EVALUATION AND STUDY OF USER INTERFACE WITH WEARABLE DEVICES BASED ON COMPUTER VISION - MODEL BASED ON VR KEYBOARD AND HAND INTERACTION

Lappeenranta–Lahti University of Technology LUT

Bachelor's thesis, Software and System Engineering

2023

Author: Jing Fan

Primary Supervisor: Doc. Dr. Damian Kedziora

Secondary Supervisor: Professor Ming Yu

ABSTRACT

Lappeenranta–Lahti University of Technology LUT

Software and System Engineering

Jing Fan

Evaluation and study of user interface with wearable devices based on computer vision

Bachelor's thesis

2023

52 pages and 9 figures

Primary Examiner: Doc. Dr. Damian Kedziora

Secondary Examiner: Professor Ming Yu

Keywords: wearable devices, computer vision, fingertip interaction, mathematical model,

Abstract:
This project was initiated with the goal of significantly enhancing the user experience in virtual reality (VR) environments through the development of an advanced mathematical model for camera-based fingertip tracking. The inception of this project involved an exhaustive literature review, which not only established the feasibility of the concept but also provided critical insights into existing methodologies and technological limitations. Building upon this foundational knowledge, the model was meticulously developed using MATLAB, a choice driven by its robust computational capabilities and suitability for algorithmic modeling. The core of the model centered around the precise measurement of distance between the camera and the fingertips, employing various triangular theorems to accurately triangulate the position of the fingertips in the virtual space. This innovative approach enabled the algorithm to effectively map and position fingertips in the VR environment, thereby creating a more interactive and immersive experience for users. The algorithm's efficiency in tracking and interpreting complex finger movements marked a notable advancement in the field of interactive VR technology. Despite its success, the project identified key limitations that posed challenges to its effectiveness. One of the primary issues was the resolution of the cameras used; the fidelity of fingertip tracking is heavily dependent on the camera's ability to capture fine details and subtle movements, which is constrained by its resolution. Additionally, the accuracy of the hand model used in the algorithm was a limiting factor. The model's ability to precisely interpret and replicate finger movements was intrinsically tied to the realism and detail of the hand

model. These limitations, while highlighting the current constraints of the technology, also pave the way for future research and development. Addressing these issues will be crucial in advancing the technology further, allowing for greater precision in tracking and an even more immersive VR experience. Future enhancements could involve integrating higher resolution cameras, refining the hand model for increased accuracy, and exploring the use of machine learning techniques to improve the algorithm's adaptability and responsiveness to a wider range of hand shapes and movements. The long-term implications of these improvements could be substantial, potentially revolutionizing the way we interact with virtual environments and expanding the applications of VR technology in various fields such as education, medicine, and entertainment.

Abbreviations

ToF	Time-of-Flight
VR	Virtual Reality
OpenCV	Open-Source Computer Vision Library
MCDCNN	Multi-Channels Deep Convolutional Neural Networks
DHG	Dynamic Hand Gesture
RGB	Red Green and Blue
HCI	Human Computer Interaction
YOLO v3	You Only Look Once, version 3.
STr-GCN	Spatial-Temporal Graph Convolutional Network
ICP	Iterative Closest Point
LiDAR	Light Detection and Ranging

Table of contents

1. Introduction

1.1 Background

Virtual Reality (VR) technology is increasingly vital in various sectors, offering unparalleled immersive experiences. In gaming, VR transports players into vivid, interactive worlds, enhancing engagement and realism. For example, in a VR game, players can physically turn their heads to look around, and use hand controllers as virtual weapons or tools, providing a level of interaction traditional gaming cannot. In the realm of video and audio entertainment, VR brings a new dimension to media consumption. Viewers can experience concerts or movies as if they were physically present, offering a 360-degree view and spatial audio that traditional screens can't provide. VR's impact in education and training is profound, particularly in high-stakes fields like medicine, aviation, and the military. Medical students can practice complex surgeries in VR without risk, gaining experience before handling real-life cases. Pilots use VR for flight simulations, offering realistic training environments without the high costs and risks associated with actual flights. Similarly, military training utilizes VR for safe, cost-effective combat simulations. In education, VR transforms learning experiences, making them more engaging and interactive. For instance, history students can virtually visit historical sites or witness historical events, while science students can explore the human body or outer space in a more interactive way. In business and industry, VR aids in product design and visualization. Designers and engineers use VR to create and test models in a virtual environment, significantly reducing the time and resources needed for physical prototyping. For remote work, VR enables virtual meetings where participants feel like they're in the same room, fostering better collaboration and communication. Socially, VR introduces a novel way of interaction. Virtual social spaces allow people to meet and interact in a more immersive way than traditional online platforms, potentially revolutionizing how we socialize in the digital age. Overall, VR's diverse applications are transforming industries, enhancing experiences, and paving the way for future technological advancements.

Keyboard interaction is an integral aspect of user experience in Virtual Reality (VR) environments. The evolution of hand keyboard interaction in VR signifies a remarkable advancement in VR technology, focusing on how users input data and interact with virtual settings using keyboard-like interfaces or hand gestures. This progression is pivotal in enhancing the naturalism and intuitiveness of VR interactions, especially for activities necessitating precise typing or detailed hand movements. Initial VR Interaction Paradigms: Early VR systems primarily depended on conventional input devices such as joysticks, buttons, and specialized controllers. While functional for specific applications, these interfaces lacked the natural feel necessary for a fully immersive VR experience, particularly in tasks that required typing or intricate hand movements. As VR technology evolved, there was an increasing focus on creating more natural interaction within VR spaces. This shift was driven by the desire to enhance user immersion and ease of use, thereby making VR experiences more accessible and engaging. The development of hand tracking technologies marked a significant milestone in VR interaction. This technology allowed users to use their hands directly as input devices, bridging the gap between physical and virtual interaction.

Types of current hand tracking technology:

Optical Hand Tracking: Utilizing cameras, this technology captures the position and movement of the user's hands in real-time. It represents a significant leap in VR technology, offering a more natural and intuitive way of interacting with VR environments. Gesture Recognition: Alongside hand tracking, the ability to recognize and interpret hand gestures became crucial. This allowed users to interact with VR environments using natural hand movements and gestures, moving away from reliance on physical controllers.

However, there is currently no complete model for keyboard and fingertip interaction.

The proposed model presents an advanced method for virtual keyboard interaction integrating technologies such as optical hand tracking, openCV and Mediapipe. We would employ a rotating camera to precisely track fingertip positions. Designed for deployment on real-life flat surfaces, the system necessitates an initial spatial scan to discern and establish the planes of these surfaces. Once these planes are identified, the virtual keyboard is strategically anchored adjacent to the plane's edge, thereby providing a consistent reference point.

This design ensures that the virtual keyboard remains constantly visible in the user's field of vision, even when minor movements or rotations of the VR glasses occur, thus offering a stable and intuitive interface for typing and interaction. Importantly, the incorporation of a physical flat surface significantly enhances the user's typing experience within the virtual environment, lending a more tangible and realistic sensation to the act of typing, akin to the tactile feedback of traditional keyboards.

The fingertip tracking technology introduced here holds significant potential for application in various virtual reality (VR) office software, poised to substantially enhance the user experience in office settings. This technology allows users to virtually access a flat keyboard in any location, thereby markedly increasing the functionality and convenience of VR products. As VR devices evolve towards greater portability, this innovation paves the way for users to seamlessly integrate VR glasses into their work and leisure activities.

The flexibility to adapt the virtual environment to any setting, combined with the realistic touch sensation of typing, is expected to significantly elevate the overall user experience. Furthermore, this advancement contributes to the practicality of VR technology. Looking ahead, there is a strong potential for VR to emerge as a portable device, integrating seamlessly with smartphones and computers, and thereby revolutionizing how we interact with technology in our daily lives.

1.2 Objectives and goals

The primary aim of this research is to develop an advanced model that enhances keyboard and fingertip positioning within virtual reality (VR) environments. This model is particularly designed for integration with virtual office software requiring a virtual keyboard interface, aiming to streamline and improve the user experience.

The implementation process of this model encompasses several critical phases:

Hand Scanning: A camera is employed to conduct a comprehensive 360° scan of the user's hand. This scan meticulously captures the hand's shape and size from multiple angles, leveraging basic geometric principles alongside sophisticated image processing techniques. The system then utilizes the perceived dimensions of the hand to estimate its distance from the camera, a key factor in accurate spatial positioning.

Distance Measurement and Keyboard Positioning: The VR device scans the immediate environment and strategically positions the virtual keyboard based on the user-selected plane. This includes fixing the keyboard's location in relation to reference points such as the plane's edge. Subsequently, the camera calculates the vertical distance between the plane of the virtual keyboard and its position, which is essential for accurately situating the keyboard within the virtual space.

Fingertip Detection: Utilizing mathematical theories, notably Pythagoras' theorem, the system determines if the user's fingertip is within the operational range of the virtual keys and in contact with a designated flat surface. This precise detection activates the appropriate key response in the VR setting.

The successful execution of this model is expected to establish a robust mathematical foundation for VR devices, significantly aiding in the evolution and enhancement of virtual keyboard systems. This advancement is anticipated to markedly refine user interactions with a variety of virtual office tools, fostering a more natural and efficient interface.

1.3 Structure of the thesis

This article adopts the most direct academic paper structure, which includes the following sections:

Literature Review:

The literature cited in this article plays a crucial role in supplementing and refining my mathematical model. The referenced studies are highly valuable and offer insights that are worthy of emulation. This section will delve into existing research, highlighting how these works have contributed to the development of the mathematical model used in this study.

Methodology:

This section outlines the methods employed for data collection and analysis. It describes the systematic approach taken to gather relevant data, including the selection of sources, criteria for data inclusion, and the analytical tools and techniques used. The methodology section is critical in establishing the validity and reliability of the research findings.

Results:

Here, this article explains how to display research results. This section details the results of the mathematical model and presents them in a clear and concise manner. Emphasis will be placed on the use of spatial stereograms to illustrate key points, ensuring that results are easy to understand and interpret.

Discussion:

This part of the article will focus on explaining the results and their implications. It involves critically analyzing the findings, discussing their significance in the context of a mathematical model, and how they are consistent or different from previous research. The discussion is intended to provide a deeper understanding of the results and to gain insight into their significance within the broader field of research.

2. Related research

2.1 Historical development of wearable devices and their user interfaces

Skeleton-based gesture recognition has evolved into a dynamic and prominent research area, particularly in the last few years, largely fueled by advancements in deep learning technologies. This field focuses on analyzing the skeletal structure of the human body to recognize and interpret various gestures. The surge in research and development in this domain has resulted in the creation of numerous sophisticated methods for gesture recognition [1], [2], [3], [4], [5], [6], [7], [8], [9].

In the evolving field of hand gesture recognition, several significant advancements have been made in recent years, each addressing unique challenges and expanding the capabilities of gesture-based interaction systems.

In 2018, a novel approach was introduced, enhancing the MC-DCNN framework with a convolutional neural network specifically designed for classifying hand gestures using skeletal data. This method incorporated parallel processing branches, residual connections, and dropout regularization, achieving substantial improvements in classification accuracy on the DHG dataset for both 14 and 28 gesture classes [10].

