PALM PLAY

A Mini Project Report submitted in partial fulfillment of the

requirements for the award of the degree of

Bachelor of Technology in

Computer Science and Engineering

By

C. DATTASAI ADITYA AMMAN (21011P0511)

MOHAMMED ZAFEER TALHA (21011P0521)

SANKALA SAI RUTHVIZ (21011P0527)

Under the guidance of

DR. I. LAKSHMI MANIKYAMBA

Associate Professor

Department of CSE

Department of Computer Science and Engineering

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY HYDERABAD

UNIVERSITY COLLEGE OF ENGINEERING, SCIENCE & TECHNOLOGY HYDERABAD

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY HYDERABAD

UNIVERSITY COLLEGE OF ENGINEERING, SCIENCE & TECHNOLOGY HYDERABAD

Kukatpally, Hyderabad 500085

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

DECLARATION BY THE CANDIDATES

We hereby declare that the Project work entitled **"PALM PLAY"** submitted in partial fulfillment of the requirements for the award of the degree BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE AND ENGINEERING which was carried out under the supervision of **Dr I. Lakshmi Manikyamba,** Associate Professor.

Also, we declare that the matter embedded in the thesis has not been submitted by us in full or partial thereof to any other University or Institute for the award of any degree previously.

Chintala Dattasai Aditya Amman (21011P0511)

Mohammed Zafeer Talha (21011P0521)

Sankala Sai Ruthviz (21011P0527)

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY HYDERABAD

UNIVERSITY COLLEGE OF ENGINEERING, SCIENCE & TECHNOLOGY HYDERABAD

Kukatpally, Hyderabad 500085.

Department of Computer Science and Engineering

CERTIFICATE BY THE SUPERVISOR

This is to certify that the Project report on "**PALM PLAY**" is a bonafide work carried out by **Chintala Dattasai Aditya Amman (21011P0511), Mohammed Zafeer Talha (21011P0521)**, **Sankala Sai Ruthviz (21011P0527)** in the partial fulfillment for the award of B.Tech degree in Computer Science and Engineering, Jawaharlal Nehru Technological University, Hyderabad under our guidance and supervision.

The results embodied in the project work have not been submitted to any other University or Institute for the award of any degree or diploma.

Dr. I. Lakshmi Manikyamba

Associative Professor

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY HYDERABAD

UNIVERSITY COLLEGE OF ENGINEERING, SCIENCE & TECHNOLOGY HYDERABAD

Kukatpally, Hyderabad 500085.

Department of Computer Science and Engineering

CERTIFICATE BY THE HEAD OF THE DEPARTMENT

This is to certify that the Project report on "**PALM PLAY**" is a bonafide work carried out by **Chintala Dattasai Aditya Amman (21011P0511), Mohammed Zafeer Talha (21011P0521)**, **Sankala Sai Ruthviz (21011P0527)** in the partial fulfillment for the award of B.Tech degree in Computer Science and Engineering, Jawaharlal Nehru Technological University, Hyderabad under our guidance and supervision.The results embodied in the project work have not been submitted to any other University orInstitute for the award of any degree or diploma.

Head of the Department

Dr. K.P. Supreethi

Professor and Head

Department of CSE

ACKNOWLEDGEMENTS

The satisfaction that accompanies the successful completion of the task would be put incomplete without the mention of the people who made it possible, whose constant guidance and encouragement crown all the efforts with success.

We wish to express my deep sense of gratitude to **Dr. I Lakshmi Manikyamba**, Associate Professor and Project guide, for her able guidance and useful suggestions which helped in completing the project work intime.

We are extremely grateful to our HOD- **Dr. K. P. Supreethi**, Professor & Head of CSE for her immense support and cooperation that contributed a lot in the completion of the task.

We show gratitude to our beloved Principal **Dr. G. Venkata Narasimha Reddy**, Senior Professor, Principal & Director of R&D for providing necessary infrastructure and resources for the accomplishment of my project report at JNTUH University College of Engineering, Hyderabad.

We also thank all the staff members of Computer Science & Engineering department, JNTUH University College of Engineering, Hyderabad for their valuable support and generous advice.

Finally, thanks to our parents, all our family members and friends for their support and enthusiastichelp.

Chintala Dattasai Aditya Amman (21011P0511)

Mohammed Zafeer Talha (21011P0521)

Sankala Sai Ruthviz (21011P0527)

ABSTRACT

Gesture prediction with landmark detection is an emerging research area at the intersection of computer vision, machine learning, and human-computer interaction. The primary focus is on predicting hand signs by accurately detecting and analyzing key landmarks on the hand. Landmark detection involves identifying specific points on the hand, such as fingertips, knuckles, and the wrist, which are critical for interpreting hand gestures accurately. This research aims to push the boundaries of current gesture recognition systems by improving accuracy, real-time performance, and robustness, thereby paving the way for more sophisticated and user-friendly human-computer interfaces.

The main goal is to improve real-time prediction and recognition of hand signs and enhance prediction accuracy through precise landmark detection. At its core, the project leverages advanced AR technology to enable real-time hand tracking, allowing individuals to communicate, interact and influence the digital spectrum. This natural interaction bridges the gap between physical and virtual expression, offering a seamless and immersive experience. The collaborative capabilities of the project are particularly transformative.

By enabling multiple users to interact within the same virtual environment, we are able to achieve understanding through media and opening the world of technology to all despite their differences in form or ability.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

1.1 Introduction

In the new era of the digital evolution, computer vision and recognition software has transformed into a key field and has proceeded to revolutionize the concept of human-computer interaction that seeks to bridge the gap between human gestures and digital interfaces. By accurately identifying and interpreting hand signs, facial structures, body language and human behavioral patterns, computers can be trained to understand and respond to these gestures, enabling more intuitive and natural interactions between both parties.

Hence, our project focuses on developing an advanced system for hand sign recognition by leveraging state-of-the-art technologies in landmark detection and gesture prediction. Landmark detection involves pinpointing specific points on the hand, such as fingertips, knuckles, and the wrist, which are crucial for accurately interpreting hand gestures. By refining these techniques, we aim to improve the accuracy, real-time performance, and robustness of gesture recognition systems.

The ultimate goal of our project is to create a sophisticated and user-friendly interface that enhances how people interact with digital systems. Through precise landmark detection and gesture prediction, we aim to transform user experiences across various applications, making interactions with technology more seamless and intuitive.

1.2 Problem Statement

In today's rapidly advancing technological landscape, there is a growing need for more intuitive and natural human-computer interactions. Traditional input methods such as keyboards, mice, and touchscreens, while effective, can be limiting and cumbersome, especially in dynamic and immersive environments like augmented reality (AR), virtual reality (VR), and smart home automation.

Current gesture recognition systems often struggle with issues of accuracy, real-time performance, and robustness, particularly in recognizing complex hand signs. These limitations hinder the development of more sophisticated and user-friendly interfaces that can seamlessly interpret and respond to human gestures.

Our project focuses on the developing this communication channel that goes beyond the scope of simple spoken words or letters that are input on a screen. This allows for an entirely dynamic environment where the weight of human interaction exists only through ourselves rather than any specialized external media, common or otherwise.

LITERATURE SURVEY

2.1 Motivation

The motivation for our hand sign recognition project stems from the desire to create more natural, intuitive, and efficient ways for humans to interact with digital systems. In a world increasingly dominated by technology, the limitations of traditional input methods such as keyboards, mice, and touchscreens have become more apparent.

