Natural Gesture Based Interaction for Handheld Augmented Reality

---

A thesis submitted in partial fulfilment of the

requirements for the Degree of Master of

Science in Computer Science

By Lei Gao

Supervisors:
Assoc. Prof. R. Mukundan
Prof. Mark Billinghurst

Department of Computer Science
University of Canterbury
June, 2013

Table of Contents

Acknowledgements

First of all, I would like to thank my supervisors: R Mukundan, Mark Billinghurst and Jihad El-Sana, who supported me throughout the course of this thesis. Their extensive support and suggestions for development, experiment design, writing and data analysis helped me a lot with writing a solid thesis. I would also like to thank Huidong Bai, a PhD student in the Hit Lab NZ, who provided many useful ideas for this thesis. In addition, I would like to thank all the people who were willing to participate in my user study for their cooperation.

Abstract

The goal of this research thesis is to explore and evaluate a novel interaction interface performing canonical manipulations in 3D space for Augmented Reality (AR) on handheld devices. Different from current handheld AR applications usually using touch-screen based interaction methods, we developed a 3D gesture based interaction approach for handheld AR using an attached RGB-Depth camera to provide intuitive 3D interaction experience in 3D space. By identifying fingertips and mapping their 3D positions into the coordinate system of AR virtual scene, our proposed method allows users to perform operations on virtual objects using their fingers in midair with six-degrees-of-freedom (6DOF). We applied our methods in two systems: (1) a client-server handheld AR system, and (2) a standalone handheld tablet AR system. In order to evaluate the usability of our gesture-based interface we conducted a user study in which we compared the performance to a 2D touch-based interface. From the results, we concluded that traditional 2D touch-based interface performed faster than our proposed 3D gesture-based interface. However, our method proved a high entertainment value, suggesting great possibilities for leisure applications.

Chapter 1

Introduction

Augmented Reality (AR) is the technology that augments the real world by using virtual information overlaid on a view of reality. In this way AR is different from Virtual Reality (VR), which uses a simulated environment to replace the real world. In other words, AR aims at developing new user interfaces to enhance the interaction between users and the real world. Traditional AR requires high-end computers and special equipment such as head mounted displays; however, with the development of faster hardware in recent years, handheld devices with digital cameras have become suitable for AR. Using a handheld video see-through AR interface on smart phones or tablets, a user can see virtual scenes superimposed on live video of the real world. In this case, built-in cameras capture a view of the real world, visual tracking or location-based sensing is used to find the camera pose, and finally graphics rendering is used to overlay virtual objects on the video view. Figure 1.1 shows some examples of current handheld AR applications.

Figure 1.1 Augmented Reality on handheld devices

In order for handheld AR to reach its full potential, users should be able to create, and manipulate virtual objects and their properties with respect to the real world in 3D space. Intuitive interaction and manipulation is a primary challenge in handheld AR. State of the art handheld AR applications offer a variety of AR interaction techniques that can be used to performing translation, rotation or scaling to alter the location, pose or size of virtual objects in space. For example, freeze view interaction method [33] and device pose based interaction method [20].

Using a handheld AR system is very different from using a more traditional head mounted display (HMD) based AR system since the handheld device combines both display and input components. Therefore, interaction methods that work well in HMD based AR may not work well for handheld AR. For example, users may not be able to easily interact with virtual objects in one-handed handheld AR if they are holding the handheld device with one hand and touching the screen with the other, while at the same time trying to maintain visual tracking of an AR target.

Current interaction with handheld AR environments is often limited to pure 2D pointing and clicking via the device's touch screen. Touch based interaction

suffers from several problems. First, users are required to point the handheld device in the direction of a virtual object's position in the real world, thus forcing them to hold it in a position that might not be optimal for interaction. Second, the size of AR virtual objects on the screen is dictated by their position in the real world, and thus they might be too small for users to comfortable interact via touch screen. In addition, moving fingers over the screen covers large parts of the virtual content making it difficult to select. Based on above discussion, 3D natural gesture interaction is considered an alternative input method for handheld AR. However, manipulations in 3D space are uncommon in handheld AR applications at present, where the registration of the virtual content to the physical world adds to the complexity of interactions.

In this thesis our main motivation is to investigate the potential of 3D gesture interaction for handheld AR. We explore a markerless 3D gesture-based interaction approach for two systems: (1) a client-server handheld AR system, and (2) a standalone handheld tablet AR system. Each system provides 6DOF manipulation by capturing midair gestures using an RGB-Depth camera, performing image processing to obtain 3D position of fingertips, and then mapping the movement of the fingertips onto the position, orientation, and scale of the selected virtual. Hence, our technique improves intuitiveness by allowing users to interact with virtual objects in a similar way that they interaction with objects in the real world.

Finger gesture interaction helps users to interact with handheld mobile AR applications using their bare hands. The RGB-Depth camera of the handheld device can track the user's fingertips in 3D space and allows users to reach out their hands and directly interact with virtual content in the AR scene. The system segments the hand region and identifies fingertips, which are treated as 3D

pointers. This avoids the need for touch screen interaction, and so could offer a more intuitive experience.

In order to evaluate the effectiveness of our gesture-based interface we conducted a user study in which we compared performance to a 2D touch-based interface. From the results of this study, we conclude that a traditional 2D touch-based interface performs faster than our proposed 3D gesture-based interface on the test task. However, users found the gesture interface method to be very entertaining, suggesting great possibilities for leisure applications. The gesture-based interface also provides a similar positive user experience as the 2D touch-based interaction method in terms of naturalness and less mental stress.

The main contributions of this master's research are:

- Development of an approach for markerless fingertip detection and tracking based on skin-color segmentation methods.
- The definition and design of some natural gestures for users to use to interact with virtual objects in a handheld AR environment.
- Developing two gesture based interaction systems for handheld AR. One based on a client-server system, and the other using a tablet as the handheld AR device.
- Development of a simple tablet-based touch based interaction system for handheld AR.
- Conducting a comprehensive user study to compare the prototype gesture-based interaction technique with a common screen-touch interface.
- Build a software library that can be used to easily develop gesture based interaction methods for handheld AR applications.

In this paper, we start by discussing recent related work in chapter 2. Chapter 3 introduces the design and implementation of our 3D gesture-based and 2D touch-based interaction methods respectively. Chapter 4 presents a user study of our prototype handheld AR interaction techniques, comparing them with a traditional screen-touch interaction method in terms of operation time and user experience. This was done in order to investigate the usability of gesture-based methods for handheld mobile AR. In chapter 5, we analyze the results we've acquired from the experiment. We finish this thesis in chapter 6 with the lessons learned, conclusion and directions for future work

Chapter 2

Literature Review and Related Work

Augmented Reality (AR) is technology that uses virtual information to augment the real world. Azuma states that AR scenes have three common characteristics: (1) the real and virtual images are combined together, (2) there is real-time interaction with the virtual content and (3) the virtual content is registered in 3D space [1]. These properties have been considered to be the key parts of an AR interface since the first AR system was created in the 1960's by Ivan Sutherland [2]. However, most of the researchers in the field of AR have been focused on developing the technology to improve the user's visual experience, such as tracking and display devices [3]. The user interaction methods for AR interface have been limited to basic viewing of the virtual information added to the real world. Only in recent years, have some researchers started to shift their research interests to improving the user's interaction experience.

In this thesis our focus is on exploring the potential of 3D gesture interaction

for handheld AR. In order to do that we need to base our research on earlier work on handheld AR and AR user interfaces. Therefore, in this chapter, we first provide a brief review of AR technology and tracking methods. Then, we focus on reviewing interaction methods for the handheld AR experiences and especially previous work on gesture interaction.

2.1 Traditional Augmented Reality and Handheld Augmented Reality

2.1.1 Wearable Augmented Reality

Figure 2.1: MARS (left) and Tinimth (right)

Handheld Augmented Reality systems can trace their roots back to early mobile and wearable AR systems. The MARS system developed by Columbia University in 1996 was the first mobile Augmented Reality system (Figure 2.1) [5]. It was a backpack system, which allowed the user to walk around and experience AR outdoors. However, this system had certain limitations. The main disadvantage was that the user had to wear over 40 pounds of equipment, including GPS hardware, a wearable PC, head tracking system and a head mounted display. Furthermore, limited battery life also restricted the working time of the system. Finally, the user was restricted to the area around a local base station for its

wireless communication infrastructure, so they could not roam freely. Other wearable AR systems, such as BARS [6] and Tinimth [7], suffered the same disadvantages as MARS.

Figure 2.2: Video See-through HMD (left) and Optical See-through HMD (right)

These wearable systems used a head mounted display (HMD) [4] for viewing the AR content, as shown in Figure 2.2. The HMD could be either optical see-through or video see-through. However, due to their high price and limited functionality, HMDs are not commonly used in daily life.

2.1.2 Handheld Augmented Reality

Wearable AR devices are often bulky and expensive; furthermore, they have complicated user interfaces that are difficult to be operated by novices [8]. As hardware developed over the years, the computing and graphics power of handheld devices, such as tablet PC, personal digital assistants (PDA) and mobile phones, became significant enough for AR applications. Therefore, researchers began to become interested in using handheld devices for Augmented Reality due to their portability and popularity.

The first handheld AR devices used a client-server approach due to their low

CPU power, such as the mobile AR-PDA introduced by Gausemeier et al. (Figure 2.3) [9]. In this case a photo was taken by the PDA and wirelessly sent to a remote PC where it was processed and virtual graphics overlaid on the image before it was sent back to the PDA for viewing. In 2003, the popular AR library, ARToolKit, was ported to a handheld platform by Wagner [34], and so handheld devices could run stand-alone AR applications without the help of a remote server.

Figure 2.3 The framework of the AR-PDA

Mobile phones based AR is a type of handheld Augmented Reality which uses smart phones as the AR device rather than a PDA or tablet PC. Current smart phones provide several advantages for using as an AR platform: a high-quality color display, a fast processor, high-resolution digital camera, large memory space, affordable price, small size and light weight. Furthermore, smart phones can also use GPS and compass sensors to find their location, and they can exchange data with other device over broadband data connections [10]. All of these capabilities

of smart phones support mobile AR in a more natural way than other wearable and handheld AR systems. For example, the navigation systems introduced by Dünser et al. in 2012 [35].

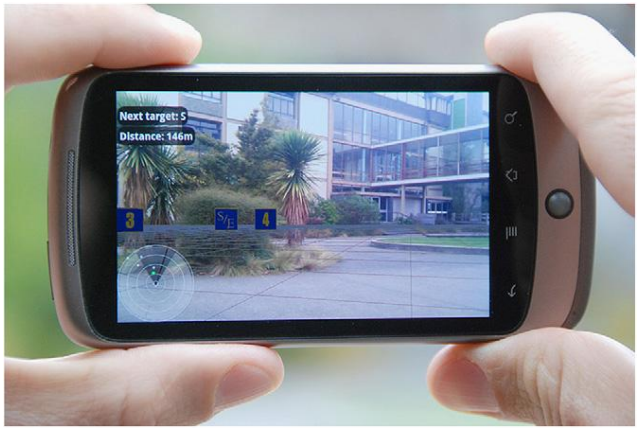

Figure 2.4 The navigation system

2.2 Tracking Methods for Handheld AR

With the development of high-powered mobile CPUs, real-time tracking has become available for handheld AR system. Computer vision based tracking algorithms from desktop computers can be ported to handheld AR system with almost no modification. Traditional computer vision tracking algorithms can be organized into three main types: marker based tracking, markerless tracking (natural feature tracking) and hybrid tracking.

2.2.1 Marker Based Tracking

Traditional marker based tracking is based on printed black and white markers. The computer vision system searches for markers from the camera video stream, and then calculates the marker's 3D position and orientation from the known

features in the marker image. After the marker has been identified, the system renders the virtual objects in the video frame. This processing procedure was made popular by the ARToolKit library, and has been available on mobile phones since 2005 [36].

In order to balance robustness with computational complexity, Wagner et al. introduced several new marker tracking techniques in 2008 [11], which are based on frame markers, split markers and dot markers (Figure 2.5). They indicated that these new marker tracking techniques can effectively reduce image clutter and more easily blend into typical AR environment.

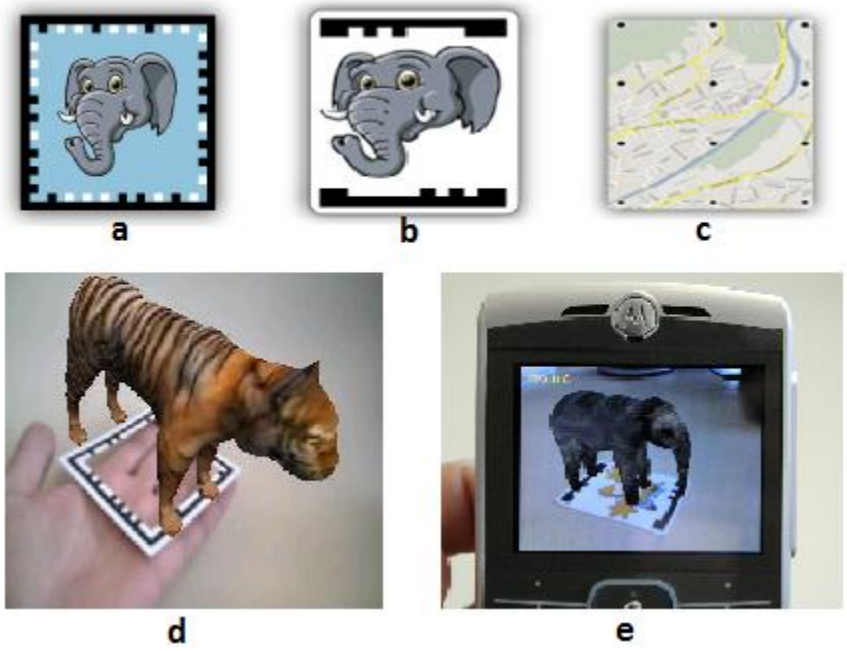

Figure 2.5: Frame marker (a); Split marker (b); Dot marker (c);
Example for frame marker tracking (d); Example for split marker tracking (e)

Compared to natural feature tracking, marker based tracking requires less processing resources and so is faster. The computing power of handheld devices

has been limited by the battery technology; therefore, marker based tracking techniques are very advantageous for handheld AR. However, the tracking can fail when portions of the markers are covered up, so maker based tracking is not as robust as natural feature tracking.

