Exploring User Reflexes by Enhancing Remote Control Systems through Virtual Reality Integration with an RC Car.

Frances Raphael
Stanford University
fraphael@stanford.edu

Sylvia Chin
Stanford University
sylc@stanford.edu

Abstract

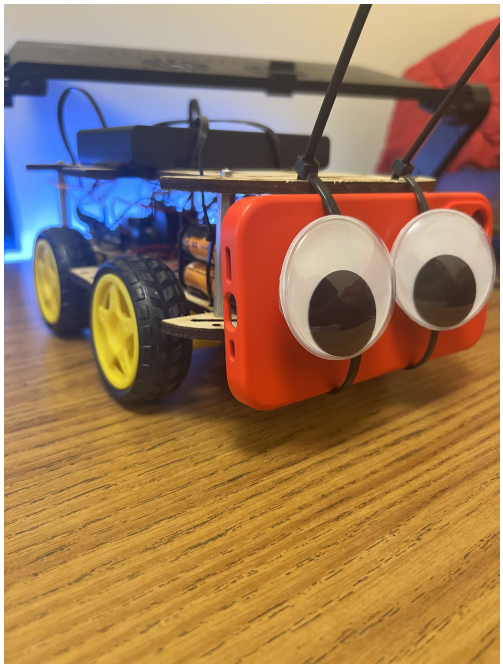

Figure 1. Pictured above is LadyBug

This study explores the implications of integrating head movements alongside traditional hand controls in immersive VR teleoperation, particularly within the context of remote-controlled car driving. We posit that the incorporation of head movements alongside manual controls will not only enhance user experience but also improve overall performance metrics. Participants will engage in navigating a designated course using a combination of head tilting and two distinct hand control methods. By conducting this experiment, our objective is to uncover nuanced insights into the effectiveness of integrating head movements within VR teleoperation scenarios. Such findings hold the potential to significantly elevate the intuitiveness and immersion of VR applications, paving the way for more seamless interactions and enriched user experiences in virtual environments.

1. Introduction and Motivation

The emergence of applied virtual reality (VR) research is bringing about significant advancements in both technology and society. There is growing interest in achieving a deeply immersive experience by combining virtual and physical realities. However, a notable gap remains: our reflexes and visual perceptions do not always align smoothly in VR applications. This is evident even in simple gaming scenarios, where users control avatars using thumb movements on a screen, yet their reactions often involve full-body reflexes. While VR aims to incorporate more natural body movements, it struggles to fully understand the complexities of human motion. Therefore, before we can confidently use VR for complex tasks like surgical simulations, we need to understand how different types of body movements impact user performance in simpler contexts like gaming. Our research project aims to investigate how different types of body movements affect user performance in gaming. To do this, we plan to conduct user tests using a first-person virtual reality remote-controlled (RC) car. We will compare two scenarios:

1. Users will navigate the RC car through a course by viewing the path through a VR headset while employing keyboard arrow controls and lateral head movements to control it.

2. Users will navigate the same course using a VR headset to view the path and employing both a mouse keypad joystick and lateral head movements to control the RC car. For instance, tilting the head to the right will cause the car to turn right, without requiring the use of the handheld controller.

This research not only aims to improve the effectiveness of body movements in VR but also serves as a reality check for how physical movements translate into virtual environments. If a simple head tilt can successfully steer a car away

from an obstacle, this could be a valuable feature to incorporate into VR systems. However, if unintended movements cause users to lose control, these issues need to be addressed in future VR designs. Will users navigate more cautiously with a fixed camera angle, or will they still drive recklessly, relying solely on their instincts? Our experiments aim to capture and analyze users' behaviors and strategies in both scenarios to better understand how humans adapt and correct their spatial perceptions and navigation skills in teleoperated systems. In summary, our contributions are as follows:

- **Investigating Teleoperation in VR** Our project explores teleoperation within VR environments and enhances user experiences and interactions through innovative applications in remote-controlled vehicles.

- **Innovative Dual Operation Modes Implementation** We contribute to VR by proposing and implementing a system with two distinct operation modes, offering users the flexibility to control the virtual environment using both handheld controllers and head movements. This approach enhances interactivity and immersion, allowing users to navigate and interact with the virtual world more naturally and intuitively.

