REAL TIME HUMAN MACHINE INTERACTION IN VIRTUAL ENVIRONMENT

A PROJECT REPORT

Submitted by

BOOBASH AYYANAR M	**RA2211028020021**
HARISH RAJ M	**RA2211028020025**
KARTHIK R	**RA2211028020026**

Under the guidance of

Dr. S. Rubin Bose
(Assistant Professor)

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

with specialization in

CLOUD COMPUTING

of

FACULTY OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
RAMAPURAM, CHENNAI – 600089

APRIL 2025

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Deemed to be University U/S 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that this project report titled **"REAL TIME HUMAN MACHINE INTERACTION IN VIRTUAL ENVIRONMENT"** is the bonafide work of **BOOBASH AYYANAR.M [REGNO:RA2211028020021], HARISHRAJ.M [REGNO:RA2211028020025], KARTHIK. R [REG NO: RA2211028020026]** who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an occasion on this or any other candidate.

SIGNATURE

Dr. S. Rubin Bose, M.E, M.B.A, Ph.D.

Assistant professor,

School of Computer Science Engineering
SRM Institute of Science and Technology,
Chennai-89.

SIGNATURE

Dr. A. Umamageswari, M.Tech., PhD.,

Professor and Head of Cloud Computing

School of Computer Science Engineering,
SRM Institute of Science and Technology,
Chennai-89.

Submitted for the project viva-voce held on _____ at SRM Institute of Science and Technology, Chennai -600089.

INTERNAL EXAMINER I **INTERNAL EXAMINER II**

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY, RAMAPURAM, CHENNAI - 89

DECLARATION

We hereby declare that the entire work contained in this project report titled "**REAL TIME HUMAN MACHINE INTERACTION IN VIRTUAL ENVIRONMENT**" has been carried out by **BOOBASH AYYANAR. M [REG NO: RA2211028020021]**, **HARISH RAJ.M [REG NO: RA2211028020025], KARTHIK. R [REG NO: RA2211086020026]** at SRM Institute of Science and Technology, Ramapuram, Chennai- 600089, under the guidance of **Dr. S. Rubin Bose, M.E, M.B.A, Ph.D.,** Assistant professor**,** School of Computer Science Engineering.

Place: Chennai

Date:

BOOBASH AYYANAR. M

HARISH RAJ. M

KARTHIK. R

ACKNOWLEDGEMENT

We place on record our deep sense of gratitude to our lionized Chairman **Dr. R. SHIVAKUMAR, MBBS., MD.,** for providing us with the requisite infrastructure throughout the course.

We take the opportunity to extend our hearty and sincere thanks to our **Dean, Dr. M. SAKTHI GANESH., Ph.D.,** for maneuvering us into accomplishing the project.

We take the privilege to extend our hearty and sincere gratitude to the Professor and Chairperson, **Dr. K. RAJA, Ph.D.,** for his suggestions, support and encouragement towards the completion of the project with perfection.

We thank our honorable Head of the department **Dr. A. UMAMAGESWARI, M. Tech., Ph.D.,** Computer Science and Engineering With specialization in Cloud Computing for her constant motivation and unwavering support.

We express our hearty and sincere thanks to our guide **Dr. S. RUBIN BOSE, M.E, M.B.A, Ph.D.,** Assistant Professor**,** School of Computer Science Engineering for his encouragement, consecutive criticism and constant guidance throughout this project work.

Our thanks to the teaching and non-teaching staff of the Department of Computer Science and Engineering of SRM Institute of Science and Technology, Chennai, for providing necessary resources for our project

ABSTRACT

Hand tracking within virtual environments has emerged as a vital aspect of modern human-computer interaction (HCI), facilitating more automatic and natural control. In this paper, we present a real-time 3D hand tracking and interaction method that provides an interesting and natural way for human-computer interaction with numerous applications in a virtual environment, without requiring any external device. The technology we used in this system is Mediapipe, a machine learning technology developed by Google. It tracks the 21 hand landmarks in 3D using a webcam. The spatial coordinates for the real-time movement of the user's hand are processed and sent to the virtual environment designed using Unity, where they are simulated for interaction with virtual objects. The system employs OpenCV and CVZone for video capture, landmark display, and efficient packing of 21 hand coordinates. Socket programming is used for the communication between the tracking module and Unity environment. The data being transferred is quite low as well. This system runs on motion tracking instead of gesture tracking, which results in the overall smoothness, responsiveness, and immersion experience in the virtual world. The suggested framework has immense possible use in virtual reality (VR), augmented reality (AR), gaming and simulation training for individuals with disabilities. It provides a cheaper and easier option compared to hardware-reliant tracking systems. It focuses on flexibility, accuracy, and user immersion.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS AND ABBREVIATIONS

VR- VIRTUAL REALITY

AR- AUGMENTED REALITY

HCI- HUMAN COMPUTER INTERACTION

UDP- USER DATAGRAM PROTOCOL

CHAPTER 1
INTRODUCTION

Hand-tracking technology allows a user to trigger and control objects virtually using their hand rather than any manipulator. The project based on a MediaPipe Hand Model, OpenCV, CVZone and Unity has developed an interactive system. In contrast to gesture-based systems, this system converts users' hand movements into a 3D environment in real time so that the user can interact without a gesture. Using computer vision and game development software, it makes the experience seamless and responsive that enhances the applications like virtual reality (VR), augmented reality (AR), interactive simulations, etc. We capture your hands in real-time through a webcam, frames are processes by MediaPipe to detect hand landmark, sending coordinates to Unity via UDP connection. Unity translates and maps hand movements to control 3D objects and engage with interaction. Due to its high accuracy and low latency, it provides a natural and intuitive interaction mechanism. The project can be useful in gaming, education, remote collaboration, and digital art, allowing the user to interact with 3D virtual environments using their hand without any external devices.

1.1 PROBLEM STATEMENT

The traditional way of communicating or interacting with digital systems or computers via an external controller, keyboard, touch screen, etc. delays the natural interaction between humans and machines. Many existing solutions using hand tracking, although more natural and intuitive to interact with, are still faces issues related to accuracy, latency, depth perception, and versatility.

1.2 AIM OF THE PROJECT

The project incorporates accurate hand landmark detection using Mediapipe, enabling precise tracking of hand movements. It ensures smooth data transmission between Python and Unity, facilitating seamless integration for real-time 3D visualization. Unity is utilized to render realistic, low-latency hand movements, enhancing the user's immersive experience within the virtual environment. Additionally, the system is designed with scalability in mind, allowing for future enhancements such as sign language interpretation and gesture-based command functionality.

1.3 PROJECT DOMAIN

The domain of the project is Computer Vision and Human-Computer Interaction (HCI), particularly emphasizing their applications in real-time 3D hand tracking and virtual environment interaction using machine learning frameworks.

1.4 SCOPE OF THE PROJECT

The scope of real-time 3D hand tracking system in virtual environments (without gesture control) using MediaPipe, OpenCV, CVZone is that it helps users to see the movement of their hand in a 3D space and interacting with the objects thus creating an immersive experience. The system can be used in training simulations dealing with virtual reality, analysis of hand movement, art and design, rehabilitation in medical field, educational tools, etc. where gesture control is not required. By eliminating the difficulty of gesture control, we let the focus shift towards smooth and precise real-time tracking. Thus, the system becomes lightweight, easy to implement, and platform-independent using any standard webcams thus making it affordable and accessible.

1.5 METHODOLOGY

The proposed system employs a real-time 3D hand tracking approach using a webcam and MediaPipe to detect and process hand landmarks. Initially, a webcam captures live video which is processed using OpenCV and MediaPipe to extract 21 hand landmark coordinates. These coordinates are refined through CVZone for better tracking performance and then transmitted to the Unity engine via UDP sockets, ensuring low-latency communication. Unity receives and parses this data to animate a 3D virtual hand that mimics the user's real-time movements. Users can interact with virtual objects, pushing and manipulating them within the simulated environment. This methodology avoids predefined gestures and instead enables smooth and natural hand motion tracking, enhancing responsiveness and immersion. The integration of Python scripting (via PyCharm) and Unity for 3D rendering results in an affordable, scalable, and flexible human-computer interaction framework suited for VR, AR, simulation, gaming, and educational applications.

1.6 ORGANIZATION OF THE REPORT

Chapter 1 introduces a real-time 3D hand-tracking system that enables natural interaction with virtual environments without gesture control. Using MediaPipe, OpenCV, CVZone, and Unity, the system captures hand movements via webcam, processes them for accurate tracking, and transmits data to Unity for 3D visualization. The project aims to improve accuracy, reduce latency, and enhance immersion in applications like VR, AR, and gaming. The methodology outlines a scalable, accessible framework for interactive human-computer interactions.

Chapter 2 Contains Literature review of relevant papers.

Chapter 3 examines the proposed system offers a real-time 3D hand tracking solution that eliminates the need for predefined gestures or external hardware, enabling intuitive and accurate interactions within virtual environments using only a standard webcam. By leveraging open-source tools like MediaPipe, OpenCV, CVZone, and Unity, the system achieves low-latency tracking (~17.12 ms) and high accuracy with 21 hand landmarks, ensuring smooth, stable hand movements. This approach makes it cost-effective, as it requires no specialized hardware, and is suitable for a wide range of applications in fields like gaming, education, and healthcare. The system's scalability, cross-platform compatibility, and accessibility for users with physical disabilities further enhance its appeal, offering an inclusive and immersive experience with minimal computational requirements. Feasibility studies confirm its practicality from economic, technical, and social perspectives, making it an affordable and efficient solution for real-time human-computer interaction in virtual environments.

