MEDIA PLAYER CONTROL USING HAND GESTURES DETECTION

A PROJECT REPORT

Submitted by

MADHAN RAJ R	**-**	**513520104021**
NITISH C	**-**	**513520104025**
THILAK A	**-**	**513520104039**
WASEEMULLAH K	**-**	**513520104043**

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

ANNAI MIRA COLLEGE OF ENGINEERING AND TECHNOLOGY

ARAPAKKAM, RANIPET-632517.

ANNA UNIVERSITY :: CHENNAI 600 025

APRIL/MAY 2023

BONAFIDE CERTIFICATE

Certified that this project report **"MEDIA PLAYER CONTROL USING HAND GESTURES DETECTION"** is the bonafide work of **"MADHAN RAJ R (513520104021), NITISH C (513520104025), THILAK A (513520104039), WASEEMULLAH K (513520104043)"** who carried out the project work under my supervision.

SIGNATURE

Mr.S.SRINIVASAN M.TECH,(Ph.D)
HEAD OF THE DEPARTMENT

Dept. of Computer Science and Engg,
Annai Mira College of Engineering and Technology, Arapakkam
Ranipet-632517

SIGNATURE

Mrs.A.SATHYA M.E
SUPERVISOR

Dept. of Computer Science and Engg,
Annai Mira College of Engineering and Technology, Arapakkam
Ranipet-632517

Submitted for the project viva voce held on _____

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

I first offer my deepest gratitude to Almighty God who has given me strength and good health during the course of this project.

I am conscious about my indebtedness our Honorable Chairman **Mr.S.Ramadoss**, Secretary **Mr.G.Thamothiran** and our Principal **Dr.T.K.Gopinathan, Ph.D.**, and our Vice Principal **Dr.D.Saravanan, Ph.D.**, who inspired me reach greater heights in the pursuits of knowledge.

We proudly express our esteemed gratitude to our Head of the Department **Mr.S.SRINIVASAN, M.TECH.,(Ph.D.)** for his encouragement and assistance towards the completion of the project.

Special thanks must be mentioned to our internal guide **Mrs. A. SATHYA, M.E.**, Assistant Professor, Computer science Department for his expert advice, valuable information and guidance throughput the completion of the project. We also express our sincere thanks to our project coordinator, guide and all our staff of CSE Department who helped us in the completion of this project.

We also express our sincere thanks to our project coordinator, guide and all our staff of CSE Department who helped us in the completion of this project.

TABLE OF CONTENT

MEDIA PLAYER CONTROL USING HAND GESTURES DETECTION

ABSTRACT

Hand Gesture Detection for Media Player Control aims to revolutionize the way users interact with media players by harnessing the power of hand gestures. Traditional input devices such as keyboards or remote controllers often pose limitations in terms of user experience and convenience. By leveraging computer vision techniques and machine learning algorithms, this project proposes a novel approach to media player control, enabling users to navigate and control their media playback with intuitive hand movements. Through the use of cameras or depth sensors, the system captures and analyzes hand gestures in real time, translating them into specific commands or actions like play, pause, volume adjustment, or track selection. This technology promises to deliver a more natural and immersive user experience, eliminating the need for physical interfaces and opening up possibilities for hands-free control in various contexts.

LIST OF FIGURES

LIST OF ABBREVIATIONS

OpenCV	-	Open-Source Computer Vision
CNN	-	Convolutional Neural Network
UML	-	Unified Modeling Language

INTRODUCTION

CHAPTER 1

1. INTRODUTION

1.1. PROBLEM STATEMENT

The conventional methods of interacting with media players, such as keyboards or remote controllers, often lack intuitiveness and convenience, hindering the overall user experience. These traditional input devices can be cumbersome, require physical contact, and may not be suitable for certain environments or users with limited mobility. Additionally, they may impose a steep learning curve for individuals unfamiliar with complex control schemes. Therefore, there is a need for a more natural and seamless approach to media player control that overcomes these limitations and enhances user interaction.

This project addresses the challenge of developing a hand gesture detection system for media player control. The aim is to design a solution that recognizes and interprets hand movements in real time, mapping them to specific commands or actions associated with media playback. The system should be able to accurately detect a wide range of hand gestures, including play, pause, volume adjustment, and track selection. It should provide a user-friendly interface that is intuitive, reliable, and responsive, enabling users to control media players effortlessly and without physical contact.

1.2. THE SOLUTION

The proposed solution for enhancing media player control through hand gestures involves the development of a robust and intuitive hand gesture detection system. This system will leverage computer vision techniques and machine learning algorithms to accurately recognize and interpret hand movements in real time. By employing image segmentation and feature extraction algorithms, the system will be able to isolate and track the user's hands, enabling precise gesture recognition.

To achieve accurate gesture classification, the system will utilize machine learning models such as convolutional neural networks (CNNs) or recurrent neural networks (RNNs). These models will be trained on a comprehensive dataset of labeled hand gestures, allowing them to associate specific gestures with media player commands or actions. The training process will enable the system to accurately map gestures to functions such as play, pause, volume adjustment, or track selection, ensuring seamless control of media playback.

In order to provide a seamless user experience, the hand gesture detection system will prioritize real-time interaction. The algorithms and processes involved in gesture recognition will be optimized for efficiency, ensuring minimal delay between gesture recognition and media player response. The system will also feature a user-friendly interface with visual feedback, providing users with a clear indication of recognized gestures and on-screen prompts to guide their interactions. Additionally, the system will be designed to adapt to different user variations, accommodating variations in hand sizes, orientations, and movement speeds, while also accounting for environmental conditions such as lighting variations and background clutter.

1.3. EXISTING SYSTEM

1.3.1. EXISTING CONCEPT

The existing systems for media player control primarily rely on traditional input devices like keyboards, remote controllers, or touchscreens. While these methods have been widely used, they often lack the intuitiveness and natural interaction desired by users. Some media players offer voice control options, but these can be limited in accuracy and may not be suitable for all environments. Motion-based control systems, such as Microsoft Kinect, have been introduced but are not widely adopted due to the need for specialized hardware and limited gesture recognition capabilities.

Some research projects and prototypes have explored hand gesture recognition for media player control. These systems utilize computer vision techniques and machine learning algorithms to detect and interpret hand movements. However, existing solutions often suffer from challenges such as limited gesture recognition accuracy, slower real-time performance, and difficulties in adapting to varying environmental conditions and user variations. Commercial products claiming hand gesture control for media players are available, but they may still face limitations in terms of accuracy, responsiveness, and ease of use.

1.3.2. DRAWBACKS

- Limited gesture recognition accuracy
- Slow real-time performance leads to noticeable delays
- Lack of adaptability

1.4. PROPOSED SYSTEM

1.4.1.PROPOSED CONCEPT

The proposed system aims to develop a robust and intuitive hand gesture detection system for media player control, leveraging computer vision techniques and machine learning algorithms to accurately recognize and interpret hand movements in real time. This system will prioritize high gesture recognition accuracy, ensuring reliable control of media players through hand gestures. It will also emphasize fast real-time performance, minimizing delays and enabling seamless interaction with media players. Additionally, the proposed system will focus on adaptability to varying environmental conditions and user variations, enhancing the system's robustness and usability in different contexts.

1.4.2. ADVANTAGES

- It enables intuitive and immersive control of media players through hand gestures.
- The system ensures high accuracy.
- Its adaptability to varying environmental conditions.

LITERATURE SURVEY

CHAPTER 2

2. LITERATURE SURVEY

2.1.HAND GESTURES DETECTION

Hand gesture detection refers to the process of identifying and interpreting hand movements or gestures using computer vision techniques and machine learning algorithms. It involves capturing and analyzing visual data, typically from a camera or sensor, to recognize specific patterns or poses created by hand movements. The goal of hand gesture detection is to accurately interpret these gestures and map them to corresponding commands or actions in various applications or systems.

The process of hand gesture detection typically involves multiple steps, including hand segmentation, feature extraction, and gesture classification. Hand segmentation involves isolating the hand region from the background or other objects in the scene. Feature extraction involves extracting relevant information or characteristics from the segmented hand region, such as hand shape, position, or motion. Finally, gesture classification involves using machine learning algorithms, such as neural networks, decision trees, or support vector machines, to classify the extracted features into specific gestures.

In hand gesture detection for media player control, commonly used models include convolutional neural networks (CNNs) for gesture recognition. Packages such as OpenCV provide image processing and computer vision functions, while deep learning frameworks like TensorFlow, PyTorch, and Keras are used for training and deploying CNN models. Pre-trained models like MobileNet or VGGNet can be fine-tuned or utilized for transfer learning. Gesture recognition libraries like mediapipe offer pre-built models and functions, and machine learning packages like scikit-learn provide algorithms for gesture classification.

