VR Sign Language Calculator

Adam Meyers
meye3224@umn.edu
University of Minnesota
Minneapolis, Minnesota, USA

Ekin Ercetin
ercet001@umn.edu
University of Minnesota
Minneapolis, Minnesota, USA

Jacob Secunda
secun006@umn.edu
University of Minnesota
Minneapolis, Minnesota, USA

ABSTRACT

The VR Sign Language Calculator is a virtual reality application that uses hand tracking for input, eliminating the need for virtual keypads. Users can perform mathematical operations by signing in American Sign Language (ASL) to input numbers and operations. Built using the Unity game engine for the Meta Quest 3 headset, the system recognizes hand gestures in real-time using the XR Hands and XR Interaction Toolkit packages. The resulting application provides an accessible and easy-to-use interface for those both with and without ASL experience. The calculator operates in a simple virtual environment, providing clear feedback to guide users. Early testing reveals that the application accurately recognizes gestures and has potential for use in other virtual reality applications, especially those focused on accessibility and/or immersive design.

CCS CONCEPTS

• **Human-centered computing** → **Virtual reality**; • **Human-centered computing** → **Accessibility technologies**; • **Computing methodologies** → **Gesture recognition**;

KEYWORDS

Virtual Reality, Sign Language, Accessibility, Gesture Recognition, Calculator, Human-Computer Interaction, VR Interfaces, Inclusive Design

ACM Reference Format:
Adam Meyers, Ekin Ercetin, and Jacob Secunda. 2024. VR Sign Language Calculator. In *Proceedings of* . ACM, New York, NY, USA, 7 pages.

1 INTRODUCTION

Hand-tracking technology in virtual reality has come a long way since its earliest implementations in the 1980s. Nowadays, even the relatively affordable Meta Quest 3 boasts accurate hand tracking using the built-in cameras on the front of the headset. With how accessible hand tracking in virtual reality has become, we have the opportunity to explore more uses of hand tracking as a method of input and interaction in virtual reality environments.

One such method of interaction is allowing users to use sign language, or other hand symbols, as separate types of input. By tracking the position and orientation of a user's hands and fingers in a virtual environment, we allow for a far greater range of potential inputs than currently available on a typical set of virtual reality controllers. We can also achieve this greater range of inputs without the extensive use of virtual keypads, buttons, or other special input methods. Instead, a single virtual input area can serve a wide range of inputs, solely based on the hand symbol the user creates when interacting with this input area. With proper integration of this input area into an environment, this could even serve as a completely diegetic form of user input.

To show the capabilities of this input method, we have created a virtual reality calculator that features no keypad for numbers or mathematical operations, with these inputs instead being controlled solely by hand symbols that represent each of these numbers, operations, and symbols. A small number of virtual buttons are still used for submitting the calculations and clearing or deleting inputs, but the calculator is otherwise completely controlled with a single input area in which hand symbols are detected and interpreted as separate inputs. While this iteration of the calculator only features a few basic operations, and cannot be used in conjunction with any other programs, it still serves as a proof of concept for sign language-based input in virtual reality.

2 REVIEW OF PREVIOUS WORK

There have been multiple previous implementations of calculators within virtual reality prior to our project. Some prominent implementations include the program CalcVR [1] and a calculator implementation created by GameDev.tv [3]. The GameDev.tv calculator is a recreation of a simple physical calculator, featuring a small display showing the current expression inputs, and buttons to input the digits 0-9, addition, subtraction, division, multiplication, exponentiation, and parentheses, as well as some buttons for deleting inputs and calculating the result of the expression. The computational complexity of this calculator is very similar to the desired computational complexity for the sign language calculator. However, the main difference comes in the method of input, with the GameDev.tv calculator featuring a virtual keypad of buttons. With the current complexity of the calculator, this is a rather convenient method of input. However, If the computational complexity of the calculator was scaled up to include more complex expressions, the size of this virtual calculator could quickly become very large.

