Integration of Leap Motion Controller on Mobile Devices for AR Applications

by

Dawei Huang

A Project Report Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science in Computer Science

Supervised by

Dr. Reynold Bailey

Department of Computer Science

B. Thomas Golisano College of Computing and Information Sciences Rochester Institute of Technology Rochester, New York

May 2016
Acknowledgments

I want to provide gratitude to Dr. Reynold Bailey who helped this project survive and provided guidance to me. Thank you to all participants who help this project. Thank you to Dr. Joe Geigel who inspired me about Computer Graphics. Thank you to Dr. Warren R. Carithers whose lectures are interesting.
Abstract

Integration of Leap Motion Controller on Mobile Devices for AR Applications

Dawei Huang

Supervising Professor: Dr. Reynold Bailey

Google Cardboard is a low cost of augmented reality and virtual reality platform. Interaction is accomplished using a manual lever which provides magnetic variation that can be detected by a smartphone. This paper presents a framework of AR interface to control cell-phone based on the hand gesture by integrating Leap Motion, which was designed for PC and Mac, as a 3D controller. One of big advantages of this project is to provide a affordable and interactive AR framework to manipulate smartphone.
Chapter 1

Introduction

The main goal of this project is to build a low cost framework of AR interface compared to other AR device(such as Microsoft HoloLens) to control cellphone. One of usages of this project would be help people to stay away from walking while texting since this system would provide a friendly and interactive information by adding a layer of information between human and real world.

1.1 Background

Augmented reality(AR) is a process to make augmented or supplemented on the live video or picture. [3] first defined a basic idea of augmented reality in 1901. First HUD(Head Mounted Display) device that was invented in 1968 [7]. Google published a simple head mounted device at the Google I/O 2014, called Google Cardboard, which assembles a smartphone as an augmented reality device. One advantage of Google Cardboard is that the cost to build a VR or AR applications is very low. The price at Google store is 15 dollars. And Google also published the blueprint of Google Cardboard, which means anyone could use this blueprint to customize their own AR or VR devices. Before Google Cardboard, Google had another AR device published, called Google Glass, however, this device was failed due to many reasons, such as battery, interaction way and privacy bills. Recently, Microsoft published a new augmented reality device called HoloLens in 2015[6] which is a pair of mixed reality head-mounted smartglasses with a high incredible price, 3000 dollars.
1.2 Importance

There are so many AR devices invented since AR has many applications that enhance performance in different fields, such as art, commerce and so on. For instance, augmented reality technology assists student to learn chemistry by visualizing the spatial structure of a molecule based on the picture of textbook [4]. Another famous application of the AR is to be a task supporter. For instance, when a beginner of machinist just start their career, and they don’t know how to operate the machine, AR could be a good introduction or manual supporter to help.

1.3 Limitations

The controller of Microsoft HoloLens is to use gesture input, so far, the interactive effect is unclear, since it is still a prototype, and the price, $3000, is expensive. Google Cardboard is one of most popular AR device which have been sold over 5 million [2]. The price and operability makes it popular. However, the controller is a limitation of Google Cardboard. Users could not control naturally because there is only a control button for confirmation in Google Cardboard.

1.4 Assumptions

To provide more control options and smooth the control way, this paper discusses a new controlling interface for Google Cardboard by combining leap motion as a controller. Leap motion is a small hand gesture detected device which have two monochromatic IR cameras and three infrared LEDs to collect 3D position data of hand [8]. Then using those 3D position data of hand as an input controlling option for Google Cardboard.
Chapter 2

Design

On this section, it depicts the main components of the whole system and the overall design. A full functional AR interface to cellphone includes three parts: hardware, software and the connection.

2.1 Hardware

Fig. 2.1 is a picture of all hardware components which are Google Cardboard, Android cellphone and Leap Motion.

2.1.1 Google Cardboard

Google Cardboard, which was published by Google at the Google I/O 2014, is one of main components of this project. It was designed to integrate an Android cellphone to be a low cost augmented reality(AR) and virtual reality(VR)[2]. Fig. 2.2 is an original Google Cardboard[1]. The blueprint of Google Cardboard was published, which allows companies or customers to customize their own style Google Cardboard. In this project, we are using one of those customize Google Cardboard because of comfort level. It’s possible to use other different type of Google Cardboard for this system.