2.2.SOME OTHER RESEARCHERS

Title: Gestures based audio/video players

Authors: Indrajeet Vadgama, Yash Khot, Yash Thaker, Pranali,
Jourasand Yogita Mane

Year: 2022

In this research paper we propose a system to control VLC media player without physical interaction with the computer. Image is captured and verified with our system in which image pre-processing and other techniques are used for detection of gesture. Image is captured and provided as the input to the application via a camera of minimum 1 MP quality for good results. Processing will take place in the system after providing the input as image. Here the input gesture will be recognized on the basis of finger count. The desired action will be performed. Errors may arise due to invalid gestures or quick movement of hand resulting in the system not recognizing the gesture. Invalid gestures will result in no action being performed similarly quick movement of hand will be ignored to avoid accidental gestures. System uses frames from an input video stream, performs morphological filtering upon it, RGB/HSV filtering is performed to detect hand glove, contours and defects are detected and gesture is recognized and compared with number of outstretched fingers, based on the number of outstretched fingers operations are performed in VLC media player. Communication with VLC media player is done via TCP Connection.

Title: A Vision based Hand Gesture Interface for Controlling VLC Media Player

Author: Siddharth Swarup Rautaray, Anupam Agrawal

Year: 2021

Human Computer Interaction can acquire several advantages with the introduction of different natural forms of device free communication. Gestures are a natural form of actions which we often use in our daily life for interaction, therefore to use it as a communication medium with computers generates a new paradigm of interaction with computers. This paper implements computer vision and gesture recognition techniques and develops a vision based low cost input device for controlling the VLC player through gestures. VLC application consists of a central computational module which uses the Principal Component Analysis for gesture images and finds the feature vectors of the gesture and save it into a XML file. The Recognition of the gesture is done by K Nearest Neighbour algorithm. The theoretical analysis of the approach shows how to do recognition in static background. The Training Images are made by cropping the hand gesture from static background by detecting the hand motion using Lucas Kanade Pyramidical Optical Flow algorithm. This hand gesture recognition technique will not only replace the use of mouse to control the VLC player but also provide different gesture vocabulary which will be useful in controlling the application.

Title: A Comparative Analysis on Hand Gesture Recognition using Deep Learning

Author: Preeti Nutipall, Surya Pavan Kumar Gudla, B.Yogith, Gujjala Rajesh

Year: 2021

In recent years, the vision-based innovation of hand motion acknowledgment is a significant piece of human computer interaction (HCI).In the last decades; keyboard and mouse play a significant role in human-computer interaction. However, owing to the rapid development of hardware and software, new types of HCI methods have been required. In particular technologies, such as speech recognition and gesture recognition receive great attention in the field of HCI. Hand gesture recognition is very significant for human-computer interaction. On survey many models were used to recognize hand gestures with different custom images from various datasets captured by camera. One of the approaches that highly used in image feature extraction is Convolution neural network (CNN). In this work, we present a novel real-time method for hand gesture recognition. In our framework, the hand gesture is detected from the trained dataset of images. Then, the palm and fingers are segmented so as to detect and recognize the fingers. Finally, a rule classifier is applied to predict the labels of hand gestures. The experiments on the data set of 2569 images show that our method performs well and is highly efficient. Finally we have shown the comparison among the CNN with Softmax and KNN classifier on the images which leads accurate result.Keywords: HCI, Hand Gestures, Machine Learning, CNN, KNN, Softmax Classifier.

Title: Media player with face detection and hand gesture

Author: Mukesh Vishwakarma, Akshay navratna, Sneha ghorpade, Saket thombre, Trupti kumbhare

Year: 2020

When you are playing a video on your laptops or PC and somebody calls you, you have getaway from your device or simply look away from it due to this you miss some port of video. Then you have to drag back the video to see the video from where you have left. Now to overcome this problem we can build a media player that detects human face and for the time it is detecting a face video will be played else video will pause automatically. The player starts running again as soon as it detect a face. This can beachieved using the camera or webcam on top of the computer. As long as the camera detects a face the video keeps on playing. The player pauses as soon as users face is not completely seen. Here we have also added some extra feature that track hand movement to control mouse cursor and left click operation which is used to control media player functions which include dragging a video forward/backward, increasing and decreasing volume, choosing other videos to play .

Title: Look Based Media Player with Hand Gesture Recognition

Author: Saritha M , Soniya N , Sharanya, Skandashree H M, Veena B

Year: 2020

When you are watching a video and someone calls you to have to look somewhere else or go away from PC for some time, so you miss some part of the video. Later you need to drag back the video from where you saw it. Look Based Media Player with Hand Gesture Recognition, is a media player which pauses itself when the user is not looking at it. The player starts running again as soon as the user looks at it. This is done using the camera module or web camera on top of the computer. In this project, we are developing an advanced media player that uses face detection to automatically play and pause the video, hand gesture recognition system to forward and backward the video. The system monitors whether the user is looking at the screen or not using Haar-cascade Classifier. In case if the user is not looking at the screen or if the system couldn't detect the user's face then it immediately stops the video. Controlling other functions of a media player such as playing next and previous videos is done using Convolutional Neural Network. It also provides the feature of controlling functions of media players such as detection and comparing noise from the environment's input to the machine's output .

SYSTEM SPECIFICATION

CHAPTER 3

3. SYSTEM SPECIFICATION

3.1. HARDWARE REQUIREMENT

- **SYSTEM** : Intel Core i5 or higher
- **HARDDISK** : 50 GB
- **MONITOR** : 15’ LED
- **INPUT DEVICES** : Keyboard, Mouse, Camera
- **RAM** : 8 GB
- **DISPLAY** : Computer moniter
- **CONNECTIVITY** : LAN or Wi-Fi, Camera

3.2. SOFTWARE REQUREMENT

- **SOFTWARE USED** : Python idle, VS code
- **LANGUAGES** : Python
- **MEDIA PLAYER APIS** : VLC Media player API or others API
- **DATA COLLECTION** : Hand gestures dataset
- **ANNOTATION TOOLS** : Web camera

PROJECT IMPLEMENTATION

CHAPTER 4

4. PROJECT IMPLEMENTATION

4.1. MODULES

- Requirement Analysis
- Data Collection
- Preprocessing
- Hand Detection
- Gesture Recognition
- Mapping Gestures to Actions
- Integration with Media player
- User interface
- Test refinement
- Deployment

4.1.1. REQUIREMENT ANALYSIS

Define the specific requirements and goals of the media player control system. Determine the desired gestures, functionalities, and performance expectations.

4.1.2. DATA COLLECTION

Gather a dataset of hand gesture samples that represent the desired commands for controlling the media player. Ensure the dataset includes a variety of lighting conditions, backgrounds, and hand variations.

4.1.3. PREPROCESSING

Apply preprocessing techniques to the input video frames, such as resizing, normalization, or noise reduction, to enhance the quality and facilitate efficient sing.

4.1.4. HAND DETECTION

Utilize computer vision techniques or pre-trained models to detect and localize the hand region in each frame of the video. This step is crucial for isolating the hand and ignoring irrelevant background information.

4.1.5. GESTURE RECOGNIZATION

Apply machine learning or deep learning algorithms to recognize and classify the detected hand gestures. Train a model using the collected dataset, or utilize pre-trained models for gesture recognition tasks.

4.1.6. MAPPING GESTURES TO ACTION

Define a mapping between recognized gestures and media player commands. Assign specific gestures to actions like play, pause, volume control, or seeking within the media content.

4.1.7. INTEGRATION WITH MEDIA PLAYER

Implement the communication between the media player control system and the media player software or API. Ensure the system can send the appropriate commands based on the recognized gestures.

4.1.8. USER INTERFACE

Design and develop a user-friendly graphical interface to display the media content and provide visual feedback to the user about the recognized gestures and corresponding actions.

4.1.9. TESTING AND REFINEMENT

Test the system with different hand gestures and scenarios to evaluate its accuracy, responsiveness, and usability. Iterate on the implementation to improve performance and address any issues or limitations.

4.1.10. DEPLOYMENT

Package the media player control system into a deployable form, considering factors such as compatibility, installation requirements, and user documentati

SYSTEM DESIGN

CHAPTER 5

5. SYSTEM DESIGN

5.1. GENERAL

System design is the phase that bridges the gap between problem domain and the existing system in a manageable way. This phase focuses on the solution domain.

It is the phase where the SRS document is converted into a format that can be implemented and decides how the system will operate. In this phase, the complex activity of system development is divided into several smaller sub-activities, which coordinate with each other to achieve the main objective of system development

Inputs to System Design

System design takes the following inputs -

- Statement of work
- Requirement determination plan
- Current situation analysis
- Proposed system requirements including a conceptual data model and Metadata.