The following year, 2019, saw the development of a real-time hand gesture identification system utilizing a Kinect sensor. This system focused on acquiring skeleton index data and hand images, using a skin color model for hand segmentation and calculating hand and finger positions for gesture detection. Tested with ten volunteers across six gesture instructions, it demonstrated a high average recognition accuracy of 94.5% for dynamic motion instruction identification [11].

In 2020, the focus shifted to the automotive sector with the introduction of a multi-modal dataset for dynamic recognition of driver micro hand gestures, recorded using RGB, infrared, and depth data modalities. An HCI system was proposed for online gesture recognition, aiming to enhance safety and efficiency in automotive applications. Despite high offline accuracy, the need for further improvement in real-time recognition strategies was identified [12].

That same year, a comprehensive review on hand gesture recognition based on computer vision was published. This review highlighted the importance of developing reliable and robust algorithms, especially considering the specific challenges posed by different camera sensors, to ensure effective and efficient gesture recognition systems [13].

In 2021, a new Multiview gesture recognition network was proposed, utilizing multi-view images and supervised fusion for gesture classification. This network achieved a remarkable 94.71% accuracy with four-view input streams, indicating its effectiveness in addressing occlusion problems and suggesting potential areas for future advancements [14]. A study introduces a hypergraph neural network (Hyper-GNN) for skeleton-based human action recognition in computer vision, focusing on capturing spatiotemporal information and high-order dependencies between joints, including those without physical connections joint. Leveraging a three-stream Hyper-GNN fusion architecture augmented by hypergraph attention and an improved residual module, this approach outperforms existing state-of-the-art skeleton-based methods when tested on two benchmark datasets [15].

By 2022, the field had progressed to include deep learning-based systems for human sign language gesture recognition, with one such system employing a Raspberry Pi-connected monocular camera for image acquisition and processing, along with an improved YOLO v3 for gesture recognition. This system, particularly useful in assisting hearing-impaired individuals and smart home applications, demonstrated effective recognition under various conditions [16].

Also in 2022, a method was introduced for continuous hand gesture recognition using skeleton information. This method showed promising results in both isolated and continuous gesture recognition, outperforming other skeleton-based, RGB, and optical flow methods, and opened avenues for future exploration into end-to-end networks [17].

Most recently, in 2023, the introduction of STr-GCN marked another leap forward. This deep learning architecture for 3D skeleton-based hand gesture recognition combined Spatial Graph Convolutional Network and Transformer Graph Encoder, proving its robustness across various datasets and contexts. Future work is expected to focus on enhancing online hand gesture recognition and exploring multi-modal approaches [18].

These advancements collectively reflect the rapid progression and diverse applications of hand gesture recognition technologies, underscoring their growing importance in human-computer interaction.

Based on established hand models, these models can be effectively leveraged to perform advanced fingertip localization and optimization. This approach draws on the foundation laid by previous work in the field, leveraging detailed hand models to improve the accuracy and efficiency of fingertip tracking and motion analysis. The application of these models not only extends their usefulness but also contributes to the continued advancement of accuracy in gesture recognition technology.

2.2 Related research of wearable devices and their user interfaces

2.2.1 360-Degree Photogrammetric Scanning

This study details a workflow for underground tunnel digitization using a 360-degree camera and structure-from-motion multi-view stereo photogrammetry, demonstrating that the method is efficient and accurate for creating 3D models of tunnels. This technology not only reduces image acquisition time, but also achieves high resolution and accuracy in modeling and achieves significant improvements when adding high-resolution images. The study also explores the use of the model for rock mass property mapping, with results correlating closely with traditional field measurements and recommending future testing in more challenging tunnel environments. Relying on the results of this article, we can lay the foundation for 360° scanning of the hand and localization of hand shape and orientation [19].

2.2.2 Photogrammetric Distance Estimation

This paper provides a comprehensive review and performance analysis of existing point cloud registration algorithms, focusing on coarse registration using sparse feature-based correspondences and fine registration via C2C (cloud-to-cloud) registration. The study tested various methods on different datasets and concluded that deep learning-based methods are effective for coarse registration but face generalization challenges, while fast ICP and robust ICP due to their accuracy The balance between robustness and robustness makes it the first choice for fine registration. Based on these findings, this paper recommends specific algorithm combinations for different scenarios, including arbitrary datasets, datasets with well-estimated initial registration parameters, and real-time 3D mapping tasks. Referring to the technology in this article, you can perform 3D mapping and analyze data such as the vertical distance from the camera to the plane based on real-time coordinates, which will help improve the accuracy of our mathematical models in the future [20].

2.2.3 Edge Detection in Computer Vision

This research paper proposes a 3D localization method for virtual reality (VR) applications in indoor environments. It utilizes light detection and ranging (LiDAR) sensors to create 3D point clouds of the real world and applies the Hough transform algorithm for boundary wall detection. This method scales the virtual environment to match the user's real environment, enhancing the user experience by preventing collisions with real-world objects. In addition, the paper also proposes a wireless network composed of multiple Kinect sensors to fully track the user's position and gestures. This setup allows for accurate, real-time user positioning and interaction in VR environments, as demonstrated by the development of VR games. The research aims to improve the VR experience by ensuring VR synchronization and free movement within the VR space. By analyzing the real world, objects in the real world can be better utilized to improve the user's sense of use [21].

2.3 Wearable devices and their limitations with hand interaction

This paper by Rolands Kromanis and Prakash Kripakaran proposes a method for accurate displacement measurement in bridge monitoring using multiple camera positions and computer vision. The study focuses on utilizing images from different camera positions and angles, typically captured during routine bridge inspections, and transforming these images into a common coordinate system for structural analysis. The method is based on the principle that spatial relationships between structural features remain consistent even when captured from varying angles. The authors demonstrate the effectiveness of their approach through laboratory experiments on a beam and field tests on a full-scale pedestrian suspension bridge. The results show accurate displacement measurements with discrepancies of around 5-6% compared to the mean, validating the feasibility of this approach for condition assessment in civil infrastructure monitoring [22]. However, utilizing multiple cameras for measurement can have certain drawbacks. For example, multiple cameras have high storage requirements as more data is generated. Setup can also be difficult hence the flexibility of multiple cameras approach is lower compared to single camera systems.

Hong An Pham's bachelor's thesis, titled "The Challenge of Hand Gesture Interaction in the Virtual Reality Environment," evaluates in-air hand gestures using the Leap Motion Controller. This research, conducted at Oulu University of Applied Sciences, explores the usability and challenges of hand gesture interactions in VR. The study involves user tests in a VR environment with an Oculus headset and Leap Motion controller, focusing on the human-object relationship in 3D environments. The thesis identifies key challenges in VR hand gesture interaction, such as imprecise operation, and aims to understand the factors affecting user experience in VR [23].

The Microsoft Research blog post on "Typing in the Virtual Office Environment: Examining the Effects of Keyboard and Hand Representation in VR" discusses the challenges and innovations in VR keyboard development. The research focuses on enhancing VR office work like editing documents or sending messages. The study found that while typing skills transfer to VR, typing speed decreases due to the novelty of VR and display limitations. The team experimented with different hand representations and keyboard positions in VR, noting that the minimalistic model of transparent hands with visible fingertips was surprisingly effective. Future systems may track hands from the headset or keyboard, utilizing inside-out tracking for mobile VR office environments. The research aims to engage the community and build upon these findings, exploring new conditions and VR technology advancements [24].

3. Research methodology

3.1 Requirement Analysis

The choice of a camera for a computer vision system can be affected by several factors. For example. Pixal density, field of view, lens type, frame rate, low light performance and environmental considerations are some of the important factors. As hand can be recognised as smaller items, high pixal density is prefered. As fingers may move fast while typing, a high frame rate, for example,15frames per second is prefered. High low light performance is also designed since the system should be able to function in almost all scenarios [25].

The rotating range of the camera highly deponds on the field of view of the camera. The idea is to make projected keyboard always within the range of the camera.

VR devices with positional tracking capabilities are designed to detect the position and orientation of the user within a virtual environment. They use various tracking technologies to achieve this, allowing users to move and interact with virtual objects. Here are some examples of VR devices with positional tracking and their principles of detection:

Oculus Rift: Principle of Detection: Oculus Rift uses external sensors called "Constellation" sensors or built-in sensors, such as accelerometers and gyroscopes, to track the movement of the headset in 3D space. The external sensors emit infrared light, which is captured by the headset's sensors to determine its position and orientation.

HTC Vive: Principle of Detection: HTC Vive relies on a system of external base stations that emit lasers and sensors on the Vive headset and controllers. These lasers sweep across the room, and when they hit the sensors on the headset and controllers, the system can calculate their precise position and orientation in real-time.

PlayStation VR (PSVR): Principle of Detection: PSVR uses a combination of external PlayStation Camera and built-in sensors in the headset and PlayStation Move controllers. The camera tracks the position of LEDs on the headset and controllers, allowing it to determine their position and orientation within the tracked area.

Valve Index: Principle of Detection: Valve Index employs external base stations similar to the HTC Vive, emitting lasers that are picked up by sensors on the headset and controllers. These lasers provide sub-millimeter tracking accuracy for precise positioning and orientation detection.

Oculus Quest: Principle of Detection: Oculus Quest utilizes built-in sensors, including accelerometers and gyroscopes, to track the headset's movement in 3D space. It also has built-in cameras for inside-out tracking, allowing it to map the environment and detect the user's position without external sensors.

These VR devices employ a combination of sensors, cameras, and external tracking systems to continuously monitor the user's movements and provide an immersive and interactive virtual reality experience. The specific tracking technology and methods may vary, but they all aim to accurately detect the user's position and orientation within the virtual environment.

Identify the specific requirements and constraints for the VR keyboard and fingertip positioning model. This may involve understanding user needs, hardware limitations, and software capabilities.

3.2 Designing the Model

3.2.1 A Time-of-Flight (ToF) camera

A Time-of-Flight (ToF) camera measures distance by calculating the time it takes for a light signal (often infrared) to travel from the camera to the subject and back again. The basic equation to calculate distance using a ToF camera is derived from the speed of light and the measured time of flight.

The equation is:

Distance = (Speed of Light × Time of Flight) / 2

Where:

- Distance is the distance from the camera to the subject.
- Speed of Light is approximately 3 ×10^8 meters/second.
- Time of Flight is the round-trip time it takes for the light to travel from the camera to the subject and back.

It's important to note that the time of flight is divided by 2 because the light travels to the subject and back, so you need only the one-way travel time to calculate the distance.