This system sees many uses while considering outreach and inclusivity of user-base. We have used Open CV technology to build this project.

2.2 Objectives

Through this model, we plan to focus on the development of the following features:

- Implement advanced landmark detection algorithms to precisely identify key points on the hand such as fingertips, knuckles, and the wrist.
- Achieve high accuracy in recognizing a wide range of hand signs and gestures.
- Minimize latency and improve the responsiveness of the hand sign recognition system.
- Create gesture-based interfaces that provide alternative communication methods for individuals with disabilities.
- Ensure consistent performance across different hand shapes, sizes, and movements.

- Develop gesture-based controls for smart home devices, allowing users to manage lights, thermostats, entertainment systems, and more using hand signs.

2.3 History

The development of hand sign recognition systems has its roots in the broader fields of computer vision and human-computer interaction, which have been evolving since the mid-20th century. Early efforts in gesture recognition began with simple computer algorithms designed to detect basic hand movements using 2D images. However, these early systems were limited by the technology of the time, which lacked the processing power and sophisticated algorithms needed for accurate and real-time interpretation.

The advent of machine learning in the late 20th and early 21st centuries marked a significant turning point. Researchers began applying neural networks and other advanced algorithms to improve the accuracy and robustness of gesture recognition systems. Landmark detection, which involves identifying key points on the hand such as fingertips, knuckles, and the wrist, emerged as a critical component of these systems. However, challenges remained in achieving real-time performance and handling the variability of hand shapes, sizes, and movements.

With the rise of deep learning and the availability of large datasets, significant strides were made in the 2010s. Projects like MediaPipe by Google introduced frameworks that made it easier to build complex machine learning pipelines for hand tracking and gesture recognition. These advances allowed for more precise and reliable landmark detection, enabling more accurate interpretation of hand signs.

Our project builds on these advancements, leveraging state-of-the-art technologies in computer vision and machine learning. Recognizing the

limitations of existing systems, we set out to develop a more sophisticated and user-friendly hand sign recognition system.

2.4 Applications

Hand sign prediction has vast potential applications in various fields, including:

- **Augmented Reality (AR) and Virtual Reality (VR):** Enhancing user experiences by enabling intuitive gesture-based interactions.
- **Assistive Technologies:** Providing accessible communication methods for individuals with disabilities.
- **Interactive Gaming:** Creating more immersive and interactive gaming experiences through natural hand movements.
- **Smart Home Automation**: Controls smart home devices such as lights, thermostats, and entertainment systems using simple hand gestures.
- **Security and Surveillance**: Provides gesture-based authentication and control systems for enhanced security in sensitive environment.
- **Education and Training**: Facilitates interactive learning and training by allowing users to manipulate digital content and perform tasks using gestures.

SYSTEM ANALYSIS

3.1 Existing System

In modern times, we have come across cutting edge technology representative of the advancements in Artificial reality and the AR scene. Augmented Reality (AR) and Deep Learning technologies can significantly address the limitations faced without gesture detection in modern interactions. AR enhances user experiences by overlaying digital content onto the real world, creating immersive environments where users can interact naturally with virtual objects. Integrating gesture detection into AR systems enables more intuitive and seamless interactions, allowing users to manipulate digital elements using hand gestures. This makes AR applications in fields like education, gaming, and healthcare more engaging and effective.

The combination of AR and deep learning can transform smart home automation, making it more intuitive and user-friendly. Users can control devices with simple hand gestures, enhancing convenience and accessibility. In interactive gaming, this integration allows for more natural and immersive gameplay, where players can use gestures to interact with virtual environments and characters. Additionally, deep learning-driven gesture detection can enhance security and authentication systems, providing a robust layer of verification based on unique hand movements.

However, it is important to note that system is not without its disadvantages. The most common issue faced by people accessing and utilizing these technologies are:

1. Limited Interaction Modalities
    - Traditional input methods like keyboards, mice, and touchscreens restrict the ways users can interact with digital devices, reducing the potential for more natural and intuitive interactions.

2. Accessibility Issues
    - Individuals with physical disabilities or mobility impairments may find it difficult or impossible to use conventional input devices, limiting their ability to interact with technology effectively.

3. Reduced Immersiveness in AR/VR
    - Augmented reality (AR) and virtual reality (VR) experiences rely heavily on natural interaction methods. Without gesture detection, these environments are less immersive and engaging, as users cannot interact intuitively with virtual objects.
4. Inconvenience in Smart Home Automation
    - Controlling smart home devices through traditional methods like mobile apps or voice commands can be less convenient and intuitive compared to using hand gestures, which can offer quicker and more direct control.

5. Inefficiency in User Interfaces
    - Gesture detection can streamline and enhance user interfaces, making them more efficient. Without it, users may need to navigate through multiple steps or menus, increasing interaction time and reducing overall efficiency.

6. Security and Authentication Limitations
    - Gesture-based authentication provides an additional layer of security. Without it, systems rely on traditional methods like passwords or PINs, which can be less secure and more vulnerable to breaches.

7. Inadequate Support for Multimodal Interaction

- Modern interfaces benefit from multimodal interaction, combining gestures, voice, and touch. Without gesture detection, the ability to create rich, multimodal experiences is significantly reduced.

8. Challenges in Remote Collaboration
    - Gesture detection can enhance remote collaboration by allowing participants to interact more naturally and expressively. Without it, remote interactions can feel less personal and dynamic, affecting communication and teamwork.

Through these reasons, it is quite common to see the scope for traditional speech-based system being quite limited as opposed to catering for an audience that lacks ability to access these devices.

3.2 Proposed System

An innovative proposed system for Art genius could leverage the powerful capabilities of OpenCV and MediaPipe to create a dynamic and intuitive digital canvas experience. OpenCV, a widely-used computer vision library, offers robust tools for image processing, object detection, and tracking, which are essential for accurately interpreting gestures and movements in real-time. By harnessing OpenCV's functionalities, the system could efficiently detect hand movements, gestures, and the trajectory of a virtual pen or brush.

Integrating MediaPipe into the system further enhances its capabilities by providing pretrained machine learning models specifically designed for hand tracking and pose estimation. MediaPipe's Hand Tracking and Background Subtraction solutions offer high accuracy and real-time performance, enabling precise tracking of hand movements and gestures with minimal latency. By utilizing these models, the system can accurately capture the user's hand movements and translate them into digital strokes on the canvas with smoothness and precision.

Together, OpenCV and MediaPipe form a powerful framework for building a Gesture Prediction system that offers users a seamless and intuitive drawing experience in the air. With real-time hand tracking, gesture recognition, and precise pen control, users can unleash their creativity and express themselves freely in a virtual environment. Whether for artistic expression, educational purposes, or interactive presentations, this proposed system holds immense potential for transforming the way people interact with digital content and explore their creativity.

3.3 Methodology

- The system recognizes the position of a writing hand and distinguishes it from a non-writing hand by counting the number of raised fingers. Proposed System Hand Region Segmentation Once we have accurately captured the hand using the above technique.

- The proposed algorithm works well in real-time and provides relatively accurate segmentation results. Although skin colors vary greatly from breed to breed, it has been observed that skin color has a small area between different skin types, while skin luminosity differs significantly.

