I. 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.

A. 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.

B. Objective

The goal of this project is to create a system that allow user to interactively control the virtual environments using only their hands.

- Accurate hand landmark detection leveraging Mediapipe.
- Smooth data transmission between Python (for Mediapipe processing) and Unity (for 3D visualization).
- Realistic, low-latency movement rendering in Unity.
- Scalability for future enhancements, such as sign language interpretation and gesture-based commands.

C. Scope

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.

II. 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.

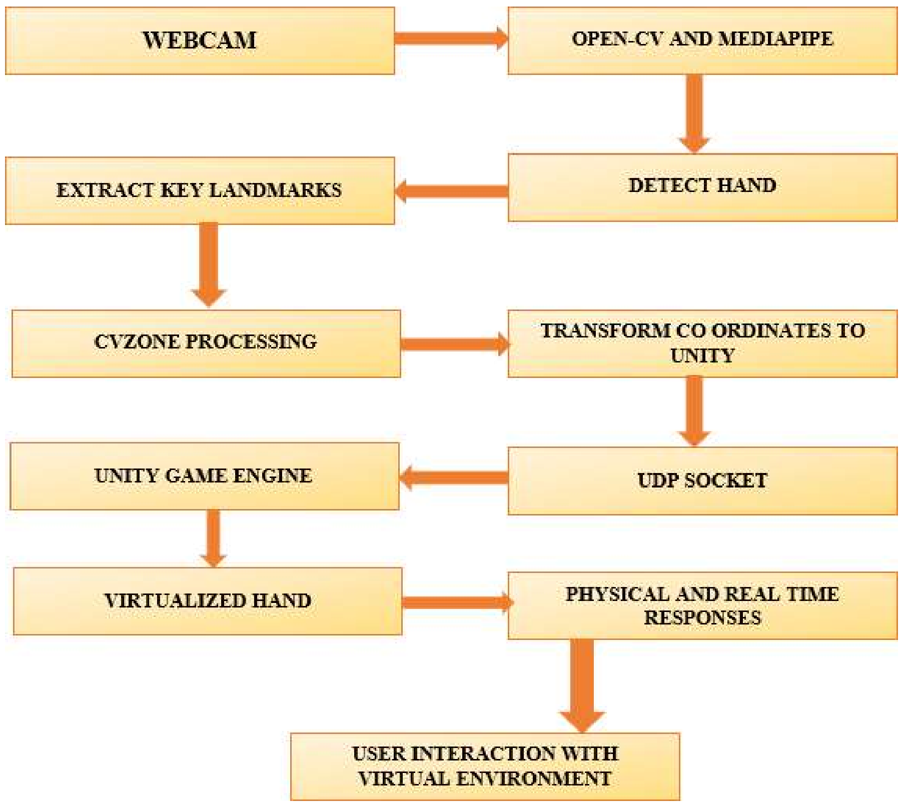

Fig. 1. Proposed Method

The Fig 1 illustrates the block diagram for 3D hand tracking scheme in a virtual environment using OpenCV. MediaPipe, CVZone, and the Unity game engine It begins with a webcam that records a live video. Then, it is translated using OpenCV and MediaPipe to recognise the user's hand. When a hand is identified, major Key details such as where the fingers are found and palm orientation are collected to track the hand movement efficiently. The extracted landmarks go through CVZone, a computer vision library that allows for hand tracking and interaction easier.

The landmarks are converted to Unity-compatible coordinates and sent over a UDP (User Datagram Protocol) socket. This exchange of information ensures that Unity gets real-time hand position information, enabling smooth integration between the tracking system and the virtual world.

After Unity receives the hand tracking data, it renders a virtualized hand that follows the actual hand's movement. Physical and real-time feedbacks, including object interactions and real time operations, are utilized to enrich the user experience. Ultimately, the user can naturally interact with the virtual world, so this system can be used for gaming, simulations, and immersive experiences where hand movements operate digital objects.

Below is a wide-ranging description of each module:

1. Video Capture

This module is responsible for capturing webcam video in real time. The camera keeps streaming frames that act as feed to system. The resolution and quality of the captured video influence the accuracy of hand detection and tracking. When a clearer feed is provided, it becomes easier for the system to see the hand better.

2. Pre-processing

Before the video frames goes to the hand tracking system, this module processes it. It uses image processing
methods like resizing and cropping to focus on the hand area, adjusts contrast and brightness for different lighting conditions, and uses noise reduction and background filtering to lessen the noise around the hand. These steps isolate the hand better and avoid sending extra data.