2. Related Work

2.1. Integration of Motion Sensors in VR

Through our short literature review, we identified a few papers that offer valuable insights and potential methodologies for our project. The study by Quang Hoan Le and his colleagues, on a virtual reality crossbow game provides a compelling example of integrating external motion sensors to capture user behavior [3]. Additionally, this paper focused on a remote control system integrated with an intuitive vision-based system for Field Robots, and this was done so by utilizing a head tracker system [3]. This research is important because it addresses the challenges associated with operating excavators in hazardous environments by employing a head-mounted display (HMD) equipped with an IMU sensor to return the orientation. This motion data was then used to control the pan or tilt motion of a camera on the Field of the Robot, demonstrating the feasibility of our project. Moreover, existing research has explored methods to measure physical reactions to virtual environments, particularly when users are startled or prompted [5]. Building on this foundation, we are optimistic that further exploration into integrating head movements, in particular, into virtual environments can provide valuable insights into enhancing performance in such settings.

2.2. Remote Control Systems for VR

In addition to exploring the tracking systems, our second choice paper presents a novel approach to remote control systems for microelectric vehicles (EVs) using downward-facing wide-angle cameras [7]. Unlike most other papers, this paper focused on vehicles operating on different terrains, and this will be relevant for our project as we start making different tracks. Furthermore, the fourth paper focuses on home automation systems controlled by head tracking in Ambient Assisted Living (AAL) applications and introduces software based on computer vision techniques [6]. The results of this paper will inform us of how we can hopefully replicate their accurate data transfers between the different devices on their system. With the use of user testing, such as in one paper that considered head gaze as an additional input to user control, we aim to determine a reasonable difference in using more physical movement to enhance a VR experience [1].

2.3. Communication Protocols and Controls

In addition to the studies previously mentioned, a potentially helpful paper compares the response times of two input modalities (handheld controller and hand tracking) in a VR-based reach-grab-place task [2]. This study evaluated the usability and user performance differences between hand-tracking and handheld-controllers, as well as their overall quality of experience. While handheld-controllers provided reliable input, hand-tracking encountered issues with tracking and self-occlusion. Interestingly, there was no significant difference in user-perceived immersion, involvement, and realness between the two modalities. However, virtual hands were perceived to be mentally, physically, and temporally demanding compared to handheld-controllers. Additionally, the third related research that was done will likely be of the scope of this project; nonetheless, all the techniques used will be important for us when we begin writing a communication protocol for the different devices [4]. This paper investigates the viability of hand-tracking as a naturalistic input modality in VR environments, comparing it with handheld-controllers in a simulation-based reach-grab-place task. The study reveals that while hand-tracking did not improve subjective feelings of presence and engagement, it led to longer task completion times and higher perceived mental workload compared to handheld-controllers. These findings underscore the importance of considering both performance and user experience factors when designing communication protocols and control mechanisms for VR applications.

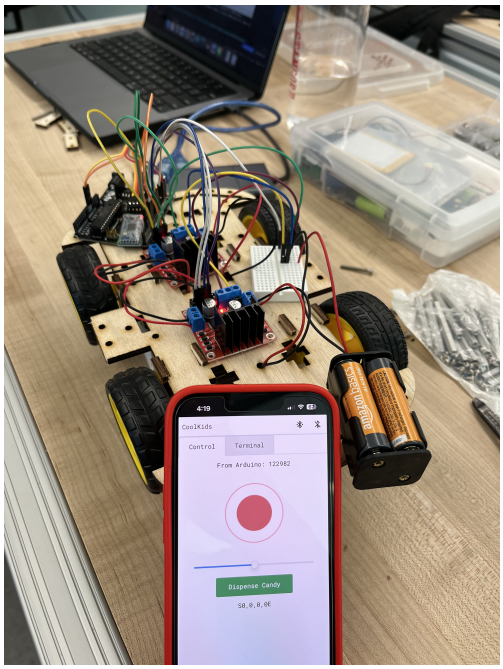

Figure 2. A picture of the car's chassis.