An example of a calculator with a much higher computational complexity would be CalcVR, which is a graphing calculator program that allows users to perform not just simple arithmetic, but also more complex calculations such as derivatives and integrals. It also provides the ability to display both two and three-dimensional graphs. In order to achieve this greater range of potential calculations, CalcVR does away with the virtual handheld calculator, instead using a variety of floating input and display windows. There are a very large number of different input and display windows

, ,

featured within CalcVR, but the program never displays all windows at the same time, instead only displaying what is currently relevant for the type of graph or calculation that the user is trying to perform. These windows primarily feature buttons that can be selected via a ray interactor and text inputs, which require a virtual keypad to pop up in order to complete.

This user interface system allows for an incredibly large range of potential inputs in CalcVR without completely overloading the user's view with unnecessary input options. However, in many cases, the amount of options present at any given time is still often pretty high, ultimately dominating the user's field of view with the calculator user interface in many situations. This means the CalcVR program could not easily be used in conjunction with other programs or environments. In addition, unlike the GameDev.tv calculator which attempts to recreate a physical calculator, CalcVR makes no attempt to make its user interface appear diegetically within the environment.

Recent research about opportunities and challenges regarding hand tracking for immersive virtual reality has shown that we use our hands to gesture, feel, and interact with our environment almost every minute of our lives. This fact is strengthened by the fact that we have an almost preternatural sense of our hand's positions and shape when they are obscured, and when our hands are removed from our visual worlds, it's a stark reminder of our disembodiment [2]. A key goal of virtual reality is to allow its users to interact with the environment in a natural fashion. For example, many VR applications require their user to either hold objects, press buttons or use their hands to interact with the objects around them. An accurate tracking of hands allows for far more precision that wouldn't be possible with controllers. A study called "Human Interaction in Virtual and Mixed Reality Through Hand Tracking" talked about the possibilities of hand tracking in VR and MR, focusing its role in human interaction dynamics [4]. The authors design a VR application focusing on three scenes, daily life, education, and recreation. For all of the created scenes, users agree that hand tracking proves to be the more natural, engaging, and easy-to-use option, reducing the learning curve, and allowing for accessibility. Additionally, discarding the controllers eliminates the issue of users not being able to see the buttons and the triggers on the controllers, making the interactions more straightforward.

There also have been multiple implementations of sign language detection in virtual reality using hand tracking. An early example of sign language recognition via hand tracking can be seen in a 2015 article displaying the potential of the Microsoft Kinect to detect and differentiate various American Sign Language gestures [6]. Visual and depth-based data was used from the Microsoft Kinect in order to detect various hand positions. This approach was somewhat limited, with users being required to wear a black wristband in order to segment the hand from the rest of the user's arm, and with the final program only being able to differentiate between 17 different hand gestures. However, it was able to differentiate these gestures with a high success rate of 90.4%.

Another development in hand gesture recognition is the use of artificial intelligence, which has been shown to be capable of ever higher rates of success, such as in 2021, when artificial intelligence was employed to recognize hand gestures within virtual reality [5]. This implementation uses the Leap Motion Controller to gather detailed information on the position and velocity of the user's hands. This information is then passed into a convolutional neural network, which was trained using previously captured hand gesture data. This neural network is then used to classify the data into various distinct hand gestures, with a high accuracy rate of 93% for dynamic hand gestures. This is a very different implementation compared to our solution, as we do not have access to Leap Motion Controller data, and do not use any artificial intelligence in our program. However, similar techniques could theoretically be used with our user interface design to allow for highly accurate detection of dynamic hand gestures.

3 METHODOLOGY

Our methodology for developing an accessible VR sign language calculator was structured into three primary sections: system design, architecture, and UI; hand tracking and gesture recognition; and calculator backend. Each phase was carefully crafted to ensure that the final system would be functional and easy to use for our user group. Below, we provide a detailed description of the processes and techniques employed at each step of development.

