# UnityBox: Using Natural Gestures for Direct Manipulation in Virtual Reality

**Sanchayan Sarkar**
Department of Computer Science
University of Pittsburgh
Pittsburgh, USA
sanchveda@cs.pitt.edu

**Akhil Yendluri**
Department of Computer Science
University of Pittsburgh
Pittsburgh, USA
akhily@cs.pitt.edu

**ABSTRACT**
*At present, interactions with Virtual Reality are mostly done with the use of buttons and controllers. While these are effective, they do not capture natural movements. On the other hand, sensor data have been used for gameplay in mobile devices but not in Virtual reality environment. In this work, we present UnityBox as an application that effectively integrates natural hand Gestures to movement of a cube in the Virtual reality environment. Also, we have demonstrated a test condition that can evaluate algorithms in VR that deals with pointing movements. In the process, we have obtained the time taken for a cube to hit a target sphere of varying size. Our quantitative analysis shows that smaller the size of this target, more time it takes to hit it thereby indicating the difficulty of directness. Our corresponding qualitative study indicates the user perception of difficulty level. Finally, we have concluded from our user study that even though the movement feels direct, it is not completely natural.*

**Author Keywords**
Virtual Reality, Sensor Data, Direct Manipulation

## 1. INTRODUCTION

Virtual Reality (VR) [1] has evolved into a technology which can emulate an user's physical presence in the a virtual environment. It has multiple applications in the field of medicine, gaming and movie making. Over the course of time, games like House of the Dying Sun, Minecraft, Robo Recall, etc. have been developed in Virtual reality Environment. However, all of the existing methods have used specially designed input controllers and buttons. An example is shown in Figure 1. While these are highly effective in nature, they fail to capture natural movements that follows human intuition like objects moving in sync with hand movements, waving of hands, etc.

On the other hand, sensor data from mobile phones and wearable devices have been heavily used as an input to control objects in games. Sensor information like the tilt of a mobile phone or touch are popular choices in tackling video games [2]. However, applications taking in sensor data does not have a Virtual Reality User Interface. As a result, there are no techniques that effectively uses natural hand gestures to interact with Virtual Reality and therefore there is a lack of direct interaction.

**Figure 1 Virtual Reality with Controller**

To generate such directness, our objective was to reduce the gulf of execution by creating more natural translations of our hand movement [3]. For this, we have created an application, *UnityBox*, which is a basic game in virtual reality environment. The objective of the game is to move a cube to hit a sphere in the VR. The cube is considered as the player object and will be controlled by the Smartphone held at hand. The system takes in the sensor information from the smartphone and transforms it to map movements in the VR space. The sphere in the game is used as the target whose size correspond to difficulty level. We have calculated the movement time taken by the box to hit the sphere. We have modelled our experiment as that of a pointing task; similar to the pointing task experiment mentioned in Card et al [4]. The main motivation for this was to understand the nature of natural movements in a VR space. Further, we have conducted a qualitative user study to get the user perception of difficulty of the ease of movement. Our main motivation to do this was to understand how direct our application feels.

The following are our main contributions in this work:

- Successfully integrating continuous sensor data from the smartphone to control the movement of an object in Virtual Reality
- Using natural hand gestures to move player objects in the game.
- Propose a test condition that can be used as an evaluation benchmark for other methods that map sensor data to movement in virtual reality.

The rest of the paper is structured as follows. Section *2 gives a brief explanation of the related work. Section 3 shows our methodology for using hand gestures to control VR movement. Section 4 shows our experimental setup and the user study. Section 5 discusses the results and Section 6 concludes our work.*

## 2. RELATED WORK

Although there has been no existing work that uses android sensor data to interact with virtual reality game objects, there are two major inspirations to our work. First, on the sensor information side, the work done in 'Wearabird' [5] is of interest because it has used linear acceleration data in making the flappy bird move. Second, on the virtual reality side, the work done in 'Exo: A daydream Experience' [6] is important because it creates a VR environment in Unity platform that uses a controller to control a Quadcopter in VR space. We were motivated to use linear acceleration data to control a cube in the VR space. Finally, the modelling of our experiment is inspired from 'Pointing to Pondering' done by Card et. al [4]. We have presented our task condition similar to that of a pointing task but in the Virtual Reality space where the sphere is the target body.