Chapter 4 outlines the proposed work, starting with real-time 3D hand tracking solution that integrates OpenCV, MediaPipe, CVZone, and Unity to enable natural interaction within a virtual environment. A webcam captures live video, which is pre-processed to enhance clarity before MediaPipe detects and tracks 21 hand landmarks in 3D. The landmark data is streamlined by CVZone and transmitted to Unity using the UDP protocol. In Unity, the coordinates are mapped onto a virtual hand model, allowing it to mimic the user's hand movements in real time. The system supports object interaction and VR integration, making it suitable for immersive applications in gaming, training, and simulations.

Chapter 5 details the implementation and testing of the 3D hand tracking system. It begins by demonstrating how the system accurately detects hand orientation from various angles and maps them into a virtual 3D environment. Through Unity simulations, users can see a virtual hand model with tracked landmarks interacting with objects like cubes, showcasing real-time responsiveness and natural motion. Testing was conducted at multiple levels—unit testing verified each module individually, integration testing ensured seamless communication between components, and functional testing validated the system's performance in real-world scenarios. Overall, the chapter confirms that the system delivers precise, real-time hand tracking and object interaction, supporting immersive applications.

Chapter 6 presents the performance outcomes and evaluation of the 3D hand tracking system. The results demonstrate that tracking accuracy improves rapidly over time, reaching up to 98% after 350 frames, showcasing system stability and efficiency. Evaluation metrics such as Detection Success Rate, Landmark Instability Score, and Latency confirm the system's high detection reliability, smooth landmark tracking, and minimal processing delay of just 17.12ms. A comparative analysis with existing systems highlights the proposed method's superior real-time responsiveness and stability, making it highly effective for immersive and interactive applications in virtual environments.

Chapter 7 concludes the project by highlighting the effectiveness of the real-time 3D hand tracking system in enabling natural, controller-free interaction within virtual environments. The system, built using MediaPipe, OpenCV, CVZone, and Unity, delivers precise motion tracking with low latency (17.12 ms), high detection success rate (~90%), and smooth landmark stability. It allows intuitive interaction with virtual objects without predefined gestures, confirming its potential in VR, AR, and simulation contexts. The chapter also outlines future enhancements such as multi-hand tracking, gesture recognition, improved physics, occlusion handling, AI prediction, mobile/AR deployment, multi-user support, accessibility features, haptic feedback, and performance optimization—aimed at expanding usability, realism, and interactivity for broader real-world applications.

Chapter 8 contains the source code and details about the poster presentation. It includes sample code snippets for reference

CHAPTER 2
LITERATURE REVIEW

V. Chunduru et al [1] present a system for Hand Tracking in 3D Space that utilizes MediaPipe along with the PnP (Perspective-n-Point) Method. This research highlights enhanced tracking precision, fluid virtual gestures, and capabilities for real-time hand tracking. It significantly improves interaction within virtual environments, making it highly beneficial for VR applications, gaming, and digital simulations. However, it is constrained by the limitations of MediaPipe, has difficulty with rapid hand movements, and struggles to track fingers accurately in situations where they are occluded or obstructed.

Gangurde R et al [2] propose an Air Canvas system employing MediaPipe is put forth, integrating MediaPipe, Unity, and computer vision techniques. This system lets you draw in the air while being tracked in real-time. This offers an innovative way to produce digital art, which requires no traditional devices such as styluses or mice. Even so, it has trouble making the stroke precise. Further, it is also unable to track complex movements of the hand. For example, when the hand rotates too much or goes outside the camera, accuracy and functionality lessen.

Y. Che et al [3] this paper presents a Detection-Guided 3D Hand Tracking system that can run on mobile devices. The method relies on detection to estimate hand pose accurately. The shows resilience to occlusions, improved real-time tracking, and making it suitable for mobile applications like AR so one can interact smoothly. But it relies greatly on the mobile hardware and degrades in lowlight situations, hampering tracking accuracy and latency.

Naganandini. k et al [4], describes a system for Hand Tracking-based Human-Computer Interaction for educational contexts based on gestures in teaching contexts. One important advantage is in making online learning experiences more enjoyable with gesture-based teaching applications. Despite that, it has a limited training dataset and the gesture performance might vary in other cases and thus tweaked and trained more on various cases.

C. Tsai A et al [5] showcasing a mini drone that uses 3D gesture recognition technology and Unity. The study stresses real-time processing, accurate gesture control, and interactive human-drone communication. Yet, this is not a very effective solution as it can only recognize a limited number of predefined gestures.

Reimer. D et al [6] this paper presents a real-time hand tracking approach for AR interfaces assisted by deep learning. The system ensures that hand tracking is reliable, that augmented reality interaction is smooth, and that latency is low which makes it very good for use in gaming, training, and AR applications. But, it Needs Heavy Computing Power and High-End GPU which isn't consumers friendly and Increase hardware dependencies.

N. Voigt-Antons et al [7] the given paper presents Gesture Based Virtual Painting using Hand Tracking and Unity through which digital art can be made was focused. Real time stroke changes and involvement of user for the painting in virtual environment were also focused in the paper. This system enhances creativity and presents a fresh approach for digital canvas development. But, difficulties of restricted colors, difficulty in exact monitoring of hand rotations, sensitivity to lighting conditions which affects the art one produces and usability.

F. Mueller et al [8] the paper presents an AI-Driven Hand Gesture Recognition system based on a CNN for the classification and prediction of gestures for sign language interpretation. The main advantage is to improve accessibility for people with hearing disabilities, high accuracy for sign language recognition and multilingual sign detection. Despite its advantages, the system needs large datasets for training and suffers from overlapping gestures that leads to failure in recognizing signs. It also fails in a cluttered environment.

C. Elbrechter et al [9] studies the Adaptive Hand Tracking for gaming applications via Machine Learning techniques. The achieves a higher degree of accuracy that can be customized depending upon personal requirements. However, it requires lots of calibrations for different sizes of hands and doesn't work well in dark and fails in various gaming scenarios, especially fast-paced ones.

Oliveira et al [10] introduces a system that enhances remote collaboration through Hand Tracking in virtual meetings. The study has shown the use of intuitive gesture-based controls, improved interaction with virtual objects, and more naturalness in remote communication. When a virtual meeting consists of many more people or when teams from different locations work together, issues like network delay, hardware reliance, and coordination can come up.

H. Sampson et al [11] a study on Hand Gesture Recognition for medical application using Deep Learning and Computer Vision in healthcare. It highlights its advantages in operation simulations, rehabilitation of patients, and telemedicine to reduce hands-free operations. The medical field requires things to be very precise and reliable, especially for real-time applications.

S. Geetha et al [12] a recent research study which has appeared on Multi-View Hand Tracking for Robotic Control will be discussed here along with its new approach which will be using Stereo Vision and a few artificial intelligence algorithms. The system improves the robotic handling of objects, increases grasping precision, and encourages human-robot collaboration in industrial and automation environments. But it needs huge computational requirements, and it is sensitive to occlusions which makes it less practical for real-life use with dynamic environment constraints.

C. R. Cameron et al [13] this paper presents a Real-Time Hand Tracking for Virtual Training Environments, aimed at creating immersive training simulations for various sectors including aviation, healthcare and industrial training. The study shows that training is better when you can use your hands in simulation so the hands-on task is enhanced greatly for proper task application. But, problems like motion blur effects, high processing requirements, and the need for powerful hardware can make smooth interactions harder according to the training scenarios.

Shravya et al [14] the authors present a system for AI-Assisted Hand Tracking (AI-AHT) for interactive storytelling with a focus on Natural Interaction, adaptability from AI, and VR Narratives. Interactive storytelling is possible through modifying the story according to user gestures. This will enhance user audience engagement and immersion Yet, it has to face the challenges of gesture misinterpretation along with requiring high-quality datasets for better performance, making it hard to generalize across various storytelling methods.

Finally, D. Yang et al [15] it provides a comparison of hand tracking methods for medical training simulators. It reveals how accurate and good the simulations must be and specifies which technique is best to apply for which scenario. The research compares three different kinds of techniques of hand tracking, which involve marker based, marker less and deep learning-based hand tracking techniques. These comparisons can be useful for future work design in medical training simulations.

As observed from the above analysis, although existing hand tracking systems have benefits such as real-time interaction and gesture control and they make technology more affordable, hand tracking systems tend to have limitations which restrict their usability. To solve these problems, our proposed method will guarantee gesture less control which is much accurate, adaptable, and efficient outputs and be used for different real-time applications.

Author (Ref No, Year)	Method	Data	Inference
Chunduru et al. [1], 2021	MediaPipe + PnP	Hand landmarks	Effective for real-time virtual globe control
Gangurde et al. [2], 2023	MediaPipe	Hand landmarks for drawing	Enables intuitive gesture-based drawing in Unity
Che and Qi [3], 2021	Detection-guided 3D tracking	Mobile AR hand data	Enhances robustness in mobile AR hand tracking
Naganandhini et al. [4], 2022	MediaPipe + HCI system	Gesture input data	Useful in educational HCI systems for hand-based input
Tsai et al. [5], 2020	3D gesture recognition in Unity	Gesture control for drones	Demonstrates reliable drone control via 3D gestures
Reimer et al. [6], 2023	Ownership estimation in VR	Tracked hand positions	Improves multi-user VR experiences
Voigt-Antons et al. [7], 2020	User experience study	VR hand tracking interaction	Positive impact on immersion and interaction
Mueller et al. [8], 2017	Egocentric RGB-D sensor	Hand under occlusion	Achieves real-time tracking even with occlusions
Schröder et al. [9], 2012	Color glove tracking	Glove color patterns	Enables precise control of robotic hands
de Castro et al. [10], 2021	VR hand tracking	VR interaction data	Facilitates immersive VR interactions
Sampson et al. [11], 2018	Hand gesture set	Defined gesture commands	Provides a gesture vocabulary for VR/3D navigation
Geetha et al. [12], 2024	Hand tracking in MR	Mixed reality input	Bridges hand interaction in MR environments
Cameron et al. [13], 2011	VR simulation hand tracking	Hand simulation data	Supports hand visualization in virtual simulations
Shravya et al. [14], 2022	3D tracking with Unity	Virtual hand data	Enables real-time gesture recognition in 3D environments
Yang et al. [15], 2022	Marker-enhanced + DL	Marker-based hand tracking	Improves reliability of hand tracking using deep learning

Table 2.1 Literature Review

CHAPTER 3

PROJECT DESCRIPTION

As observed from the above analysis, although existing hand tracking systems have benefits such as real-time interaction and gesture control and they make technology more affordable, hand tracking systems tend to have limitations which restrict their usability. To solve these problems, our proposed method will guarantee gesture less control which is much accurate, adaptable, and efficient outputs and be used for different real-time applications.

