VR Ping Pong Team

Logan Herman, William Wang, Henry Yi

Department of Electrical and Computer Engineering, Carnegie Mellon University

***Abstract*—**Our goal is to create a virtual reality table tennis gaming system capable of resembling the real world game, using computer vision, sensors, and physics computation to create a game experience that approximates real-time interaction with the opponent.

***Index Terms*—**computer vision (CV), Elastic Compute Cloud (EC2), inertial measurement unit (IMU)

I. Introduction

Our desired end product is a virtual reality table tennis setup, including the actual game software, "paddles" with sensors, VR headsets, and a camera setup. The application/motivation is the prevalence of social distancing during the pandemic and the implications the isolation has on many peoples' lives and interactions. We want to create a fun and interactive experience that approaches real-time interaction as closely as possible.

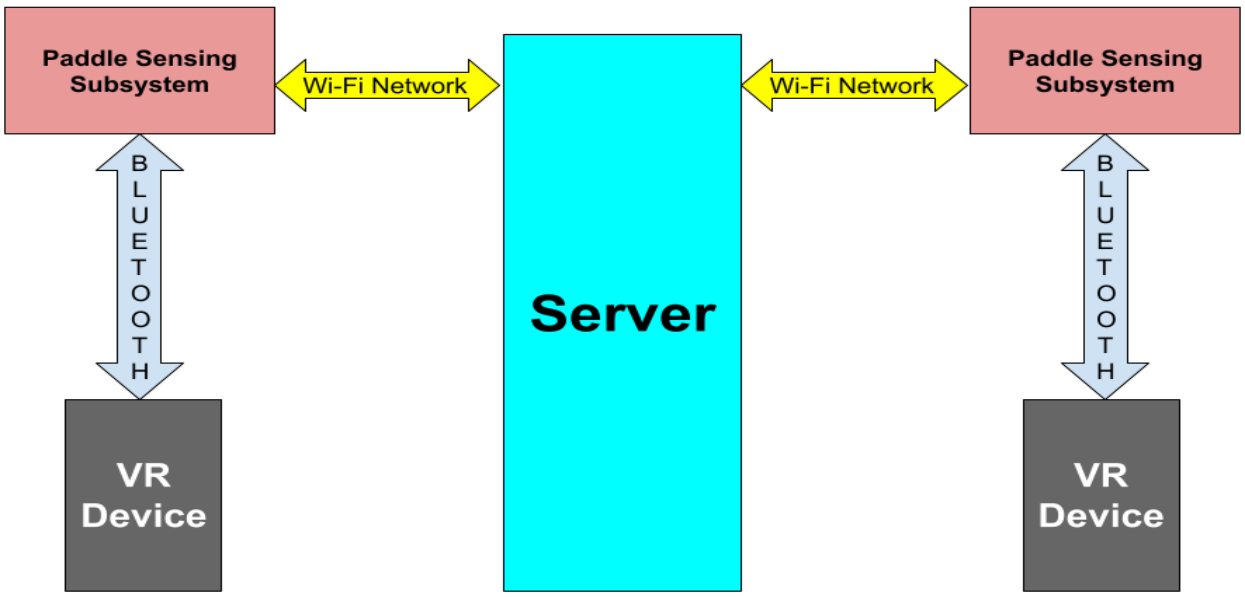
II. Use-Case Requirements

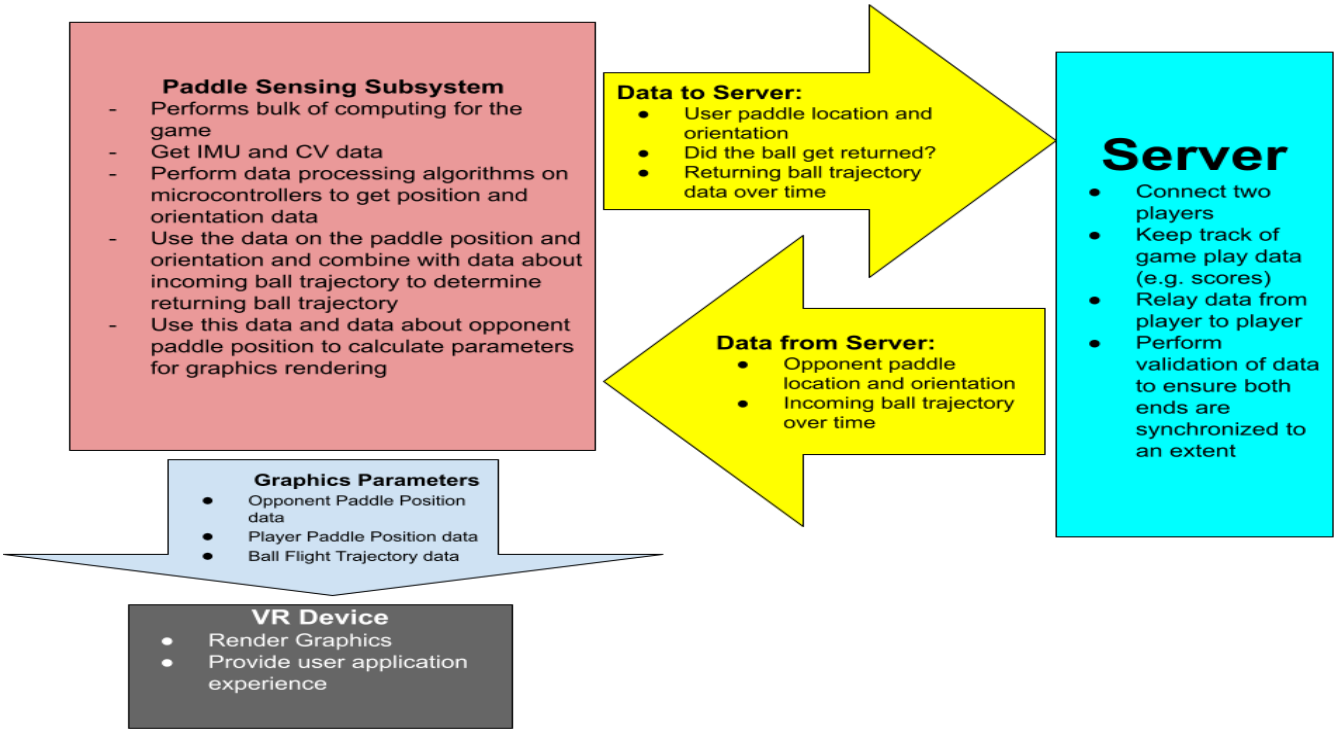
The most desired characteristic of our end product is a realistic feeling when playing. This is difficult to quantify but can be broken down into a handful of metrics that allow us to evaluate this feeling in a quantitative manner. The first and most important is latency. We would like our latency between players to be less than 100 milliseconds, but up to 150 milliseconds is still decent if 100 milliseconds is not possible. The next requirement would be the motion of the paddle. We require the position of the paddle to have at most 3 inches of error. Since the paddle face is about 6 or 7 inches across, then most hits that are in reality near the center of the face will still hit even with an error of about 1 paddle face radius. As for the orientation of the paddle we are aiming for the maximum error angle to be 7.5 degrees. Since for a particular type of shot, the