Outputs for System Design

System design gives the following outputs -

- Infrastructure and organizational changes for the proposed system.
- A data schema, often a relational schema. Metadata to define the tables/files and columns/data-items.
- A function hierarchy diagram or web page map that graphically describes the program structure.
- Actual or pseudocode for each module in the program.
- A prototype for the proposed system.

5.2. SYSTEM ARCHITECTURE

A system architecture is a conceptual model that outlines a system's structure, behavior, and additional viewpoints. An architectural description is a formal description and representation of a system that is arranged in a way that allows for reasoning about the system's structures and actions.

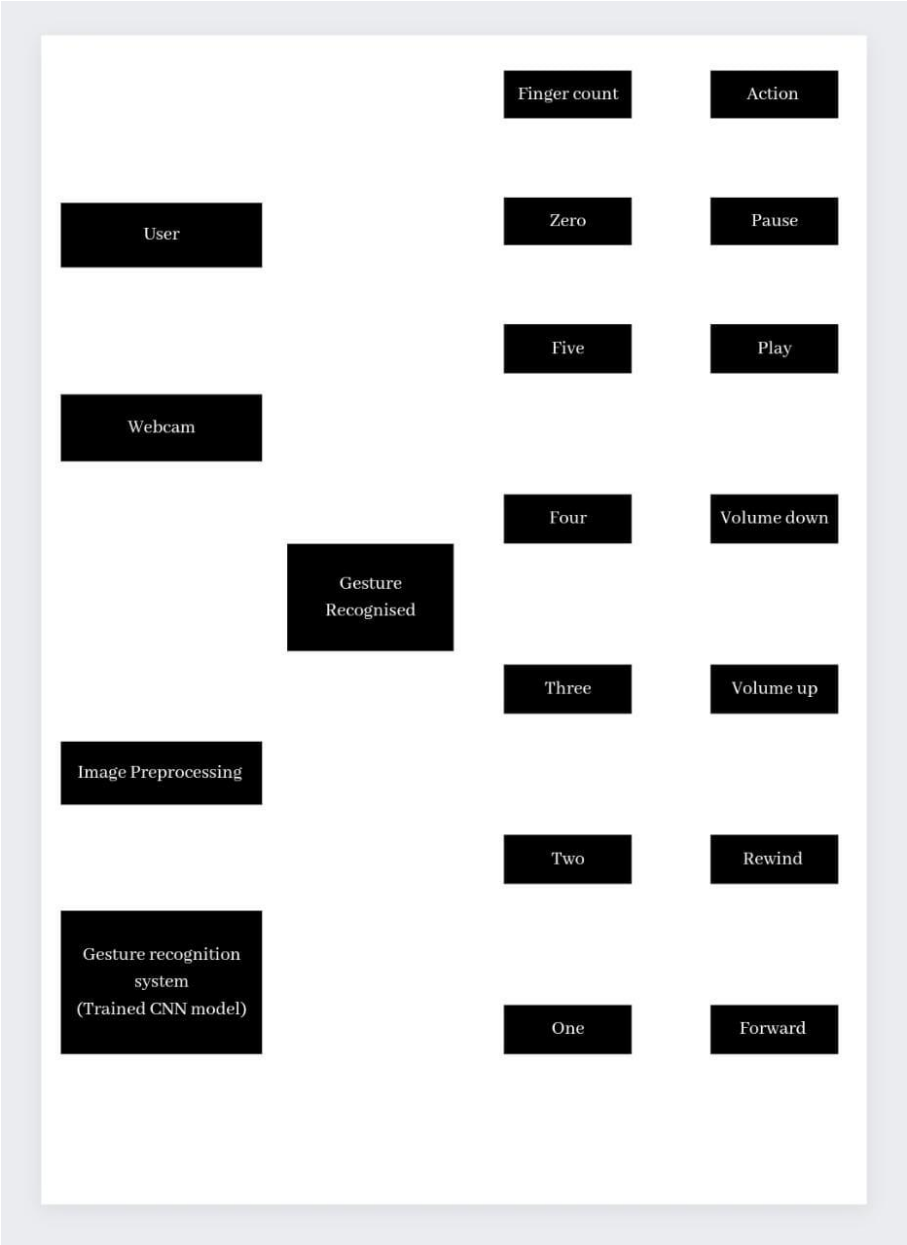

FIG 5.1. SYSTEM ARCHITECTURE

5.3. USECASE DIAGRAM

A use case diagram can summarize the details of your system's users (also known as actors) and their interactions with the system. To build one, you'll use a set of specialized symbols and connectors. A use case diagram is a graphical depiction of a user's possible interactions with a system. A use case diagram shows various use cases and different types of users the system has and will often be accompanied by other types of diagrams

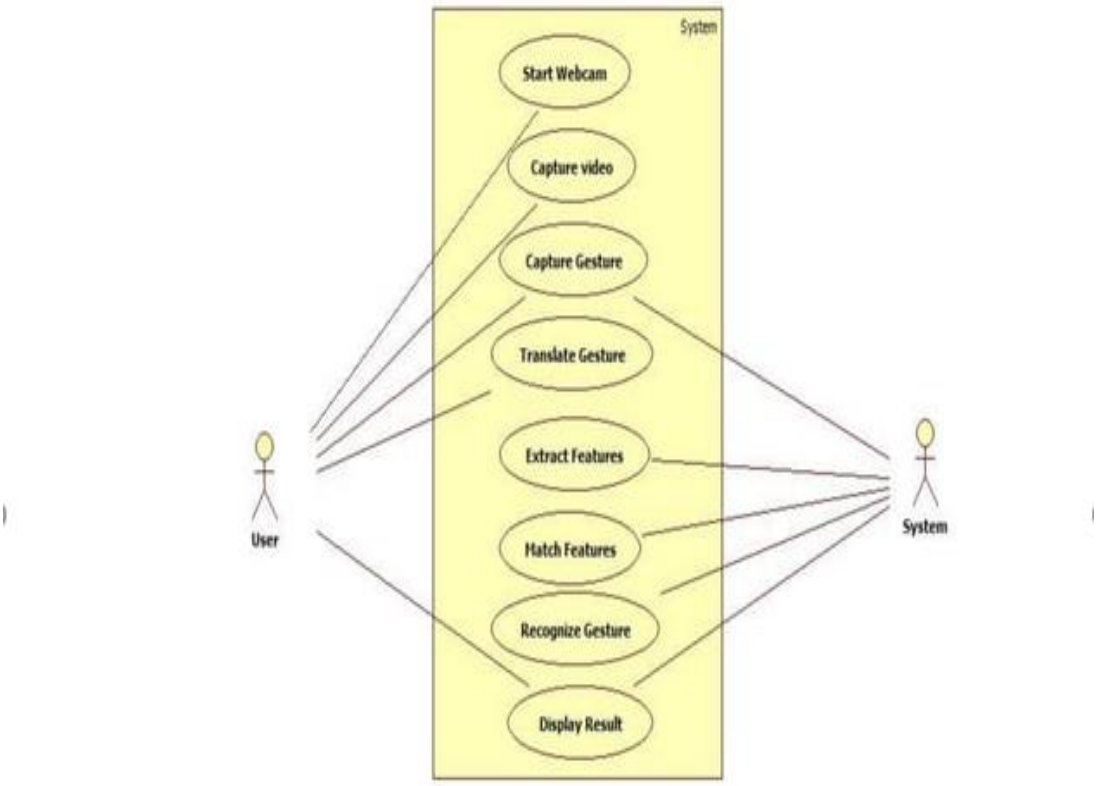

FIG 5.2. USECASE DIAGRAM

5.4. CLASS DIAGRAM

Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application.

Class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modeling of object-oriented systems because they are the only UML diagrams, which can be mapped directly with object-oriented languages.

Class diagram shows a collection of classes, interfaces, associations, collaborations, and constraints. It is also known as a structural diagram.

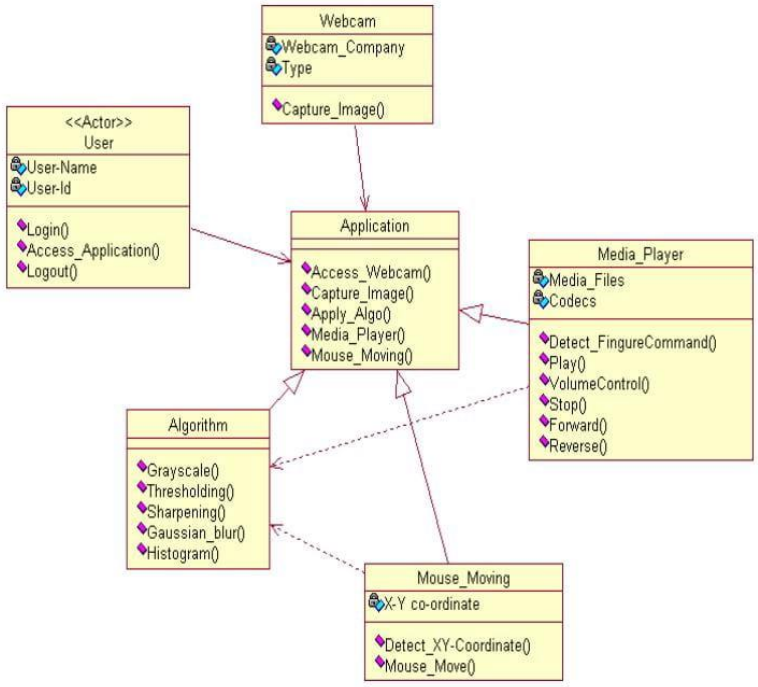

FIG 5.3. CLASS DIAGRAM

5.5. ACTIVITY DIAGRAM

An activity diagram is essentially a flowchart that shows how one activity leads to another. The action can be referred to as a system operation. One operation leads tothe next in the control flow. This flow may be parallel, concurrent, or branched.