3.1 System Design

Our system design included selecting the appropriate hardware and software tools to create a seamless and fun VR experience. The project was developed using the Meta Quest 3 headset, which provides an immersive VR environment with sufficient computational power and tracking accuracy for our application. The VR environment was built using Unity 3D, which allowed us easy integration of VR interactions and rendering of the assets and components used to design our scene. The virtual environment was designed to offer a calm and immersive experience, with a green forest scene chosen to promote a more relaxed atmosphere during interaction. The environment also included various interactable objects, such as a virtual calculator screen and buttons, which the user could manipulate using hand gestures. For the calculator interface, we designed a digital screen where users could perform calculations, with buttons representing operations for clearing the screen, deleting the last digit or operation, and evaluating the equation on the screen. The screen was updated dynamically to reflect the calculations based on the user's gestures and the button interactions. The signed digits and operations were mapped to specific hand gestures, and when the user signed in the interactive space, the calculator screen displayed the corresponding operation or number.

3.2 Gesture Recognition and Tracking with XR Hands

The XR Hands package was utilized for hand gesture recognition and tracking. It provided precise tracking of hand movements and enabled the system to recognize gestures in real-time. The XR Interaction Toolkit was used to detect interactions with virtual objects like the calculator buttons.

Each gesture was designed to correspond to specific calculator functions or digits. For example, performing a specific sign represented a digit, an arithmetic operation, an open/close parenthesis, an exponent, or a dot to represent floating point numbers. The

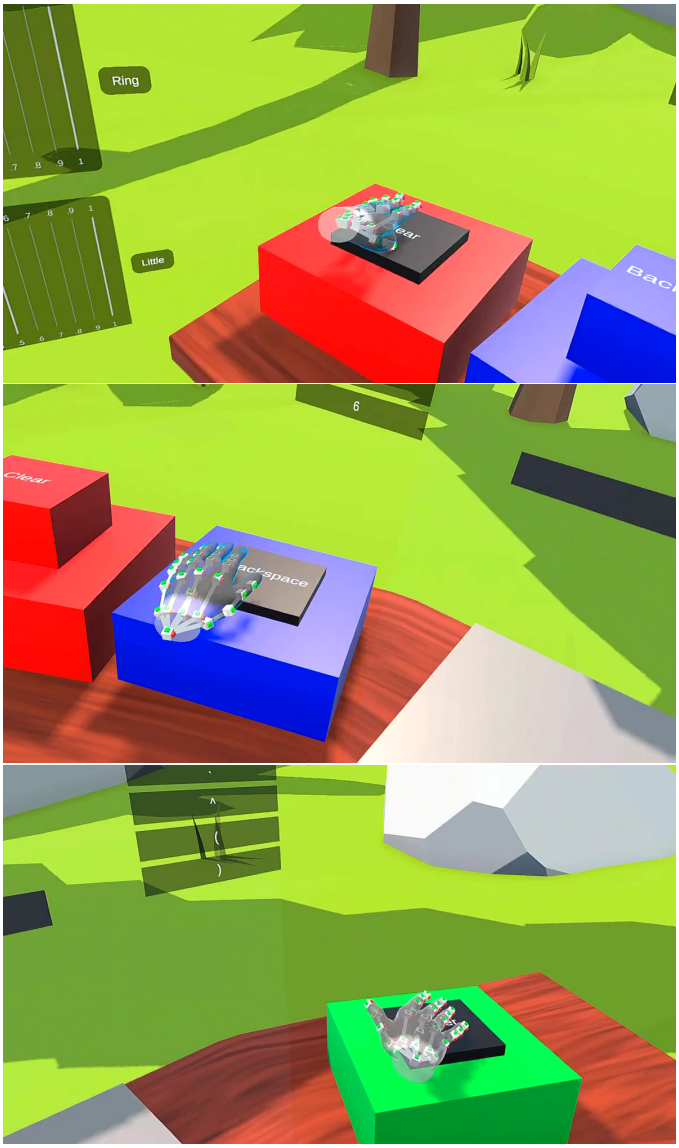

Figure 1: Buttons

system was designed to recognize these hand gestures as input for the calculator.

