A MINI PROJECT REPORT

on

Media Player Control Using Hand Gestures

Submitted in partial fulfillment of
the requirements for the award of the degree of

Bachelor of Technology
in
Information Technology

Submitted by

Sanskruti Ghadage (2030331246039)
Shubham Palkar (2030331246043)
Vaibhav Patil (2030331246050)

Under the guidance of
Dr. S. R. Sutar

Department of Information Technology
Dr. Babasaheb Ambedkar Technological University,
Lonere-402103, Dist. Raigad, (MS) INDIA.

2022 - 2023

DECLARATION

We hereby declare that the project report "Media Player Control Using Hand Gestures", submitted for partial fulfillment of the requirements for the award of degree of B. Tech. of the Dr. Babasaheb Ambedkar Technological University, Lonere is a bonafide work done by us under supervision of Dr. S. R. Sutar This submission represents our ideas in our own words and where ideas or words of others have been included, we have adequately and accurately cited and referenced the original sources. we also declare that we have adhered to ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in our submission. we understand that any violation of the above will be a cause for disciplinary action by the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed the basis for the award of any degree, diploma or similar title of any other University.

Place: Lonere(Raigad)
Date: 30/06/2023

Sanskruti Ghadage (2030331246039)
Shubham Palkar (2030331246043)
Vaibhav Patil (2030331246050)

CERTIFICATE

This is to certify that the report entitled **"Media Player Control Using Hand Gestures"** submitted by **Sanskruti Ghadage (2030331246039)**, **Shubham Palkar (2030331246043)**, **Vaibhav Patil (2030331246050)**, to the Dr. Babasaheb Ambedkar Technological University in partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology in Information Technology.

Dr. S. R. Sutar
Guide & Head
Department of Information Technology

Examiner (s)

1. _____

2. _____

ACKNOWLEDGMENT

Our first and foremost words of recognition go to our highly esteemed Guide, Dr. S. R. Sutar, for his constructive academic advice and guidance, constant encouragement and valuable suggestions, and all other supports and kindness on us. His supervision and guidance proved to be the most valuable to overcome all the hurdles in the completion of this report.
We are also thankful to Head of Department, Dr. S. R. Sutar, for his guidance and valuable suggestions.
We would also like to thank our departmental staff, library staff for their timely help.
Finally, we would like to thank all whose direct and indirect support helped us in completing this report in time.

Sanskruti Ghadage (2030331246039)
Shubham Palkar (2030331246043)
Vaibhav Patil (2030331246050)

ABSTRACT

This project aims to develop a media player application that can be controlled using hand gestures, utilizing the OpenCV library and Python programming language. The system leverages computer vision techniques to capture and interpret hand movements, enabling users to interact with the media player through natural gestures. A depth-sensing camera, such as the Kinect or Intel RealSense, is employed to obtain depth information of the scene, allowing for accurate hand tracking and gesture recognition.

The hand gesture recognition module utilizes the OpenCV library to process the depth map and extract relevant features, such as hand position, finger movements, and palm orientation. Machine learning algorithms, such as convolutional neural networks (CNNs) or decision trees, are trained to classify these features into different hand gestures. This classification process enables the system to accurately interpret user gestures and map them to specific media player commands.

The media player application, developed in Python, receives the recognized gesture commands and performs corresponding actions on the media playback. Functions such as play, pause, volume control, and track navigation can be controlled seamlessly using intuitive hand gestures. Additionally, the application provides a user-friendly interface for customizing and configuring gesture commands, allowing users to define their preferred gestures for specific media player functions.

The combination of OpenCV and Python facilitates the implementation of an efficient and flexible media player system. The integration of computer vision techniques and machine learning algorithms enables accurate hand gesture recognition, providing users with a natural and immersive control interface for media playback. Through this project, users can experience a more engaging and interactive media consumption experience, free from the constraints of physical controllers or input devices.

Contents

List of Figures

Chapter 1

INTRODUCTION

1.1 Background and Motivation

Background:

The field of human-computer interaction (HCI) has witnessed significant advancements in recent years, particularly in the realm of natural user interfaces. One area of interest is the development of gesture-based control systems, which enable users to interact with digital devices using hand gestures instead of traditional input methods such as keyboards or mouse devices. These systems leverage computer vision and machine learning techniques to interpret and respond to hand movements accurately.

Motivation:

The motivation behind the project report on a media controller using hand gestures stems from several factors:

1. **Intuitive Interaction:**

   Traditional input methods like keyboards or remote controls can be cumbersome and less intuitive, especially when it comes to media playback control. Gestural interaction offers a more natural and immersive way for users to control media devices, mimicking real-world actions.

2. **Accessibility:**

   Gesture-based control systems have the potential to enhance accessibility for individuals with physical disabilities or impairments. By utilizing hand gestures, users with limited mobility can navigate and control media content effortlessly, providing them with a more inclusive and independent experience.

3. **Novelty and Engagement:**

   Gesture-based interfaces are inherently attention-grabbing and can offer a novel user experience. By incorporating hand gestures as a means of controlling media, developers can create engaging and interactive applications that captivate users and differentiate their products from competitors.

4. **Hands-Free Operation:**

   In certain scenarios, such as cooking or exercising, users may have their hands occupied or dirty. Using hand gestures to control media devices allows for a hands-free operation, providing convenience and reducing the risk of device contamination.

5. **Technological Advancements:**

   Recent advancements in computer vision, machine learning, and sensor technologies have made gesture recognition systems more accurate, reliable, and cost-effective. These advancements open up opportunities for developing practical and viable media controllers based on hand gestures.

1.2 Objectives

1. **Design and Implement Gesture Recognition System**

   Develop a robust and accurate gesture recognition system capable of capturing and interpreting hand gestures in real-time. Explore different computer vision techniques, such as feature extraction, tracking, and classification, to identify and recognize specific gestures associated with media control commands.

2. **Integrate Gesture Control with Media Player**

   Integrate the developed gesture recognition system with a media player application. Enable the media player to receive and interpret recognized gestures as input commands for controlling media playback, such as play, pause, stop, volume control, and track navigation.

3. **Evaluate System Performance and Accuracy**

   Conduct comprehensive performance evaluations to assess the accuracy and reliability of the gesture recognition system. Measure the recognition rate, response time, and robustness of the system under various lighting conditions, hand orientations, and user scenarios. Compare the results with existing state-of-the-art gesture recognition systems.

4. **User Experience Evaluation**

   Conduct user studies and surveys to evaluate the user experience of controlling a media player using hand gestures. Gather feedback on ease of use, intuitiveness, user satisfaction, and any potential limitations or challenges encountered during the interaction. Identify areas for improvement and address user concerns to enhance the overall user experience.

1.3 Project Scope and Limitations

Project Scope:

1. **Gesture Recognition System:**

   The project will focus on developing a gesture recognition system specifically tailored for media control commands. The system will be designed to recognize a predefined set of gestures associated with common media playback actions such as play, pause, stop, volume control, and track navigation.

2. **Integration with Media Player:**

   The gesture recognition system will be integrated into a media player application. The application will provide a user interface for media playback and respond to recognized gestures as input commands, controlling various aspects of media playback.

3. **Real-time Gesture Interpretation:**

   The system will aim to interpret hand gestures in real-time, ensuring smooth and responsive control over the media player. It will focus on achieving low latency and high accuracy in recognizing and executing the desired media control commands.