- To get accurate hand detection we have used media pipe library followed by filtering of skin color at the boundary of the candidate's hand provides a reasonably good segmentation result, the subtraction step of background is only used to remove skin-colored objects (not part of the hand) that are in the bounding box of the recognized hand may be present. Hand Centroid Localization Since determining the exact center of gravity of the hand is critical in the following steps, the system uses two algorithms to determine the initial estimates of the center of gravity, and the final center of gravity is calculated as the average of the two.

- The algorithm code converts the incoming image into HSV space, which is a very suitable color space and perfect for color tracking. The tracking bars have organized the HSV values into the required color range of the colored object you placed on your finger. Contour Detection of the Mask of Color Object After detecting the Mask in Art Genius, it is now time to locate its center position for drawing the Line. The system will perform some morphological operations on the Mask to make it free of impurities and detect contour easily.

- The actual logic behind this computer vision project is to create a Python deque (a data structure). The deque will memorize the position of the outline in each subsequent frame, and we will use these accumulated points to create a line using OpenCV's drawing capabilities. Now use the outline position to decide whether to click a button or draw on the provided sheet. Some of the buttons are at the top of the canvas. When the pointer enters this area, it is activated according to the method present in this area.

- The system proposed in this method consists of following steps. It tracks the motion of colored fingertip, find its coordinates and plot the coordinates. After plotting coordinates optical character reorganization (OCR) is applied on plotted image.

SYSTEM REQUIREMENTS

4.1 Functional Requirements

1. Real-Time Hand Sign Detection
    - The system must accurately detect and track hand signs in real-time using a camera.
2. Landmark Detection
    - Identify and track specific points on the hand (e.g., fingertips, knuckles, wrist) to interpret gestures accurately.
3. Gesture Interpretation
    - Accurately recognize and classify a predefined set of hand signs and gestures.
4. User Feedback
    - Provide real-time visual or auditory feedback to the user upon successful gesture recognition.
5. Gesture Customization
    - Allow users to customize and add new hand signs or gestures for recognition.
6. Multimodal Integration
    - Integrate with other input modalities (e.g., voice commands, touch input) for a seamless user experience.
7. Data Logging and Analytics
    - Log user interactions and provide analytics for performance monitoring and system improvement.

4.2 Non-Functional Requirements

1. Performance
    - Ensure the system operates with low latency, providing real-time feedback within milliseconds.
2. Accuracy
    - Achieve a high recognition accuracy rate, ideally above 95%, to ensure reliable gesture interpretation.
3. Scalability
    - Design the system to handle multiple users and high-frequency interactions without performance degradation.
4. Robustness
    - Ensure the system performs reliably in various lighting conditions and with different hand shapes, sizes, and skin tones.
5. Usability
    - Create an intuitive and user-friendly interface that requires minimal training for users to start using the system effectively.
6. Compatibility
    - Ensure compatibility with various hardware platforms (e.g., smartphones, tablets, VR/AR headsets) and operating systems.
7. Security
    - Implement robust security measures to protect user data and prevent unauthorized access.
8. Privacy
    - Ensure user privacy by securely handling and storing gesture data, complying with relevant data protection regulations.

4.3 Hardware Requirements

1. Camera
    - A high-resolution camera capable of capturing detailed images of hands for accurate landmark detection.
    - Ideally supports at least 720p resolution with a high frame rate (30 FPS or higher).
2. Processing Unit
    - A powerful CPU or GPU to handle real-time image processing and machine learning computations.
    - Minimum: Quad-core CPU, 2.5 GHz or higher; Recommended: Dedicated GPU (e.g., NVIDIA RTX series).
3. Memory
    - Sufficient RAM to support real-time processing and data handling.
    - Minimum: 8 GB RAM; Recommended: 16 GB RAM or higher.
4. Storage
    - Adequate storage for software installation, datasets, and logging data.
    - Minimum: 256 GB SSD; Recommended: 512 GB SSD or higher.
5. Display
    - A display device for visual feedback and interaction.
    - Supports HD resolution or higher.
6. Peripheral Devices
    - Optional: Additional input devices such as touchscreens, VR/AR headsets, or microphones for multimodal interaction.

4.4 Software Requirements

1. Operating System
    - Compatible with major operating systems such as Windows, macOS, or Linux.

2. Programming Languages
    - Python: Primary language for implementing algorithms and developing the application.

3. Development Frameworks and Libraries
    - **Mediapipe**: Mediapipe is an advanced framework designed for constructing multimodal machine learning pipelines. It excels in applications requiring real-time hand tracking and gesture recognition. This framework provides pre-built models and tools that facilitate the detection and analysis of hand movements, making it a crucial component for interpreting gestures in the hand-cricket game. By leveraging Mediapipe, the project benefits from robust hand tracking capabilities, which are essential for accurate gesture recognition and interaction.

    - **OpenCV**: OpenCV (Open-Source Computer Vision Library) is a widely-used library dedicated to real-time computer vision tasks. It supports various functionalities such as image processing, video capture, and landmark detection. In the context of this project, OpenCV is utilized to handle video frames captured by the webcam, perform image processing operations, and assist in visualizing hand gestures. Its capabilities are integral for processing the visual data required for gesture recognition.

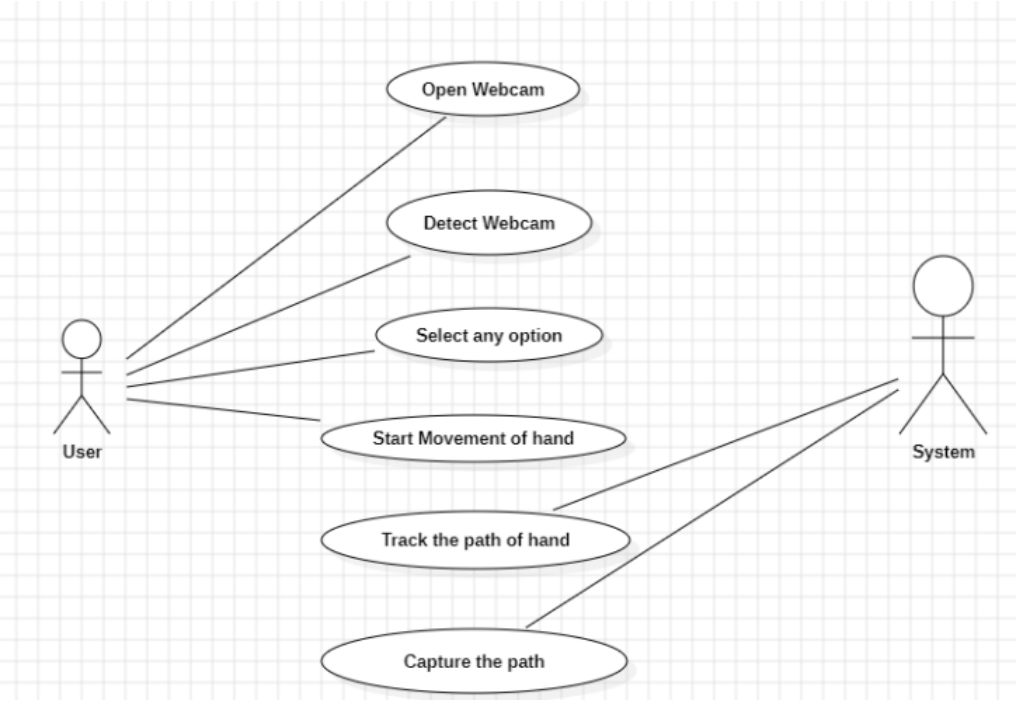
- **TensorFlow**: TensorFlow is a comprehensive open-source framework for deep learning and machine learning. It is employed for training and deploying machine learning models, providing a robust environment for developing and refining the gesture recognition model. TensorFlow's flexibility and scalability allow for the implementation of complex neural networks and the handling of large datasets, which are essential for training a model that can accurately predict hand gestures based on the input data.