3. Implementation

3.1. Designing a Remote Controlled Car

For this project, we designed a Bluetooth-controlled car. Initially, we considered several approaches, including creating our car model using CAD. However, we opted to source an existing model online due to design constraints. [Figure 1] shows an image of our chosen car's chassis. For the electronics, we chose an Arduino Uno as the car's brain, along with L298H motor drivers to control the four brushed DC motors. After obtaining the CAD models, we proceeded with laser cutting the necessary parts. For the car's controls, we built on the EE64SI controller code to communicate with the Ladybug and send it commands. We used a Bluetooth module for communication and adapted motor control code from a previous project in CS107E. Below is a high-level overview of how these modules function.

3.2. Bluetooth Communication Module

The Bluetooth Communication Module oversees the reception and transmission of commands between Ladybug's remote and its Arduino system. It utilizes a `SoftwareSerial` interface to establish communication with the Bluetooth module, initialized through the `config_remote` function. The `remote_set_name` function simplifies device naming. Command retrieval is handled by the `remote_get_command` function, which extracts joystick and button data from the Bluetooth module and structures it into a Command format. Feedback from Ladybug to the remote is managed by the `remote_send` function. The Motor Control Module, adapted from a CS107E final project, regulates Ladybug's motor speed and direction. The `config_motors` function initializes motor pins, while the `spin_motor` function adjusts individual motor speeds and directions based on input. These modules are seamlessly integrated into the car control script, `car.ino`, which orchestrates setup and looping operations. In setup, serial communication, motor configuration, and Bluetooth module initialization occur. During the loop, commands from the remote are continuously processed to adjust motor speeds, and feedback is transmitted back to the remote.

3.3. Implementing the Camera Module

Initially, we chose the ESP32 Camera module due to its ability to stream its camera feed to a website accessible by any device. However, we encountered significant challenges with this sensor, including the need to manually boot and load code into the module, as well as frequent partition scheme errors. Additionally, the ESP32's range and latency were problematic, likely due to limitations in the hardware's video codec. Due to these issues, we switched to using a Raspberry Pi, expecting better performance. Unfortunately, the latency was just as bad as with the ESP32 module. Ultimately, we decided to attach an iPhone to our car, referred to as "Ladybug," to capture the perspective. This approach significantly improved the feed quality and reduced latency. To stream the camera data, we developed a Python script using OpenCV and Flask. The script captures video frames from the iPhone, processes the frames to create a virtual reality (VR) effect, and streams the processed video feed to a web interface. The main steps of the implementation are as follows:

- **Frame Capture:** The script continuously captures frames from the camera.
- **Frame Processing:** Each captured frame is divided into left and right halves. These halves are resized to a desired width while maintaining the aspect ratio. The resized frames are then combined side by side to simulate a VR feed.
- **Streaming:** The combined frames are encoded in JPEG format and streamed to a web interface using Flask. The web interface displays the video feed in two rectangles, which align with a VR headset for an immersive experience.

In our HTML implementation, we ensured that the two video streams appeared in separate rectangles and aligned correctly with the VR headset. Despite using a single camera and splitting the video feed into two streams, we found

that this approach provided a usable VR experience for short durations, with acceptable motion parallax for about five minutes. This method allowed us to achieve a practical solution with the available hardware, significantly improving the overall user experience.

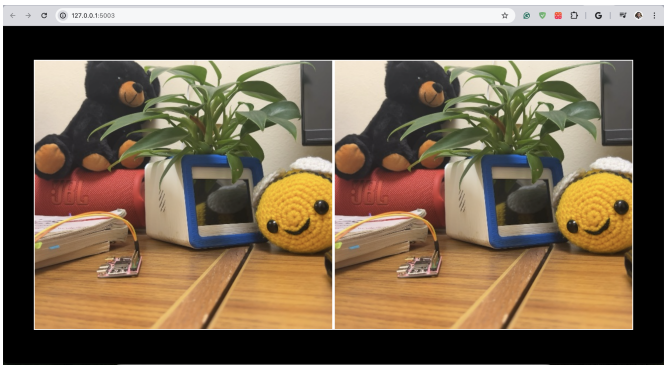

Figure 3. A picture of the website interface.