3.2.2 Reference coordinate positioning

Locate a random non-moving target on the table, such as the corner of the table. And model them through MATLAB.

```
% Create a new figure window
figure;

% Define the range of the 3D space
xlim([0 10]);
ylim([0 10]);
zlim([0 10]);

% Enable grid for better observation
grid on;
hold on;
xlabel('X-axis');
ylabel('Y-axis');
zlabel('Z-axis');
title('Keyboard, Camera Position, Finger Touch, Desktop and Control Points in 3D Space');

% Draw a large rectangle to represent the desktop
rectangle('Position',[1 1 8 8], 'FaceColor', [0.6 0.6 0.6], 'EdgeColor', 'none');

% Draw a simple keyboard shape on the xy-plane (on top of the desktop)
```

```
rectangle('Position',[2 2 6 3], 'FaceColor', [0.8 0.8 0.8], 'EdgeColor', 'b');
text(5, 3.5, 'Keyboard', 'HorizontalAlignment', 'center');

% Define and plot the camera position
cameraPos = [5 5 8]; % x, y, z coordinates
plot3(cameraPos(1), cameraPos(2), cameraPos(3), 'ro', 'MarkerSize', 10, 'MarkerFaceColor', 'r');
text(cameraPos(1), cameraPos(2), cameraPos(3)+0.5, 'Camera', 'HorizontalAlignment', 'center');

% Define and plot the finger touch point on the keyboard
fingerTouchPos = [4 3 0]; % x, y, z coordinates (z is 0 because it's on the keyboard)
plot3(fingerTouchPos(1), fingerTouchPos(2), fingerTouchPos(3), 'go', 'MarkerSize', 10, 'MarkerFaceColor', 'g');
text(fingerTouchPos(1), fingerTouchPos(2), fingerTouchPos(3)+0.5, 'Finger Touch', 'HorizontalAlignment', 'center');

% Mark reference point 1 (1, 1, 0) on the desktop
plot3(1, 1, 0, 'ko', 'MarkerSize', 5, 'MarkerFaceColor', 'k');
text(1, 1, 0.5, 'Ref Point 1', 'HorizontalAlignment', 'center');

% Mark reference point 2 (9, 1, 0) on the desktop
plot3(9, 1, 0, 'ko', 'MarkerSize', 5, 'MarkerFaceColor', 'k');
text(9, 1, 0.5, 'Ref Point 2', 'HorizontalAlignment', 'center');
```

```
% Mark control point 1 (2, 2, 0) on the keyboard

plot3(2, 2, 0, 'mo', 'MarkerSize', 5, 'MarkerFaceColor', 'm');

text(2, 2, 0.5, 'Control Point 1', 'HorizontalAlignment', 'center');


% Mark control point 2 (8, 2, 0) on the keyboard

plot3(8, 2, 0, 'mo', 'MarkerSize', 5, 'MarkerFaceColor', 'm');

text(8, 2, 0.5, 'Control Point 2', 'HorizontalAlignment', 'center');


% Draw a line connecting the reference point1 and the camera position

line([1, cameraPos(1)], [1, cameraPos(2)], [0, cameraPos(3)], 'Color', 'k', 'LineStyle', '--');

text((1+cameraPos(1))/2, (1+cameraPos(2))/2, (0+cameraPos(3))/2, 'A1', 'HorizontalAlignment', 'center', 'BackgroundColor', 'white');


% Draw a line connecting the control point1 and the camera position

line([2, cameraPos(1)], [2, cameraPos(2)], [0, cameraPos(3)], 'Color', 'k', 'LineStyle', '--');

text((2+cameraPos(1))/2, (2+cameraPos(2))/2, (0+cameraPos(3))/2, 'A2', 'HorizontalAlignment', 'center', 'BackgroundColor', 'white');


% Draw a line connecting the reference point1 and control point1

line([1, 2], [1,2], [0, 0], 'Color', 'k', 'LineStyle', '--');

text((1+2)/2, (1+2)/2, (0+0)/2, 'A3', 'HorizontalAlignment', 'center', 'BackgroundColor', 'white');


% Draw a line connecting the reference point2 and the camera position
```

line([9, cameraPos(1)], [1, cameraPos(2)], [0, cameraPos(3)], 'Color', 'k', 'LineStyle', '--');

text((9+cameraPos(1))/2, (1+cameraPos(2))/2, (0+cameraPos(3))/2, 'B1', 'HorizontalAlignment', 'center', 'BackgroundColor', 'white');

% Draw a line connecting the control point2 and the camera position

line([8, cameraPos(1)], [2, cameraPos(2)], [0, cameraPos(3)], 'Color', 'k', 'LineStyle', '--');

text((8+cameraPos(1))/2, (2+cameraPos(2))/2, (0+cameraPos(3))/2, 'B2', 'HorizontalAlignment', 'center', 'BackgroundColor', 'white');

% Draw a line connecting the reference point2 and control point2

line([9, 8], [1,2], [0, 0], 'Color', 'k', 'LineStyle', '--');

text((9+8)/2, (1+2)/2, (0+0)/2, 'B3', 'HorizontalAlignment', 'center', 'BackgroundColor', 'white');

% Adjust the view for better observation

view(3);

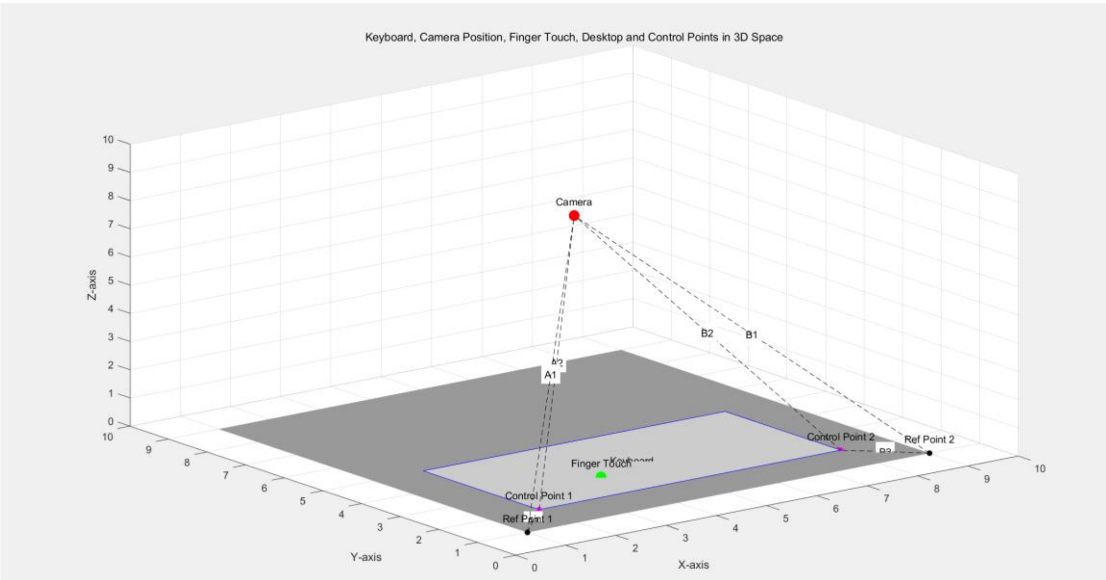

Figure 1

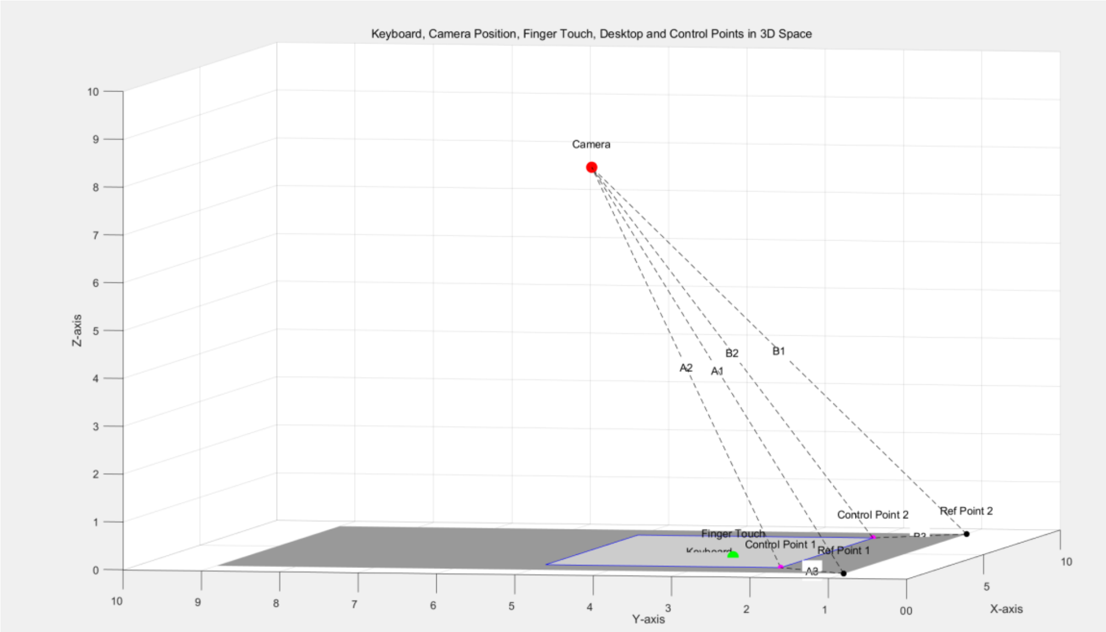

Figure 2

As shown in the figure legend, we use two table corners in the known physical space as reference points to model the virtual keyboard.

∠(A1, camera, A2) and∠(B1, camera, B2) can be captured through the camera space.

A3 = ((A1)^2 + (A2)^2 – 2 * A1 * A2 * cos∠(A1, camera, A2))^(-2)

B3 = ((B1)^2 + (B2)^2 – 2 * B1 * B2 * cos∠(B1, camera, B2))^(-2)

After getting A3 and B3, we can determine the specific position of the keyboard on the desktop, so that no matter how the VR device moves slightly during office work, it can be firmly positioned on the desktop.

3.2.3 Fingertip z coordinate positioning

Verify the z values of the fingertip and model them through MATLAB.

% Create a new figure window

figure;

```
% Define the range of the 3D space
xlim([0 10]);
ylim([0 10]);
zlim([0 10]);

% Enable grid for better observation
grid on;
hold on;
xlabel('X-axis');
ylabel('Y-axis');
zlabel('Z-axis');
title('Keyboard, Camera Position, Finger Touch, and Right Angle in 3D Space');

% Draw a simple keyboard shape on the xy-plane
rectangle('Position',[2 2 6 3], 'FaceColor', [0.8 0.8 0.8], 'EdgeColor', 'b');
text(5, 3.5, 'Keyboard', 'HorizontalAlignment', 'center');

% Define and plot the camera position
cameraPos = [5 5 8]; % x, y, z coordinates
plot3(cameraPos(1), cameraPos(2), cameraPos(3), 'ro', 'MarkerSize', 10, 'MarkerFaceColor', 'r');
text(cameraPos(1), cameraPos(2), cameraPos(3)+0.5, 'Camera', 'HorizontalAlignment', 'center');

% Define and plot the finger touch point on the keyboard
```

```
fingerTouchPos = [4 3 0]; % x, y, z coordinates (z is 0 because it's on the keyboard)
plot3(fingerTouchPos(1), fingerTouchPos(2), fingerTouchPos(3), 'go', 'MarkerSize', 10, 'MarkerFaceColor', 'g');
text(fingerTouchPos(1), fingerTouchPos(2), fingerTouchPos(3)+0.5, 'Finger Touch', 'HorizontalAlignment', 'center');

% Draw a line connecting the finger touch point and the camera position
line([fingerTouchPos(1), cameraPos(1)], [fingerTouchPos(2), cameraPos(2)], [fingerTouchPos(3), cameraPos(3)], 'Color', 'k', 'LineStyle', '--');
text((fingerTouchPos(1)+cameraPos(1))/2, (fingerTouchPos(2)+cameraPos(2))/2, (fingerTouchPos(3)+cameraPos(3))/2, 'c', 'HorizontalAlignment', 'center', 'BackgroundColor', 'white');

% Draw a vertical line from the camera position to the xy-plane
cameraToPlanePos = [cameraPos(1), cameraPos(2), 0]; % Intersection point on xy-plane
plot3([cameraPos(1), cameraToPlanePos(1)], [cameraPos(2), cameraToPlanePos(2)], [cameraPos(3), cameraToPlanePos(3)], 'Color', 'b', 'LineStyle', '--');
text(cameraToPlanePos(1), cameraToPlanePos(2), cameraPos(3)/2, 'a', 'HorizontalAlignment', 'center', 'BackgroundColor', 'white');

% Label the intersection point
text(cameraToPlanePos(1), cameraToPlanePos(2), cameraToPlanePos(3), 'Vertical Intersection', 'VerticalAlignment', 'bottom', 'HorizontalAlignment', 'right');
```

```
% Connect the intersection point with the finger touch point
line([cameraToPlanePos(1), fingerTouchPos(1)], [cameraToPlanePos(2), fingerTouchPos(2)], [cameraToPlanePos(3), fingerTouchPos(3)], 'Color', 'm', 'LineStyle', '-.');
text((cameraToPlanePos(1)+fingerTouchPos(1))/2, (cameraToPlanePos(2)+fingerTouchPos(2))/2, (cameraToPlanePos(3)+fingerTouchPos(3))/2, 'b', 'HorizontalAlignment', 'center', 'BackgroundColor', 'white');

% Plot the right angle symbol on the plane
angleSize = 0.5; % Size of the right angle symbol arms
% Draw horizontal arm
plot3([cameraToPlanePos(1), cameraToPlanePos(1) + angleSize], [cameraToPlanePos(2), cameraToPlanePos(2)], [cameraToPlanePos(3), cameraToPlanePos(3)], 'Color', 'r', 'LineWidth', 2);
% Draw vertical arm in the plane formed by the camera, intersection point, and finger touch point
verticalArmEnd = [cameraToPlanePos(1), cameraToPlanePos(2) + angleSize, cameraToPlanePos(3)];
plot3([cameraToPlanePos(1), verticalArmEnd(1)], [cameraToPlanePos(2), verticalArmEnd(2)], [cameraToPlanePos(3), verticalArmEnd(3)], 'Color', 'r', 'LineWidth', 2);

% Adjust the view for better observation
view(3);
```

