VirtuWheel

James Smith, Raquel Izquierdo, Caitlin Kwan, Greg Giebel
james.smith,raquel_izquierdo,cykwan,ggiebel@berkeley.edu

ABSTRACT

A major problem with virtual reality is the lack of feedback the user receives during interactions with virtual objects. One solution to this is to use physical props to provide haptic feedback, but this still lacks visual feedback cues that may be helpful or necessary for some tasks. This can negatively affect the user's sense of embodiment in the virtual space.

Our project, VirtuWheel, aims to provide visual feedback during virtual object interactions by using touch sensors on the surface a steering wheel controller which we can use to visualize interactions with the steering wheel in VR.

1 INTRODUCTION

Virtual reality (VR) is a promising new field that presents users the opportunity to experience something that they are unable to experience in person. However, VR has a serious problem with *embodiment*, which is the user's ability to feel present in and able to interact with the virtual environment. To improve user embodiment, researchers have begun developing ways to provide physical haptic responses to virtual interactions. One solution to this is a set of haptic gloves that use mechanical brakes to stop the user's hand when it touches virtual objects [3]. Another approach is to augment VR controllers to introduce constraints on the motion relative between them so they can represent objects in a variety of configurations [5]. These represent *active haptic* solutions; they use some generalized mechanism to stop the user's hand from moving by providing a normal force. Other researchers have begun looking at *passive haptic* solutions, which are objects in the physical environment that represent virtual objects. This solution uses the existing normal forces afforded by the physical object to allow users to naturally reach out and touch them. Research has even been published recently on ways of using a single physical prop to represent multiple virtual objects [1].

Because our project involves interactions with a car steering wheel, we have a natural passive haptic object that we can use. Unfortunately, using the wheel remains imperfect as the user lacks a visual representation of their hands in the virtual world.

Most solutions that currently exist involve expensive and glitchy vision systems that attempt to track user's hands with cameras and then render them in VR. Some recent research in this area include using a Kinect depth camera to create a 3d reconstruction of the user's arms [2]. Work has also gone into how to best represent the user's hands in a virtual environment, and have found that often a more realistic representation can lower the user's sense of immersion due to being more sensitive to errors [4].

For this project, we attempted to solve this problem in a cheaper and more reliable manner by using touch sensors on the wheel to visualize the user's interactions with the wheel.

EECS 149/249A, December 14, 2018, UC Berkeley
2018. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

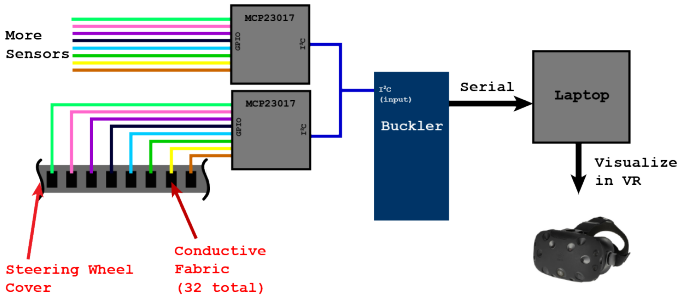

Figure 1: VirtuWheel system architecture diagram.

In order to accomplish this goal, multiple topics from the course were utilized. *Sensors* were necessary in order to determine where someone touched the wheel. *Input and Output* were required to allow the NRF board to communicate with the sensor circuits, as well as the laptop running the VR environment. As the NRF board lacked enough ports for communication with all of the sensors, GPIO expanders were used. The NRF board was able to communicate with these chips using memory-mapped I/O, a topic covered in the *Memory Architecture* portion of the course.

A video for our project can be viewed at https://www.youtube.com/watch?v=wzfDXCfKiVs.

2 DESIGN AND IMPLEMENTATION

The VirtuWheel system is an accessory for a VR driving system. It is made up of 32 2x1cm rectangles of conductive fabric attached with conductive epoxy to a standard steering wheel cover. Each of these fabric squares is attached by wire to an RC circuit connected to one of three MCP23017 I2C GPIO expanders. These expanders communicate with a NRF52832 board using the I2C communication protocol. The NRF board uses two GPIO pins as the SDA and SCL pins for I2C communication, enabling two GPIO ports on the NRF to communicate with all 32 sensors.