2.2.2 Natural Feature Tracking

Natural feature tracking (NFT) techniques track from features of the surrounding environment such as corners, edges, surface texture, interest points and even 3D objects. Compared to marker based tracking, natural feature tracking can work well when the targets are only partially in the view. However, a database of key points is required for NFT and it needs more computational resources to detect the key points from the environment. Therefore, NFT techniques are much slower than marker tracking, especially for handheld AR [12].

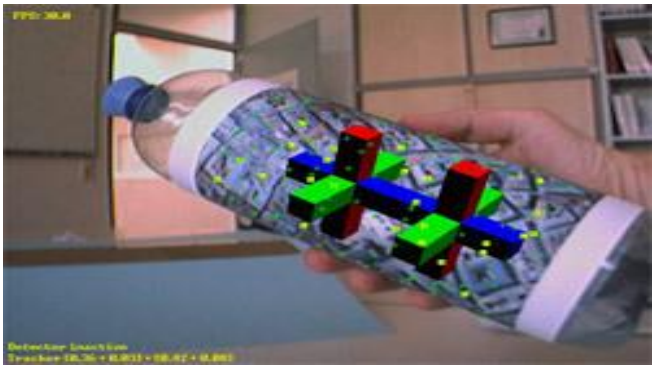

Figure 2.6 Natural feature tracking

During the natural feature tracking process, the system detects features from the initial image and tracks the detected features in the following video stream. Once features have been found, the camera position can be calculated from them and used for rendering the virtual objects.

In order to use NFT in mobile AR system, Park et al. found several cues to speed up the tracking process [37]. First of all, the system does not need to extract features for each frame. Secondly, the number of feature points tracked can be reduced to an acceptable level for mobile devices. Thirdly, the feature points which will be disappeared soon can be excluded to reduce the computational complexity.

2.2.3 Hybrid Tracking

In order to overcome the limitations of computer vision based tracking, and to increase robustness, speed and accuracy, some researchers are have been exploring how other sensors can be used together with vision input in hybrid tracking systems [13]. For example, GPS, compass and inertial sensors can be used together with camera input to improve outdoor tracking. In this way, an AR tracking system can provide more accurate spatial data (position and orientation) in real time. As the GPS, electronic compasses, inertial and optical sensors have become common in the handheld devices, hybrid tracking is becoming more available for handheld AR.

2.3 Interaction Methods for Handheld AR

In handheld AR applications, users often need to be able to interact with the virtual objects. In other words, the virtual objects can be created, modified or manipulated in the real world environment by the users [14]. Most of the commonly used interaction techniques are based on pointing and clicking on the touch screen of mobile phones (device based interaction); however, these approaches have several problems, such as limited 2D manipulation and fat finger problem. To deal with these problems, alternative approaches, such as gesture

interaction, have been explored. In the following sections, we will present a review of these methods.

2.3.1 Device Based Interaction Approaches

The input of device based interaction approaches for handheld AR is based on the keyboard, touch screen and sensors such as camera and electronic compass. For example, after the tracking system captures the position and orientation of the virtual objects and renders them in the video stream, it is possible for users to provide 6 DOF (degree of freedom) interactions with virtual objects via the direction keys on the keyboard of a mobile phone. These traditional interaction techniques have been widely adopted in the handheld AR.

In 2005, Henrysson et al. [15] presented an interaction approach which used both the phone camera and a traditional button interface as input to provide a 6 DOF isomorphic interaction with the virtual 3D objects. When a button on the phone joypad was pressed, the virtual object currently selected was locked to the phone. Then, the object could be rotated and translated by moving the mobile phone (see Figure 2.7). However, in order to extend the virtual tracking range, multiple markers were needed, which increased the complexity of the tracking calculation.

Figure 2.7: Interaction approach based on keyboard input

Chen et al. used a similar interaction technique in a mobile AR Chinese chess game developed in 2008 [16]. In this application, a chess piece is selected when the camera is aiming at it and a button is clicked (see Figure 2.8). To move the chess piece, the user just needs to move the camera and release the button at the target position. They report that this approach provided a very natural human-computer interface for the users.

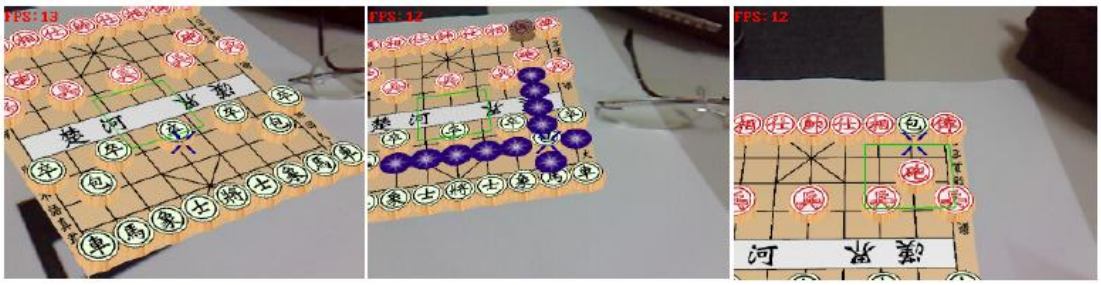

Figure 2.8: The virtual chess board (left); The user clicks a button after aiming the chess piece (middle); The chess piece moved to the destination (right)

With the development of touch screen hardware, more and more handheld devices and mobile phones have no physical keyboard; therefore, all of the manipulations need to be done by using touch screen or virtual buttons. In this case, Langlotz et al. presented a new interaction approach, which is called freeze mode, in their mobile AR authoring system [17]. Once the user is holding the mobile phone in the position where he or she wants to begin editing, he or she can freeze the view by clicking a certain button indentified on the touch screen. The current camera frame will be frozen and the user can manipulate virtual content directly on the phone screen. After all the manipulations have been completed, the user can unfreeze the view (Figure 2.9).

Figure 2.9: Freeze the view in the position (left); Manipulation of the virtual content (middle); Unfreeze the view after the task is completed

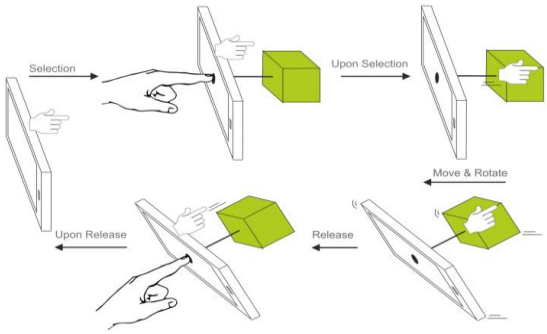

Figure 2.10: 6 DOF manipulation using HOMER-S

Many other handheld AR applications based on touch screen input have been introduced by researchers, such as a location annotation system [18] and a note-taking system [19]. However, interaction by pointing and clicking via the touch screen suffers from some common problems [14]. The most important issue is that manipulations on the touch screen are limited to pure 2D pointing and clicking, which cannot support effective 3D manipulation in the real world. In order to provide full 3D manipulation using a touch screen, Mossel et al. presented a new interaction approach called HOMER-S (Figure 2.10) [20]. For translation and rotation tasks, the system directly maps the handheld device's pose onto the object. For scaling tasks, the system simplify the task to 3 DOF based on the device's position information. They found that HOMER-S can increase the interaction

speed but decrease the input accuracy.

2.3.2 Gesture Based Interaction Approaches

Compared to device-based interaction, gesture-based interaction provides a natural and intuitive way to interact with virtual objects in mobile AR applications. The gesture here mainly involves deictic pointing gestures, using finger or hand movement. Gesture based interaction can effectively deal with the manipulation issues of touch screens which we have mentioned in the previous section; therefore, in recent years, more and more research has been done in this area.

Henrysson et al. first presented an evaluation of several different interaction approaches for translation and rotation of virtual objects on mobile phones, which includes marker based gesture input captured by the phone's front camera, tangible input, keypad input and phone tilting input [21]. These usability experiments indicated that tangible input techniques were more advantageous for translation tasks, while keypad input was best for rotation tasks. However, they also pointed out that the tilting method performed almost as well as keypad input in some cases for rotation tasks. That is to say, vision based input methods can perform better for natural and intuitive interaction.

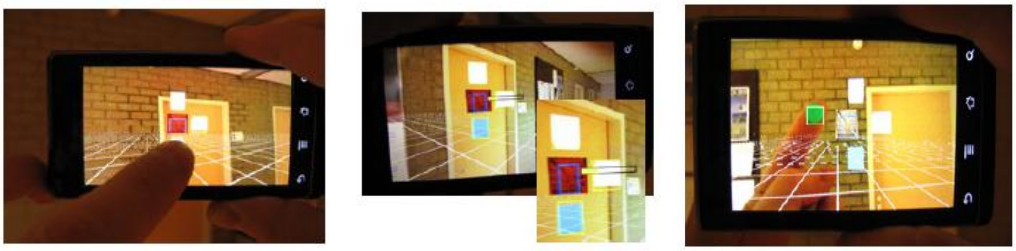

Figure 2.11: Touch based interaction (left); Device based interaction (middle); Fingertip based interaction (right)

In 2011, Hürst and Wezel also presented a user study about three different interaction approaches, which includes touch based interaction, device based interaction (using the position and orientation of the mobile phone) and single green marker based finger tracking interaction (see Figure 2.11) [22]. The user study showed that touch based interaction is best for selection tasks, while the device based interaction is more suitable for translation tasks. However, finger based interaction is considered to be more interesting for the users. In this case, finger based interaction seems to be a solution for mobile gaming and other leisure applications. Consequently, there is no clear answer about which interaction method is better. The choice depends on the type of the task.

In 2012, Hürst and Wezel extended their finger based interaction from one marker to two markers, which used a green marker and a red marker on two fingers (see Figure 2.12) [14]. They indicated that this improved interaction method can nicely integrate the virtual and real world through a natural and intuitive interaction. The user study also provided positive feedback on translation, rotation and scaling tasks. Moreover, they also mentioned that the tracking performance can be significantly reduced when more than two markers are used together.

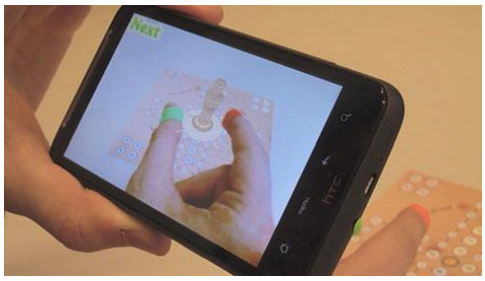

Figure 2.12: A green marker and a red marker are used to track fingers

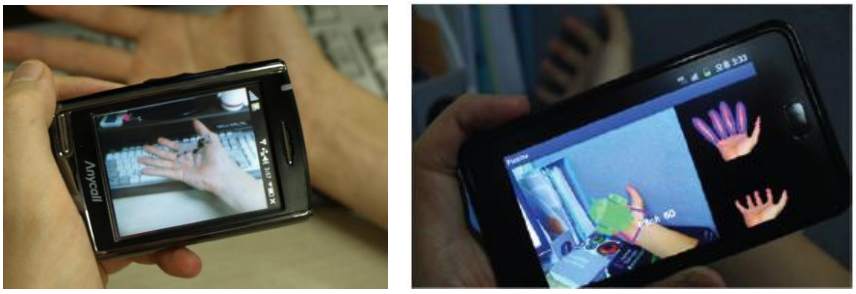

Figure 2.13: One-handed interaction (left); Bare-hand-based interface (right)

In order to provide more natural interaction, researchers have also tried gesture-based interaction with no fingertip markers on mobile devices. Seo et al. presented an approach in which the virtual object is rendered on the user's palm and reacted to the changes of palm pose such as opening or closing the hand [23]. No markers were used in this technique, so it can provides a very natural interaction for mobile AR that can be used anytime and anywhere. In 2011, Choi et al. proposed a bare-hand-based AR interface which is similar to the above technique [24]. Compared to the approach presented by Seo et al., this new technique could more accurately estimate all possible palm poses (see Figure 2.13).

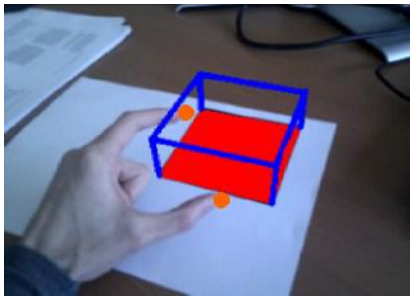

Figure 2.14: Markerless visual fingertip interaction

In 2011, one promising gesture interaction approach for mobile AR was

presented by Baldauf et al., which used markerless visual fingertip tracking to interact with the virtual objects [25]. The fingertips of the user's thumb and index finger were detected in real time in their system (see Figure 2.14). However, they just used a single camera mobile phone in their research; therefore, no depth information was provided and we could not find out how the gesture-based interaction performed in 3D space.

2.4 Summary and Motivation for Current Research

From the reviews above, we can see that most of the previous researches on interaction in the field of handheld AR focused on a device based interaction approach or a gesture based interaction approach with color markers. However, these two approaches both suffer from some problems.