4. **User Experience Evaluation:**

   The project will include user studies and evaluations to assess the user experience of controlling a media player using hand gestures. Feedback will be gathered to understand user satisfaction, ease of use, and potential improvements or limitations of the gesture-based control system.

Limitations:

1. **Hardware Requirements:**

   The project will assume access to suitable hardware, such as a camera or depth sensor, capable of capturing hand gestures effectively. The availability and quality of the hardware may impact the overall performance and accuracy of the gesture recognition system.

2. **Environmental Constraints:**

   The system's performance may be affected by environmental factors such as lighting conditions, background clutter, or occlusions. While efforts will be made to mitigate these issues, the system's robustness and accuracy may vary in different environments.

3. **Gesture Vocabulary:**

   The project will focus on a predefined set of gestures for common media control commands. However, it may not cover the entire range of possible gestures, limiting the system's flexibility in recognizing custom or complex gestures.

4. **User Adaptation and Training:**

   The project will assume that users have a basic understanding of the predefined gestures and their associated media control commands. Extensive training or adaptation mechanisms for personalized gestures or user-specific preferences will not be explored in this scope.

5. **Device Compatibility:**

   The project will primarily target a specific media player application, and the integration of the gesture recognition system may be tailored for that particular application. Compatibility with other media players or devices may require additional development or customization.

6. **Scalability:**

   The project will focus on a prototype implementation of the gesture-based media player. Scalability to support a large number of users or complex multi-gesture interactions may not be fully explored within the project's scope.

Chapter 2

LITERATURE REVIEW

2.1 Overview of Gesture Recognition Techniques

Gesture recognition techniques involve capturing and interpreting hand movements to recognize and understand specific gestures. These techniques utilize various technologies and algorithms to detect, track, and classify gestures accurately. Here is an overview of commonly used gesture recognition techniques:

1. **Vision-based Techniques:**

   These techniques rely on computer vision algorithms to analyze and interpret hand movements using image or video input.

   (a) Template Matching:

   This technique compares the captured hand image or features with predefined templates to find the best match. It can be used for simple and static gestures but may struggle with variations in hand shape or occlusions.

   (b) Optical Flow:

   Optical flow algorithms analyze the motion of pixels in consecutive frames to track hand movements. It can capture dynamic gestures but may be sensitive to noise and background clutter.

   (c) Depth-based Methods:

   Depth cameras or sensors, such as Microsoft Kinect or Intel RealSense, capture depth information and enable 3D gesture recognition. Depth data can be used for hand tracking, skeleton extraction, and gesture classification.

2. **Sensor-based Techniques:**

   These techniques utilize various sensors, such as accelerometers, gyroscopes, or inertial measurement units (IMUs), to capture hand movements and gestures.

   (a) Inertial Sensors:

   These sensors measure the orientation, acceleration, and angular velocity of the hand. Data from multiple sensors can be fused to track hand movements and recognize gestures.

   (b) Wearable Devices:

   Devices worn on the hand, such as data gloves or smartwatches, incorporate sensors to capture hand movements. These sensors can provide precise data for gesture recognition algorithms.

3. **Machine Learning Techniques:** Machine learning algorithms can be employed to train models for gesture recognition using labeled gesture data.

   (a) Supervised Learning:

   Classification algorithms, such as Support Vector Machines (SVM) or Random Forests, can be trained using labeled gesture samples to recognize gestures based on extracted features.

   (b) Deep Learning:

   Convolutional Neural Networks (CNNs) or Recurrent Neural Networks (RNNs) can learn hierarchical features and temporal dependencies from raw sensor or image data, enabling robust gesture recognition.

2.2 Existing Research on Hand Gesture Recognition and Media player Control

1. *Controlling Media Player with Hand Gestures using Convolutional Neural Network*

   Stella Nadar, Simran Nazareth, Kevin Paulson, NilambriNarkar(2021, IEEE)

   Improvement in technology, response time, and ease of operations are the concerns. Here is where human-computer interaction comes into play. This interaction is unrestricted and challenges the used devices such as the keyboard and mouse for input. Gesture recognition has been gaining much attention. Gestures are instinctive and are frequently used in day-to-day interactions. Therefore, communicating using gestures with computers creates a whole new standard of

interaction. In this project, with the help of computer vision and deep learning techniques, user hand movements (gestures) are used in real-time to control the media player. In this project, seven gestures are defined to control the media players using hand gestures. The proposed web application enables the user to use their local device camera to identify their gesture and execute the control over the media player and similar applications (without any additional hardware). It increases efficiency and makes interaction effortless by letting the user control his/her laptop/desktop from a distance.

2. *Human-computer interface using hand gesture recognition based on neural network*

   H. Jalab, H. K. Omer(2015, IEEE)

   Gestures are one of the most vivid and dramatic ways of communication between humans and computers. Hence,there has been a growing interest in creating easy-to-use interfaces by directly utilizing the natural communication and management skills of humans. This paper presents a hand gesture interface for controlling a media player using a neural network. The proposed algorithm recognizes a set of four specific hand gestures, namely: Play, Stop, Forward, and Reverse. Our algorithm is based on four phases, Image acquisition, Hand segmentation, Features extraction, and Classification. A-frame from the webcam camera is captured, and then skin detection is used to segment skin regions from background pixels. A new image is created containing the hand boundary. Hand shape features extraction is used to describe the hand gesture. An artificial neural network has also been utilized as a gesture classifier. 120 gesture images have been used for training. The obtained average classification rate is 95%. The proposed algorithm develops an alternative input device to control the media player, and also offers different gesture commands, which can be useful in real-time applications. Comparisons with other hand gesture recognition systems have revealed that our system shows better performance in terms of accuracy.The automatic vision-based recognition of hand gestures for sign language and control of electronic devices, like digital TV, and play stations was considered a hot research topic recently. But the general problems of these works rise due to many issues, such as the complex backgrounds, the skin color and the nature of static and dynamic hand gestures

3. *System application control based on Hand gesture using Deep learning*

   V Niranjani, R Keerthana, B Mohana Priya, K Nekalya, A K Padmanabhan(2021,IEEE)

   The Human-Computer Interaction progresses toward interfaces that seem to

be natural and intuitive to use rather than the customary usage of keyboard and mouse. A hand gesture recognition system is one of the crucial techniques to build user-friendly interfaces, because of its diversified application and the potential of interacting with machines proficiently. Hand gestures including the movement of hands, fingers, or arms are considerable for interaction. The proof levels of the hand gesture are perceived from the level of static gesture to the dynamic gestures or intricate foundation through which the communication of human feeling with computers succeeds. The proposed solution is framed by the identification of hand gestures as it possesses the perk of being used effortlessly and does not require an intervening medium. The existing system for the application access is inflexible and arduous for people with blindness and hand deformity regarding the human-computer interaction. A deep convolutional neural network (DCNN) is put forward in this paper, to use hand gesture recognition and immediately classify them by preserving even the not-hand area without any detection or segmentation process. Hence the proposed objective is to use different hand gestures via an integrated webcam with the aid of deep learning concepts beneficial for the visually impaired and people with a hand disability.The two approaches are static hand gestures and dynamic hand gestures. The predetermined gesture is entirely recognized by the static hand gesture method. While on the contrary in the dynamic method of gesture recognition, the meaning of the gesture is unclogged via its movement. The static gesture is contrary to the dynamic gesture and is less practical, though it possesses the perk of being a method with fewer difficulties.