2.1 Capacitive Touch Sensing

Underneath each capacitive touch fabric sensor, electrically connected via the epoxy, is a wire leading to one of the 32 GPIO pins on the three MCP23017 GPIO expanders. Every GPIO pin is connected to an RC circuit (Figure 2). Each GPIO pin is connected via an 8.2MΩ resistor to V_{dd}. Each GPIO pin has an internal capacitance to ground. The GPIO pins are regularly charged and discharged through software, by periodically setting each pin low and swapping to input. While the sensors are not being touched, they build charge at a constant rate which is measured in software as the baseline charge rate. When the sensor is touched, the human body creates another path to ground, and acts as a capacitor in parallel to the touch sensor. This increases the total capacitance of the circuit, increasing the amount of time it takes for the touched touch sensor

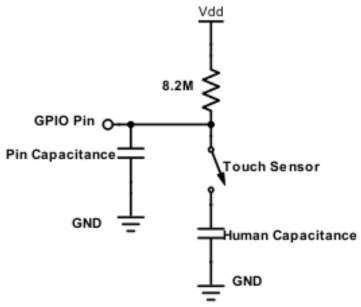

Figure 2: Capacitive touch RC circuit.

to charge. By measuring the time it takes for each sensor to charge and comparing to a threshold value based on the baseline charge rate, software can detect which sensors are being touched.

2.2 NRF Software and Communication

In order to communicate with 32 distinct sensor circuits, the NRF board is connected to three MCP23017 GPIO Expanders. These expanders communicate via the I2C protocol to GPIO pins 28 and 29 on the NRF board. In implementing the driver for this communication, the NRF two wire interface (TWI) was used. The TWI is a protocol very similar to I2C, with most I2C devices being TWI compatible as well.

The I2C driver for this project uses the NRF's built in TWI Manager library to control the two GPIO pins. This library allows specification of a clock rate,, pins to use as SDA and SCL, and the creation of read and write transfers. By following an Arduino tutorial for communicating with the MCP23017 and looking over the data sheet, we created functions to set the mode of pins to input or output, as well as reading and writing from those pins. Since I2C has a slave select function, only one set of SDA and SCL lines was needed to communicate with all three chips because each chip was given a different slave address. To send commands to the MCP chip, TWI transfers were created to write specific values to memory-mapped I/O addresses on the chip. The base address for each chip was determined by its slave number. Two bytes are sent in each transaction, the first being a register address related to the command being sent and the second a value associated with the command, if needed. Each chip had two "sides", or banks, of 8 GPIO ports. To write to pins a message consisting of the register address associated with the side was sent (0x12 for the first side and 0x13 for the second) followed by a byte representing which pins should be high and which should be low. Reading was done similarly, but with the second byte being sent by the slave back to the master.

Serial communication was necessary for the NRF board to send data back to the laptop. We utilized the NRF52832 serial library to send data in a JSON format so that it could be easily deserialized on the laptop. We initialized the library to send data as fast as possible, 115200 bps, so that there was minimal delay in the transmission. We send 32 integers, each representing the charge time for a single capacitive touch circuit.

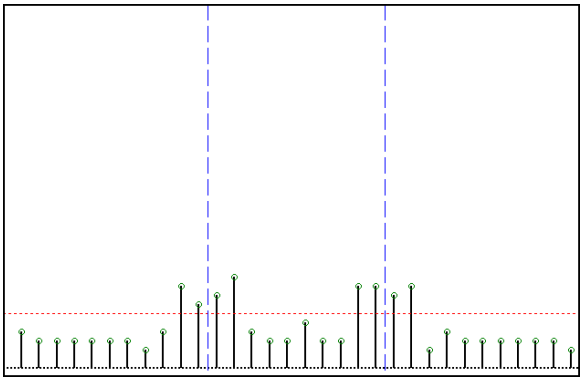

Figure 3: Plot of circuit charge times. The horizontal red line is the maximum threshold at which we detect touch events. The vertical blue lines are the centers of clusters of touch points.