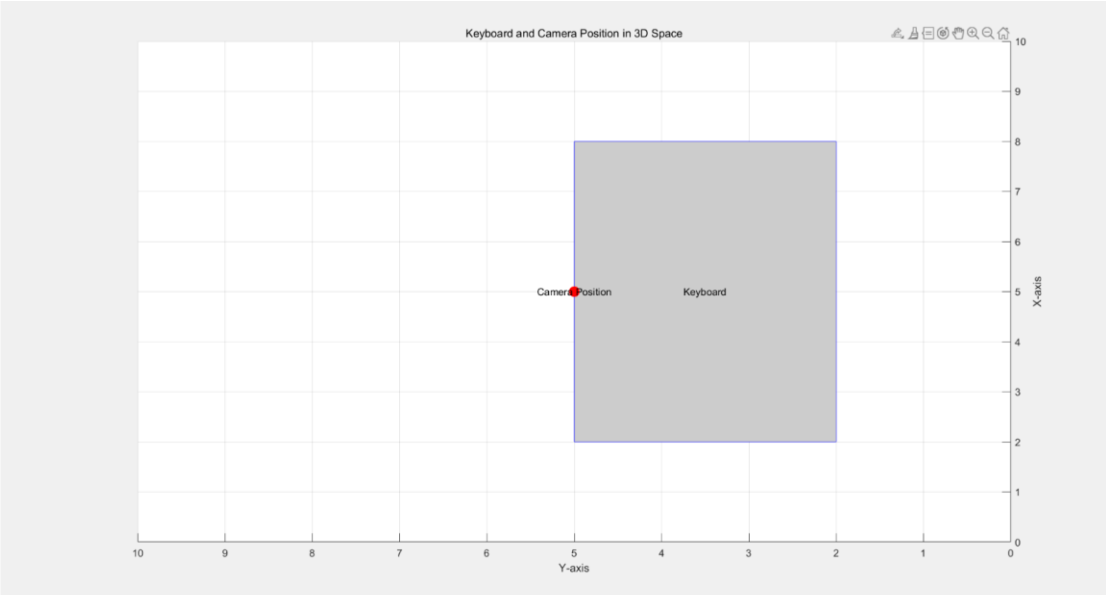

Figure 3

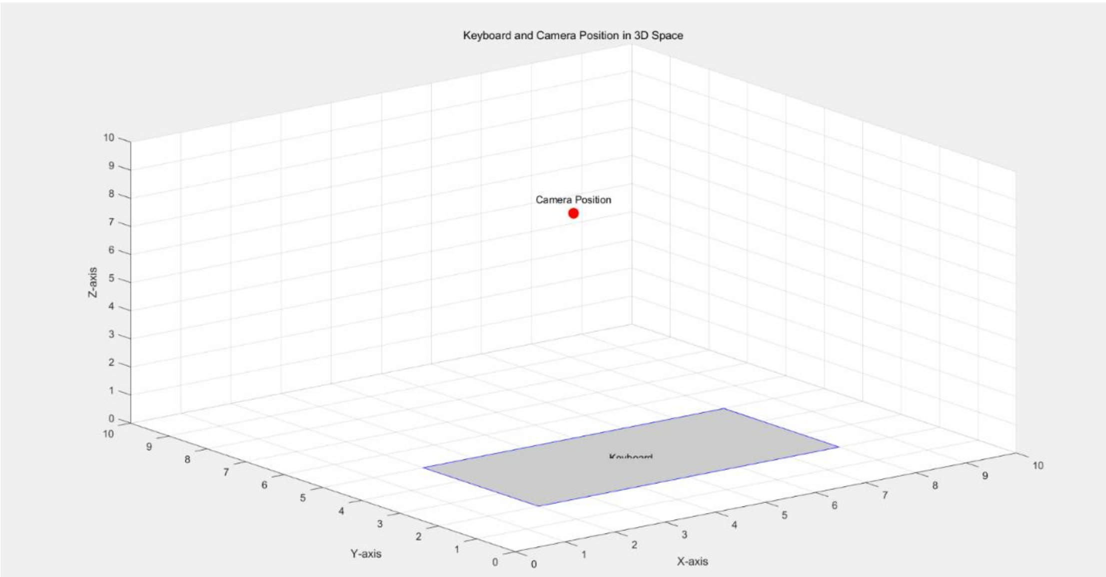

Figure 4

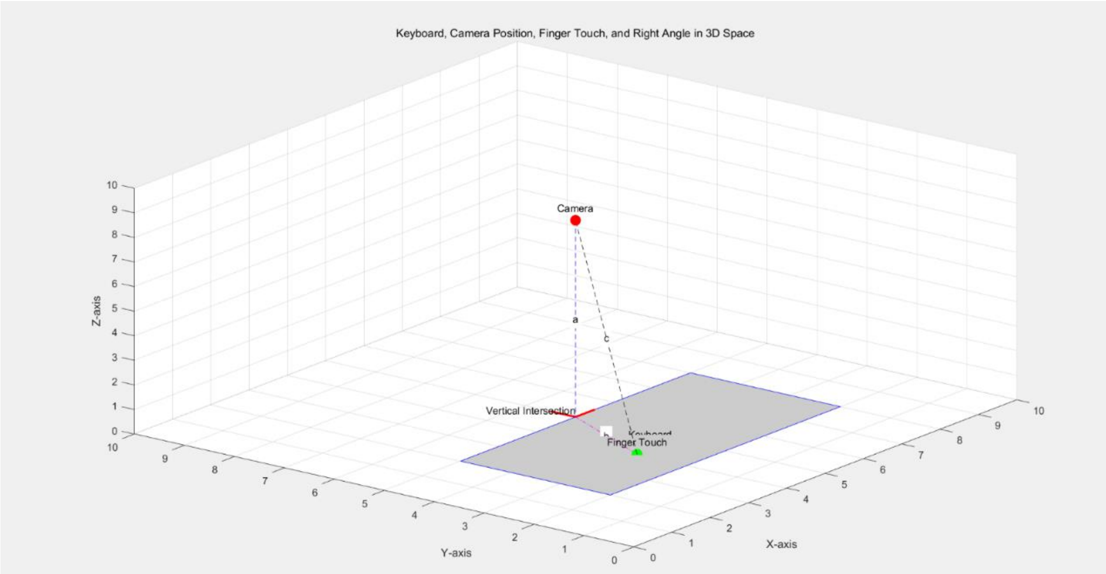

Figure 5

∠(a, camera, c) can be captured through the camera space.

a is the vertical distance from the camera to the plane.

c is the distance from the camera to the fingertip.

We need to detect whether cos∠(a, camera, c) * c is equal to a. If it is equal, it proves that the fingertip is on the keyboard plane. If cos∠(a, camera, c) * c is not equal to a, it means the fingertip Does not meet the z-value criteria. If cos∠(a, camera, c) * c equals a, it means that the value of the z-axis of the fingertip is 0 as shown in the legend.

3.2.4 Fingertip x and y coordinate positioning

Verify the x, y values of the fingertip and model them through MATLAB.

```
% Create a new figure window
figure;


% Define the range of the 3D space
xlim([0 10]);
```

```
ylim([0 10]);

zlim([0 10]);


% Enable grid for better observation

grid on;

hold on;

xlabel('X-axis');

ylabel('Y-axis');

zlabel('Z-axis');

title('Keyboard, Camera Position, Finger Touch, and Right Angle in 3D Space');


% Draw a simple keyboard shape on the xy-plane

rectangle('Position',[2 2 6 3], 'FaceColor', [0.8 0.8 0.8], 'EdgeColor', 'b');

text(5, 3.5, 'Keyboard', 'HorizontalAlignment', 'center');


% Define and plot the camera position

cameraPos = [5 5 8]; % x, y, z coordinates

plot3(cameraPos(1), cameraPos(2), cameraPos(3), 'ro', 'MarkerSize', 10, 'MarkerFaceColor', 'r');

text(cameraPos(1), cameraPos(2), cameraPos(3)+0.5, 'Camera', 'HorizontalAlignment', 'center');


% Define and plot the finger touch point on the keyboard

fingerTouchPos = [4 3 0]; % x, y, z coordinates (z is 0 because it's on the keyboard)
```

plot3(fingerTouchPos(1), fingerTouchPos(2), fingerTouchPos(3), 'go', 'MarkerSize', 10, 'MarkerFaceColor', 'g');

text(fingerTouchPos(1), fingerTouchPos(2), fingerTouchPos(3)+0.5, 'Finger Touch', 'HorizontalAlignment', 'center');

% Draw a line connecting the finger touch point and the camera position

line([fingerTouchPos(1), cameraPos(1)], [fingerTouchPos(2), cameraPos(2)], [fingerTouchPos(3), cameraPos(3)], 'Color', 'k', 'LineStyle', '--');

text((fingerTouchPos(1)+cameraPos(1))/2, (fingerTouchPos(2)+cameraPos(2))/2, (fingerTouchPos(3)+cameraPos(3))/2, 'c', 'HorizontalAlignment', 'center', 'BackgroundColor', 'white');

% Mark reference point 3 (2, 2, 0) on the keyboard

plot3(2, 2, 0, 'mo', 'MarkerSize', 5, 'MarkerFaceColor', 'm');

text(2, 2, 0.5, 'Reference Point 3', 'HorizontalAlignment', 'center');

