Low Cost Magnetic Field-Based VR Gesture Tracking by Machine Learning

Levi Oliver and Kai Zhang

Abstract—Low cost and accurate gesture tracking is a very important aspect of interactive virtual reality (VR) applications. However, current optical gesture tracking methods usually require extra equipment and relatively expensive depth camera. Taking advantage of low cost magnets and magnetometer on VR headset we used machine learning algorithms to define several gestures and build an interactive VR game.

Index Terms—VR gesture tracking, magnetic field, machine learning

1 Introduction

While VR has obtained increasing popularity in the field of gaming, education and medical training in the recent years. Similar to most consumer electronics devices, VR also requires an intuitive user control scheme. To this end, multiple optical tracking systems have been developed to track user gestures. However, these optical tracking systems usually require a depth IR camera which can be relatively expensive and mitigate room light. To solve this problem, we developed a gesture tracking technique by measuring the magnetic field. With the user holding a magnet in hand, we process measurement data from the magnetometer to determine the hand's relative position with a microcontroller so our device can understand the user's gesture.

Hard-coded methods to process magnetic field data have been developed by qualitatively observing measurements of magnetometer. However, those methods are limited by number of gestures they can properly recognize due to complexity of magnetic field distribution in the 3D space. By using machine learning support vector machine algorithms, we solved the gesture tracking problem using a multi-class classification approach. Based on gesture tracking results we built an interactive VR tank game.

2 Magnetometer Measurements

An inertial Measurement unit (IMU) (InvenSense MPU-9250) is connected to a microcontroller (Arduino Teensy 3.2) to provide a 3 DOF of magnetic field readings for our gesture tracking process. As Figure 1a depicts, IMU and microcontroller are mounted on a VRduino board. Magnetic field measurement data is sent from the IMU to the microcontroller over I2C pins. We process raw measurements data of measured magnetic field $H_x, H_y and H_z$ on microcontroller by machine learning algorithms to predict hand position and further sent to a laptop by serial communication port.

In our VR game, some magnets, as Figure 1b shows, will be held in the user's hand. Relative to the earth's magnetic field, the magnetic field of these neodymium magnets are sufficiently strong that they can be used as control input without serious interference.

- *The authors are with Department of Electrical Engineering, Stanford University, 440 Escondido Mall, Stanford, CA 94305, USA. E-mail: lpoliver@stanford.edu, kzhang3@stanford.edu.*

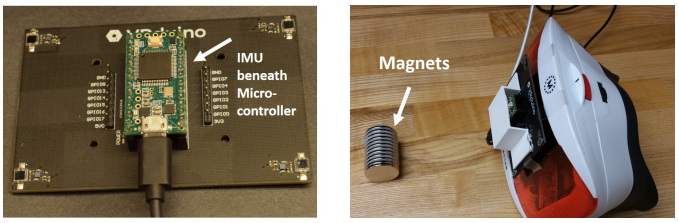

Fig. 1. (A) Teensy 3.2 microcontroller and MPU 9250 IMU for magnetic field sensing (B) Magnets for user to hold and ViewMaster head mount display for virtual environment rendering

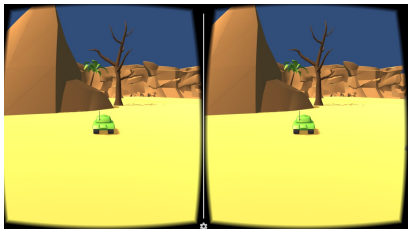

Fig. 2. Tank game virtual environment in Unity

3 Virtual Environment Set Up

We build a VR tank game by Unity engine (Figure 2). Tank, as an Unity game object, can be controlled by a *C#* file which we modified from an online source. A collider is created around the tank so that it will behave as a physical object when it hits any object in the VR environment. We also defined a shell prefab to enable the tank to fire with another *C#* file to define the explosion effect of the shell. We also added a particle system to the shell to mimic dust effect of the shell explosion on a virtual object in the environment. Most of the prefab and *C#* codes we use here are downloaded and modified from Unity asset store. Audio files for background music, shell firing and detonation, and engine idling and movement are all included.