2.3 Touch Detection

After receiving the charge time data over serial, the software on the laptop can determine which sensors are being touched by comparing each to its own threshold value. Thresholds are determined for each sensor individually because they all have different discharge times. We believe this is due to some of them having longer wires to their piece of conductive fabric. To determine these thresholds, we observe the sensors when they are untouched. We determine a max and average value over time, which we refer to as the sensor's baseline reading, and set their touch thresholds slightly higher than their baselines. An example of this is shown in Figure 3.

2.4 Virtual Reality Environment

Once touch points are detected, they are rendered in the virtual environment. We set up the VR environment using Unity. A 3d model of the steering wheel was created using Blender. Each sensor has a known, fixed location on the steering wheel, so if we know which sensor is touched, we can render a visualization at that point on the wheel. The first sensor is at the bottom of the wheel, 20 degrees to the right. We enumerate all of the touched points, rotating around the wheel as necessary.

We considered 2 designs for our visualization. Originally we were planning on rendering a hand at the center of a cluster of touch points (see Figure 3). We determined that there were many edge cases in which this time of visualization would fail. How would we render a single sensor being touched? We don't know which finger is touching the sensor, so we wouldn't know how to position the hand. Also, we didn't want to make an assumption that every user of our system was equally able bodied. We decided on a more abstract visualization because we felt it would still provide the visual feedback mechanism we were looking for, but not be body-specific. It is also supported by the findings of Singh et al [4].

The final design of the visualization makes the steering wheel glow blue where touched (Figure 4). The glow has 2 benefits. First, our sensors are a fixed size, so the user could technically touch the sensor in two different locations but have the same sensor detect it. Having touches rendered as a region allows for a believable

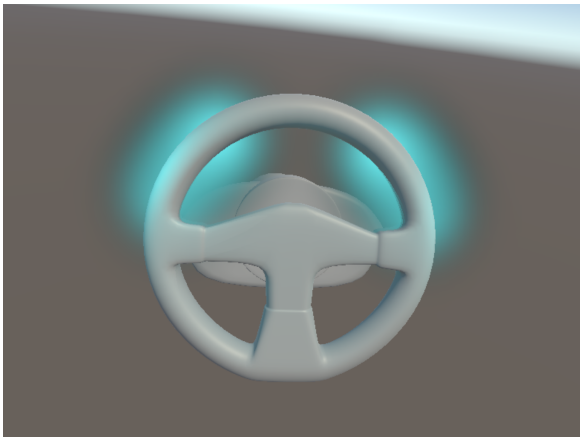

Figure 4: Visualization of two touch events in the virtual environment.

uncertainty in the visualization. Second, if a user grabs the wheel and there is a sensor that fails to detect a touch, its neighboring sensors will overlap with it to rendering a complete touch.

2.5 Challenges

The team encountered a number of challenges in the implementation of this project, especially when scaling from one sensor to 32 sensors, and thus from one touch point to multiple touch points.

(1) Using a send line on I2C to charge all the sensors simultaneously led to discharging all the sensors at the same rate.
(2) Each sensor has a slightly different baseline charge time, making calibration difficult.
(3) A hardware bug on the MCP23017 caused the board to lock whenever pin 7 on each side was accessed as the value on pin 7 changed.

The initial design of the system architecture used a send line to charge all the sensors simultaneously. This created parallel circuits between each sensor, which caused them to all discharge at the same rate whether they were being touched or not. To solve this issue, we removed the send line and used a single line to both read and write to the sensors. This is possible because reading and writing are never performed simultaneously.