3. Hand Tracking

The user's hand is detected and tracked in real-time by this core module. The system makes use of MediaPipe's Hand Tracking model and performs two main tasks: Palm Detection and Landmark Detection.The first task Palm Detection finds the hand in the video feed, and subsequently Landmark Detection finds 21 (x,y,z) coordinates from the identified hand in the video. This means a palm will be having 21 coordinates. These landmarks will allow you to have realistic and accurate hand motion.

4. Data Transmission

After landmark detection, we need to send 3D coordinates to Unity for visualization. This modulator converts data from the detected landmarks into a modulation of networking format. It makes use of UDP (User Datagram Protocol) to send Python script data to Unity in real time. The use of UDP assures low latency transmission which is very important for fluid tracking performance during live sessions.

5. 3D Hand Mapping

In Unity, the incoming hand tracking data is responsible for animating a 3D hand. This system takes the UDP data that we got and parses it so that it can then map the (x, y, z) coordinates of the hand into the model. The objective of this module is to make the virtual hand behave like a real one with smooth hand motion and low delay.

6. Interaction

With this module your virtual hand and objects in the virtual environment may interact. Users can basically move and push simple objects using this module within the given scene. It also allows VR compatibility and the ability for users to navigate and interact in VR spaces without the use of controllers. This helps the system in the performance of more immersive tasks.

III. EXPERIMENTAL SETUP

The implementation of 3D hand tracking in a virtual environment includes a combination of computer vision and devices use 3D visual to send instant data. The project begins with Media Pipe hand Tracking the system detects and tracks hand features from a webcam. Using OpenCV, the captured frames process the images and use CVZone to help with hand landmark detection and coordinate operations. The extracted hand landmark data is dispatched to Unity through UDP communication for efficient and real-time integration transmission. This makes it possible for Unity to receive hand movement coordinates and deploy virtual objects in a 3D environment.

In unity custom script is used for mapping the values to virtual objects. The hand tracking manager takes in coordinates and then applies them on 3D models. In contrast to his system translates the user's hand movement immediately through without gesture recognition to make it more natural and inherent. This can be applied in several applications including virtual reality (VR), interactive gaming, and an environment Interactions, simulations, and digital art make this a versatile tool for immersive human-computer interaction.

A. Algorithm

These algorithms describe in detail how to track hand movement and interact with virtual objects in real time without gesture recognition.

```
BEGIN
    - Import necessary libraries: mediapipe, opencv, cvzone, socket
    - Start webcam
    - Set up hand tracking with mediapipe
    - Create a UDP socket to send data
```

```
WHILE program is running:
    - Capture a frame from webcam
    - Convert it to RGB format
    - Detect hand using mediapipe
    - IF a hand is found:
        - Get 21 hand landmark points
        - Format the points into a string
        - Send the data to Unity via UDP
    In Unity:
        - Receive the landmark data
        - Convert the coordinates to 3D
        - Move the virtual hand to match
        - IF hand touches a virtual object:
            - Apply movement or physics to the object
```

Algo 1. Algorithm for Hand Tracking

The algorithm-1 describes how the proposed system tracks the actual movements of a hand using a webcam and sends that information to Unity to animate a virtual hand. It uses MediaPipe to find 21 hand locations from frames and prepares the positions to send over UDP. In Unity, we received this information, changed it to 3D coordinates, and used it to control the virtual hand. If the virtual hand comes into contact with an object, it can trigger a movement or a physics-based interaction. While loop continues until the user press ‘q’ and the webcam and all the windows are released properly.

```
 BEGIN
   // Part 1: Receiving Data (in background)
   - Start a thread to listen for data on a UDP port.
   - WHILE running:
       - Wait for new data.
       - Convert it to text.
       - Save it for use in Unity.
       - (Optional) Print it in the console.
   // Part 2: Updating Hand Points (every frame)
   - Get the latest received data.
   - Clean the text.
   - Split it into x, y, z values.
   FOR each of the 21 hand points:
       - Convert x, y, z to numbers.
       - Adjust the values to fit Unity's 3D world.
       - Move the hand point to the new position.
       - Continue in loop till hand not detected
END
```

Algo 2. Algorithm for UDP Connection

The algorithm-2 shows that the system, upon receiving the necessary data from UDP, processes everything in Unity. A background thread is calm and listens to incoming UDP data which contains 3D coordinates of 21 hand landmarks. Each frame, Unity 3D reads the latest data, cleans and splits it, transforms coordinates to fit its 3D space, and updates virtual hand point positions.