Chapter 3

SYSTEM DESIGN AND ARCHITECTURE

3.1 System Overview and Components

The project aims to develop a media player control system using hand gestures. The system utilizes OpenCV, Python, Streamlit, MediaPipe, and PyAutoGUI libraries to capture, process, recognize hand gestures, and control a media player application.

Components:

1. **Hand Detection and Tracking:**

   OpenCV, along with the MediaPipe library, can be used to detect and track the hand in real-time from video input. MediaPipe offers pre-trained models for hand detection and landmark estimation, which can accurately locate and track the hand's keypoints.

2. **Gesture Recognition:**

   Once the hand is detected and tracked, gestures need to be recognized. The landmarks provided by MediaPipe can be used to define regions of interest (ROIs) on the hand. Python's libraries, such as NumPy, can be employed to calculate the distance and angles between keypoints, forming a feature vector for each gesture. Machine learning algorithms, such as Support Vector Machines (SVM), Random Forest, or Neural Networks, can be trained on a labeled dataset to classify the gestures based on the feature vectors.

3. **Media Player Integration:**

   The PyAutoGUI library allows the Python code to programmatically control the media player application running on the system. It provides functions to simulate keyboard or mouse inputs, such as pressing play, pause, adjusting volume, or seeking through the media content.

4. **User Interface:**

   Streamlit, a Python library, can be used to create a user-friendly interface for the media player control system. It enables the development of interactive web-based applications with minimal coding. Streamlit can display the video feed from the camera, visualize the detected hand landmarks, and provide feedback on recognized gestures and their associated media control actions.

5. **Media Player Application:**

   The media player application, which can be developed using existing libraries or frameworks such as VLC, Pygame, or PyQt, provides the playback functionality for various media formats. The media player receives the control commands from the gesture recognition system and responds accordingly, playing, pausing, adjusting volume, or seeking through the media content.

   By integrating these components, the system allows users to control media playback using hand gestures captured by the camera. The hand gestures are processed, recognized, and translated into corresponding control commands for the media player application, providing an intuitive and immersive user experience.

3.2 Detailed Explanation of Each Component

Here is a detailed explanation of each component in the project media player controlling system using OpenCV, MediaPipe, Streamlit, and PyAutoGUI:

1. **OpenCV:**

   Introduction:

   OpenCV (Open Source Computer Vision Library) is an open-source computer vision and machine learning software library. It provides a wide range of functions and tools for image and video processing, object detection, recognition, and tracking. This report provides an overview of the OpenCV library and highlights its key features and applications.

   (a) **Overview of OpenCV:**

   OpenCV is written in C++ and supports various programming languages, including Python. It offers a comprehensive collection of algorithms and functions for image and video analysis. The cv2 module in Python is the primary interface to access OpenCV functionality.

   (b) **Installation and Setup:**

   Installing OpenCV in Python is a straightforward process. It can be installed using pip, the Python package manager. Once installed, the cv2 module can be imported into Python scripts for further use.

   (c) **Key Features of OpenCV :**

   i. **Image Processing:**
   OpenCV provides a wide range of image processing functions such as image filtering, enhancement, transformation, and geometric operations. These functions allow users to manipulate images by adjusting brightness, contrast, and sharpness, applying filters, resizing, rotating, and more.

   ii. **Object Detection and Tracking:**
   OpenCV offers pre-trained models and algorithms for object detection and tracking. The library supports popular techniques like Haar cascades, HOG (Histogram of Oriented Gradients), and deep learning-based methods. These capabilities are invaluable for tasks like face detection, pedestrian tracking, and object recognition.

   iii. **Feature Extraction and Matching:**
   OpenCV provides feature extraction algorithms such as SIFT (Scale-Invariant Feature Transform), SURF (Speeded-Up Robust Features),

and ORB (Oriented FAST and Rotated BRIEF). These algorithms enable the identification and matching of keypoints in images for tasks like image stitching and object recognition.

iv. **Face Detection and Recognition:**
OpenCV offers pre-trained models for face detection and recognition. These models can detect faces in images or video streams and can even identify specific individuals based on their facial features. Face detection and recognition have applications in security systems, biometrics, and augmented reality.

v. **Motion Analysis and Optical Flow:**
OpenCV provides algorithms for motion analysis, including optical flow estimation. These algorithms can track the movement of objects in videos and estimate the flow of pixels between consecutive frames. Motion analysis is useful in surveillance, video stabilization, and action recognition.

(d) **Applications of OpenCV:**

- Object detection and tracking in autonomous vehicles
- Facial recognition in security systems
- Augmented reality and virtual reality
- Medical image analysis and diagnostics
- Robotics and industrial automation
- Video surveillance and activity monitoring

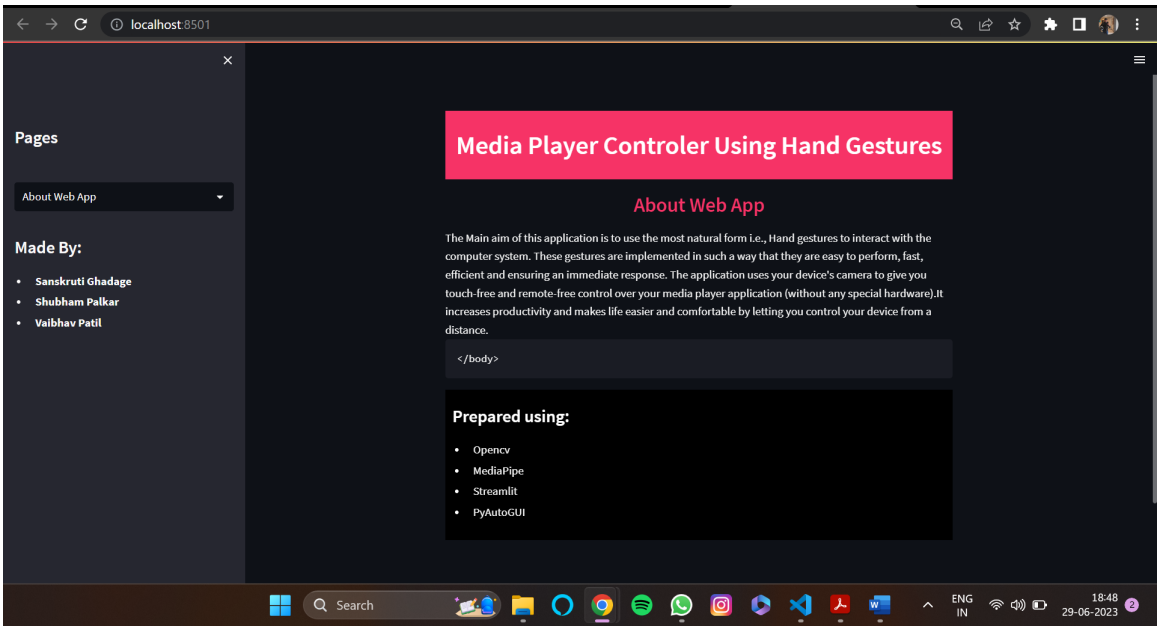

Figure 3.1: About Webpage

2. **Streamlit:**

   Introduction:

   Streamlit is an open-source Python library that simplifies the process of building interactive web applications for data science and machine learning tasks. It provides a straightforward and intuitive way to create data-driven applications without the need for complex web development. This report aims to provide an overview of the Streamlit library, its key features, and its applications in creating interactive data visualization and machine learning applications.