For device based interaction, in most situations, the virtual object's position in the real world may not be optimal for interaction when users hold the mobile phone [14]. For example, when a user uses one finger to move a virtual object via the touch screen, the finger may move out of the screen boundary before the virtual object reaches the target position. Furthermore, the size of the virtual objects depends on their position in the real world environment, which means that they may be too large or too small for users to control on a touch screen. In some cases, the users' fingers may also cover a large part of the content on the touch screen (the fat finger problem). The most important issue is that manipulations on the touch screen are limited to pure 2D pointing and clicking, which cannot support effective 3D manipulation in the real world. Although Mossel et al. have provided a solution for this issue (HOMER-S) [20], the system still requires the user to indicate which kind of manipulation he wants to take before the tasks start.

In this case, it would increase the manipulation complexity and decrease the usability of the system.

For gesture based interaction, researchers prefer to use hand detection and tracking methods based on color markers, which can provide an accurate detection and tracking result. However, in the real world environment, it is impractical to require users to wear color markers on their hands before they use a handheld AR application. Although Baldauf et al. have introduced one alternative approach based on markerless finger tracking system [25], there is no depth information provided in the system. In this case, the algorithm may not be ideal for 3D interaction, which could significantly reduce the usability of the system.

In order to avoid the issues we discussed above and provide a natural and intuitive interaction method for the users, we will focus on studying gesture-based interaction method that use markerless finger tracking algorithms. In addition, we will use special depth sensors to get the depth information of the user's hand in order to support true 3D interaction in the real world. Here are the main features of our research:

- Natural gesture based interaction method;
- Markerless hand and finger detection and tracking;
- 3D interaction with virtual objects in the real world;
- Real-time interaction;
- Handheld AR application (no time and space limitation);
- Different manipulation tasks (translation, rotation and scaling) which can be processed at the same time;

Chapter 3

Research Implementation

Since research is focused on using fingertips to interact with virtual objects in the real 3D environment, hand and fingertip detection and tracking are key points. In this chapter, we will first introduce the work we have done implementing a gesture tracking system, such as the hardware and software used, and the detection and tracking algorithms used. In the following sections of this chapter, we will describe three handheld AR interaction systems we designed and studied in our research.

3.1 Gesture Tracking System

In our research, we plan to use fingertips to interact with virtual objects in 3D space; therefore, developing the gesture tracking system can be divided into two research areas: hand detection and fingertip tracking. As we intend to provide a full 3D manipulation experience for the users in the real world, we need to

calculate the depth information for the user's fingertips. In order to achieve this, specific hardware and software is needed. In the following sections, we will describe the hardware, software and algorithms we used for our gesture tracking system.

3.1.1 Hardware

In our research we developed gesture-tracking systems for the PC, tablet and smart phone platforms. For the PC, we choose a desktop computer (Intel 2.4Ghz Dual-Core, 4GB RAM, Windows 7). For the tablet, we ran the tracking system on a Samsung XE500T1C tablet, featuring an Intel(R) Core™ i5 CPU, 4.00 GB RAM and an Intel(R) HD integrated graphics, running the Windows 7 operating system. The Samsung Galaxy S2 (1.2GhzDual, Android2.3.5) was used as the main mobile platform.

Figure 3.1: Microsoft Kinect (left); Primesense depth sensor (right)

The depth sensors used to acquire the depth information in our research were Microsoft Kinect and Primesense depth sensors (see Finger 3.1). The Microsoft Kinect can work well with a PC; however, it requires an additional power supply and so cannot be used with the tablet platform. The Primesense depth sensor just requires an USB connection and so is ideal for the tablet platform.

3.1.2 Software

To obtain image data from the depth sensor, we combine the OpenCV [38] and the OpenNI [39] libraries together. We Configure OpenCV with OpenNI support by setting WITH_OPENNI flag in CMake. The object rendering function and 3D graphics in our system is based on the OpenGL library [40]. AR tracking is implemented using a natural feature-tracking library called OPIRA [26]. We choose this library due to its robustness and fast computation time. Furthermore, the library has a convenient interface for the OpenGL library, which can help us easily finish scene-rendering tasks. OPIRA can track a target image in real-time and provide accurate camera pose calculation.

3.1.3 Hand Detection and Fingertip Tracking

Among feature-based detection methods, skin color has been proven to be a robust cue for human face and hand detection [27]. Color detection allows for fast processing without the requirement of high-powered computing or graphics processors; therefore, it also can perform well on handheld devices. Research shows that human skin has a characteristic color which can be recognized by computers after image processing and a variety of skin color segmentation methods for face and hand detection have been developed in recent years. However, some challenges need to be overcome when using skin color for face and hand detection, such as which color space to choose, and how to calculate the skin color distribution.

3.1.3.1 Color space

The color model, also called color space, is an abstract mathematical model which uses numbers to present colors, such as RGB, HSV, HSL and YCrCb. RGB is a

color space describing color as a combination of three colored rays, which has been widely used for processing and storing of digital image data. In this color space, one color can be described by three independent numbers (stand for Red, Green and Blue) between 0 and 255.

HSV and HSL color spaces rearrange the geometry of RGB color in an intuitive way based on the artist's idea of tint, saturation and tone. H, which stands for hue, describes the dominant color of the image. S, which stands for saturation, defines the colorfulness of an area in proportion to its brightness. V stands for value or brightness and L stands for lightness. These two values both measure the color luminance of an image. In HSV and HSL color spaces, the intuitive components can provide explicit discrimination between luminance and chrominance properties; therefore, these two color space have been popular and widely used in the research on skin color segmentation. The following function shows the conversion between RGB and HSV color space:

$$
\begin{aligned}
H &= \arccos \frac{\frac{1}{2}((R-G)+(R-B))}{\sqrt{((R-G)^2+(R-B)(G-B))}} \\
S &= 1-3\frac{\min(R,G,B)}{R+G+B} \\
V &= \frac{1}{3}(R+G+B) \\
\mathrm{L} &= \frac{1}{2}\left(\mathrm{Max(R,G,B)}+\mathrm{Min(R,G,B)}\right)
\end{aligned}
$$

YCrCb is the color space using a nonlinear procedure to calculate values from the RGB information. In this color space, Y is the luminance component and Cb and Cr are the blue-difference and red-difference components. The transformation from RGB to YCrCb is found using the following function:

$$Y = 0.299R + 0.587G + 0.114B$$
$$C_r = R - Y$$
$$C_b = B - Y$$

3.1.3.2 Skin Colour Modeling

The goal of skin colour detection is to separate all of the pixels of an image into two parts: skin and non-skin pixels. There are three possible and widely used approaches to segment the skin color area from one image: using an explicitly defined skin region, using a nonparametric skin distribution model and using a parametric skin distribution model [27].

The fast and easily way to segment a skin color area from one image is to build a skin classifier based on explicitly defined boundaries in some color spaces through a set of rules. The nonparametric skin modeling method detects the skin color area by using training data which has been calculated and prepared before the system starts, such as a lookup table [28]. For each pixel, it only needs to check its skin color probability from the lookup table; therefore, it is considered to be fast in processing. However, it requires a large storage space for the training data, which is a challenge for most mobile devices. Furthermore, the performance of this method directly depends on the training images collected. If the set of the training data is too small, the performance may be even worse than the method based on explicitly defined boundaries. In order to improve the accuracy of the training data in nonparametric skin modeling, a new method, which uses Gaussian function to calculate the skin color distribution, has been presented. This method has been called parametric skin modeling.

3.1.3.3 Hand Detection Algorithm

One major task of our research is to detect the hand region by using a skin color segmentation method; therefore, in the first step of our research, we focus on searching and studying some previous hand segmentation methods. At the early stage, we aimed at methods using an explicitly defined skin region for they can provide real-time hand detection, especially for handheld devices. These methods are carried out in different colour spaces, such as RGB, HSV and YCrCb colour space. Table 3.1 shows some typical skin colour segmentation methods based on explicitly defined boundaries:

Table 3.1: skin color segmentation methods

Color Space	Rules
RGB	R > 95, G > 40, B > 20 max{R, G, B} – min{R, G, B} > 15 |R – G| > 15, R > G and R > B
Normalized RGB	Gup = -1.8423*r*r + 1.5294*r + 0.0422; Gdown = -0.7279*r*r + 0.6066*r + 0.1766; Wr = (r-0.33) * (r-0.33) + (g-0.33) * (g-0.33); g < Gup, g > Gdown, Wr > 0.004
YCrCb	133 <= Cr <= 173, 77 <= Cb <= 127
HSV	7 < H < 29

We have tested the performance of the skin colour segmentation methods listed in table 3.1 and the outputs have been shown in Figure 3.2. The method using RGB colour space seems to be more easily affected by the light condition (B). On the other hand, the hand detection performance can be affected by the background environment when normalized RGB, YCrCb color space and HSV colour space

are used (C, D and E).

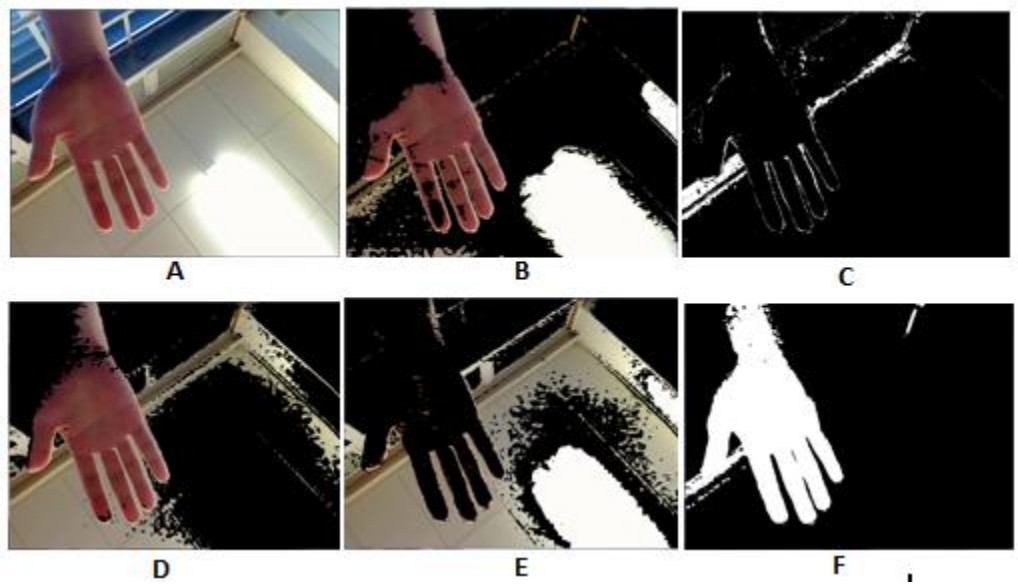

Figure 3.2: Input image (A); RGB colour space (B);
Normalized RGB (C); YCrCb colour space (D);
HSV colour space (E); YCrCb and Ostu threshold (F);

In order to decrease the noise produced by the background environment and the light condition, we moved to an alternative method that uses the Ostu threshold algorithm [29] and YCrCb colour space together to segment the hand region. In image processing, the Otsu algorithm is used to threshold the input image into two classes of pixels (e.g. skin colour and non-skin colour) and output the result as a binary image. The input image will first be transferred from RGB colour space to the YCrCb colour space. Then the Otsu algorithm will be applied to segment the skin color region. From the example shown in Figure 3.2, this approach appears more robust than others. However, the output is still affected by backgrounds with similar colour to human skin.

In order to achieve the aim of using the mobile device anywhere (any background environment) and anytime (any light condition), we considered using a parametric skin distribution model to improve the detection performance. We found a new method (Handy AR) which used training data set and a lookup table to detect the hand region [30]. This method also can perform in real-time and seems more robust in different background environment and lighting conditions. However, some changes and improvements need to be carried out for our mobile system.

We combine the Ostu threshold algorithm and the general skin color model to detect the skin color pixel this is different from the Handy AR which uses an adaptively learned histogram together with a lookup table to restrict the hand region. Our approach works as follows. The input image will be converted into a grey scale image first. A lookup table trained from a set of collected data and the Ostu algorithm are used to segment the skin color region separately. We compare the outputs of these two methods and only the regions considered to be skin colour region for both methods are treated as the hand area. Then, a distance transformation will be applied to find the point exhibiting the maximum distance value. The region that includes this point is considered to be the hand region and other regions are considered to be noise. Furthermore, this point will be recorded and used for the next frame to avoid the mistake when the hand moves out of the camera view.

3.1.3.4 Fingertip Tracking Algorithm

In order to locate the fingertips of the user's hand, we first used a curvature-based algorithm. We assume that when the curvature of the hand contour is larger than a defined threshold (here we define the curvature as the angle cosine value and the

threshold is 0.8), it is considered as the fingertip. However, as with the filtrated hand region shown in Figure 3.3, the hand region may be cut by the screen boundary. The two intersection points always have large curvature values and may be mistakenly considered as the fingertips by the algorithm.

Figure 3.3: Filtrated hand region (left); Finger region (middle); Fingertips located (the blue point is the one has the maximum distance value in distance transformation) (right)

In order to locate the real fingertips and solve the issue we discussed above, we found an alternative fingertip tracking approach based on Rajesh's method [31]. As we described, when we separate the hand contour in one image, we also locate the point that has the maximum distance value in distance transformation and consider it as the center point of the palm. In addition, the maximum distance D is considered half of the palm width and twice of one finger width. Then the fingers can be eroded completely by using an element with the width D/2 to only leave the palm region; therefore, the finger areas can be calculated by erasing the palm region from the whole hand region. For each finger area, we find the minimum area rectangle (a rotated rectangle fitted to the finger area) and calculate the midpoints of the four edges of the rectangle. The fingertip position is defined as the midpoint that is farthest from the center point of the hand region. The whole procedure is presented in Figure 3.3.