B. Flow Diagram

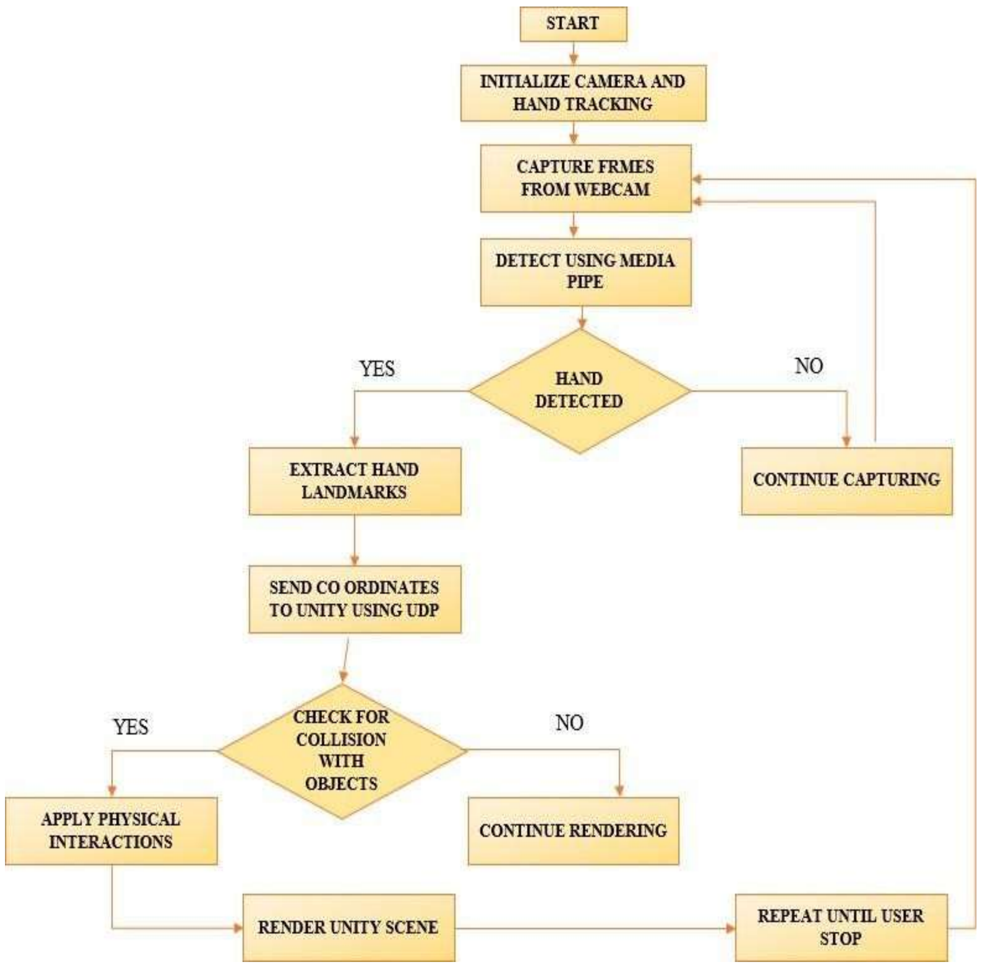
START
INITIALIZE CAMERA AND HAND TRACKING
CAPTURE FRMES FROM WEBCAM
DETECT USING MEDIA PIPE
YES
HAND DETECTED
NO
EXTRACT HAND LANDMARKS
CONTINUE CAPTURING
SEND CO ORDINATES TO UNITY USING UDP
YES
CHECK FOR COLLISION WITH OBJECTS
NO
APPLY PHYSICAL INTERACTIONS
CONTINUE RENDERING
RENDER UNITY SCENE
REPEAT UNTIL USER STOP

Fig. 2. Flow Diagram

The Fig 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 begins by setting up the camera and hand Tracking until receiving webcam frames continuously. These frames are processed using MediaPipe to detect hands. If the system detects the hand, the system gets hand landmarks and sends the coordinates to Unity using the UDP. If a hand is not detected, the system will keep capturing frames until a hand is found. After getting the hand landmarks, these are used to detect collisions of virtual objects environment.