2.1.2 Leap Motion

The Leap Motion is the another primary device of this project illustrated in Fig. 2.3. Leap Motion is a 3D controller designed for PC or Mac, and it was equipped with 2 IR cameras.
Figure 2.1: Hardware components
Figure 2.2: Google Cardboard
Figure 2.3: Leap Motion

and 3 IR LEDs. The main function of Leap Motion is to detect and track hand position and gestures. Another feature of Leap Motion is wide detect range of a hemispherical area with 3 feet. Leap Motion also provide a higher accuracy, for instance, it could detect the difference when two fingers are 5cm away[5]. In summary, it’s a suitable 3D controller based on hand gesture for this project.

2.1.3 Android cellphone

The Android cellphone was used as a main screen and computation source of this project. On this project, Nexus 6P was being used as a main test device. Fig. 2.4 shows the connection between Android cellphone and Leap Motion.

2.2 Software

For software development, this project choose Unity3D, a game engine, to as our main develop platform. Google has a good support of Google Cardboard to Unity develop, this project used CardboardSDKForUnity.unitypackage to develop. Another thing is that Leap
Figure 2.4: Connection between Android and Leap Motion
Motion do have a project, called project alpha, which provides a basic hardware driver to connect Leap Motion to Android cellphone. In this project, it used LeapMotionCoreAsset 2.3.2Android.unitypackage to develop.

For the program language, C# is the first option to Unity. And this project need to call the internal API of Android system, so Java also is one of program language of this project. So this project was developed by C# and Java.

2.3 Overall design

The primary goal of this project is to build a framework of AR interface, which could be used in the daily life, to help users to get more information and to avoid some daily dangers, such as texting while walking. The successor of this project could also develop and add more other useful applications, such as picture search function.

Fig. 2.5 depicts the overall architecture of this project. When the user makes some gestures, the Leap Motion is being used to track those gestures and position of hand, and transfer those raw data to cellphone. The gestures detection part of cellphone would receive those data to translate then projects the virtual hand to Google Cardboard, which is actual screen. At the same time, augmented reality applications part would be noticed to trigger specific function, which are designed to control daily function of cellphone, if the hand accomplished some specific gestures or tap some specific place. Fig. 2.6 is an intuitive view of the whole project.

For gestures detection, which is a control part, there are three type control options designed: widgets, static gestures and dynamic gestures. The option of widgets provides a few virtual control button, so users could use their virtual hand, that was projected on the screen, to click specific button. The option of static gesture allows user make a static gesture to control cellphone, such as extending a specific number of fingers. Users could make swipe, circle or other dynamic gestures to control cellphone with the option of dynamic gestures.
2.4 Milestones

There are three milestones for this project. Each milestone represents different process of the whole project.

The first milestone is to search and read related documents about the Leap Motion, Google Cardboard and how to connect them. So, the installation of SDKs, building development will be finished and connection between Leap Motion and cellphone will be finished in the first milestone.

For milestone 2, it should be able to make functional widgets and implement some basic trigger functions to manipulate the cellphone. At the end, there will be a first basic and simple version of interface to this project.

In the last milestone, three types of control option will be implemented and tested. For trigger function part, there are four kinds of daily function working at least.
Figure 2.6: An intuitive view of the project system
Chapter 3

Implementation

On this section, it’s going to talk about the implementation of the whole project by illustrating each milestone.

3.1 Milestone One

3.1.1 Connection

After checking the develop documents, develop forum, and requesting the leap motion team about connection between cellphone and Leap Motion. There is a inside test program, called project alpha, provide a basic driver to connect Android cellphone.

3.1.2 Development

The following information is the specification of the development.