3.2 Working Systems Design

In our research, we have designed two working systems using gesture-based interaction methods for handheld AR and one controlled working system using touch based interaction method. The first working system is based on a client-server framework that uses a PC as the server to track user fingers and a mobile phone as the client to render the virtual objects. As this working system can only be used in a fixed experiment environment, we rebuilt our system on a tablet so that the server part and the client part can work together. We also design a touch based interaction method that could be used as a controlled working system to provide a comparative analysis. Table 3.2 shows an overview of these three systems:

Table 3.2: working systems

	Client-server	Tablet (gesture)	Tablet (touch)
System framework	Server (PC): detect and tracking fingertips; sending fingertip coordinate to the client; Client (mobile phone): rendering scene; providing gesture based interaction;	Using depth sensor to calculate the fingertip coordinate; rendering scene; providing gesture based interaction;	Rendering scene; providing touch based interaction;
Hardware requirement	Desktop PC; Android smart phone; Microsoft Kinect; wireless internet connection	Tablet; Primesense depth sensor	Tablet; Primesense depth sensor

Operation system	Server: windows 7 Client: Android 4.1	Windows 7	Windows 7

3.2.1 Gesture-Based Interaction using Client-Server Framework

3.2.1.1 System setup

Figure 3.4: (a) System framework; (b) System setup

We setup the whole system in a small workspace where the desktop computer server and the mobile phone client share the same interaction volume, as shown in Figure 3.4b. The Kinect is positioned above the desired interaction space facing downwards to identify the 3D position of the fingertip. Meanwhile, a printed reference image marker is placed on the table right below the Kinect, establishing the geometry transformation relationship between the server and the client cameras.

Since the Kinect and the mobile camera track the same target, and share the same world coordinate system (WCS) of the image marker, the fingertip's

coordinate in the WCS on the server side can be easily used on the mobile client. Therefore, by applying a coordinate transformation and using wireless data communication, the user can hold the mobile phone with one hand while interacting with the AR scene with the other hand.

3.2.1.2 System Implementation

The PC server uses the Kinect to acquire RGB and depth images and combines them together to enable a mapping from pixel to pixel in order to provide the depth information for each pixel in the RGB image. The fingertip 3D coordinates are detected by this calibrated RGB and depth output and sent to the mobile client for mobile AR 3D interaction via a wireless connection. The detailed process within the framework is illustrated in Figure 3.5.

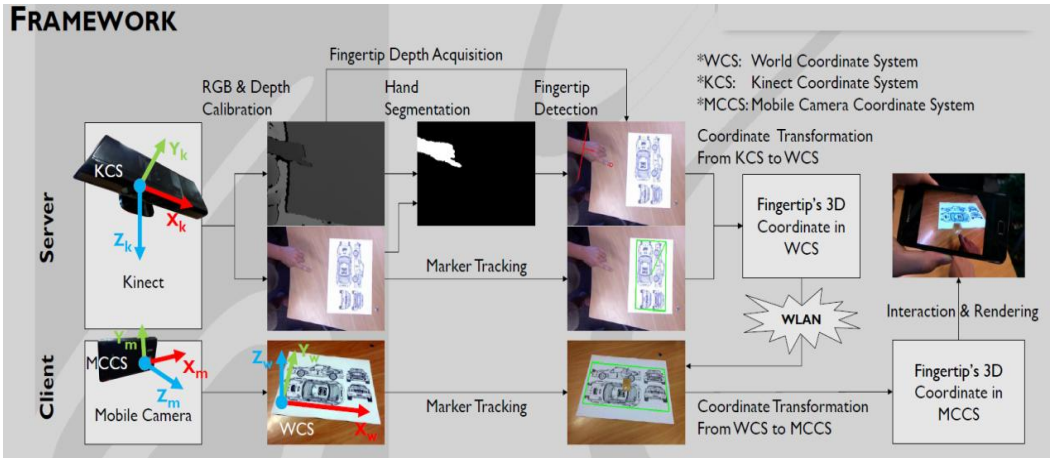

Figure 3.5: System Framework

Before we segment the hand region from the RGB frame, we first use a threshold (a distance between 35cm and 90cm from the Kinect camera) in the depth image to remove the noise, such as the table in the background with a similar color to human skin. Then we use the algorithm described in the previous

section to locate the fingertip position. The fingertip's depth that we defined is not the point value in the corresponding coordinate of the depth frame mapping from the RGB frame, but the average depth value of a small nearby region, which circularly fits the fingertip curvature while excluding the non-skin part. This can avoid the invalid depth value of a single fingertip point, which could be frequently located in the shadow of the infrared area in the depth frame.

The AR tracking is implemented using the OPIRA natural feature-tracking library [26]. We choose this library due to its robustness and fast computation time. The result shows that the library can track the marker in real-time and provide accurate marker based coordinate system in our application.

The system projects the fingertip 2D position and the depth into the marker's world coordinate system by using the homography matrix obtained from the server tracking section, and then sends this data to the mobile client. This 3D coordinate is projected again into the mobile camera coordinate system based on the homograph matrix calculated from the mobile tracking procedure.

We deploy a small socket library both on the desktop server and the mobile client, and use it for real time User Datagram Protocol (UDP) based communication. The data can be communicated in real-time in our private testing network.

3.2.1.3 Manipulation Design

Three atomic operations (translation, rotation, scaling) can be conducted to manipulate virtual objects by using finger gestures. To do this, we map the translation value onto the position change of a single fingertip movement. For

example, in Figure 3.6, the user holds the mobile phone with one hand, while stretching out the other and using one fingertip to interact with a virtual box to lift it up from a low position to a higher one in 3D space.

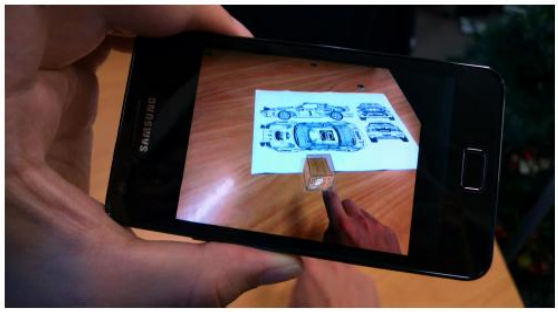

Figure 3.6: Translation Task

The rotation and scale values are controlled by the orientation and distance change respectively between two fingertips (the thumb and the index finger) in midair. We use the same rotation and scaling manipulation design in the following working system using tablet. In next section, when we introduce the tablet gesture based interaction system, we will provide a detail description about these two manipulation techniques.

3.2.1.4 System Performance

We run the server on a desktop computer (Intel 2.4Ghz Dual-Core, 4GB RAM, Windows 7) and the client on a smart phone (ARM 1.2GHz Dual-Core, 1GB RAM, Android2.3 OS). Our prototype runs robustly on the target phone at an interactive frame rate of around 10 to 15 fps with 640*480 pixel camera resolution while the network delay between the server and the client is around 20ms on average. The Kinect provides an error of less than half a centimeter when placed around 75cm from the marker plane. The manipulation accuracy of our

system is within 1cm under the physical WCS, which indicates that the prototype is not suitable for high-precision mobile AR applications, but it meets the basic requirements of general interaction tasks.

3.2.2 Gesture-Based Interaction using a Tablet

3.2.2.1 System Setup

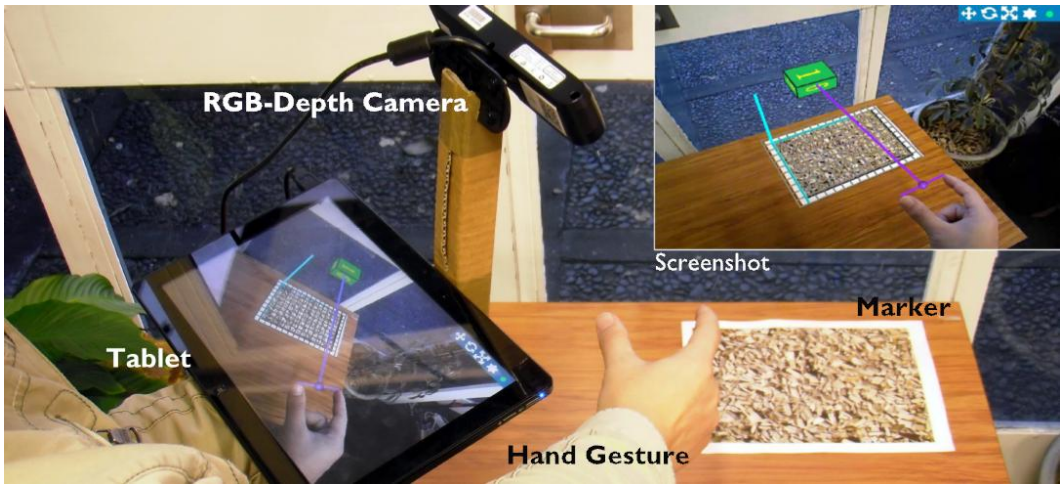

Figure 3.7: System Setup

To conduct gesture based interaction in a real handheld AR system, we built a prototype with a tablet connected to a Primesense Depth sensor via a USB cable. The reason for us to use Primesense Depth sensor instead of the Kinect is that this sensor provides a short detection range so that users can operate in a natural way. Users can hold the tablet with one hand while controlling virtual objects with the other hand in front of the sensor (Figure 3.7). The system can calculate the fingertip 3D position in the screen coordinate system using the depth sensor; then project it into the marker's world coordinate system, while the virtual objects are rendered in the same coordinate system. Our prototype interface improves the intuitiveness of AR manipulation by enabling the user to use a pinch-like gesture

with thumb and index fingers in six-degrees-of-freedom (6DOF).

3.2.2.2 System Implementation

The framework of this gesture based interaction system is almost the same as the client-server system. It just combines the PC server system and the mobile client system together into a tablet. RGB and depth images are acquired from the depth sensor and combined together to support a pixel to pixel mapping for the system. The fingertip coordinates are calculated based on the combined images and a full 3D manipulation is supported for the users by using the received fingertip coordinates.

Hand detection and fingertip tracking algorithms are the same as we used in the client-server system. The points with a distance larger than 90cm or smaller than 35cm from the sensor camera are removed from the RGB image first. The hand region is separated based on the skin-color segmentation algorithm. Then we use Rajesh's method [31] to separate the fingers from the whole hand region detected. The fingertips are located based on the fitted minimum area rectangle. As we discussed above, the fingertip's depth information is calculated by the average depth value of a small nearby region to avoid an invalid depth value in the combined depth image.

In order to detect whether the virtual object is selected or not, we project both the fingertip 2D position and the virtual object 3D position into the camera coordinate system to provide a comparative analysis.

$$\begin{bmatrix} F_{camera_x} \\ F_{camera_y} \\ F_{camera_z} \end{bmatrix} * K = \begin{bmatrix} F_{screen_x} \\ F_{screen_y} \\ 1 \end{bmatrix} \quad (1)$$

K is the camera matrix that is related to the camera's internal parameters. F_{camera} is the fingertip coordinate in the camera coordinate system and F_{screen} is the fingertip coordinate in the tablet screen coordinate system. The fingertip's z value is what we receive from the combined depth map. Based on the equation 1, we can calculate the 3D coordinate of the fingertip in the camera coordinate system.

For the virtual object in the world coordinate system, we get equation 2. R and T are the rotation and translation matrices, and P is the projection matrix. Therefore, we can build a relationship between the virtual object's camera coordinate (I_{camera}) and the world coordinate (I_{world}). Based on this relationship, we can find the virtual object position in the camera coordinate system, and future manipulations to the object are feasible.

$$\begin{bmatrix} I_{camera\_x} \\ I_{camera\_y} \\ I_{camera\_z} \end{bmatrix} * K = \begin{bmatrix} I_{screen_x} \\ I_{screen_y} \\ 1 \end{bmatrix} = \begin{bmatrix} I_{world_x} \\ I_{world_y} \\ I_{world_z} \end{bmatrix} * T * R * P \quad (2)$$

3.2.2.3 Manipulation Design

We directly use the absolute position, like current 3D position of the fingers, as the input value for the interaction of the virtual objects. Since the user conducts finger movement in physical space, there was no limitation on the interaction volume as long as the hand was visible inside the camera image.

We use pinch-like gestures with the thumb and index fingers to select a virtual target, which is very similar to how people grasp real objects in their daily life. When the midpoint ($M \in \mathbb{R}^3$) of two detected fingertips is completely located

inside the geometric space of the virtual object ($V \in \mathbb{R}^3$), we define that the virtual object is selected and is attached to the fingers for further possible manipulations. We use the space distance $|MV|$ comparing with a predefined threshold of $\sigma = 50\text{mm}$ to check whether the midpoint of fingers is close enough to the virtual object center. If the distance between the two centers is shorter than system's pre-defined threshold, the object will turn red to show that it is has been selected (see Figure 3.8).

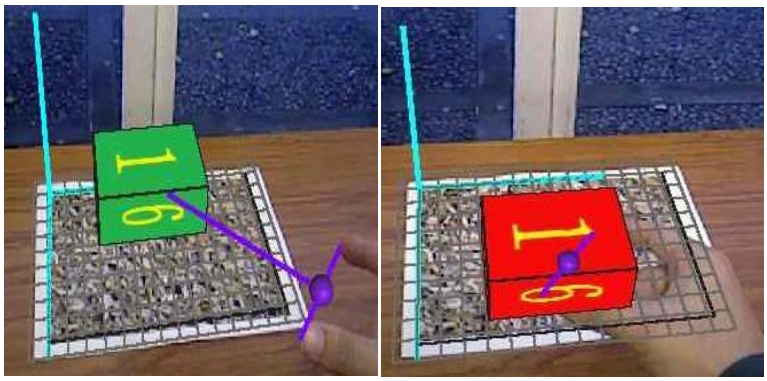

Figure 3.8: Selecting a green box with two fingers (left); The successfully selected box is turned into red (right).