4 Gesture Design

We define six gesture for user to interact with virtual environment as Figure 3 shows. To make gestures very intuitive for users, moving magnets up and down relative to the IMU leads to acceleration and deceleration, respectively. Moving magnets left and

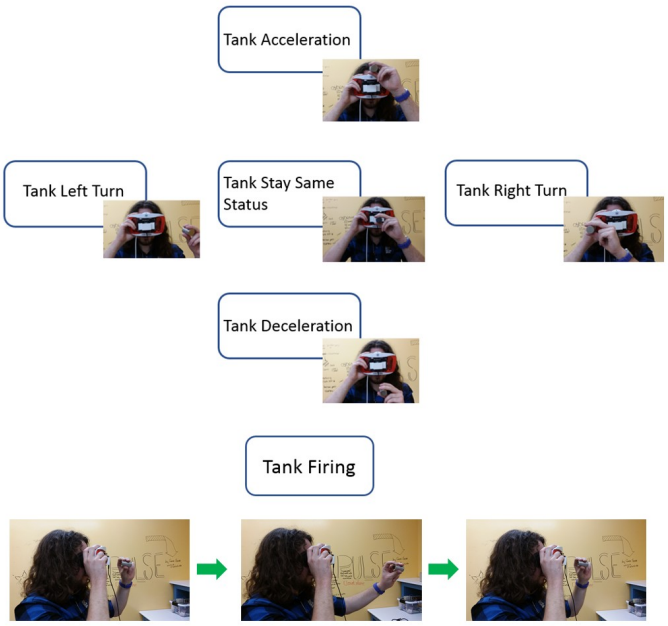

Fig. 3. Gestures designed to move tank around and fire

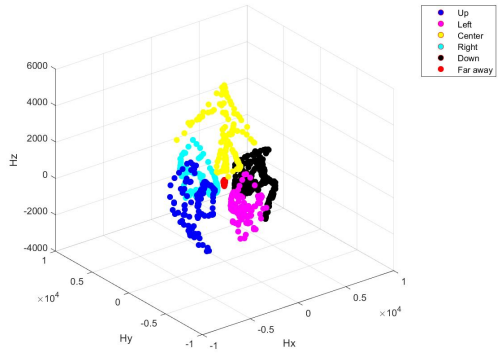

Fig. 4. Magnetic field data by holding magnet around magnetometer.

right indicates to turning left and turning right. Holding magnets at the center indicates no active command. Quickly moving the magnets away from the IMU and back will cause the tank to fire a shell.

5 Magnetic Field Data Processing by Machine Learning

One of the most challenging aspects of this project was determining the correlation between magnetometer measurements and the position of the magnet relative to the IMU. We tried to observe some qualitative rules to differentiate gestures of the user by magnitude, sign, and derivatives of each H_x, H_y and H_z and hard-code these values. This method works if we only defines two gestures. For more gestures, it's not straightforward to figure out those rules. For example, the magnitude of the measurements would not fall off quadratically depending on the orientation. After plotting out our magnetic field data as Figure 4 shows, it became clear to us that if we held the magnet still at a position, the magnetic data readings will be a cluster. Holding the magnet at different spatial regions leads to distinct clusters with relatively clear boundaries. This behavior enabled us to detect the user's gesture in a simple but effective way.

We defined magnetic field recognition problem as a multiclass classification machine learning problem. Collecting data was straightforward. We held magnets at specific spatial regions around the magnetometer, and associated magnetometer measurements with the regions they were measured in. This defined our training data set. We then used a python machine learning library [1] to train magnetic field data we labeled. The Support vector Machine (SVM) model yielded high accuracy and turned out successful. (Although convolution neural networks and k-means unsupervised learning investigated, but ultimately ruled out. CNNs were unnecessarily complex and computationally intensive, which could yield high latencies. K-means clustering was attempted, but inaccurate.) Among SVM models, we tested linear support vector classification (SVC) and nonlinear SVM models. While nonlinear svm models sometimes become unstable solving the optimization problem, linear SVC gave us consistently good results in training error and test error because the magnetic field data in different regions have relatively clear boundaries. Thus, we used a training data set of approximately 100 samples measured over 15 seconds per region to train our model. A training error rate of 1.01% is achieved. The test data yielded an error rate of 0.95%. Upon further investigation, it seemed that the mispredictions occurred only at the boundaries between two regions. Considering the relatively small amount of samples we used to train our model, our test and training error rate were satisfactory. To predict the region where the user's hand was, we took the coefficients and offsets yielded by the model and implemented the simple classifier on the Arduino. While we considered using the python library directly while running, this would add a great deal of latency and complexity for little forseeable benefit.

Even more gestures up to a functional distance of an arm length away from magnetometer are possible with a more careful tuning of machine learning model. We observed that the same values may be yielded by the magnetometer if the magnet was in different positions and orientations. We avoided this problem by selecting regions that do not have such overlap, but we believe this problem could also be solved using at least two magnetometers.

6 Conclusion

We built an interactive VR game based on magnetic field detection. By holding a magnet in hand, the user's gestures are tracked by magnetic field measurements in the magnetometer. SVM machine learning algorithms are used to do a multiclass classification problem so that user's gestures are recognized. A virtual environment is setup in Unity to translate user's hand gestures into tank's movements and firing.

7 Acknowledgements

We appreciate Prof. Gordon Wetzstein's and other instructors' advise during our project.

1. http://scikit-learn.org/stable/index.html