% Mark reference point 4 (8, 2, 0) on the keyboard

plot3(8, 2, 0, 'mo', 'MarkerSize', 5, 'MarkerFaceColor', 'm');

text(8, 2, 0.5, 'Reference Point 4', 'HorizontalAlignment', 'center');

% Draw a line connecting the reference point 3 and the camera position

line([2, cameraPos(1)], [2, cameraPos(2)], [0, cameraPos(3)], 'Color', 'k', 'LineStyle', '--');

text((2+cameraPos(1))/2, (2+cameraPos(2))/2, (0+cameraPos(3))/2, 'C1', 'HorizontalAlignment', 'center', 'BackgroundColor', 'white');

% Draw a line connecting the reference point 4 and the camera position

line([8, cameraPos(1)], [2, cameraPos(2)], [0, cameraPos(3)], 'Color', 'k', 'LineStyle', '--');

text((8+cameraPos(1))/2, (2+cameraPos(2))/2, (0+cameraPos(3))/2, 'C2', 'HorizontalAlignment', 'center', 'BackgroundColor', 'white');

% Draw a line connecting the reference point 3 and the fingerTouchPos

line([2, fingerTouchPos(1)], [2, fingerTouchPos(2)], [0, fingerTouchPos(3)], 'Color', 'k', 'LineStyle', '--');

text((2+ fingerTouchPos(1))/2, (2+ fingerTouchPos(2))/2, (0+ fingerTouchPos(3))/2, 'D1', 'HorizontalAlignment', 'center', 'BackgroundColor', 'white');

% Draw a line connecting the reference point 4 and the fingerTouchPos

line([8, fingerTouchPos(1)], [2, fingerTouchPos(2)], [0, fingerTouchPos(3)], 'Color', 'k', 'LineStyle', '--');

text((8+ fingerTouchPos(1))/2, (2+ fingerTouchPos(2))/2, (0+ fingerTouchPos(3))/2, 'D2', 'HorizontalAlignment', 'center', 'BackgroundColor', 'white');

% Draw a line connecting the FingerTouchPos perpendicular to edge

line([4, fingerTouchPos(1)], [2, fingerTouchPos(2)], [0, fingerTouchPos(3)], 'Color', 'k', 'LineStyle', '--');

text((4+ fingerTouchPos(1))/2, (2+ fingerTouchPos(2))/2, (0+ fingerTouchPos(3))/2, ' vertical segment', 'HorizontalAlignment', 'center', 'BackgroundColor', 'white');

% Draw a line connecting the FingerTouchPos perpendicular to edge

line([4.2,4], [2.2,2.2], [0,0], 'Color', 'r', 'LineStyle', '-');

line([4.2,4.2], [2.2,2], [0,0], 'Color', 'r', 'LineStyle', '-');

% Draw a line on keyboard

text(5, 2, 0, 'D3', 'HorizontalAlignment', 'center', 'BackgroundColor', 'white');

% Draw a line of x

line([4,8], [2,2], [0,0], 'Color', 'r', 'LineStyle', '-');

text(6, 2, 0, 'x', 'HorizontalAlignment', 'center', 'BackgroundColor', 'white');

% Adjust the view for better observation

view(3);

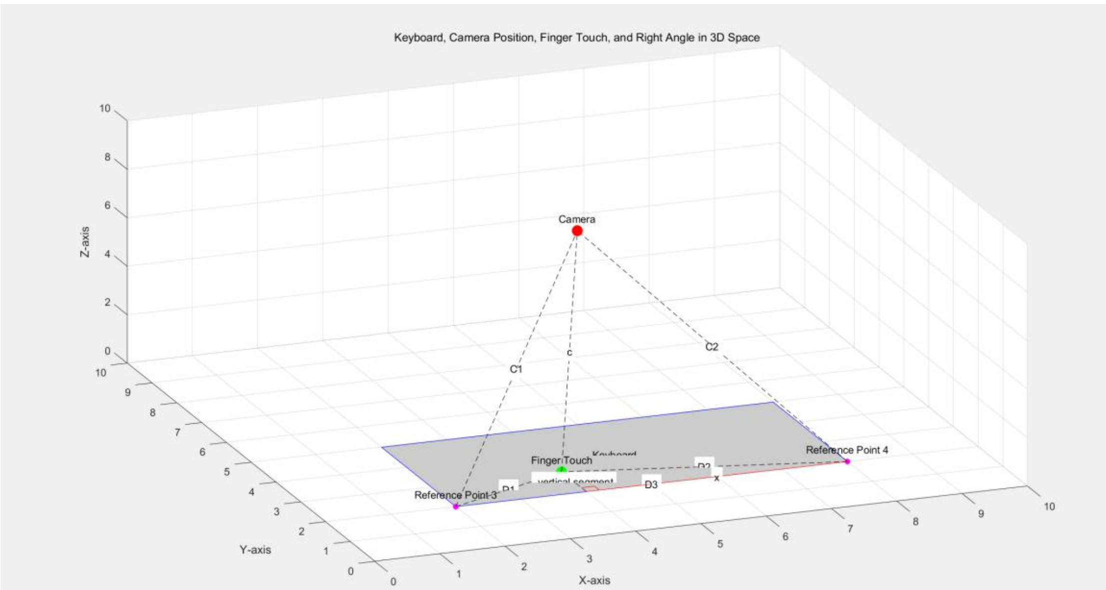

Figure 6

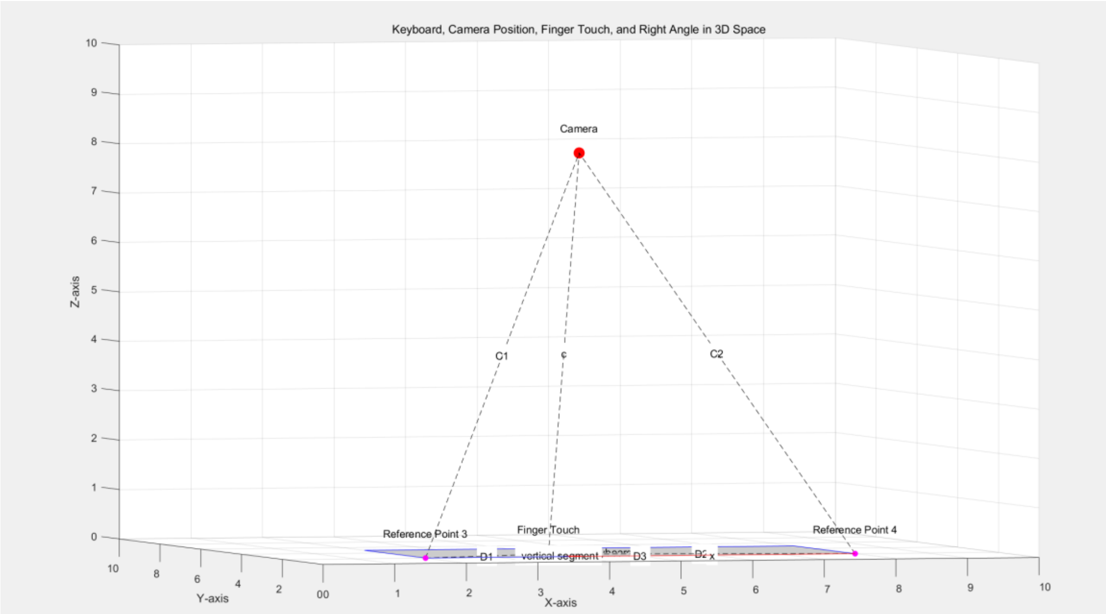

Figure 7

$$D1 = ((C1)^2 + (c)^2 - 2 * C1 * c * \cos\angle(C1, camera, c))^{(-2)}$$

$$D2 = ((C2)^2 + (c)^2 - 2 * C2 * c * \cos\angle(C2, camera, c))^{(-2)}$$

$$(D1)^2 - (D3 - x)^2 = (D2)^2 - (x)^2$$

$$x = (D2)^2 / (2 * D3) - (D1)^2 / (2 * D3) + D3 / 2$$

$$\text{Vertical segment} = ((D2)^2 - x^2)^{(-2)}$$

We can use the values of x and vertical segment to determine the relative values of x and y where the finger is on the keyboard plane.

3.3 Algorithm Development

```
Algorithm CheckKeyPress(D1, D2, D3, keyBounds)

    // Calculate the X and Y coordinates of the fingertip

    x = (D2^2) / (2 * D3) - (D1^2) / (2 * D3) + D3 / 2

    y = sqrt((D2^2) - x^2)


    // Check the cosine condition for the fingertip being on the keyboard plane
```

```
// Assuming 'a' and 'c' are known vectors from the fingertip and camera to a reference point
if cos(angle(a, camera, c)) * c != a
    return False

// Check if the fingertip is within the key bounds
for each key in keyBounds
    if x is within key.xMin and key.xMax and y is within key.yMin and key.yMax
        return True

return False
```

4. Results

4.1 Introduction to the Model

This mathematical model is being developed to enhance the precision and responsiveness of fingertip tracking in VR environments. It is currently in the idealized design phase, where theoretical foundations are being laid out before practical implementation. The model primarily utilizes principles of triangle geometry, including the Pythagorean theorem and the cosine theorem, to analyze and determine the position of a user's fingertips relative to the camera.

The crux of the model is to enable a single camera, preferably a Time-of-Flight (ToF) camera, to accurately capture and interpret the spatial positioning of fingertips. ToF cameras are particularly suited for this application due to their ability to measure distances accurately by recording the time it takes light to travel to objects and back. This feature is crucial for the model, as it provides the essential data for calculating distances and angles necessary for fingertip localization.

The model's functionality hinges on the camera's ability to precisely identify angles when detecting objects (in this case, fingertips). Once the camera captures the necessary spatial data, the mathematical model processes this information to calculate the position of the fingertips. This calculation involves using the aforementioned geometric theorems to interpret the data from the camera and translate it into accurate positional coordinates within the VR environment.

Implementing this model requires collaboration with experts across various fields, including electronic manufacturing, to develop the necessary hardware, and physics, to ensure accurate interpretation of spatial data. The interdisciplinary nature of this project highlights its complexity and the need for diverse technical expertise.

In essence, this mathematical model aims to significantly improve the interaction fidelity in VR systems by providing a highly accurate, real-time tracking of fingertip positions. This advancement could enhance user experience in VR applications, making interactions more intuitive and natural, and has potential applications in various domains, including gaming, education, and professional training simulations.

By bridging the gap between theoretical mathematics and practical electronic design, this model stands to offer a significant contribution to the field of VR technology, pushing the boundaries of how we interact with virtual environments.

