathered an enormous amount of 3D city data through the contribution of 3D modelers.

1.2 Purpose and Scope of This Thesis

With a long term goal of creating a fully immersive interactive 3D city planning system, the work described in this thesis focuses on interaction design in a virtual city scenario. Specifically, more effective and more user friendly interactions in an immersive VR environment are explored that support various functions needed for a virtual city application, such as navigation and path finding, object selection and manipulation.

Though the proposed interaction techniques are implemented and tested in the CAVE system, they can be adopted to other VR systems equipped with a tracking ability, such as various head mounted devices recently released in the commercial market, for example the Oculus Rift [65].

1.3 Contributions

The contributions of this work can be summarized as follows:

- The implementation of an integrated virtual city system that combines immersive visualization of 3D urban data in a CAVE, a geographic data structure to support high-level interactions, an optical tracking ability, and realtime gesture recognition. In order to

achieve natural and effective 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.

- The realization of a gesture interface through the application of a Hidden Markov Model (HMM) to recognize more complete 3D dynamic gestures. To the best of our knowledge, this is among the first systems that use a 3D tracker based HMM to assist in dynamic gesture recognition in a VR environment.
- The exploration and study of novel interaction techniques for each of the following interaction categories:
 - System control: compared to traditional button based menu control techniques, we propose two gesture based menu control methods: “Touch Menu” and “Slide Menu”;
 - Navigation: a “Posture Triggered Flying” and a “Waving Hand” technique are proposed for navigation in a large scale city scene, needed for applications such as virtual city touring;
 - Selection: a “Paint Selection” and a “Stretch Selection” technique are proposed for selecting multiple objects in order to simplify and make more efficient the subsequent manipulation of multiple virtual buildings;
 - Manipulation: a manipulation technique using two-handed gestures is proposed that provides the ability to translate, rotate, and scale virtual objects.

Our system is evaluated with empirical and quantitative user studies, 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 evidences from the user study, in the selection and manipulation categories, the proposed multiple objects selection techniques and two-handed gesture manipulation technique are significantly better than traditional ones. In the system control and navigation categories, the results are mixed: one part of the user evaluations does not show a significant difference among different techniques. However, another part of the user study results demonstrates that experienced users clearly prefer gesture based interactions. These findings could guide future interface design in immersive VR systems.

1.4 Overview of This Thesis

The remainder of this thesis is organized as follows: Chapter 2 reviews existing work related to the visualization and interaction of virtual cities in a VR environment; Chapter 3 illustrates our proposed methods and the system implementation in detail; Chapter 4 describes our experiments and the results from both qualitative and quantitative user studies; Chapter 5 introduces two prototype applications built on top of the various proposed methods; Chapter 6 draws conclusions and discusses possible future work.

Chapter 2

Literature Review

2.1 Immersive Displays and VR Systems

The term Virtual Reality (VR) should be differentiated with Augmented Reality (AR) [5] [4] where 3D virtual objects are integrated (augmented) into a real environment (*e.g.* an image captured by a camera). VR is a term used for fully computer generated 3D environments and it allows users to enter and interact with virtual environments [64] [23]. In VR applications, users can be partially or fully immersed in a synthesized 3D world generated by computers.

Slater [57] [58] proposed that the term “immersion” should only be used to represent an objective state that occurs when technology effectively mimics human sensory perception of the real world:

“The more that a system delivers displays (in all sensory modalities) and tracking that preserves fidelity in relation to their equivalent real-world sensory modalities, the more that it is ‘immersive’.”

And the term “presence” should be used to stand for the subjective experience of being in a virtual environment, as “a human reaction to immersion” [58]. However, in many other research papers in the literature, the terms immersive display or immersive system are not differentiated and represent display systems that deliver such an experience.

The overall level of immersion can be determined by many factors, including:

- field of view (FOV)—the size of the visual field (measured by degrees of the viewing angle),
- display size—which determines FOV together with viewing distance,
- display resolution,
- stereoscopy—displaying different images to left and right eye,

- head tracking—providing the ability to display based on location and orientation of the user’s head,
- display frame rate,
- realism of 3D models.

Depending on the degree of immersion, VR systems can be classified into non-immersive, semi-immersive, and fully immersive VR systems [9] [46] [7]. Non-immersive VR system usually means displaying via a desktop monitor, without stereoscopy and head tracking. Fully immersive VR systems, on the other hand, cover the full field of view of the human eye (approximately 210 degrees [3]), and are stereoscopic, with the added capability of head tracking. Other semi-immersive VR systems exist in the middle of non-immersive and fully immersive, such as various kinds of Head Mounted Display (HMD), and the so called “Fish Tank” VR [68], which combines a traditional 2D display with head tracking. Table 2.1 shows the comparison of several different VR or display systems.

System	Field of View	Stereoscopy	Head Tracking
Desktop PC	~20-30 degrees	No	No
DSharp [61] [53]	~110 degrees	No	No
GeoWall [26] [12]	~60 degrees	Yes	No
CAVE	~270-360 degrees	Yes	Yes

Table 2.1: Summary of immersion components for several VR or display systems.

The Cave Automatic Virtual Environment (CAVE), a fully immersive VR system, was proposed back in 1992 [18] [17]. Over the past two decades of virtual reality technology development, CAVE systems have been equipped with displays that have increasingly higher resolution and frame rate, as well as more graphics computing power. As shown in Figure 2.1, a CAVE system usually consists of 4 or more stereoscopic screens, configured as a cube surrounding the user. A tracking system detects the user’s head location. The displayed images are calculated in real time based on the user’s head position and the physical location of each screen.

As a result of this configuration, CAVE system achieves high level of immersion. It has been successfully used in many applications [12], such as military and medical training, entertainment, data visualization in the oil mining industry and automobile industry, etc.

2.1.1 Tracking in VR systems

Recent years saw various exciting new results in computer vision based tracking [30] [48], which are proposed for tracking in Augmented Reality (AR) and potentially could be used in VR.



Figure 2.1: A user exploring a virtual city in a CAVE system.

However, current vision based tracking still lacks the speed, accuracy, and robustness required in VR systems in order to support realtime rendering of 3D scenes. Popular tracking systems used in VR include magnetic tracking, mechanical tracking, acoustic tracking, inertial tracking, optical tracking, and hybrid tracking [69] [11].

Different from vision based tracking, all the above mentioned tracking systems need to attach trackers of some kind to the user's body. Among these solutions, optical tracking offers accurate results at relatively high speed and low latency with small user-worn components [69]. The biggest disadvantage of an optical tracking system is that it suffers from occlusion problems, since light travels in direct lines. However, this problem can be partly solved by using multiple camera arrangements that optimize tracking performance, at least within a limited range.

In an optical tracking system, the object whose position needs to be tracked is equipped with markers, which can be passive or active [42] [47]. Active markers are usually in the form of LEDs (Light-Emitting Diode) and emit internally generated light; while passive markers are made with light reflective materials and reflect ambient light or light emitted elsewhere. Light sources used in optical tracking systems are usually in the infrared range in order to avoid any influence of the display system.

Figure 2.2 shows a tracking camera and a target consisting of 4 markers [2]. The tracking cameras emit infrared light in a certain wavelength, which is then reflected by the markers and captured by the camera. Multiple cameras are mounted, scanning a certain volume and detecting the light that comes from the markers. Their images are processed to identify and calculate potential marker positions in 2D image coordinates, as illustrated in Figure 2.2(c).

These two dimensional locations from different cameras, together with known camera locations, are combined to calculate the 3D position of a single marker. If multiple markers are

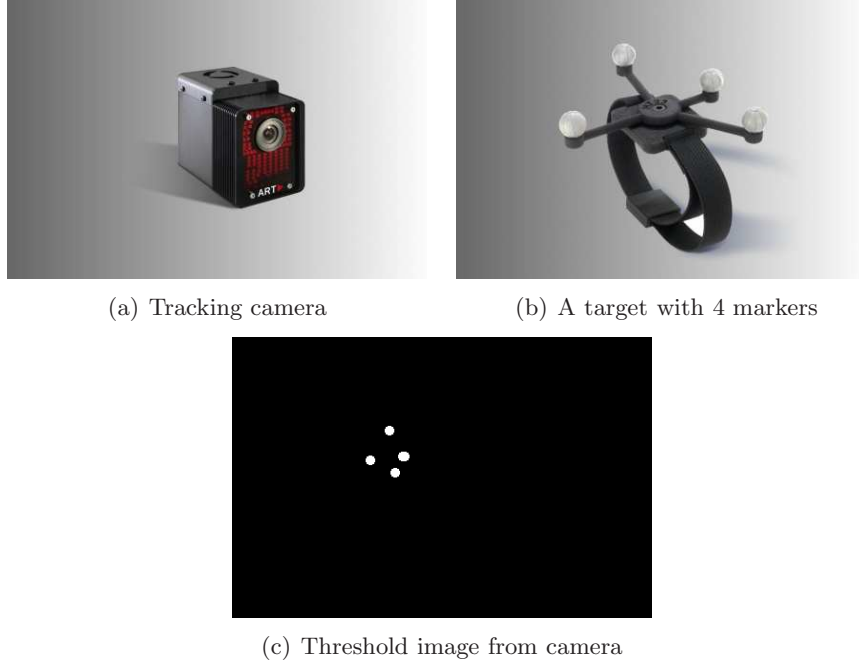


Figure 2.2: Optical tracking camera and target [2].

combined into a target, with their relative locations known, the 6DOF (6 Degree of Freedom) data of the target can be calculated, which include a 3D position and a 3D orientation.

2.2 Modelling and Visualization of 3D Urban Data

The computing and rendering power of graphics hardware has been dramatically increasing since the first dedicated graphics cards appeared for desktop PCs. Researchers and developers have also proposed new software and many algorithms that make the modelling and rendering of 3D scenes more realistic. Therefore virtual 3D cities have seen increased usage in applications such as Geographic Information Systems (GIS), urban planning, and gaming. Moreover, the popularization of the internet has made it simpler and easier for sharing and distributing 3D data and the applications that utilize them. All these factors have lead to the adoption of 3D cities into web-based map services and virtual globes, which have subsequently changed the way people acquire and interact with geographic information.

Considering 3D city data in particular, researchers have focused on the automatic or semi-automatic construction of 3D city models [22][71], as well as network based rendering of large scale data [54], to facilitate rendering in mobile devices with limited compute and rendering

power.

While the majority of research work use traditional 2D displays for visualizing 3D city data, some researchers have investigated the use of VR technology to achieve more immersive visualization. Kang *et al.*[27] developed middle-ware 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.

Isaacs *et al.*[25] developed a 3D virtual city application as a decision support tool for urban planners. Engel *et al.*[21] built a system that enables the display of virtual 3D city models in a semi-immersive display system with 360° cylindrical projections. Their work focused on the visualization side, using stereo screens to visualize an urban sustainability simulation, and lacked the ability to interact with the virtual city using a natural 3D interface.

Johnson *et al.*[26] reviewed the development of the “GeoWall” project, a low-price 3D display system designed for geoscience researchers. It was argued that GeoWall’s large stereoscopic display helps the users’ understanding of spatial relationships, which is crucial in geoscience research. However, once again the GeoWall system is designed mainly for visualization. It is not fully immersive (see Table 2.1) and lacks the capability for natural 3D interaction.

2.3 Human Computer Interaction (HCI) in VR

2.3.1 Input Hardware

A common input device in VR systems is the “wand”. It is a remote control that consists of positional trackers, buttons and sometimes a joystick or trackball. Individual buttons or a combination of buttons correspond to certain actions such as menu selection, up/down directions, etc. Abramyan *et al.* at the Jet Propulsion Laboratory [1] used a Nintendo Wii remote controller to control the viewing angle in a cylindrical displaying system that surrounds the users, for the purpose of controlling surface spacecraft. A wand was used as hand tracker in [33], to control a virtual table tennis game in an immersive VE that has 2 large stereoscopic screens and head tracking. Koike and Makino [31] proposed a 3D solid modeling system using wand to draw sketches on the screen, and a 3D model was then constructed from the original manual sketch. An experimental study on interaction in the CAVE showed that the displaying of a virtual hand is helpful in reducing interaction errors [63].

Besides the widely used wand devices, many other input devices have been invented for more intuitive and natural interactions. Bowman *et al.*[11] gave a comprehensive review of

the input devices developed for 3D interactions, for example various kinds of wearable devices like data gloves and shoes with pressure sensors. However most of these devices were designed for specific purposes and introduced intrusions (*e.g.* users have to wear complex devices) and consequently were not adopted for general use.

Some recent research explores the use of touch screens on mobile devices as an input device for Virtual Environments (VE). For example, Medeiros *et al.*[40] developed a system that use a tracked tablet as an all-in-one controller for immersive VR applications. They implemented a touch screen interface that supports selection, manipulation and navigation in 3D scenes. Song *et al.*[60] and Khan *et al.*[28] both proposed to use touch screen devices, an Apple iPod or an iPad, to control the rendering of 3D volume data.

2.3.2 Interaction Techniques

In addition to the design and configuration of input devices, the user experience in a VE is also largely determined by the design of interaction techniques (*i.e.* the mapping of input device actions into actions in the application). Although WIMP (Windows, Icons, Menus, Pointer) has been a dominant metaphor in User Interface (UI) design in the 2D world, no interaction techniques have reached this dominance in the 3D domain [70]. Along with the advance of VR technology, various 3D interaction techniques have been designed and explored, both for different kinds of input devices and for different interaction categories, such as navigation, selection, manipulation, etc. Some examples of 3D interaction techniques are as follows:

- Ray-Casting is an intuitive and widely adopted selection technique, where a ray is defined from a user defined location (derived from the input device position) to somewhere in the 3D space and objects intersecting the ray are selected. The ray casting technique has been modified in various ways to meet the requirements of specific conditions in a VE [11]. The modifications are generally related to how the pointing direction and the selection volume are defined.
- Stoakley *et al.*[62] proposed a manipulation technique called “Worlds in Miniature” (WIM) to manipulate objects out of a user’s physical range. Instead of directly manipulating a far away object, the user manipulates a smaller representation of that object in a smaller version of the whole 3D scene.
- Kim *et al.*[29] proposed using a touch screen device to control navigation in VEs. The fingers touch and movements onto the screen are mapped to navigation actions in the CAVE. They compared this technique with traditional joystick controlled navigation by

measuring the error of a user replicating a travelling (navigation) route, and concluded their proposed technique outperforms joystick navigation.

- Bowman *et al.*[13] compared three different menu control techniques in a HMD VR environment: their proposed TULIP menu, a floating menu and a tablet menu. The TULIP menu is displayed at the end of each finger of a virtual hand, and is controlled by a glove the user wears that can detect pinching; The floating menu is a widely adopted form in VR applications, where the 2D menu is “floating” in front of the user; a tablet menu, on the other hand, is displayed on the surface of a virtual tablet while the user also holds a physical piece of cardboard, using a physical pen to select on that physical cardboard. They concluded that the floating menu and tablet menu are the fastest, while the TULIP menu was the technique most preferred by the users.

Pattern recognition methods have also been applied to support interactions with computers, particularly in the form of speech recognition and gesture recognition. Laviola [32] investigated applying speech commands in VEs. However, the biggest problem of speech input is the accuracy with which the input processing technique can distinguish between the user issuing a command or just talking with a co-worker. In addition to the lack of robustness, as well as privacy issues, speech recognition has not seen widespread usage, both in VR systems and traditional computing environments. Biomedical signals such as electroencephalogram (EEG) signals from the brain and electromyography (EMG) signals from muscles have also been used as inputs [41] [50]. However these technologies are still in their infancy and their potential to be used as general purpose input devices may be limited.

In recent years many researchers have been issuing commands via gestures interpreted using gesture recognition techniques, partly due to the widely affordable motion tracking devices such as the Microsoft Kinect [51] [67]. Most of these new developments use gesture recognition for non-VR environments. For example, Schlomer *et al.*[55] 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.* [52] 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. However, only a few researchers have reported the use of gesture recognition in VR systems.

Chapter 3

Methodology

In this chapter, the integrated system is described and the developed interaction techniques are presented. We start with introducing the hardware and software configuration of our VR system, the CAVE, and the optical tracking system used for user inputs. Then, in the system control section, functionalities of the proposed virtual city are summarized and organized into a menu structure. In addition to traditional button controlling, two different gesture based techniques for 3D menu control are proposed, which utilize the recognition power of a Hidden Markov Model.

Navigation, selection, and manipulation are three of the most basic operations in all 3D interactive applications. In the case of a virtual city, different interaction techniques for each of these three operations are described and illustrated. While the joystick is one of the most widely used means for navigating (travelling) in a virtual scene, in this thesis two posture and gesture based travelling techniques are proposed in order to achieve more natural interaction. For the selection operation, a ray casting technique is used to select single buildings. However, ray casting is not as effective for selecting multiple buildings. Therefore, we propose “paint” select and “stretch” select methods for multiple object selection. Finally, three methods are proposed to deal with occluded viewpoints (i.e. when the desired view of a building/region is blocked by one or more buildings).

The sum of all these general interactions is not enough to achieve natural interaction. User friendliness is often achieved by utilizing application-specific and task-specific constraints. For example, in medical data visualization users may prefer a selection technique that distinguishes among different organs and tissues, over a general selection method that selects whatever volume data is “pointed at”. In the case of a virtual city, the structure and layout of buildings, blocks and streets can be used as “natural” constraints for interactions. Therefore, in this thesis a map data structure is presented that encodes the geographical data in a high-level way. Special care

is given to link the map structure with the 3D city data in order to create effective interaction constraints. For example, using the map data structure, an implementation of Dijkstra's path finding algorithm has been added to enable high level navigation in the virtual city.

3.1 System Configuration

3.1.1 The CAVE

As illustrated in Figure 3.1, our CAVE consists of 4 wall size displays, covering the front, left and right sides, and the floor of the user's viewpoint. Each of these displays is driven by a stereo projector and a workstation PC equipped with high-end graphics card. These workstations, together with a server node, form a graphics cluster that runs the VR application.