Calibration of each sensor was made difficult by several factors. Firstly, the length of the wires from the touch sensors to their respective GPIO pins varies, with the wires at the top of the wheel being the longest. Therefore, each touch sensor is connected to V_{dd} with a slightly different resistance. The changes in resistance from sensor to sensor caused some sensors to have a different baseline charge time. Secondly, as more people use the wheel and the touch sensor cloth degrades, the electrical properties of the capacitive touch sensor changes, which causes the baseline charge time to shift. Lastly, different people have different skin resistances, so the measurement of the charge time can vary from person to person. These problems are solved by observing the circuit charge times at startup and adjusting the threshold for sensors that have different charge times, and observing for false positives during use of the system so that the thresholds can be adjusted over time. The thresholds are set to just over the baseline charge time so that the system will work for most people with typical electrical properties. However, the system still does not work as consistently for people whom the system is not calibrated for. This is due to the changing electrical properties from body to body, which cannot be predicted. Auto-calibration may help to solve this problem in the future.

Lastly, a hardware bug on the MCP23017 was discovered by the team after the proto-boards had been soldered. This bug causes the MCP23017 to lock if pin 7 on any side was accessed as the value on that pin changes. The bug caused the system to crash periodically as pin 7 was accessed, and the NRF52832 would have to be reset. The bug also led to touch not being consistently detected at 12 o'clock, 3 o'clock, 6 o'clock, and 9 o'clock on the wheel (where the sensors connected to pin 7 on each side were located). To fix the crash issue, the wires connected to the four faulty pins (originally, only two MCP23017 boards were being used, each with two sides) were rerouted to a third MCP23017, where they were connected to four non-faulty pins. Due to time constraints, the third MCP23017 is located on a prototyping breadboard, which has additional capacitances over the soldered breadboards. This causes those four pins to have a different RC circuit from the rest of the pins. Readings from those sensors can still be inconsistent but are more consistent than previously and do not lock the board.

3 RESULTS AND EVALUATION

We evaluated VirtuWheel in two ways. First we did a performance evaluation of each sensor, detecting how many physical touch events were correctly detected and rendered in the virtual environment. Then we ran an exploratory user evaluation to see how different people were able to interact with the wheel while using, and not using, VirtuWheel.

3.1 Performance Evaluation

To test VirtuWheel's ability to detect various types of touch events with different people, we ran an experiment where members of our group touched the wheel in various ways.

(1) Each sensor was touched on it's own.
(2) Pairs of adjacent sensors were touched by the same hand.
(3) Pairs of non-adjacent sensors were touched by different hands.
(4) The steering wheel was grasped in a natural fashion at 8 points.

As these experiments were carried out, the number of touch locations that the system detected were counted. This led to 576 total individual sensor touch events that we could analyze. The results of these experiments are displayed in Figure 5.

If we include the 4 problematic sensors, our success rate in detecting touch events is **85%** across all experiments. As discussed earlier, we believe that if these sensors had been built in the same way as all the others, they would not have been an issue. We also analyzed the recorded touch events that *did not* use these sensors, and can report a **94%** accuracy. We believe this to be a more accurate representation of how our approach succeeds in detecting and visualizing touch interactions.

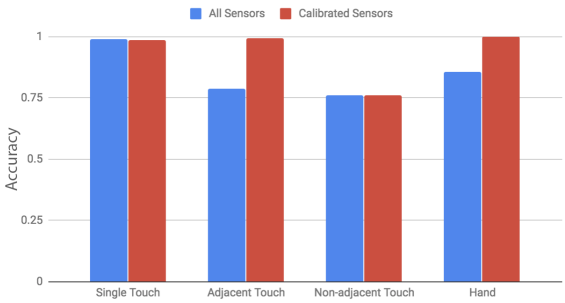

Figure 5: Touch Event Detection Success Rate.

Figure 6: User study setup.

3.2 Exploratory User Evaluation

We ran an exploratory user evaluation of 6 participants, 3 male and 3 female, all licensed drivers with a variety of VR experience, recruited by word of mouth. We ran an A/B test, where group A had the VirtuWheel interaction disabled, and group B had the interaction enabled. The experiment consisted of a variety of tasks interacting with the wheel.