range of paddle orientation will be 90 degrees (aiming over the ball, or under the ball), 7.5 degrees divides this range into about 12 sectors which are notably quite narrow. With an error of 7.5 degrees, the calculated angle will only be off by at most one of these small sectors, resulting in a fairly accurate orientation. In other aspects, we would like our game to have at least 30 frames per second, about the framerate of most video playback and a moderate resolution (360p). We believe that this is enough for the user to comfortably discern what is happening without risking the video quality (which is a lesser concern) costing us more latency time. Lastly we require our power supply in the paddle to last for 1 hour of continuous gameplay. This is because we do not want the paddle to have a wired connection to an external power supply. As for the size/weight of the paddle, most ping pong paddles fall in the range of 200-400 grams. We believe that a little bit of extra weight will not be detrimental as even real paddles deviate greatly in terms of weight. For now we will aim for 500 grams or less, and for the size to be roughly the same size as real ping pong paddles - handle about 4 in x 1 in x 1in, face about 6-7 inches diameter, ½ in thick.

III. Architecture and/or Principle of Operation

The system features a central server, a virtual reality device, the paddle, and paddle sensing system. The paddle sensing system tracks the paddle's position and orientation using a combination of an inertial measurement unit and computer vision, in order to determine the flight path of the ball from the paddle-ball reaction. As seen in the diagrams, the data generated from the sensing system is sent to both the virtual reality device via bluetooth, whose sole purpose is for rendering the graphics, and to the central server via the internet, which relays the data regarding the path of the ball to the other player's device.