4.2 Visualizations

Three-dimensional (3D) overall modeling was conducted using MATLAB.

```
% Create a new figure window

figure;


% Define the range of the 3D space

xlim([0 10]);

ylim([0 10]);

zlim([0 10]);
```

```
% Enable grid for better observation
grid on;
hold on;
xlabel('X-axis');
ylabel('Y-axis');
zlabel('Z-axis');
title('Keyboard, Camera Position, Finger Touch, and Right Angle in 3D Space');

% Draw a simple keyboard shape on the xy-plane
rectangle('Position',[2 2 6 3], 'FaceColor', [0.8 0.8 0.8], 'EdgeColor', 'b');
text(5, 3.5, 'Keyboard', 'HorizontalAlignment', 'center');

% Define and plot the camera position
cameraPos = [5 5 8]; % x, y, z coordinates
plot3(cameraPos(1), cameraPos(2), cameraPos(3), 'ro', 'MarkerSize', 10,
'MarkerFaceColor', 'r');
text(cameraPos(1), cameraPos(2), cameraPos(3)+0.5, 'Camera',
'HorizontalAlignment', 'center');

% Define and plot the finger touch point on the keyboard
fingerTouchPos = [4 3 0]; % x, y, z coordinates (z is 0 because it's on the
keyboard)
plot3(fingerTouchPos(1), fingerTouchPos(2), fingerTouchPos(3), 'go',
'MarkerSize', 10, 'MarkerFaceColor', 'g');
```

text(fingerTouchPos(1), fingerTouchPos(2), fingerTouchPos(3)+0.5, 'Finger Touch', 'HorizontalAlignment', 'center');

% Mark reference point 1 (1, 1, 0) on the desktop

plot3(1, 1, 0, 'ko', 'MarkerSize', 5, 'MarkerFaceColor', 'k');

text(1, 1, 0.5, 'Ref Point 1', 'HorizontalAlignment', 'center');

% Mark reference point 2 (9, 1, 0) on the desktop

plot3(9, 1, 0, 'ko', 'MarkerSize', 5, 'MarkerFaceColor', 'k');

text(9, 1, 0.5, 'Ref Point 2', 'HorizontalAlignment', 'center');

% Mark control point 1 (2, 2, 0) on the keyboard

plot3(2, 2, 0, 'mo', 'MarkerSize', 5, 'MarkerFaceColor', 'm');

text(2, 2, 2, 'Control Point 1', 'HorizontalAlignment', 'center');

% Mark control point 2 (8, 2, 0) on the keyboard

plot3(8, 2, 0, 'mo', 'MarkerSize', 5, 'MarkerFaceColor', 'm');

text(8, 2, 2, 'Control Point 2', 'HorizontalAlignment', 'center');

% Draw a line connecting the reference point1 and the camera position

line([1, cameraPos(1)], [1, cameraPos(2)], [0, cameraPos(3)], 'Color', 'k', 'LineStyle', '--');

text((1+cameraPos(1))/2, (1+cameraPos(2))/2, (0+cameraPos(3))/2, 'A1', 'HorizontalAlignment', 'center', 'BackgroundColor', 'white');

```
% Draw a line connecting the control point1 and the camera position
line([2, cameraPos(1)], [2, cameraPos(2)], [0, cameraPos(3)], 'Color', 'k', 'LineStyle', '--');
text((2+cameraPos(1))/2, (2+cameraPos(2))/2, (0+cameraPos(3))/2, 'A2', 'HorizontalAlignment', 'center', 'BackgroundColor', 'white');

% Draw a line connecting the reference point1 and control point1
line([1, 2], [1,2], [0, 0], 'Color', 'k', 'LineStyle', '--');
text((1+2)/2, (1+2)/2, (0+0)/2, 'A3', 'HorizontalAlignment', 'center', 'BackgroundColor', 'white');

% Draw a line connecting the reference point2 and the camera position
line([9, cameraPos(1)], [1, cameraPos(2)], [0, cameraPos(3)], 'Color', 'k', 'LineStyle', '--');
text((9+cameraPos(1))/2, (1+cameraPos(2))/2, (0+cameraPos(3))/2, 'B1', 'HorizontalAlignment', 'center', 'BackgroundColor', 'white');

% Draw a line connecting the control point2 and the camera position
line([8, cameraPos(1)], [2, cameraPos(2)], [0, cameraPos(3)], 'Color', 'k', 'LineStyle', '--');
text((8+cameraPos(1))/2, (2+cameraPos(2))/2, (0+cameraPos(3))/2, 'B2', 'HorizontalAlignment', 'center', 'BackgroundColor', 'white');

% Draw a line connecting the reference point2 and control point2
line([9, 8], [1,2], [0, 0], 'Color', 'k', 'LineStyle', '--');
```

```
text((9+8)/2, (1+2)/2, (0+0)/2, 'B3', 'HorizontalAlignment', 'center', 'BackgroundColor', 'white');

% Draw a line connecting the finger touch point and the camera position
line([fingerTouchPos(1), cameraPos(1)], [fingerTouchPos(2), cameraPos(2)], [fingerTouchPos(3), cameraPos(3)], 'Color', 'k', 'LineStyle', '--');
text((fingerTouchPos(1)+cameraPos(1))/2, (fingerTouchPos(2)+cameraPos(2))/2, (fingerTouchPos(3)+cameraPos(3))/2, 'c', 'HorizontalAlignment', 'center', 'BackgroundColor', 'white');

% Mark reference point 3 (2, 2, 0) on the keyboard
plot3(2, 2, 0, 'mo', 'MarkerSize', 5, 'MarkerFaceColor', 'm');
text(2, 2, 0.5, 'Reference Point 3', 'HorizontalAlignment', 'center');

% Mark reference point 4 (8, 2, 0) on the keyboard
plot3(8, 2, 0, 'mo', 'MarkerSize', 5, 'MarkerFaceColor', 'm');
text(8, 2, 0.5, 'Reference Point 4', 'HorizontalAlignment', 'center');

% Draw a line connecting the reference point 3 and the camera position
line([2, cameraPos(1)], [2, cameraPos(2)], [0, cameraPos(3)], 'Color', 'k', 'LineStyle', '--');
text((2+cameraPos(1))/2, (2+cameraPos(2))/2, (0+cameraPos(3))/2, 'C1', 'HorizontalAlignment', 'center', 'BackgroundColor', 'white');

% Draw a line connecting the reference point 4 and the camera position
```

```
line([8, cameraPos(1)], [2, cameraPos(2)], [0, cameraPos(3)], 'Color', 'k', 'LineStyle', '--');

text((8+cameraPos(1))/2, (2+cameraPos(2))/2, (0+cameraPos(3))/2, 'C2', 'HorizontalAlignment', 'center', 'BackgroundColor', 'white');


% Draw a line connecting the reference point 3 and the fingerTouchPos

line([2, fingerTouchPos(1)], [2, fingerTouchPos(2)], [0, fingerTouchPos(3)], 'Color', 'k', 'LineStyle', '--');

text((2+ fingerTouchPos(1))/2, (2+ fingerTouchPos(2))/2, (0+ fingerTouchPos(3))/2, 'D1', 'HorizontalAlignment', 'center', 'BackgroundColor', 'white');


% Draw a line connecting the reference point 4 and the fingerTouchPos

line([8, fingerTouchPos(1)], [2, fingerTouchPos(2)], [0, fingerTouchPos(3)], 'Color', 'k', 'LineStyle', '--');

text((8+ fingerTouchPos(1))/2, (2+ fingerTouchPos(2))/2, (0+ fingerTouchPos(3))/2, 'D2', 'HorizontalAlignment', 'center', 'BackgroundColor', 'white');


% Draw a line connecting the FingerTouchPos perpendicular to edge

line([4, fingerTouchPos(1)], [2, fingerTouchPos(2)], [0, fingerTouchPos(3)], 'Color', 'k', 'LineStyle', '--');

text((4+ fingerTouchPos(1))/2, (2+ fingerTouchPos(2))/2, (0+ fingerTouchPos(3))/2, ' vertical segment', 'HorizontalAlignment', 'center', 'BackgroundColor', 'white');


% Draw a line connecting the FingerTouchPos perpendicular to edge
```

```
line([4.2,4], [2.2,2.2], [0,0], 'Color', 'r', 'LineStyle', '-');

line([4.2,4.2], [2.2,2], [0,0], 'Color', 'r', 'LineStyle', '-');


% Draw a line on keyboard

text(5, 2, 0, 'D3', 'HorizontalAlignment', 'center', 'BackgroundColor', 'white');


% Draw a line of x

line([4,8], [2,2], [0,0], 'Color', 'r', 'LineStyle', '-');

text(6, 2, 0, 'x', 'HorizontalAlignment', 'center', 'BackgroundColor', 'white');


% Adjust the view for better observation

view(3);
```