3.1 EXISTING SYSTEM

The existing hand tracking systems primarily rely on gesture-based recognition or external hardware like controllers, gloves, or sensors to interact with virtual environments. Many of these systems use predefined gestures to trigger specific actions, which can limit natural interaction. Though some use MediaPipe or similar frameworks, they often face challenges such as accuracy drops during fast movements, occlusions (e.g., when fingers overlap), and sensitivity to lighting conditions. Furthermore, gesture recognition can sometimes be unintuitive and difficult for users to learn and perform consistently. Existing systems also tend to have higher latency and may require significant computational resources, making them less suitable for real-time applications, especially on low-end hardware. Overall, current solutions may lack the flexibility, responsiveness, and immersive feel required for fluid human-computer interaction in VR/AR settings.

3.2 PROPOSED SYSTEM

Natural and immersive interaction in virtual environments has been demanded in recent years, especially for applications such as virtual reality (VR), gaming, simulation training, human-computer interaction, etc. Users typically interact with VR environments primarily through basic body language. To overcome this limitation, the project proposes a real-time 3D hand tracking system that enables users to intuitively interact with digital objects using their bare hands. The system tracks the user's hand and mimics the action in the virtual space. It will capture video of the hand and then system will process it to locate the hand and create a virtual hand. Unlike gesture recognition systems that look for specific hand signs, this system uses full tracking of the user’s hand position and orientation to allow free-form interaction.

The system integrates multiple modules for smooth and accurate tracking developed by OpenCV, MediaPipe, CVZone and Unity game engine. The palm of the hand is captured through the webcam in real-time. MediaPipe is used to detect palm and 21 landmark points on the hand. The 21 landmarks are processed through CVZone for better tracking and is later converted for use in Unity via UDP, which is low latency transmission. The data of hand tracking which we are getting in Unity is animated on a 3D hand which imitates the user's movement. The system can also enable simple moving and pushing of objects along with the virtual environment. This system shows a practical, scalable way to engage in the virtual world without a controller, potentially impacting such areas as VR interaction and gaming.

3.2.1 ADVANTAGES

The system offers gesture-less natural interaction, allowing users to move their hands freely without relying on predefined gestures, resulting in more intuitive and fluid virtual experiences. It requires no external hardware, functioning solely with a standard webcam, eliminating the need for gloves, sensors, or other expensive equipment. Leveraging UDP and MediaPipe, it ensures real-time processing with minimal latency (~17.12 ms), delivering smooth and responsive tracking. With high accuracy and stability, it reliably detects 21 hand landmarks using MediaPipe, ensuring jitter-free motion. Built with Python and Unity, the system is cross-platform and scalable, supporting future enhancements. Its integration with Unity enables real-time object interaction, making the virtual environment more immersive and engaging. Additionally, it is cost-effective and accessible, using open-source tools and basic hardware suitable for educational and personal use. The system is also highly customizable, adaptable for diverse applications such as gaming, education, healthcare, and digital art.

3.3 FEASABILITY STUDY

A feasibility study assesses the practicality and potential success of the proposed system from different perspectives—economic, technical, and social—to ensure the solution is viable and beneficial in real-world scenarios.

3.3.1 ECONOMIC FEASIBILITY

The proposed system is highly economical as it uses open-source libraries like MediaPipe, OpenCV, and Unity, and requires only a standard webcam. By eliminating the need for specialized hardware, it reduces development and deployment costs, making it a budget-friendly option for both personal and institutional use.

3.3.2 TECHNICAL FEASIBILITY

Technically, the system is feasible as it integrates well-established tools and frameworks such as Python, MediaPipe, and Unity. It achieves real-time hand tracking with low latency and high accuracy using widely available computing resources, making the implementation technically sound and achievable without high-end infrastructure.

3.3.3 SOCIAL FEASIBILITY

Socially, the system promotes inclusivity and accessibility by allowing natural, controller-free interaction, benefiting users with physical disabilities or limited mobility. Its applications in education, gaming, and rehabilitation enhance engagement and interaction, making it well-accepted across various user groups and societal settings.

3.4 SYSTEM SPECIFICATION

The system specification outlines the essential hardware and software requirements, along with relevant standards and policies, necessary for the successful development and operation of the hand tracking system.

3.4.1 HARDWARE SPECIFICATION

The hardware requirements for the system include a minimum of an Intel i5 processor or its equivalent, with at least 8 GB of RAM to ensure smooth performance. For graphics, an integrated GPU is sufficient for basic usage, while a dedicated GPU is recommended for enhanced visual performance. A standard HD webcam is required for real-time hand capture, and the system should have at least 500 MB of free storage space.

3.4.2 SOFTWARE SPECIFICATION

On the software side, the system is compatible with Windows 10 or 11 operating systems. It is developed using Python 3.x, with essential libraries and tools including MediaPipe for hand tracking, OpenCV for video processing, CVZone for simplified landmark handling, and Unity for 3D visualization. Development is typically carried out using PyCharm or any suitable Python IDE. A UDP networking protocol is utilized for fast and real-time data transmission between Python and Unity.

3.4.3 STANDARDS AND POLICIES

Regarding standards and policies, the project adheres to PEP8 guidelines for Python coding to ensure clean and readable code. The system respects data privacy by not storing any personal information or logging tracking data. All libraries used are open-source and comply with their respective licenses, such as MIT and Apache 2.0. The system is also designed with user accessibility in mind, particularly supporting users with physical disabilities and following best practices in Human-Computer Interaction (HCI). Data communication occurs locally via UDP, ensuring security without reliance on an internet connection.

CHAPTER 4

PROPOSED WORK

4.1 GENERAL ARCHITECTURE

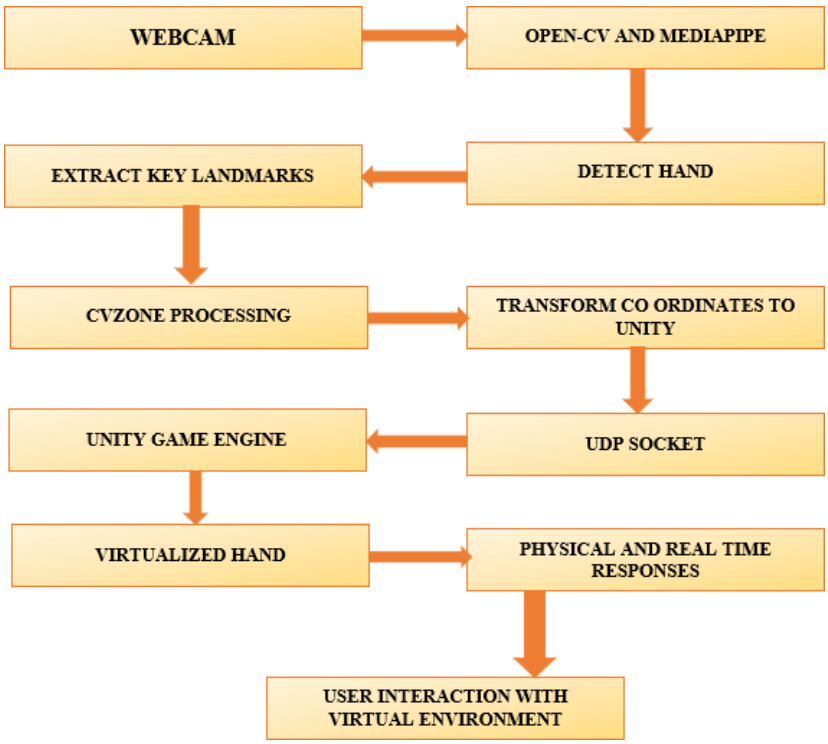

Fig 4.1 Block Diagram

The Fig 4.1 illustrates the block diagram for 3D hand tracking scheme in a virtual environment using OpenCV that the system enables 3D hand tracking in a virtual environment using a combination of OpenCV, MediaPipe, CVZone, and the Unity game engine. A webcam captures live video, which is processed to detect hand landmarks using OpenCV and MediaPipe. CVZone simplifies interaction by managing the landmark data, which is then converted into Unity-compatible coordinates and sent via UDP. Unity renders a virtual hand that mirrors the user's real-time hand movements, allowing natural interaction with digital objects in gaming, simulations, and immersive experiences.