- **Scikit-Learn**: Scikit-Learn is a powerful library for implementing traditional machine learning algorithms and evaluation metrics. It offers a range of tools for model training, evaluation, and hyperparameter tuning. In this project, Scikit-Learn is used for developing and assessing the machine learning models involved in gesture recognition. Its functionalities enable the implementation of various algorithms and provide metrics for evaluating model performance, ensuring that the gesture recognition system operates effectively and accurately.

4. Integrated Development Environment (IDE)
    - Tools for code development, debugging, and testing.
    - Examples: PyCharm, Visual Studio Code, Jupyter Notebook.

SYSTEM DESIGN

5.1 Use Case Diagram

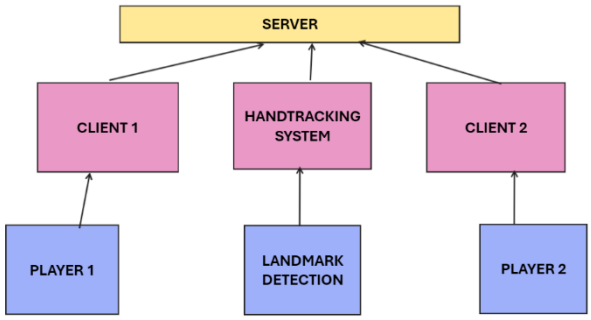


5.1.1

A use case diagram in the Unified Modeling language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use case), and any dependencies between those cases. The main purpose of a use case diagram is to show system functions are performed for which actor. Roles of the actors in the system can be depicted.

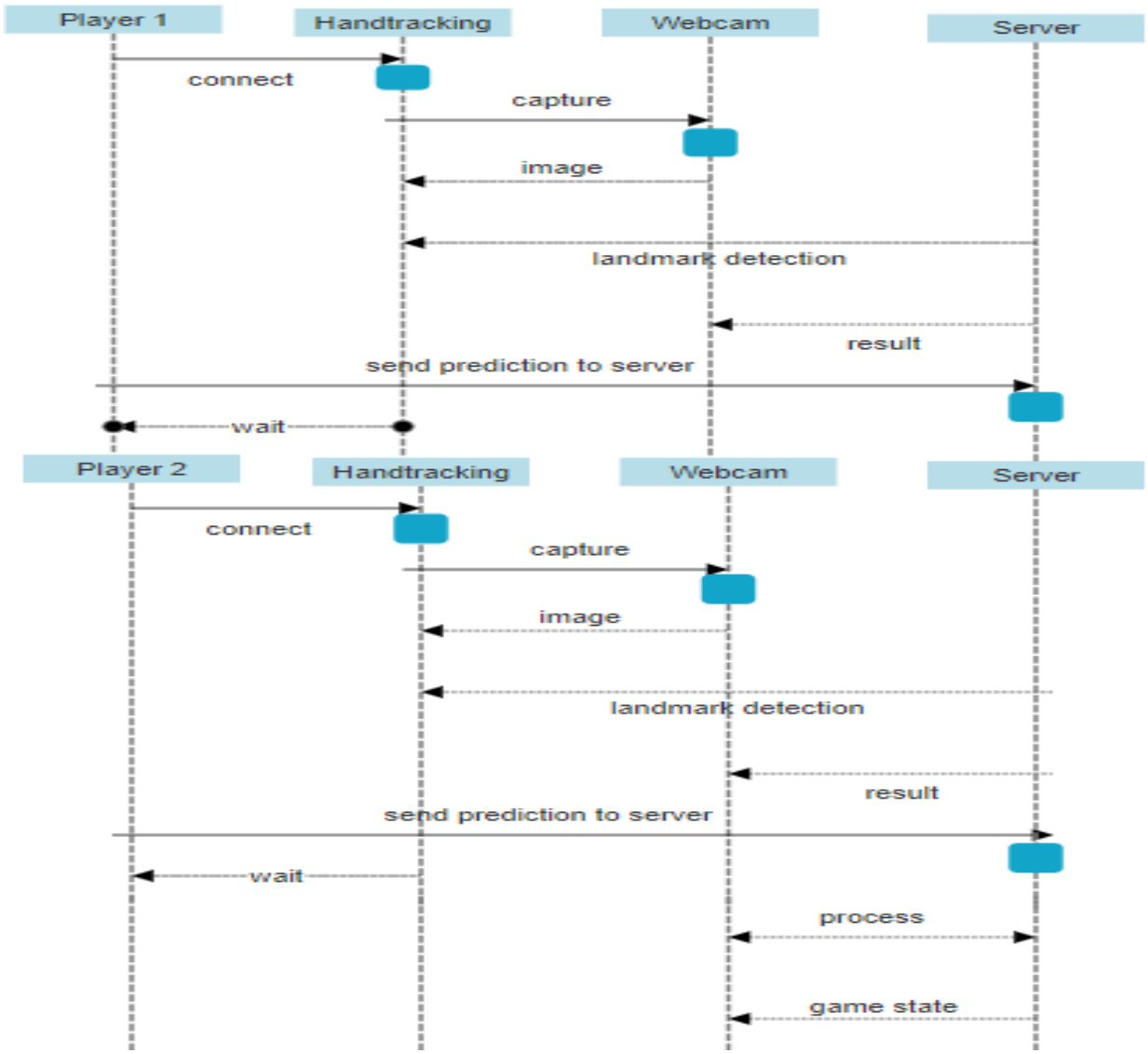
Fig 5.1.1 shows the use case diagram of our system which describes the interaction between actors which are the one will interact with the subjects. In our project there are mainly two actors involved in it namely User(clients) and Server. In this diagram we will going to see the interaction involved to them. 14 Initially, User gives the hand gesture input in which our tool reads it. Then the inputs from two users are matched. Finally, the Final score is

displayed until matched. So, these diagram helps our model the interaction between the system and the user.Fig 5.1.2 can also be referred.



5.1.2

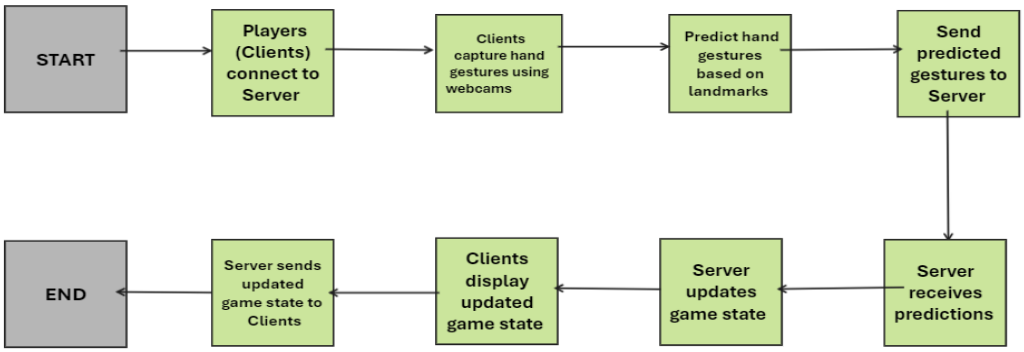
5.2 Sequence Diagram



A sequence diagram in UML is a kind of interaction diagram which shows how each process of the system operates with one another and in what order. It is a constructed as a message sequence chart. Sequence diagrams are sometimes called event diagrams or timing diagrams.