Once the data is relayed to the other player, the player's sensing device will perform the same process, combining the motion of the paddle with the incoming ball flight data to determine how the ball will be returned. This process happens in a loop for the duration of gameplay.


Paddle Sensing Subsystem
Wi-Fi Network
Server
Wi-Fi Network
Paddle Sensing Subsystem
BLUETOOTH
BLUETOOTH
VR Device
VR Device


Paddle Sensing Subsystem
- Performs bulk of computing for the game
- Get IMU and CV data
- Perform data processing algorithms on microcontrollers to get position and orientation data
- Use the data on the paddle position and orientation and combine with data about incoming ball trajectory to determine returning ball trajectory
- Use this data and data about opponent paddle position to calculate parameters for graphics rendering
Data to Server:
• User paddle location and orientation
• Did the ball get returned?
• Returning ball trajectory data over time
Server
• Connect two players
• Keep track of game play data (e.g. scores)
• Relay data from player to player
• Perform validation of data to ensure both ends are synchronized to an extent
Data from Server:
• Opponent paddle location and orientation
• Incoming ball trajectory over time
Graphics Parameters
• Opponent Paddle Position data
• Player Paddle Position data
• Ball Flight Trajectory data
VR Device
• Render Graphics
• Provide user application experience

IV. Design Requirements

In order to reach our latency requirement, we will need to be able to transfer the required data between players in a certain amount of time. This is essentially the trajectory of the ball, which could be described by its origin point (when struck), velocity, and spin. Each of these will only be a few bytes, and therefore it is likely that overhead when sending information will dominate our transmission latency, but some time must also be allocated for computation. How much is uncertain but we must be able to compute, send data, and render in less than 100 milliseconds. As for the position of our paddle, our CV setup must be able to discern the location of the paddle and its motion to within 3 inches. Our IMU and Raspberry Pi will handle the orientation, in order to get an accurate representation of the shot, we will try to make at least 3 measurements during the duration of the swing. For the video portions of our requirement, we will have the application (which will run on the phone used on the headset) take in the data and render what the user sees - this will eliminate the need to transmit video frames which would consume a lot of bandwidth and more than likely bloat the latency. We will have to optimize our game application in order to leave as much time for computation and especially transmission as possible in order to meet the requirement of 100 milliseconds for the complete process. As for the weight and size of the paddle, the components we have are the Raspberry Pi, the IMU, and the power supply. We plan to use a power converter such as a Buck converter in order to maximize the battery life of a simple source such as a battery. Aside from the Raspberry Pi which is also quite small and the power supply, each of our components are quite light, which should allow us to have some weight left over for a decent size battery (perhaps one or two AA/AAA size batteries). The Raspberry Pi weighs about 20 grams, Buck converters and our IMU weigh less than that, which leaves only a few short and thin cables before allowing us to allocate most of our weight budget (several hundred grams) to batteries and possibly a casing for the components. Even AA batteries are only about 30 grams each, we should need no more than two, and may even use smaller batteries such as AAA. As for the casing, we can allocate quite a bit of weight. Wii Remotes for example only weigh about 90 grams and our casing should not need to be much larger than those in the worst case scenario.

V. Design Trade Studies

The adjustments that we made to our initial design for this project focused mainly on two requirements: paddle accuracy and latency. The tradeoffs we discussed for latency in particular included communication protocol and inclusion of a server.

A. Paddle Accuracy

In our initial design, we wanted to take advantage of the linear acceleration data that our IMU outputted and take its double integral to find the displacement over time. We imagined that if we calibrated the paddle at the beginning of the game to determine a starting position, we would have the ability to locate the ball accurately at all times. The problem with this idea, however, is drift. Since the IMU is sampling after specific time intervals, double integrating the acceleration is a decent estimation of how the paddle travels over that time interval. But as the game continues, the error in each time interval compounds to the point where the accuracy of the paddle would be very low after a few sets. Our solution of using computer vision to find the lateral position of the paddle is a lot more accurate than our original design because it processes image data to get paddle position at each sample time rather than predicting it. This way, there isn't any compounding error as well.