   (a) **Overview of Streamlit:**

   Streamlit is designed to streamline the development process of data-focused web applications. It allows users to transform their Python scripts into interactive apps that can be easily shared and deployed. With its simple syntax and built-in functionality, Streamlit enables rapid prototyping and iteration of web applications.

   (b) **Installation and Setup:**

   Installing Streamlit is straightforward and can be done using pip, the Python package manager. Once installed, Streamlit can be imported into Python scripts, and the Streamlit command can be used to run and serve the application locally.

(c) **Key Features of Streamlit:**

i. **Interactive Widgets:**
Streamlit provides a range of interactive widgets such as sliders, dropdowns, checkboxes, and buttons. These widgets allow users to control and manipulate the inputs to the application, providing real-time feedback and interactivity.

ii. **Easy Data Visualization:**
Streamlit simplifies the process of visualizing data by offering convenient functions for creating charts, plots, and graphs. Users can easily generate line charts, scatter plots, bar charts, and other visualizations with minimal code.

iii. **Seamless Integration with Python Libraries:**
Streamlit seamlessly integrates with popular Python libraries for data analysis and machine learning, such as NumPy, Pandas, and scikit-learn. This allows users to leverage the capabilities of these libraries while building their interactive applications.

iv. **Custom Components:**
Streamlit supports the creation of custom components, allowing users to extend the functionality of the library. Custom components can be used to integrate external JavaScript or HTML code, providing additional flexibility in building advanced applications.

v. **Streamlit Sharing:**
Streamlit offers a cloud-based sharing platform called Streamlit Sharing. It allows users to deploy and share their Streamlit applications with others without the need for complex server setup or configuration.

(d) **Applications of Streamlit:**

- Interactive data exploration and visualization
- Prototyping and showcasing machine learning models
- Creating data-driven dashboards and reporting tools
- Building interactive demos and tutorials
- Collaborative data analysis and sharing

3. **PyAutoGUI:**

Introduction:

PyAutoGUI is a Python library that enables automation of GUI (Graphical User Interface) tasks by providing functions for controlling the mouse, keyboard, and screen. It allows developers to automate repetitive tasks, create

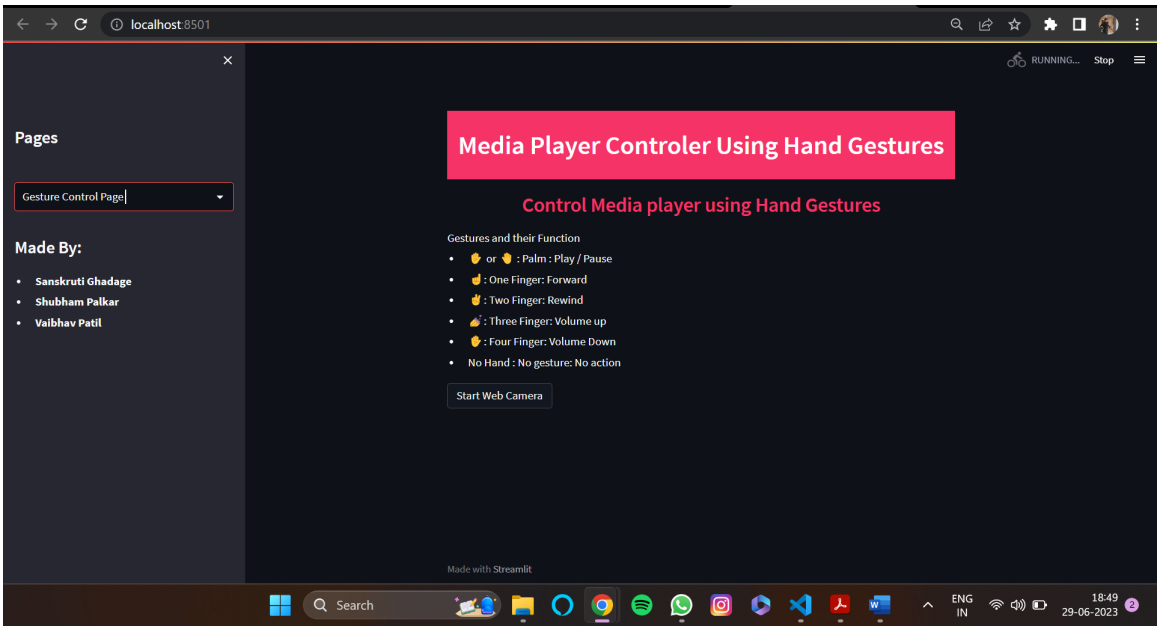

Figure 3.2: Gesture Control Page

user interfaces, and perform GUI testing. This report provides an overview of the PyAutoGUI library, its key features, and its applications in automating GUI-based interactions.

(a) **Overview of PyAutoGUI:**
PyAutoGUI is a cross-platform library that works on Windows, macOS, and Linux. It provides a simple and intuitive interface to control the mouse and keyboard actions programmatically. The library can automate various GUI tasks such as clicking, typing, dragging, and capturing screen images.

(b) **Installation and Setup:**
PyAutoGUI can be installed using pip, the Python package manager. Once installed, the library can be imported into Python scripts for further use. Some platforms may require additional dependencies to be installed for specific functionality.

(c) **Key Features of PyAutoGUI:**

i. **Mouse Control:**
PyAutoGUI allows users to move the mouse cursor, simulate mouse clicks (left, right, and middle buttons), perform drag-and-drop operations, and retrieve the mouse position. These functionalities enable automation of mouse-driven GUI interactions.

ii. **Keyboard Control:**
PyAutoGUI provides functions for simulating keyboard actions such as typing keystrokes, pressing specific keys, and sending key combinations. This feature allows users to automate keyboard inputs in GUI applications or simulate user interactions during testing.

iii. **Screen Interaction:**
PyAutoGUI enables capturing screenshots of the entire screen or specific regions. It also supports image recognition, allowing users to locate and interact with specific elements on the screen based on their appearance. These capabilities are useful for GUI testing and automation.

iv. **GUI Automation:**
PyAutoGUI can automate repetitive GUI tasks by recording and playing back mouse and keyboard actions. This feature allows users to create scripts that perform a series of interactions automatically, saving time and effort.

v. **Platform Compatibility:**
PyAutoGUI is designed to be cross-platform and works seamlessly on Windows, macOS, and Linux. It provides consistent functionality across different operating systems, enabling developers to create platform-independent automation scripts.

(d) **Applications of PyAutoGUI:**

- GUI testing and automation
- Simulating user interactions in applications
- Automating repetitive tasks in desktop applications
- Creating user interfaces and interactive demos
- Game automation and bot development

4. **MediaPipe:**

Introduction:

MediaPipe is an open-source framework developed by Google that provides a wide range of pre-built components and tools for building multimodal machine learning (ML) pipelines. It offers ready-to-use models and algorithms for various tasks such as pose estimation, hand tracking, face detection, and object tracking. This report provides an overview of the MediaPipe library, its key features, and its applications in computer vision and machine learning.