4.2 DESIGN PHASE

4.2.1 FLOW DIAGRAM

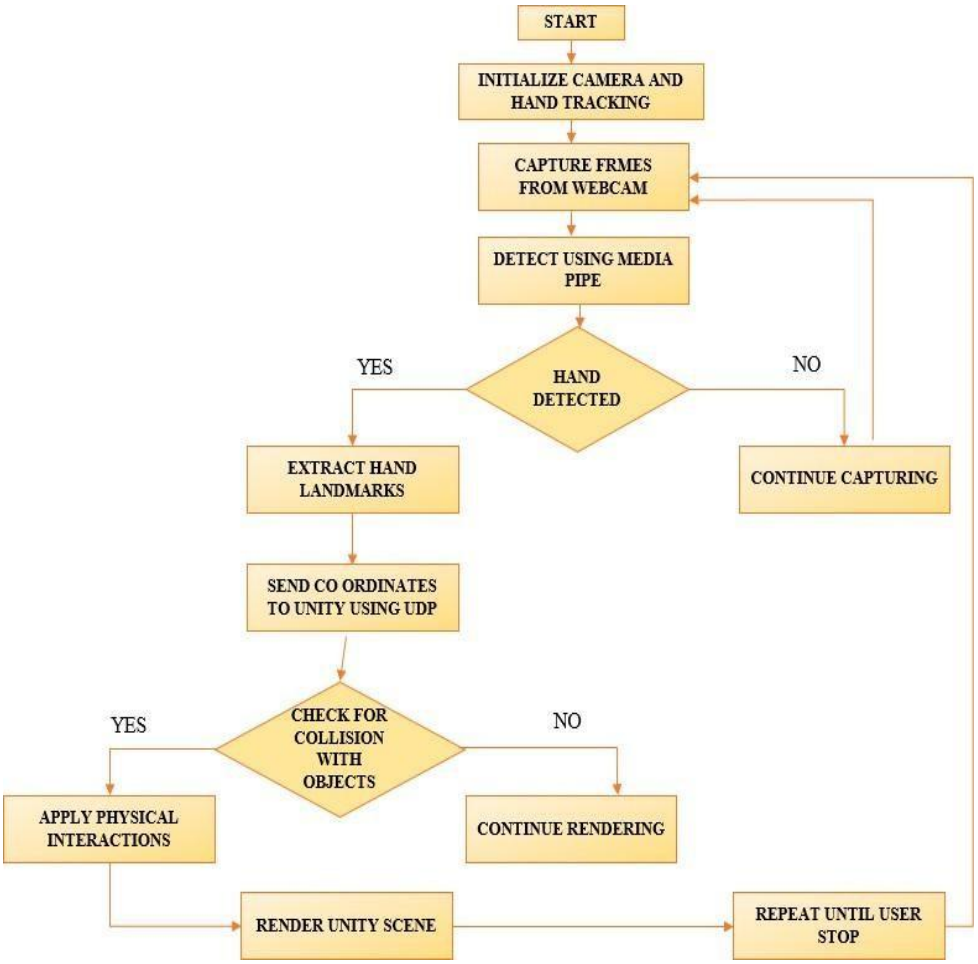

Fig 4.2 Flow Diagram

The Fig 4.2 shows the flowchart that describes how 3D hand tracking in an area is done with a webcam and MediaPipe hand recognition and also Unity for rendering the virtual scene. The system enables real-time 3D hand tracking using MediaPipe and Unity. It starts by capturing webcam frames and processing them to detect hand landmarks with MediaPipe. These coordinates are sent to Unity via UDP. If no hand is detected, the system continues scanning until one appears. Once detected, the hand landmarks enable interactions with virtual objects, including collision detection and simulated physical responses. Unity updates the scene in real time, allowing users to interact naturally with virtual elements.

4.2.2 UML DIAGRAM

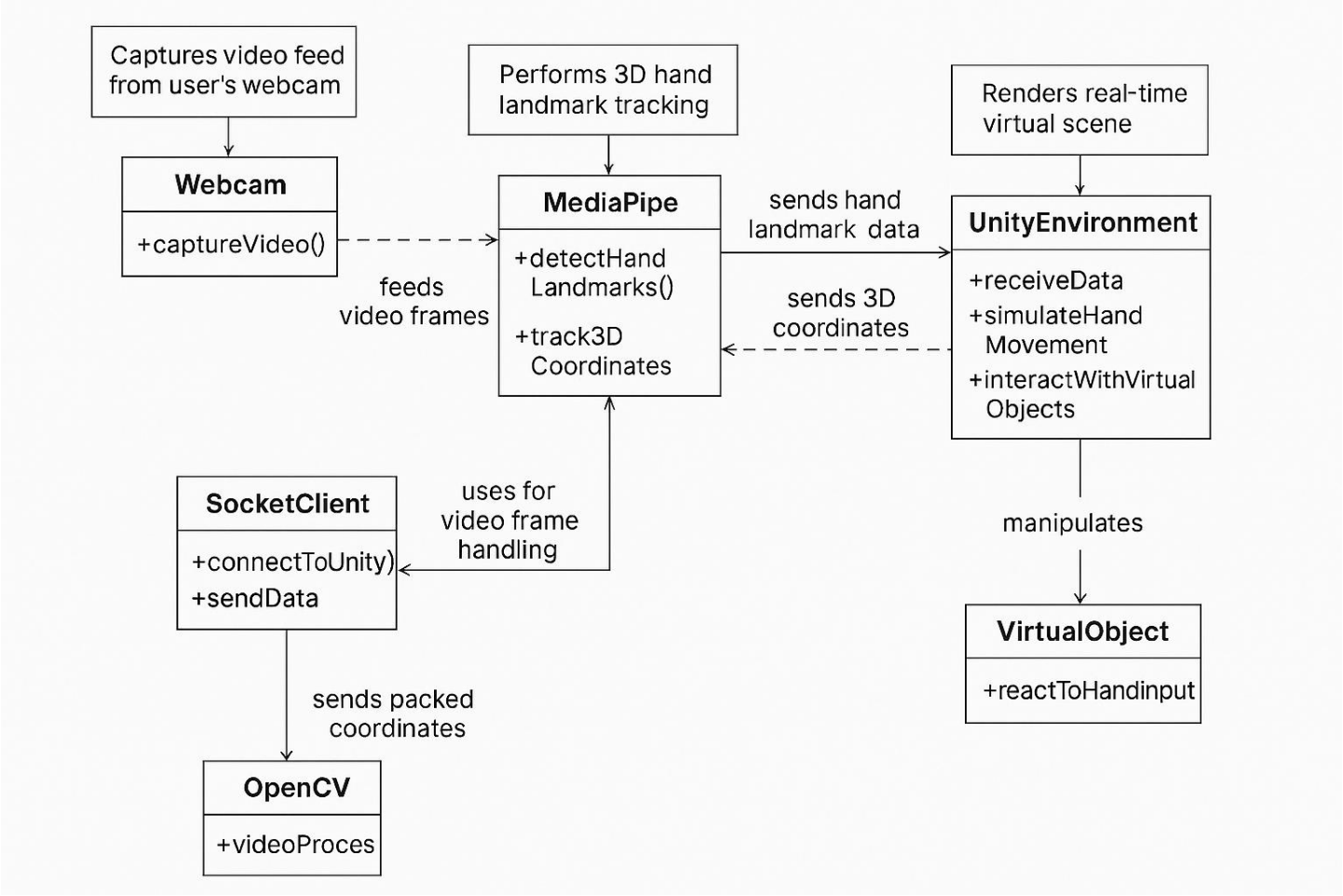

Fig 4.2 UML Diagram

The Fig 4.2 shows a system pipeline for real-time 3D hand tracking and interaction. Webcam input is processed using OpenCV and MediaPipe to detect hand landmarks. CVZone simplifies landmark handling in Python, which is sent to Unity using socket programming. A virtual 3D physics engine enhances landmark data for realistic 3D hand simulation and interaction with objects in Unity's 3D environment, enabling immersive virtual hand control and manipulation.

4.2.3 USECASE DIAGRAM

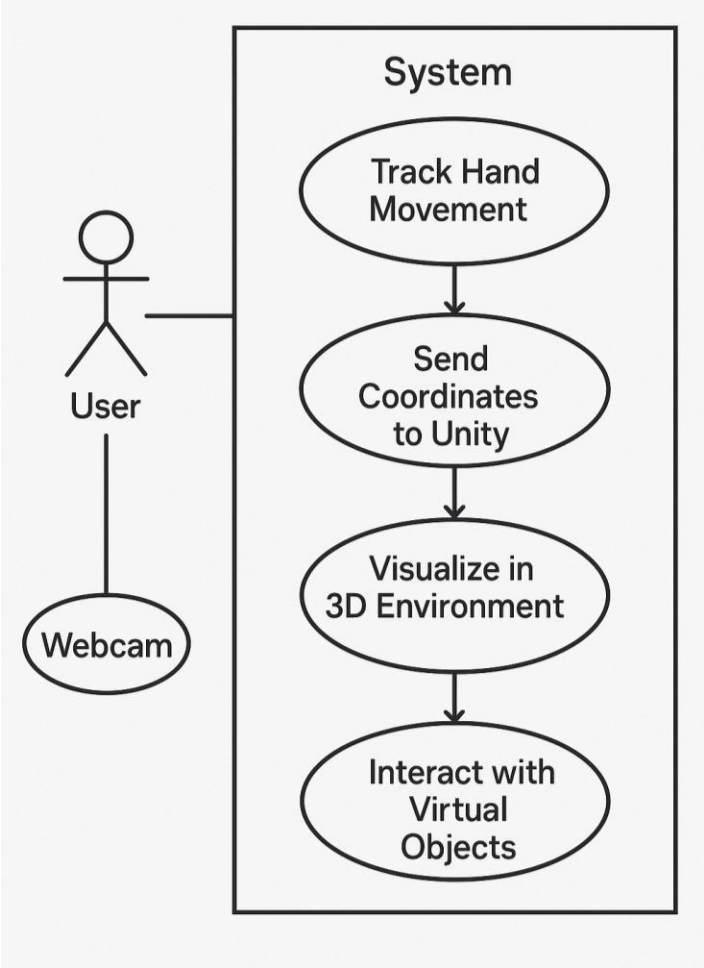

Fig 4.4 Use Case Diagram

The Fig 4.4 outlines a user-interactive hand tracking system. A webcam captures the user's hand movements, which are tracked by the system. The system extracts the hand's coordinates and sends them to Unity. Within Unity, these coordinates are visualized as a 3D hand model. The user can then interact with virtual objects in the 3D environment, enabling real-time simulation and control using natural hand movements for immersive interaction.