In XR Hands, gesture recognition works by tracking hand positions, finger movements, and hand orientations within the VR environment whenever the hand is static and in sight of the headset camera direction. Using the XR Hands custom gesture recognizer, we added 17 different custom gestures to our scene. In our scene, we created and configured a Static Hand Gesture component, which references a Hand Shape or Hand Pose asset and an XR Hand Tracking Events object. This component dispatches UnityEvents when it detects that the user's hand matches the configured hand shape and orientation and when a gesture is no longer detected.

To design the digits, we used American Sign Language (ASL) signs to represent each digit from 0 to 9. Each sign was mapped to

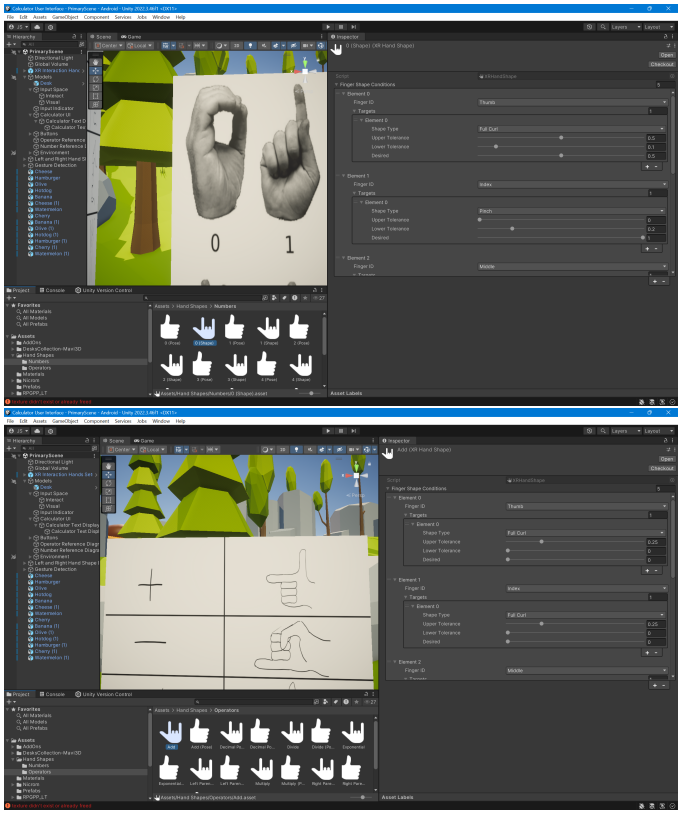

Figure 2: (a) Settings for "0" (b) Settings for "+"

its corresponding numerical value, allowing users to input digits directly by performing the appropriate ASL hand gestures. However, to design the operations, we relied on an intuitive set of hand movements. For example, a fist would correspond to "." while an upside-down two would correspond to "^". Figure 2 shows the gesture configuration settings for "0" and "+". Figure 3 shows a visual graphic of each hand gesture and the corresponding symbols

These gestures were tested and refined for accuracy, ensuring that the system could correctly interpret the user's signs in various lighting conditions and angles.

3.3 Interaction Model and Calculator Functionality

The interaction model was designed with several key features to ensure functionality and ease of use.

3.3.1 Input Area. The interface was built with an interactive area where users could sign their gestures. This area was mapped to the screen of the calculator, and as the user signed, the screen would update to show the corresponding input.

3.3.2 Basic and Advanced Operations. The calculator supports common arithmetic operations such as addition, subtraction, multiplication, and division. It also includes more advanced features like floating-point calculations, use of parentheses, and exponents.