B. Latency

Our process to optimize latency did not involve any complete changes of course, but rather gradually incorporating optimizations like parallelism, preprocessing, and caching to make data transfers faster and more robust. In the beginning, our plan was to send the raw IMU data to our phones, where it would be processed to determine the orientation and position of the paddle. Each phone would also keep track of the ball position, and when the ball hit the paddle, one player's phone would send the new ball trajectory to the opponent's phone. In this initial design, there is too much computational onus on the phones, too much network communication, and a very high data processing compute time.

Our solution to fix the amount of network communication was to predict the ball trajectory when the ball hits the paddle. Compared to the two phones sending ball location in a continuous stream, predicting the entire trajectory beforehand lowers the frequency of data transfers and more importantly prevents lag in gameplay that can occur when network connections break for short periods of time. The tradeoff in this case is the lag that occurs as the ball is hitting the paddle as a result of the phone computing the new ball trajectory. But we believe that this computing time is short enough where users will not be able to tell.

As for lowering computation time and computational onus on the phone, we decided to introduce EC2 servers or our own computers to project ball trajectory and keep track of the game state, such as the score. With the tradeoff of increased complexity and cost, adding a server can lower the latency

because as the phone is processing the graphics for the VR headset, our server can be processing the new ball trajectory at the exact same time. Compared to our initial procedure where all of this computation would have been done sequentially. Also, Amazon Web Services has great computer networking specs that make data transfer a lot faster than using Bluetooth, which is how we send data from our paddle to the headset.

VI. System Implementation

As earlier described, our system consists of the paddle tracking subsystem, server, and virtual reality device. A key element of our system implementation is how we plan to communicate between subsystems. We plan to address this question by sending packets of information containing data of ball position with respect to time. In other words, we can map out the trajectory of the ball as well as the time frame in which it is happening. This helps us toward our goal of maintaining a real-time game using techniques in each subsystem that will be specified below.

The structure of our system is set up so that our paddle sensing subsystem does the bulk of the computation regarding the paddle-ball interaction in its microcontrollers. All data will pass through this subsystem, and then either be sent to the server for relaying and bookkeeping, or to the virtual reality device for graphics rendering.

A. Paddle Sensing Subsystem

The paddle itself will be a subsystem with two main components in the context of our project. One component is the paddle itself, which will host an IMU and microcontroller to determine orientation and position. The other component is a camera and microcontroller unit that will be placed at a distance from the player in order to provide computer vision help on determining the location of the paddle.

The paddle component will consist of a battery powered Raspberry Pi Zero Wireless connected to a Adafruit MPU6050 6-DOF Inertial Measurement Unit. The battery power functionality will be provided by using a JuiceBox Zero, which holds a battery and provides power conversion functionality so that it can provide the proper 3.3 volts to the Raspberry Pi.

The Raspberry Pi uses its Wi-Fi feature to take in data about the incoming ball trajectory. It also uses an I2C connection with the IMU to take in data regarding the position and orientation of the paddle. Then, it will use a sensor fusion algorithm to process the IMU data and calculate if and how the paddle's movement ends up impacting the incoming ball flight. Once it generates the projected ball trajectory over the next few moments of time, it will send the data to the server via network. In addition, it will send the information about paddle position and ball flight to the virtual reality device using the Raspberry Pi's bluetooth functionality.

The computer vision component will use a camera connected to an NVIDIA Jetson to help in determining the paddle's position. The cameras will be stationed so that we can track the motion of the paddle using tracking algorithms such as color detection. This data will also be relayed to the server and virtual reality device in similar fashion as the IMU unit.