4.2.4 SEQUENCE DIAGRAM

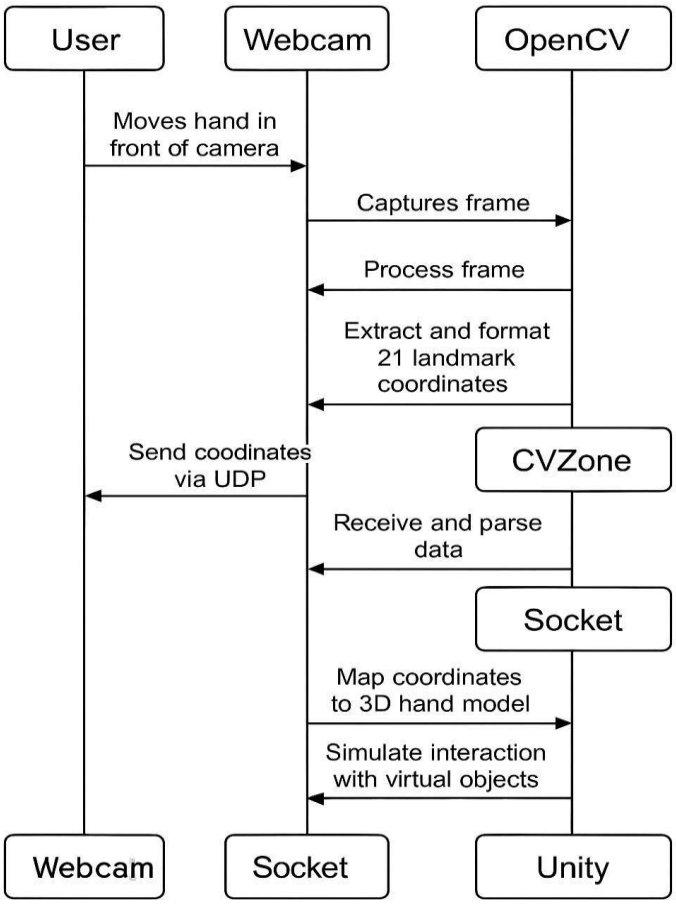

Fig 4.5 Sequence Diagram

The Fig 4.5 sequence diagram shows how a user's hand movement is captured and simulated in a virtual 3D environment. The webcam captures frames as the user moves their hand. OpenCV processes the frames, and CVZone extracts and formats 21 hand landmark coordinates. These coordinates are sent via UDP using sockets to Unity. Unity receives and maps the data to a 3D hand model, enabling realistic interaction with virtual objects in real time for an immersive experience.

4.3 MODULE DESCRIPTION

4.3.1 VIDEO CAPTURE

This module is responsible for capturing real-time video using a webcam. It continuously streams frames that are used as the input for the hand tracking system. The resolution and quality of the video feed significantly impact the effectiveness of the detection and tracking process. High-quality video ensures that fine features like finger joints, fingertips, and palm contours are visible and distinguishable. A stable and well-lit feed enables the system to consistently identify and follow the user's hand. This reliable input feed forms the base of the entire system, ensuring all subsequent stages perform accurately and efficiently.

4.3.2 PRE-PROCESSING

Pre-processing prepares each video frame before it is analyzed for hand detection. This module crops and resizes images to focus on the region containing the hand, reducing unnecessary processing of the background. It also adjusts brightness and contrast to handle variations in lighting, which is critical for maintaining detection accuracy in different environments. Additionally, it applies noise reduction and background filtering techniques to remove visual clutter and highlight the hand clearly. These enhancements make the hand stand out in each frame, allowing the tracking model to detect it more reliably and quickly, improving both performance and accuracy.

4.3.3 HAND TRACKING

This module is the core of the system, responsible for detecting and tracking the user's hand in real time using MediaPipe. It first locates the palm using a trained palm detection model. Once identified, the system uses a hand landmark model to detect 21 key points on the hand, including fingertips, knuckles, and the wrist. These landmarks are given as 3D coordinates (x, y, z), providing an accurate representation of the hand’s structure and motion. This data is essential for translating real-world hand movements into virtual actions and is foundational for interaction in digital environments.

4.3.4 DATA TRANSMISSION

After the 3D hand landmarks are identified, this module handles sending the data to the Unity environment. The information is formatted into a compact and structured string containing the coordinates of each landmark. It uses the UDP (User Datagram Protocol) for data transmission because it provides fast, connectionless communication ideal for real-time applications. Though UDP doesn't guarantee delivery, its speed and low overhead make it suitable for live systems where performance is prioritized. This module ensures that hand tracking data is quickly and efficiently delivered to Unity with minimal delay, preserving the real-time experience.

4.3.5 3D HAND MAPPING

Inside Unity, this module receives the UDP-transmitted hand tracking data and maps the 3D coordinates to a virtual hand model. It converts the raw positional data from the MediaPipe format into Unity's coordinate space, ensuring accurate placement and orientation. Each landmark point is applied to specific joints and fingers on the 3D hand, animating it to mimic real hand motion. The model responds instantly to changes in position, allowing for smooth and lifelike movement. This process creates a seamless and immersive link between the user's physical actions and the virtual hand's behavior.

4.3.6 INTERACTION

The interaction module brings life to the virtual hand by allowing it to engage with objects in the 3D environment. It supports basic manipulations like touching, moving, and pushing objects, enhancing realism and immersion. When the virtual hand contacts items in the Unity scene, collision detection and physics simulations allow those objects to react naturally. This module also supports VR integration, enabling users to explore and interact in immersive environments without controllers. By translating physical hand movements into digital actions, it provides intuitive control, making the system ideal for simulations, games, art, and training.

4.4 HAND TRACKING USING MEDIAPIPE

4.4.1 MEDIAPIPE LIBRARY

MediaPipe, Google's open-source framework that enables developers to build cross-platform, multimodal machine learning pipelines. It is very popular in real-time computer vision applications such as face detection, pose estimation, and hand tracking. The library contains a lot of pre-trained models that make it easy to incorporate advanced ML models into your applications. MediaPipe works very well on desktops, mobile phones, and web browsers (with JavaScript), which makes it practical in real life.

4.4.2 PALM DETECTOR MODEL

The hand tracking solution in MediaPipe begins with a palm detection model, which serves as the first stage in the pipeline. Instead of detecting the entire hand, the model focuses on detecting the palm, which is more consistent in shape and orientation than fingers. This makes detection faster and more reliable. Once a palm is detected, the model returns a bounding box around the palm along with handedness information, indicating whether the detected hand is left or right. The palm detector ensures that the system only proceeds to the next stage if a valid hand region is found, thus optimizing performance.

4.4.3 HAND LANDMARK MODEL

Once the palm is recognized, the next step involves a hand landmark model. The next step involves a hand landmark model that takes the cropped image of the palm and predicts 21 landmark points, which represent the various joints and tips of the fingers and thumb including the wrist. The 21 landmarks predicted in this module are the joints and tips of all fingers and the thumb, as well as palm and wrist. This landmark model is light-weight and runs in real-time so that you can have user interaction across platforms.