For each task, the user has to select the green object located in original point first to activate the manipulation procedure. As demonstrated in Figure 3.8, a small purple ball is placed between the two detected fingertips as the valid input point in 3D operational space, working like a cursor on desktop computers. This is referred to as a "space cursor" in the following description. A thin purple line is also displayed from the centre of the space cursor to the green box, visually assisting users to understand the space distance.

The selection can be cancelled by using a popular countdown timer method: keeping the midpoint of two fingertips relatively still inside a tiny space region

($10 \times 10 \times 10$mm in our case) around the current position longer than a certain time (2.5 seconds in our case), and the object selection state will be canceled, turning the target from red back to its original color. While waiting for the cancel operation to complete, there is a white circle growing in an anticlockwise motion surrounding the midpoint counting down the remaining time.

The reason for using a RGB-Depth camera is that we can retrieve the depth image synchronously instead of only the RGB frame generated from the normal camera. With this we are able to extract the 3D position coordinates of the detected fingertips and project them into the AR marker's world coordinate system as a reference for the virtual object. For object translation, we directly use the absolute position of two fingertips' midpoint in space as the 3D location of the geometric center of a virtual object to complete the translation. That is to say the virtual object is moved with the midpoint of the two fingers after it has been selected.

We define the rotation input as the absolute 3D pose of the line connecting the user's thumb and index fingers in space. Considering detected fingertips $P_1, P_2 \in \mathbb{R}^3$, the connecting vector $\vec{v}(P_1, P_2)$ has a pose in space to the original points. When a user rotates his/her two fingers in front of the camera, subsequently the pose vector $\vec{v}(P_1, P_2)$ will change the angles to the coordinate axes. This can be used as rotation input and final pose of the virtual object.

We utilize the space distance between the two fingertips as the input value for scaling. The proposed scaling algorithm supports non-uniform scaling along each axis. Detecting two fingertip positions at time t can be used to construct a vector $\vec{v}(P_1, P_2)$. Based on the vector pose and position, we determine whether the user moves both fingers together or apart and compute the length of movement. Then,

the amount of scaling S is calculated by $S = \left|\overrightarrow{v'}(P'_1, P'_2)\right| / |\vec{v}(P_1, P_2)|$. For positive scaling S is bigger than 1, for negative, the value is smaller than 1.

As we know, the human hand is not a rigid object, and could have a deformable transformation in several axis directions when doing a small rotation gesture. This will lead to possible rotations about other axes even though the user may not intend to make them, which can significantly increase the manipulation complexity. To avoid this type of input confusion, we setup an initial rotation option that no axis will be initially active. In other words, the user needs to choose which axis he or she wants to rotate about before manipulating the object. This completely removes the interference effect among different axes while rotating an object. Since the scaling gesture may have the same interference on the axis while performing the operation, we adopt the same strategy for scaling configuration as the rotation, and users have to explicitly choose the target axis.

Figure 3.9: The mode menu and axis button are located on the top right corner, and activated ones are highlighted in red.

To support an axis selection for the users, we reduce the interaction complexity by splitting 6DOF transformations into a single 3DOF translation, 3DOF rotation and an extra non-uniform 3D scaling. These three different entities can be chosen or switched respectively by clicking the corresponding mode button. When a trackable target is identified correctly and tracked successfully, the user keep the handheld AR device in an appropriate position with a clear view of overlaid virtual object, and then mode switches will be available for users to choose, which is accomplished through a simple button interface. As illustrated in Figure 3.9, three modes, translation, rotation and scaling, are presented. A new mode can be selected and activated at anytime by pressing corresponding icons, but only one mode can be activated at each moment and the icon will be highlighted in red color as showed in Figure 3.9 to indicate user's current working mode. Thus, the system will be configured in a specific operation mode in which only the selected manipulation type has an actual impact on the virtual object state.

In addition, three axis buttons are located just below the mode menu for users to choose which axis he/she wants to operate on. The coordinate axis uses the exact same world coordinate system as the tracked AR marker, which is drawn on the top left corner of the marker in green lines in Figure 3.9. Being different from the mode switches, multiple axes can be activated simultaneously for combination input, and selected buttons are highlighted for visual aids.

Finally, with options of mode and axis, we map gesture movements in the space into a state change of the selected virtual object in the AR scene. However, no input will be accepted if AR visual tracking fails, in which case the menu icons and buttons will be displayed in grey to show a tracking failure has occurred. When the tracking resumes again, the previous operational environment like the selected mode or axis will be resumed immediately.

3.2.3 Touch Based Interaction using a Tablet

In our study, we intend to conduct a comprehensive user study to evaluate our gesture based interaction technique by comparing it with a screen-touch interface; therefore, we also designed a touch based interaction system using a tablet and web camera

3.2.3.1 System Setup

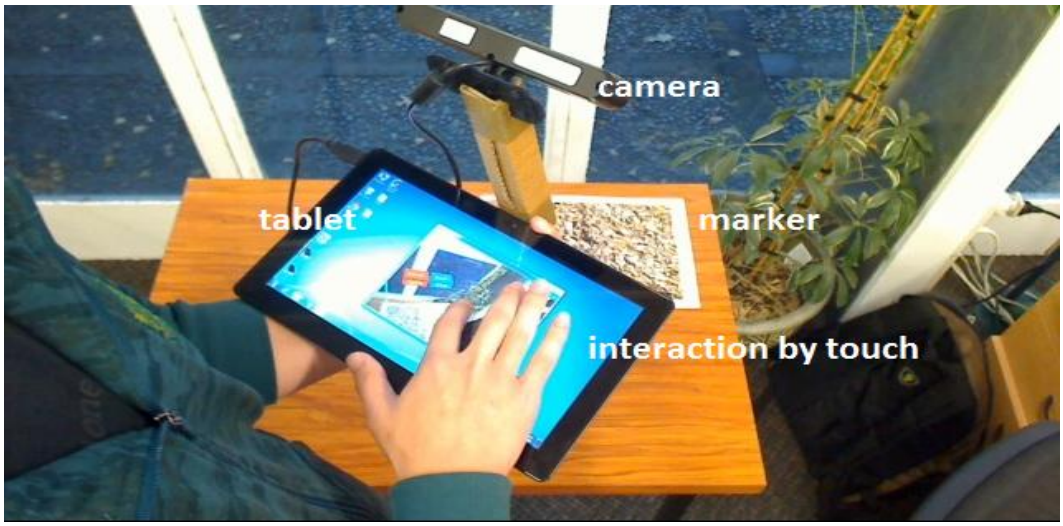

Figure 3.10: System Setup

We setup the touch based interaction system almost the same as the gesture based interaction system using tablet. In our research, we still use the Primesense depth sensor to get the video input; however, in the touch based interaction system, we do not need to require any depth information from the depth sensor; therefore, the Primesense depth sensor can also be replaced by any types of USB web cameras.

From Figure 3.10, the system uses one camera (facing downwards) to track the AR marker and build the AR world coordinate system. The user can use one

finger to interact with the virtual objects by clicking and sliding on the touch screen.

3.2.3.2 System Implementation

The touch based interaction system includes three simple components: the AR tracking system, the rendering system and the interaction system. The AR tracking is implemented using the OPIRA natural feature-tracking library [19], the same as used in the previous two working systems. As we mentioned above, we use OpenGL library to rendering the AR scene. OpenGL library is a multi-platform API and can be used to interact with a GPU to achieve hardware accelerated rendering. Because we use the Microsoft Windows 7 as the operating system, the touch manipulation works like a cursor on desktop computers in some way.

3.2.3.3 Manipulation Design

For screen-touch input, because our handheld tablet does not support multi-touch functionality, we designed all touch interaction around using a single finger.

As a controlled experiment group, the screen-touch method has the same mode switches design as the gesture-based interaction described in Section 3.2.2.3. When the mode selection is done, we select a virtual object using the screen-touch interface by clicking on it. Once the user clicks on the touch screen, the 2D coordinate of the clicked point will be back-projected into the world coordinate system of the AR trackable marker. A virtual ray is cast from the virtual camera's position in the direction of the back-projected 3D point into the handheld AR scene. Therefore, we can receive a line L between the near clipping plane and the far clipping plane. By calculating the distance from the virtual object to L, the

virtual object is considered as successfully selected if the distance is less than a predefined threshold (10 pixels in our system). The user can move the handheld device to change the view of the virtual scene to select a virtual target that may be partially overlapped by other objects.

When the mode and axis are selected and activated, any screen touch input can be converted into a change of the virtual object's position, pose or size. We map the translation, rotation or scaling values onto the position change of a single fingertip scrolling on the touch screen.

For translation we define the translation input as the absolute fingertip scrolling distance change on the touch screen. If a fingertip selects the virtual object at point P_1 and moves on the touch screen to point P_2, the distance between P_1 and P_2 is used as the virtual object translation along the axis selected.

For rotation we use the fingertip coordinate change percentage as the input value. For example, the user selects a virtual object at P_1 with the x position C_X and moves to P_2 with a new x position $C_X^{'}$. The changed angle around the selected axis can be calculated as $360 * \frac{(C_X^{'} - C_X)}{C_X}$.

For scaling, almost the same as the rotation tasks, the percentage of fingertip coordinate change is defined as the input value.

In some cases, the user can separately scroll on the screen several times with touch breaks to complete a long distance translation because of screen size limitation.

3.3 Summary

In this chapter, we discussed the hand detection and fingertip tracking algorithms used in our research. Based on these algorithms, we designed two gesture-based interfaces for handheld AR. Both of them supported users to interact with virtual objects in a 3D AR environment using fingers in midair. We used one finger based selection and translation manipulation in the client-server system, while using all pinch-like gestures to manipulate the target virtual item in the tablet based system. In addition, we also designed a touch based interaction method as a controlled working system to provide a comparative analysis.

Chapter 4

Experiment and User Study Design

In our research, we conduct a comprehensive user study to evaluate our gesture based interaction technique by comparing it with a touch-screen interface. However, the client-server system required a certain workspace with a desktop PC, so it was difficult to find enough participants for this condition. Therefore, we only focused on studying the performance and usability of the gesture based interaction system and touch based interaction system using the tablet. We expected that gesture based interaction should perform better for it is more natural and performs as the real-world interaction experience.

In the first section of this chapter, we will describe the experiment setup, the participants, and questionnaire and measurement methods. In the second section, we provide an overview of the experiment procedure.

4.1 Experiment Setup

The user study was set up as a within group test, which means that every subject tested both interfaces (gesture based interaction and touch based interaction). The experiment started with a training demo, used to teach the subjects how to use the system. After this introductory process, the subject had to perform a group of tasks for each interface.

4.1.1 System Prototype

The practical system ran on a Samsung XE500T1C tablet, featuring an Intel(R) Core™ i5 CPU, 4.00 GB RAM and an Intel(R) HD integrated graphics, running the Windows 7 operating system. The Primesense depth sensor was used as an attached RGB-Depth camera, providing RGB and depth frames to the tablet via a USB connection (see Figures 3.7 and 3.9). However, the sensor requires an indoor operational environment, which means the output can be influenced by different lighting conditions. Furthermore, the sensor has an operation range of between 350mm and 1400mm. In common handheld gesture interaction, the user uses their hand to interact with objects only a little far from the tablet; therefore, we attached the sensor a little higher than the tablet to make sure the interaction space below the tablet was inside the sensor operation range.

4.1.2 Experiment Setup

We set up the user study using a within-group factorial design where the independent variables were manipulation technique and task scenario. The manipulation techniques are our proposed novel natural 3D gesture interaction method and traditional 2D screen-touch input way, while the test scenarios contained three different experimental tasks with varying subtasks. The dependent variable is task completion time. Furthermore, we measured user preferences for

both techniques in terms of usability.

Table 4.1: Per-condition Questionnaire

The given interface was	
Q1	easy to learn
Q2	easy to use
Q3	natural (as the way you expect or get used to)
Q4	useful to complete the task
Q5	NOT mentally stressful
Q6	NOT physically stressful
Q7	offering fun and engagement

To begin the study, each participant was asked to complete a pre-test questionnaire about age, gender and prior experience with touch-based mobile devices, 3D gaming interfaces and Mixed or Augmented Reality interfaces. A brief introduction of handheld AR concept was then given to the participants, followed by a detailed instruction of the 2D touch and 3D gesture manipulations used in our testing environment.

Specifically, the participant will learn the general operation attention, basic interface usage, and overall task content. Afterwards, each participant had five minutes to practice both interaction techniques. Once they started the study, they were not interrupted or given any help. Upon the completion of the practical task using each interaction method, they were asked to fill out a per-condition questionnaire (Table 4.1) and gave further comments on a post-experiment questionnaire (Table 4.2) at the end of the evaluation. The whole user study took

approximately 45 minutes on average for each participant.

Table 4.2: Post-experiment questionnaire

Based on previous test	
Q1	Which interface do you prefer to use if you will have to do a similar task again?
Q2	When determining how much you like using a manipulation technique for handheld AR, how important in influence on your decision was ease, speed and accuracy?
Q3	Please briefly explain the reason you chose the interface above.
Q4	Did you have any problem during the experiment?
Q5	Any other comments on the interface or the experiment?