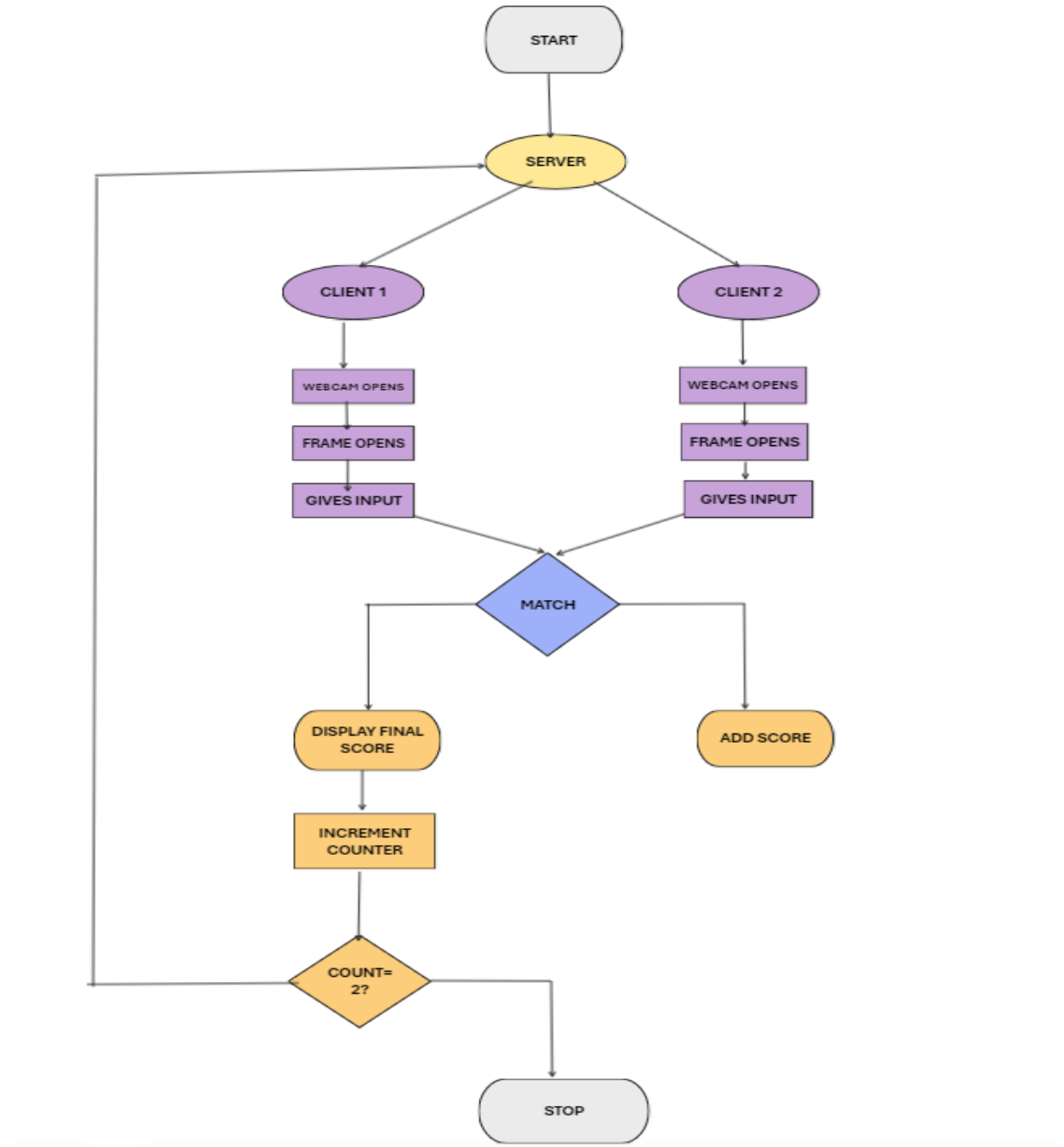
Fig 5.2 shows the Sequence diagram of our system which is an interaction diagram in which some sequence of information flow from one object to another object in a specified order to represent the time order of a process. It aims at a specific functionality of a model.

5.3 Activity Diagram

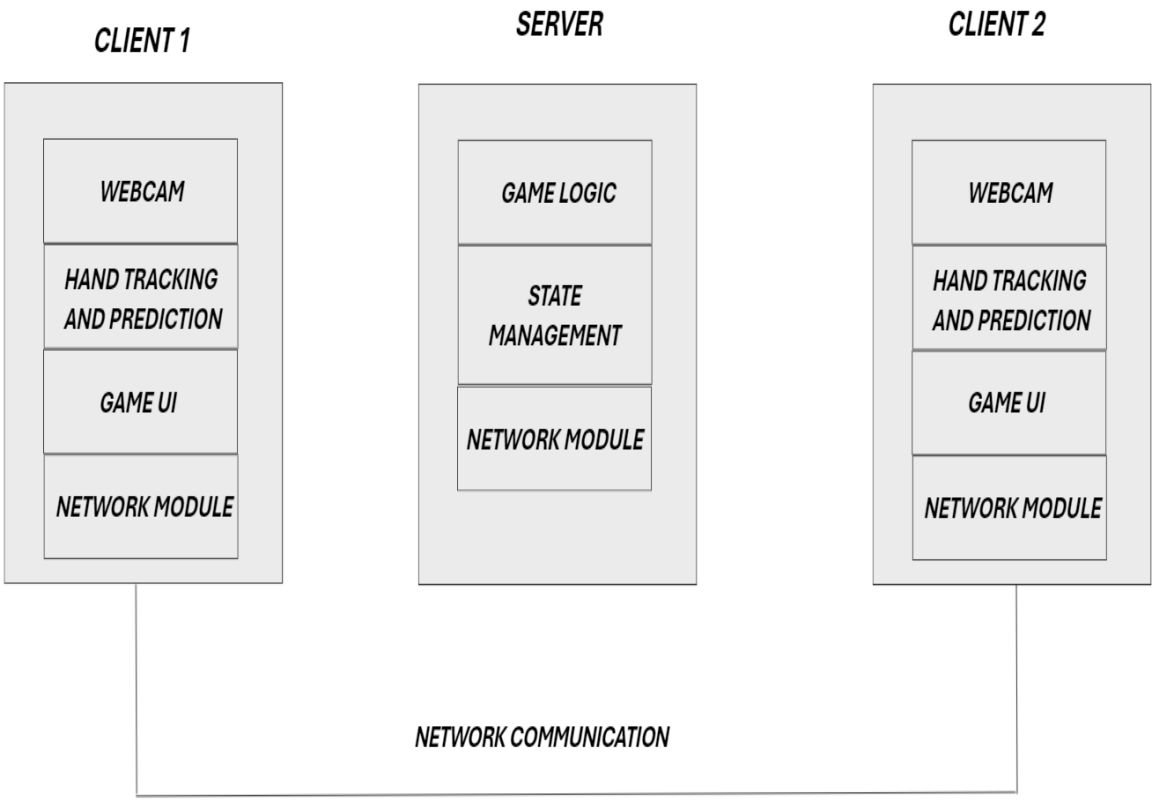


The game begins with the initialization of the clients and server. Both clients then establish a connection to the server. Using their webcams, the clients capture images of the players' hand gestures. These captured images are processed to detect hand landmarks, and hand gestures are predicted based on these landmarks. The predicted gestures are subsequently sent from the clients to the server. Upon receiving the predictions from both clients, the server updates the game state according to the received data. The server then sends the updated game state back to the clients. The clients receive and display the updated game state to the players. This activity typically leads to the next round of gesture capture or the conclusion of the game.