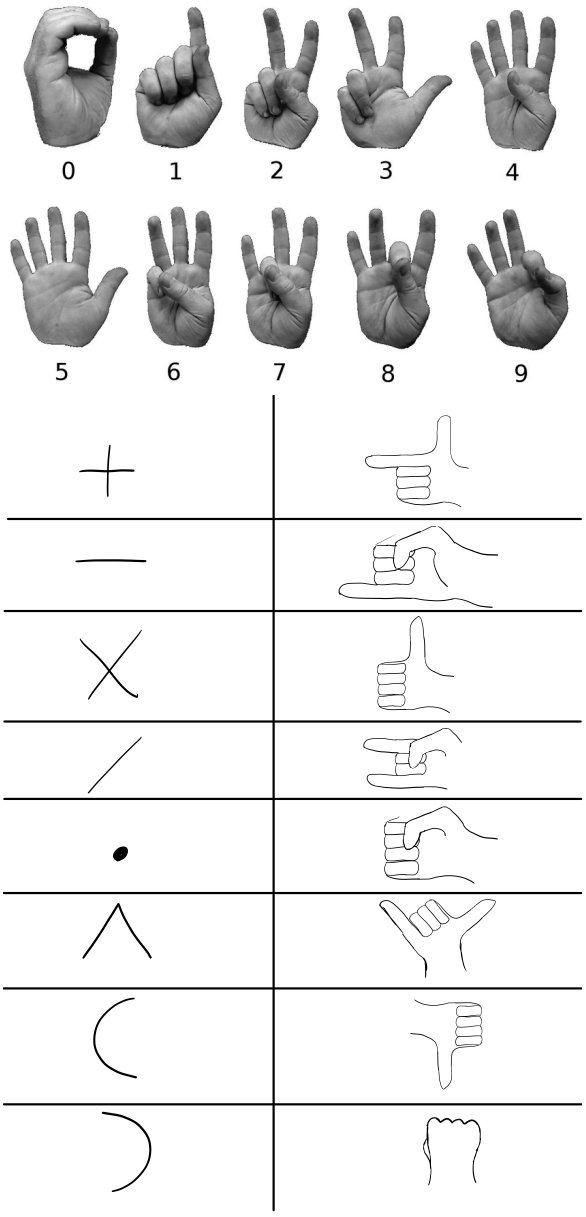

Figure 3: (a) ASL hand gestures for digits 0-9 (b) custom hand gestures for operators

3.3.3 Formatting. To improve user experience, the text displayed on the calculator was automatically formatted based on the operation being performed. For example, pressing the backspace button would automatically erase the spaces between digits and operations, and signing would insert those spaces back. Our evaluation function also checks for incorrect formatting and displays a message on the calculator screen.

3.3.4 Calculator Backend. The calculator functionality was programmed in a C# script. The script takes in the string that is currently displayed in the calculator screen. It then iterates through the characters in the string, identifying numbers and operators. If a formatting error is present, such as multiple operators in a row, or two decimal points within a number, the script detects it during this step and returns an error, which is then displayed on the calculator screen. For each set of parentheses in the string, it recursively calls the evaluation function again, but on the subexpression within the set of parentheses. Once the end of the expression is reached, it then calculates the result using the list of numbers and operators detected within the string, following the standard order of operations.

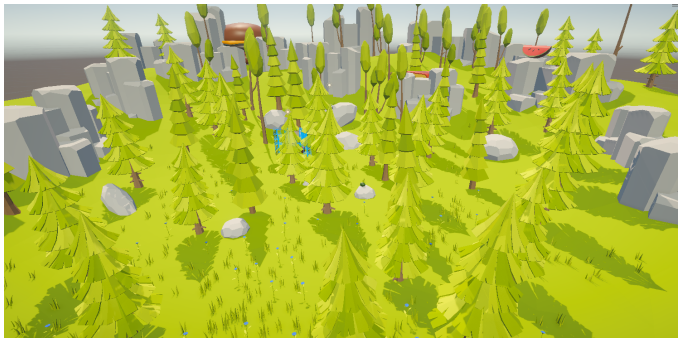

Figure 4: The virtual environment

4 EVALUATION

The system's performance was evaluated in terms of gesture recognition accuracy and user satisfaction. We conducted user testing with two participants: an individual who is experienced with sign language to ensure that the system could accurately interpret a variety of sign language gestures, and an individual who is not experienced in using sign language to test our system's usability. Our participants were asked to interact with the virtual calculator and perform a series of operations with their hands. The following aspects were evaluated: gesture accuracy, usability, latency, and any other additional user comments and feedback.