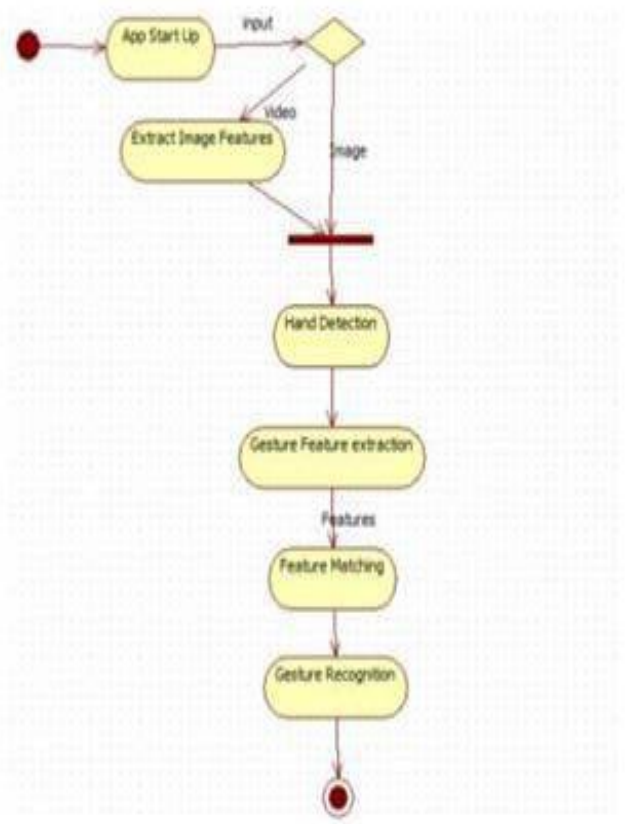

FIG 5.4. ACTIVITY DIAGRAM

5.6. SEQUENCE DIAGRAM

A sequence diagram or system sequence diagram (SSD) shows process interactions arranged in time sequence in the field of software engineering. It depicts the processes involved and the sequence of messages exchanged between the processes needed to carry out the functionality.

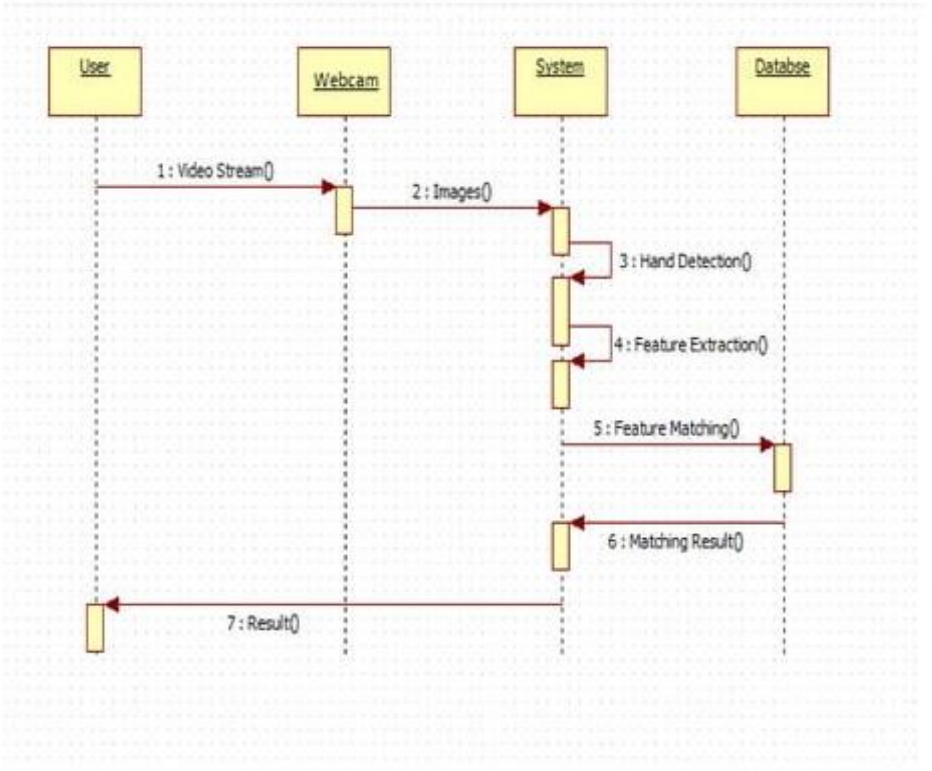

FIG 5.5. SEQUENCE DIAGRAM

SOFTWARE SPECIFICATION

CHAPTER 6

6. SOFTWARE SPECIFICATION

6.1. GENERAL

Software Specification is a list of Software, Packages, libraries required to develop a project. Software Specification contains the deep description about the project.

6.2. PYTHON

6.2.1 INRODUCTION TO PYTHON

Web development, scientific computing, data analysis, artificial intelligence, and machine learning are just a few of the tasks that Python is frequently used for. Python is a high-level, general-purpose programming language. Python, which was developed by Guido van Rossum in the late 1980s, is now among the most widely used programming languages worldwide because of its readability, flexibility, and simplicity. We'll look at Python's main attributes and use cases in this introduction.

Python's syntax is created with readability of the code in mind, making it simple to read and write. Unlike languages that use curly braces and semicolons, Python uses whitespace and indentation to structure code, making it moreaesthetically pleasing and simpler to understand. Python is therefore perfect for both seasoned programmers who want to write clear, concise, and maintainable code as well as for newbies who are just starting to learn how to program.

6.2.1. FEATURES OF PYTHON

Python is a versatile and powerful programming language with a variety of features that make it popular among developers. Here are some of the key features of Python:

- **Easy to Learn**: Python is a beginner-friendly language that is easy to learn and understand. Its simple syntax and emphasis on readability make it an ideal language for new programmers.
- **Interpreted**: Python is an interpreted language, which means that code is executed line by line at runtime. This makes it easier to debug and test code, as changes can be made and tested immediately without the need to recompile the entire program.
- **Object-Oriented**: Python is an object-oriented language, which means that it supports the creation of classes and objects, encapsulation, inheritance, and polymorphism.
- **High-level Language**: Python is a high-level language, which means that it abstracts away many low-level details, such as memory management and system calls. This makes it easier to write code and focus on solving problems rather than worrying about low-level details.
- **Cross-platform**: Python is a cross-platform language, which means that it can run on different operating systems, such as Windows, Linux, and macOS.
- **Large Standard Library**: Python has a vast standard library that contains many useful modules and functions for performing a wide range of tasks, such as file I/O, networking, and regular expressions.

  - **Third-Party Libraries:** Python has a rich ecosystem of third-party libraries and frameworks that extend the language's functionality, such as NumPy, Pandas, Django, Flask, and TensorFlow..

6.2.2. OBJECTIVES OF PYTHON

Python is a robust and functional programming language that serves a wide range of functions. Its main goals are to be productive, adaptable, and simple to learn.Here are a few sentences that go into greater detail about Python's goals:

Python's main goal is to be simple to use and learn. It's a great option for beginning programmers thanks to its clear, simple syntax that prioritizes readability. Python's design ethos, which places a high value on simplicity and minimalism, makes the language simple to learn and retain. Additionally, Python has a sizable standard library that is filled with numerous built-in functions and modules that facilitate common programming tasks.

Being productive and flexible is another goal of Python. The Python design philosophy emphasizes the value of modularity, extensibility, and reuse of code. Because of this, writing code that is reusable and simple to maintain is simple. Python's dynamic typing system eliminates the need for type declarations, which speeds up code writing compared to languages with static typing. Additionally, Python has a large community of third-party libraries and frameworks that make it simple to expand the language's functionality for particular tasks like data analysis, web development, and machine learning. Python is a highly productive language that is suitable for a variety of programming tasks thanks to all of these features.