5.4 State Diagram



5.5 System Architecture



Client 1 and Client 2 each have a webcam that captures images of the players' hand gestures. These images are then processed by the Hand Tracking and Prediction module to detect hand landmarks and predict gestures. The Game UI component displays the game state and user interface to the players. Additionally, the Network Module handles communication between the clients and the server. On the server side, the Game Logic component implements the rules and logic of the hand cricket game. The State Management module maintains the current state of the game and updates it based on the players' gestures. The server's Network Module manages network communication with the clients, ensuring seamless interaction between all components.

IMPLEMENTATION

The hand-cricket game employs gesture recognition to facilitate interactive gameplay. Users make specific hand gestures in front of a webcam, which acts as the input device for capturing and interpreting these gestures. This method of gesture-based control, often referred to as "air gestures," uses the webcam to detect and analyze hand movements, enabling a novel and intuitive way to interact with the game.

Hardware and Software Components

- **Webcam**: The webcam is utilized to capture real-time video of the player's hand gestures. It records the movements and transmits this data to the system for processing.
- **Hand Landmark Model**: The game leverages a pre-trained model bundle consisting of:
    - **Palm Detection Model**: This component detects the presence and location of hands within the video feed. It isolates the hand from the rest of the image to focus on gesture analysis.
    - **Hand Landmarks Detection Model**: Following palm detection, this model identifies key landmarks on the hand. It maps the positions of various points on the hand, including fingers and joints.

Gesture Recognition Process

- **Video Capture**: The webcam continuously captures video frames, which are analyzed in real-time.
- **Hand Detection**: The palm detection model scans each frame to find the hand, cropping the hand region for detailed examination.
- **Landmark Detection**: The cropped hand image is processed by the hand landmarks detection model, which identifies and tracks specific hand landmarks.

- **Gesture Interpretation**: The identified landmarks are used to recognize specific hand gestures. For example, different hand signs can be mapped to numeric values, such as 1 through 10.
- **Game Interaction**: Based on the recognized gestures, the game performs corresponding actions. Players take turns as batter or bowler, with gestures determining their inputs. The game updates scores and displays results based on the gestures interpreted during play.

This approach integrates computer vision techniques from OpenCV and MediaPipe to create an engaging and interactive hand-cricket experience, transforming hand gestures into game commands and actions.

Python Code for Client Implementation (Client.py)

```
import socket

server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server_socket.bind(('127.0.0.1', 12345))  # Bind to any IP address and port 12345
server_socket.listen(2)

print("Client is waiting for connections...")

conn1, addr1 = server_socket.accept()
print(f"Player 1 connected: {addr1}")

conn2, addr2 = server_socket.accept()
print(f"Player 2 connected: {addr2}")

# Now both players are connected, you can send and receive data
# Example: conn1.send(b'Hello Player 1')
# Example: conn2.send(b'Hello Player 2')
```

Python Code for Server Implementation (Server.py)

```
import socket

server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

```
server_socket.bind(('127.0.0.1', 12345))  # Bind to any IP address and port 12345
server_socket.listen(2)

print("Server is waiting for connections...")

conn1, addr1 = server_socket.accept()
print(f"Player 1 connected: {addr1}")

conn2, addr2 = server_socket.accept()
print(f"Player 2 connected: {addr2}")

# Now both players are connected, you can send and receive data
# Example: conn1.send(b'Hello Player 1')
# Example: conn2.send(b'Hello Player 2')
```

Python Code for Primary Test Case (Game.py)

```
def capture_hand_gesture():
    cap = cv2.VideoCapture(0)
    while True:
        ret, frame = cap.read()
        if not ret:
            break

        frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        results = hands.process(frame_rgb)
        if results.multi_hand_landmarks:
            data_aux = []
            x_ = []
            y_ = []
            for hand_landmarks in results.multi_hand_landmarks:
                for i in range(len(hand_landmarks.landmark)):
                    x = hand_landmarks.landmark[i].x
                    y = hand_landmarks.landmark[i].y

                    x_.append(x)
                    y_.append(y)

                for i in range(len(hand_landmarks.landmark)):
                    x = hand_landmarks.landmark[i].x
                    y = hand_landmarks.landmark[i].y
                    data_aux.append(x - min(x_))
                    data_aux.append(y - min(y_))
```

```
            prediction = model.predict([np.asarray(data_aux)])
            predicted_number = labels_dict[int(prediction[0])]
            return int(predicted_number)
        cv2.imshow('frame', frame)
        cv2.waitKey(1)
    cap.release()
    cv2.destroyAllWindows()

def game_loop():
    global is_batter, score
    while True:
        print(f"{'Batter' if is_batter else 'Bowler'}'s turn")
        time.sleep(3)  # Countdown
        number = capture_hand_gesture()
        print(f"Your number: {number}")
        client_socket.send(pickle.dumps(number))
        opponent_number = pickle.loads(client_socket.recv(1024))
        print(f"Opponent's number: {opponent_number}")

        if is_batter:
            if number == opponent_number:
                print(f"Out! Final score: {score}")
                break
            else:
                score += number
                print(f"Score: {score}")
        else:
            is_batter = not is_batter

if __name__ == "__main__":
    game_thread = threading.Thread(target=game_loop)
    game_thread.start()
    game_thread.join()
```

Python Code of User Interface (UI_Game.py)

```
import tkinter as tk
from tkinter import messagebox
import socket
import pickle
import cv2
```

```
import mediapipe as mp
import numpy as np
import threading
import time

# Load the model
model_dict = pickle.load(open('./model.p', 'rb'))
model = model_dict['model']

# Mediapipe setup
mp_hands = mp.solutions.hands
hands = mp_hands.Hands(static_image_mode=True, min_detection_confidence=0.3)
labels_dict = {0: '1', 1: '2', 2: '3', 3: '4', 4: '5', 5: '6', 6: '7', 7: '8', 8:
'9', 9: '10'}

# Networking setup
client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client_socket.connect(('127.0.0.1', 12345))  # Replace SERVER_IP with the
server's IP address

# Game state
is_batter = True
score = 0
opponent_number = None

# Tkinter setup
root = tk.Tk()
root.title("Hand Cricket Game")

status_label = tk.Label(root, text="Waiting for opponent...", font=("Helvetica",
16))
status_label.pack(pady=20)

score_label = tk.Label(root, text="Score: 0", font=("Helvetica", 16))
score_label.pack(pady=20)

def update_status(text):
    status_label.config(text=text)

def update_score(new_score):
    score_label.config(text=f"Score: {new_score}")
```