3.4. HMD Tilt Angle Implementation

For the implementation of the head-mounted display (HMD) tilt angle detection, we utilized an inertial measurement unit (IMU) sensor mounted on the HMD, which interfaced with a Teensy microcontroller. The goal was to accurately read the orientation of the HMD and translate it into usable data for our system.

- **Initialization and Setup:**
  We used a complementary filter with a value of 0.5 to balance between the accelerometer and gyroscope readings.
- **Bias Measurement and Initialization:**
  During setup, the Teensy initializes the IMU and either measures the bias and variance of the sensor data or sets a pre-determined bias. This bias adjustment is crucial for accurate orientation tracking.
- **Orientation Tracking Loop:**
  The main loop processes IMU data continuously, calculating the orientation in the form of a quaternion.
  From this quaternion, we compute the yaw angle, which represents the horizontal rotation of the HMD. Based on our experimentation, we found that the yaw angle provided the most reliable results, especially when the user is seated.
- **Data Transmission:**
  The yaw angle is then sent over the serial interface to be used by other parts of the system. This data is crucial for controlling the perspective of the camera feed to align with the user's head movements.

This implementation ensures that the tilt angle of the HMD is accurately tracked and used to enhance the user experience by synchronizing the camera feed with head movements. By focusing on the yaw angle, we were able to achieve the best results in terms of responsiveness and accuracy while the user is seated. In addition to tracking the tilt angle of the head-mounted display (HMD), we also ensure seamless integration with the website that controls the remote-controlled car. The server.js script serves as the intermediary between the Teensy microcontroller, which reads the HMD tilt data, and the web interface.

- **WebSocket Server:**
  The script sets up a WebSocket server on port 8081, allowing multiple clients to connect simultaneously.
- **Serial Port Communication:**
  It establishes communication with the Teensy over the serial port, continuously reading data from the HMD sensor.
- **WebSocket Transmission:**
  When new data is received from the Teensy, it is broadcast to all connected WebSocket clients. This includes the web interface responsible for controlling the car.

By streaming the HMD tilt data in real-time to the car control website, we enable responsive and intuitive control of the remote-controlled car. Users can simply tilt their head in the desired direction, and the car will respond accordingly.

3.5. Controlling the Car via Website Interface

The web-based interface, encapsulated in `website_control.html`, facilitates remote control of the car through intuitive mechanisms. Leveraging WebSocket communication, the interface establishes a real-time connection with the server to receive updates on the tilt angle of the head-mounted display (HMD) worn by the user. This tilt angle data is prominently displayed on the interface, offering users immediate feedback regarding the orientation of the HMD. Complementing this feedback mechanism is a joystick controller implemented using `joy_stick.js`, enabling users to intuitively manipulate the car's direction and speed. Additionally, the integration of arrow keys, including left, right, up, and down arrows, extends control options, allowing users to steer the car in all directions, while the space bar acts as a brake for immediate halting.

4. Analysis, Evaluation, and Comparative Study of Methods

Our implementation of the remote-controlled VR car, Ladybug, presents several distinct differences compared to the methods described in the related work section. Firstly, while prior studies such as Le et al. utilized head-mounted displays (HMDs) with IMU sensors to control the orientation of external devices like cameras, our project expanded this concept by integrating a full remote control system for a car using both head tilt and traditional input methods like arrow keys and joysticks. Unlike the vision-based head tracking used in Tsujita et al.'s remote control system for EVs, we focused on real-time feedback and control through a Bluetooth module interfaced with an Arduino, leveraging `SoftwareSerial` for communication. Our unique approach to combining head movement for directional control and traditional input methods for speed and braking adds an extra layer of intuitiveness and immersiveness to the user experience. Additionally, instead of relying on dedicated hardware sensors for head tracking, we opted for a more accessible setup using a Teensy microcontroller to interpret IMU data from the HMD, thereby simplifying the hardware requirements. Our method of using an iPhone for video streaming with OpenCV and Flask also diverges from the approaches taken in related works, offering a practical and low-latency solution to visual feedback. These differences highlight the versatility and adaptability of our implementation in creating an immersive and responsive VR remote control experience.