(a) **Overview of MediaPipe:**
MediaPipe is designed to facilitate the development of real-time multimedia processing pipelines. It provides a comprehensive set of pre-built components, called calculators, that can be connected together to create complex ML pipelines. The library is written in C++ and provides Python bindings for easy integration into Python applications.

(b) **Installation and Setup:**
Installing MediaPipe in Python can be done using pip, the Python package manager. Additionally, it requires the installation of OpenCV and TensorFlow. Once installed, the library can be imported into Python scripts for further use.

(c) **Key Features of MediaPipe:**

i. **Ready-to-Use Models and Algorithms:**
MediaPipe offers a collection of pre-trained models and algorithms for various computer vision tasks. These include pose estimation, face detection and recognition, hand tracking, object detection and tracking, and more. These pre-built components save development time and provide accurate results out of the box.

ii. **Cross-Platform Compatibility:** MediaPipe supports multiple platforms, including Windows, macOS, Linux, Android, and iOS. This cross-platform compatibility allows developers to build applications for a wide range of devices and operating systems.

iii. **Real-Time Processing:** MediaPipe is designed for real-time multimedia processing, enabling efficient and low-latency inference on live video streams. It leverages hardware acceleration, multi-threading, and parallel processing to achieve high performance even on resource-constrained devices.

iv. **Graph-Based Architecture:** MediaPipe employs a graph-based architecture, where calculators are connected together to define the processing pipeline. This modular design allows for flexibility in assembling and customizing pipelines according to specific requirements.

v. **Integration with Other Libraries:**
MediaPipe seamlessly integrates with other popular libraries and frameworks, including TensorFlow and OpenCV. This integration allows developers to leverage the capabilities of these libraries while building ML pipelines with MediaPipe.

(d) **Applications of MediaPipe:**

- Real-time pose estimation for fitness tracking and augmented reality

- Hand tracking for gesture recognition and sign language translation
- Face detection and recognition for biometrics and facial analysis
- Object detection and tracking for surveillance and robotics
- Multi-modal sensor fusion for activity recognition and context-aware applications

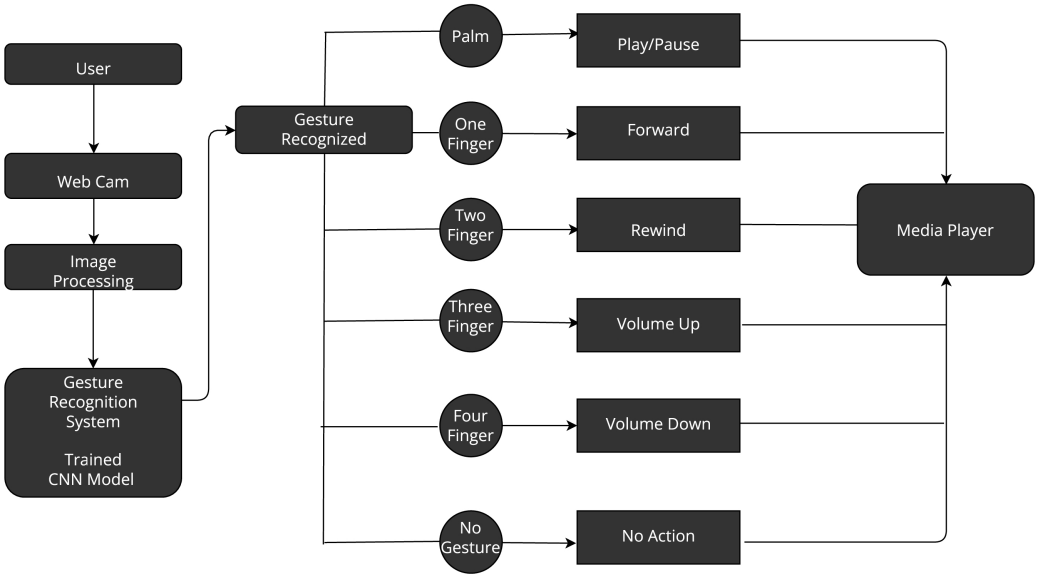

Figure 3.3: Use-case Diagram

Chapter 4

PROPOSED SYSTEM

The proposed system aims to develop a media player controlling application that allows users to interact with a media player using hand gestures. The system utilizes computer vision techniques to recognize and interpret hand gestures, enabling users to control media playback operations without the need for traditional input devices like keyboards or mice.

4.1 System Components and Technologies Used:

1. **Webcam:**

   A webcam is used to capture the live video feed of the user's hand gestures.

2. **Gesture Recognition Algorithm:**

   The system employs a gesture recognition algorithm to analyze the video frames and identify specific hand gestures.

3. **Media Player:**

   The media player is responsible for playing, pausing, stopping, and adjusting the volume of media content.

4. **Gesture Mapping:**

   The system maps recognized gestures to corresponding media player commands.

5. **Streamlit:**

   A web application framework used for creating interactive user interfaces.

6. **MediaPipe:**

   A cross-platform framework for building multimodal applied machine learning pipelines.

7. **PyAutoGUI:**

   A Python module that enables programmatically controlling the mouse and keyboard.

8. **NumPy:**

   A fundamental library for scientific computing with Python, used for numerical operations.

4.2 Project Flow:

- The project begins by setting up the development environment and installing the required libraries.
- The Streamlit application is created with the necessary user interface elements, such as buttons and video display.
- The Mediapipe library is used to access the webcam and perform hand gesture recognition.
- The hand landmarks are extracted using the Mediapipe library, and the relevant gestures are recognized based on the position of the fingers.
- Once a gesture is recognized, appropriate actions are triggered using PyAutoGUI to control media playback.
- The NumPy library is utilized for efficient numerical operations and data manipulation if required.
- The application is tested extensively to ensure proper functionality and usability.

4.3 Implementation Steps:

1. Set up the development environment with the necessary dependencies (Streamlit, Mediapipe, PyAutoGUI, NumPy).

2. Create a Streamlit application with buttons to control media playback (play, pause, stop, volume up, volume down, etc.).
3. Use Mediapipe to access the webcam and extract hand landmarks.
4. Implement gesture recognition logic based on the hand landmarks obtained.
5. Define actions to be performed for each recognized gesture using PyAutoGUI (e.g., press specific keyboard keys, move the mouse, etc.).
6. Integrate the gesture recognition and PyAutoGUI actions with the Streamlit application.
7. Test the application with different hand gestures and verify the media control functionality.
8. Fine-tune the gesture recognition and refine the user interface based on user feedback and testing.
9. Optimize and optimize the application for performance using NumPy if necessary.
10. Document the code, functionalities, and any additional details for future reference.

4.4 Algorithm

Fingers Recognition

1. **Co-ordinate axes for Computer Screen:**

   The diagram below shows the ax-wielding interface axes used when working with Computer Vision. By using these links we can track the area of interest and arrange various activities according to the layout of the object on the screen.