```
def capture_hand_gesture():
    cap = cv2.VideoCapture(0)
    while True:
        ret, frame = cap.read()
        if not ret:
            break

        frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        results = hands.process(frame_rgb)
        if results.multi_hand_landmarks:
            data_aux = []
            x_ = []
            y_ = []
            for hand_landmarks in results.multi_hand_landmarks:
                for i in range(len(hand_landmarks.landmark)):
                    x = hand_landmarks.landmark[i].x
                    y = hand_landmarks.landmark[i].y

                    x_.append(x)
                    y_.append(y)

                for i in range(len(hand_landmarks.landmark)):
                    x = hand_landmarks.landmark[i].x
                    y = hand_landmarks.landmark[i].y
                    data_aux.append(x - min(x_))
                    data_aux.append(y - min(y_))

            prediction = model.predict([np.asarray(data_aux)])
            predicted_number = labels_dict[int(prediction[0])]
            return int(predicted_number)
        cv2.imshow('frame', frame)
        cv2.waitKey(1)
    cap.release()
    cv2.destroyAllWindows()

def game_loop():
    global is_batter, score, opponent_number
    while True:
        update_status(f"{'Batter' if is_batter else 'Bowler'}'s turn")
        time.sleep(3)  # Countdown
        number = capture_hand_gesture()
        update_status(f"Your number: {number}")
        client_socket.send(pickle.dumps(number))
```

```
            opponent_number = pickle.loads(client_socket.recv(1024))
            update_status(f"Opponent's number: {opponent_number}")

            if is_batter:
                if number == opponent_number:
                    messagebox.showinfo("Game Over", f"Out! Final score: {score}")
                    break
                else:
                    score += number
                    update_score(score)
            is_batter = not is_batter

    if __name__ == "__main__":
        game_thread = threading.Thread(target=game_loop)
        game_thread.start()
        root.mainloop()
```

TESTING

7.1 Introduction

Testing is the debugging program is one of the most critical aspects of the computer programming triggers, without programming that works, the system would never produce an output of which it was designed. Testing is best performed when user development is asked to assist in identifying all errors and bugs. The sample data are used for testing. It is not quantity but the quality of the data used the matters of testing. Testing is aimed at ensuring that the system is accurate and efficient before live operation commands. Testing today is most effective when it is continuous, indicating that testing is started during the design, continues as the software is built out, and even occurs when deployed into production. Continuous testing means that organizations don't have to wait for all the pieces to be deployed before testing can start. Shift-left, which is moving testing closer to design, and shift-right, where end-users perform validation, are also philosophies of testing that have recently gained traction in the software community. When your test strategy and management plans are understood, automation of all aspects of testing becomes essential to support the speed of delivery that is required.

7.2 Types of Software Testing

Here are the software testing types: Typically Testing is classified into three categories:

i) Functional Testing

ii) Non-Functional Testing or Performance Testing

iii) Maintenance

7.2.1 Functional Testing

Functional Testing deals with:

i) Unit Testing

ii) System Testing

iii) Integration Testing

Unit Testing: This software testing basic approach is followed by the programmer to test the unit of the program. It helps developers to know whether the individual unit of the code is working properly or not. Integration testing: It focuses on the construction and design of the software. You need to see whether the integrated units are working without errors or not. 25 System testing: In this method, your software is compiled as a whole and then tested as a whole. This testing strategy checks functionality, security, and portability, among others.

7.2.2 Non-Functional Testing

Non-functional Testing deals with:

i) Performance

ii) Endurance

iii) Scalability

Performance: Performance refers to the capability of a system to provide a certain response time. server a defined number of users or process a certain amount of data.

Endurance: Endurance Testing is a type of Software Testing that is performed to observe whether an application can resist the processing load it is expected to have to endure for a long period. During endurance testing, memory consumption is considered to determine potential failures.

Scalability: Scalability refers to the characteristic of a system to increase performance by adding additional resources.

7.2.3 Maintenance

Maintenance deals with:

i) Regression

ii) Maintenance

Regression: Regression testing is testing existing software applications to make sure that a change or addition hasn't broken any existing functionality.

Maintenance: Software maintenance is a part of the Software Development Life Cycle

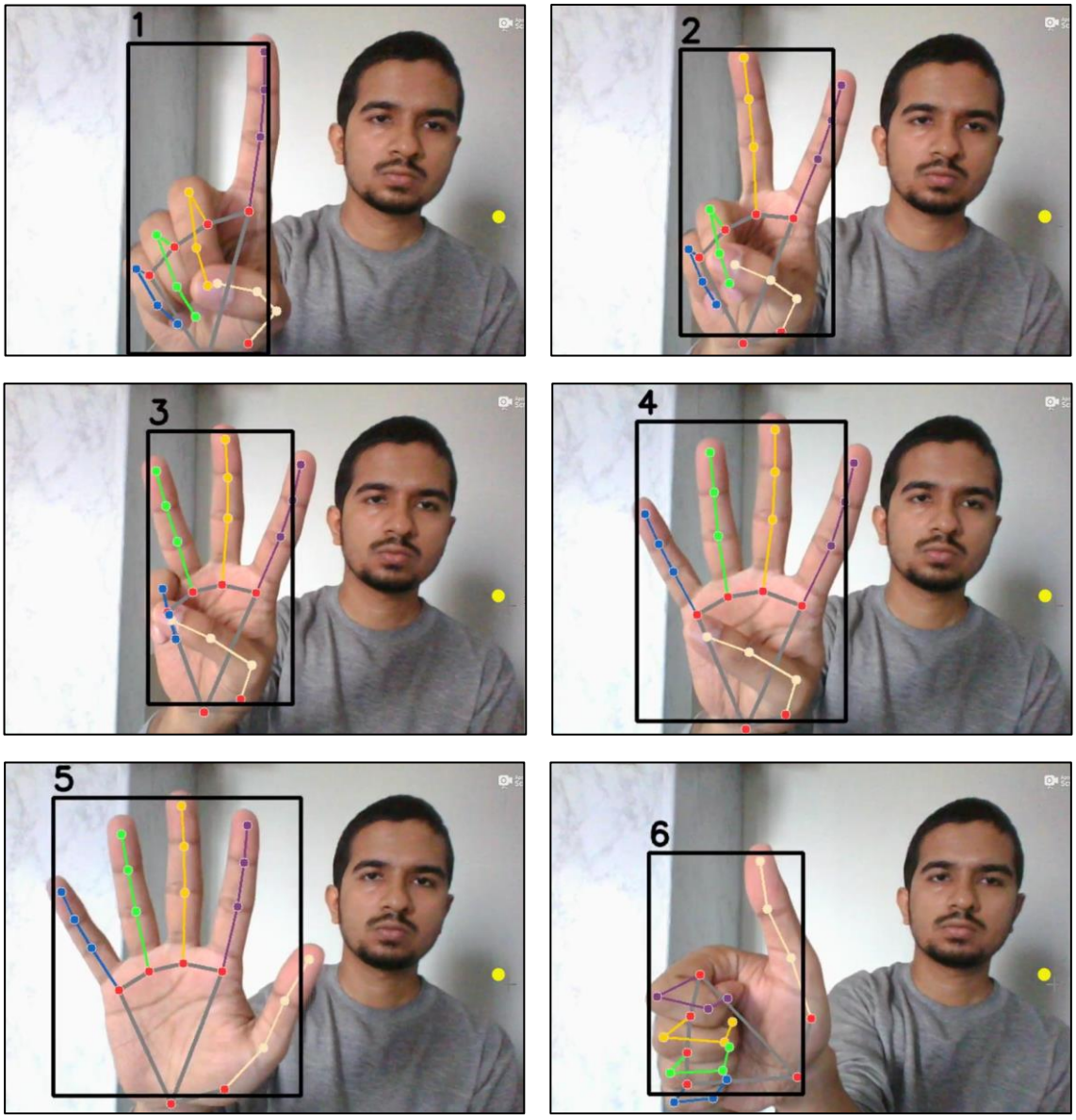
7.3 Test Cases

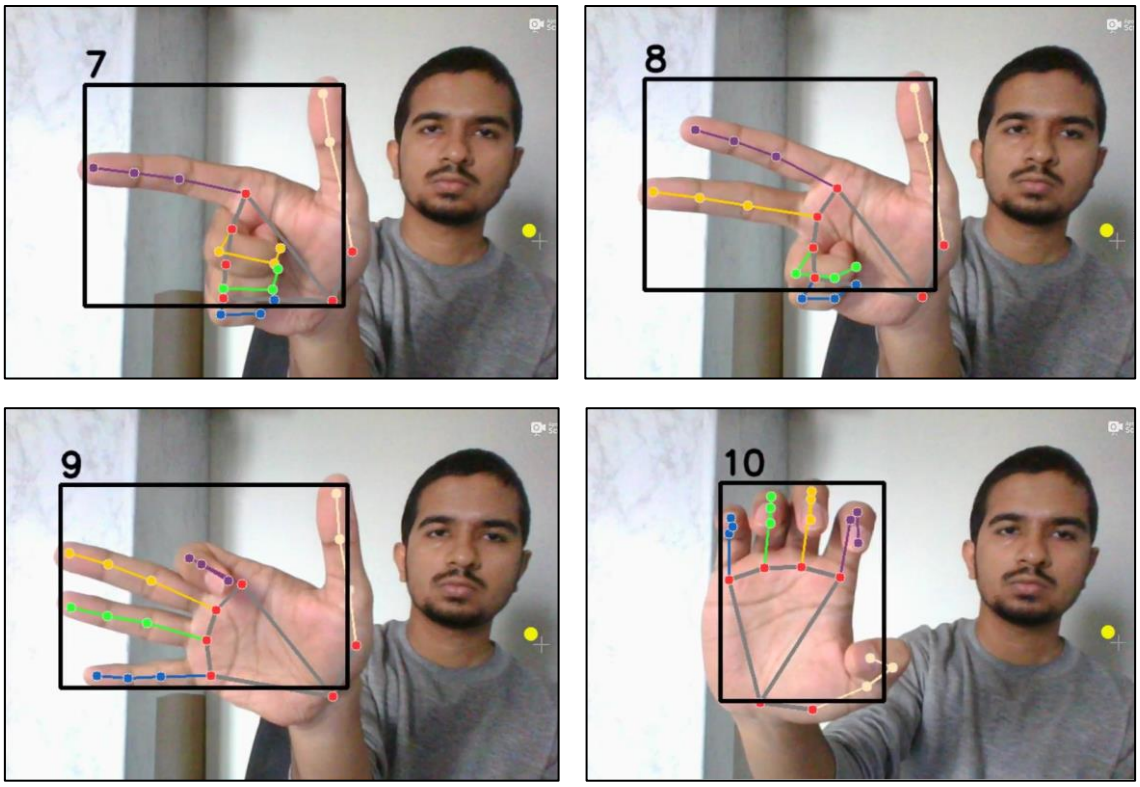
S.No	Description	Input	Expected Value	Actual Value	Result
1	Assessing model accuracy	Test dataset containing unlabeled images	Prediction accuracy ≥ 90%	Accuracy = 100%	PASS
2	Verifying correct number (1-10) recognition via gestures	Hand tracking through webcam	Accurate overlay of the corresponding number on the presented gesture	Accurate overlay of the corresponding number on the presented gesture	PASS
3	Testing multiplayer connectivity	Testing multiplayer connectivity	Successful establishment of connection between users	Successful establishment of connection between users	PASS
4	Validating score calculation functionality	Differing inputs using hand tracking through webcam	The number displayed by the batter is correctly added to the score	The number displayed by the batter is correctly added to the score	PASS

5	Verifying the 'out' functionality	Identical inputs using hand tracking through webcam	Display of 'Out' message and final score message	Display of 'Out' message and final score message	PASS
6	Display of 'Out' message and final score message	Identical inputs using hand tracking through webcam	Proper exchange of roles between Batter and Bowler	Proper exchange of roles between Batter and Bowler	PASS
7	Ensuring game state synchronization	Actions performed by both users	Consistent game state display on both players' screens	Consistent game state display on both players' screens	PASS
8	Verifying timer functionality for each turn	Actions performed by both users	Program captures input at intervals of 3 seconds	Program captures input at intervals of 3 seconds	PASS

Table Number 7.1: Test Cases

RESULT





8.1 Gesture Detection Overlay