4.0.1 Environment. User testing results showed that all three of our users had mostly positive opinions and feedback about the green forest environment (Figure 4 and 5). They thought the forest scene was relaxing and nice to look at. Some specific comments we got were about the swaying trees and the food item easter eggs that we put for people who wanted to look around the scene. However, one weakness that two of our users pointed out was how big the desk and the interactable objects were compared to the XR player object. Users mentioned feeling smaller and having to reach far away to be able to interact with the input screen and buttons. Another big concern was the fear of tapping or hitting objects in the real world while trying to reach for the buttons. One proposed solution to this issue was getting a longer cable for the headset and using the application standing up rather than in a sitting position in front of a desk. This would help to reduce the number of real-world obstacles around the user. Lastly, users mention feeling little to no cybersickness which might be due to being seated in place and the limited turning and locomotion ability.

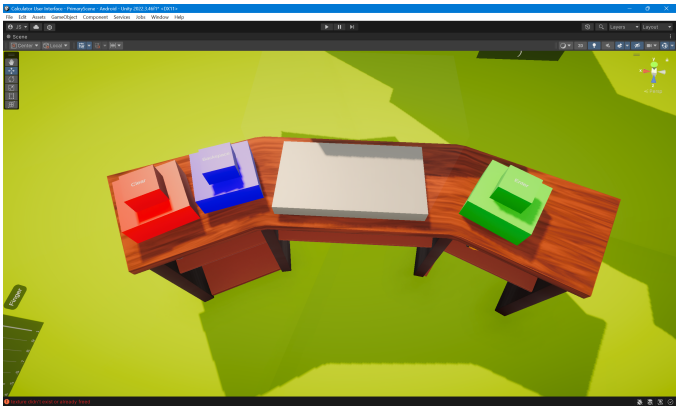

Figure 5: The input space and buttons

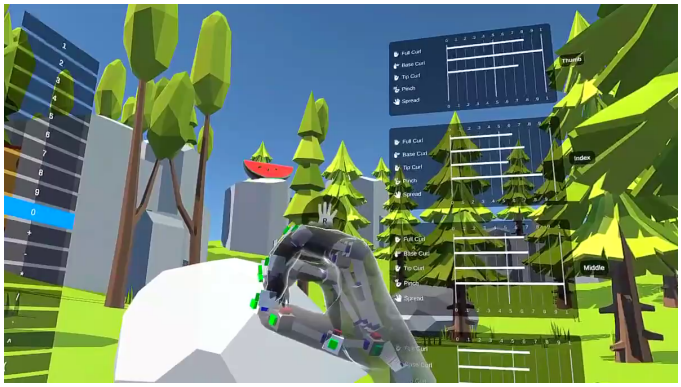

Figure 6: Hand gesture for zero, with debug info in background

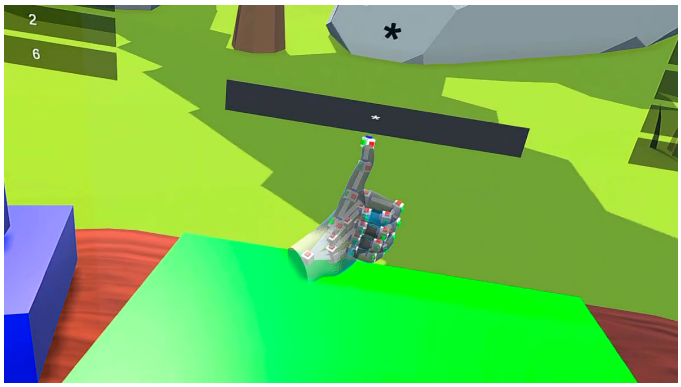

Figure 7: Inputting a hand gesture in the input space

4.0.2 User Interface. The users mention that the provided diagrams for the operators and the ASL digit representations are helpful and do not look out of place. One weakness pointed out by our user testing is that the calculator input text looks squished. One solution to this would be to make our calculator screen bigger, dynamically change the size of the input area, and allow for wrapping the text. Users also note the difficulty of getting the signs to be recognized inside the input space, which leads to frustration regarding the efficiency of the calculator. This highlights the need for increased accuracy and testing regarding detection inside our input space.