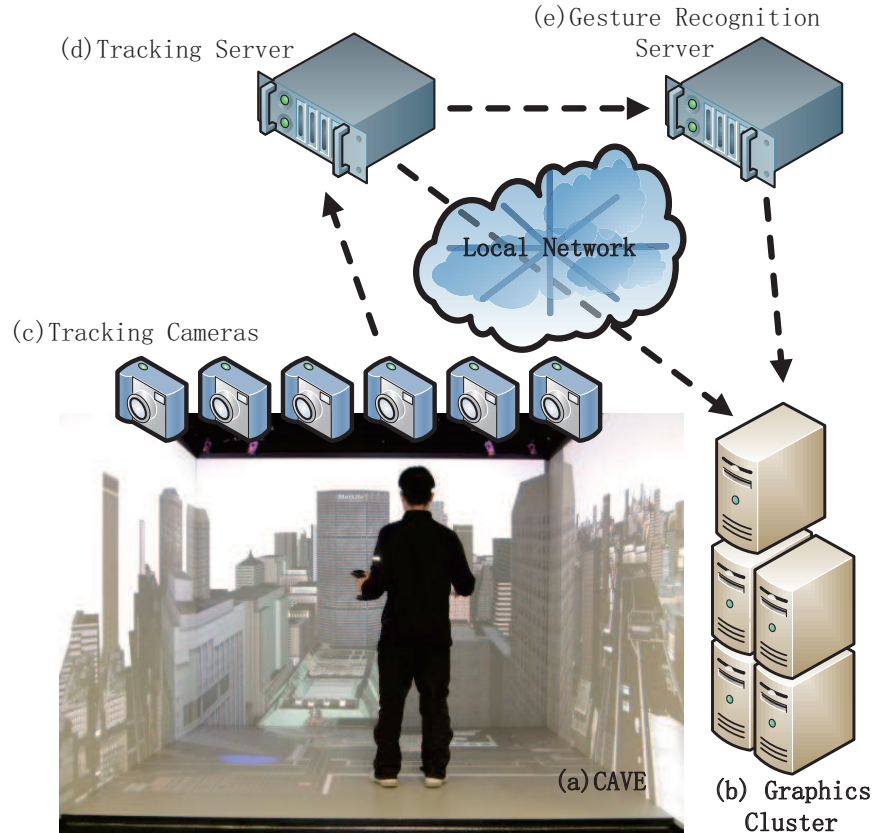


Figure 3.1: Overview of the implementation in the CAVE.

Optical tracking cameras are mounted on top of the screens to capture the user's movement.

They emit infrared light and capture the light reflected by markers worn on the user’s head and hands, as introduced in section 2.1.1. 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. The gesture recognition algorithm will be described in section 3.2.

3.1.2 Software Configuration

We have constructed our virtual city application on top of Open Scene Graph (OSG) [66], a popular cross platform visualization toolkit that supports a wide range of 3D data formats. A highly configurable VR toolkit, VR Juggler [6], is used to direct the visualization into the CAVE screens. The use of VR Juggler supports portability to various other display systems, as well as a traditional desktop monitor. Figure 3.1(a) shows a user experiencing the virtual city in the CAVE.

3.1.3 Input Devices

A “wand” is an input device that is commonly used in traditional VR systems. It consists of a combination of a positional tracker and buttons/joysticks. Figure 3.2 shows the wand used in our CAVE, which comes as an accessory to the ART optical tracking system, and is named “Flystick” [2].



Figure 3.2: A wand input device (Flystick).

Holding the wand provides 6DOF tracking data of the user’s hand. However, in tasks such as rotating and scaling an object, using 2 hands is more natural, as will be demonstrated later

in chapter 4. In the proposed gesture interface, both hands of the user are tracked. One way of doing this is to wear a set of markers on both hands, as shown in Figure 3.3(a). But the downside is losing the power of wand buttons. We then have to design other ways to issue commands to the system, such as the use of gestures. This means the user needs to memorize many gesture commands resulting in a more complex interface and a reduction in the easiness of learning and using the interface. Note that this complexity is not due to limitations of the gesture recognition algorithm. The proposed gesture recognition scheme is capable of recognizing more gestures than actually used, which will be shown in chapter 4. The primary reason for reducing the number of gestures is the goal of creating a simple and natural user interface. Simplicity and naturalness have been emphasized by experts and practitioners in user interface design [70][11].

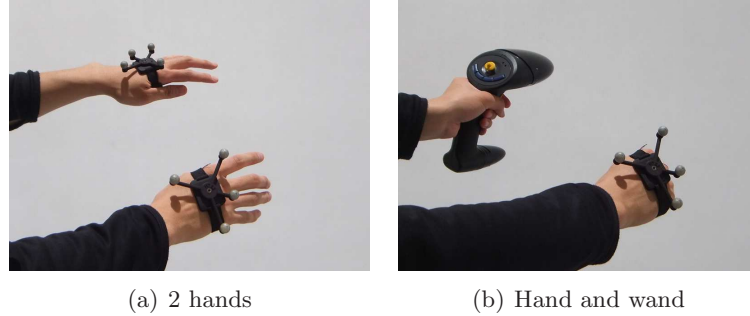


Figure 3.3: Input configurations for gesture interaction.

Furthermore, for manipulation and selection tasks that require precise control, performing another gesture to trigger the start of these operations could affect the accuracy of the interactions themselves. Therefore, we believe a better solution is to use as few gestures as possible, and complement them with wand buttons. Wand buttons are more robust, precise and familiar to users from their use of a computer mouse. To make the interactions 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. The input configuration is as follows: 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 flystick, as shown in Figure 3.3(b). In this way, we have the ability to use two-handed gestures and keep the buttons/joysticks for robust command triggering.

3.2 HMM Realtime Gesture Recognition

Because of its ability to handle stochastic signals, Hidden Markov Model (HMM) has been widely used in speech recognition and gesture recognition, and has achieved good results [49]

[20]. This section introduces our approach of applying HMM to recognize 3D trajectories of dynamic hand gestures.

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 the multi-touch control of an image on a tablet or smartphone. These simple direct manipulation gestures are commonly referred to as *online* gestures.

Figure 3.4 shows the framework of the proposed gesture recognition system.

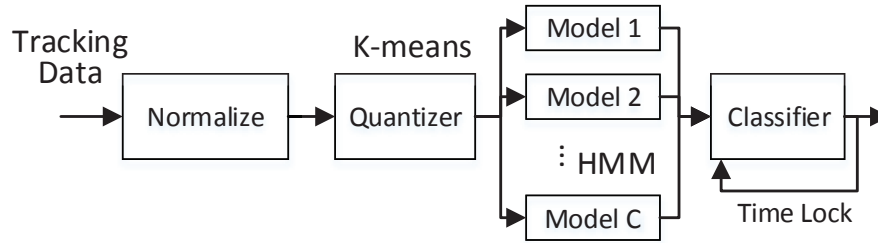


Figure 3.4: HMM gesture recognition framework.

3.2.1 Normalization and Quantization

The optical tracking system provides very accurate positional data. However, the position is in a predefined coordinate system. For example, in the coordinate system defined by Figure 3.5, when performing the same gesture at different locations the resulting hand positions are different. Therefore, without normalization, if the recognition system is trained by data collected while standing in the center of the CAVE facing the front screen, exactly the same gesture performed while standing in one side and facing the other side cannot be recognized properly.

One possible solution is to project the 3D position to a plane, and do recognition in a 2D space. While it is true that most human hand gestures are 2-dimensional, projecting hand positions to a plane will completely lose the ability to recognize 3-dimensional gestures.

In this work, a local reference coordinate system is used to normalize the hand position data. The user's head position is used as the origin for a reference coordinate. The normalized

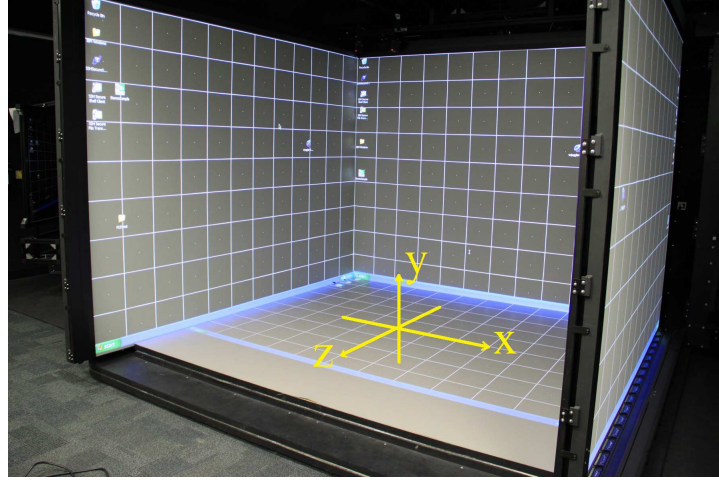


Figure 3.5: CAVE and tracking coordinate system.

hand position \mathbf{p} is given by:

$$\mathbf{p} = \mathbf{p}_h * \mathbf{M} - \mathbf{p}_r, \quad (3.1)$$

where $\mathbf{p}_h = (x_h, y_h, z_h)$ is the position of the 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:

$$\theta_y = \arctan \frac{\sin \psi \sin \phi + \cos \phi \cos \psi \sin \theta}{\cos \psi \cos \theta}, \quad (3.2)$$

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 nodding 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:

$$\mathbf{v} = \mathbf{p} - \mathbf{p}_{t-1} \quad (3.3)$$

where \mathbf{p}_{t-1} is the normalized hand position in the last frame. Then a gesture sample can be represented by a sequence of velocities:

$$\mathbf{G} = \{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_N\} \quad (3.4)$$

where N is the length of this sequence, in terms of the number of frames.

Since a HMM has a finite number of states and observation symbols, as will be introduced in the next section, speed data needs to be quantized into a finite number of states. K-means clustering is used here to accomplish this.

K-means is an un-supervised clustering algorithm (semi-supervised in some sense, because the number of clusters, K still need to be specified) [36]. It partitions the data $(\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_N)$ into K sets $\mathbf{V} = \{V_1, V_2, \dots, V_k\}$, that have the minimum representation error:

$$\arg \min_{\mathbf{V}} \sum_{i=1}^K \sum_{\mathbf{v}_j \in V_i} \|\mathbf{v}_j - \boldsymbol{\mu}_i\| \quad (3.5)$$

The most common algorithm to solve the K-means problem is an iterative refinement technique, each iteration of which includes 2 steps:

- Assignment step: Assign each data point to the cluster whose mean is closest to it:

$$V_i^{(t)} = \{\mathbf{v}_p : \|\mathbf{v}_p - \boldsymbol{\mu}_i^{(t)}\| \leq \|\mathbf{v}_p - \boldsymbol{\mu}_j^{(t)}\| \quad \forall 1 \leq j \leq k\} \quad (3.6)$$

- Update step: Calculate the new means to be the centroids of the data points in the new clusters.

$$\boldsymbol{\mu}_i^{(t+1)} = \frac{1}{|V_i^{(t)}|} \sum_{\mathbf{v}_j \in V_i^{(t)}} \mathbf{v}_j \quad (3.7)$$

In our practical recognition system, the K value is chosen by testing the recognition system on a gesture database. The K value that achieves the best recognition rate will be used. After clustering, each incoming data \mathbf{v} can be clustered to its nearest centroid, which is then fed into the HMMs.

3.2.2 Hidden Markov Model and Classification

A discrete Hidden Markov Model (HMM) is an extension on top of a Markov chain, a random process where the current state is only dependent on its previous state:

$$P[q_t = S_j | q_{t-1} = S_i, q_{t-2} = S_k, \dots] = P[q_t = S_j | q_{t-1} = S_i]. \quad (3.8)$$

where q_t denotes the state at time t . There are N distinct states, S_1, S_2, \dots, S_N .

Furthermore, these transition probabilities are independent of time, represented by a matrix $A = \{a_{ij}\}$:

$$a_{ij} = P[q_t = S_j | q_{t-1} = S_i], \quad 1 \leq i, j \leq N, \quad (3.9)$$

where

$$\begin{aligned} a_{ij} &\geq 0 \\ \sum_{j=1}^N a_{ij} &= 1. \end{aligned} \quad (3.10)$$

Another parameter that describes a Markov process is the initial state probability, $\pi = [\pi_1, \pi_2, \dots, \pi_N]$, where

$$\pi_i = P[q_1 = S_i], \quad 1 \leq i \leq N. \quad (3.11)$$

Figure 3.6(a) shows a Markov chain with 5 states, while Figure 3.6(b) shows its extension to HMM. For a HMM, the state at each time, q_t is no longer observable. Instead, the only output of the HMM is the observation symbol O_t , which could be one of M possible observations, denoted as V_1, V_2, \dots, V_M . We use $B = \{b_j(k)\}$ to denote the probability of observing symbol V_k at state S_j :

$$\begin{aligned} b_j(k) &= P[O_t = V_k | q_t = S_j], \quad 1 \leq j \leq N \\ &1 \leq k \leq M. \end{aligned} \quad (3.12)$$

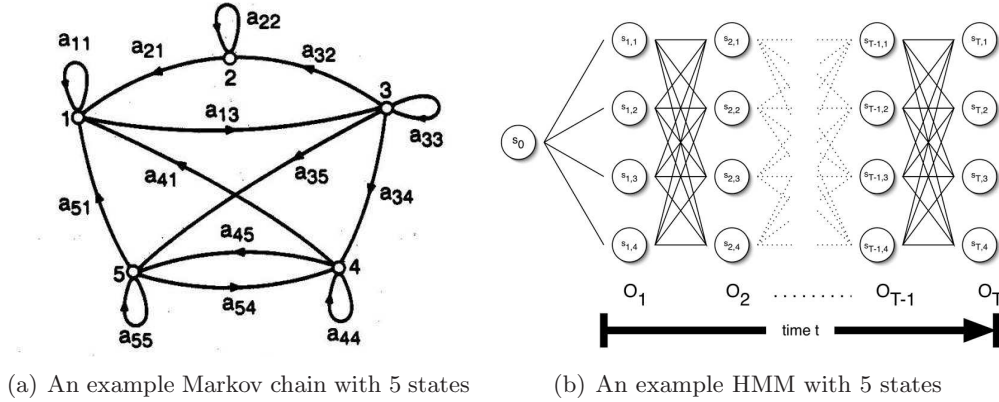


Figure 3.6: Example of Markov chain and HMM.

Therefore, a HMM can be fully specified by the parameter set λ :

$$\lambda = (A, B, \pi) \quad (3.13)$$

Classification

Appendix 1 gives a detailed description of training and recognizing algorithms of a HMM. For each of C classes of gestures, we use some training sequences to adjust the HMM's parameters. The resulting HMM represents that specific class of gesture. As shown in Figure 3.4, in the realtime recognition, a gesture sequence is fed into each of C trained gesture models. The output of each model is the probability of the gesture sequence belonging to that model. Then the classifier assigns this gesture sequence to the model with the highest probability, and outputs this result. However if the highest probability is not big enough, the sequence is classified as containing no gestures.

In a real time implementation, the frame rate of the tracking data (60Hz in our case) is much higher than the perceivable speed of hand movements. Therefore tracked position data often remains similar for multiple frames, and the recognizer often outputs the same class during all these frames. This often happens 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 of time which is roughly half the length of the average gesture duration.

3.3 System Control

Now all inputs to the system are configured, including the wand and hand markers for tracking (as illustrated in section 3.1.3), and the recognized gestures (the definition of gesture commands will be given in section 4.1). This section will introduce system control methods, which enable controlling the VR experience within the CAVE.

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.

We have therefore defined 3 basic interaction modes that support the navigation and selection/manipulation system functions: “FreeView”, “Selection”, and “RouteView”. In each of these modes, there are functions such as selecting buildings, streets and waypoints, transforming or replacing selected buildings, etc. In order to keep the interactions as simple and natural as possible, a menu is used as the system control center, where the user selects different functions or modifies the behaviour of the whole VR application. Figure 3.7 shows an image of

a user operating a menu in a CAVE.

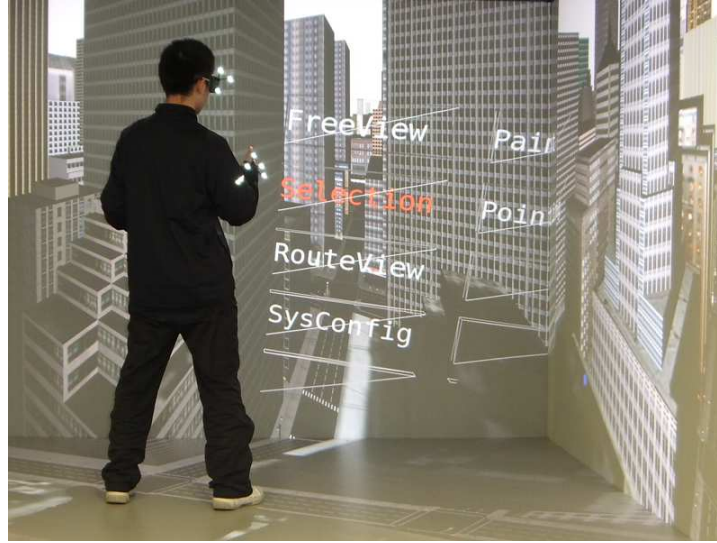


Figure 3.7: User interacting with a menu in a CAVE.

In a 3D immersive VR system, the user is immersed in 3D content. Thus the menu is displayed in 3D to provide a consistent experience and to enable the interaction of “touch” as will be shown later. After the menu is activated (i.e. triggered), menu items, which consist of 3D texts or icons, will appear to float in front of the user.

3.3.1 Menu Control by Button

Controlling a menu by buttons is one of the oldest and most widely used interactions in Graphical User Interfaces (GUI). Here, 4 buttons (on the Flystick, as shown in section 3.1.3) are assigned to control the menu. The assignment is given in Table 3.1.

Action	Flystick Button
Confirm/Trigger Menu	Button 1
Cancel	Button 2
Up	Button 3
Down	Button 4

Table 3.1: Flystick button assignment for menu control.

Due to limited number of buttons, some of them are used for multiple functions. For example. Button 1 is used to trigger the menu, and is also used to confirm a selected menu

item. There are other functions, such as selection and manipulation, that will be triggered using buttons and which will also be assigned to duplex function buttons.

Up and Down buttons are used to select different menu items. The selected menu item is then highlighted to give a visual feedback. A press on the confirm button will call the function or the submenu that a menu item represents. A press on the cancel button will cancel the whole menu interface, or return to the parent level, if currently inside a submenu.