Engine: Unity 5.3
Program language: C# & Java
Packages (Support Windows better):
  LeapDeveloperKit_develop_android_2.3.2_35031
  Google Cardboard for Unity

Fig. 3.1 is an intuitive view of development on PC, green parts are virtual widgets.
3.1.3 Access to outside camera

Fig. 3.2 is an example of projection of outside world by using the internal camera of Android cellphone.

```csharp
// Sample Code
void Start () {

    back = new WebCamTexture ();

    GetComponent<Renderer>().material.mainTexture = back;
    back.Play ();
}
```

3.2 Milestone two

3.2.1 Taking a picture

One of the basic functions on this project is to take a picture.

```csharp
// Sample code
```
using UnityEngine;
using System.Collections;
using LMWidgets;

// take a screenshot
Application.CaptureScreenshot(pic.ToString() + ".png");

Fig. 3.3 is a sample result.

3.2.2 A virtual functional widget: Clock

One of advantage of this project is that user could load different virtual widgets designed by the developer. For example, in this project, we are providing a sample widget: a smart virtual watch.
3.3 Milestone three

3.3.1 Creating a plugin to call internal function of Android

We have to build a plugin to call Android API directly since Unity hasn’t permissions. Unity provides some APIs to make a phone call or send a SMS message, however, those APIs will lead the user to the default application, such as SMS application, which doesn’t meet the demand of this project. In order to call internal API of Android, it has to write a plugin of Java for Unity, then uses C# to call the functions. Fig. 3.4 shows the basic working flow. The system call a function with C# code from plugin wrote by Java, then plugin would send a API request to Android with specific parameters. Android finish the requirements and return the result to plugin which will transfer to Unity.

3.3.2 Sending a SMS message

One of basic function of this system is text which is one of major functions of cellphone. The following code is a sample code to send a message. This project implemented the function, but the UI still need to be improved. The developer of this system could develop
different UI for this function, such as colorful keyboard.

```java
// Sample code: sending a SMS
// Java
public static void SMSSend(String number, String content){
    SmsManager sms = SmsManager.getDefault();
    sms.sendTextMessage(number, null, content, null, null);
}
```

### 3.3.3 Dialing a phone call

Making a phone call is one of daily functions of cellphone, so this project integrate this function as the following sample code showing.

```java
// Sample code: make a phone call
// Java
public void callSomeone(String number) {
    Intent intent = new Intent(Intent.ACTION_DIAL);
    intent.setData(Uri.parse("tel:**********"));
    startActivity(intent);
}
```

### 3.3.4 Creating static gesture model

A static gesture means that the hand gestures is static, for example, a hand gesture of V for a victory sign. The following code is an example to use the number of extended fingers to control the system. For instance, if the user extend a finger, then it will trigger specific functions.
Frame frame = controller.Frame ();
HandList hands = frame.Hands;

foreach (Hand hand in hands) {
    FingerList extendeFingerList = hand.Fingers.Extended ();

    int fingerSign = extendeFingerList.Count;
    if (fingerSign == 3) {
        menu.SetActive (true);
        menu2.SetActive (true);
        menu3.SetActive (true);
        menu4.SetActive (true);
        menu5.SetActive (true);
    } else if (fingerSign == 4) {
        menu.SetActive (false);
        menu2.SetActive (false);
        menu3.SetActive (false);
        menu4.SetActive (false);
        menu5.SetActive (false);
    }
}

3.3.5 Making dynamic gesture model

A dynamic gesture means a moving gesture, such as swipe. The following code is one of example to use dynamic gesture model of swipe. The following code describes that when user makes a swipe gesture it will active something.

[obeytabs, tabsize = 4]
// Sample code: a dynamic gesture : swipe

void Start () {
    controller = new Controller ();
    controller.EnableGesture (Gesture.GestureType.TYPESWIPE);
    controller.Config.SetFloat ("Gesture.Swipe.MinLength", 50f);
    controller.Config.SetFloat ("Gesture.Swipe.MinVelocity", 50f);
controller.Config.Save();
menu.SetActive(false);
}

// Update is called once per frame
void Update() {
    Frame frame = controller.Frame();
    GestureList gestures = frame.Gestures();

    for (int i = 0; i < gestures.Count; i++) {
        Gesture gesture = gestures[i];
        if (gesture.Type == Gesture.GestureType.TYPESWIPE) {
            SwipeGesture Swipe = new SwipeGesture(gesture);
            Vector swipeDirection = Swipe.Direction;
            if (swipeDirection.z < 0) {
                menu.SetActive(true);
            } else {
                menu.SetActive(false);
            }
        }
    }
}
Chapter 4

Analysis

This section demonstrates the result and depicts the evaluation of the project. This evaluation has been done by two parts. First is a comparison of frame per second (fps). Another part is to invite tester to test, then give a score.