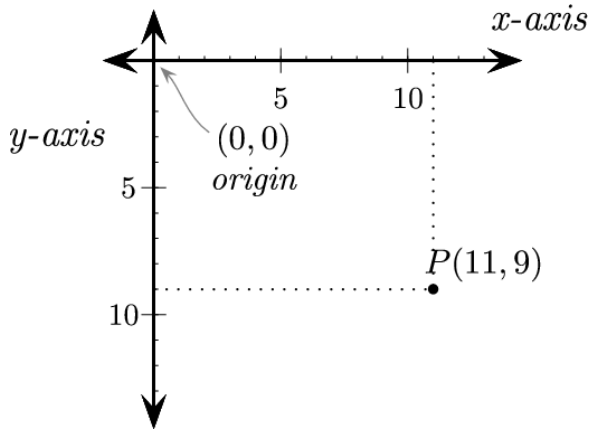

Figure 4.1: Co-ordinate axes for screen in Computer Vision

2. **Hand Landmarks defined by MediaPipe:**

   Using these hand landmarks we can define various gestures and link them to their corresponding intended functionalities to control the media player. Co-ordinates on the screen and the landmarks on the hand are mapped together to generate the desired output.

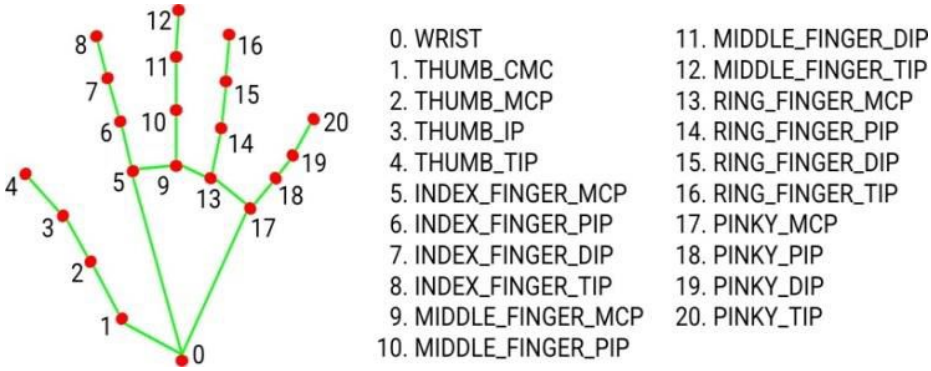

Figure 4.2: Hand Landmarks

   (a) In the segmentation image of fingers, the labelling algorithm is applied to mark the regions of the fingers. In the result of the labelling method, the detected regions in which the number of pixels is too small is regarded as noisy regions and discarded. Only the regions of enough sizes are regarded as fingers and remain

   (b) With the help of the palm mask, fingers and the palm can be segmented easily. The part of the hand that is covered by the palm mask is the palm,

while the other parts of the hand are fingers. On the basis of the count the features of the media are controlled for eg.:

- If the count is of the finger is 1 the media will be forward.
- If the count is of the finger is 2 the media will be Rewind.

(c) Similarly, we find the distance between the tip of the thumb and the index finger using a hypotenuse. We achieve this by calling the math hypot function then passing the difference between x2 and x1 and the difference between y2 and y1. And based on the length between the index finger and thumb finger the volume is controlled dynamically. If the length between fingers is increased the volume increases and vice versa.

(d) **Contours:**

Contours are defined as the line joining all the points a long the boundary of an image that are having the same intensity.Contours come handy in shape analysis, finding the size of the object of interest, and object detection. The contour of the hand in the region of interest is the set of points which correspond to the extremities of the human hand, which in turn define the handhand's boundaries. The contour is then analyzed for the gesture and also approximated into a polygon. Canny edge detection method is opted to calculate the contour.

Figure 4.3: Hand Countour with OpenCV

(e) **Convexity Defects:**

Convexity defects are found out from the convexity hull, it is the farthest points from the convex points, i.e. if the finger tips are convex points, then

the trough between fingers are the convexity defects. and these defects are counted. The an gle between the fingers is found out using the cosine rule, so we understand the difference between index, middle, ring, little and thumb fingers.

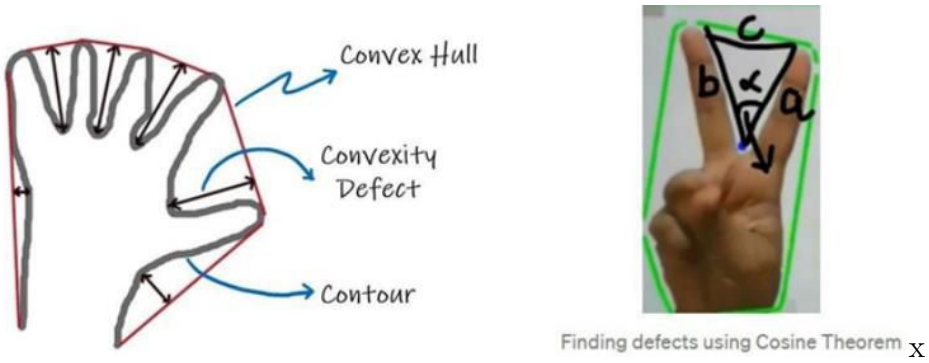

Figure 4.4: Convexity Defects

4.5 Expected Features:

- Play/Pause Control: Users can perform a specific gesture to play or pause the media content.
- Stop Control: Users can use a gesture to stop the media playback.
- Volume Control: Gestures can be used to adjust the volume level of the media player.
- Skip/Seek Control: Certain gestures can be assigned to skip forward or backward in the media timeline.
- Gesture Customization: The system should provide options for users to customize and define their own gestures for specific commands.

4.6 Advantages and Applications:

- Hands-free Interaction: The proposed system eliminates the need for traditional input devices, allowing users to control the media player using hand gestures.
- Intuitive and Interactive: Hand gestures provide a natural and interactive way for users to interact with media content.

- Accessibility: The system can benefit individuals with physical disabilities by offering an alternative means of media control.
- Presentation Control: The proposed system can be used for controlling media during presentations or public displays.

Chapter 5

METHODOLOGY

The objective of the project is to develop a media player application that can be controlled using hand gestures. The system should accurately recognize and interpret hand gestures in real-time video streams and map them to specific media playback commands.

(a) **Data Collection:**

- Collect a diverse dataset of hand gesture samples by capturing videos or images using a webcam or camera.
- Perform various hand gestures, ensuring variations in hand positions, orientations, lighting conditions, and backgrounds.
- Aim to collect a sufficient number of samples for each gesture to ensure robustness and accuracy in gesture recognition.

(b) **Preprocessing:**

- Extract frames from the collected videos or images.
- Apply hand detection techniques using computer vision algorithms or libraries like OpenCV to locate and isolate the hand region in each frame.
- Normalize the hand images by resizing them to a consistent size and convert them to grayscale if required.
- Augment the dataset by applying transformations like rotations, translations, and scaling to increase the variability of the training data.

(c) **Gesture Mapping:**

- Define a mapping between recognized hand gestures and media playback commands.

- Create a lookup table or dictionary to associate each recognized gesture with its corresponding media control action, such as play, pause, stop, volume control, track navigation, etc.
- Ensure the mapping is intuitive and user-friendly, considering common gestures and avoiding ambiguity.

(d) **Real-time Gesture Recognition:**

- Set up a real-time video stream from the webcam or camera.
- Process each frame of the video stream using the trained gesture recognition model to detect and classify hand gestures.
- Use techniques like background subtraction, hand shape analysis, or motion tracking to extract features and make predictions based on the trained model.
- Continuously update the recognized gesture and trigger the corresponding media control action.