CHAPTER 5

IMPLEMENTATION AND TESTING

5.1 INPUT AND OUTPUT

5.1.1 HAND DETECTED IN DIFFERENT ORIENTATION

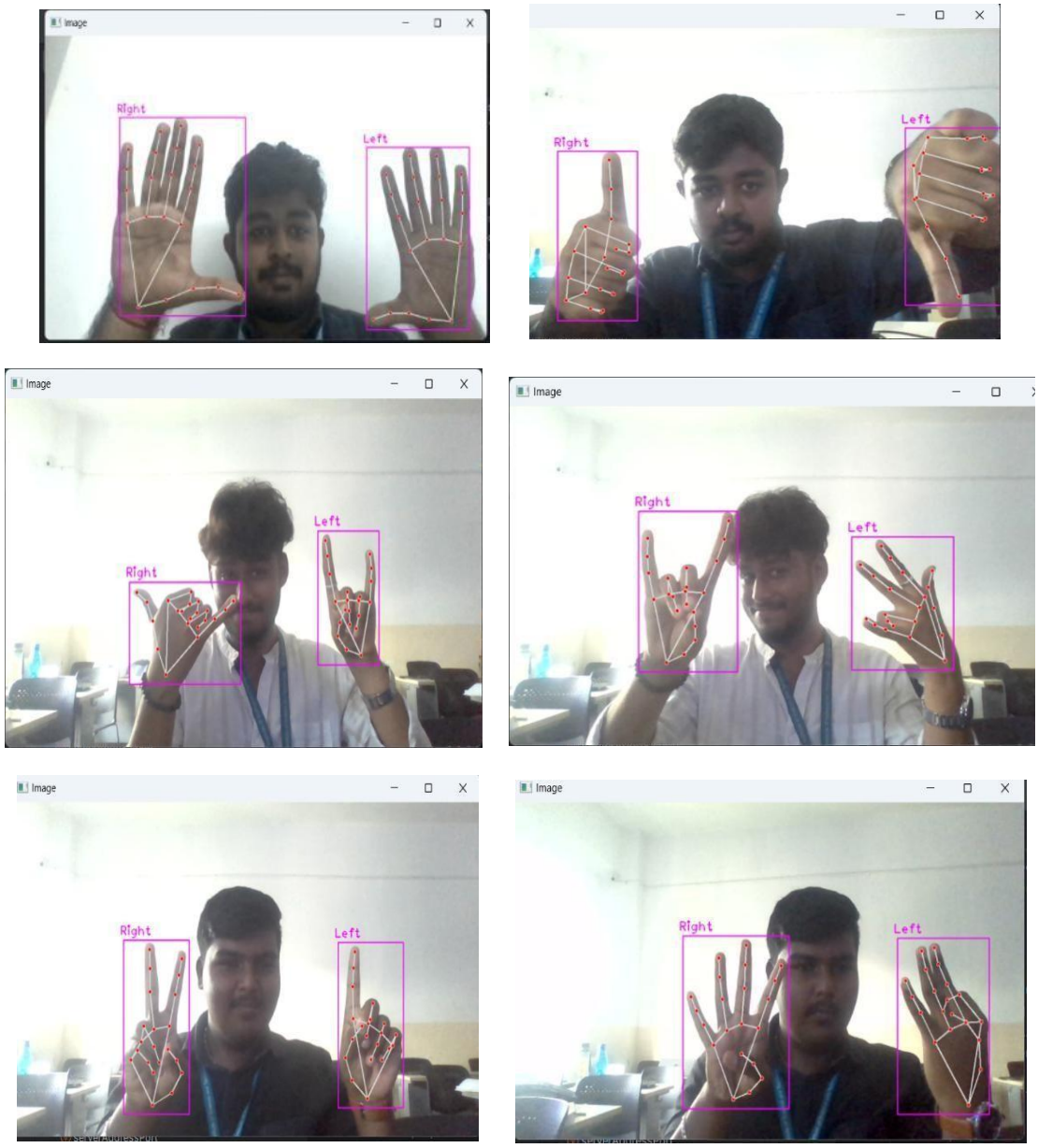

Fig 5.1 Hand detected in different Orientations

5.1.2 SIMULATION

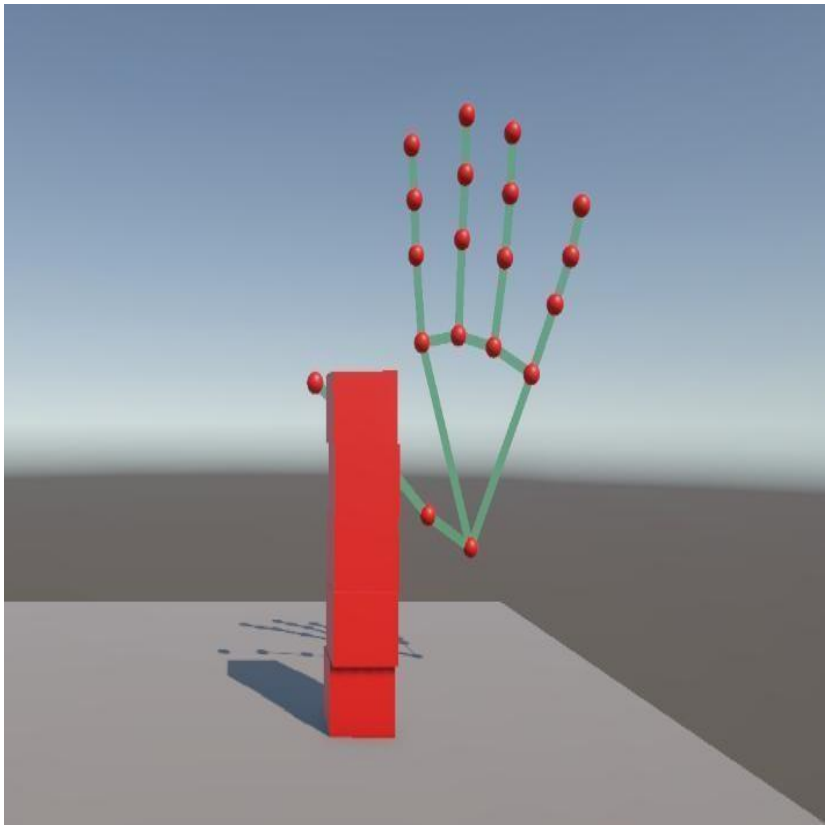

Fig 5.2 Simulated Hand

The image depicts a 3D virtual hand model within a Unity environment, showcasing real-time hand tracking using MediaPipe. The hand is visualized through 21 red spheres representing the landmark points of the fingers, knuckles, and wrist, connected by green lines that form the skeletal structure of the hand. In front of the hand is a red cube, likely used for demonstrating interactive capabilities such as touching or pushing virtual objects. Set on a flat gray surface with a simple horizon background, the side view illustrates how physical hand movements are mapped into virtual space for immersive interaction.

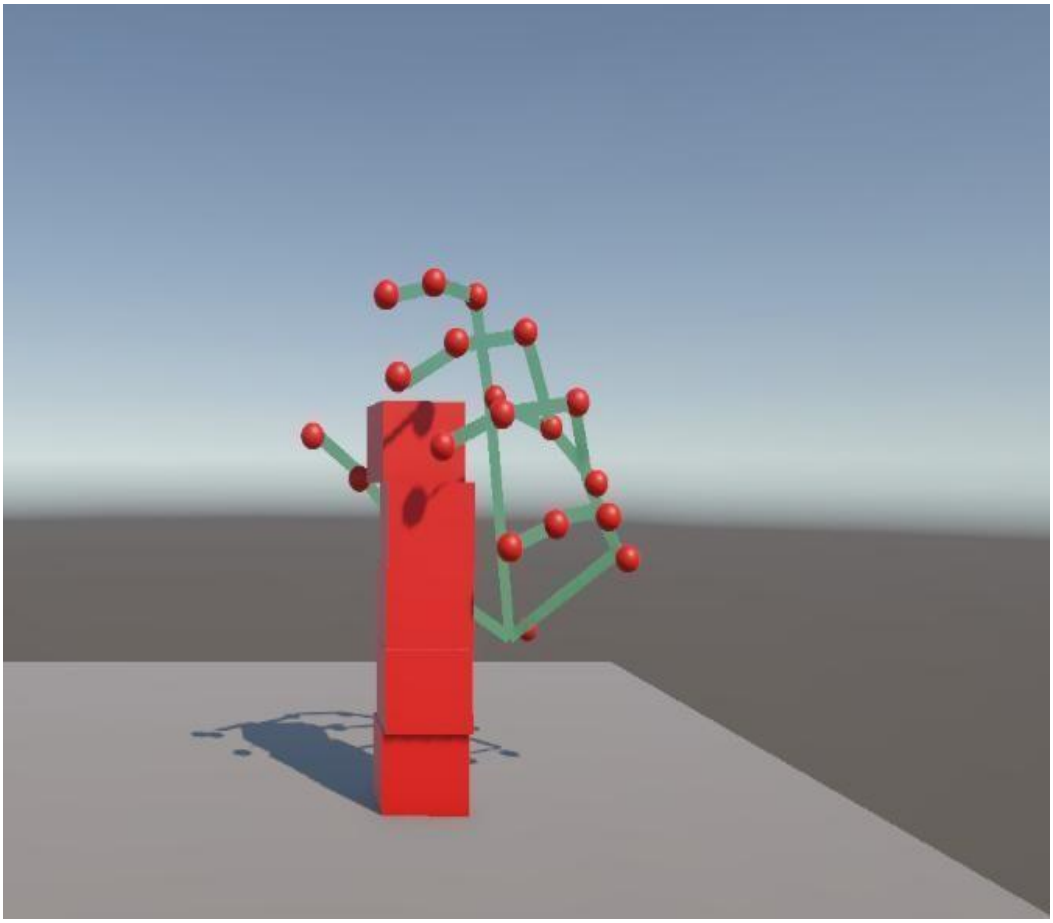

Fig 5.3 Object Interaction with Hand 1

This image shows a dynamic interaction between the virtual hand and a red rectangular object within the Unity environment. The 3D hand model, formed by 21 red spheres connected with green lines, appears to be bending or gripping, indicating a more advanced gesture or movement, such as a partial fist or grabbing motion. The positioning of the fingers suggests that the hand is in close proximity to or in contact with the red object, possibly simulating a real-time touch or grasp. The accurate alignment illustrates responsive and realistic motion tracking.

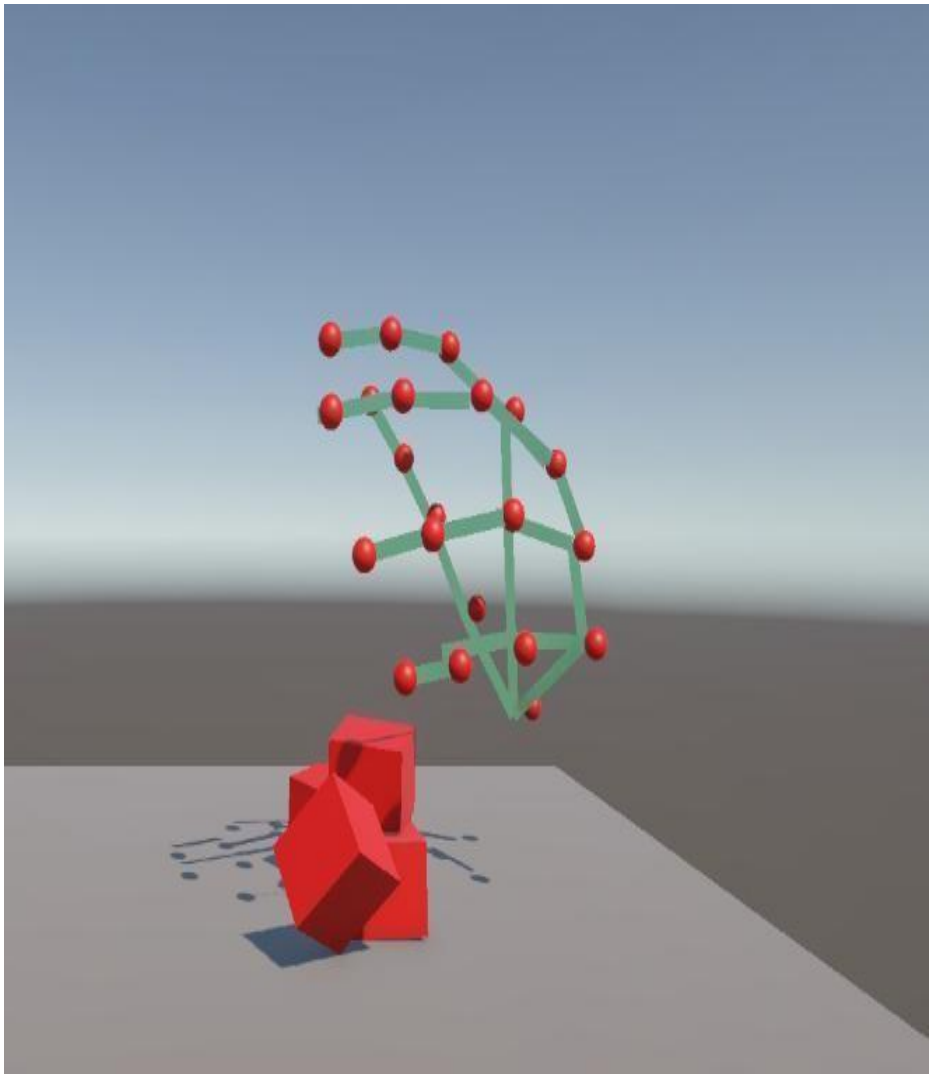

Fig 5.4 Object Interaction with Hand 2

In this scene, the virtual hand model is positioned above a red cube that appears distorted or in mid-motion, indicating interaction. The scattered positioning of the red object suggests it has been pushed or manipulated by the hand. The landmark spheres and connecting lines of the hand form a slightly curved pose, likely representing a pushing or downward motion. This visualization captures the system's ability to not only animate hand movements but also affect objects in the virtual space through real-time physical simulations, enhancing immersion and interactivity.