For the evaluation, we measured the preference of participants with seven questions related to the usability (shown in Table 4.1), and used a nine point Likert-scale (1 to 9 with 1 indicating strongly disagree while 9 indicating strongly agree) for each subjective questionnaire item. In addition, we also asked participants to provide some comments according to the questions shown in Table 4.2. Furthermore, we configured our testing system to automatically measure the performance of participants in terms of task completion time.

4.1.3 Participants

A total of 32 participants, 16 males and 16 females, were recruited from outside of the university for the experiment. Their ages ranged from 17 to 52 years old (M = 35.03, SD = 10.28). All participants were right handed. During the experiment tests, 29 participants held the device in their left hand and used the right hand for interaction, while for the other three it was the other way around. No significant differences could be observed regarding handedness. All of them used the index finger for touch input and the index finger with the thumb together for gesture manipulations. Although 27 of them used touch screen devices frequently, only six of them had some experience of using 3D interfaces, mainly from the game consoles like Microsoft Xbox and Nintendo Wii. All of them had no previous experience with using Mixed Reality or Augmented Reality interfaces. Table 4.3 gives an overview of user numbers based on their prior experience.

Table 4.3: Users' prior related experience

	Inexperienced	Experienced
Mobile touch interface	5	27
3D interface	26	6
Mixed Reality and Augmented Reality interface	32	0

4.2 Experiment Scenarios

Based on the work of Bowman [32], we used the basic canonical manipulation tasks "translation, rotation, and scaling" for task design, and built three specific

scenarios with several subtasks to cover typical 3D manipulation situations in handheld AR applications. To manually identify the desired object for subsequent manipulation, a selection operation selection was used.

Each participant was asked to perform several experimental tasks using the two interaction interfaces (traditional 2D touch and novel 3D gesture interaction) respectively. Interfaces were presented in a random order to the participants to exclude potential learning effects. One additional selection and three types of essential manipulations (translation, rotation, and scaling) were included in tasks for each interface test. The order of tasks and related sub-tests were randomized for each participant to avoid any order-related influences on the results.

The experimental task was to select and manipulate a virtual cube in a handheld AR application and match it to the indicated target position, pose or size. For all tasks, our system set the target in blue, and green for the object the participant hoped to control, and red for the selected object which was currently being manipulated. All test tasks were presented in the same virtual AR background environment. A black and white textured plane printed to an A4 sized piece of paper acted as a target marker for the AR tracking. Meanwhile, both the indicated target and manipulated object were clearly displayed on the same screen, and the participant could inspect the scenario from different perspectives by moving the tablet freely to understand the task before officially conducting the actual test (see Figure 3.7 for gesture interface and Figure 3.10 for touch interface).

Participants were told to perform the task as fast and accurate as possible. The timing of the task was automatically started after the mode and axis were activated and the virtual object was successfully selected, and was stopped automatically when the task was completed. The task was considered completed

when the cube was changed to the required state in the space by the participant's operations.

4.2.1 Translation Tasks

In the translation task, participants had to move a box in space to three different target positions with various directions and distances as well as heights. Target positions were drawn in blue lines as showed in Figure 4.1, and were relocated to different positions once the current task is completed. We defined that the box is translated to the target position if the space distance between their center positions is less then 5mm in the physical world. Three translation subtasks with various 3D positions were tested in total, in using blue line targets placed around the Z-axis clockwise at angles of 30, 180, 315 degrees square pose with distances of 500, 750, 350mm and heights of 500, 150, 250mm correspondingly to the origin. Figure 4.1 provides an example task of gesture based interaction system. The tasks were the same in the touch based system.

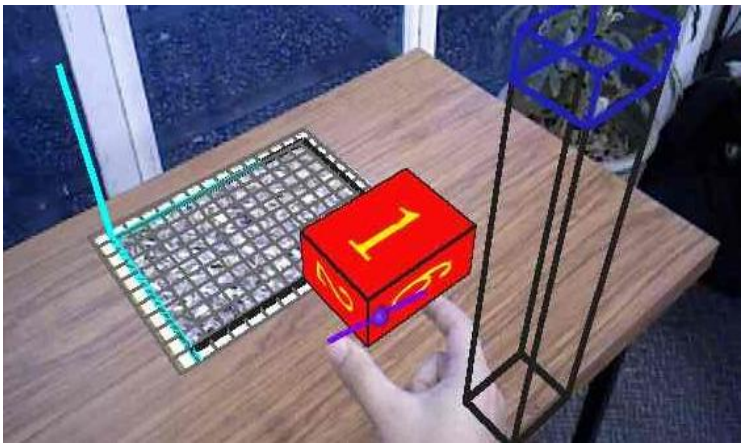

Figure 4.1: Translate a selected box to a position in blue

4.2.2 Rotation Tasks

In the rotation task, participants had to rotate a box around three axes clockwise individually at angles of 25, 120, 225 degrees from the initial horizontal pose. However, if we directly use a similar AR scene as the previous translation test, rotation tasks would have serious surface identification and overlapping issues between the operated object and the blue target. To make every target pose clear enough to the subject, we placed the target next to the operated object for comparisons, and labeled different numbers on each box surface for instructions (Figure 4.2). The target would be directly relocated by the system for the next round test if the selected one was rotated to the required pose as the target presented, which means that the space angle around each axis between their poses is less than 1 degree in the physical world. This solution can significantly reduce confusion and eliminate the overlapping problems, while keeping the test result as fair as possible. In the touch based interaction system, the rotation tasks were the same with the task shown in Figure 4.2.

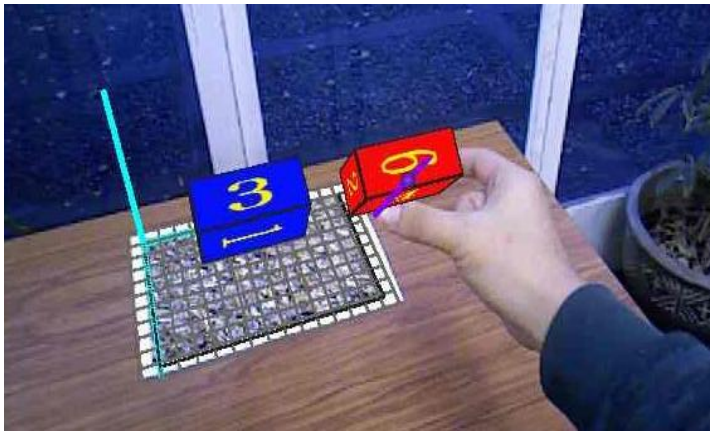

Figure 4.2: Rotate a virtual box to the target pose

4.2.3 Scaling tasks

In the scaling task, participants had to change the size of a virtual box along three axes individually with 0.4, 0.8 times smaller and 1.6 times larger on both directions asynchronously (Figure 4.3). The system had the same working principle to automatically update the resized testing target when the subject could scale the operated object with a dimensional discrepancy that is less than 5mm along the same direction of the same axis as the target. The touch based interaction system provided the same scaling tasks as the gesture based interaction system.

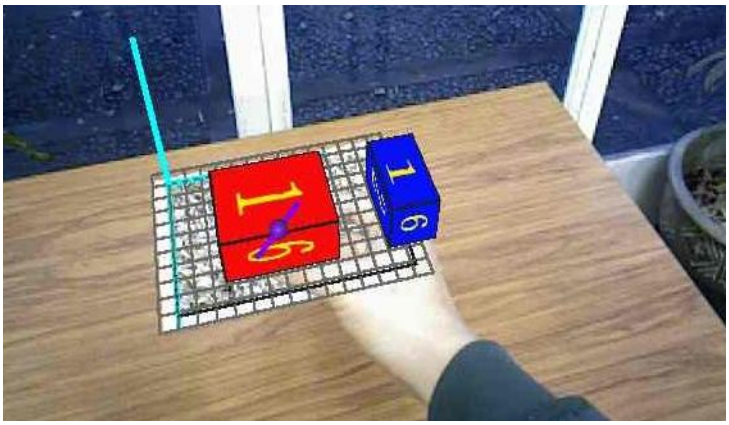

Figure 4.3: Scale a virtual box to change its size

Chapter 5

Results

In this chapter we present the results from user study described in chapter 4. A total of 32 participants, 16 males and 16 females, were recruited for this experiment. For each participant, it took approximately 45 minutes on average to finish the experiment, including system introduction, using the two interaction conditions (gesture based and touch based) and the questionnaire. The experiment was performed within a controlled indoor environment to avoid adverse tracking effects caused by varying light conditions.

As described in chapter 4, the experiment was conducted under two conditions. In touch-based condition, participants were asked to manipulate a virtual box by clicking and scrolling on the tablet touch screen using the index finger. In gesture-based condition, participants interacted with the virtual box by using pinch-like gestures.

In section 5.1 we evaluate the task performance of users in the two different conditions. In section 5.2, we evaluate the subjective feedback from the participants. Finally, in section 5.3, a discussion is provided based on the result of data analysis to explain the issues we found during the experiment.

5.1 Performance Analysis

To analyze which interface condition performed better, we looked at the task completion time of participants during their tests. In this section we report on the objective performance analysis, while in section 5.2 we report on the subjective questionnaire feedback.

For each interface condition, users completed three groups of tasks to cover typical 3D manipulation situations in handheld AR applications: translation, rotation and scaling. The system automatically recorded the task completion time for each of these three tasks for each participant. In addition users performed a selection task, however it was hard for the system to detect when the selection operation started; therefore, we do not provide objective performance measures for selection. As we mentioned in section 4.2, each participants had to take 3 translation tasks, 9 rotation tasks and 9 scaling tasks.

To analyze the collected performance times, we performed a Wilcoxon Signed-Rank Test using the Z statistic to test if there was a significant performance difference between two conditions. The Dependent T-Test for analyzing time performance was not used because the data collected was not normally distributed.

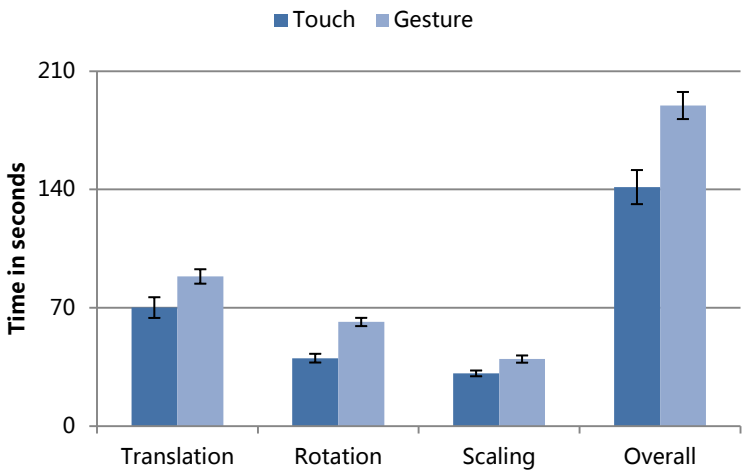

Figure 5.1: Mean completion time for overall and each task

Figure 5.1 shows the average time to perform each of the three types of tasks for both conditions, as well as the overall average across all tasks.

Table 5.1 shows the Wilcoxon Signed-Rank Test result comparing the performance times between the two conditions. We found a significant difference at the $p<0.0001$ level between the two condition in terms of performance on each type of task and overall completion time. On average the tasks were performed significantly faster with the 2D screen-touch method than with 3D gesture-based interaction. Therefore, we can conclude that subjects perform significantly better with the touch based method than with the gesture based interface. This goes against our expectation that gesture based interaction should perform better for it is more natural and performs as the real-world interaction experience.

Table 5.1 Wilcoxon Signed-Rank Test

Translation		Rotation		Scaling		Overall	
Z	*p*	*Z*	*p*	*Z*	*p*	*Z*	*p*
-4.020	0.000058	-4.937	0.000001	-4.619	0.000004	-4.862	0.000001

5.2 Subjective Evaluation

In addition to task performance, we collected the participants' subjective feedback using several questionnaires. Subjects were asked to answer the seven questions listed in Table 4.1 after finishing each basic manipulation task (translation, rotation and scaling). Subjects were also asked to provide answers for the same seven questions about selection operation after they had finished all the subtasks for each interaction interface. Finally, subjects also provided feedback about how they thought they performed for each basic manipulation task. All of these questionnaires were answered on a nine point Likert-scale (1 to 9 with 1 indicating strongly disagree while 9 indicating strongly agree).

To evaluate the quantitative data gathered from these questionnaires, we also used the Wilcoxon Signed-Rank Test for significant difference between the two conditions, using the Z statistic at the $p < 0.01$ level.

Table 5.2 provides detailed Wilcoxon Signed-Rank Test results for all seven questions across all conditions. In the rest of the chapter, we discuss the results for each individual manipulation task between the two conditions.

Table 5.2: Wilcoxon Signed-Rank Test: seven questions

	Translation		Rotation		Scaling		Selection	
	Z	*p*	*Z*	*p*	*Z*	*p*	*Z*	*p*
Q1	-3.532	0.000412	-3.500	0.000465	-2.887	0.003892	-4.973	0.000001
Q2	-1.170	0.241887	-4.476	0.000008	-3.900	0.000096	-5.064	0
Q3	-1.057	0.290578	-1.615	0.106209	-1.155	0.248213	-4.291	0.000018
Q4	-0.943	0.345779	-4.233	0.000023	-2.646	0.008151	-4.640	0.000003
Q5	-1.000	0.317311	-1.732	0.083265	.000	1	-2.828	0.004678

Q6	-3.750	0.000177	-4.409	0.00001	-4.455	0.000008	-5.055	0
Q7	-4.666	0.000003	-4.821	0.000001	-4.911	0.000001	-4.816	0.000001

Figure 5.2 shows the subjective questionnaire results following the translation manipulation task. We found significant difference in terms of (Q1) ease-of-learn ($z[n:32]=-3.532$, $p<0.0005$), (Q6) not-physically-stressful ($z[n:32]=-3.750$, $p<0.0002$), and (Q7) fun-and-engagment ($z[n:32]=-4.666$, $p<0.0001$). Users felt that when performing translation tasks, the gesture-based interface was not as easy to learn as the touch one, was physically more stressful to some extent. In contrast, they felt gesture based interaction was more fun and engaging than touch input. The gesture translation was also ranked higher in terms of its naturalness, but not significantly so. There were also no significant differences found for all of the other four questions in the translation task ($p>0.2$ for all responses).