Whenever a collision is detected, movements like pushing of the object or its translation are used to simulate real-world conditions. Moreover, the rendering procedure continues with no additional interactions.

In Unity the scene is updated based on the hand's new coordinates, and a similar process runs in real time to render a smooth. interactive experience. The application continues to change the virtual scene and observe hand movements until the user chooses to stop the interaction. This system uses MediaPipe to track hands and Unity to render in real-time which permits users to engage with virtual items in a real-life manner. This approach can be used in many best applications in games, virtual training, and interactive simulations digital content without using hand gestures.

C. PyCharm

PyCharm was used to create and run in this project the 3D hand tracking system by means of writing of Python scripts that process video input, detect hand landmarks as with the mediapipe hand model, and send the data to Unity.

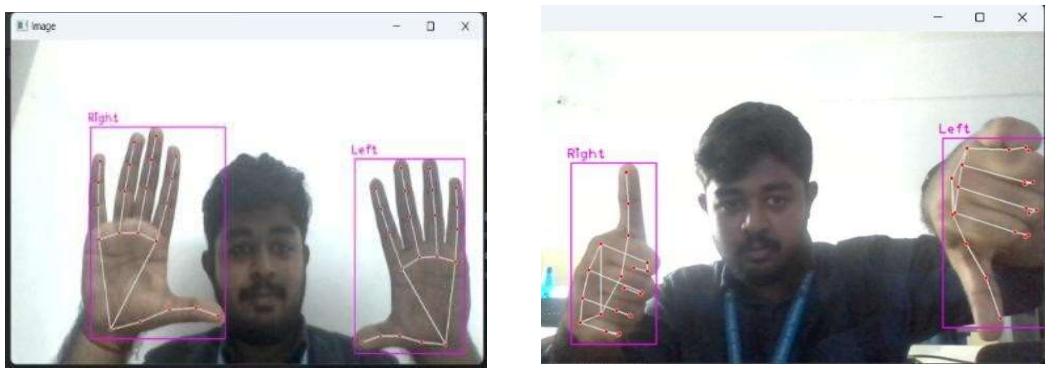

Fig. 3. Hand Detected in Different Orientations-1

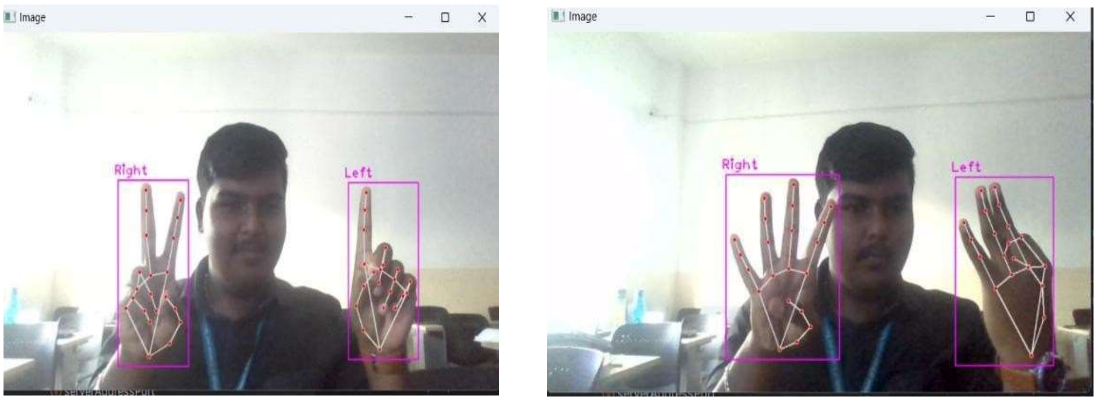

Fig. 4. Hand Detected in Different Orientations-3

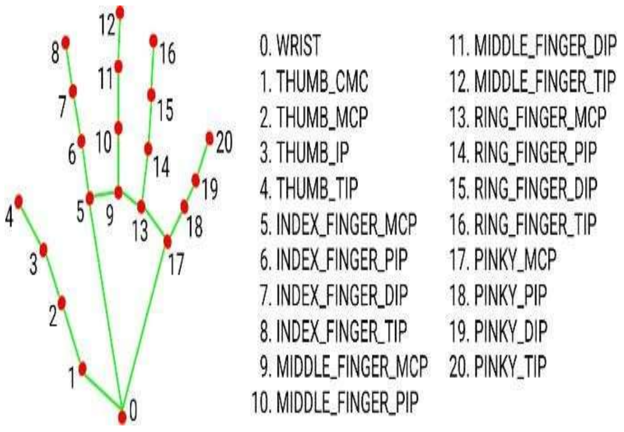

Fig.5. Hand Model

D. Unity Engine

In this phase, Unity becomes the playground where our hand Continue reading 3D hand tracking, we take the real-time position of hands in Python using the hand tracking model of MediaPipe. After that, this information is sent to Unity through UDP communication so that our virtual hand can perform the same movement. The Hand Tracking Script maps each finger and joint correctly, allowing us to interact naturally with virtual objects. The 3D hand had twenty-one points as spheres and the connection between those points as lines. The 3d hand was created by 21 points as spheres and connection between the points as lines.