5.2 TESTING

5.2.1 UNIT TESTING

Unit testing was applied to individual modules to ensure they performed correctly in isolation. The MediaPipe hand tracking module was tested for precise landmark detection. The OpenCV module ensured stable and clear webcam input. The UDP socket transmission was tested for accurate and timely data delivery from Python to Unity. Unity's rendering module was tested for correct animation of the virtual hand. Each component was verified independently to detect early-stage bugs and verify functionality before integration. These tests formed the base for ensuring system stability and reducing complexities during the later stages of development and system integration.

5.2.2 INTEGRATION TESTING

Integration testing ensured that all modules—hand tracking in Python, data transfer via UDP, and visualization in Unity—worked seamlessly together. It focused on validating end-to-end communication, data consistency, and synchronization. This testing identified potential issues such as data loss during transmission, mismatches in coordinate mapping, and frame delays. Test scenarios included continuous and rapid hand movement, temporary hand occlusion, and multiple restarts of the system. By combining individual units into a unified workflow, integration testing ensured that the virtual hand in Unity responded accurately and in real time to the actual hand tracked by the webcam via MediaPipe.

5.2.3 FUNCTIONAL TESTING

Functional testing validated the system's behavior against user expectations and real-world scenarios. It assessed whether the system could detect hand landmarks, send the data correctly to Unity, and render movements as intended. Users tested the system by performing actions like moving hands in different directions, interacting with virtual objects, and changing hand angles. Lighting and background conditions were also varied to evaluate robustness. This testing confirmed that the system could provide smooth, accurate, and intuitive real-time interaction without gesture recognition, thus meeting its goal of creating a controller-free immersive experience suitable for VR, AR, simulations, and educational applications.

CHAPTER 6
RESULT AND DISCUSSION

6.1 RESULT

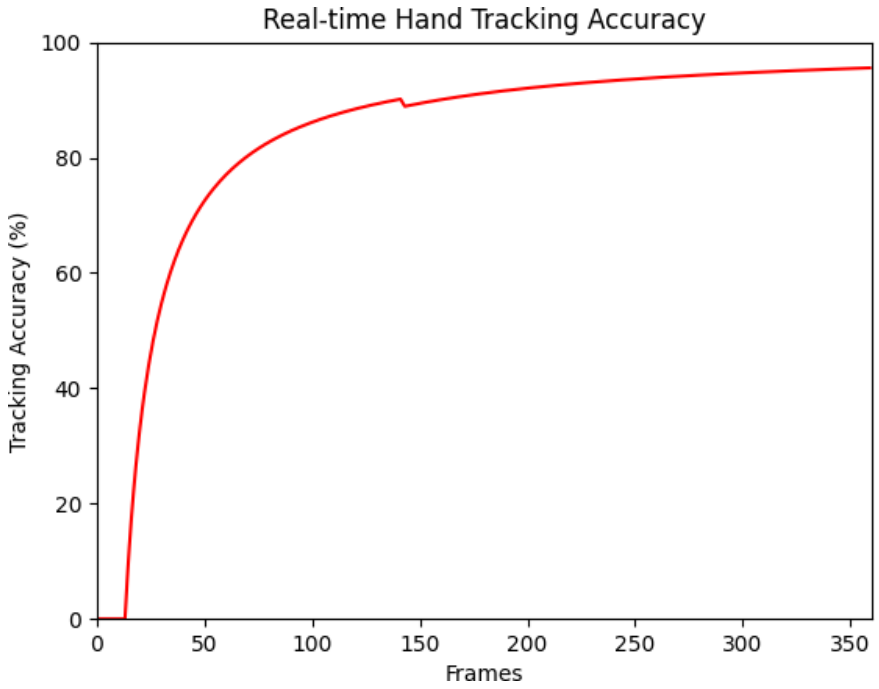

Fig 6.1 Validation Accuracy over Frames

The Fig 6.1 illustrates graph titled "Real-time Hand Tracking Accuracy" illustrates the system's tracking accuracy over a sequence of frames. The x-axis represents the number of frames processed, while the y-axis shows the tracking accuracy in percentage. Initially, accuracy is low due to system warm-up and landmark instability. However, as more frames are processed, the tracking stabilizes rapidly—accuracy increases sharply, reaching around 85% by the 140th frame. After this point, the curve continues to rise gradually, approaching 95-98% accuracy by the 350th frame. This performance indicates that the proposed hand tracking system becomes highly reliable shortly after initialization, maintaining precise tracking throughout real-time use. The smooth curve and high final accuracy reflect the system’s robustness and efficiency in landmark detection using MediaPipe and real-time data processing. This also demonstrates its potential for applications requiring consistent and accurate hand tracking, such as virtual reality, gaming, and immersive simulations.

6.2 ANALYSIS

Metrices	How to measure	Result
Detection Success Rate - DSR	Measures how often hands are detected in frames.	Higher
Landmark Instability	Measures how much hand landmarks move between consecutive frames.	Lower
Latency	Delay between real time and processing time	17.12ms

Table 6.1. Hand Tracking Evaluation Metrics

The table shows a comparison of the suggested hand tracking model on three major metrics. Detection Success Rate (DSR%), Landmark Instability Score, and Latency. The DSR% quantifies the frequency of successful detection of hands in frames, with a greater value representing stable detection. The Landmark Instability Score assesses the displacement of hand landmarks from one frame to the next, with a lower value representing smoother and more stable tracking. Latency is the time it takes for system processing from real-time hand motion, and the model has a low latency of 17.12ms, which promises near real-time performance.

In general, the results reveal that the model gives high detection precision, stable landmark estimation, and low delay, rendering it apt for real-time hand tracking purposes. Additional optimizations would improve responsiveness and eliminate remaining instability frame or occlusions.

Metrices	Proposed system	V. Chunduru et al [1]	Y. Che et al [3]
Detection Success Rate - DSR	High	Moderate (drops under obstruction)	High (strong even under obstruction)
Landmark Instability	Low (stable landmarks with smoothing)	Moderate (depends on lighting & angle)	Low (optimized stability under tracking)
Latency	17.12ms	~24ms	~29 ms

Table 6.2. Comparison of proposed System and Existing System

The comparison shows that our MediaPipe based and Unity created 3D hand tracking system provides a better real-world performance as shown in having a high Detection Success Rate, low Landmark Instability Score and most importantly a low latency of 17.12 ms. Our approach is superior to that in [1] in that it has moderate detection reliability and increased latency (~24 ms) in being more reliable and prompter. [3] claims high accuracy and reliability, even in occlusion, but its ~29 ms latency is a lot longer than yours, indicating your approach is more suitable for low-lag real-time use, especially in virtual applications.

CHAPTER 7
CONCLUSION AND FUTURE ENHANCEMENT

7.1 CONCLUSION

In Conclusion,The Real-Time Human-Machine Interaction in Virtual Environment project showcases a practical and efficient system for natural, controller-free interaction with virtual objects using real-time 3D hand tracking. By integrating MediaPipe, OpenCV, CVZone, and the Unity Game Engine, the system captures hand movements through a standard webcam and translates them into 3D virtual actions, eliminating the need for predefined gesture recognition.

Unlike traditional systems that rely heavily on hardware or gesture interpretation, this project emphasizes gesture-less free-form interaction, enabling smooth and intuitive control through real-time motion tracking. MediaPipe's 21 hand landmark model ensures high precision in identifying hand posture and orientation, while OpenCV and CVZone enhance frame preprocessing and landmark handling. The real-time landmark data is transmitted using UDP, a lightweight communication protocol that significantly reduces latency, with observed delays as low as 17.12 milliseconds, ensuring responsive and fluid interaction.

On the Unity side, the system reconstructs and animates a 3D hand model that mimics the user's physical hand movements. This allows not just motion replication but also interaction with virtual objects—including detecting collisions and applying basic physics, such as pushing or moving objects—within an immersive environment. Performance metrics such as Detection Success Rate (DSR%) reaching up to 90%, low Landmark Instability Score, and real-time feedback confirm the robustness and reliability of the proposed approach.

7.2 FUTURE ENHANCEMENTS

The system offers significant potential for future enhancements across various dimensions. One major extension is multi-hand tracking, which would allow the simultaneous tracking and animation of both hands, enabling bimanual interaction for tasks such as virtual sculpting, detailed object manipulation, or collaborative two-player simulations. Integrating gesture recognition on top of motion tracking would support intuitive commands like swipe, grab, or point, facilitating smoother UI navigation and triggering actions in interactive applications.