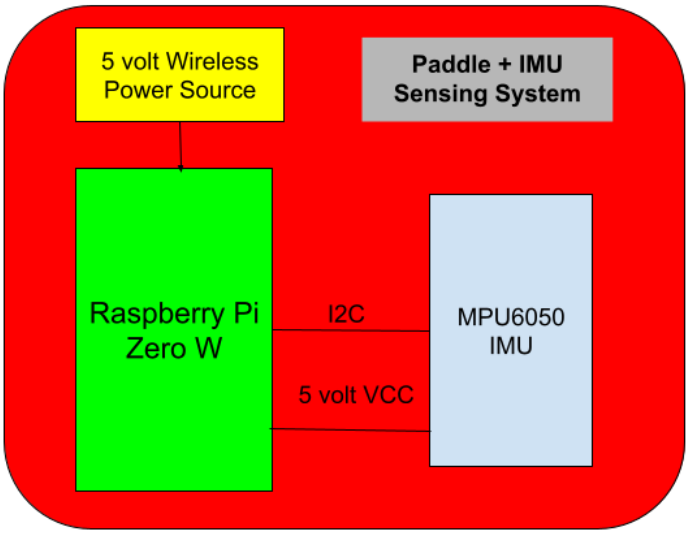
B. Server

The server is our central node for relaying data back and forth. We plan to use a third party server such as Amazon Web Service's EC2 server, which provides low latency, which is an important requirement for our product. The server will take in data from the players regarding the ball trajectory via the network, and then relay it on to the other player.

In addition, the server will perform bookkeeping tasks such as keeping the score between the two players, as well as providing timekeeping so the game speed stays close to real time. The timekeeping involves tracking the latency of the packet coming to and from the server, so when the user-end unpacks the trajectory of the ball over time, it knows which time stamp to start reading the position of the ball. In this sense, the server allows the user to "prefetch" the entire trajectory of the ball, so that it can pick up in real time the location of the ball after the latency of the transmission.

C. Virtual Reality Device

The Virtual Reality Device is the subsystem that provides users with the gaming experience. This will come in the form of a mobile phone mounted on a Google Cardboard headset. The device application will be implemented using the Google Cardboard Virtual Reality development kit. This kit provides the ability to render graphics suitable for virtual reality. The device will utilize its bluetooth connection to receive data from the sensing units regarding the position of the player's paddle and the incoming/outgoing ball trajectories to provide parameters for the graphics rendering.



VII. Test, Verification and Validation

There will be two main things that require testing. The first is to test the paddle sensing system to make sure that it accurately senses the actual movement of the paddle. The other thing to test is the latency from player to server, as we want to make sure that the latency is small enough for the game to feel real-time.

A. Tests for Paddle Sensing System

To test the paddle sensing system, we need to compare physical measurements of various paddle positions and orientations to the reported sensor measurements. For example, we can compare the angles of orientation and physical position coordinates of the paddle in space to what is reported by our sensor. We can also swing the paddle between two points in space and see if the sensor accurately reflects those movements. In addition, we can also test how our graphics display the ball trajectory following a paddle hit to the real ball trajectory after the ball is physically hit by a paddle. We can measure things like where a physical ball lands on the table, the apex of the ball in flight, the time it takes for the ball to complete its trajectory, and even how the ball reacts to different types of spin, and compare it with what our sensing system generates in the graphics. The primary metric we need to address would be the error rate of our calculated data to the true physical elements such as predicted position, orientation, and speed of the ball.

B. Tests for Latency

To test latency, we can check the times that our packets were sent from the device and the time it is received by the other player to see the delays through the network. We can do similar things to test the bluetooth latency between our sensing system and our virtual reality device. For this, the primary metric we would address is the end-to-end latency..

VIII. Project Management

A. Schedule

On the last page of the report is our schedule for designing this project. We want to finish our MVP by the end of March, which leaves us about a month's time to incorporate additional features such as spin and acceleration to the game as well in April.

B. Team Member Responsibilities

For the MVP, Logan is in charge of using computer vision to get the position of the paddle and William is using the IMU to get the orientation. Henry is in charge of creating bluetooth connections between headsets, paddles, and cameras as well as creating the basic graphics for the game