IV. Result and discussion

A. Result

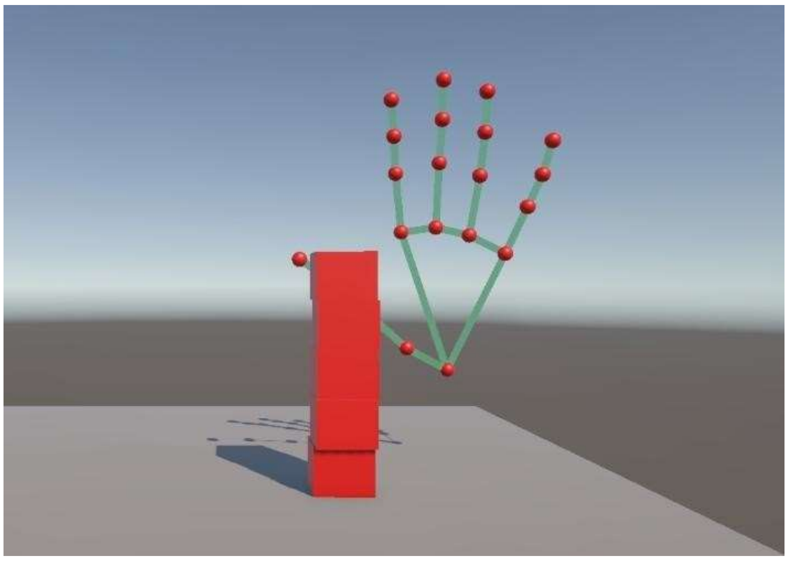

Fig. 7. Simulated Result-1

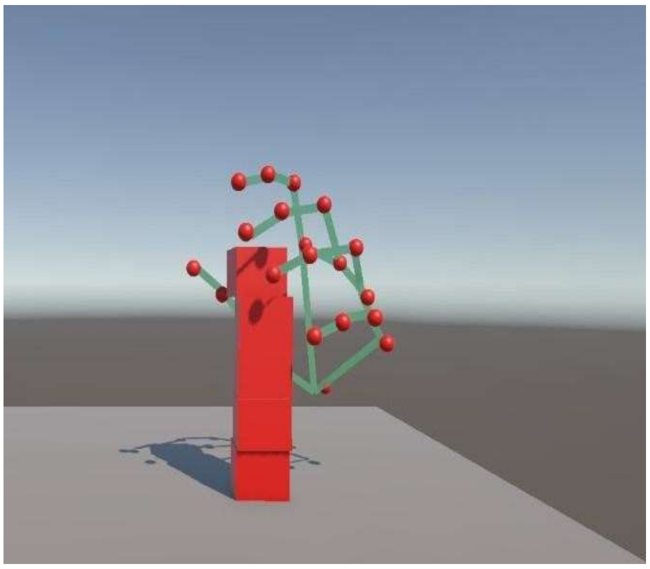

Fig. 8. Simulated Result-2

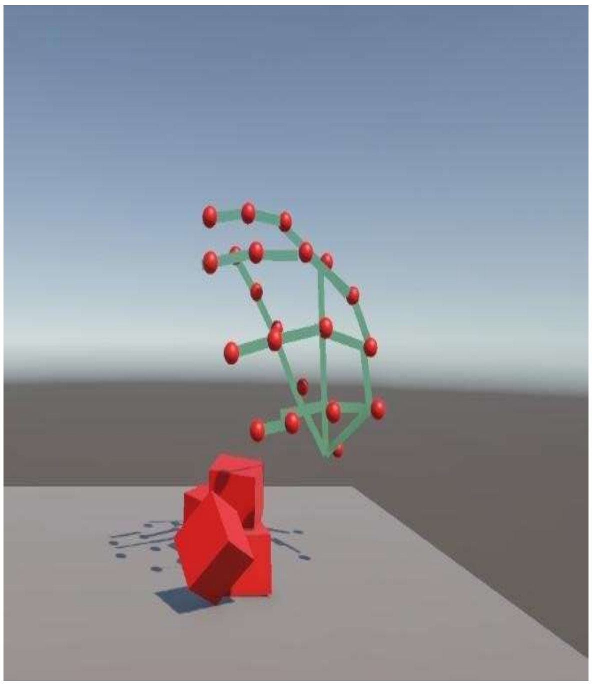

Fig. 9. Simulated Result-3

These pictures show hand tracking in the 3D world with Unity and MediaPipe. The workspace in Unity is filled with red cubes and an instance of tracked hand model comprising 21 landmark points, each rendered as red dots connected with green lines. Real-time shading and axis labels reveal object and scene manipulation through live rendering. A closer examination to the image depicts the hand interacting with cubes, suggesting gesture-less control and the system operates by real time tracking using MediaPipe Hand Tracking, sending landmark information (x, y, z coordinates) to Unity, and displaying the hand model for real-time use.

B. Analysis

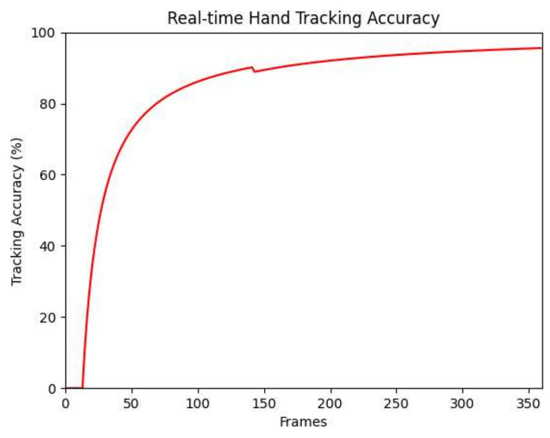

Fig. 10. Validation Accuracy over Frames

This Fig.6 plot shows real-time hand tracking accuracy across a sequence of frames. The X-axis is the number of processed frames, and the Y-axis is the percentage of frames where hands were accurately detected. The accuracy is low at first due to potential detection instability but then increases quickly as the system settles down and stabilizes. Near the 100th frame, the accuracy settles around 80–90%, showing stable tracking performance.

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

Table 1. Hand Tracking Evaluation Metrics

V. CONCLUSION

In summary, the Real-Time Human-Machine Interaction project effectively applies real-time hand tracking through MediaPipe, OpenCV, and Unity to provide smooth interaction with virtual objects. As opposed to conventional gesture recognition systems, this project is aimed at mirroring real-time hand movement to control objects within a virtual environment. With MediaPipe's 21 