## 3. METHODOLOGY

We have three components in our application: An input android side to take in the sensor information, an output VR side to present our gameplay and a interconnecting layer sets up the low level connection between the two platforms. The components are delineated in the next section. In this section, we focus on handling the data and the transform functions only.

**Input Side**

We have used the Accelerometer data from the Android phone as our input. However, there two major challenges to this: i) Change of axis of the mobile phone can severely affect the accelerometer data in the coordinate system and ii) the raw accelerometer data is coupled with the gravitational force of the earth. These two challenges are extremely detrimental to our problem since this gives data which is against our intuitive perception of the motion. Figure 2 gives a view of the change of accelerometer data with respect to the orientation.

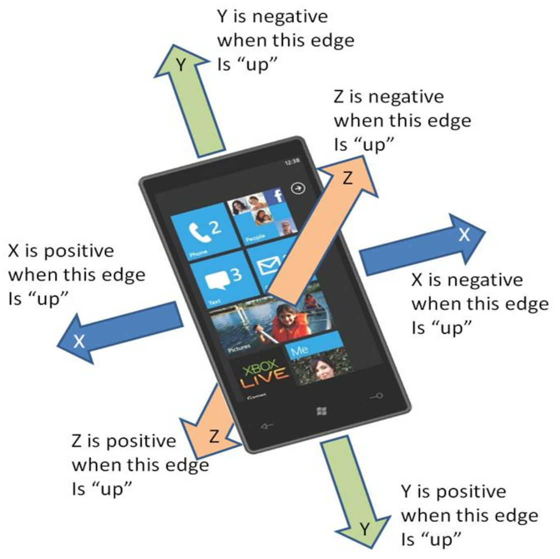
Y is negative when this edge Is "up"
Y
Z is negative when this edge Is "up"
Z
X is positive when this edge Is "up"
X
X
X is negative when this edge Is "up"
Z is positive when this edge Is "up"
Z
Y
Y is positive when this edge Is "up"

**Figure 2 Accelerometer Data with respect to Orientation of the Mobile phone**

To abet these challenges, we have used a function to decouple the force of gravity from the accelerometer data. Let G( x, y, z) be the gravitational force, accl ( x, y, z) be the accelerometer data along x, y and z directions respectively. Using equations (1) and (2), we obtain the Linear acceleration, La(x, y, z) along the x, y and z directions where α is the tuning factor.

$$G(x,y,z)=\alpha * G(x,y,z)-(1-\alpha) * accl(x,y,z) \quad -(1)$$

$$La(x,y,z)=accl(x,y,z) - G(x,y,z) \quad -(2)$$

The obtained linear acceleration will be only positive if the movement of the phone is towards the right, up or forward directions. It would be negative if the movement is towards the left, down or downwards direction. As a result, our phone movements would produce correct data corresponding to our hand movement.

**Output Side**

The VR side of the problem takes in the Linear acceleration data as input. However, there are three major challenges:

1. The Linear acceleration data is extremely noisy and as a result even for static positions, it generates linear acceleration of smaller magnitude. To overcome this, we have used the following thresholding function (equation 3).

   $La''(x, y, z) = La(x, y, z) \text{ iff } |La(x, y, z|) \geq \beta$ - (3)

   Here, β is the thresholding limit which we have set to 2 since the noisy data have smaller magnitude. The value 2 is set after tuning.

2. Since our data is Linear acceleration, it crosses the threshold only during the onset of motion and during the apex of the motion. Since the apex of the motion creates a negative acceleration, this effectively negates the positive movement. We have also observed that the direction of the negative acceleration for the positive movement is in opposite to that of the movement. However, the direction of the negative acceleration for the negative movement is in the same direction to that of the movement. To overcome this, we have created an adjustment function that accounts for this phenomenon. This is given in Equation 4.