3.3.2 Menu Control by Gestures

Though button interactions are familiar to most users and therefore easy to get used to, gestures are considered to be a more intuitive way of interacting [70][11].

Two gesture based menu control techniques are proposed: “Slide Menu”, and “Touch Menu”. Similar concepts exist in a 2D domain. However, unlike the traditionally dominant “WIMP” (Windows, Icons, Menus, Pointer) metaphor, the proposed slide menu does not need a pointer for positioning on menus and icons. Instead, it’s totally controlled using 3D gestures. The Touch Menu resembles common menu interactions on a touch screen. However, touching on a 3D menu in an immersive environment gives far more realism and feeling of a true physical menu floating in front of the user.

To activate the menu, the user performs a “sliding down” gesture, as if pulling down something above his/her head. This action will trigger the menu to be displayed in front of the user.

Slide Menu

The Slide Menu works as follows: the user moves his/her hand up or down to highlight different menu items. After the target item has been highlighted, a slide gesture towards the right will confirm that selection, as if the confirm button had been pressed; a slide gesture towards the left will count as a cancel action. Recognition results for left and right sliding gestures, as well as the sliding down gesture to call out the menu, work as trigger signals and replace buttons. Figure 3.8 shows a simulated view of the slide menu.

A problem with this design is that, after confirming a menu selection with a right slide, the user needs to withdraw his/her hand from the right side. This withdrawing action is sometimes recognized as a left slide and immediately cancels that menu selection. To solve this problem, a lock mechanism is implemented. Specifically, left slide gestures that come within a short period of time after a right slide gesture are ignored.

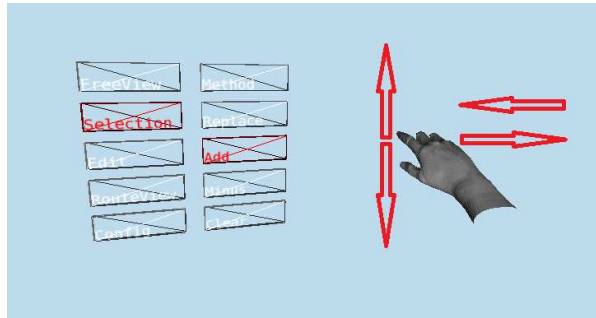


Figure 3.8: A simulation view of the slide menu.

Touch Menu

In a touch menu, the user moves his/her hand to “touch” a menu item on the 3D menu to select it as well as confirm that selection. Returning to the parent level menu requires no additional effort - just touching the parent menu item. Cancelling the menu is done by touching a menu item named “cancel”. Figure 3.9 shows a virtual hand touching on a menu item. Since the virtual hand is constantly following the user’s actual hand, it delivers a feeling of actually reaching out and “touching” the menu.

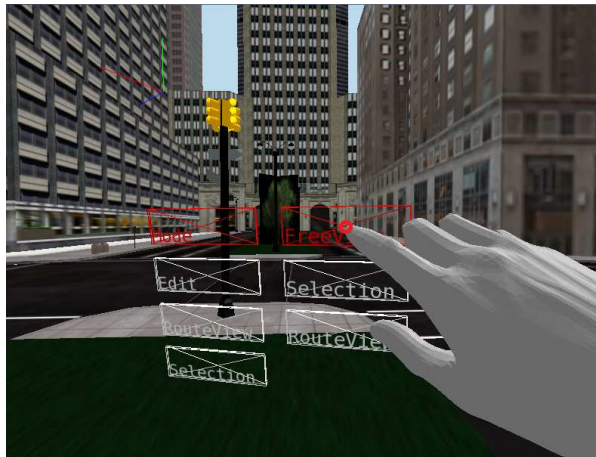


Figure 3.9: A simulation view of the touch menu.

3.4 Navigation

Navigation in a VR application means to travel from one place to another in the virtual scene. Though navigation generally means the changing of camera position, one effective way of travelling is by manipulating and moving the whole virtual scene, since in a VR system the camera position is actually the user's eye location. Movement of the user is always confined by space limitations of the VR system, for example the effective tracking range. Therefore the only way of performing large scale travelling is by moving the virtual objects instead of physical eye movement.

Manipulation techniques that will be introduced in section 3.6 are used here to support scaling, rotating and moving of the entire virtual scene. 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 [62][8] in VR navigation, which manipulates a small version of much larger scene to navigate in the large scene.

In order to have access to this navigation function from anywhere in the application and in any mode, a wand button is reserved to trigger the manipulation of entire virtual scene. 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. Also, the triggering gesture itself would affect the precision of subsequent manipulation gestures.

In addition to manipulating the scene for navigation, sometimes the user may want to “fly” in the scene without changing its scale, for example in a virtual tour. There are 3 such navigation techniques implemented in this work.

3.4.1 Joystick

The use of a joystick is a popular way of navigating in many 3D applications, especially computer games. Together with the direction pointed by the user, a 2-axis joystick offers the ability of flying forward, backward and turning freely in 3D space. Figure 3.10 illustrates the usage of joystick navigation.



Figure 3.10: Demonstration of joystick navigation: the right hand is used to point out a direction and pushing on the joystick activates flying.

3.4.2 Posture Triggered Flying

To offer an option of navigation without joystick and buttons, a posture triggered flying interaction is proposed. As shown in Figure 3.11, the user raises his/her left hand above the head to fly to the direction pointed by the right hand. The speed of travelling is calculated by a sigmoid function:

$$|v| = \begin{cases} \frac{v_{max}}{1+e^{d_y/k}} & : d_y > 0 \\ 0 & : d_y < 0 \end{cases} \quad (3.14)$$

and,

$$d_y = y_{hand} - y_{head} \quad (3.15)$$

where y_{hand} and y_{head} represent the height of the user's left hand and head, respectively. The user has some freedom of controlling the travel speed by the height of their left hand over their head. The choice of a sigmoid function over a linear one is to place a limit on the highest speed. Feedback from a user study finds that dizziness occurs when the speed is too high. v_{max} and k are constants that control the highest speed and the sensitivity over hand height change.

3.4.3 Waving Hand

The waving hand technique comes from the desire to use manipulation like methods to travel. Compared to the two-handed gesture manipulation described in section 3.6, the waving hand navigation interaction differs in several ways. Firstly, only translation is considered, without

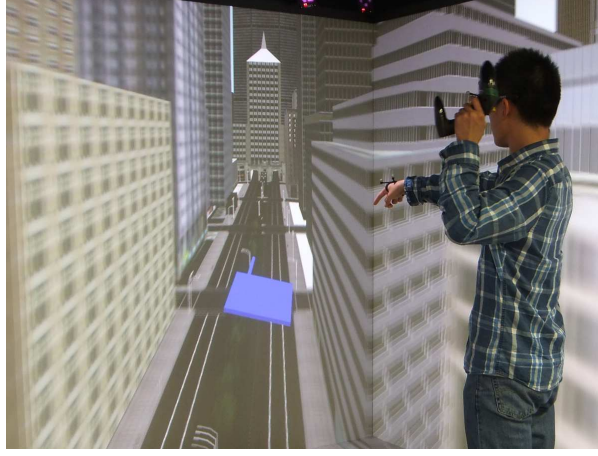


Figure 3.11: Demonstration of posture triggered flying: the right hand points out a direction and raising the left hand activates flying.

rotation and scaling of the scene. Secondly, the motion of one hand is considered, instead of two, since the position of a single hand is sufficient for translation. Thirdly, an inertial motion continues after the hand motion is complete to achieve a feeling of physical movement, which greatly enhances the user experience.

Specifically, the speed and direction of travelling during current frame is determined by the movement of the user's right hand:

$$\mathbf{v} = \mathbf{p}_{r,t-1} - \mathbf{p}_r \quad (3.16)$$

where \mathbf{p}_r and $\mathbf{p}_{r,t-1}$ represent the right hand position in the current frame and in the last frame. After the navigation triggering button is released, the flying will continue until the speed decreases below a threshold close to zero:

$$\mathbf{v} = \frac{|\mathbf{v}_{t-1}| - a}{|\mathbf{v}_{t-1}|} \mathbf{v}_{t-1} \quad (3.17)$$

where a is the acceleration factor for inertial effect, and \mathbf{v}_{t-1} is the travelling velocity in the last frame.

3.5 Selection

Object selection is another basic function in 3D applications. In the virtual city, three selection techniques are proposed. The first one is a traditional ray casting method. However, it selects one building at a time and lacks effectiveness for selecting multiple buildings: a task often



Figure 3.12: Demonstration of waving hand navigation: fly to the direction of hand waving.

performed in a city planning scenario. Therefore two effective ways of selecting multiple objects are proposed in this thesis: “paint” selection and “stretch” selection.

3.5.1 Ray Casting Selection

The ray casting selection, as its name implies, is performed by casting a ray from user’s specified position and direction, which in our case is pointed directly by user’s hand. As shown in Figure 3.13, the first object that intersects with the ray is selected.

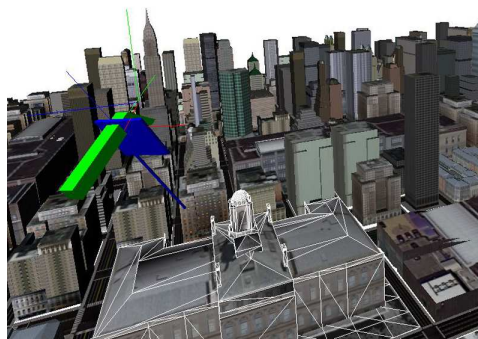


Figure 3.13: Ray casting selection: a ray is cast by the user’s hand pointing at an object. The first object that the ray intersects is selected.

3.5.2 Paint Selection

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, with the movement of user’s hand controlling the position of the brush (once the selection function is triggered). In our implementation, a simple sphere brush is used, as shown in Figure 3.14.

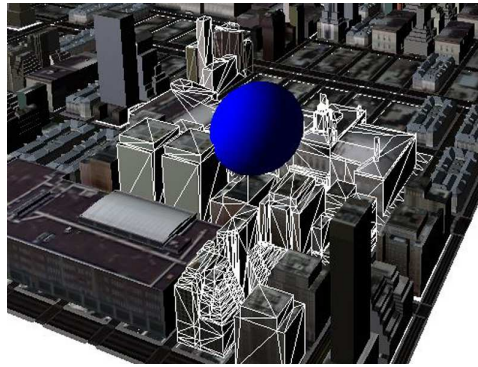


Figure 3.14: Paint selection: the user moves a spherical brush to “paint” in the scene. Every object that intersects the brush is selected.

3.5.3 Stretch Selection

Stretch selection is a concept borrowed from 2D GUIs, where users can select icons by creating and stretching a rectangular box; those icons inside the box are selected. In the 3D world, we can still create a box but it will be a cuboid instead of a 2D rectangle. When holding the selection button, the user’s hand position is used to stretch the box as shown in Figure 3.15. Since buildings are not the same height, the height of the box might not be suitable for all buildings intended to be selected. Therefore, the stretched box is projected to the ground, into a rectangular area. Every building inside that area will be selected.

3.6 Manipulation

According to the Oxford English Dictionary, to “manipulate” means to handle or control (a tool, mechanism or information, etc.) in a skilful manner. In the context of 3D user interfaces, the definition of manipulation usually narrows to the action of handling spatially rigid objects

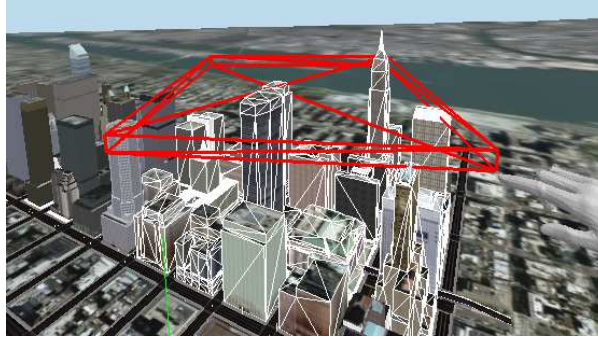


Figure 3.15: Stretch selection: the user stretches a 3D box, which is then projected to a rectangular area on the ground. Every building inside that area is selected.

[11], which means changing the position, rotation, or size of a target object (or objects), and preserving its (their) shape. The virtual city application is built in an immersive VR environment whose main advantage is the realism of visualization. Therefore the purpose of our manipulation interaction design is focused on the easiness of usage and support for visualization tasks rather than precise 3D modeling manipulations required when constructing individual objects.

3.6.1 Wand Manipulation

As introduced in section 3.1.3, a wand in a VR system has its 6DOF position and orientation tracked. With the addition of a triggering button, the wand is able to act like a virtual hand, picking up a selected object, and then moving and rotating it. This is one of the common interactions implemented in many existing VR applications.

Figure 3.16 shows the usage of wand manipulation. To move (translate) the target object (or objects), simply move the wand while holding the manipulation triggering button. The movement of the wand is mapped directly to the movement of virtual objects. This also applies to rotation, as shown in Figure 3.16(b). Rotation of the wand will rotate the virtual objects. The user has the option to set rotation pivot point to the center of target object or the current position of wand.

While the rotation and translation action use up all the 6 degree of freedom tracking information, including position and orientation, we cannot build the ability of scaling on top of the tracking data (i.e. using a gesture to scale at the same time with rotation and translation). One solution to this problem is to add a separate scaling mode. For example, user calls out the menu to switch to scaling mode, instead of rotation and translation, and then uses the slide up or down gesture to enlarge or diminish target objects. However, this would increase the

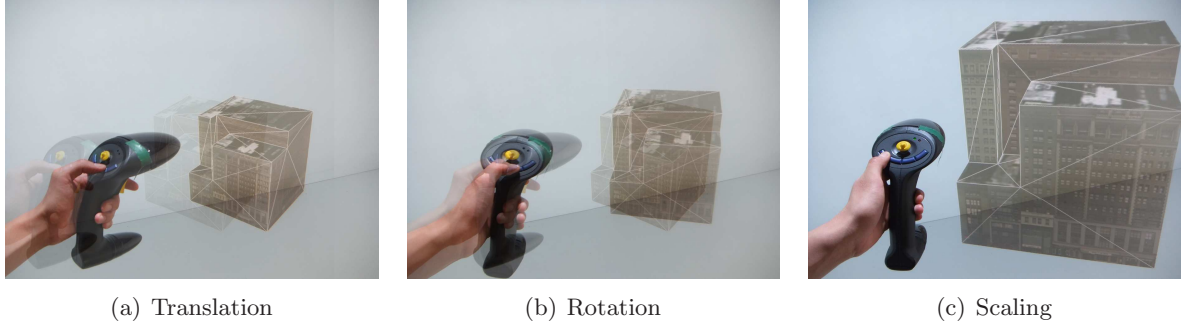


Figure 3.16: Manipulation by wand.

complexity of manipulation interface.

Therefore, two buttons are used for scaling. When the manipulation triggering button is on hold, the scale up button is pressed to increase the size of selected objects and the scale down button to decrease the size, as shown in Figure 3.16(c).

3.6.2 Two-Handed Gestures

The idea of two-handed gestures for manipulation is based on multi-touch gestures used on smartphones for moving, zooming and rotating pictures. By adding another tracked hand position, two-handed manipulation achieves a more natural user experience by imitating the physical action of moving and rotating. Suppose \mathbf{p}_{l0} and \mathbf{p}_{r0} denote the position of left and right hands upon manipulation triggering time t_0 , respectively, and \mathbf{p}_l and \mathbf{p}_r represent the corresponding position at the current time t . Calculation of the rotation, scaling and moving parameters is given as follows:

- **Rotation.** As shown in Figure 3.17(a), rotation of the virtual object can be represented by the rotation from vector \mathbf{l}_0 to \mathbf{l} , where

$$\begin{aligned}\mathbf{l}_0 &= \mathbf{p}_{r0} - \mathbf{p}_{l0} \\ \mathbf{l} &= \mathbf{p}_r - \mathbf{p}_l\end{aligned}\tag{3.18}$$

- **Scaling.** As in Figure 3.17(b), the scaling factor of the virtual object is given by:

$$s = \frac{|\mathbf{l}|}{|\mathbf{l}_0|}\tag{3.19}$$

- **Moving.** Translation can be represented by:

$$\mathbf{t} = \frac{\mathbf{p}_r + \mathbf{p}_l}{2} - \frac{\mathbf{p}_{r0} + \mathbf{p}_{l0}}{2} \quad (3.20)$$

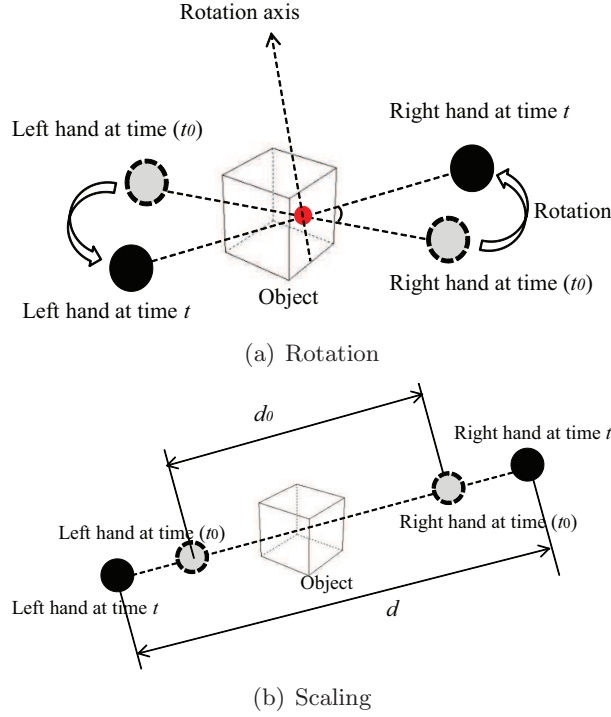


Figure 3.17: Illustration of two-handed manipulation.

Figure 3.18 depicts a two-handed gesture for 3D manipulation and a multi-touch gesture for 2D manipulation.