4.1 Results

Fig. 4.1 is an example of using virtual widgets model. There are several buttons near by arm, users could click those buttons to control Android cellphone. In the image, the user tries to active the virtual clock. A big issue of this model is that Users have to learn appropriate speed and hand positioning in order to click the button. Another problem is that poor tracking may result in the need for repeating the gesture.

Fig. 4.2 is an example of using static gesture model. In this model, the user could extend specific number of fingers to control. The image shows that a user extend two fingers to send a SMS message. This model has a good response speed.

Fig. 4.3 is a sample of operating dynamic gesture model. This model allows users to make a different dynamic gesture to control the cellphone. For instance, the picture shows that a user tries to make a gesture of swipe to surface the control button. User has to learn appropriated speed. Also, the poor tracking is running their experiments.
Figure 4.1: Example: Virtual widgets model
Figure 4.2: Example: Static gesture model
Figure 4.3: Example: Dynamic gesture model
4.2 FPS Comparison

This project compares different control models. Leap Motion was designed for PC and Mac, so the performance of Leap Motion on PC is considered for the comparison. Fig. 4.4 presents the result. According to the picture, PC has the highest performance of fps. Usually, the minimum fps people wouldn’t feel lag is higher than 60 fps at least. The maximum fps of three control models is the same, 40 fps. However, their minimum fps are significantly different. The dynamic gesture has the lowest fps, which means there is a latency issue. The minimum fps of the static gesture model is the highest model among the three control models, which means that static gesture model may have a better performance compared to other two control models.
4.3 Preference rating

The preference rating is an experiment to analysis the user's preference after they tried three control model. So far, we only six participants to test. Fig. 4.5 is a result. As the table showing, testers prefer the static model. The reason may be the immediate response of static gesture model. Dynamic gesture model has a low score because of the latency issue and inaccuracy.

<table>
<thead>
<tr>
<th>Type</th>
<th>Widgets</th>
<th>Static</th>
<th>Dynamic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Score</td>
<td>3.5</td>
<td>4</td>
<td>2.5</td>
</tr>
</tbody>
</table>

Figure 4.5: Average preference rating
Chapter 5

Conclusions

5.1 Current Status

This project provides a solution for incorporating hand gesture controller into Google Cardboard. The static gestures version has the best performance and does not suffer from tracking issues. The virtual widget version is more visual to manipulation. The dynamic gestures version is fancy but needing more computation and ram. So far, this system is working but not desirable. With the developments of increasing computation and ram of smartphones, this system would be first choice of AR products instead of other devices, such as expensive Microsoft HoloLens.

5.2 Future Work

The future work stresses on improving the response speed. One idea is to use a newer cellphone with higher computation and bigger ram to test this system. Another direction is to customize the Android OS system to reduce any background applications that this project dont need in order to increase the response speed.

Another fair suggestion for this project is to connect the ROS robotic system, and use this system to control the robot.
5.3 Lessons Learned

At the beginning of this project, I hadn’t any experiences of developing Leap Motion application, C# language and Unity development. So I have to learn all of three techniques, which are the most significant part of the whole project, as a rookie. With the passion of the whole project and the help of advisor and the tutorials, I was able to complete this project.

An efficient frame augmented reality and virtual reality for mobile device is important and necessary. In the future, the interaction between human and cellphone would change. People don’t need to hold cellphone all time because the cellphone would attach on the head. And it would be more convenient and efficient by inserting a layer of useful information between human eye and real world.
Bibliography