6.2.3. PYTHON FRAMEWORKS

Python is a strong and adaptable programming language that is used in avariety of applications, including machine learning, data analysis, scientific computing, and web development. Developers have developed a variety offrameworks that offer pre-built code and tools for specific programming tasks tomake programming with Python simpler and more effective. Here are a few sentences discussing Python frameworks:

Python frameworks are collections of pre-written tools and code that make it simpler to create sophisticated applications. By providing pre-built code for common tasks, they can save time and effort for developers by giving them a structure and set of guidelines for creating applications. Django, Flask, and Pyramid are just a few of the well-liked web development frameworks available in Python. These frameworks offer a variety of tools and features, including database management, routing, and templating, which make it simpler to create dependable and scalable web applications.

Python frameworks are also frequently used in scientific computing and data analysis. Numerous tools and functions are available for working with large datasets, conducting statistical analysis, and visualizing data in Python frameworks like NumPy, SciPy, and Pandas. These frameworks simplify complex calculations and analysis for researchers and scientists, and they can be applied to a variety of disciplines, including biology, engineering, physics, and finance. Artificial intelligence and machine learning can both benefit from the use of Python frameworks. Machine learning models can be built and trained using frameworks like TensorFlow and PyTorch, which are popular in both academia and business.

6.2.4. PACKAGES OF PYTHON

TensorFlow: Developed by Google, TensorFlow is a powerful open-source library for deep learning. It offers a comprehensive ecosystem for building and training neural networks, supporting both low-level operations and high-level abstractions. TensorFlow provides flexibility, scalability, and compatibility across a range of platforms and devices.

Keras: Keras is a user-friendly and high-level deep learning library that runs on top of TensorFlow, Theano, or Microsoft Cognitive Toolkit (CNTK). It simplifies the process of building deep neural networks by providing a simple and intuitive API for model creation and training..

Theano: Theano is a library that enables efficient mathematical computations, especially for deep learning tasks. It provides a low-level interface for defining and optimizing mathematical expressions, making it useful for implementing custom deep learning models.

MXNet: MXNet is a flexible and efficient deep learning framework that supports both imperative and symbolic programming. It offers a high-level interface for constructing neural networks and provides scalability across multiple GPUs and distributed system.

PyAutoGUI: PyAutoGUI is a Python package that enables GUI (Graphical User Interface) automation by providing functions to control the mouse, keyboard, and perform screen-related operations. It allows you to automate repetitive tasks, simulate user interactions, and create automated scripts or applications that interact with graphical elements on the screen.

NumPy: A key Python package for scientific computing is called NumPy. It offers a potent N-dimensional array object that is useful for effectively manipulating sizable numerical data arrays. For performing mathematical operations on arrays, NumPy offers a number of functions, including those for linear algebra, Fourier analysis, and random number generation. For anyone working with numerical data in Python,NumPy is a crucial tool. It is widely used in scientific computing, data analysis, and machine learning.

Opencv: A well-known computer vision library called OpenCV (Open-Source Computer Vision Library) offers a number of tools and algorithms for image and video analysis. Numerous tasks, such as object detection, face recognition, motion analysis, and others, can be carried out using OpenCV. It provides a variety of functions for working with images and videos, including image processing, filtering, and transformation.

Mediapipe: The MediaPipe package in Python is an open-source framework developed by Google that provides a high-level API for accessing and utilizing the functionality of the MediaPipe framework. It offers pre-built models, tools, and a modular architecture for building real-time video and audio processing applications, including tasks like face detection, hand tracking, pose estimation, and more

SOFTWARE TESTING

CHAPTER 7

7. SOFTWARE TESTING

7.1. GENERAL

Software Testing is a method to check whether the actual software product matches expected requirements and to ensure that software product is Defect free. It involves execution of software/system components using manual or automated tools to evaluate one or more properties of interest. The purpose of software testing. is to identify errors, gaps or missing requirements in contrast to actual requirements.

Some prefer saying Software testing definition as a White Box and Black Box Testing. In simple terms, Software Testing means the Verification of Application Under Test. This Software Testing course introduces testing software to the audience and justifies the importance of software testing.

Importance of Software Testing

Software Testing is Important because if there are any bugs or errors in the software, it can be identified early and can be solved before delivery of the software product. Properly tested software product ensures reliability, security and high performance which further results in time saving, cost effectiveness and customer satisfaction.

Need of Testing

Testing is important because software bugs could be expensive or even dangerous. Software bugs can potentially cause monetary and human loss, and history is full of such examples.

Here are the benefits of using software testing:

1. **Cost-Effective**: It is one of the important advantages of software testing.

Testing any IT project on time helps you to save your money for the long term. In case if the bugs caught in the earlier stage of software testing, it costsless to fix.

2. **Security**: It is the most vulnerable and sensitive benefit of software testing. People are looking for trusted products. It helps in removing risks and problems earlier.

7.2. TYPES OF TESTING

7.2.1.UNIT TESTING

A unit test is a way of testing a unit - the smallest piece of code that can be logically isolated in a system. In most programming languages, that is a function, a subroutine, a method or property. The isolated part of the definition is important. In his book "Working Effectively with Legacy Code", author Michael Feathers states that such tests are not unit tests when they rely on external systems: "If it talks to the database, it talks across the network, it touches the file system, it requires system configuration, or it can't be run at the same time as any other test."

Modern versions of unit testing can be found in frameworks like JUnit, or testing tools like Test Complete. Look a little further and you will find a Unit, the mother of all unit testing frameworks created by Kent Beck, and a reference in chapter 5 of The Art of Software Testing. Before that, it's mostly a mystery. I asked Jerry Weinberg about his experiences with unit testing -- "We did unit testing in 1956.As far as I knew, it was always done, as long as there were computers".

Regardless of when and where unit testing began, one thing is for sure. Unit testing is here to stay. Let's look at some more practical aspects of unit testing.

7.2.2.FUNCTIONAL TESTING

Functional testing is a sort of black-box testing that relies its test cases on the requirements of the software component being tested. Internal program structure is rarely considered when testing functions by feeding them input and reviewing the result. Functional Testing is a type of software testing that validates the software system against the functional requirements/specifications. The purpose of Functional tests is to test each function of the software application, by providing appropriate input, verifying the output against the Functional requirements.

Functional testing mainly involves black box testing and it is not concerned about the source code of the application. This testing checks User Interface, APIs, Database, Security, Client/Server communication and other functionality of the Application Under Test. The testing can be done either manually or using automation.

7.2.3.SYSTEM TESTING

System testing is a sort of software testing that is conducted on a wholeintegrated system to assess the system's compliance with the associated requirements. Passed integration testing components are used as input in system testing. The purpose of integration testing is to discover any discrepancies between the components that are being integrated together. System testing finds flaws in both integrated modules and the entire system. The observed behavior of a component or system when tested is the result of system testing. System testing is performed on the entire system in the context of either system requirement specifications, functional requirement specifications, or both. System testing examines the system's design and behavior, as well as the customer's expectations.

7.2.4. PERFORMANCE TESTING

Performance testing is a type of testing that assesses the speed, responsiveness, and stability of a computer, network, software program, or device when subjected to a workload. Organizations will conduct performance testing to discover performance bottlenecks.

Performance testing can involve quantitative tests done in a lab, or in some scenarios, occur in the production environment. Performance requirements should be identified and tested. Typical parameters include processing speed, data transfer rates, network bandwidth and throughput, workload efficiency and reliability. As an example, an organization can measure the response time of a program when a user requests an action; the same can be done at scale.

7.2.5. INTEGRATION TESTING

Integration testing is the second level of the software testing process comes after unit testing. In this testing, units or individual components of the software are rested in a group. The focus of the integration testing level is to expose defects at the time of interaction between integrated components or units.

Unit testing uses modules for testing purpose, and these modules are combined and tested in integration testing. The Software is developed with a number of software modules that are coded by different coders or programs. The goal of integration testing is to check the correctness of communication among all the modules.

Integration testing Once all the components or modules are working independently, then we need to check the data flow between the dependent modules is known as integration testing.

7.2.6.ACCEPTANCE TESTING

Acceptance testing is formal testing based on user requirements and function processing. It determines whether the software is conforming specified requirements and user requirements or not. It is conducted as a kind of Black Box testing where the number of required users involved testing the acceptance level of the system. It is the fourth and last level of software testing.

User acceptance testing is a type of testing, which is done by the customer before accepting the final product. Generally, it is done by the customer (domain expert) for their satisfaction, and check whether the application is working according to given business scenarios, real-time scenarios.

In this, we concentrate only on those features and scenarios which are regularly used by the customer or mostly user scenarios for the business or those scenarios which are used daily by the end-user or the customer. However, the software has passed through three testing levels (Unit Testing, Integration Testing, System Testing)But still there are some minor errors which can be identified when the system is used by the end user in the actual scenario.