3.7 Visibility of Hidden Buildings

Visibility issues are a common and well known problem in 3D applications since the area of interest is often occluded (i.e. blocked) by other objects. For example, in a virtual city visualization, a city planner may want to have a whole view of a community from a specified view point, which is however blocked by a high rising tower close to that community. In this case, having an option to look “through” the obstacle is one way of solving the occlusion problem. In this work, three options are proposed to tackle the occlusion problem. These options can be individually selected from the control menu.

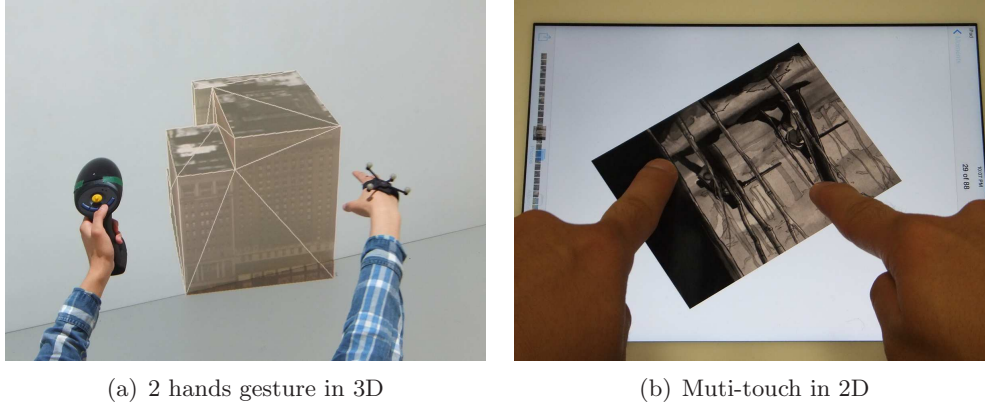


Figure 3.18: Manipulation by wand.

3.7.1 Collapse

The first method is named “collapse”, since it resembles a physical collapsing of a building. Figure 3.19 shows a building halfway collapsing down to show the view blocked by it.

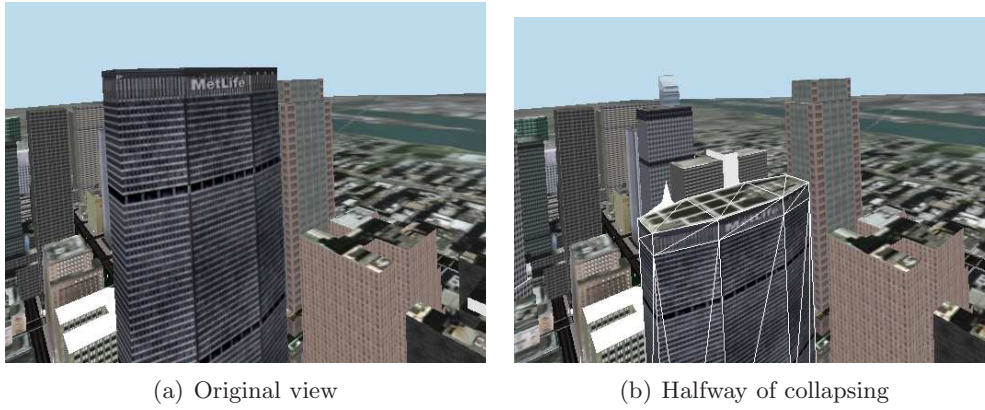


Figure 3.19: The use of animated “collapse” to visualize hidden buildings.

3.7.2 Semi-Transparent

Another way to deal with the occlusion problem is to render the the occluding building semi-transparent, as shown in Figure 3.20.



Figure 3.20: View from the same point when occluding building is semi-transparent, with a transparency of 20%.

3.7.3 Wireframe

In addition to displaying buildings semi-transparently, wireframe rendering can also be used to enable seeing through the obstacle. In this mode only the outline of a building is displayed, as demonstrated in Figure 3.21.

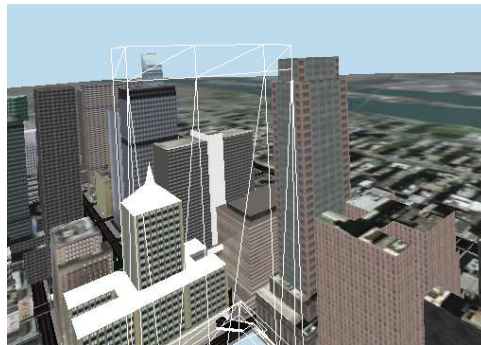


Figure 3.21: View from the same point when the occluding building is displayed in wireframe.

3.8 Adding Constraints

The selection and manipulation techniques introduced in the previous sections can all be used for general VR applications, not just the virtual city. However, designing application specific interactions could potentially significantly improve the user experience [8]. This means utilizing constraints in a specific application and limiting the degrees of freedom in interactions.

For example, one of the natural constraints in a virtual city is that buildings and other

infrastructure objects have a vertical orientation with respect to the ground. Thus, during manipulation, a building should only have the freedom of rotating around the vertical axis. Equation 3.18, which describe the rotating manipulation of buildings, will therefore change to:

$$\begin{aligned} \mathbf{l}_0 &= (x_{r0}, 0, z_{r0}) - (x_{l0}, 0, z_{l0}) \\ \mathbf{l} &= (x_r, 0, z_r) - (x_l, 0, z_l) \end{aligned} \tag{3.21}$$

Note that $\mathbf{p} = (x, y, z)$. In the new equation 3.21, y values in all hand positions are set to 0, thereby eliminating rotation around y axis.

Using the same constraint (i.e. the relationship between the vertical axis and the ground), the degrees of freedom in a selection interaction can also be limited. In the stretch selection method introduced in section 3.5.3, target objects need to be inside the box to be selected. However, in a virtual city, all buildings are on top of the ground, and no building can float in the air on top of one another. In this case there is no need to consider height while drawing the selection box. Wherever the user draws a selection box, it can be projected to a rectangular box on the ground and buildings inside that area will be selected, as already shown in Figure 3.15.

Yet another higher level constraint in a virtual city is the geographic information. A city consists of a large number of different buildings, which are connected by streets and other kinds of transportation systems such as subway lines. A city can also be divided into multiple districts and communities serving different purposes. City planners or tourists usually want high level functions that understand geographic relationships. For example, they may want to tour a specific street or a certain area, and have the tour path automatically constrained to the center of streets; they may also want to check the nearest way to shopping centers from a proposed new residential project. The remaining part of this section will introduce the map data structure and path finding algorithm in order to achieve these goals.

3.8.1 Map Data Structure

There are 2 types of data models used in the field of Geographic Information System (GIS): vector data models, and raster data models [16]. Vector data models use points and their x, y -coordinates to represent discrete features with a clear spatial location and boundary; a raster data model, instead, uses a grid and grid cells to represent continuous features.

Modern web-based map services, such as Google Maps and Microsoft Bing Maps, use a combination of raster data and vector data to deliver realistic map visualizations. For example, raster images from satellite imagery are used as a basic background map. Vector layers consisting of boundaries, roads and place markers are added on top for the purpose of searching

and path finding.

In this work, a vector based data structure is implemented to represent geographic information, similar to the data structure in Open Street Map (OSM) [24], an open-source, public-edited online map service. Different from OSM, which is designed for general maps covering both urban and rural areas, the data structure implemented here focuses on urban situations where the most interesting objects are streets and buildings.

Figure 3.22 shows the class hierarchy of the proposed data structure for storing city maps. All classes are inherited from the *base_obj*, which has unique ID for searching and referencing. The *node* class stores the location of a point in both local 3D coordinate with x, y, z values and in an Earth coordinate system with a latitude and longitude.

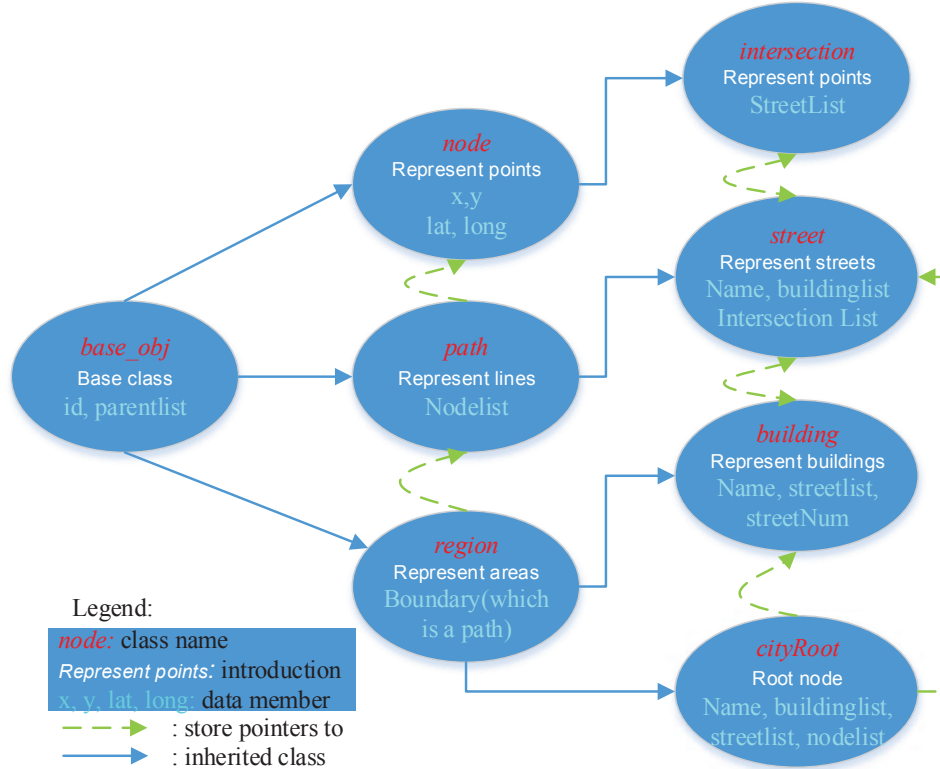


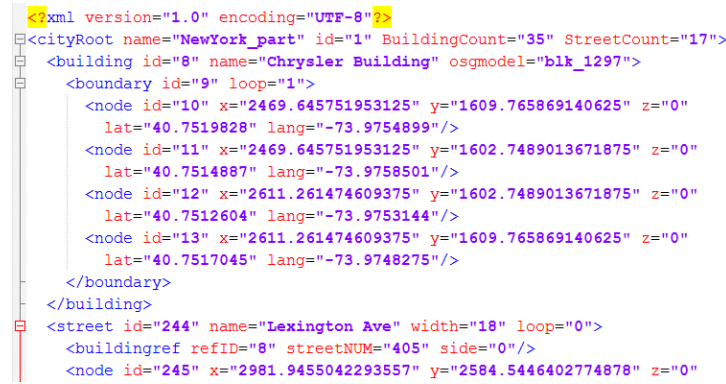
Figure 3.22: Class hierarchy of the map data structure.

The *path* class stores line segments formed by multiple points, which can then be used to define the boundary of a *region* class object, or can be used to define a *street*. The *building* class derives from class *region* and represent a building, with a list of pointers to its adjacent streets. In the same way, a *street* object has a list of pointers to all buildings along that street.

Also, a list of pointers to all its intersections is stored in a *street* object. The *intersection* class is derived from class *node* and stores pointers to connected streets.

An object of class *cityRoot* is created for a city and stores pointers to all its streets, buildings, intersections, etc. In this way, the entire map can be organized into a tree structure, with *cityRoot* as its root, *streets* and *buildings* as its branches, and *nodes* as end leaves.

To store the map data, a file input and output interface is implemented using the Extensible Markup Language (XML) [15], which is an open standard document encoding format that focuses on simplicity and generality. Figure 3.23 gives an example of a saved map file.



```
<?xml version="1.0" encoding="UTF-8"?>
<cityRoot name="NewYork_part" id="1" BuildingCount="35" StreetCount="17">
  <building id="8" name="Chrysler Building" osgmodel="blk_1297">
    <boundary id="9" loop="1">
      <node id="10" x="2469.645751953125" y="1609.765869140625" z="0"
        lat="40.7519828" lang="-73.9754899"/>
      <node id="11" x="2469.645751953125" y="1602.7489013671875" z="0"
        lat="40.7514887" lang="-73.9758501"/>
      <node id="12" x="2611.261474609375" y="1602.7489013671875" z="0"
        lat="40.7512604" lang="-73.9753144"/>
      <node id="13" x="2611.261474609375" y="1609.765869140625" z="0"
        lat="40.7517045" lang="-73.9748275"/>
    </boundary>
  </building>
  <street id="244" name="Lexington Ave" width="18" loop="0">
    <buildingref refID="8" streetNUM="405" side="0"/>
    <node id="245" x="2981.9455042293557" y="2584.5446402774878" z="0">
```

Figure 3.23: An example of a XML map file.

3.8.2 Linkage Between Map Data and 3D City Data

In a traditional digital map, 3D buildings are added “on top of” the map by simply assigning 3D models their geographic location, such as latitude, longitude and direction. This scheme is used in osgEarth, an open source digital globe [38] and Google Earth. User interactions such as location-based searching and path finding are fundamentally dealing with 2D map information. This approach is designed mainly for adding 3D visualization to a 2D map, and works well for 2D desktop environments.

However, in an immersive VR system, interactions are often 3D in nature. Therefore, a stronger link between 3D data and 2D map data is needed. For example, when selecting a building or a street, the application needs to know what the user is selecting, its name, its location, and its neighbors. In a traditional approach, the location of selected objects in the 3D coordinate system is known, and is mapped into global coordinates. Then, a search in the map database using this global location returns what object is in that place. This process could be significantly faster if we directly link 3D objects to the corresponding objects in the map

database.

In this work, an explicit linkage between 3D buildings, street meshes and their corresponding objects in the map data set is proposed and implemented. As shown in Figure 3.24, 3D objects and map objects are linked together by pointers, which are initialized at the application launch time by checking model names and ID numbers stored in 3D city and map files.

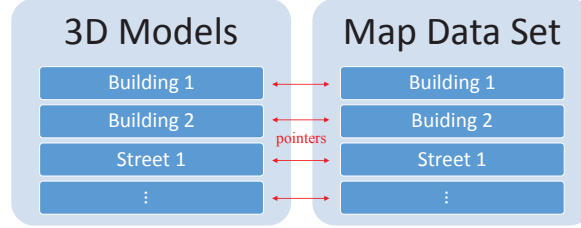


Figure 3.24: Linkage between 3D city and map data.

3.8.3 Path Finding by Dijkstra's Algorithm

Path finding, also known as navigation in some online map services, is a frequently used function when people plan trips to unfamiliar areas. A good path finding system considers the speed limit on different streets, realtime traffic, or even weather conditions, as well as distances, to find a way to reach a destination in the shortest time.

Dijkstra's algorithm is one of the most widely used graph search algorithms that find the shortest path between 2 points on a graph with non-negative edges [19] [59]. It is implemented on top of our map data structure and is used for path finding.

To give a brief description of Dijkstra's algorithm, the terms *vertex*, *edge*, *graph* will be used, which correspond to intersection, street, and map in a city, respectively. Figure 3.25 gives an example of a graph. There are 6 vertices, denoted as a , b , c , d , e , and f . Edges connect some of the vertices together, with a number denoting the *cost* value, in our case the distance between 2 vertices. From an initial vertex, for example vertex a , the algorithm will find the shortest distance and path to the initial vertex, in an iterative way:

- 1. Assign to every vertex a tentative distance D : 0 for initial vertex and infinity for all others.
- 2. Mark all vertices unvisited; Set the initial vertex as the current vertex.
- 3. For the current vertex, consider all of its unvisited neighbors and calculate their tentative distances, by adding the edge's distance to the current vertex's distance. If the

newly calculated tentative distance is smaller than the current assigned value, assign the newly calculated one and set the current vertex as that neighbor's previous vertex. For example, in the first iteration in Table 3.2, b and c are assigned new distance values 2 and 3, since they are smaller than the original infinity, and current vertex a is set to be the previous vertex along the shortest path to b and c .

- 4. After all the neighbors are considered, mark the current vertex as visited, which will not be checked again.
- 5. If the destination vertex has been marked visited or if the smallest tentative distance among unvisited vertices is infinity (when there is no connection between the initial vertex and remaining unvisited vertices), then stop. The algorithm has finished.
- 6. Select the unvisited vertex with the smallest tentative distance, and set it as the new "current vertex", then go back to step 3.

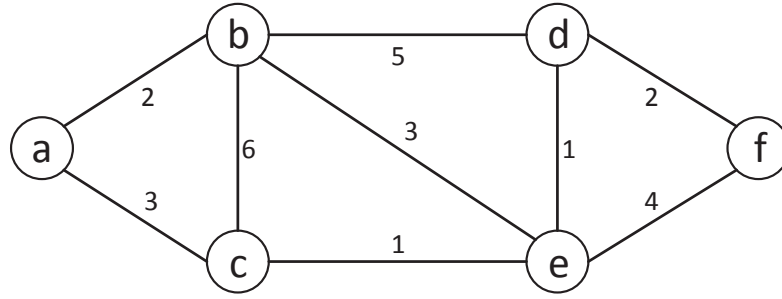


Figure 3.25: An example graph to demonstrate Dijkstra's algorithm.

Iteration	Current	Unvisited	$D(a)$	$D(b)$	$D(c)$	$D(d)$	$D(e)$	$D(f)$
0		$\{a, b, c, e, d, f\}$	[0]	∞	∞	∞	∞	∞
1	a	$\{b, c, d, e, f\}$		[2]	3	∞	∞	∞
2	b	$\{c, d, e, f\}$			[3]	7	5	∞
3	c	$\{d, e, f\}$				7	[4]	∞
4	e	$\{d, f\}$				[5]		8
5	d	$\{f\}$						[7]
6		\emptyset						

Table 3.2: Example iteration of Dijkstra's algorithm for Figure 3.25: each row represents an iteration, with the shortest distance in $[]$ brackets, which will serve as current in next iteration.