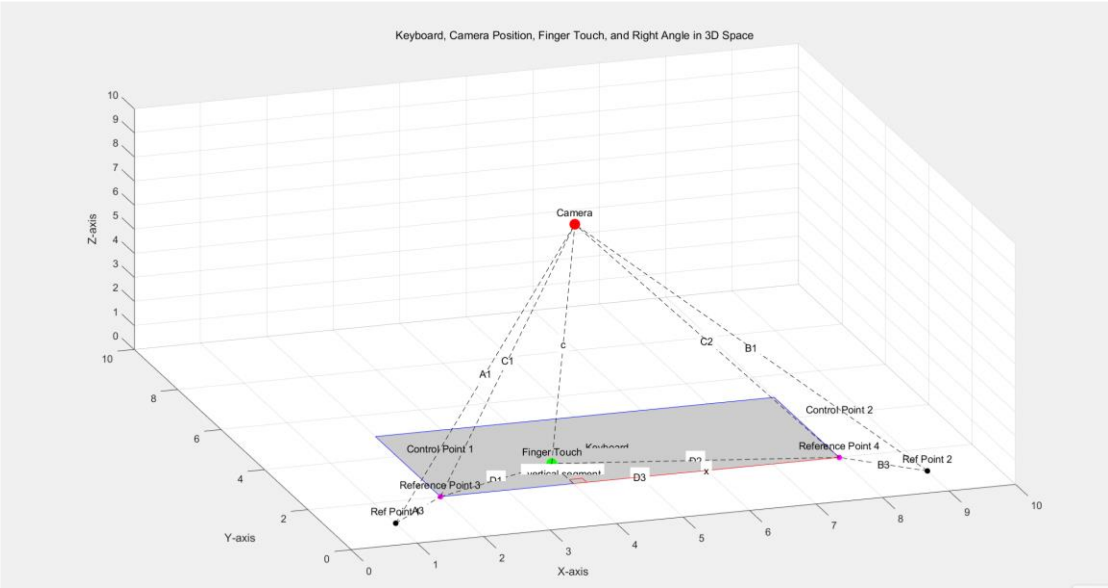

Figure 8

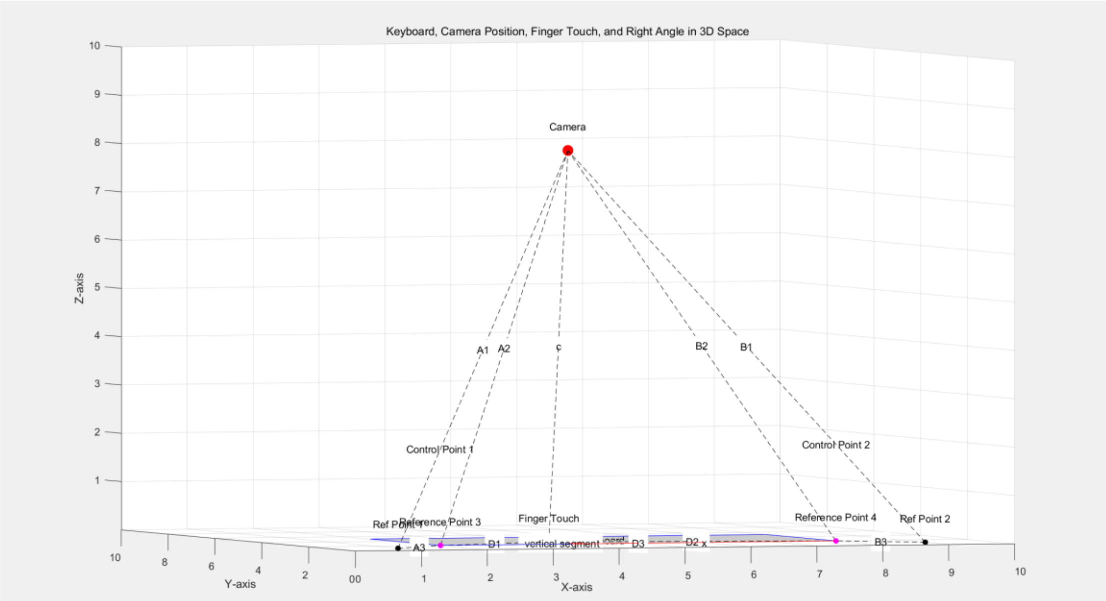

Figure 9

4.3 Model Outputs

In constructing the mathematical model for the camera-based fingertip tracking system, the process began with the application of the distance calculation formula specific to the LoF camera. This initial step was crucial in determining the spatial coordinates of the fingertips relative to the camera. Following this, the model employed the Pythagorean theorem [27], a fundamental principle in geometry, to ascertain whether the movement of the finger occurred within a specific plane. This involved calculating the distance between various points in three-dimensional space to verify if the movements conformed to a two-dimensional plane. Additionally, the cosine theorem was utilized as a pivotal tool to determine if the finger's movements were confined within a predefined range on this plane. The integration of these mathematical principles - the LOF camera's distance formula, the Pythagorean theorem, and the cosine theorem [28] - was vital in ensuring the accuracy and reliability of the system in tracking finger movements in a defined spatial context. This meticulous approach allowed for precise positioning and movement tracking of the fingers in the virtual environment, enhancing the overall VR experience.

Distance_D1 = (Speed of Light_D1 × Time of Flight_D1) / 2

Distance_D2 = (Speed of Light_D2 × Time of Flight_D2) / 2

Distance_D3 = (Speed of Light_D3 × Time of Flight_D3) / 2

Distance_a = (Speed of Light_a × Time of Flight_a) / 2

Distance_c = (Speed of Light_c × Time of Flight_c) / 2

```
Algorithm CheckKeyPress(Distance_D1, Distance_D2, Distance_D3, keyBounds)
    // Calculate the X and Y coordinates of the fingertip
    x = (Distance_D2^2) / (2 * Distance_D3) - (Distance_D1^2) / (2 * Distance_D3) + Distance_D3 / 2
    y = sqrt((Distance_D2^2) - x^2)

    // Check the cosine condition for the fingertip being on the keyboard plane
    // Assuming 'a' and 'c' are known vectors from the fingertip and camera to a reference point
    if cos(angle(a, camera, c)) * Distance_c != Distance_a
        return False

    // Check if the fingertip is within the key bounds
    for each key in keyBounds
        if x is within key.xMin and key.xMax and y is within key.yMin and key.yMax
            return True

    return False
```

5. Discussion

5.1 Interpretation of Results

This mathematical model is designed to handle camera positioning in VR devices to accurately track the position and movement of the user's fingertips. Key algorithms may include object recognition (identifying a fingertip), spatial analysis (determining its position in 3D space), and possibly motion tracking (tracking its movement over time). Algorithms utilize geometric principles such as Pythagoras' theorem or trigonometry, specifically for calculating angles and distances in 3D space. The significance of the model is: 1. Enhanced VR interaction. Accurate fingertip tracking enables more natural and intuitive interactions in VR environments, making the experience more immersive. 2. Impact in multiple fields. This technology affects fields such as virtual training offices, telemedicine, education and rehabilitation, providing new tools and methods.

The research objectives and hypotheses are:

1. Camera accuracy: A basic assumption is that the camera used is sufficiently accurate in locating physical objects. That means high-resolution imaging and possibly advanced features like depth sensing. Accurate spatial positioning is critical for the model to correctly interpret the position of the fingertip relative to the camera. 2. Hand model accuracy: Another aspect is the accuracy of the hand model used by the system, especially its ability to take into account the thickness and detailed shape of the human hand.

This detail is necessary to accurately convert a 3D object (hand) into the digital realm, especially for complex movements. Actually. This kind of precision can greatly enhance the VR experience, allowing for more detailed and diverse interactions.

5.2 Theoretical Framework

"Advanced camera systems, such as those equipped with Lidar (Light Detection and Ranging) and Time-of-Flight (ToF) sensors, represent a significant leap in depth sensing technology. Lidar systems work by emitting a light signal, often a laser, and measuring the time taken for the light to reflect back from a surface. This travel time is then converted into distance measurements. Similarly, ToF cameras, a distinct yet related technology, also measure the distance by calculating the time it takes for a light signal to return. However, the accuracy of ToF cameras is notably superior, except when compared to structured light cameras. According to 'Advantages and Disadvantages of Time-of-Flight (ToF) Cameras - FRAMOS' (www.framos.com), ToF cameras can achieve an accuracy ranging from 1 mm to 1 cm, depending on the operating range of the camera[26].

Further enhancing the capabilities of ToF cameras, the integration of Inertial Measurement Units (IMUs) allows for precise calculations of rotation angles. As noted in 'Benefits of Inertial Measurement Unit (IMU) Archives' by e-con Systems (September 26, 2023), the presence of an IMU sensor in some ToF cameras adds a layer of spatial orientation data, which is critical in applications requiring movement tracking [29].

Given these technological advancements and the existing literature's comprehensive coverage of hand positioning models, my findings suggest a promising direction for integrating these results. This integration could significantly benefit the development of more convenient and accurate mathematical models for VR wearables, particularly in the realm of fingertip tracking. Such developments hold the potential to revolutionize user interactions in virtual environments, offering a level of precision and naturalism previously unattainable."

5.3 Practical Implications

This model has many potential benefits. Specifically, this includes enhancing the user experience in VR: The ability to accurately track fingertips using a single camera in a VR device can lead to a more immersive and interactive virtual reality experience. Users can interact with virtual environments more naturally, making VR more intuitive and engaging. Applications in games and simulations: In games, this precise tracking enables more complex and realistic gameplay, where players can perform subtle movements with their hands, such as picking up objects, painting, or playing virtual instruments. Likewise, in training and educational simulations, the technology can provide hands-on experience without the need for real-world materials. Accessibility and remote control: This technology could be a game-changer for individuals with limited mobility. It will allow them to control devices and interfaces with subtle finger movements, thereby enhancing their ability to interact with technology and improving their quality of life. Medical Rehabilitation: In physical therapy, especially for hand injuries, precise tracking of finger movements can help create customized rehabilitation plans. It also provides detailed feedback and progress tracking to patients and therapists.

But this model also has potential limitations. Hardware limitations: The effectiveness of the mathematical model depends on the camera quality and processing power of the VR device. A low-resolution camera or slower processing speed may affect the accuracy and responsiveness of fingertip tracking. Environmental limitations: Tracking accuracy may be affected by environmental factors, such as lighting conditions or background clutter, which may limit the effectiveness of the system in different real-world scenarios. User diversity: Differences in hand size, finger length, or skin color among different users can create challenges in achieving universally accurate tracking without extensive calibration or adaptive algorithms. Privacy concerns: Using cameras for tracking may raise privacy concerns, especially if data is stored or transmitted. Ensuring data security and privacy is crucial in practical applications. Cost: Implementing high-precision tracking technology may increase the cost of VR equipment, potentially limiting the experience for budget-conscious consumers. Accuracy of the hand model: This requires high accuracy in identifying the hands of different people, and locating the specific position of the fingertip press according to different sizes of hands.

The practical application of mathematical models for single-camera fingertip tracking in VR devices opens up countless possibilities for enhanced virtual interactions, accessibility, gaming, and medical rehabilitation. However, its successful implementation depends on overcoming challenges related to hardware capabilities, environmental changes, user diversity, privacy issues, and cost. Future developments in technology and more refined models could further alleviate these limitations and expand the scope and efficacy of this innovative approach.

5.4 Contributions to the Field

The contribution of this model to computer vision is mainly reflected in the following aspects. Advanced object recognition and spatial mapping: Camera-based fingertip tracking in VR devices pushes the boundaries of object recognition and spatial mapping in computer vision. To accurately interpret and track the fine movements of the fingertips, these systems must process complex visual information, including distinguishing the fingers from various backgrounds and understanding their location in three-dimensional space. Real-time processing and responsiveness: The need to track finger movements in VR in real time has driven advances in fast image processing technology. Computer vision algorithms have become more efficient at processing large amounts of data in real time, resulting in enhanced responsiveness that is critical for immersive VR experiences. Machine learning and artificial intelligence integration: To improve the accuracy of fingertip tracking, VR devices often leverage machine learning models to learn a variety of hand shapes, sizes, and movements. This integration advances AI's ability to understand and interpret human gestures, making a significant contribution to the field of computer vision.

At the same time, the mathematical model also has wider implications and applications. Human-Computer Interaction (HCI): One of the most straightforward applications is HCI, where fingertip tracking allows for more natural and intuitive interaction with virtual environments. This has implications for user interface design, with gestures replacing traditional input devices such as keyboards and mice. Teleoperation and remote control: VR devices with precise fingertip tracking can enable precise remote control of robotic systems in situations where direct human intervention is risky or impractical, such as in surgery or handling hazardous materials. Education and training: In educational settings, VR with precise fingertip tracking provides a level of interactivity that can enhance the learning experience, especially in fields such as medicine, engineering, and the arts, where hands-on exercises are invaluable. Rehabilitation and therapy: In medical rehabilitation, VR devices equipped with this technology can aid in the recovery of fine motor skills by providing patients with detailed and engaging exercises to track and encourage their progress. Entertainment and Games: In games and entertainment, the technology enhances user engagement and interactivity, making virtual experiences more immersive and real.

The contribution of VR devices, especially those equipped with camera-based fingertip tracking models, to computer vision and its related applications is enormous. Not only do they advance technical capabilities in the field of computer vision, but they also pave the way for innovative applications in various fields, enhancing the way humans interact with each other in digital systems and virtual environments. As technology continues to evolve, we can expect more breakthrough contributions from this synergy.

5.5 Limitations

Camera object detection distance accuracy. Resolution limitations: A camera's resolution affects its ability to accurately detect small movements or distances. Lower resolution may result in a less accurate representation of finger position, especially at longer distances. Depth Sensing Limitations: The effectiveness of depth sensing can be affected by environmental factors such as lighting conditions, reflective surfaces, or background complexity. Inconsistent depth data can lead to inaccurate finger position modeling. Range Limitations: Most cameras have an optimal operating range. Beyond this range, the accuracy of distance measurement will decrease significantly, affecting the accuracy of the virtual hand model.

Angular accuracy of camera object detection. Directional sensitivity: Accurately detecting the direction of a finger requires high sensitivity, especially for complex or subtle gestures. Any limitations in detecting the precise angles of finger joints may lead to errors in gesture interpretation. Occlusion: A common problem in VR systems is occlusion of fingers or hand parts. When fingers are aligned in a way that blocks the camera's view, angular accuracy can be affected, resulting in incorrect model interpretation. Calibration errors: Inaccuracies in camera calibration (including lens distortion or alignment errors) can affect angle measurements, causing discrepancies between actual and virtual hand positions.