5. Results

We had two scenarios for our user tests. In Scenario 1, users utilized arrow keys combined with head movement to control the car. In Scenario 2, users used a mousepad joystick in conjunction with head movement. For each scenario, we had five users run a simple obstacle course twice. The first run allowed users to familiarize themselves with the controls, while the second run assessed their ability to optimize their performance.

In Scenario 1, users averaged 2.63 minutes on their first run and improved to 1.42 minutes on their second run, resulting in an overall average of 2.02 minutes. For Scenario 2, users averaged 2.72 minutes on their first run and 1.52 minutes on their second run, yielding an overall average of 2.12 minutes. Notably, the best performance across both scenarios was achieved in Scenario 2, with the fastest time being 49 seconds.

All users managed to complete the course within a maximum of 5 minutes using the first-person VR stream and dual controls specific to their assigned scenario. However, a few individuals could not finish the course due to VR-induced motion sickness and dizziness, a common challenge in VR applications but beyond the scope of this project. Additionally, some undocumented volunteers chose to run the course and successfully completed it at their discretion. In the exit survey, users were asked to rate their control on a scale from 1 (weak control) to 10 (strong control), as shown in Figure 5. They were also asked to rate the immersiveness of their

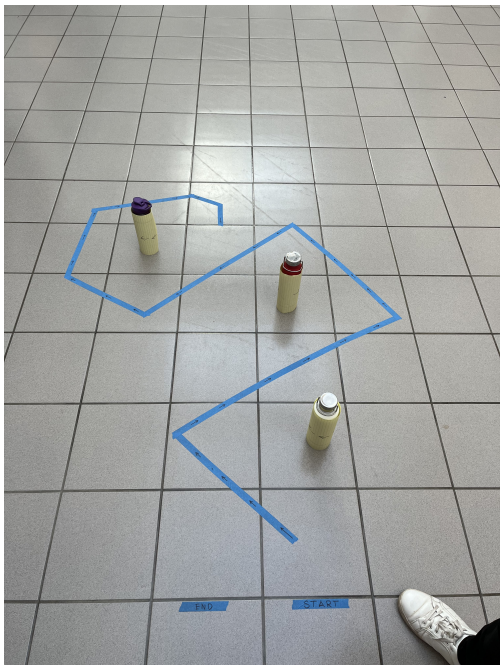

Figure 4. This figure shows a top view of the test setup terrain

Figure 5. This figure shows a participant using the trackpad.