Table 3.2 shows all the iterations of Dijkstra’s algorithm applying to Figure 3.25. For a graph G with vertices v_0, v_1, \dots, v_n , and edge distance $w(v_i, v_j) > 0$, to find the shortest path between v_{start} and v_{end} , Dijkstra’s algorithm in pseudo-code is as follows:

```

for  $i = 1$  to  $n$  do
     $D(v_i) = 0$ 
     $P(v_i) = v_i$ 
    add  $v_i$  to  $Q$ 
end for
 $D(v_{start}) = 0$ 
while  $v_{end} \in Q$  and  $Q$  not empty do
     $u =$  vertex in  $Q$  with minimum  $D(u)$ 
    remove  $u$  from  $Q$ 
    for each neighbor  $v$  of  $u$  do
        if  $D(u) + w(u, v) < D(v)$  then
             $D(v) = D(u) + w(u, v)$ 
             $P(v) = u$ 
        end if
    end for
end while
return  $D(v_{end})$  and  $P$  for previous path

```

A data conversion is performed at application initialization to construct a graph from our map data structure. Intersections are used as vertices. For each street that pass an intersection, the previous and next intersection on that street will be the neighbors. Distances are calculated by a sum of distances among each intermediate node on street, instead of calculating the straight line distance between 2 neighboring intersections. This takes into consideration that some streets are curved.

3.8.4 High Level Interactions Enabled by Map

After the integration of the map data structure and the path finding algorithm, our virtual city application has the interaction ability that constrains to geographic information.

The selection of buildings benefits from this integration. For example, the user can select those buildings on one side of a street, by first selecting that street using ray casting technique, and then finishing the selection by calling the corresponding function in the menu. The user can also select buildings or streets based on a city block, which is usually defined as the smallest area that is surrounded by streets. This is because that an object of the *region* class (as in

Figure 3.22) can store the boundary of city blocks, as well as pointers to all the inside buildings and streets.

Navigation also benefits from the integration of the map data and path finding algorithm. The user is able to define an arbitrary route, which means a line that connecting any number of user defined points. The *path* class enables this definition. For example, a user can make a route from the ground outside his apartment building under construction, all the way to the window in his suite, looking outside, and then have a virtual walk through along that route, to see what it will look like from that window.

Defining a route can also be automatically done with the path finding algorithm. For example, a user may want to have a virtual tour from the Wall Street to the World Trade Center, both in New York City. He/she can simply select the beginning point and the end point. The path that having the shortest distance will be automatically generated, probably along the Broadway and Liberty Street. Then the user can go through the virtual tour along this path.

3.9 Summary

This chapter covers the methods and contributions used or proposed in our virtual city system. Hardware and software configurations are first introduced, followed by arguments on the choice of input devices.

The second section of this chapter describes the realtime gesture recognition system based on a Hidden Markov Model. Methods for system control are proposed in the third section, featuring a button menu, and 2 gesture controlling techniques, touch menu and slide menu, respectively.

Section 3.4 to 3.7 introduces the proposed interaction techniques for each of the basic elements of a 3D application: navigation, selection, manipulation and visibility issues.

The last section of this chapter discusses application specific constraints, which could further improve the user experience. A map data structure that links with 3D city data is proposed to achieve the ability of higher level interaction. A path finding function based on Dijkstra's algorithm is also implemented on top of map data set.

As a system application that exploits various sources of speciality, and integrates different kinds of advanced algorithms, the proposed CAVE based virtual city has 3 main contributions. First, a HMM based pattern recognition system that detects 3 dimensional gestures is proposed to assist user interaction in a VR environment. Second, novel interaction techniques are proposed for system control, navigation, selection and manipulation in immersive VR environments. To be specific, gesture based techniques, controlling a 3D menu by touching and sliding,

navigation triggered by posture and a waving hand, selecting virtual objects by painting and stretching, and manipulation by two-handed gestures are proposed. A third contribution of this work is the integration of application specific constraints, which focus on interactions in a virtual city scenario, that is implemented for selection, manipulation, as well as high level navigation and path finding.

Chapter 4

Experiments and Results

This chapter will describe the experiments that were performed to evaluate our proposed methods. The results of the experiments are presented and analyzed.

For the HMM based gesture recognition system, an independent test on a collected gesture database is performed first. The test achieves an overall recognition rate of 96.8%. However, while independent off-line testing is easy to conduct and appropriate for testing algorithms, it does not necessarily reflect realtime performance. This is because in a recorded gesture database the start-point and end-point of a gesture sequence are known, which is not true while running in realtime.

Therefore a realtime recognition test is performed together with the empirical user study, which is conducted to evaluate the proposed interaction techniques in the CAVE. Participants are invited to perform several tasks that involve navigation, selection, and menu control using different techniques. User feedback is then collected via questionnaires and used to evaluate the proposed interaction techniques. The results show that while new users tend to have no clear preferences, expert users prefer gesture based techniques over traditional ones. Also, we count the failures of gesture recognition while the user is doing their tasks and record a realtime gesture recognition rate of 89.11%.

A quantitative user study is done to evaluate the proposed two-handed gesture manipulation, compared with traditional wand manipulation. Times and distortions to finish a manipulation task are collected as evaluation metrics. The results show that two-handed gesture is significantly faster than traditional wand manipulation.

4.1 Independent Gesture Recognition Test

4.1.1 Gesture Database

As discussed in section 3.1.3, there are only 3 gestures used to control the virtual city application. However, to test the capability of the recognition system while leaving room for future updates of controlling gestures, a database containing 7 gestures is collected. The definition of gestures is shown in Figure 4.1.

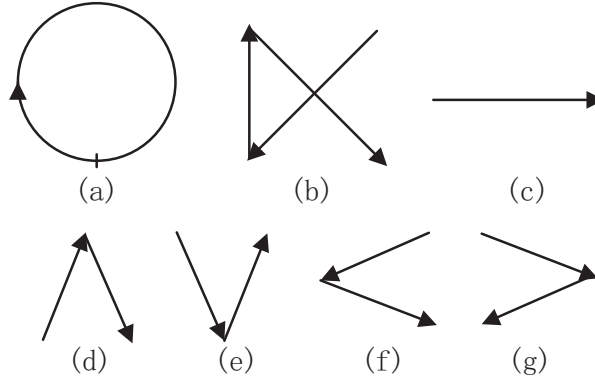


Figure 4.1: Gesture definition in a recorded database: (a) Circle; (b) Crossing; (c) Sliding; (d)-(g) Arrows.

The database was collected from 5 participants, among which there were 4 males and 1 female. Each of them performed 25 samples for each of the 7 gestures. Therefore a database containing 875 samples was collected.

4.1.2 Gesture Recognition Results

Ten samples are randomly selected for each gesture as training data, while the remaining 115 samples are used as testing data. Under this testing arrangement, we have an average recognition rate of 96.8%. The confusion matrix is shown in Figure 4.2.

In Figure 4.2, each column represents the recognition rate of the corresponding gesture and the rate of incorrect recognition as other gestures, while the black color represents recognizing no gestures (i.e. a miss). For example, Figure 4.2 shows gesture (a) is recognized 100% correctly, while around 95% of the time gesture (b) is recognized correctly, with a 5% miss rate. We can see from these results 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)

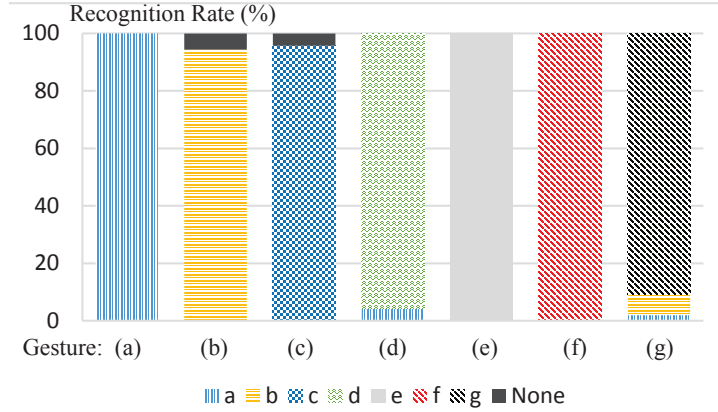


Figure 4.2: Graphical demonstration of confusion matrix in the independent recognition test.

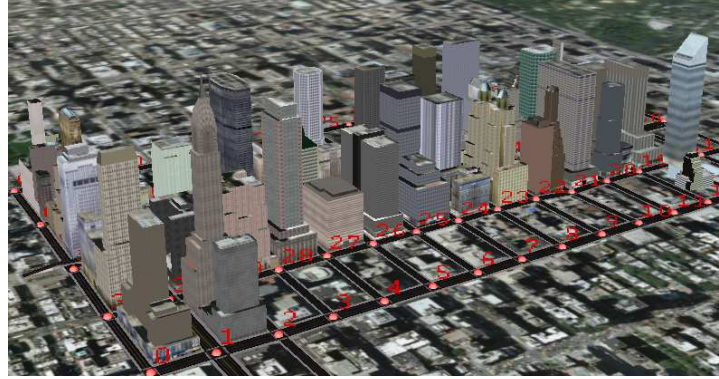
are occasionally mis-classified as other gestures. This may be due to the similarity among gesture definitions. For example, the first half of the right arrow resembles the sliding gesture.

4.2 3D City Data for Experiment

A group of 3D models are used for the prototype applications and the user study. These models include 35 buildings in Midtown Manhattan, New York City, USA, spanning from 41st St. on the south side to 54th St. on the north side. In the east-west direction, these buildings are located in the middle of 3rd Ave. and 5th Ave. Figure 4.3(a) shows a screenshot of the 3D data, with all the intersection points marked with dots and numbers. Figure 4.3(b) shows a map from OpenStreetMap [24] [45] covering the same area.

4.3 User Study for Interaction Techniques

In the field of user interface design, designers usually follows a procedure that iteratively improves the design through informal and formal user studies [70] [11], as summarized in Figure 4.4. After analyzing user tasks and finishing an initial design, there are 2 stages of evaluation, *heuristic evaluation* and user studies in a larger scale. Heuristic evaluation is a method in which several expert users evaluate a UI design by applying a set of heuristics or design guidelines [44] [11]. After rounds of heuristic evaluations and refinements, the design is then evaluated by user studies that involve more users from diversified backgrounds. If problems are reported, the design would be sent back for refinements again, until a satisfactory result is achieved.



(a) 3D data used for testing



(b) Open Street Map showing the same area [45].

Figure 4.3: Sample city data.

In this work the interaction design also undergo iterative user evaluation, both informal and formal, and a refinement process. Many details in the techniques described in chapter 3 have been modified multiple times based on feedbacks from user studies. For example, gestures chosen for menu control were originally designed using gesture *a*-circle, *b*-crossing and *c*-right sliding defined in Figure 4.1. However users reported experiencing lagging due to the relatively long length of these gestures. Users commented that they desired simple and short gestures. Therefore, the design in section 3.3.2 was finalized with 3 sliding gestures for menu triggering, confirming and canceling.

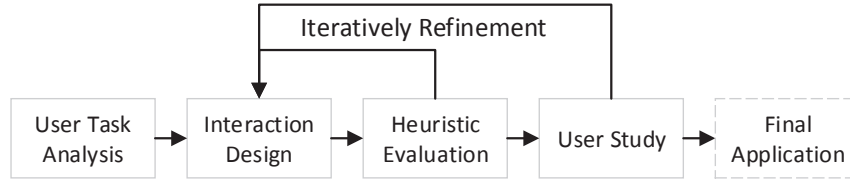


Figure 4.4: UI design procedure.

4.3.1 User Study Design

After design refinements based on feedback from expert users, a user study involving a simpler experience for novice users was conducted. Interaction techniques for comparison are summarized in Table 4.1. There are 4 interaction categories, each containing 3 techniques for comparison.

Category	Technique	Code*
Menu Control (section 3.3)	Button	C1
	Touch	C2
	Slide	C3
Navigation (section 3.4)	Joystick	N1
	Posture Triggered Flying	N2
	Waving Hand	N3
Selection (section 3.5)	Ray Casting	S1
	Paint Selection	S2
	Stretch Selection	S3
Visibility (section 3.7)	Collapse	V1
	Semi-transparent	V2
	Wireframe	V3

Table 4.1: Interaction techniques for comparison. * Code is a short representative of a technique name to simplify the description of the user study procedure.

Tasks Description

In order to cover all 4 interaction categories in a reasonable way, participants of the user study were asked to complete 2 different tasks, each involving 2 categories of interaction. While a detailed description of user study procedure will be given in Appendix 2, the 2 tasks are summarized here:

- Task-1 Menu Control and Building Selection. The user needs to: use the control menu to choose a selection technique; select a specified building; clear that selection from the menu; then select 6 other designated buildings; and clear the selection again using menu.
- Task-2 Navigation and Visibility. In this task, the user is asked to navigate along a designated path, and then deal with a high building that blocks a desired view.

In order to minimize the influence of unrelated variables, the number of degrees of freedom in the test application is controlled in a careful way. For example, in the selection task, participants are not able to move or change the size of buildings by manipulating the whole scene, though doing this could facilitate the selection process with a better viewing angle. However, if manipulation is allowed, the evaluation of the selection techniques would be affected, since different participants may have different habits or skills of manipulating while selecting. Also the scale(size) of the virtual city is fixed in the navigation task. So the user can focus on the 3 techniques of “flying” in the city.

Another issue in an empirical user study is that, the order of the test could influence the results. To be more specific, for a new user, the experience gained in testing the first technique could have some effects such as improvement in the performance of the next technique. The Latin square [14] [35] is a method commonly used to deal with this problem. The idea is to divide participants into random groups, use different testing orders for different groups. For example, Table 4.2 shows the Latin Square used for the order of menu control techniques. Participants are randomly divided into 3 groups. The first group uses the order shown in first row: C1, C2, C3, which correspond to button, touch and slide, respectively, as assigned in Table 4.1. The second and third group will use the second and third row in the Latin square. Latin squares are designed such that an element occurs only once in each row and in each column.

C1	C2	C3
C2	C3	C1
C3	C1	C2

Table 4.2: 3×3 Latin square used for the menu control techniques.

A detailed procedure of the user study can be found in Appendix 2.

Evaluation Metrics

A Likert scale [39] [34] is a common way of collecting participants’ attitudes in an empirical study. To use a Likert scale, the users are given a series of statements and asked to indicate

whether they agree or disagree with each statement. For example, the following statements are used to evaluate the joystick navigation technique:

1. The “Joystick” technique is easy to learn and use.
1. The “Joystick” technique can take you from one point to another effectively.
3. The “Joystick” technique, as a whole, is a good navigation method.

Strongly Disagree	Disagree	Somewhat Disagree	Neutral	Somewhat Agree	Agree	Strongly Agree
1	2	3	4	5	6	7

The participant is asked to give a scale number indicating his/her degree of agreement or disagreement to each statement, as shown above.

For each technique in the navigation and menu control category, participants are asked to evaluate 3 statements, corresponding to the easiness, effectiveness, and overall experience of a technique. For a technique in the selection category, the evaluation involves its overall experience in selecting single object and multiple adjacent objects. For techniques in the visibility category, their overall experiences are evaluated.

In addition to a Likert scale, the participants are also asked to choose a best technique for each category. For example:

1. Please choose a menu control technique you think is best overall, ☐Button, ☐Touch, or ☐Slide?
2. Please choose a navigation technique you think is best overall, ☐Joystick, ☐Posture Triggered Flying , or ☐Waving Hand?

Appendix 3 gives the full questionnaire used in this user study.

Experiment Setup

There were 12 participants in this user study, including 3 females and 9 males, with an average age of 27.

A participant’s previous experience with 3D user interfaces and virtual reality is divided into 3 levels:

1. Totally new to VR, with very limited experience with 3D applications like video games or modeling software.

2. Medium previous experience, for example users that often play with 3D modelling softwares and other 3D applications, and have some knowledge about VR but have never played with a CAVE.
3. Rich previous experience in 3DUI and VR, including developing and testing 3D or VR applications.

In the 12 participants, 6 were in the first level, 3 in the second, and 3 in the third. In our statistical analysis of the results, level 1 participants are considered as novice users, while level 2 and 3 are considered as expert users.

A participant was first introduced to the system for about 5 minutes. Then, the participant was asked to finish 2 tasks defined previously. For each task, different techniques were tested, using the order from the Latin square. Before performing a task using a designated interaction technique, the participant had 2-5 minutes to learn and practise that interaction technique. After finishing all tasks, the user was then asked to fill out the questionnaire. The experiment procedure, questionnaire, and user responses can be found in Appendix 2, 3, and 4.

4.3.2 Results on Menu Control

Table 4.3 shows the mean and standard deviation of the participants overall scores for different menu control techniques.

Interaction	Group	Mean	Standard deviation
C1-Button	Expert	5.5	0.96
	Novice	6	1.00
	All	5.75	1.01
C2-Touch	Expert	6.17	1.21
	Novice	5.67	0.94
	All	5.91	0.95
C3-Slide	Expert	5.5	1.50
	Novice	4.67	1.37
	All	5.08	1.49

Table 4.3: Overall score for menu control techniques.

The statistics in Table 4.3 show that expert users give 12.2% higher scores for touch menu than for button menu and slide menu, while their average scores for slide menu and button menu are the same. However, for new users, the button menu gets the highest scores, followed by the touch menu and the slide menu.

An Analysis of Variance (ANOVA) [14] [35] is performed on the overall scores to analyze the significance of differences. The result is shown in Table 4.4.

Source of variation	Sum of squares (SS)	Degree of freedom (df)	Mean Square (MS)	F	p
Interaction technique	4.667	2	2.333	1.54	0.2299
Total	54.75	35			

Table 4.4: ANOVA results for overall score for menu control techniques.

However, the ANOVA result in Table 4.4 shows the difference is not statistically significant. This is caused by similar average scores and relatively large deviation among participants. The probable reason for this result could be that the difference of the user experience among these 3 techniques are not large enough to cause a significant deviation of scores. We should also note that different participants have different criteria on giving scores, resulting in a large variance, or using a 7 point Likert scale limits the range to 1 – 7, resulting in similar average scores.

On the other hand, the ANOVA only gives results about statistical significance, which evaluates the probability that the results are due to chance. This is not the same as practical significance [35]. From another perspective, one may check responses to the “best choice” question, as given in Table 4.5.

Group	C1-Button	C2-Touch	C3-Slide
Expert	0%	66.67%	33.33%
Novice	50%	33.33%	16.67%
All	25%	50%	25%

Table 4.5: Percentage of choices for best menu control technique.