Further improvements can be made by enhancing object interaction and physics in Unity. By integrating advanced physics engines, such as Rigidbody manipulation and soft-body physics, users could experience more realistic object dynamics, including grabbing, throwing, and pressure-sensitive responses. To ensure consistent performance in real-world scenarios, the system can be made more robust through occlusion handling and improved environmental adaptability, maintaining tracking accuracy despite partial hand occlusion or changes in lighting and background.

The incorporation of AI-powered prediction models, such as deep learning-based path estimators, could help mitigate brief tracking losses, enhancing overall responsiveness and continuity. The system can also be optimized for mobile devices and AR/VR headsets, including platforms like HoloLens and Meta Quest, thereby expanding its utility for education, fieldwork, and gaming on-the-go.

Additionally**,** multi-user collaborative features could be introduced, allowing multiple users to interact within a shared virtual space using synchronized hand tracking, ideal for remote training or collaborative design. The system also has the potential to support sign language interpretation, creating an inclusive platform for users with speech or hearing impairments.

Another promising enhancement is the integration of haptic feedback through wearable devices like gloves or wristbands, offering a tactile dimension to virtual interaction that deepens immersion. Finally, performance optimization using techniques like data compression, parallel processing, or GPU acceleration would allow for reduced latency, improved frame rates, and smoother operation in larger virtual environments with multiple points of interaction.

7.3 RESULTS

The results of the Real-Time Human-Machine Interaction in Virtual Environment project demonstrate the system’s ability to accurately track and simulate human hand movements in a virtual space with high precision and responsiveness. Using a standard webcam, the system captures live video input and processes it through the MediaPipe framework to extract 21 landmark points representing hand joints and fingertips. These coordinates are then transmitted in real-time via UDP to Unity, where a virtual hand mimics the actual hand's movements.

The system achieved a Detection Success Rate (DSR%) of around 90%, indicating consistent and reliable hand detection across varying frames and orientations. The Landmark Instability Score remained low, showcasing smooth tracking with minimal jitter. A key highlight of the system is its low latency of 17.12 milliseconds, enabling near real-time interaction without noticeable delay. Visual simulations in Unity confirm that hand gestures such as pushing, reaching, or rotating result in accurate movement and interaction with virtual objects.

The system performed well even with moderate hand movement speeds and in varying lighting conditions. These outcomes affirm the robustness, efficiency, and practicality of the proposed system for real-world use in VR, AR, simulation training, and digital interaction applications.

CHAPTER 8

SOURCE CODE

8.1 MAIN.PY

```
from cvzone.HandTrackingModule import HandDetector
import cv2
import socket
cap = cv2.VideoCapture(0)
cap.set(3, 1280)
cap.set(4, 720)
success, img = cap.read()
h, w, _ = img.shape
detector = HandDetector(detectionCon=0.8, maxHands=2)
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
serverAddressPort = ("127.0.0.1", 5052)
while True:
    # Get image frame
    success, img = cap.read()
    # Find the hand and its landmarks
    hands, img = detector.findHands(img)  # with draw
    # hands = detector.findHands(img, draw=False)  # without draw
    data = []
    if hands:
        hand = hands[0]
        lmList = hand["lmList"]  # List of 21 Landmark points
        for lm in lmList:
            data.extend([lm[0], h - lm[1], lm[2]])
        sock.sendto(str.encode(str(data)), serverAddressPort)
    img=cv2.resize(img,(0,0),None,0.5,0.5)
    cv2.imshow("Image", img)
    cv2.waitKey(1)
```

8.2 HANDTRACKING.CS

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class HandTracking : MonoBehaviour
{
    public UDPReceive udpReceive;
    public GameObject[] handPoints;
    void Start()
    {
    }
void Update()
    {
        string data = udpReceive.data;
        data = data.Remove(0, 1);
        data = data.Remove(data.Length-1, 1);
        print(data);
        string[] points = data.Split(',');
        print(points[0]);
        //0        1*3      2*3
        //x1,y1,z1,x2,y2,z2,x3,y3,z3
        for ( int i = 0; i<21; i++)
        {

            float x = 7-float.Parse(points[i * 3])/100;
            float y = float.Parse(points[i * 3 + 1]) / 100;
            float z = float.Parse(points[i * 3 + 2]) / 100;
            handPoints[i].transform.localPosition = new Vector3(x, y, z);
        }
    }
}
```

8.3 LINE.CS

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class LineCode : MonoBehaviour
{
   LineRenderer lineRenderer;
   public Transform origin;
   public Transform destination;
   // Start is called before the first frame update
   void Start()
   {
      lineRenderer = GetComponent<LineRenderer>();
      lineRenderer.startWidth = 0.1f;
      lineRenderer.endWidth = 0.1f;
   }

   // Update is called once per frame
   void Update()
   {
      lineRenderer.SetPosition(0, origin.position);
      lineRenderer.SetPosition(1, destination.position);
   }
}
```

8.4 UDP RECEIVE.CS

```
using UnityEngine;
using System;
using System.Text;
using System.Net;
using System.Net.Sockets;
using System.Threading;

public class UDPReceive : MonoBehaviour
{

    Thread receiveThread;
    UdpClient client;
    public int port = 5052;
    public bool startRecieving = true;
    public bool printToConsole = false;
    public string data;

    public void Start()
    {

        receiveThread = new Thread(
            new ThreadStart(ReceiveData));
        receiveThread.IsBackground = true;
        receiveThread.Start();
    }

    private void ReceiveData()
    {

        client = new UdpClient(port);
        while (startRecieving)
        {

            try
            {
                IPEndPoint anyIP = new IPEndPoint(IPAddress.Any, 0);
                byte[] dataByte = client.Receive(ref anyIP);
                data = Encoding.UTF8.GetString(dataByte);

  if (printToConsole) { print(data); }
            }
            catch (Exception err)
            {
                print(err.ToString());
            }
        }
    }
```

REFERENCES

[1]. Chunduru, V., Roy, M. and Chittawadigi, R.G., 2021, September. Hand tracking in 3d space using mediapipe and pnp method for intuitive control of virtual globe. In *2021 IEEE 9th Region 10 Humanitarian Technology Conference (R10-HTC)* (pp. 1-6). IEEE.

[2]. Gangurde, R., Ahire, K., Tadage, S. and Lavangale, S.A., Air Canvas Using MediaPipe for Computer Vision in Unity 3D Hand Tracking. *International Journal of Advanced Research in Science, Communication and Technology*.

[3]. Che, Y. and Qi, Y., 2021, October. Detection-guided 3d hand tracking for mobile ar applications. In *2021 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)* (pp. 386-392). IEEE.

[4]. Naganandhini, K., Sowmya, B., Pavish, S., Nitesh, J., Karthika, R. and Prabhu, E., 2022, November. Hand Tracking Based Human-Computer Interaction Teaching System. In *2022 3rd International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT)* (pp. 1-5). IEEE.

[5]. Tsai, C.C., Kuo, C.C. and Chen, Y.L., 2020, August. 3D hand gesture recognition for drone control in unity. In *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)* (pp. 985-988). IEEE.

[6]. Reimer, D., Scherzer, D. and Kaufmann, H., 2023. Ownership Estimation for Tracked Hands in a Colocated VR Environment. In # PLACEHOLDER_PARENT_METADATA_VALUE# (pp. 105-114). Eurographics Association.

[7]. Voigt-Antons, J.N., Kojic, T., Ali, D. and Möller, S., 2020, May. Influence of hand tracking as a way of interaction in virtual reality on user experience. In *2020 twelfth international conference on quality of multimedia experience (QoMEX)* (pp. 1-4). IEEE.

[8]. Mueller, F., Mehta, D., Sotnychenko, O., Sridhar, S., Casas, D. and Theobalt, C., 2017. Real-time hand tracking under occlusion from an egocentric rgb-d sensor. In *Proceedings of the IEEE international conference on computer vision* (pp. 1154-1163).

[9]. Schröder, M., Elbrechter, C., Maycock, J., Haschke, R., Botsch, M. and Ritter, H., 2012, November. Real-time hand tracking with a color glove for the actuation of anthropomorphic robot hands. In *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012)* (pp. 262-269). IEEE.

[10]. de Castro, M.C., de A Xavier, J.P., Rosa, P.F. and de Oliveira, J.C., 2021, March. Interaction by Hand-Tracking in a Virtual Reality Environment. In *2021 22nd IEEE International Conference on Industrial Technology (ICIT)* (Vol. 1, pp. 895-900). IEEE.

[11]. Sampson, H., Kelly, D., Wünsche, B.C. and Amor, R., 2018, November. A hand gesture set for navigating and interacting with 3d virtual environments. In *2018 International Conference on Image and Vision Computing New Zealand (IVCNZ)* (pp. 1-6). IEEE.

[12]. Geetha, S., Aditya, G., Reddy, C. and Nischith, G., 2024, July. Human Interaction in Virtual and Mixed Reality Through Hand Tracking. In *2024 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)* (pp. 1-6). IEEE.

[13]. Cameron, C.R., DiValentin, L.W., Manaktala, R., McElhaney, A.C., Nostrand, C.H., Quinlan, O.J., Sharpe, L.N., Slagle, A.C., Wood, C.D., Zheng, Y.Y. and Gerling, G.J., 2011, April. Hand tracking and visualization in a virtual reality simulation. In *2011 IEEE systems and information engineering design symposium* (pp. 127-132). IEEE.

[14]. Shravya, M., Swathi, C., Vaishnavi, T.N., Aditya, T.G., Dr Pavithra, G. and Dr Manjunath, T.C., 2022. 3D Hand Tracking in Virtual Environments. Grenze International Journal of Engineering and Technology, GIJET, Jan Issue© Grenze Scientific Society, ISSN, pp.2395-5295.

[15]. Yang, D., Cao, X., Liu, Y. and Xu, S., 2022, October. Marker-enhanced hand tracking with deep learning. In *2022 International Conference on Virtual Reality, Human-Computer Interaction and Artificial Intelligence (VRHCIAI)* (pp. 37-42). IEEE.