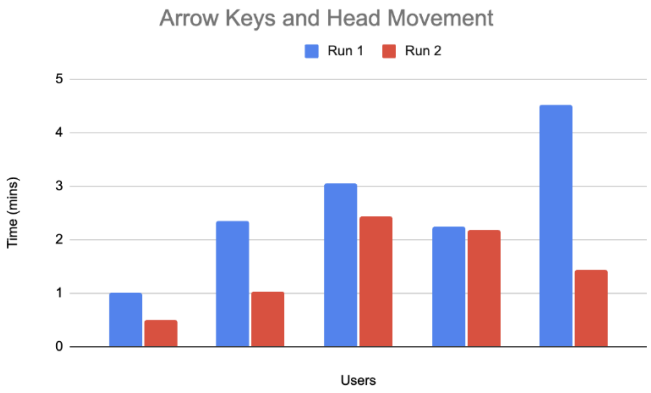

Figure 6. This figure illustrates the times for Scenario 1

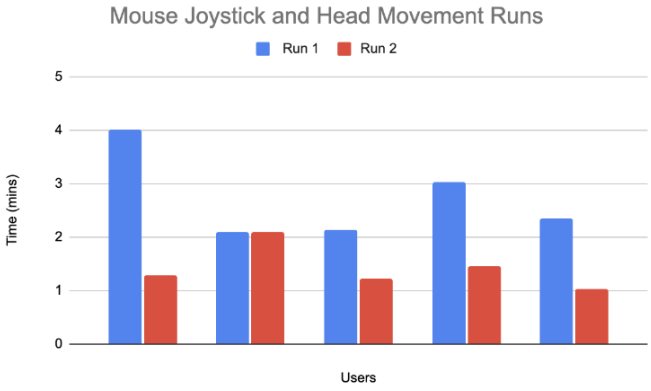

Figure 7. This figure illustrates the times for Scenario 2

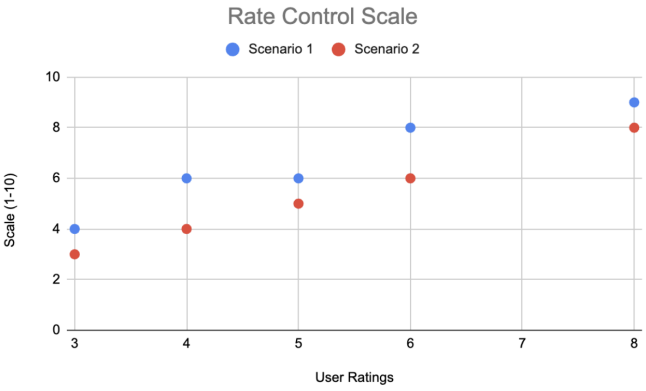

Figure 8. This figure illustrates the times for Scenario 2

experience on a scale from 1 (low) to 10 (high), as depicted in Figure 6.

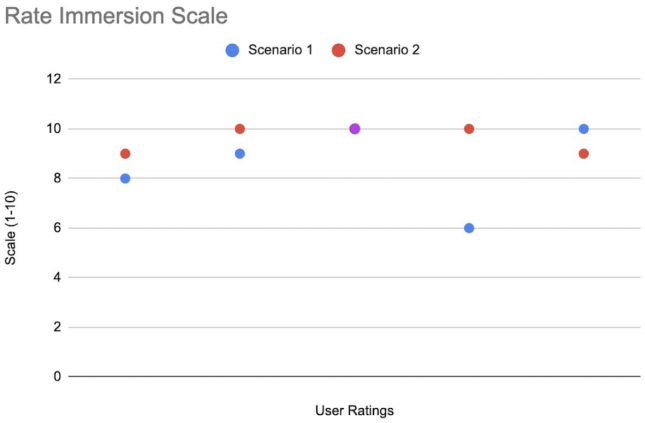

Figure 9. This figure illustrates the times for Scenario 2

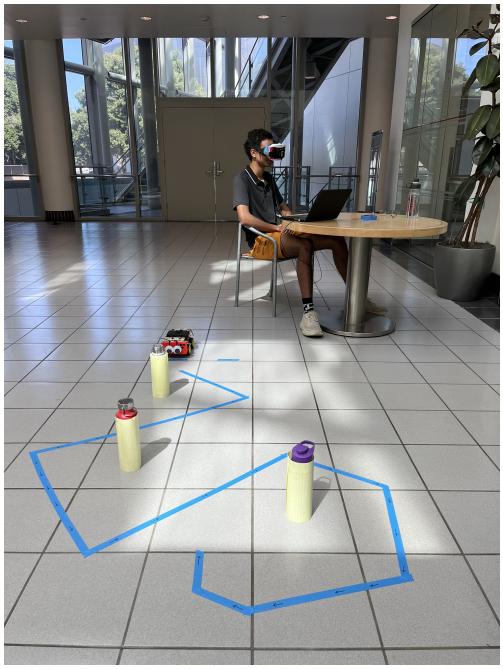

Figure 10. This figure shows the full setup and a participant finishing the course

6. Discussion

6.1. Experiment Analysis

While the simultaneous head and hand controls were initially disorienting for users, our brief experiment yielded valuable insights into the relationship between physical instinct and remote control.

Qualitatively, several users reported that the head movement was unexpectedly intuitive and helpful in traversing the course. Some users noted that the head tilt was extremely sensitive; we hypothesize this is due to the high mathematical interpretation of the tilt angle we pro-

grammed and the unfamiliarity with head movement affecting game movement. Some users focused more on using the hand controls and neglected the head tilt. Interestingly, in those cases, their heads still moved forward and backward unconsciously in an attempt to move the car while avoiding the left/right tilt, knowing it might affect the car's movement. This suggests that head movement is an instinct users try to project onto the car, but our rudimentary and highly sensitive implementation may have incentivized them to rely more on hand controls.