Another concern was about the placement of the detected text. Currently, the text is floating above the calculator screen without a background and the users note that it would be more appropriate if the text would have its own UI element.

4.0.3 Hand Gesture Input. Each participant was directed to do a few basic calculations with the calculator. First, a simple calculation was asked to "multiply six by three". Then, after completing that task, they were asked to do "six plus nine divided by five, all to the power of three," which required them to use multiple operations and parentheses. After completing these two tasks, participants were then allowed to interact with the calculator in any way they wished. Figures 6 and 7 show hand gesture input information and display area.

Both participants very quickly picked up on the proper hand gestures for each input. Regardless of their previous knowledge of sign language, each user was able to quickly identify the correct gestures they were expected to make using the charts provided behind the calculator.

However, even though it was easy to identify the hand gestures they were expected to create, both participants still had some difficulties actually inputting the gestures. First of all, both participants required additional clarification on where the input space was, as it was not immediately clear that they were required to make their hand gestures above the gray rectangle in the center of the desk. Without some form of tutorial or additional indicators on the input space, we were required to verbally explain the input space. Once they understood the input space, both users had a much easier time interacting with the calculator. However, there were still occasional issues with an input not being detected by the input space. While the gestures were being accurately identified, the input space did not always activate when the participants attempted to enter an input. This is because, in the current implementation, an input is only taken if the hand is pushed directly forward into the input space, and participants frequently attempted more lateral inputs. After being given a "pushing a drawer" analogy, the participants had a much easier time with making correct inputs. However, on rare occasions certain hand gestures still wouldn't be detected by the input space when they were held up at certain orientations.

The participant who had experience with sign language also noted that it felt unusual making signs facing towards herself rather than away from herself. This is due to the fact that usually when conversing in sign language, signs must be directed towards the person they are speaking to, meaning signs must be made facing away from themselves. However, the camera which is used to detect the hand gestures in our program is on the user's headset, meaning that in order to get the best detection of hand gestures, signs must be made towards oneself. In addition, she also noted that making inputs in the calculator felt very slow compared to how quickly she can makes ASL gestures in real life, largely due to the fact that she had to spend some time putting taking her hand in and out of the input space for every single input.

5 LIMITATIONS AND FUTURE WORK

One of the primary limitations of this project is the limited range of different hand gestures it is capable of differentiating between, especially when compared to the many possible hand gestures that exist in American Sign Language. For one, the program can only detect hand gestures that use one hand and is unable to detect any multi-hand gestures. It also is not capable of differentiating between gestures that use movement as a distinguishing trait. These two limitations are the reason that the final program does not use the ASL gestures for each mathematical operation, instead using custom-made static single-hand gestures, as the ASL gestures for each operation require multiple hands, and in some cases, hand movement. Since we used custom-made hand gestures, this means that even those who are familiar with ASL are required to learn new gestures in order to operate the calculator.

While this approach of creating unique control gestures for each program still has plenty of potential use cases, the implementation of multi-hand and movement-based gestures should be a primary focus in future iterations of this program. Multi-hand and movement-based gestures would greatly increase the number of potential gestures that can be used, including a greater number of gestures that are comfortable for an average user to make with their hands. It will also be far easier for each gesture to feel notably distinct from the others, making it easier for new users to learn. Importantly, this would also allow for a significantly larger number of ASL hand gestures to be detected, allowing for programs that completely use ASL hand gestures, thus allowing any users with previous knowledge of ASL a far easier time learning the controls, as well as allowing users unfamiliar with ASL to learn new ASL gestures by using the program.

In addition, the accuracy and intuitiveness of the input space could be greatly increased. In the current implementation, users frequently had issues with getting inputs detected as the the program only registered an input if the user pushed their hand directly forward over the input space. In addition, even when pushing their hand directly forward, some hand gestures still only activated the input space when their hands were in specific orientations. Due to these issues, future implementations should focus on redesigning the input space to be both more intuitive and more consistent. Ideally, a hand entering the input space should cause an input to register, regardless of the direction or orientation of the hand.