We can see from Table 4.5 that none of the participants in the expert group chose the button menu, however half of the novice group chose the button menu. This might be due to the fact that new users are more familiar with button interactions through their previous usage of similar keyboard controlled menus on computers, TV sets, ATMs etc. After getting used to 3D environments and interactions, as shown in choices by the expert group, gesture based techniques start to show their strength.

4.3.3 Realtime Gesture Recognition Records

While participants doing tasks that involve using a gesture controlled menu, whether touch or slide, the performance of realtime gesture recognition is recorded. To be more specific, let

A denotes the total number of attempts that a user performs a gesture to be recognized; F denotes the total number of false positives, which means a gesture is recognized while the user does not perform one; S denotes the number of successful recognitions. During the test, A , F , and S values are recorded for each user and then used to calculate the recognition rate S/A and false positive rate F/A . The average rates are given in Table 4.6.

The possible relationship between recognition rates and a user’s evaluations of the slide menu is also investigated. Figure 4.5 shows the plot of overall scores given by a user for the slide menu, versus the gesture recognition rate for that user.

Group	False Positive Rate	Recognition Rate
Expert	1.05%	91.05%
Novice	3.41%	87.32%
All	2.28%	89.11%

Table 4.6: Realtime gesture recognition rate for different groups.

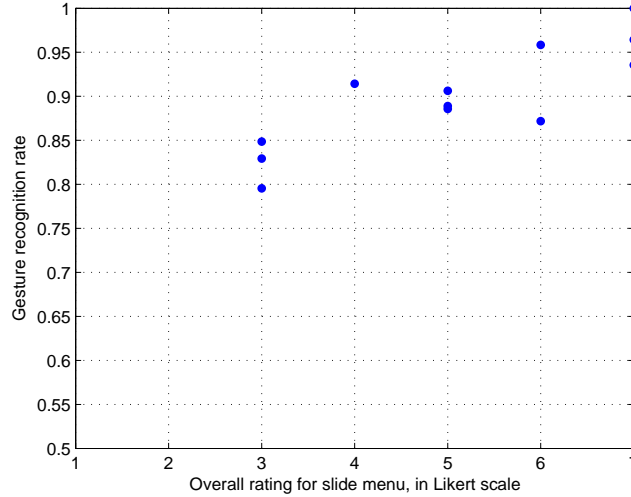


Figure 4.5: Relationship between gesture recognition rate and user’s rating for slide menu: the 2 variables have a correlation of 0.851.

From Table 4.6, we can see that our system achieves a 89.11% realtime recognition rate, which is lower than the independent off-line test in section 4.1. This is reasonable since a realtime system has no start and end point information available as in a recorded gesture database, and it is subject to more uncertainties such as an unstable frame rate and noise.

It is also observed that the expert group has a lower rate of false positives and a higher rate

of successful recognition than the novice group. This matches the result in Table 4.3 that the expert group gives higher scores to the slide menu than the novice group. Figure 4.5 further confirms the relationship between recognition rates and subjective scores (scores for the slide menu are used because the slide menu is controlled fully by recognized gestures). These 2 variables have a correlation coefficient of 0.851, indicating a strong linear relationship. This indicates that the user’s experience could be improved by further increasing the accuracy of the gesture recognition system.

4.3.4 Results on Navigation

Table 4.7 shows the mean and standard deviation of participants’ overall scores for 3 navigation techniques. The corresponding ANOVA result is given in Table 4.8.

Interaction	Group	Mean	Standard deviation
N1-Joystick	Expert	5.83	1.07
	Novice	5.5	0.96
	All	5.67	1.03
N2-Posture Triggered Flying	Expert	5.67	1.25
	Novice	6.33	0.75
	All	6	1.08
N3-Waving Hand	Expert	5.83	1.07
	Novice	4.5	2.34
	All	5.17	1.95

Table 4.7: Overall score for navigation techniques.

Source	SS	df	MS	F	p
Navigation technique	4.222	2	2.111	0.96	0.3922
Total	54.75	35			

Table 4.8: ANOVA results for overall score of navigation techniques.

Again, average scores for the 3 techniques are close to each other. ANOVA also shows the difference is not statistically significant ($F = 0.96$, $p = 0.39$). This result somehow indicates that using any of the 3 tested navigation techniques does not have too much impact on a user’s experience. Answers to the question “choose a best technique” are summarized in Table 4.9.

We can see from Table 4.9 that a higher percentage of novice users choose N2-Posture Triggered Flying as their favorite, which matches the observation in Table 4.7 that novice users give higher scores for N2, than expert users. This might be caused by the simplicity of the N2

Group	N1	N2	N3
Expert	16.67%	33.34%	50%
Novice	0%	50%	50%
All	8.33%	41.67%	50%

Table 4.9: Percentage of choices for best navigation technique.

technique: simply raising the left hand leads to a navigation action. In comparison, the joystick has 2 degrees of freedom to control for navigation.

Another phenomenon worth noting is that, though the waving hand technique does not get the highest score, half of participants in both the expert and novice groups still choose it as their favorite. And expert users give 29.56% higher scores to the waving hand technique than novice users. This indicates the waving hand navigation is harder to learn than the other 2 techniques. However, once mastered, waving hand technique starts to gain preference.

4.3.5 Results on Selection

Evaluation of 3 selection techniques are done separately, for selecting single buildings, or multiple adjacent buildings. The mean and standard deviation, as well as ANOVA results on overall scores for selecting single buildings are given in Table 4.10 and Table 4.11. Results for selecting multiple buildings are shown in Table 4.12 and Table 4.13.

Interaction	Group	Mean	Standard deviation
S1-Ray Casting	Expert	7	0
	Novice	6.5	0.76
	All	6.75	0.59
S2-Paint	Expert	5.5	1.26
	Novice	5	1.63
	All	5.25	1.48
S3-Stretch	Expert	3.83	1.21
	Novice	3.83	1.77
	All	3.83	1.52

Table 4.10: Overall score for single object selection techniques.

Scores in the selection task category demonstrate a more significant difference among the 3 techniques. Ray casting gets the highest scores for selecting single building, followed by paint selection and stretch selection. The differences are statistically significant ($F = 14.48, p = 3.055 \times 10^{-5}$).

Source	SS	df	MS	F	p
Selection technique	51.056	2	25.5278	14.48	3.055×10^{-5}
Total	109.222	35			

Table 4.11: ANOVA results for overall score of single object selection techniques.

Interaction	Group	Mean	Standard deviation
S1-Ray Casting	Expert	4.83	0.69
	Novice	4.5	1.5
	All	4.67	1.17
S2-Paint	Expert	6.67	0.47
	Novice	6.5	0.76
	All	6.58	0.64
S3-Stretch	Expert	5.83	1.07
	Novice	5.33	1.37
	All	5.58	1.25

Table 4.12: Overall score for multiple adjacent objects selection techniques.

For selection of multiple adjacent buildings, paint selection gets the highest scores, while ray casting scores the lowest, with a statistically significant difference ($F = 8.99, p = 0.0008$). This is due to the low efficiency of ray casting, since it only selects one object at a time, while paint selection and stretch selection are designed for selecting multiple objects.

4.3.6 Results on Visibility

Table 4.14 gives the scores for 3 visibility techniques. Table 4.15 shows the ANOVA result.

The differences in scores for the 3 visibility techniques are not statistically significant, and are due to chance with a high probability ($F = 0.06, p = 0.9393$). This makes sense that preference in this category is highly determined by personal choice.

Source	SS	df	MS	F	p
Selection technique	22.0556	2	11.0278	8.99	0.0008
Total	62.5556	35			

Table 4.13: ANOVA results for overall score of multiple object selection techniques.

Interaction	Group	Mean	Standard deviation
V1-Collapse	Expert	6.17	0.69
	Novice	5.33	0.75
	All	5.75	0.83
V2-Semi-Transparent	Expert	5.67	0.94
	Novice	5.67	0.75
	All	5.67	0.85
V3-Wireframe	Expert	5.33	1.49
	Novice	5.83	1.46
	All	5.58	1.49

Table 4.14: Overall score of visibility techniques.

Source	SS	df	MS	<i>F</i>	<i>p</i>
Visibility technique	0.1667	2	0.0833	0.06	0.9393
Total	44	35			

Table 4.15: ANOVA results for overall score of visibility techniques.

4.4 Quantitative Comparison of Manipulation Techniques

In addition to the qualitative user study described in the previous section, a quantitative user study is also conducted, to compare the proposed two-handed gesture manipulation with traditional wand manipulation.

4.4.1 User Study Design

Task Description

The task is set in an interior design scenario, where the user is asked to arrange furniture in a virtual building. As shown in Figure 4.6(b), a virtual computer is located initially on a table. The initial state represents the position, the orientation, and the size of the object before manipulation while the target state represents the expected state after manipulation. The user is asked to put the virtual computer to the target state displayed as wire frame on another table. The manipulation involves translating, rotating and scaling.

Evaluation Metrics

The performance of a manipulation technique can be evaluated with two metrics: distortion and completion time. Distortion is a time variant variable that represents the deviation between

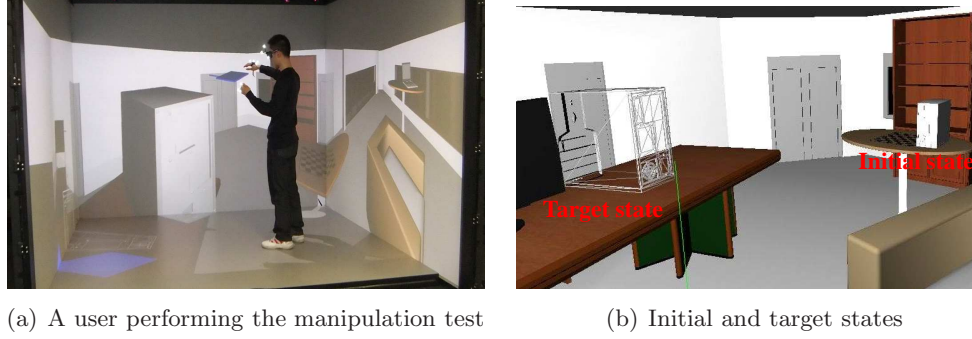


Figure 4.6: Task description for quantitative manipulation test: user is asked to put a virtual computer from its initial state to a target state.

the current state and the target state of the object. For a virtual object, let \mathbf{V} denote the set of all vertices on the object represented by a 3D polygonal mesh. The distortion $D(t)$ at time t , in the form of Mean Squared Error (MSE), is given by:

$$D(t) = \frac{1}{|\mathbf{V}|} \sum_{i=1}^{|\mathbf{V}|} (d_i(t))^2 \quad (4.1)$$

where $|\mathbf{V}|$ represents the number of vertices in the object to be manipulated, and $d_i(t)$ represents the distance of the i -th vertex at time t with the target position of that vertex. The distortion curve, measured in squared feet (f^2) versus time in seconds, is dependent on the the arrangement of initial and target state, as well as the size of object. However, as the task is fixed during the whole test, we can directly use the computed distortion for comparison without additional normalization. In our task setup (Figure 4.6), the initial distortion is $47f^2$.

Another metric, the completion time, is defined as the time spent manipulating measured from the initial state until a required satisfactory distortion value is reached, which we set at $0.1f^2$.

Experiment Setup

We invited 15 students from Ryerson University to participate in this test, including 2 females and 13 males, 1 left-handed and 14 right-handed. The participants were divided into 2 groups: 8 of them were in the expert group, who had been developing or using VR applications for more than 6 months and are familiar with the interaction techniques to be tested; another 7 participants were in the novice group, who had no previous experience on 3D interactions and VR systems.

Both the wand manipulation and two-handed manipulation were tested for the same task. Each participant performed the task, 2 times with wand manipulation, and 2 times with two-handed manipulation.

4.4.2 Results for Manipulation

The distortion curve, which represents the change of distortion versus time, as described previously, is collected during the user test. The distortion curves for the expert group and novice group are averaged separately. The average distortion curve for both groups are given in Figure 4.7.

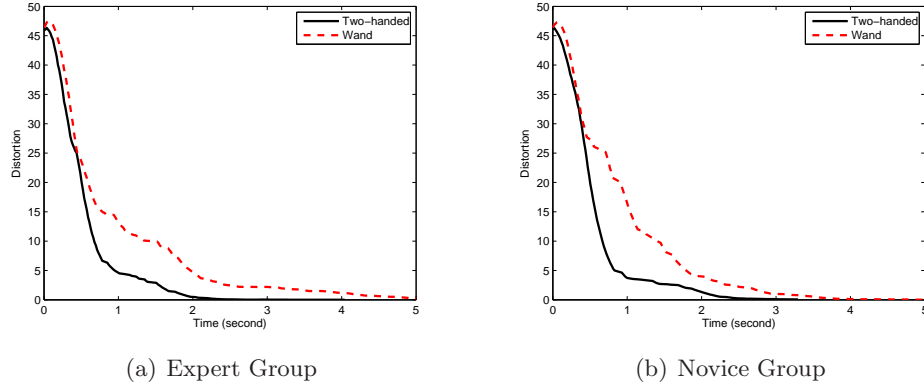


Figure 4.7: Distortion curves collected for the manipulation test.

We can see from Figure 4.7 that the distortion curves for two-handed manipulation decrease and converge faster towards zero than wand manipulation, for both the expert and novice groups. The fastest deviation of distortion between two interaction methods happens around $t = 0.7s$, where the users generally start to adjusting the size of target object. This observation meets the problem described in section 3.6, that scaling can not be integrated together with rotation and translation in wand interaction. Since buttons are used for scaling, it takes more time to adjust the size than a two-handed gesture does.

The completion times for both manipulation techniques are also recorded. The mean and standard deviation values are given in Table 4.16. Table 4.17 shows the analysis of variance (ANOVA) results.

The result in Table 4.16 shows smaller completion times for two-handed manipulation than wand manipulation, in both expert and novice groups. The ANOVA results shows the difference is statistically significant ($F = 67.698$ and $p = 5.908 \times 10^{-9}$).

Interaction	Group	Mean(s)	Standard deviation
Hand	Expert	4.04	0.13
	Novice	4.34	0.54
	All	4.18	0.48
Wand	Expert	5.34	0.13
	Novice	5.77	0.32
	All	5.54	0.32

Table 4.16: Completion time (in seconds) recorded for the manipulation test.

Source	SS	df	MS	F	p
Interaction Technique	11.154	1	11.154	67.698	5.908×10^{-9}
Error	4.613	28	0.165		
Total	15.767	29			

Table 4.17: ANOVA results for completion time.

4.5 Summary

This chapter describes experiment setups and results to evaluate the proposed methods.

First, an offline performance of the gesture recognition system is tested in an independent test, with a gesture database containing 875 samples. A recognition rate of 96.8% is achieved.

Second, an empirical user study is performed to compare different techniques in each interaction category. The users' evaluations of menu control and navigation techniques do not show statistically significant difference among techniques. This might indicate there is no significant difference in a user experience between button based menu and gesture triggered menu, or between joystick navigation and gesture based navigation techniques. However, the expert user group shows preference for gesture based techniques over traditional ones, while the novice group does not. This could be attributed to the familiarity that novice users have with traditional ways of interaction, such as buttons and joysticks. After getting familiar with the gesture based interface, the user preference could shift, as the expert group indicates. An experiment confirming this trend will be considered in future work.

Realtime gesture recognition performance is also evaluated in the user study, with reduced gesture number and simplified definition of gestures. An overall recognition rate of 89.11% is recorded. Moreover, a relationship between realtime gesture recognition rate and a user's degree of satisfaction with gesture triggered menu control is recorded, which indicates a user's experience could be improved by developing more accurate gesture recognition systems.

Finally, a quantitative user study compares the proposed two-handed manipulation tech-

nique with the traditional direct wand manipulation. The completion time and a distortion measure are recorded for an object manipulation task. The results show that two-handed manipulation is significantly faster than wand manipulation.

Chapter 5

Prototype Applications

Based on the methods proposed in chapter 3, two prototype applications have been developed. Figure 5.1 overviews the structure of the prototype functions. Supported by data sources that provide 3D models, map data, and demographic data, the applications provide functions such as basic selection and manipulation, constrained selection and navigation based on geographic information, as well as viewing demographics.

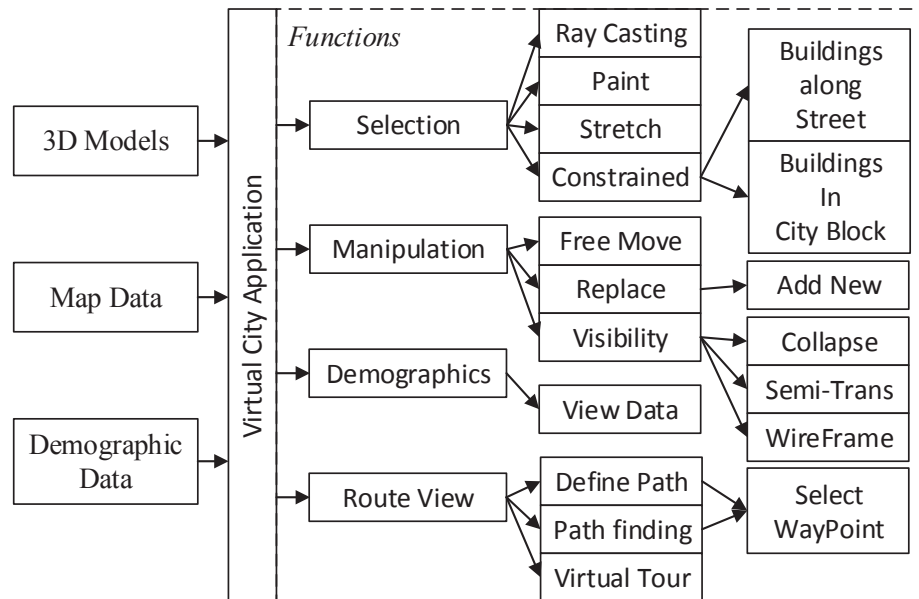


Figure 5.1: Structure of prototype applications.

5.1 Urban Planning

This application can be used as a decision support tool in urban planning. It gives realistic and immersive visualization of proposed constructions, which helps comparing, reviewing, or demonstrating project plans. Figure 5.2 shows an example where a user is viewing the demographic data of an area.