Quantitatively, the timed results suggest that adjusting to the new controls nearly halves the time it takes to complete the course on Run 2 compared to Run 1 for both Scenario 1 and Scenario 2. While we cannot directly attribute this improvement to the success of head movement, it is promising to see that head movement can be practically incorporated into better performance. Users rated their control in Scenario 1 higher than in Scenario 2, which anecdotally had the more difficult hand control (mouse joystick). Overall, users rated the experience as highly immersive due to the first-person perspective and the incorporation of head movement into the controls.

6.2. Limitations

In our allotted time, we were only able to incorporate roll head movement (tilt left and tilt right). With a more accurate IMU on the HMD, we would like to incorporate a more directionally sensitive set of head movement controls — at minimum forwards and backward pitch. We were also unable to host a private server for a mobile controller, so we had to use a computer for the mouse joystick or arrow keys. Having a separate hand controller would be ideal because it would allow for greater physical mobility. We also recognize that we were user testing on a very small sample of users and see the value in a more thorough study.

7. Future Work

Beyond technical improvements, there is potential to incorporate additional physical movements, such as torso or foot gestures, into game control. Using Ladybug as a quick proof of concept, we aim to explore new methods that seamlessly blend physical and virtual realities. Enhancing the first-person perspective is also a priority. This could involve mounting a higher, downward-facing camera and potentially implementing camera rotation for a full 360-degree immersive experience.

Currently, we use a monocular view due to having a single camera attached to the car, resulting in significant motion parallax. By integrating a second camera, we could reduce motion parallax and introduce depth to our video feed. Furthermore, acquiring AWS credits would enable us to publish data to the website across multiple devices, enhancing the accessibility and functionality of the website controller.

8. Conclusion

Blending the physical and virtual worlds to create an optimal immersive experience remains a significant challenge for VR applications. This is particularly evident in the gaming market, where physical reflexes and minimizing virtual latency are crucial for performance, making integrated solutions highly valuable.

Testing the Ladybug VR RC car highlighted several phenomena expected in a gaming product. One key insight from the Ladybug user tests was that incorporating head movement can effectively reduce dimensionality. Specifically, using the x-axis for head tilt and the y-axis for hand controls proved to be a successful strategy. This approach of physical dimension reduction challenges traditional notions of remote control, and our preliminary results suggest a promising future for utilizing various physical instincts to manage different aspects of direction.

Additionally, an interesting takeaway is that users naturally prefer to use their heads for controlling movement, even if not as initially intended. The Ladybug trials represent a step towards bridging the gap between physical instinct and virtual movement. With further iteration, we are confident that precise head movement could become a key factor for optimized navigation in virtual reality.

References

[1] B. R. S. M. I. C. J. . F. V. Atienza, R. Interaction techniques using head gaze for virtual reality. *IEEE Xplore*, 2016.

[2] A. Hameed, S. Möller, and A. Perkis. How good are virtual hands? influences of input modality on motor tasks in virtual reality. *Computers in Human Behavior*, 29(6):2447–2455, 2013.

[3] Q. H. Le, K.-T. Park, Y.-M. Jeong, and S.-Y. Yang. A study on remote control system integrated intuitive vision system for field robot using a head tracker system. *IEEE Xplore*, 2014.

[4] Y. Li, X. Zhou, and R. Yu. Control system for a light and small pan-tilt based on multirotor uav for aerial remote sensing applications. In *IGARSS 2018 - 2018 IEEE International Geoscience and Remote Sensing Symposium*, 2018.

[5] J. Schirm, G. Tullius, and J. Habgood. Towards an objective measure of presence: Examining startle reflexes in a commercial virtual reality game. *ACM Digital Library*, 2019.

[6] S. Spinsante and E. Gambi. Home automation systems control by head tracking in aal applications. In *2012 IEEE First AESS European Conference on Satellite Telecommunications (ESTEL)*, 2012.

[7] N. Tsujita, M. Mitsuno, Y. Usui, and S. Kaneda. Remote-controlled micro ev system using downward-facing wide-angle video camera. In *2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)*, 2016.