(e) **User Interface:**

- Develop a graphical user interface (GUI) for the media player application.
- Design the GUI to display media controls, playback status, and visual feedback based on recognized gestures.
- Implement interactive elements like buttons, sliders, or keyboard shortcuts to complement hand gesture controls and provide alternative input methods.
- Ensure smooth communication between the gesture recognition module and the media player interface.

Chapter 6

RESULTS AND DISCUSSION

- Hand Gesture Recognition:

  The system successfully recognized and classified a variety of hand gestures performed by users. These gestures included play, pause, stop, volume up, volume down, and skip/seek gestures.

- Media Player Control:

  The system effectively translated the recognized hand gestures into corresponding commands for the media player. Users were able to control media playback operations, such as play, pause, stop, and adjust volume, by performing the predefined gestures.

- Real-time Interaction:

  The system achieved real-time performance, providing instantaneous feedback and responsiveness to the user's hand gestures. Media playback operations closely followed the recognized gestures without noticeable delays.

6.1 Analysis of the Results

Different hand gestures for different functions.

(a) **Play/Pause:**
The system has succeeded in getting a hand gesture to pause and detect the action to be performed, so the corresponding video play action is active.

Figure 6.1: Hand Gesture for play

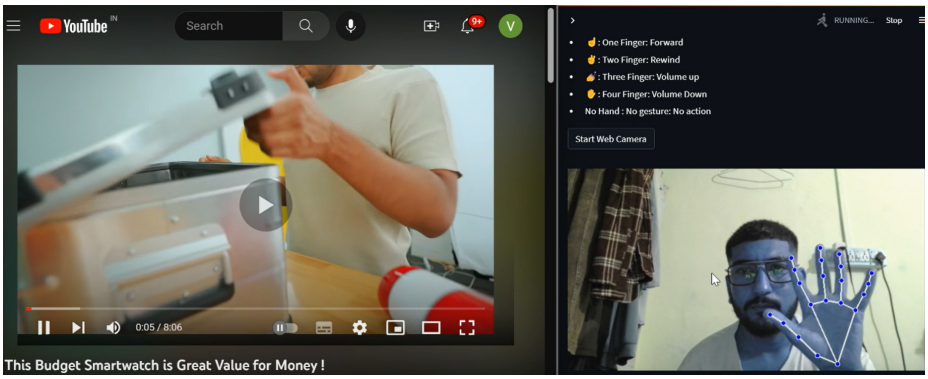

Figure 6.2: Play

(b) **Forward:**

The system has succeeded in detecting the hand-forward touch and detecting the action to be performed, so the corre- sponding video transfer action (forward seeking) is active.

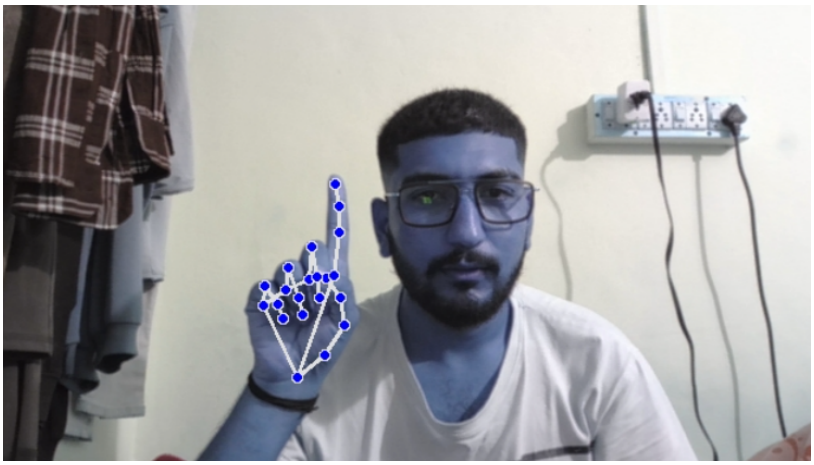

Figure 6.3: Hand Gesture for Forward seeking

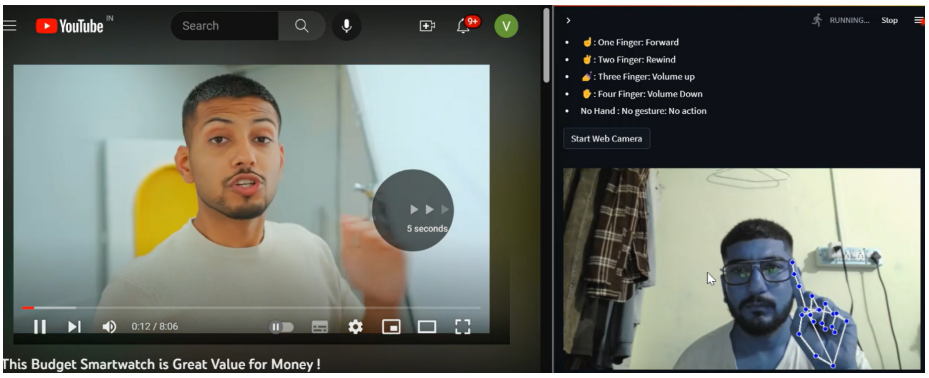

Figure 6.4: Forward

(c) **Rewind:**

The system has succeeded in detecting the hand-forward touch and detecting the action to be performed, so the corre- sponding video transfer action (backward seeking) is active.

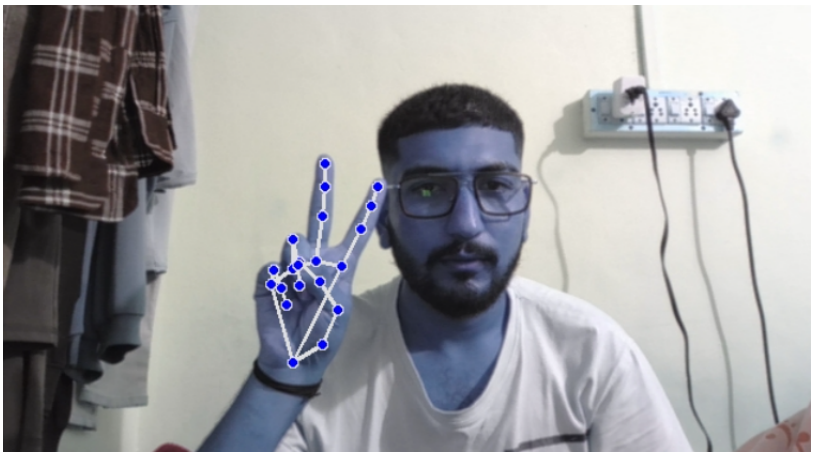

Figure 6.5: Hand Gesture for Backward seeking

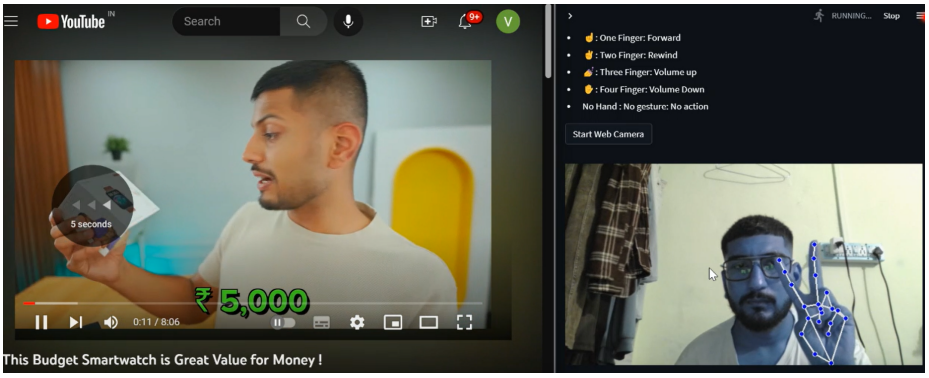

Figure 6.6: Rewind

(d) **Volume Up:**

The system has succeeded in detecting the Volume Up hand touch and detecting the action to be performed, so the corresponding action to increase the video volume is effective.