   $M(x, y, z)\ (+ve) = 2 * M(x, y, z)\ (-ve)$ -(4)

   Here, *M(x, y, z)* is the movement of the cube in the VR space and is measured in Unity units.

3. Since our movement of the smartphone is in natural space, using those values in the virtual space would be too large. For this, we have created a scaling function based on our empirical observation of the movement correspondences from the real space to the VR space. Equation 5 gives the scaling function:

   $M_{VR}(x, y, z) = (1/9) * M_{RR}(x, y, z)$ - (5)

   Here, $M_{VR}$ is the movement of the cube in the virtual space while $M_{RR}$ is the movement of the hand in real space. The value of 9 is tuned after running the experiment several times.

After processing the input data through these functions, we are able to get the movements in the appropriate directions which aligns with our perception of the movement in the virtual space. For example, moving our hand upwards would effectively make the box move in the y-direction in VR space but with a downscaled speed. Figure 3 summarizes all the stages of Data Processing.

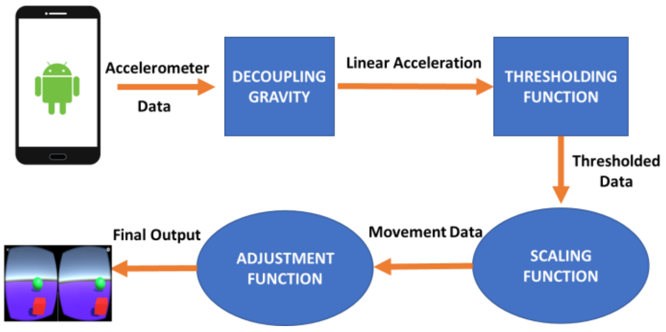

**Figure 3 Data Processing Stages for UnityBox**

## 4. EXPERIMENT AND USER EVALUATION

### 4.1 PLATFORMS USED

**Android Studio**

We used Android Studio to build the interaction between the sensors in the smartphone and the 3D environment. The android app was created as a library which was imported into Unity3D. This app mainly runs in the background frequently acquiring data from the sensors and sending them over to Unity. To do this we also had to build the library as a package for the Unity engine. This is done to ensure the cross-platform portability as Android Studio mainly uses the Java SDK while Unity3D runs on a C# environment.

**Unity3D**

Unity is a game engine used for developing games for both PC and mobile. It can also be used for developing Virtual Reality environment. For our project, we used Unity 3D to develop the entire Virtual Reality environment. To do this, we used the Google VR SDK for Unity. This helped in simplifying the development process for Virtual Reality. As mentioned before, the android plugin helped unity to communicate with the device and receive the sensor data. The sensor information was sent over to Unity as a string which had to be parsed to get the required values. All data received are real time which changes as per the user's relative movement. This information is used to control the relative movement of the box in the virtual environment. Figure 4 shows a screenshot from the actual environment we have created with the cube and the sphere in VR.

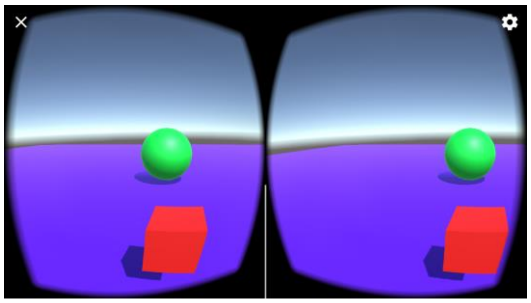

**Figure 4 Virtual Reality in Unity3D**

## 4.2 STUDY DESIGN

As mentioned in Section 2, we have modelled our experiment based upon 'pointing' task as mentioned by Card. Et al [4]. Since there is no existing work on this nature of experiment for Virtual Reality, we have developed our own control conditions and we hope it can be extended for any other method that maps natural gestures movement to movement of player objects in Virtual Reality.