Accuracy of hand virtual model. Model Summary: Virtual hand models need to account for variations in hand size, shape, and finger length for different users. A model that is too general may not accurately represent the hand movements of an individual user.

Dynamic vs. static models: Static models may not effectively capture the full range of hand motion. Dynamic models, while more general, require more complex algorithms and more processing power, which can introduce latency. Feedback and interaction limitations: The lack of haptic feedback in VR may limit the realism and accuracy of hand interactions because users cannot feel the objects they are interacting with in the virtual environment.

Limitations in the distance and angle accuracy of camera detection, as well as limitations in the virtual hand model, present significant challenges in developing a fully accurate and reliable VR-based fingertip positioning system. Improvements in camera technology, more sophisticated modeling algorithms, and enhanced processing power can help alleviate some of these limitations. However, the complexities inherent in accurately replicating the subtleties of human hand movements in virtual environments mean that some degree of approximation is often unavoidable.

5.6 Future Research

Advanced camera technology and algorithms. High-resolution and high-speed cameras: Research into integrating higher-resolution and faster cameras could improve the accuracy of distance and angle detection, especially for subtle and fast finger movements. Improved depth-sensing technologies: Develop more sophisticated depth-sensing technologies that are less affected by environmental variables such as lighting and background complexity.

Enhanced object detection algorithms: Create algorithms that more accurately detect and track fingers, even in the presence of occlusions and orientation changes.

Dynamic and personalized hand models. Customizable hand models: Develop adaptive models that can be calibrated to the individual user's hand shape and size for more accurate tracking. Dynamic Modeling: Research into dynamic models that better capture the range and fluidity of human hand motion. Machine learning methods: Leverage machine learning techniques to develop models that learn from large datasets of hand movements and improve over time.

Improved calibration and error correction

Advanced calibration methods: Create more robust calibration methods to reduce errors due to camera alignment and lens distortion. Real-time Error Correction: Implements a real-time error correction algorithm that can adjust when errors occur in angle and distance measurements.

Enhanced user interaction and feedback. Tactile feedback integration: Study methods of integrating tactile feedback into VR systems to provide users with a sense of touch and enhance the realism of virtual hand interaction. Natural User Interaction (NUI) enhancements: Develop more intuitive and natural user interaction models for VR that closely mimic real-world hand usage.

Address environmental and operational constraints. Robustness in different environments: Make the system more robust under various environmental conditions, such as different lighting settings or backgrounds. Minimize Latency: Focus on reducing the delay between the user's actual actions and the corresponding virtual representation.

Interdisciplinary approach. Collaboration with Neuroscience and Psychology: Collaborate with experts in fields such as neuroscience and psychology to gain a deeper understanding of how humans perceive and interact with virtual environments. Ergonomics and Usability Studies: Conduct extensive ergonomics and usability studies to understand how users interact with VR systems and how to improve those interactions.

Future research into VR fingertip tracking and modeling should aim to balance technological advances with user-centered design principles. By addressing current limitations and exploring these research avenues, next-generation VR systems can deliver more accurate, responsive, and immersive experiences.

6. Conclusions

This article presents a detailed study of a fingertip positioning model for virtual reality (VR) devices, utilizing a single built-in camera. The motivation behind this research stems from the current lack of a comprehensive keyboard operating system specifically designed for VR environments. The primary objective is to enhance the tracking accuracy of the positional relationship between the fingertips and the VR interface. The core of this research is the development of a mathematical and computational model that accurately tracks fingertip movement using a single camera embedded in VR devices. The model leverages advanced algorithms and geometric principles to interpret camera data, thereby determining the precise location of the fingertips in the virtual space. A significant contribution of this study is in the realm of algorithmic development. The algorithms are designed to process visual data efficiently, ensuring high accuracy in fingertip tracking. These algorithms allow for a reduction in the number of cameras required, contributing to a decrease in the overall weight and bulk of VR wearables. The findings and technological advancements from this study have practical applications in various VR products. This includes VR gloves and controllers that require precise fingertip tracking. By improving the accuracy of fingertip positioning, the study aims to optimize data processing efficiency and enhance the user experience in VR applications, such as gaming, education, and professional simulations. The paper discusses the future trajectory of VR technology, emphasizing the need for lighter, more convenient, and more accurate VR devices. As VR technology evolves, there will be increasing demands for precision in both the algorithmic and mechanical aspects of VR devices. The model proposed in this study is adaptable and scalable for future advancements, especially with the integration of ultra-high-quality mechanical components.

References

[1] M. S. Akremi, R. Slama, and H. Tabia. Spd siamese neural network for skeleton-based hand gesture recognition. In VISIGRAPP (4: VISAPP), pages 394–402, 2022.

[2] F. M. Caputo, P. Prebianca, A. Carcangiu, L. D. Spano, and A. Gia chetti. Comparing 3d trajectories for simple mid-air gesture recogni tion. Computers & Graphics, 73:17–25, 2018.

[3] Q. De Smedt, H. Wannous, and J.-P. Vandeborre. Skeleton-based dynamic hand gesture recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pages 1–9, 2016.

[4] H. Liu, J. Tu, M. Liu, and R. Ding. Learning explicit shape and motion evolution maps for skeleton-based human action recognition. In 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 1333–1337. IEEE, 2018.

[5] J. Hou, G. Wang, X. Chen, J.-H. Xue, R. Zhu, and H. Yang. Spatial temporal attention res-tcn for skeleton-based dynamic hand gesture recognition. In Proceedings of the European conference on computer vision (ECCV) workshops, pages 0–0, 2018.

[6] J. Liu, Y. Liu, Y. Wang, V. Prinet, S. Xiang, and C. Pan. Decoupled representation learning for skeleton-based gesture recognition. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 5751–5760, 2020.

[7] X. S. Nguyen, L. Brun, O. Lezoray, and S. Bougleux. A neural ´ network based on spd manifold learning for skeleton-based hand gesture recognition. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 12036–12045, 2019.

[8] Z. Huang and L. Van Gool. A riemannian network for spd matrix learning. In Thirty-first AAAI conference on artificial intelligence, 2017.

[9] Z. Huang, J. Wu, and L. Van Gool. Building deep networks on grassmann manifolds. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 32, 2018.

[10] G. Devineau, F. Moutarde, W. Xi and J. Yang, "Deep Learning for Hand Gesture Recognition on Skeletal Data," 2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018), Xi'an, China, 2018, pp. 106-113, doi: 10.1109/FG.2018.00025.

[11] R. Kabir, N. Ahmed, N. Roy and M. R. Islam, "A Novel Dynamic Hand Gesture and Movement Trajectory Recognition model for Non-Touch HRI Interface," 2019 IEEE Eurasia Conference on IOT, Communication and Engineering (ECICE), Yunlin, Taiwan, 2019, pp. 505-508, doi: 10.1109/ECICE47484.2019.8942691.

[12] O. Köpüklü, T. Ledwon, Y. Rong, N. Kose and G. Rigoll, "DriverMHG: A Multi-Modal Dataset for Dynamic Recognition of Driver Micro Hand Gestures and a Real-Time Recognition Framework," 2020 15th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2020), Buenos Aires, Argentina, 2020, pp. 77-84, doi: 10.1109/FG47880.2020.00041.

[13] M. Oudah, A. Al-Naji, and J. Chahl, “Hand Gesture Recognition Based on Computer Vision: A Review of Techniques,” Journal of Imaging, vol. 6, no. 8, p. 73, Jul. 2020, doi: 10.3390/jimaging6080073.

[14] M. Garg, P. M. Pradhan and D. Ghosh, "Multiview Hand Gesture Recognition using Deep Learning," 2021 IEEE 18th India Council International Conference (INDICON), Guwahati, India, 2021, pp. 1-6, doi: 10.1109/INDICON52576.2021.9691523.

[15] X. Hao, J. Li, Y. Guo, T. Jiang and M. Yu, "Hypergraph Neural Network for Skeleton-Based Action Recognition," in IEEE Transactions on Image Processing, vol. 30, pp. 2263-2275, 2021, doi: 10.1109/TIP.2021.3051495.
[16] Z. Zhang, B. Wu and Y. Jiang, "Gesture Recognition System Based on Improved YOLO v3," 2022 7th International Conference on Intelligent Computing and Signal Processing (ICSP), Xi'an, China, 2022, pp. 1540-1543, doi: 10.1109/ICSP54964.2022.9778394.
[17] T. -T. Nguyen et al., "A Continuous Real-time Hand Gesture Recognition Method based on Skeleton," 2022 11th International Conference on Control, Automation and Information Sciences (ICCAIS), Hanoi, Vietnam, 2022, pp. 273-278, doi: 10.1109/ICCAIS56082.2022.9990122.
[18] R. Slama, W. Rabah and H. Wannous, "STr-GCN: Dual Spatial Graph Convolutional Network and Transformer Graph Encoder for 3D Hand Gesture Recognition," 2023 IEEE 17th International Conference on Automatic Face and Gesture Recognition (FG), Waikoloa Beach, HI, USA, 2023, pp. 1-6, doi: 10.1109/FG57933.2023.10042643.
[19] M. Janiszewski, M. Torkan, L. Uotinen, and M. Rinne, “Rapid Photogrammetry with a 360-Degree Camera for Tunnel Mapping,” Remote Sensing, vol. 14, no. 21, p. 5494, Oct. 2022, doi: 10.3390/rs14215494.
[20] Xu, Ningli, et al. "Point Cloud Registration for LiDAR and Photogrammetric Data: A Critical Synthesis and Performance Analysis on Classic and Deep Learning Algorithms." ISPRS Open Journal of Photogrammetry and Remote Sensing, vol. 8, 1 Apr. 2023, p. 100032, doi: 10.1016/j.ophoto.2023.100032.
[21] W. Song, L. Liu, T. Yifei, G. Sun, S. Fong, and K. Cho, "A 3D localisation method in indoor environments for virtual reality applications," Human-centric Computing and Information Sciences, vol. 7, 2017. doi: 10.1186/s13673-017-0120-7.
[22] R. Kromanis and P. Kripakaran, "A multiple camera position approach for accurate displacement measurement using computer vision," Journal of Civil Structural Health Monitoring, vol. 11, no. 3, pp. 661-678, 2021.
[23] H. Pham, “THE CHALLENGE OF HAND GESTURE INTERACTION IN THE VIRTUAL REALITY ENVIRONMENT Evaluation of in-air hand gesture using the Leap Motion Controller.”
[24] E. Maryatt, “Typing in the Virtual Office Environment: Examining the Effects of Keyboard and Hand Representation in VR,” Microsoft Research, Feb. 05, 2018.
[25] SparkCognition, “What are the Best Cameras for Visual AI Systems?,” SparkCognition, Jul. 01, 2023.
[26] Advantages and Disadvantages of Time-of-Flight (ToF) Cameras – FRAMOS,” www.framos.com.
[27] [4]Wikipedia Contributors, “Pythagorean theorem,” Wikipedia, Sep. 26, 2019. https://en.wikipedia.org/wiki/Pythagorean_theorem
[28] [5]“Law of cosines,” Wikipedia, Dec. 04, 2020. https://en.wikipedia.org/wiki/Law_of_cosines
[29] Benefits of the Inertial Measurement Unit (IMU) Archives,” e-con Systems, Sep. 26, 2023.