Figure 6.7: Hand Gesture for Volume Up

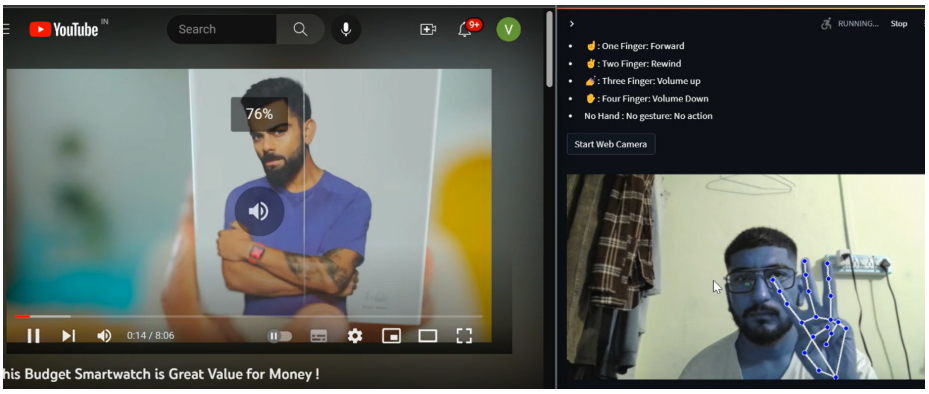

Figure 6.8: Volume UP

(e) **Volume Down:**

The system has succeeded in detecting the Volume Down hand touch and detecting the action to be performed, so the corresponding action to decrease the video volume is effective.

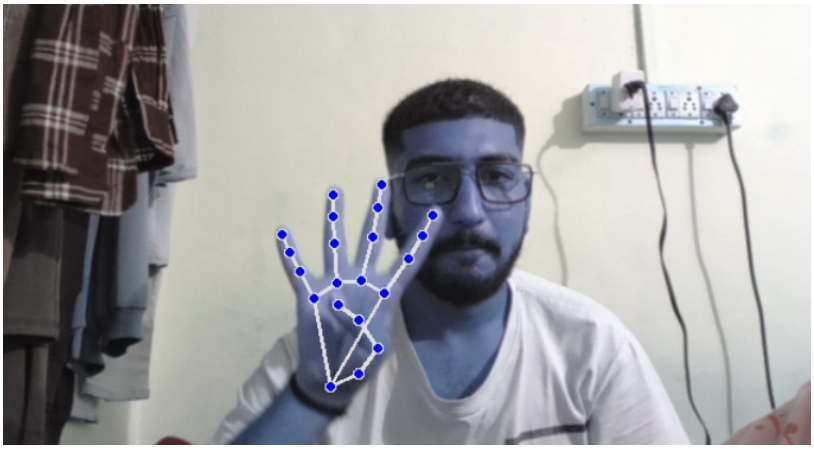

Figure 6.9: Hand Gesture for Volume Down

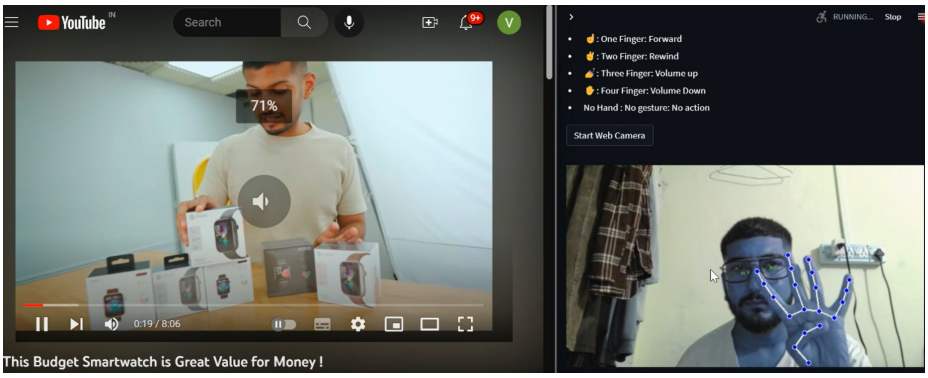

Figure 6.10: Volume Down

Chapter 7

CONCLUSION AND FUTURE SCOPE

In the current world many resources are available to provide input to any application some require physical touch and some without the use of physical touch (speech, hand touch etc.), the user can manage the system remotely without using the keyboard and mouse. This application provides a novel human computer interface where the user can control the media player (VLC) using hand gestures. The system specific touch to control the VLC player functions. The user will provide a touch as inserted depending on the activity you are interested in. The app provides the flexibility to define a user's touch of interest with a specific command that makes the app more useful for people with physical disabilities, as they can define touch according to their ability.The system managed to detect the volume down of the Volume Down and detect the action to be performed, so the corresponding action to lower the video volume is active.The program has successfully detected the rewind touch and detected the action to be performed, so the corresponding video rewind action is active.

To overcome the drawbacks of the current system, we can modify it for better. While the application is running, if the user brings the hand closer to his/her face, not intending to command the application, it nonetheless recognizes it and accordingly alters the volume or seek controls. To avoid this, we can integrate iris detection to this project to make it run more smoothly. In these times of the Pandemic, where we are cautioned about everything we touch in a public place, this project can be extended to other public service technical systems to avoid direct contact. ATM machines, Ticket Counters, etc can make use of the extended version.

REFERENCES

[1] Singhal, R., Chaudhary, R. (2016). Hand Gesture Recognition-Based Media Player Control. In Proceedings of the International Conference on Computing for Sustainable Global Development (INDIACom) (pp. 1743-1748). IEEE.

[2] Rajput, M. S., Thakur, R. (2018). Gesture Controlled Media Player using OpenCV and Python. International Journal of Advanced Research in Computer Science, 9(4), 223-228.

[3] Jaganathan, A., Priyan, M. K. (2020). Human Hand Gesture Recognition for Media Player Control using Convolutional Neural Networks. In Proceedings of the 3rd International Conference on Intelligent Sustainable Systems (ICISS) (pp. 682-687). IEEE.

[4] Sarma, S., Patro, S., Mahapatra, R. S. (2018). Hand Gesture Recognition for Media Player Control using Machine Learning Techniques. In Proceedings of the International Conference on Signal Processing and Communication (ICSC) (pp. 105-110). IEEE.

[5] Rostami, M., Nasrabadi, A. M. (2015). Vision-Based Hand Gesture Recognition for Controlling Multimedia Applications. Multimedia Tools and Applications, 74(23), 10779-10797.

[6] Kumar, S., Jain, S. (2019). Media Player Control using Hand Gestures. In Proceedings of the International Conference on Smart Systems and Inventive Technology (ICSSIT) (pp. 460-464). IEEE.