We have designed a 1 factor 3 level within-subjects design with the following variables:

1. Our Independent variable is the size of the Sphere. The levels are low, medium and high The size of the sphere represents the difficulty of the pointing task. So the level 'low' represents the largest size of the sphere while the level 'high' represents the smallest size of the sphere.
2. The dependent variable is the time taken for the cube to hit the sphere.
3. The control variables is the Hardware used (Google Nexus 5), Software used (Google VR SDK, Unity 3D) and the Distance between the box and the sphere.

## 4.3 USER STUDY

For our user study we recruited 6 participants (2 females 4 males) to use our UnityBox application. Each of the participants had to play all the three conditions. Before getting our formal data, every participant was given 3 minutes of practice time to interact with our UnityBox application with difficulty level set to high. After this, they had to play each of the conditions in a randomized order. The order was randomized in order to negate any kind of carry over effects in the within subjects design.

**Table 1 Time Taken for the Box to Hit the Sphere**

| Users | Low | Medium | High |
|---|---|---|---|
| **1** | 17.9 | 28.6 | 43.6 |
| **2** | 14.6 | 23.3 | 36.5 |
| **3** | 18.8 | 33.8 | 49.8 |
| **4** | 19.0 | 35.7 | 51.3 |
| **5** | 16.5 | 33.1 | 45.7 |
| **6** | 15.8 | 32.4 | 47.8 |

The users were also asked to rank on a scale of 1 to 3 on the difficulty of movement for moving the cube in the VR environment with 1 being Hard (most difficult) and 3 being Easy (least difficult). This was essentially done to get the subjective evaluation of the task. Beside this, each of the user were asked to give a feedback on how natural the movements felt.

In order to determine the effect of size of the target (sphere) on the time taken for the box to reach the sphere, we conducted a One-Way Repeated Measures ANNOVA [7] test with the distance between the cube and the sphere set to constant.

## 5.RESULTS AND DISCUSSION

Table 1 gives the time taken for the box to reach the sphere. The time is measured in seconds. From the table, we can clearly see that the time taken for the box to hit the smaller sphere is much higher than that of the larger sphere for every participant. An understanding can be seen from Figure 5. None of the lines intersect each other and clearly the gray line is much greater than the blue line indicating that the size has an impact on the time for moving the cube.

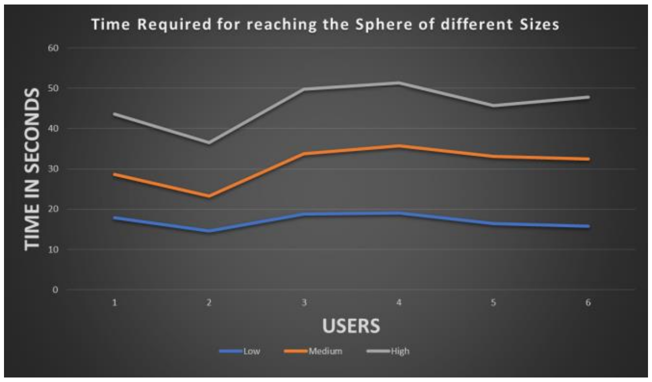

Figure 5 Time Taken for the Different Sizes of the Sphere

From our One-Way Repeated Measures ANNOVA test, our we can reject the null hypothesis($p=0.002<0.05$). So, we can say that our result is statistically significant. The degrees of freedom for our design is 2. Also, the $F_2$ value for the sphere is 238.711 which shows that there is a significant effect of the size of the target sphere on the movement time.