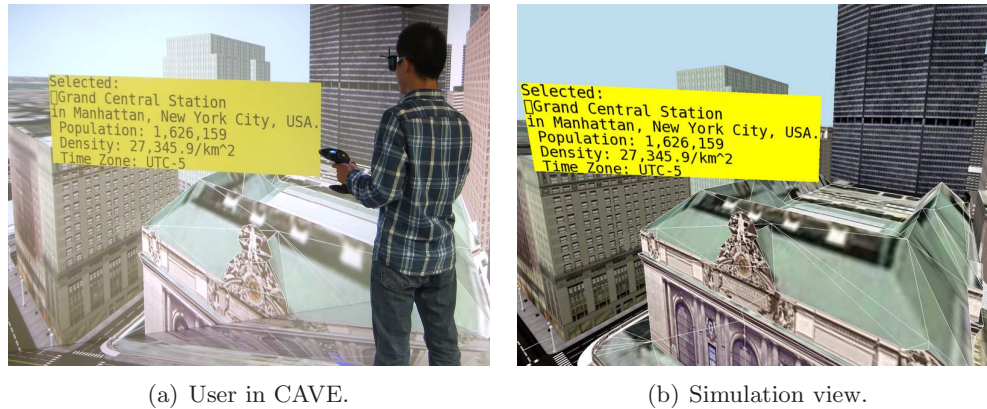


Figure 5.2: A user viewing demographic information for a selected area.

In the future, the virtual city experience could be further improved by adding data sources from various sensor networks that monitor a city's status, such as realtime traffic, temperature and weather conditions, power and water supply system status, etc. The feel of immersion, freedom of navigation and manipulation would help build a good visualization end node for a future smart city.

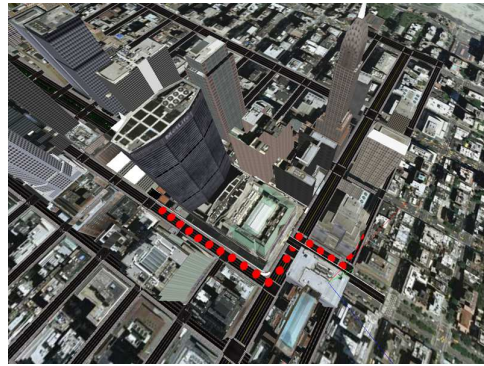
5.2 Virtual Touring

The best usage of an immersive virtual city might be virtual tourism. In the prototype application, a user can travel freely in a virtual city, or view the scene along pre-defined, manually defined or automatically searched routes. Figure 5.3 shows an example of a virtual tour along a manually selected path.

Audio information can also be added to give the user an improved feeling of presence, for example, from the background noise or from people and traffic in a downtown area, or an introductory footage to a place of attraction.



(a) User in CAVE.



(b) Simulation view of user defined route.

Figure 5.3: A user in the middle of a virtual tour.

Chapter 6

Conclusions and Discussions

6.1 Summary of This Thesis

In this thesis, designing user interactions for an interactive virtual city application in the immersive CAVE system is investigated. Guided by the goal of developing a natural and intuitive user interface that fits with the virtual city scenario, prototype applications are built, and interaction techniques are tested. The system implementation combines immersive visualization, optical tracking for gesture control, and realtime gesture recognition. Various interaction techniques are proposed for object selection, manipulation, navigation, system control, as well as solving visibility issues.

First, a Hidden Markov Model based realtime gesture recognition system is built as a supporting part for system inputs. Special care is taken for the normalization of 3D positional data and realtime implementation. To the best of our knowledge, this is among the first systems that use a 3D tracker based HMM to assist in dynamic gesture recognition in a VR environment.

The gesture recognition system is tested both by a gesture database and in realtime usage. In a database that contains 875 samples and 7 gesture classes, a recognition rate of 96.8% is achieved. In the realtime case, the gestures are simplified for better user experience, and an overall recognition rate of 89.11% is recorded. It is also discovered that users with previous experience tend to have better performance in realtime gesture recognition, and there is a linear relationship between gesture recognition rate and the user's evaluation of the system control interfaces that use those gestures.

Second, interactions that are primarily controlled by gestures are proposed for system menu control and navigation. To be more specific, a touch menu and a slide menu are proposed, in which the user can call out the menu by a gesture, and select menu items by "touching" or "sliding" gestures. In the navigation part, a "Posture Triggered Flying" technique and a

“Waving Hand” technique are proposed. In an empirical user study, these 2 menu control techniques are compared with traditional button controlled menu, while the 2 navigation techniques are compared with traditional joystick navigation. Results show no statistically significant differences among users’ evaluations of the compared techniques. However, a clear preference is found towards gesture based touch menu and slide menu in the expert user group. Furthermore, all user groups prefer the gesture based navigation techniques: the expert group prefer the “Waving Hand” navigation technique, while the novice group prefer the simpler “Posture Triggered Flying” technique.

Third, novel interaction techniques are designed for object selection and manipulation. A paint selection technique and a stretch selection technique are proposed in order to select multiple adjacent objects more efficiently. The empirical user study shows significant improvement of user evaluation for these 2 techniques over a traditional ray casting selection technique. Also, a two-handed manipulation technique is proposed to achieve more intuitive manipulation interactions. We conduct a quantitative user study to compare this technique with traditional direct wand manipulation. Evaluated by completion time and distortion of a manipulation task, the two-handed manipulation technique is significantly faster than wand manipulation.

6.2 User Interface Design for Immersive Virtual City

The interface designing procedure and user studies in this thesis work generate some insight on general UI design for an immersive VR environment. Users usually want an interface that is:

- Intuitive – the interface should match the users’ expectations or real world experience in such interactions. It should be easy to learn and use;
- Simple – users do not want to get lost in complex commands, modes, and control structures, so the interface need to be simple;
- Efficient – the interface should get things done quickly;
- Smart – the interface should be capable of understanding and supporting some higher level decisions. For example, it is more desirable in a manipulation task that the object “snap” to the target location when it’s close enough.

It is shown statistically that in our virtual city application, the gesture controlled menu and navigation have similar scores with traditional button and joystick techniques. This result is reasonable, since the most exciting experience in such an application is the immersive and realistic visualization. The users might not pay full attention to the experience of how controlling is done.

Based on these statistics, it could be argued that developing gesture interfaces is useless in this case. However, our user study also shows that the majority of users prefer gesture based navigation techniques, and experienced users prefer gesture based menu control. These preferences can be viewed in 2 perspectives. First, new users usually are more attracted by the simpleness of a gesture interface, for example, their preference to the “Posture Triggered Flying” technique. Second, some gesture interfaces need time and experience to be mastered, for example the gesture based menu control and “waving hand” navigation. Once getting familiar with these techniques, experienced users show a preference towards them, probably due to their effectiveness and intuitive mimicking of real world experiences.

6.3 Possible Future Work

This thesis work presents an immersive interactive virtual city application with gestural interfaces. This prototype system and the proposed interaction techniques could be improved in the future.

In this prototype application, the map data is linked to 3D city models manually. In the future, this process could be simplified by an automatic method that loads map data from public sources like OpenStreetMap, and links to 3D models by reading the geographic location data attached with 3D models.

As the user study results suggest, the user experience with gesture based menu control could be improved by increasing realtime gesture recognition rate. Therefore this could be a possible direction of future work.

Selection techniques could also be improved. Ray casting and painting could be combined together, by replacing the “ray” with a small paint brush “tip” that can be easily positioned, and offering a mechanism to easily adjust the size of paint brush. In this way, the user can select single or multiple objects without changing the selection method.

A 3D modelling process often requires accuracy in manipulation of virtual objects. However, in an immersive environment, users usually focus on the visualization of virtual scenes, not modelling. Nevertheless it could still be beneficial if the manipulation techniques became more accurate. For example, the rotation of an object can be achieved by a coarse rotation first and then a fine rotation, therefore more precisely controllable.

A possible reason that novice users prefer a button based menu control is that they have previous experience using buttons, while not being familiar with gesture based interactions. Further research into this possibility could also be conducted in the future, for example by a within-subject user study, where the same users are invited to perform the same task from time to time, and observe the change of their attitude after becoming familiar with a specific

interaction technique.

Appendix 1

HMM Algorithms

HMM for Recognition

Given a HMM, λ and an observation sequence $O = O_1O_2, \dots O_T$, one basic problem is how to efficiently compute $P[O|\lambda]$, the probability of the observation sequence, given the model.

The solution to this problem is the way to test an observed sequence in a classification application, since the probability of the sequence represents the probability of this sequence belonging to that specified model. If we have a trained HMM model for each class, a simple way is to classify the observed sequence to the model that gives the largest probability.

There is an efficient way to solve this problem, namely the *Forward-Backward Procedure* [49]. Consider the forward variable $\alpha_t(i)$ defined as:

$$\alpha_t(i) = P(O_1O_2 \dots O_t, q_t = S_i | \lambda) \quad (1.1)$$

i.e., the probability of the partial observation sequence until time t , and state S_j at time t , given the model. Then the following procedure solves for $\alpha_t(i)$ and $P[O|\lambda]$ inductively:

1. Initialization:

$$\alpha_1(i) = \pi_i * b_i(O_1), \quad 1 \leq i \leq N. \quad (1.2)$$

2. Induction:

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^N \alpha_t(i) a_{ij} \right] * b_j(O_{t+1}), \quad 1 \leq t \leq T-1 \quad (1.3)$$
$$1 \leq j \leq N.$$

3. Termination:

$$P[O|\lambda] = \sum_{i=1}^N \alpha_T(i). \quad (1.4)$$

Similarly, define the backward variable $\beta_t(i)$ as:

$$\beta_t(i) = P(O_{t+1}O_{t+2}\dots O_T | q_t = S_i, \lambda) \quad (1.5)$$

i.e., the probability of the partial observation sequence from $t + 1$ to the end, given state S_i at time t and the model λ . The inductive procedure is:

1. Initialization:

$$\beta_T(i) = 1, \quad 1 \leq i \leq N. \quad (1.6)$$

2. Induction:

$$\begin{aligned} \beta_t(i) &= \sum_{j=1}^N a_{ij} b_j(O_{t+1}) \beta_{t+1}(j), \quad 1 \leq t \leq T-1 \\ &1 \leq i \leq N. \end{aligned} \quad (1.7)$$

3. Termination:

$$P[O|\lambda] = \sum_{i=1}^N \pi_i b_i(O_1) \beta_1(i). \quad (1.8)$$

Train HMM from Training Sequences

Another basic problem of HMM is that, given the observation sequence O , how to adjust the model parameter λ , to maximize $P[O|\lambda]$.

The solution to this problem is the way to train a HMM from training samples. We adjust the model parameter to achieve the largest probability on all the observation sequences used for training, thus resulting in a *trained* model.

Define the variable:

$$\gamma_t(i) = P[q_t = S_i | O, \lambda] \quad (1.9)$$

which represents the probability of being in state S_i at time t , given the observation sequence O , and the model λ . $\gamma_t(i)$ can be derived using forward-backward variables:

$$\gamma_t(i) = \frac{\alpha_t(i) \beta_t(i)}{P[O|\lambda]} = \frac{\alpha_t(i) \beta_t(i)}{\sum_{i=1}^N \alpha_t(i) \beta_t(i)} \quad (1.10)$$

Define another variable $\xi_t(i, j)$ as the probability of being in state S_i at time t and state S_j at time $t + 1$, given the model and the observation sequence:

$$\xi_t(i, j) = P[q_t = S_i, q_{t+1} = S_j | O, \lambda] \quad (1.11)$$

this can also be derived from forward-backward variables:

$$\begin{aligned} \xi_t(i, j) &= \frac{\alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{P[O | \lambda]} \\ &= \frac{\alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{\sum_{i=1}^N \sum_{j=1}^N \alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)} \end{aligned} \quad (1.12)$$

As long as these variables are ready, we can re-estimate the model parameter:

$$\sum_{t=1}^{T-1} \gamma_t(i) = \text{expected number of transitions from state } S_i \quad (1.13a)$$

$$\sum_{t=1}^{T-1} \xi_t(i, j) = \text{expected number of transitions from } S_i \text{ to } S_j. \quad (1.13b)$$

A maximum likelihood estimation of a HMM's parameters is given as:

$$\bar{\pi}_i = \text{expected probability in state } S_i \text{ at initial time} = \gamma_1(i) \quad (1.14a)$$

$$\begin{aligned} \bar{a}_{ij} &= \frac{\text{expected number of transitions from } S_i \text{ to } S_j}{\text{expected number of transitions from state } S_i} \\ &= \frac{\sum_{t=1}^{T-1} \xi_t(i, j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \end{aligned} \quad (1.14b)$$

$$\begin{aligned} \bar{b}_j(k) &= \frac{\text{expected number of times in state } S_j \text{ and observing } v_k}{\text{expected number of times in state } S_j} \\ &= \frac{\sum_{t=1}^T \gamma_t(j)}{\sum_{t=1}^T \gamma_t(j)} \quad \text{s.t. } O_t = v_k \end{aligned} \quad (1.14c)$$

It is proved that each time the model is re-estimated using (1.14), the probability of O being observed is improved [49]. After repeating this procedure iteratively, an optimal (sometimes locally optimal) model parameter set can be achieved. Therefore (1.14) is a maximum likelihood estimation of HMM, which is also named the *Baum-Welch method*.

Appendix 2

User Study Procedure

For each interaction category, a Latin square is used to produce 3 different versions, with different test orders. Please refer to section 4.3.1 for details about Latin square and interaction techniques' code names.

Version 1:	C1	C2	C3	S2	S3	S1	N1	N2	N3	V2	V3	V1
Version 2:	C2	C3	C1	S3	S1	S2	N2	N3	N1	V3	V1	V2
Version 3:	C3	C1	C2	S1	S2	S3	N3	N1	N2	V1	V2	V3

Therefore 3 versions of procedure are made. As shown in Table 2.1, in each version, task 1 is performed 3 times, with different menu control and selection techniques; task 2 is also performed 3 times, with different navigation and visibility techniques.

	Version 1	Version 2	Version 3
Task 1, first try	C1/S2	C2/S3	C3/S1
Task 1, second try	C2/S3	C3/S1	C1/S2
Task 1, third try	C3/S1	C1/S2	C2/S3
Finish questionnaire about menu control and selection.			
Task 2, first try	N1/V2	N2/V3	N3/V1
Task 2, second try	N2/V3	N3/V1	N1/V2
Task 2, third try	N3/V1	N1/V2	N2/V3
Finish questionnaire about navigation and visibility.			

Table 2.1: Three versions of user study procedure: order of techniques tested.

In task 1, the virtual city is set to a small size, with a scale factor of 1 : 300. In each try, the participant is first introduced to the menu control and selection techniques that will be tested in this try. Then:

0. Two to five minutes are allowed for him/her to get familiar with these techniques.
1. Call out menu, go to selection method submenu.
2. Choose the designated selection method.
3. Select the “Metlife” building.
4. Call out menu, go to selection submenu, choose “Clear” to de-select that building.
5. Call out menu, go to selection method submenu.
6. Choose the designated selection method.
7. Select the all buildings on the south side of “Metlife”.
8. Call out menu, go to selection submenu, choose “Clear” to de-select all buildings.

In task 2, the virtual city is set to a large size, with a scale factor of 1 : 3. In each try, the participant is first introduced to the navigation and visibility techniques that will be tested in this try. Then:

0. Two to five minutes are allowed for him/her to get familiar with these techniques.
1. Navigation starts on 42nd St and Park Ave, in front of the Grand Central Terminal.
2. Use the designated navigation technique, go west along 42nd St.
3. Turn right at Vanderbilt Ave.
4. Turn right again at 46th St.
5. Turn left at Park Ave.
6. Go along Park Ave until 48th St.
7. Go up to the top east corner of “JP Morgan & Chase” Tower, at 48th St and Park Ave.
8. Select “JP Morgan & Chase” Tower by ray casting, use designated visibility technique to open the blocked sight.

Appendix 3

Questionnaire Used in User Study

Participant Name:_____ Gender:_____ Age:_____

Previous Experience in VR and 3DUI:

1. Totally New User; 2. Medium Experience; 3. Expert.

Choose a number that best describe your agreement of disagreement with each statements.

Strongly Disagree	Disagree	Somewhat Disagree	Neutral	Somewhat Agree	Agree	Strongly Agree
1	2	3	4	5	6	7

C1-Q1: “Button Menu” is easy to learn and use.

C1-Q2: “Button Menu” let you select menu items effectively.

C1-Q3: “Button Menu”, as a whole, is a good menu control method.

C2-Q1: “Touch Menu” is easy to learn and use.

C2-Q2: “Touch Menu” let you select menu items effectively.

C2-Q3: “Touch Menu”, as a whole, is a good menu control method.

C3-Q1: “Slide Menu” is easy to learn and use.

C3-Q2: “Slide Menu” let you select menu items effectively.

C3-Q3: “Slide Menu”, as a whole, is a good menu control method.

N1-Q1: “Joystick” navigation is easy to learn and use.

N1-Q2: “Joystick” navigation can take you from one point to another effectively.

N1-Q3: “Joystick” navigation, as a whole, is a good navigation method.

N2-Q1: “Posture Triggered Flying” is easy to learn and use.

N2-Q2: “Posture Triggered Flying” can take you from one point to another effectively.

N2-Q3: “Posture Triggered Flying”, as a whole, is a good navigation method.

N3-Q1: “Waving Hand” navigation is easy to learn and use.

N3-Q2: “Waving Hand” navigation can take you from one point to another effectively.

N3-Q3: “Waving Hand” navigation, as a whole, is a good navigation method.

S1-Q1: “Ray Casting” selection is a good way to select single object.

S1-Q2: “Ray Casting” selection is a good way to select multiple adjacent objects.

S2-Q1: “Paint” selection is a good way to select single object.

S2-Q2: “Paint” selection is a good way to select multiple adjacent objects.

S3-Q1: “Stretch” selection is a good way to select single object.

S3-Q2: “Stretch” selection is a good way to select multiple adjacent objects.

V1: “Collapse” method is a good way to solve blocked sights.

V2: “Semi-Transparent” method is a good way to solve blocked sights.

V3: “WireFrame” method is a good way to solve blocked sights.

Best Choices:

Which one you think is the best overall, Button, Touch, or slide?

Which one you think is the best overall, Joystick, Posture Triggered Flying, or Waving Hand?