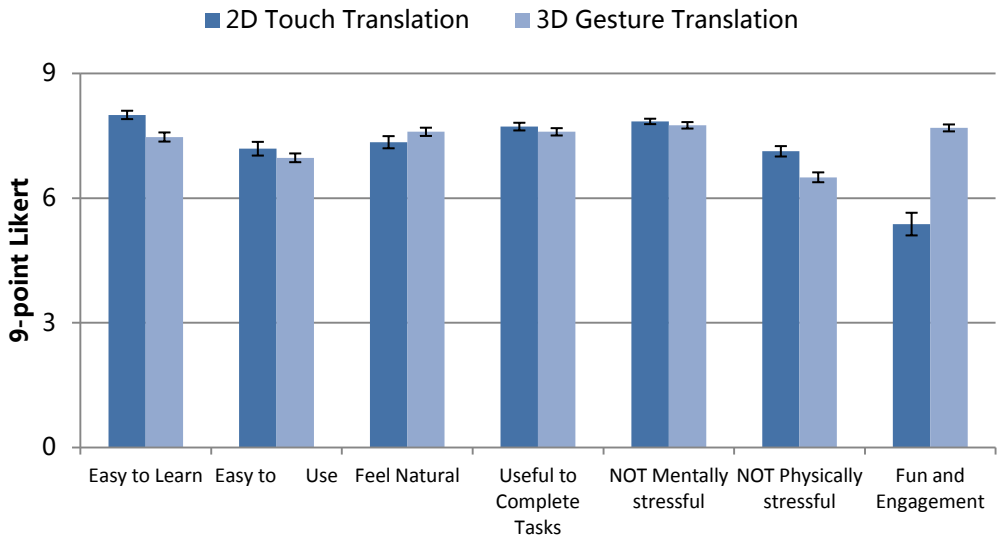

Figure 5.2: Users' average rating for the translation task

Figure 5.3 shows the subjective questionnaire results following the rotation

manipulation task. We found significant difference in terms of (Q1) ease-of-learn (z[n:32]=-3.500, p<0.0005), (Q2) ease-of-use (z[n:32]=-4.476, p <0.0001), (Q4) useful-for-tasks (z[n:32]=-4.233, p<0.0001), (Q6) not-physically-stressful (z[n:32] = -4.409, p <0.0001), and (Q7) fun-and-engagement (z[n:32]=-4.821, p<0.0001). Similar to the translation results, gesture based rotation method was not thought to be more easy to learn or less physically stressful than touch interaction. In addition, it was considered not so easy to use， and not so useful for the task completion compared to touch input. However users did feel that the performing rotations with gesture input was more fun and engaging. There was no significant differences found for the questions about (Q3) naturalness (z[n:32]=-1.615, p>0.1062) and (Q5) metal stress (z[n:32]=-1.732, p>0.0832).

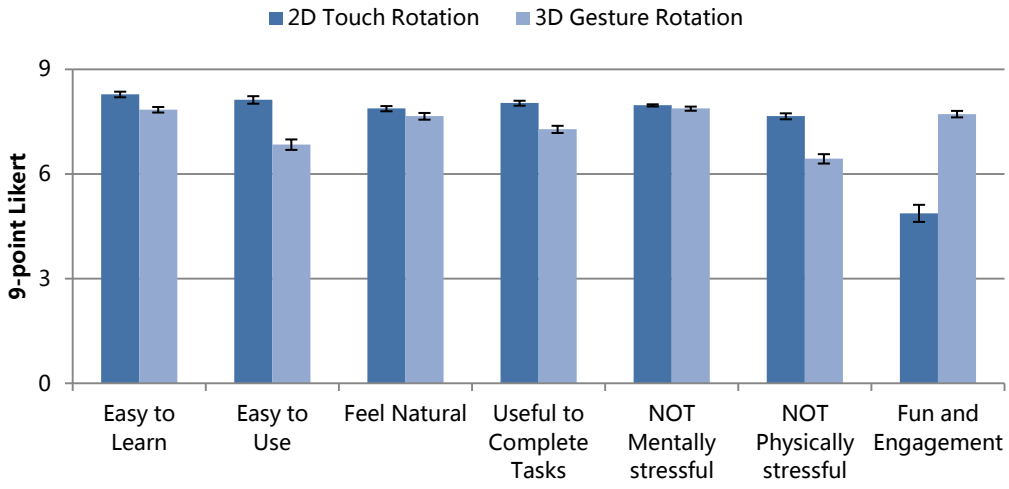

Figure 5.3: Users' average rating for the rotation task

Figure 5.4 shows the subjective questionnaire results following the scaling manipulation task. From table 5.4 we can see there was a significant difference on most questions except the (Q3) naturalness (z[n:32]=-1.155, p>0.2482) and (Q5) not-mentally-stressful (z[n:32]=-0.000, p<0.9999). As illustrated in Figure 5.4, users felt that the gesture-based method was as natural and non-mentally-stressful

as the touch interface. However, participants felt the gesture interface was more physically stressful to use and it was not as easy to complete the scaling task while using gesture to interact with virtual objects. Similar to the earlier results, gesture interaction was thought to be more fun and engaging.

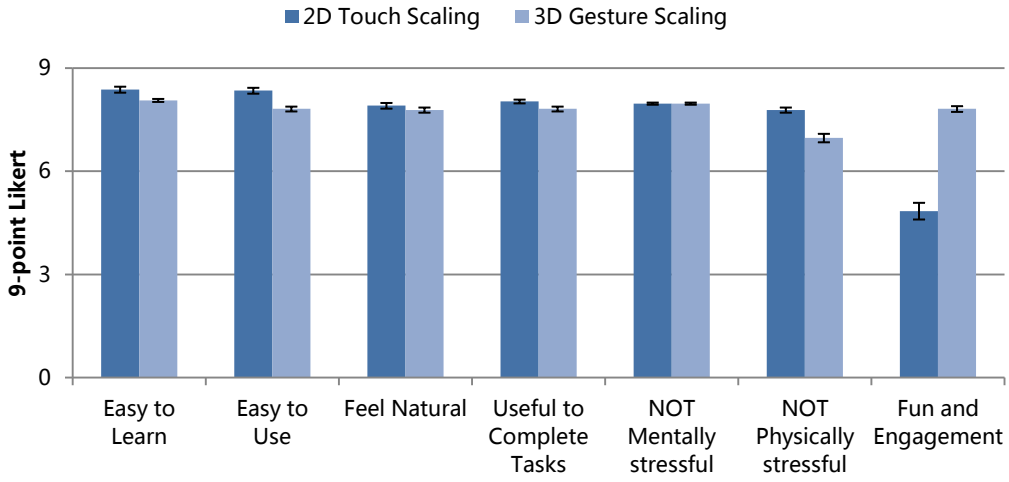

Figure 5.4: Users' average rating for the scaling task

In terms of object selection there was a significant difference between conditions across all the questionnaire items ($p<0.005$ for all), as shown in Figure 5.5. Looking in detail at users' rating in each question, the 2D touch-based selection was significantly preferred by the subjects except in terms of fun-and-engagement part. This may be because users are not used to selecting virtual items from space by using gestures without being able to feel them.

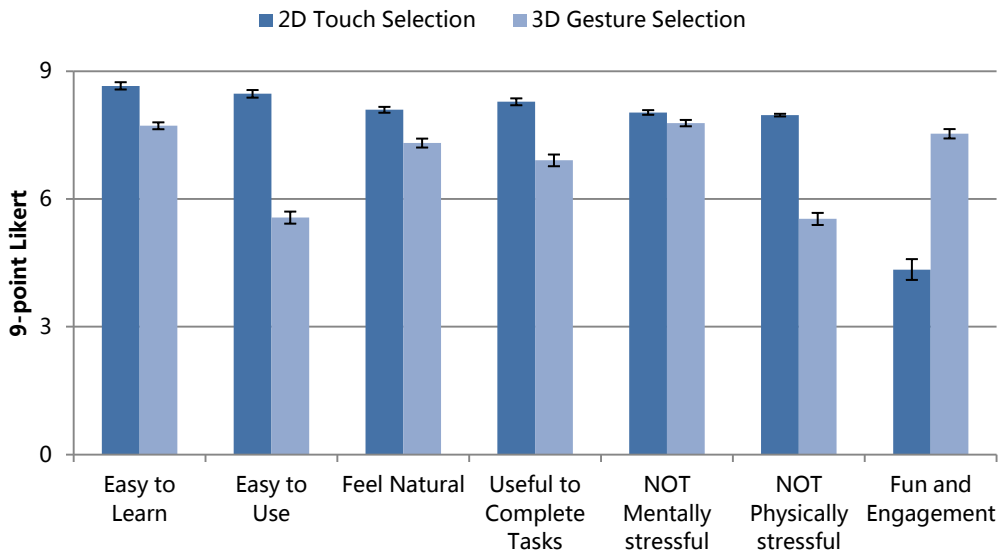

Figure 5.5: Users' average rating for the selection task

Reviewing all the results above, we found that subjects gave different ratings to the same questionnaire items for different interaction tasks. However, there was no significant difference between conditions found in terms of naturalness and mental stress for all the manipulation tasks.

Table 5.3 shows the Wilcoxon Signed-Rank Test result from the question about how participants felt they performed in each condition. There was a significant difference in perceived performance between all four types of tasks: translation ($z[n:32] = -2.982$, $p < 0.01$), rotation ($z[n:32] = -4.021$, $p < 0.0001$), scaling ($z[n:32] = -3.162$, $p < 0.01$) and selection ($z[n:32] = -4.924$, $p < 0.0001$). Therefore, we can conclude that participants felt that they performed better while using touch-based interface than the gesture interface. One possible reason for this was that the subjects were more familiar with touch manipulation than gesture manipulation in handheld applications.

Table 5.3 Wilcoxon Signed-Rank Test: participants' performance

Translation		Rotation		Scaling		Selection	
Z	*p*	*Z*	*p*	*Z*	*p*	*Z*	*p*
-2.982	0.00286	-4.021	0.000058	-3.162	0.001565	-4.924	0.000001

Users were asked to choose for each manipulation condition in terms of which interface they preferred using it for interaction. For all of the tasks, interaction based on touching was the subject's first choice, as shown in Figure 5.6.

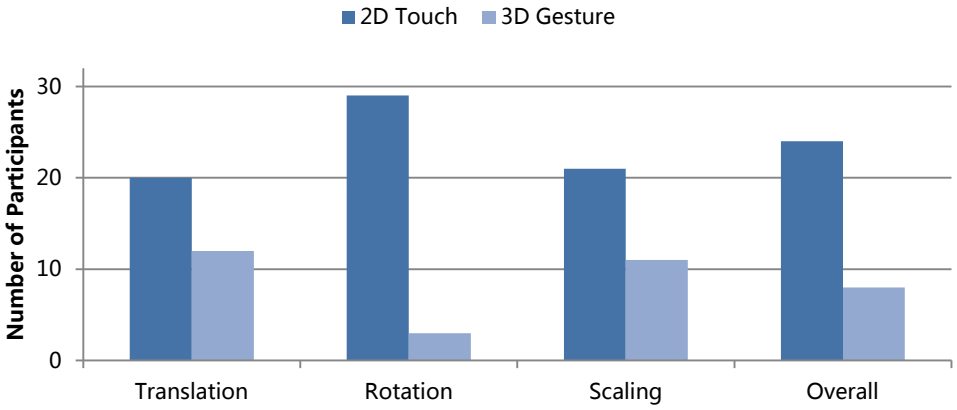

Figure 5.6: Users' overall preferences for similar tasks.

During the experiment, most participants usually stood in a fixed position to do all the interaction tasks even when they could not get an ideal view of the virtual objects and their hands. Therefore, while they used the gesture-based interface, participants found that the thumb may cover the index finger, which caused the fingertip tracking lost and increased the interaction complexity. In addition, when participants used gestures to rotate the virtual cube, sometimes they failed to control the rotation angle. In other words, the angle they rotated was always either too large or too small than the required rotation angle. We provided much more

time for the participants to learn how to use gesture interface than touch interface during the experiment; however, they still performed worse than we expected.

5.3 Discussion

Although our gesture based interaction approach can provide a more interesting and engaged experience for users to manipulate virtual objects, most participants still prefer to use screen-touch interaction for handheld AR input. One reason for this could be that most participants in our research use touch input devices such as smart phones frequently; therefore, they are used to the touch manipulation and feel comfortable when they use the 2D touch input. In contrast, the 3D gesture technique required all participants to learn a new method, and the method itself was not so easy to use for people who don't have much experience with 3D or AR interfaces.