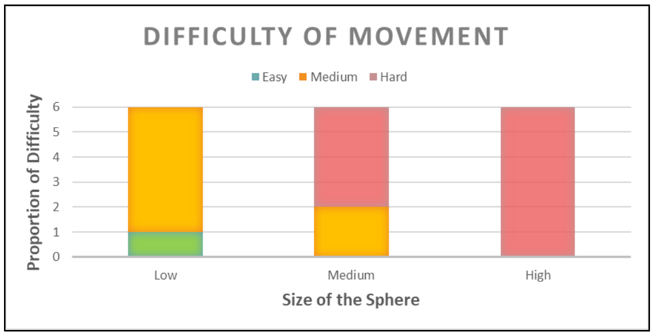

Figure 6 User Opinion on the Difficulty of Movement

On the qualitative side, we can see from Figure 6 that for the largest size of the sphere, most users have rated the game as "Medium" difficulty while for the "Medium" and "High" sphere sizes, almost all users have rated the game being as of "Hard" difficulty. This corresponds to our quantitative analysis where most people who found the "High" difficulty condition (smallest sphere size) hardest also took the most amount of time in completing the task. However, it is interesting to note that even when the difficulty level is "Low", most users perceive the movements as of "Medium" difficulty.

One reason for this phenomenon can be explained by looking at the Impulse Variability model [8] for pointing tasks where the initial movement towards the object is very fast and then as it approaches the target, there is a closed-feedback control to adjust for the target size. If the size is smaller, then the closed-feedback control is larger. A similar analogy can be drawn even in the case of VR environment as well. Empirically we can observe that while the users could reach close to the sphere quickly, they had to do more movements when it comes to actually touch the sphere. And as the size of the sphere goes smaller, the overall time taken is more thereby increasing the perception of the task difficulty.

From the user's feedback, we have observed comments like "*the motion seems jerky and not continuous*" and "*sometimes I don't feel like the movement is what I intended*". These indicate that although we have been able to bring some directness to our system, it is not completely natural yet. The main reason behind this is the instability of the sensor data and the unwanted motion it produces if it is not properly processed. We believe an exhaustive processing of the input data is the key to solve this problem.

## 6.CONCLUSION AND FUTURE WORK

In this work, we have shown how we have successfully integrated continuous sensor information as input to obtain natural hand gesture movement in directly manipulating a box in Virtual Reality environment; something which has not been done before. We have also presented a novel test condition by modelling movements in VR as a pointing task where the size of the target is an important factor in determining the difficulty of the task. We have concluded from our subjective study and user feedback that even though the movements are in correspondence to that of the movement of the hand, it is still not perfectly natural. In the future, we intend to introduce better input data features like the orientation, linear velocity of the smartphone or android wear and use sensor fusion to map movements in the Virtual Reality. Also, the current application can be extended with more interactive features with the game objects to enhance the VR experience. We believe that direct manipulation in Virtual Reality will be a major focus in the years to come.

## 7.ACKNOWLEDGEMENT

This research project was a part of the CS2610 coursework and was supported by the Department of Computer Science, University of Pittsburgh. We thank Dr. Jingtao Wang for providing us with the resources to carry out the project. We would also like to thank our colleagues from our department who volunteered as test subjects for our experimental evaluation.

## REFERENCES

1. Steuer, J., 1992. Defining virtual reality: Dimensions determining telepresence. *Journal of communication*, *42*(4), pp.73-93.
2. Lane, N.D., Miluzzo, E., Lu, H., Peebles, D., Choudhury, T. and Campbell, A.T., 2010. A survey of mobile phone sensing. *IEEE Communications magazine*, *48*(9).
3. Hutchins, E.L., Hollan, J.D. and Norman, D.A., 1985. Direct manipulation interfaces. *Human–Computer Interaction*, *1*(4), pp.311-338.
4. Card, S.K. and Moran, T.P., 1988, January. User technology: From pointing to pondering. In *A history of personal workstations* (pp. 489-526). ACM.
5. https://www.youtube.com/watch?v=pkC2iDaLuEI
6. https://www.youtube.com/watch?v=CJxUSDS7dqQ
7. von Ende, C.N., 2001. Repeated-measures analysis. *Design and analysis of ecological experiments. Oxford University Press, Oxford*, pp.134-157.
8. Balakrishnan, R., 2004. “Beating” Fitts’ law: virtual enhancements for pointing facilitation . *International Journal of Human-Computer Studies*, *61*(6), pp.857-874.