Appendix 4

Data Collected in User Study

“Exp” in following tables stands for previous experience in 3DUI and VR, see 4.3.1 for details.

ID	Exp.	C1-Button			C2-Touch			C3-Slide			
		Q1	Q2	Q3	Q1	Q2	Q3	Q1	Q2	Q3	Choice
1	3	6	4	5	7	7	7	5	6	6	2
2	3	5	7	5	6	7	6	4	5	4	2
3	3	5	6	4	6	7	7	5	3	3	2
4	2	7	6	6	6	4	5	7	7	7	3
5	2	6	5	6	6	5	5	7	7	7	3
6	2	7	6	7	6	7	7	6	5	6	2
7	1	7	7	4	7	7	6	7	7	7	3
8	1	6	6	6	7	6	6	5	5	5	2
9	1	7	5	6	7	7	7	5	5	5	2
10	1	6	4	6	6	6	6	5	3	3	1
11	1	7	7	7	5	5	5	3	3	3	1
12	1	7	7	7	4	5	4	7	5	5	1

Table 4.1: Results of user study: menu control. Q1: Easiness; Q2: Effectiveness; Q3: Overall.

ID	Exp.	S1-Ray Casting		S2-Paint		S3-Stretch	
		Q1	Q2	Q1	Q2	Q1	Q2
1	3	7	5	7	7	4	6
2	3	7	4	6	6	2	4
3	3	7	5	3	6	4	6
4	2	7	5	6	7	4	5
5	2	7	4	5	7	3	7
6	2	7	6	6	7	6	7
7	1	7	6	6	7	4	4
8	1	7	6	6	6	7	7
9	1	7	5	7	7	5	6
10	1	6	5	5	7	3	3
11	1	5	3	2	5	2	6
12	1	7	2	4	7	2	6

Table 4.2: Results of user study: selection. Q1: Selecting single building; Q2: Selecting multiple adjacent buildings.

ID	Exp.	N1-Joystick			N2-Posture Triggered			N3-Waving Hand			
		Q1	Q2	Q3	Q1	Q2	Q3	Q1	Q2	Q3	Choice
1	3	7	6	6	6	6	6	6	7	7	3
2	3	5	6	4	6	6	6	3	3	5	2
3	3	5	4	5	2	3	3	6	4	4	3
4	2	7	7	7	6	6	6	4	5	6	1
5	2	7	6	6	7	7	7	7	6	6	2
6	2	7	7	7	6	4	6	7	6	7	3
7	1	7	7	4	6	6	6	5	7	7	3
8	1	6	5	6	5	6	6	4	3	4	2
9	1	7	6	6	7	7	7	7	7	7	3
10	1	6	5	5	5	5	5	6	6	6	3
11	1	7	7	7	7	7	7	2	1	1	2
12	1	5	6	5	7	7	7	3	2	2	2

Table 4.3: Results of user study: navigation. Q1: Easiness; Q2: Effectiveness; Q3: Overall.

ID	Exp.	V1-Collapse	V2-SemiTransparent	V3-Wireframe
1	3	7	7	5
2	3	6	5	4
3	3	5	5	3
4	2	6	5	7
5	2	6	7	7
6	2	7	5	6
7	1	5	6	7
8	1	6	7	5
9	1	6	5	7
10	1	5	5	6
11	1	6	5	3
12	1	4	6	7

Table 4.4: Results of user study: visibility.

ID	Exp.	False Positive	Total Try	Success	Rate
1	3	0	24	23	95.83%
2	3	0	35	32	91.43%
3	3	1	33	28	84.85%
4	2	0	28	27	96.43%
5	2	1	31	29	93.55%
6	2	0	39	34	87.18%
7	1	1	26	26	100%
8	1	0	27	24	88.89%
9	1	0	35	31	88.57%
10	1	2	44	35	79.55%
11	1	4	41	34	82.93%
12	1	0	32	29	90.62%

Table 4.5: Results of user study: Realtime Gesture Recognition.

Appendix 5

Published Papers

The following papers related to this thesis' topic have been published or accepted during my study at Ryerson University:

Conference Paper

Ziyang Zhang, Tim McInerney, Ning Zhang, Ling Guan, A Cave Based 3D Immersive Interactive City With Gesture Interface, 22nd WSCG International Conference on Computer Graphics, Visualization and Computer Vision, June 2014.

Yifeng He, Ziyang Zhang, Xiaoming Nan, etc., vConnect: Connect the Real World to the Virtual World, 2014 IEEE International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications. (CIVEMSA 2014)

Xiaoming Nan, Ziyang Zhang, Ning Zhang, Fei Guo, Yifeng He, Ling Guan, VDesign: Toward Image Segmentation and Composition in CAVE Using Finger Interactions, IEEE China Summit / International Conference on Signal and Information Processing (ChinaSIP), 2013.

Journal Paper

Xiaoming Nan, Ziyang Zhang, Ning Zhang, Fei Guo, Yifeng He, Ling Guan, vDesign: A CAVE-based Virtual Design Environment Using Hand Interactions, Journal on Multimodal User Interfaces (Springer), Accepted in July, 2014.

References

- [1] Lucy Abramyan, Mark Powell, and Jeffrey Norris. Stage: Controlling space robots from a cave on earth. In *Aerospace Conference, 2012 IEEE*, pages 1–6. IEEE, 2012.
- [2] A.R.T. Advanced realtime tracking. <http://www.ar-tracking.com/products/>, Accessed on September 1st, 2014.
- [3] David A Atchison, George Smith, and George Smith. Optics of the human eye, 2000.
- [4] Ronald Azuma, Yohan Baillet, Reinhold Behringer, Steven Feiner, Simon Julier, and Blair MacIntyre. Recent advances in augmented reality. *Computer Graphics and Applications, IEEE*, 21(6):34–47, 2001.
- [5] Ronald T Azuma et al. A survey of augmented reality. *Presence*, 6(4):355–385, 1997.
- [6] Allen Bierbaum, Christopher Just, Patrick Hartling, Kevin Meinert, Albert Baker, and Carolina Cruz-Neira. Vr juggler: A virtual platform for virtual reality application development. In *Virtual Reality, 2001. Proceedings. IEEE*, pages 89–96. IEEE, 2001.
- [7] Frank Biocca and Ben Delaney. Immersive virtual reality technology. *Communication in the age of virtual reality*, pages 57–124, 1995.
- [8] Doug A Bowman, Jian Chen, Chadwick A Wingrave, John F Lucas, Andrew Ray, Nicholas F Polys, Qing Li, Yonca Haciahetoglu, Ji-Sun Kim, Seonho Kim, et al. New directions in 3d user interfaces. *IJVR*, 5(2):3–14, 2006.
- [9] Doug A Bowman, Sabine Coquillart, Bernd Froehlich, Michitaka Hirose, Yoshifumi Kitamura, Kiyoshi Kiyokawa, and Wolfgang Stuerzlinger. 3d user interfaces: new directions and perspectives. *IEEE computer graphics and applications*, 28(6):20–36, 2008.
- [10] Doug A Bowman, Ernst Kruijff, Joseph J LaViola Jr, and Ivan Poupyrev. An introduction to 3-d user interface design. *Presence: Teleoperators and virtual environments*, 10(1):96–108, 2001.

-
- [11] Doug A Bowman, Ernst Kruijff, Joseph J LaViola Jr, and Ivan Poupyrev. *3D user interfaces: theory and practice*. Addison-Wesley, 2004.
 - [12] Doug A Bowman and Ryan P McMahan. Virtual reality: how much immersion is enough? *Computer*, 40(7):36–43, 2007.
 - [13] Doug A Bowman and Chadwick A Wingrave. Design and evaluation of menu systems for immersive virtual environments. In *Virtual Reality, 2001. Proceedings. IEEE*, pages 149–156. IEEE, 2001.
 - [14] George EP Box, William Gordon Hunter, J Stuart Hunter, et al. *Statistics for experimenters*. John Wiley and sons New York, 1978.
 - [15] Tim Bray, Jean Paoli, C Michael Sperberg-McQueen, Eve Maler, and François Yergeau. Extensible markup language (xml). *World Wide Web Consortium Recommendation REC-xml-19980210*. <http://www.w3.org/TR/1998/REC-xml-19980210>, 1998.
 - [16] Kang-tsung Chang. *Introduction to geographic information systems*. McGraw-Hill New York, 2010.
 - [17] Carolina Cruz-Neira, Daniel J. Sandin, and Thomas A. DeFanti. Surround-screen projection-based virtual reality: The design and implementation of the cave. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, pages 135–142. ACM, 1993.
 - [18] Carolina Cruz-Neira, Daniel J Sandin, Thomas A DeFanti, Robert V Kenyon, and John C Hart. The cave: audio visual experience automatic virtual environment. *Communications of the ACM*, 35(6):64–72, 1992.
 - [19] Edsger W Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.
 - [20] Richard O Duda, Peter E Hart, and David G Stork. *Pattern classification*. John Wiley & Sons, 2012.
 - [21] Juri Engel, Sebastian Pasewaldt, Matthias Trapp, and J Dollner. An immersive visualization system for virtual 3d city models. In *Geoinformatics (GEOINFORMATICS), 2012 20th International Conference on*, pages 1–7. IEEE, 2012.
 - [22] Christian Frueh and Avidesh Zakhor. 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*, volume 2, pages II–562. IEEE, 2003.

-
- [23] GA Giraldi, Rodrigo Silva, and JC Oliveira. Introduction to virtual reality. Technical report, LNCC Research Report# 06/2003, National Laboratory for Scientific Computation, ISSN: 0101 6113, 2003.
- [24] Mordechai Haklay and Patrick Weber. Openstreetmap: User-generated street maps. *Pervasive Computing, IEEE*, 7(4):12–18, 2008.
- [25] John P Isaacs, Daniel J Gilmour, David J Blackwood, and Ruth E Falconer. Immersive and non immersive 3d virtual city: decision support tool for urban sustainability. *Journal of Information Technology in Construction*, 2011.
- [26] Andrew Johnson, Jason Leigh, Paul Morin, and Peter Van Keken. Geowall: stereoscopic visualization for geoscience research and education. *Computer Graphics and Applications, IEEE*, 26(6):10–14, 2006.
- [27] Julian Kang, Adithya Ganapathi, and Hussam Nseir. Computer aided immersive virtual environment for BIM. In *International Conference on Computing in Civil and Building Engineering*, 2012.
- [28] Naimul Mefraz Khan, Matthew Kyan, and Ling Guan. Immervol: An immersive volume visualization system. In *Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA), 2014 IEEE International Conference on*, pages 24–29. IEEE, 2014.
- [29] Ji-Sun Kim, Denis Gračanin, Krešimir Matković, and Francis Quek. The effects of finger-walking in place (fwip) for spatial knowledge acquisition in virtual environments. In *Smart Graphics*, pages 56–67. Springer, 2010.
- [30] Georg Klein and David Murray. Parallel tracking and mapping for small ar workspaces. In *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*, pages 225–234. IEEE, 2007.
- [31] Mikiko Koike and Mitsunori Makino. Crayon a 3d solid modeling system on the cave. In *Image and Graphics, 2009. ICIG'09. Fifth International Conference on*, pages 634–639. IEEE, 2009.
- [32] Joseph J LaViola Jr. Whole-hand and speech input in virtual environments. Master’s thesis, Brown University, 1999.

-
- [33] Yingzhu Li, L Shark, Sarah Jane Hobbs, and James Ingham. Real-time immersive table tennis game for two players with motion tracking. In *Information Visualisation (IV), 2010 14th International Conference*, pages 500–505. IEEE, 2010.
- [34] Rensis Likert. A technique for the measurement of attitudes. *Archives of psychology*, 1932.
- [35] I Scott MacKenzie and Steven J Castellucci. Empirical research methods for human-computer interaction. In *CHI Extended Abstracts*, pages 1013–1014, 2014.
- [36] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. California, USA, 1967.
- [37] A. Majumder and B. Sajadi. Large area displays: The changing face of visualization. *Computer*, 46(5):26–33, 2013.
- [38] Pelican Mapping. osgearth. <http://osgearth.org/>, Accessed on September 1st, 2014.
- [39] David Martin. *Doing psychology experiments*. Cengage Learning, 2007.
- [40] Daniel Medeiros, Lucas Teixeira, Felipe Carvalho, Ismael Santos, and Alberto Raposo. 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*, pages 211–218. ACM, 2013.
- [41] José del R Millán. Adaptive brain interfaces. *Communications of the ACM*, 46(3):74–80, 2003.
- [42] Jurriaan D Mulder, Jack Jansen, and Arjen van Rhijn. An affordable optical head tracking system for desktop vr/ar systems. In *Proceedings of the workshop on Virtual environments 2003*, pages 215–223. ACM, 2003.
- [43] Xiaoming Nan, Ziyang Zhang, Ning Zhang, Fei Guo, Yifeng He, and Ling Guan. vdesign: Toward image segmentation and composition in cave using finger interactions. In *Signal and Information Processing (ChinaSIP), 2013 IEEE China Summit & International Conference on*, pages 461–465. IEEE, 2013.
- [44] Jakob Nielsen and Rolf Molich. Heuristic evaluation of user interfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 249–256. ACM, 1990.

-
- [45] OpenStreetMap. Website. <http://www.openstreetmap.org/>, Accessed on September 1st, 2014.
- [46] Randy Pausch, Dennis Proffitt, and George Williams. Quantifying immersion in virtual reality. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 13–18. ACM Press/Addison-Wesley Publishing Co., 1997.
- [47] Francisco Pinto, Alexandre Buaes, Diego Francio, Alécio Pedro Delazari Binotto, and Pedro Santos. Bratrack: a low-cost marker-based optical stereo tracking system. In *SIGGRAPH Posters*, page 131, 2008.
- [48] Vivek Pradeep, Christoph Rhemann, Shahram Izadi, Christopher Zach, Michael Bleyer, and Steven Bathiche. Monofusion: Real-time 3d reconstruction of small scenes with a single web camera. In *Mixed and Augmented Reality (ISMAR), 2013 IEEE International Symposium on*, pages 83–88. IEEE, 2013.
- [49] Lawrence Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.
- [50] MBI Reaz, MS Hussain, and Faisal Mohd-Yasin. Techniques of emg signal analysis: detection, processing, classification and applications. *Biological procedures online*, 8(1):11–35, 2006.
- [51] Zhou Ren, Jingjing Meng, Junsong Yuan, and Zhengyou Zhang. Robust hand gesture recognition with kinect sensor. In *Proceedings of the 19th ACM international conference on Multimedia*, pages 759–760. ACM, 2011.
- [52] Gerhard Rigoll, Andreas Kosmala, and Stefan Eickeler. High performance real-time gesture recognition using hidden markov models. In *Gesture and Sign Language in Human-Computer Interaction*, pages 69–80. Springer, 1998.
- [53] George Robertson, Mary Czerwinski, Patrick Baudisch, Brian Meyers, Daniel Robbins, Greg Smith, and Desney Tan. The large-display user experience. *Computer Graphics and Applications, IEEE*, 25(4):44–51, 2005.
- [54] Jérôme Royan, Patrick Gioia, Romain Cavagna, and Christian Bouville. Network-based visualization of 3d landscapes and city models. *Computer Graphics and Applications, IEEE*, 27(6):70–79, 2007.

-
- [55] Thomas Schlömer, Benjamin Poppinga, Niels Henze, and Susanne Boll. Gesture recognition with a wii controller. In *Proceedings of the 2nd international conference on Tangible and embedded interaction*, pages 11–14. ACM, 2008.
- [56] Takashi Shibata. Head mounted display. *Displays*, 23(1):57–64, 2002.
- [57] Mel Slater. Measuring presence: A response to the witmer and singer presence questionnaire. *Presence: Teleoperators and Virtual Environments*, 8(5):560–565, 1999.
- [58] Mel Slater. A note on presence terminology. *Presence connect*, 3(3):1–5, 2003.
- [59] Moshe Sniedovich. Dijkstra’s algorithm revisited: the dynamic programming connexion. *Control and cybernetics*, 35:599–620, 2006.
- [60] Peng Song, Wooi Boon Goh, Chi-Wing Fu, Qiang Meng, and Pheng-Ann Heng. Wysiwyf: exploring and annotating volume data with a tangible handheld device. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1333–1342. ACM, 2011.
- [61] Gary K Starkweather. Dsharpa wide-screen multi-projector display. *Journal of Optics A: Pure and Applied Optics*, 5(5):S136, 2003.
- [62] Richard Stoakley, Matthew J Conway, and Randy Pausch. Virtual reality on a wim: interactive worlds in miniature. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 265–272. ACM Press/Addison-Wesley Publishing Co., 1995.
- [63] Alistair Sutcliffe, Brian Gault, Terence Fernando, and Kevin Tan. Investigating interaction in cave virtual environments. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 13(2):235–267, 2006.
- [64] John Vince. *Introduction to virtual reality*. Springer, 2004.
- [65] Oculus VR. Oculus rift. <http://www.oculusvr.com/>, Accessed on September 1st, 2014.
- [66] Rui Wang and Xuelei Qian. *OpenSceneGraph 3 Cookbook*. Packt Publishing Ltd, 2012.
- [67] Youwen Wang, Cheng Yang, Xiaoyu Wu, Shengmiao Xu, and Hui Li. Kinect based dynamic hand gesture recognition algorithm research. In *Intelligent Human-Machine Systems and Cybernetics (IHMSC), 2012 4th International Conference on*, volume 1, pages 274–279. IEEE, 2012.

-
- [68] Colin Ware, Kevin Arthur, and Kellogg S Booth. Fish tank virtual reality. In *Proceedings of the INTERACT'93 and CHI'93 conference on Human factors in computing systems*, pages 37–42. ACM, 1993.
- [69] G. Welch and E. Foxlin. Motion tracking: no silver bullet, but a respectable arsenal. *Computer Graphics and Applications, IEEE*, 22(6):24–38, 2002.
- [70] Daniel Wigdor and Dennis Wixon. *Brave NUI World: Designing Natural User Interfaces for Touch and Gesture*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition, 2011.
- [71] Qing Zhu, Mingyuan Hu, Yeting Zhang, and Zhiqiang Du. Research and practice in three-dimensional city modeling. *Geo-spatial Information Science*, 12(1):18–24, 2009.