CODING

CHAPTER 8

8. CODING

8.1. GATHER_IMAGE.PY

```
import cv2
import os
import numpy as np
print("Enter the label name:- right, left, up, down,space")
label_name = input("Enter the label name:-")
num_samples = int(input("Enter the number of samples:- "))
IMG_SAVE_PATH = 'train_images'
IMG_CLASS_PATH = os.path.join(IMG_SAVE_PATH, label_name)
def preprocess_image(img):
    blur = cv2.GaussianBlur(img,(3,3),0)
    hsv = cv2.cvtColor(blur, cv2.COLOR_BGR2HSV)
    mask = cv2.inRange(hsv,np.array([2,0,0]),np.array([20,255,255]))
    kernel = np.ones((5,5))
    # Apply morphological transformations to filter out the background noise
    dilation = cv2.dilate(mask, kernel, iterations=1)
    # erosion = cv2.erode(dilation, kernel, iterations=1)
    # Apply Gaussian Blur and Threshold
    # filtered = cv2.GaussianBlur(mask, (3, 3), 0)
    ret, thresh = cv2.threshold(dilation, 70, 255, cv2.THRESH_BINARY)
```

```
    return thresh
def p2(img):
    kernel = np.ones((5,5),np.uint8)
    frame2 = cv2.cvtColor(img, cv2.COLOR_BGR2HLS)
    lb = np.array([0,77,0])
    ub = np.array([255,255,255])
    mask = cv2.inRange(frame2, lb, ub)
    opening = cv2.morphologyEx(mask, cv2.MORPH_OPEN, kernel)
    img = cv2.cvtColor(opening,cv2.COLOR_BGR2RGB)
    return img
try:
    os.mkdir(IMG_SAVE_PATH)
except FileExistsError:
    pass
try:
    os.mkdir(IMG_CLASS_PATH)
except FileExistsError:
    print("{} directory already exists.".format(IMG_CLASS_PATH))
    print("All images gathered will be saved along with existing items in this folder")
cap = cv2.VideoCapture(0)
 start = False
count = 0
```

```
while True:
    ret, frame = cap.read()
    if not ret:
        continue
    if count == num_samples:
        break
    frame = cv2.flip(frame,1)
    cv2.rectangle(frame, (300, 50), (600, 350), (255, 255, 255), 2)
    # thresh = preprocess_image(frame)
    if start:
        roi = frame[50:350, 300:600]
        save_path = os.path.join(IMG_CLASS_PATH, '{}.jpg'.format(count + 1))
        cv2.imwrite(save_path, roi)
        count += 1
    font = cv2.FONT_HERSHEY_SIMPLEX
    cv2.putText(frame, "Collecting {}".format(count),
            (5, 50), font, 0.7, (0, 255, 255), 2, cv2.LINE_AA)
    # p2img = p2(frame)
    # cv2.imshow("p2",p2img)
    # cv2.imshow("thresh",thresh)
    cv2.imshow("Collecting images", frame)
    k = cv2.waitKey(10)
    if k == ord('a'):
```

```
            start = not start
        if k == ord('q'):
            break
    print("\n{} image(s) saved to {}".format(count, IMG_CLASS_PATH))
    cap.release()
    cv2.destroyAllWindows()
```

8.2. TRAIN.PY

```
import cv2
import numpy as np
from squeezenet import SqueezeNet
from keras.optimizers import Adam
from keras.utils import np_utils
from keras.layers import Activation, Dropout, Convolution2D, GlobalAveragePooling2D
from keras.models import Sequential
import tensorflow as tf
import os
IMG_SAVE_PATH = 'train_images'
CLASS_MAP = {
    "down": 0,
    "left": 1,
    "right": 2,
    "up": 3,
    "none":4,
}
```

```
NUM_CLASSES = len(CLASS_MAP)
def mapper(val):
    return CLASS_MAP[val]
def get_model():
    model = Sequential([
        SqueezeNet(input_shape=(227, 227, 3), include_top=False),
 Dropout(0.5),
        Convolution2D(NUM_CLASSES, (1, 1), padding='valid'),
        Activation('relu'),
        GlobalAveragePooling2D(),
        Activation('softmax')
    ])
    return model
# load images from the directory
dataset = []
for directory in os.listdir(IMG_SAVE_PATH):
    path = os.path.join(IMG_SAVE_PATH, directory)
    if not os.path.isdir(path):
        continue
    for item in os.listdir(path):
        # to make sure no hidden files get in our way
        if item.startswith("."):
            continue
        img = cv2.imread(os.path.join(path, item))
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        img = cv2.resize(img, (227, 227))
        dataset.append([img, directory])
```

```
'''
dataset = [
    [[...], 'up'],
    [[...], 'down'],
    ...
]
'''
data, labels = zip(*dataset)
labels = list(map(mapper, labels))
'''
one hot encoded: [1,0,0], [0,1,0], [0,1,0], [0,0,1], [1,0,0]...
'''
# one hot encode the labels
labels = np_utils.to_categorical(labels)
# define the model
model = get_model()
model.compile(
    optimizer=Adam(lr=0.0001),
    loss='categorical_crossentropy',
    metrics=['accuracy']
)
# start training
model.fit(np.array(data), np.array(labels), epochs=3)
# save the model for later use
model.save("hand_gesture.h5")
```

8.3. SQUEEZENET.PY

```
# import keras
from keras_applications.imagenet_utils import _obtain_input_shape
from keras import backend as K
from keras.layers import Input, Convolution2D, MaxPooling2D, Activation, concatenate,
Dropout
from keras.layers import GlobalAveragePooling2D, GlobalMaxPooling2D
from keras.models import Model
from keras.engine.topology import get_source_inputs
from keras.utils import get_file
from keras.utils import layer_utils
sq1x1 = "squeeze1x1"
exp1x1 = "expand1x1"
exp3x3 = "expand3x3"
relu = "relu_"
WEIGHTS_PATH = "https://github.com/rcmalli/keras-
squeezenet/releases/download/v1.0/squeezenet_weights_tf_dim_ordering_tf_kernels.h5"
WEIGHTS_PATH_NO_TOP = "https://github.com/rcmalli/keras-
squeezenet/releases/download/v1.0/squeezenet_weights_tf_dim_ordering_tf_kernels_not
op.h5"
# Modular function for Fire Node
def fire_module(x, fire_id, squeeze=16, expand=64):
    s_id = 'fire' + str(fire_id) + '/'
    if K.image_data_format() == 'channels_first':
        channel_axis = 1
    else:
```

```
    channel_axis = 3
  x = Convolution2D(squeeze, (1, 1), padding='valid', name=s_id + sq1x1)(x)
  x = Activation('relu', name=s_id + relu + sq1x1)(x)
  left = Convolution2D(expand, (1, 1), padding='valid', name=s_id + exp1x1)(x)
  left = Activation('relu', name=s_id + relu + exp1x1)(left)
  right = Convolution2D(expand, (3, 3), padding='same', name=s_id + exp3x3)(x)
  right = Activation('relu', name=s_id + relu + exp3x3)(right)
  x = concatenate([left, right], axis=channel_axis, name=s_id + 'concat')
  return x
# Original SqueezeNet from paper.
def SqueezeNet(include_top=True, weights='imagenet',
               input_tensor=None, input_shape=None,
               pooling=None,
               classes=1000):
  """Instantiates the SqueezeNet architecture.
  """
  if weights not in {'imagenet', None}:
    raise ValueError('The `weights` argument should be either '
                     '`None` (random initialization) or `imagenet` '
                     '(pre-training on ImageNet).')
  if weights == 'imagenet' and classes != 1000:
    raise ValueError('If using `weights` as imagenet with `include_top`'
                     ' as true, `classes` should be 1000')
  input_shape = _obtain_input_shape(input_shape,
                                    default_size=227,
                                    min_size=48,
                                    data_format=K.image_data_format(),
```

```
                                require_flatten=include_top)
    if input_tensor is None:
        img_input = Input(shape=input_shape)
    else:
        if not K.is_keras_tensor(input_tensor):
            img_input = Input(tensor=input_tensor, shape=input_shape)
        else:
            img_input = input_tensor
    x = Convolution2D(64, (3, 3), strides=(2, 2), padding='valid',
name='conv1')(img_input)
    x = Activation('relu', name='relu_conv1')(x)
    x = MaxPooling2D(pool_size=(3, 3), strides=(2, 2), name='pool1')(x)
    x = fire_module(x, fire_id=2, squeeze=16, expand=64)
    x = fire_module(x, fire_id=3, squeeze=16, expand=64)
    x = MaxPooling2D(pool_size=(3, 3), strides=(2, 2), name='pool3')(x)
    x = fire_module(x, fire_id=4, squeeze=32, expand=128)
    x = fire_module(x, fire_id=5, squeeze=32, expand=128)
    x = MaxPooling2D(pool_size=(3, 3), strides=(2, 2), name='pool5')(x)
    x = fire_module(x, fire_id=6, squeeze=48, expand=192)
    x = fire_module(x, fire_id=7, squeeze=48, expand=192)
    x = fire_module(x, fire_id=8, squeeze=64, expand=256)
    x = fire_module(x, fire_id=9, squeeze=64, expand=256)
    if include_top:
        # It's not obvious where to cut the network...
        # Could do the 8th or 9th layer... some work recommends cutting earlier layers.
        x = Dropout(0.5, name='drop9')(x)
```

```
    x = Convolution2D(classes, (1, 1), padding='valid', name='conv10')(x)
    x = Activation('relu', name='relu_conv10')(x)
    x = GlobalAveragePooling2D()(x)
    x = Activation('softmax', name='loss')(x)
else:
    if pooling == 'avg':
        x = GlobalAveragePooling2D()(x)
    elif pooling=='max':
        x = GlobalMaxPooling2D()(x)
    elif pooling==None:
        pass
    else:
        raise ValueError("Unknown argument for 'pooling'=" + pooling)
# Ensure that the model takes into account
# any potential predecessors of `input_tensor`.
if input_tensor is not None:
    inputs = get_source_inputs(input_tensor)
else:
    inputs = img_input
model = Model(inputs, x, name='squeezenet')
# load weights
if weights == 'imagenet':
    if include_top:
        weights_path = get_file('squeezenet_weights_tf_dim_ordering_tf_kernels.h5',
                                WEIGHTS_PATH,
                                cache_subdir='models')
    else:
```

```
            weights_path =
get_file('squeezenet_weights_tf_dim_ordering_tf_kernels_notop.h5',
                                     WEIGHTS_PATH_NO_TOP,
                                     cache_subdir='models')
        model.load_weights(weights_path)
        if K.backend() == 'theano':
            layer_utils.convert_all_kernels_in_model(model)
        if K.image_data_format() == 'channels_first':
            if K.backend() == 'tensorflow':
                # warnings.warn('You are using the TensorFlow backend, yet you '
                #               'are using the Theano '
                #               'image data format convention '
                #               '(`image_data_format="channels_first"`). '
                #               'For best performance, set '
                #               '`image_data_format="channels_last"` in '
                #               'your Keras config '
                #               'at ~/.keras/keras.json.')
                pass
    return model
```