Another notable limitation comes in the complexity of the calculator that was implemented. Due to the focus on hand gesture-based input, less time was spent on increasing the complexity of the calculator, so the calculator was ultimately limited to the basic operations of addition, subtraction, multiplication, division, and exponentiation, along with support for decimal numbers and parentheses. In addition, the calculator can only be run as a standalone program, so it can't be used in conjunction with something else. As such, it is unlikely to be actually useful as a calculator. However, the calculator has shown the potential of hand gesture-based input, and there is no reason to believe that this input method could not be used in more complex and flexible applications. For this reason, future research and development should focus on the creation of more powerful programs that can be more easily applied in a large variety of situations. For example, updating this calculator program to be able to be opened and used in any virtual environment would greatly increase the practicality of the program.

Another very notable limitation of our implementation comes with the speed at which symbols can be inputted. While the program shows potential to save space due to the lack of a requirement for individual buttons for each symbol, a button-based input system still ultimately allows for faster input, largely due to the requirement for the user's hand to exit and reenter the input space every time they wish to input a new symbol. This was done because it would otherwise be very easy for the user to accidentally make incorrect inputs, as the program may detect an additional gesture as they are switching between gestures. In addition, there was no way to detect when the user was inputting the same symbol two or more times in a row. As such, another very important area of improvement that should be focused on in future work is a way to input separate hand gestures without requiring the hand to exit and reenter the input space, or some other similarly time-consuming mediator. This is likely the most difficult of the potential improvements discussed here, but if it could be done, it would allow users to make inputs just as fast as someone can converse in ASL, and further lower the number of differences between our hand gesture-based input system and ASL.

6 CONTRIBUTIONS

6.1 Ekin

- Configured XR Hands gestures
- Configured environment for XR hand tracking and gesture detection
- Debugged errors in the program
- Designed and programmed user input detection
- Implemented buttons for calculator operations
- Wrote the Abstract, Review of Previous Work, Methodology, Evaluation, and Contributions sections

6.2 Adam

- Designed and programmed calculator back-end operations
- Designed background environment
- Designed and programmed user input detection
- Designed calculator user interface
- Debugged errors in the program
- Designed hand gestures for mathematical operations
- Implemented buttons for calculator operations
- Wrote the Introduction, Review of Previous Work, Methodology, Evaluation, Contributions, and Limitations sections

6.3 Jacob

- Designed background environment
- Designed calculator user interface
- Designed and programmed user input detection
- Debugged errors in the program
- Designed hand gestures for mathematical operations
- Implemented buttons for calculator operations
- Did the user testing
- Wrote the Abstracts and Evaluation sections
- Took pictures and videos of the project

REFERENCES

[1] Jeremy Becnel. 2017. CalcVR (Calculus in virtual reality). https://calcvr.org/

[2] Gavin Buckingham. 2021. Hand Tracking for Immersive Virtual Reality: Opportunities and Challenges. *Frontiers in Virtual Reality* 2 (2021). https://doi.org/10.3389/frvir.2021.728461

[3] GameDev.tv Team. 2024. Make a Fully Functional Calculator in Unity (Not Only for VR) – Part III. https://www.gamedev.tv/articles/make-a-fully-functional-calculator-in-unity-not-only-for-vr-part-iii [Accessed: 10-Dec-2024].

[4] S. Geetha, Aditya G, Chetan Reddy M, and Nischith G. 2024. Human Interaction in Virtual and Mixed Reality Through Hand Tracking. In *2024 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*. 1–6. https://doi.org/10.1109/CONECCT62155.2024.10677239

[5] Aamrah Ikram and Yue Liu. 2021. Real Time Hand Gesture Recognition Using Leap Motion Controller Based on CNN-SVM Architechture. In *2021 IEEE 7th International Conference on Virtual Reality (ICVR)*. 5–9. https://doi.org/10.1109/ICVR51878.2021.9483844

[6] Hee-Deok Yang. 2015. Sign Language Recognition with the Kinect Sensor Based on Conditional Random Fields. *Sensors* 15, 1 (2015), 135–147. https://doi.org/10.3390/s150100135