There are several reasons that could explain why our system is not as attractive as we expected. First of all, when users use their fingers to select virtual objects in 3D space, they have no physical feedback from the objects, which could increase the selection difficulty. In our system, although we obtain depth information for the fingertips from the depth camera, the virtual objects are simply overlaid on the video frames and always cover the user's fingers no matter how deep they are. This can generate visual confusion and increase operational complexity. Secondly, most participants prefer using pinch-like gestures with two fingers to rotate or scale the virtual target, but they insist on one finger selection since it is much straightforward and effective. Therefore, when they use to fingers to select, the distance between two fingers is either too large or too small than the size of target virtual object, which may increase selection complexity during operation. Thirdly,

after starting the manipulation, users normally hold the tablet in a fixed position while the other hand is stretched out to touch the virtual object. However, it often happens that the hand moves out of the vision of camera, which can cause the fingertip position to be lost, and participants may lose their patience to some degree, which causes a negative influence on the coming tasks. In addition, when we use two fingers to rotate the target, one finger may cover the other in some certain angles. Therefore, rotation has certain practical limitations that can also cause fingertip detection failure. In this case, participants can only rotate their fingers in a small angle range. When the target has a too large or too small angle, the manipulation task cannot be completed easily as people have to move the tablet to the camera position. To deal with these issues and improve the usability of our 3D gesture interface, a better fingertip detection algorithm and more intuitive gestures should be developed in the future.

Chapter 6

Conclusions and Future Work

In this thesis, we have investigated the potential of using markerless gesture based interaction for manipulation of virtual objects in a handheld AR application. We did this by developing a prototype gesture interface and conducting a comprehensive user study to compare it with the common touch-screen interface. In user study, we measured performance (time) and engagement (subjective user feedback) when users were conducting selection, translation, rotation, and scaling of virtual objects.

In section 6.1 we describe the lessons we learned from the research and provide some design recommendations for gesture based interaction design for handheld AR applications. We provide a conclusion of our research in section 6.2, following with a description of possible future work in section 6.3.

6.1 Lessons Learned and Design Recommendation

In our research, the gestures used are based on markerless hand and fingertip tracking algorithm; therefore, providing an accurate fingertip locating and tracking system is one major issue for implementing practical gesture based interaction interfaces. We use a skin color based segmentation algorithm to locate the hand region from the input image, which provides some useful advantages. Color detection allows fast processing without the requirement of high-powered computing and graphics processors; therefore, it also can perform well on handheld devices. However, when the camera captures a human hand in the video frame, the skin color can be easily affected by different environmental lighting conditions. Furthermore, for different devices, the displayed color is slightly different (too bright or too dark) even under the same lighting. Although we found some solutions for these issues, such as adjusting the color contrast for each input image, the detected hand region is still not as perfect as we expect. In this case, to develop a markerless hand detection system for handheld AR, researchers need to consider issues such as device computing power, device display mode, camera resolution and lighting effects.

Another issue we found in our research is the gesture defined for the interaction. We choose a pinch-like gesture with thumb and index fingers to select a virtual target. However, from the feedback of the participants, most users preferred to use one finger selection since it is more straightforward and effective. Furthermore, desktop PC using a mouse click to select or unselect one item, but in a 3D AR environment, we could not define a ‘click’ operation by using gestures; therefore, it is still a problem for us to provide an effective way to select objects or cancel manipulations for handheld AR gesture based interaction.

In the real world environment, when a person selects an object, he/she can touch the item. In other worlds, there is a physical feedback to tell the person he/she has reached the item. However, in the 3D AR environment, the user could not feel whether or not he has reached the virtual object, which may significantly increase the operational complexity. Furthermore, although we obtain the depth information for the fingertips from the depth camera, the virtual objects always appeared overlaid on the video frames and so always cover the user fingers no matter how deep they are, which could generate visual confusion. Therefore, an advanced 3D rendering system needs to be applied allow the user's hand to occlude the virtual objects that they are supposed to be in front of.

In summary, to develop a markerless gesture based interaction system for handheld AR, researchers need to consider several issues: accurate and robust hand detection and fingertip tracking, practical gestures defined for interaction and advanced 3D rendering capabilities.

6.2 Conclusion

In this thesis, we presented 3D gesture-based interaction approaches for handheld Augmented Reality on a client-server system and a tablet with an external RGB-Depth camera. Our approach was based on identifying fingerprints movements and mapping them into operations on virtual objects. The fingertip detection and tracking system uses a skin color segmentation method based on previous research. In order to use this system in a handheld device, such as a tablet or a mobile phone with limited hardware capabilities, we modified it to reduce computational complexity. For gestures, we use a pinch-like gesture with thumb and index fingers to select, translate, rotate and scale a virtual target. This is very

similar to how people grasp real objects in their daily life using two fingers.

We then conducted a comprehensive user study to evaluate our gesture based approach by comparing it with touch-screen based interaction for virtual object manipulation. This evaluation measured performance (time) and engagement (subjective user feedback). The results of this study show that the touch-based interface outperforms the gesture-based interface in terms of performance. Participants preferred to use touch screen input to interact with the virtual object and felt the touch manipulation was easier to use and more useful for completing the manipulation tasks. Users did feel that gesture input was more fun and engaging, indicating a high entertainment value. This suggests possibilities for leisure and gaming applications but limited usage for time-constrained tasks.

6.3 Future work

The scope of future work includes refining the 3D gesture interaction to overcome the limitations mentioned in this paper, and exploring scenarios in which 3D gesture interaction is preferred.

Accurate and robust hand detection and fingertip tracking system is required. Since the gesture based interaction for handheld AR is considered to be used anytime and anywhere, we have to avoid the negative effect caused by different light conditions. The skin color captured by the camera can be easily affected by the environment light; therefore, if we want to use gesture input in outdoor sunlight, we still need to work on improving our fingertip tracking system. As we discussed in section 5.3, the virtual objects just simply overlay on the video frames and always cover the user fingers no matter how deep they are in our

current system. Therefore, to improve the user experience, we need to consider an approach to cut the user's hand from the background video frame and rebuild it in the 3D AR environment. Furthermore, we also need to improve the design of gestures used in the study.

Regardless of such limitations, the research area of gesture based interaction for handheld Augmented Reality is still a challenge for researchers. We hope that our thesis could provide some help for those who wish to enter this field of research.

Reference

[1] Azuma, R. T. (1997). A survey of augmented reality. *Presence-Teleoperators and Virtual Environments, 6*(4), 355-385.

[2] Sutherland, I. E. (1965). The ultimate display. *Multimedia: From Wagner to virtual reality*.

[3] Billinghurst, M., Kato, H., & Myojin, S. (2009). Advanced interaction techniques for augmented reality applications *Virtual and Mixed Reality* (pp. 13-22): Springer.

[4] Sutherland, I. E. (1968). *A head-mounted three dimensional display.* Paper presented at the Proceedings of the December 9-11, 1968, fall joint computer conference, part I.

[5] Höllerer, T., Feiner, S., Terauchi, T., Rashid, G., & Hallaway, D. (1999). Exploring MARS: developing indoor and outdoor user interfaces to a mobile

augmented reality system. *Computers & Graphics, 23*(6), 779-785.

[6] Julier, S., Baillot, Y., Lanzagorta, M., Brown, D., & Rosenblum, L. (2001). *Bars: Battlefield augmented reality system*: DTIC Document.

[7] Piekarski, W., & Thomas, B. H. (2002). *The tinmith system: demonstrating new techniques for mobile augmented reality modelling.* Paper presented at the Australian Computer Science Communications.

[8] Wagner, D. (2007). *Handheld augmented reality.* Citeseer.

[9] Gausemeier, J., Fruend, J., Matysczok, C., Bruederlin, B., & Beier, D. (2003). *Development of a real time image based object recognition method for mobile AR-devices.* Paper presented at the Proceedings of the 2nd international conference on Computer graphics, virtual Reality, visualisation and interaction in Africa.

[10] Takacs, G., Chandrasekhar, V., Gelfand, N., Xiong, Y., Chen, W.-C., Bismpigiannis, T., et al. (2008). *Outdoors augmented reality on mobile phone using loxel-based visual feature organization.* Paper presented at the Proceedings of the 1st ACM international conference on Multimedia information retrieval.

[11] Wagner, D., Langlotz, T., & Schmalstieg, D. (2008). *Robust and unobtrusive marker tracking on mobile phones.* Paper presented at the Mixed and Augmented Reality, 2008. ISMAR 2008. 7th IEEE/ACM International Symposium on.

[12] Wagner, D., Reitmayr, G., Mulloni, A., Drummond, T., & Schmalstieg, D. (2008). *Pose tracking from natural features on mobile phones.* Paper presented at the Proceedings of the 7th IEEE/ACM International Symposium on Mixed and Augmented Reality.

[13] Pinz, A., Brandner, M., Ganster, H., Kusej, A., Lang, P., & Ribo, M. (2002). Hybrid tracking for augmented reality. *ÖGAI Journal, 21*(1), 17-24.

[14] Hürst, W., & van Wezel, C. (2013). Gesture-based interaction via finger tracking for mobile augmented reality. *Multimedia Tools and Applications, 62*(1), 233-258.

[15] Henrysson, A., Ollila, M., & Billinghurst, M. (2005). *Mobile phone based AR scene assembly.* Paper presented at the Proceedings of the 4th international conference on Mobile and ubiquitous multimedia.

[16] Chen, L.-H., Yu Jr, C., & Hsu, S.-C. (2008). *A remote Chinese chess game using mobile phone augmented reality.* Paper presented at the Proceedings of the 2008 International Conference on Advances in Computer Entertainment Technology.

[17] Langlotz, T., Mooslechner, S., Zollmann, S., Degendorfer, C., Reitmayr, G., & Schmalstieg, D. (2012). Sketching up the world: in situ authoring for mobile augmented reality. *Personal and ubiquitous computing, 16*(6), 623-630.

[18] Kim, H., Reitmayr, G., & Woo, W. (2011). *Interactive annotation on mobile phones for real and virtual space registration.* Paper presented at the Mixed

and Augmented Reality (ISMAR), 2011 10th IEEE International Symposium on.

[19] Liu, C., Diehl, J., Huot, S., & Borchers, J. (2011). Mobile Augmented Note-taking to Support Operating Physical Devices.

[20] Mossel, A., Venditti, B., & Kaufmann, H. (2013). 3DTouch and HOMER-S: Intuitive Manipulation Techniques for One-Handed Handheld Augmented Reality. *Laval Virtual VRIC'13}*.

[21] Henrysson, A., Marshall, J., & Billinghurst, M. (2007). *Experiments in 3D interaction for mobile phone AR.* Paper presented at the Proceedings of the 5th international conference on Computer graphics and interactive techniques in Australia and Southeast Asia.

[22] Hürst, W., & van Wezel, C. (2011). Multimodal interaction concepts for mobile augmented reality applications *Advances in Multimedia Modeling* (pp. 157-167): Springer.

[23] Seo, B.-K., Choi, J., Han, J.-H., Park, H., & Park, J.-I. (2008). *One-handed interaction with augmented virtual objects on mobile devices.* Paper presented at the Proceedings of The 7th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and Its Applications in Industry.

[24] Choi, J., Park, H., Park, J., & Park, J.-I. (2011). *Bare-hand-based augmented reality interface on mobile phone.* Paper presented at the Mixed and Augmented Reality (ISMAR), 2011 10th IEEE International Symposium on.

[25] Baldauf, M., Zambanini, S., Fröhlich, P., & Reichl, P. (2011). *Markerless visual fingertip detection for natural mobile device interaction.* Paper presented at the Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services.

[26] Clark, A. J. (2009). OPIRA: The Optical-Flow Perspective Invariant Registration Augmentation and Other Improvements for Natural Feature Registration.

[27] Vezhnevets, V., Sazonov, V., & Andreeva, A. (2003). *A survey on pixel-based skin color detection techniques.* Paper presented at the Proc. Graphicon.

[28] Brand, J., & Mason, J. S. (2000). *A comparative assessment of three approaches to pixel-level human skin-detection.* Paper presented at the Pattern Recognition, 2000. Proceedings. 15th International Conference on.

[29] Otsu, N. (1975). A threshold selection method from gray-level histograms. *Automatica, 11*(285-296), 23-27.

[30] Lee, T., & Hollerer, T. (2007). *Handy AR: Markerless inspection of augmented reality objects using fingertip tracking.* Paper presented at the Wearable Computers, 2007 11th IEEE International Symposium on.

[31] Ram Rajesh, J., Nagarjunan, D., Arunachalam, R., & Aarthi, R. Distance Transform Based Hand Gestures Recognition For Powerpoint Presentation Navigation. *Advanced Computing: an International Journal, 3*.

[32] Bowman, D. A., Kruijff, E., LaViola Jr, J. J., & Poupyrev, I. (2004). *3D user*

interfaces: theory and practice: Addison-Wesley Professional.

[33] Bai, H., Lee, G. A., & Billinghurst, M. (2012). *Freeze view touch and finger gesture based interaction methods for handheld augmented reality interfaces.* Paper presented at the Proceedings of the 27th Conference on Image and Vision Computing New Zealand.

[34] Wagner, D., & Schmalstieg, D. (2003). *Artoolkit on the pocketpc platform.* Paper presented at the Augmented Reality Toolkit Workshop, 2003. IEEE International.

[35] Dünser, A., Billinghurst, M., Wen, J., Lehtinen, V., & Nurminen, A. (2012). Exploring the use of handheld AR for outdoor navigation. *Computers & Graphics*.

[36] Henrysson, A., Billinghurst, M., & Ollila, M. (2005). *Virtual object manipulation using a mobile phone.* Paper presented at the Proceedings of the 2005 international conference on Augmented tele-existence.

[37] Park, J.-S., Bae, B.-J., & Jain, R. Fast Natural Feature Tracking for Mobile Augmented Reality Applications.

[38] Bradski, G. (2000). The opencv library. *Doctor Dobbs Journal*, *25*(11), 120-126.

[39] Wikipedia. OpenNI from http://en.wikipedia.org/wiki/OpenNI

[40] Woo, M., Neider, J., Davis, T., & Shreiner, D. (1999). *OpenGL programming*

guide: the official guide to learning OpenGL, version 1.2: Addison-Wesley Longman Publishing Co., Inc.