8.4. PLAY.PY

```
from keras.models import load_model
import cv2
import numpy as np
import pyautogui
REV_CLASS_MAP = {
    0:"down",
    1:"left",
    2:"right",
    3:"up",
    4:"none"
    }
def mapper(val):
    return REV_CLASS_MAP[val]
def play(gesture):
    if gesture == "left":
        pyautogui.press('left')
        # pyautogui.hotkey('ctrl','left')  # for vlc player
    elif gesture == "up":
        # pyautogui.press('up')
        pyautogui.press('0')
        # pyautogui.press('volumeup')
        # pyautogui.hotkey('fn','f3')
    elif gesture == "right":
        pyautogui.press('right')
        # pyautogui.hotkey('ctrl','right')
```

```
    elif gesture == "down":
        # pyautogui.press('down')
        pyautogui.press('9')
        # pyautogui.press('volumedown')
kernel = np.ones((5,5),np.uint8)
model = load_model("hand_gesture.h5")
cap = cv2.VideoCapture(0)
prev_move = None
FONT = cv2.FONT_HERSHEY_SIMPLEX
while True:
    ret, frame = cap.read()
    if not ret:
        continue
    frame = cv2.flip(frame,1)
    cv2.rectangle(frame, (300, 50), (600, 350), (255, 255, 255), 2)
    # extract the region of image within the user rectangle
    roi = frame[50:350, 300:600]
    img = cv2.cvtColor(roi, cv2.COLOR_BGR2RGB)
    img = cv2.resize(img, (227, 227))
    # predict the move made
    pred = model.predict(np.array([img]))
    gesture_code = np.argmax(pred[0])
    user_gesture = mapper(gesture_code)
    # play(user_gesture)
    if user_gesture == "left":
        cv2.putText(frame, "Reverse..",(5, 50), FONT, 0.7, (0, 255, 255), 2, cv2.LINE_AA)
```

```
        pyautogui.press('left')
    elif user_gesture == "up":
        cv2.putText(frame, "Volume Up..",(5, 50), FONT, 0.7, (0, 255, 255), 2,
cv2.LINE_AA)
        pyautogui.press('0')
    elif user_gesture == "right":
        cv2.putText(frame, "Forward..",(5, 50), FONT, 0.7, (0, 255, 255), 2, cv2.LINE_AA)
        pyautogui.press('right')
    elif user_gesture == "down":
        cv2.putText(frame, "Volume Down..",(5, 50), FONT, 0.7, (0, 255, 255), 2,
cv2.LINE_AA)
        pyautogui.press('9')
    cv2.imshow("frame", frame)
    k = cv2.waitKey(10)
    if k == ord('q'):
        break
cap.release()
cv2.destroyAllWindows()
```

SCREENSHOT

CHAPTER 9

9. SCREENSHOT

9.1. OUTPUT

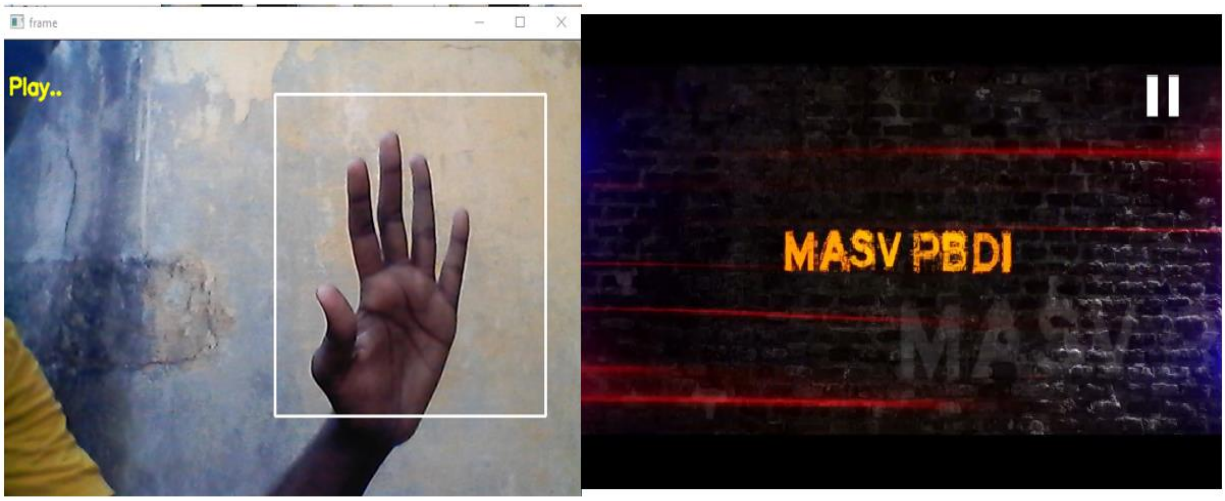

FIG 9.1 OUTPUT-1(PLAY)