hand landmarks, the system effectively tracks the position and angle of hands to offer accurate interaction. Integration with Unity enables the virtual experience where users are able to interact with virtual objects, making it suitable for applications like virtual reality (VR), augmented reality (AR), gaming, and remote collaboration. The implementation illustrates the effectiveness and precision of the MediaPipe hand tracking model. The processed landmark information is delivered via UDP communication from Python (MediaPipe) to Unity with real-time responsiveness and low latency. The high accuracy in hand tracking is effectively preserved in the project, as proven through performance tests, and ensures smooth reflection of motion in Unity. PyCharm usage for Python scripting and Unity for 3D visualization illustrates effective cross- platform integration, which enhances the interaction with users. In summary, the project shows the promise of hand tracking through AI in developing embedded and interactive virtual experiences. The capability of the system to record and respond to real-time hand movements with processing paves the way for potential applications beyond gaming, including education, healthcare, and human- computer interaction. Future developments may address multi-hand tracking, more realistic physics-based interaction, and decreased latency for the better experience. This research forms the basis for further development of real-time hand tracking in virtual worlds and, as such, is a useful addition to the computer vision and AI-based interaction systems community.

VI. CODING

MAIN PYTHON CODE

```
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 1
      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)
   # Display
   img=cv2.resize(img,(0,0),None,0.5,0.5)
   cv2.imshow("Image", img)
   cv2.waitKey(1)
```

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.