```
Client is waiting for connections...
Player 1 connected: ('127.0.0.1', 54321)
Player 2 connected: ('127.0.0.1', 54322)
```

8.2 Output on running client.py

```
Server is waiting for connections...
Player 1 connected: ('127.0.0.1', 54321)
Player 2 connected: ('127.0.0.1', 54322)
```

8.3 Output on running server.py

```
[Status Label] Waiting for opponent...
[Score Label] Score: 0
```

8.4 Initial state on running ui_game.py

```
Batter's turn
Your number: 4
Opponent's number: 6
Score: 4

Bowler's turn
Your number: 2
Opponent's number: 2
Score: 6

Batter's turn
Your number: 3
Opponent's number: 5
Score: 9

Batter's turn
Your number: 5
Opponent's number: 5

Game Over: Out! Final score: 14
```

8.5 Sample game played out

CONCLUSION

The system's architecture, combining precise landmark detection with sophisticated gesture prediction algorithms, addresses the key challenges of accuracy, performance, and robustness. By ensuring compatibility with multiple hardware platforms and operating systems, we have made significant strides towards creating a widely accessible and user-friendly interface. Furthermore, the project's emphasis on security, privacy, and maintainability guarantees a reliable and secure experience for all users.

In conclusion, our project aims to revolutionize human-computer interaction by developing an advanced hand sign recognition system. By leveraging state-of-the-art technologies in computer vision and machine learning, we have created a platform that accurately detects and interprets hand gestures in real-time. Application of extensive python libraries and a robust client server architecture will allow for widespread communication applications that makes it possible to reach a wider audience on a global scale.

BIBLIOGRAPHY

1. https://google.github.io/mediapipe/
2. https://docs.opencv.org/
3. Deep Learning: An MIT Press Book By Ian Goodfellow and Yoshua Bengio and Aaron Courville Neural Networks and Learning Machines, Simon Haykin, 3rd Edition, Pearson Prentice Hall.
4. Zhang, L., et al. "A Real-Time Hand Gesture Recognition Method Based on Deep Learning Network." *2018 24th International Conference on Pattern Recognition (ICPR)*, 2018, pp. 3466-3471.
5. Molchanov, P., et al. "Hand Gesture Recognition with 3D Convolutional Neural Networks." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2015, pp. 1-7.
6. Ge, L., et al. "3D Hand Shape and Pose Estimation from a Single RGB Image." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 10833-10842.