(1) Grab the steering wheel in your natural driving position. Turn the wheel 180 degrees to the left and right.
(2) Grab the steering wheel at the 10 and 2 clock positions. Turn the wheel 180 degrees to the left and right.
(3) Grab the steering wheel at the 8 and 4 clock positions. Turn the wheel 180 degrees to the left and right.
(4) Grab the steering wheel (with either hand) at the 12, 3, 6, and 9 clock positions.
(5) Users were invited to interact with the wheel in any way they wanted for a short time.

After the experience, users were given the Witmer & Singer Presence Questionnaire. [6]

Results from the questionnaire were mixed. We believe this is due to the fact that we ran an A/B test and the participants had a variety of VR experience, so they had no common baseline to compare their experiences against. However, some interesting qualitative observations were made in the study. During the task where we asked participants to put their hands in the clock positions, we noted that several of the participants in group A either missed the wheel entirely, or grabbed incorrect positions. All participants in group B were able to position their hands correctly. We believe they were able to use the visual feedback provided by VirtuWheel to accomplish this. Moving forward, more studies should be performed to explore this phenomenon.

4 DISCUSSION

Overall we consider the project to be quite successful at visualizing touch events, but more work can be done to make the system fully robust.

First, we noticed that certain sensors' baseline charge times would change as the environment changed. To solve this issue we could introduce an auto-calibration feature. Because we can reliably detect when touch events happen, we should be able to read from non-touched sensors and average out their baseline readings over time. This would allow their thresholds to increase as needed.

Another improvement would be to switch to the SPI version of the GPIO expander, the MCP23S17. This would allow us to use the problematic input pins, requiring only 2 chips. It would also allow us to sample at 10MHz, 2 orders of magnitude higher than our current setup. We think that a higher sample rate would allow us to characterize touch event signals better by allowing more precise baseline values.

Another important step would be to verify that this technique generalizes to other types of interactions. Because of our qualitative finding that having visual corrective feedback helps with precision grasping interactions, we could develop another device that could leverage this. For example, we believe this technique would help a lot when interacting with objects that have a dangerous part to touch (ex. soldering iron).

While it is likely infeasible to instrument all surfaces with capacitive touch capabilities, we think that engineers who are building specific props for use with virtual reality should consider adding capacitive touch surfaces so that the interactions with these props can be visualized.

A video for our project can be viewed at https://www.youtube.com/watch?v=wzfDXCfKiVs.

REFERENCES

[1] Mahdi Azmandian, Mark Hancock, Hrvoje Benko, Eyal Ofek, and Andrew D. Wilson. 2016. Haptic Retargeting: Dynamic Repurposing of Passive Haptics for Enhanced Virtual Reality Experiences. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. ACM, New York, NY, USA, 1968–1979. https://doi.org/10.1145/2858036.2858226

[2] Michael Bottone and Kyle Johnsen. 2016. Improving Interaction in HMD-Based Vehicle Simulators Through Real Time Object Reconstruction. In *Proceedings of the 2016 Symposium on Spatial User Interaction (SUI '16)*. ACM, New York, NY, USA, 111–120. https://doi.org/10.1145/2983310.2985761

[3] Inrak Choi and Sean Follmer. 2016. Wolverine: A Wearable Haptic Interface for Grasping in VR. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology (UIST '16 Adjunct)*. ACM, New York, NY, USA, 117–119. https://doi.org/10.1145/2984751.2985725

[4] Avinash Kumar Singh, Hsiang-Ting Chen, Yu-Feng Cheng, Jung-Tai King, Li-Wei Ko, Klaus Gramann, and Chin-Teng Lin. 2018. Visual Appearance Modulates Prediction Error in Virtual Reality. *IEEE Access* 6 (2018), 24617–24624.

[5] Evan Strasnick, Christian Holz, Eyal Ofek, Mike Sinclair, and Hrvoje Benko. 2018. Haptic Links: Bimanual Haptics for Virtual Reality Using Variable Stiffness Actuation. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18)*. ACM, New York, NY, USA, Article 644, 12 pages. https://doi.org/10.1145/3173574.3174218

[6] Bob G Witmer and Michael J Singer. 1998. Measuring presence in virtual environments: A presence questionnaire. *Presence* 7, 3 (1998), 225–240.