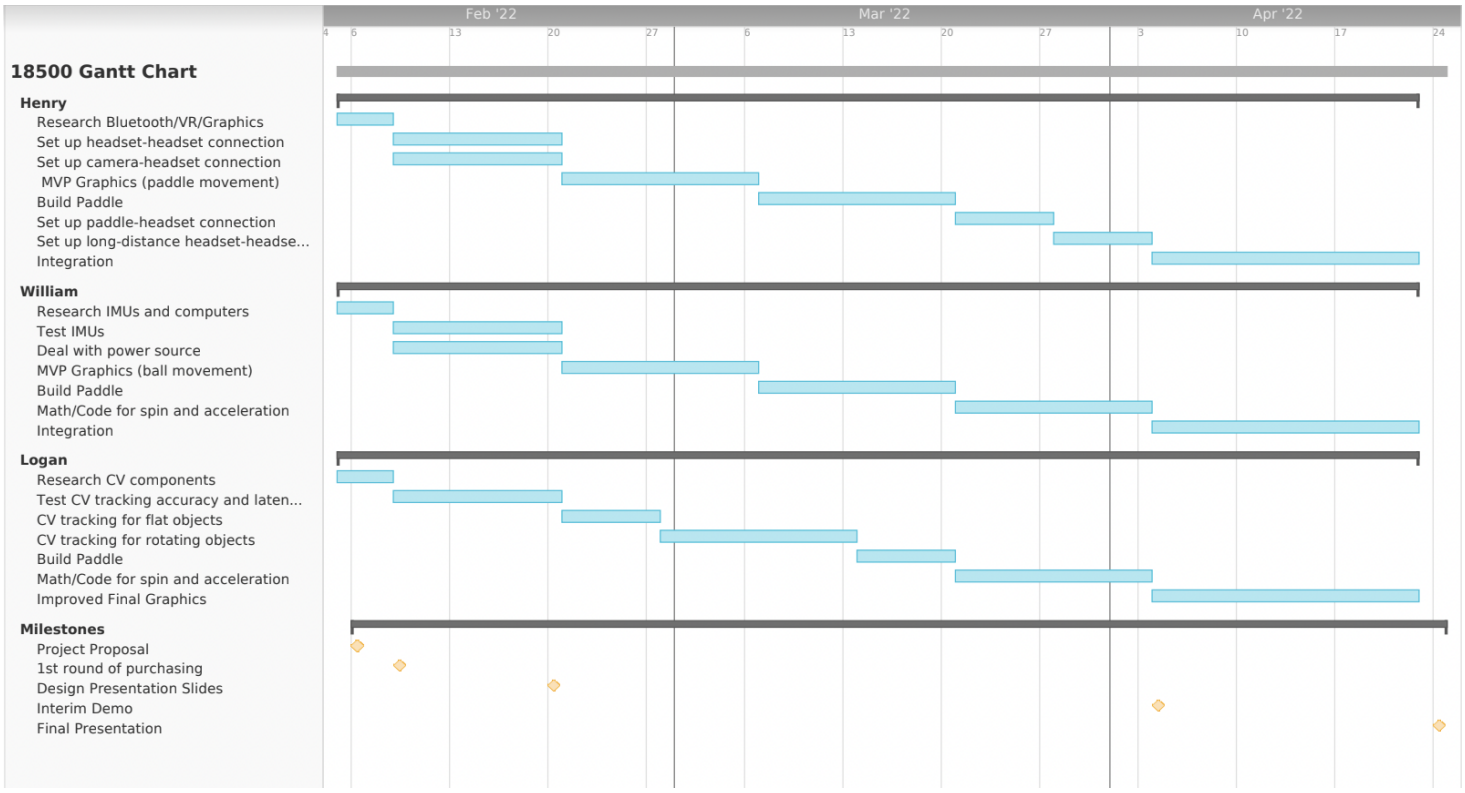
For the final project, Logan will continue to work on computer vision as well as creating more advanced graphics, as he has the most experience. Will is in charge of building the paddle and adding ball movement code to the server. Henry is in charge of creating communication channels for long-distance interactions.

C. Bill of Materials and Budget

On the last page of the document I have included a table with our Bill of Materials and budget breakdown.

D. Risk Mitigation Plans

Our biggest risk is us working with products that we do not have too much experience with. For example, Henry doesn't have much experience with Bluetooth communications and AWS. However, for all parts of our projects we have a contingency plan to change the component to something we are more comfortable with or could perform better. For our server, if AWS EC2 does not work, we can use someone's computer to do the computation. If Bluetooth ends up being too slow, Wifi-direct is an option. And if computer vision does not give accurate enough positional data, we can change the design of the paddle to make the position easier to pin down.



Item:	Quantity:	Purchased From:	Price:
Raspberry Pi Zero	2	CMU Inventory	0
Jetson Nano	1	CMU Inventory	0
Adafruit 6-DOF IMU	1	Adafruit	9
Google Cardboard VR Headset	2	Amazon	24
Bluetooth Adapter	1	Amazon	14