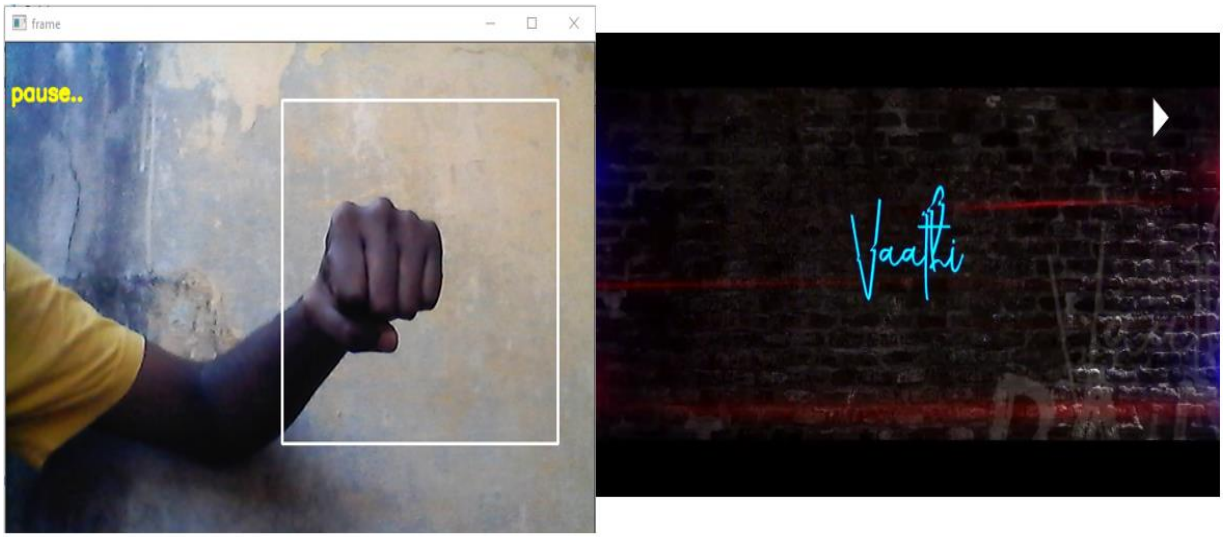

FIG **9.2 OUTPUT-2(PAUSE)**

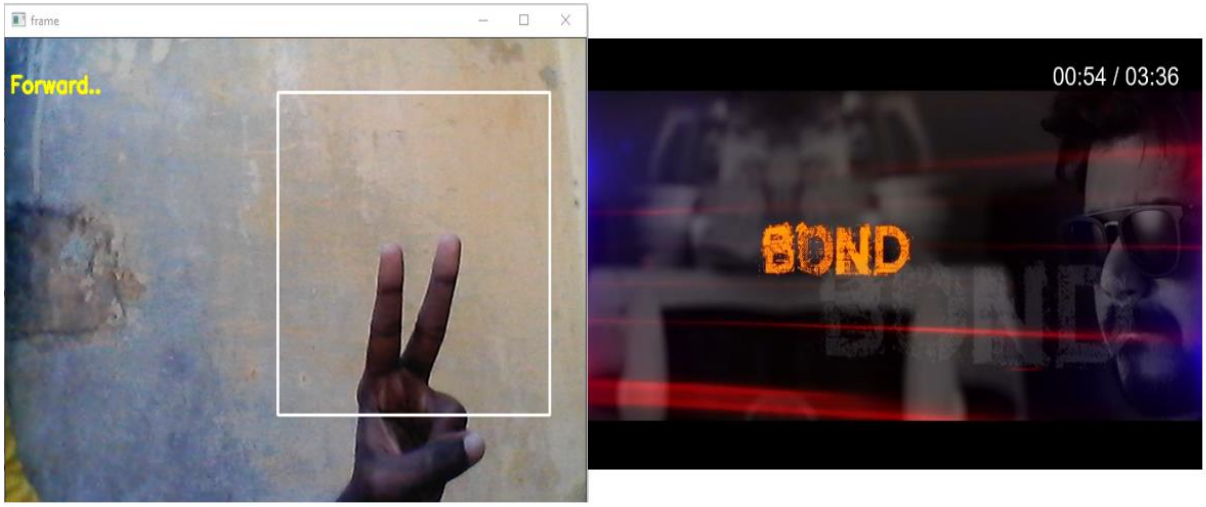

FIG 9.3 OUTPUT-3(FORWARD)

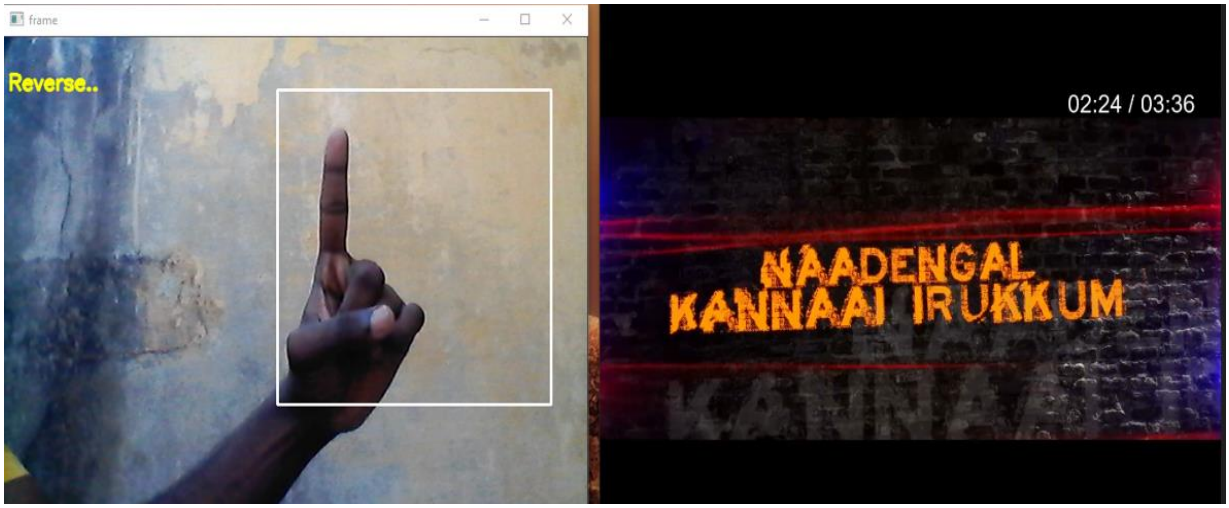

FIG 9.4 OUTPUT-4(BACKWARD)

FIG 9.5 OUTPUT-5(VOLUME UP)

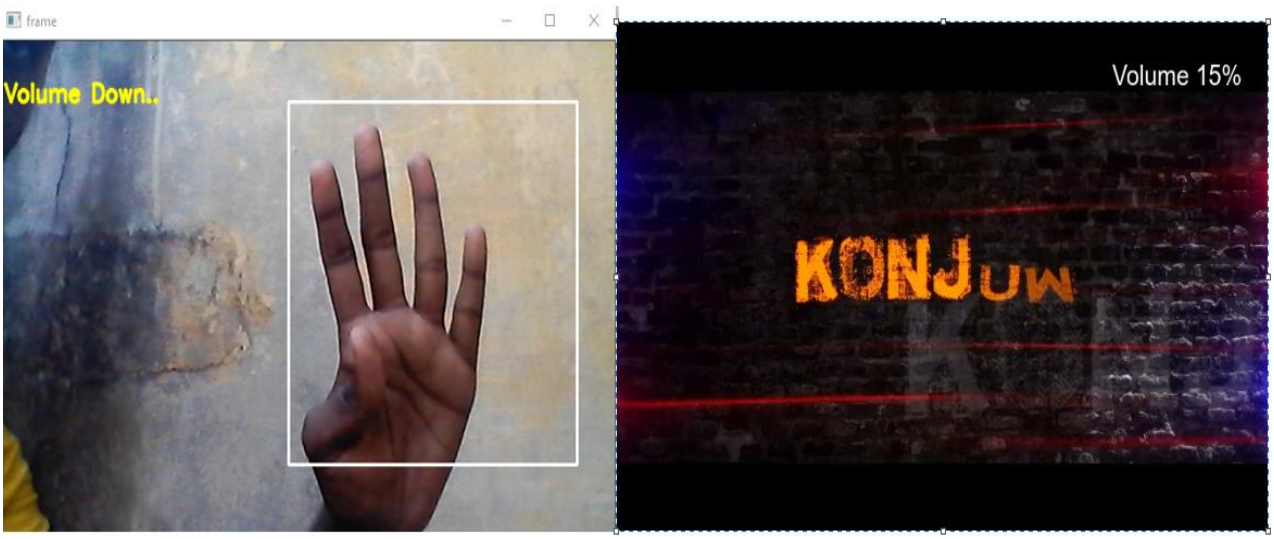

FIG 9.6 OUTPUT-6(VOLUME DOWN)

CONCLUSION

CHAPTER 10

10. CONCLUSION

Media player control using hand gesture detection offers a futuristic and user-friendly approach to interact with media players. By leveraging computer vision and machine learning techniques, users can seamlessly control various functions of media playback through intuitive hand gestures. This technology provides a hands-free and immersive experience, eliminating the need for physical contact or traditional input devices.

The proposed system overcomes the limitations of existing methods by offering improved accuracy in gesture recognition, real-time performance, and adaptability to different environmental conditions and user variations. By utilizing deep learning models and advanced algorithms, the system can accurately detect and interpret hand gestures, ensuring reliable and precise control of media players.

To implement this system, necessary software requirements include programming languages such as Python, computer vision libraries like OpenCV, mediapipe, pyautogui

In conclusion, media player control using hand gesture detection presents a promising and exciting avenue for enhancing user interaction with media players. By leveraging advanced technologies, this approach enables users to intuitively control media playback with ease, offering a more immersive and engaging experience..

FUTURE ENHANCEMENT

CHAPTER 11

11. FUTURE ENHANCEMENT

One potential future enhancement for a media player control using hand gesture recognition project is to incorporate real-time feedback. Currently, the system may recognize hand gestures and translate them into corresponding commands for the media player. However, adding visual feedback to the user can significantly improve the user experience.

Real-time feedback could involve displaying graphical elements on the screen that mimic the detected hand gestures. Additionally, incorporating real-time feedback can enable the system to provide interactive elements that respond to the user's gestures. For instance, the media player interface could display buttons or sliders that can be manipulated directly through hand gestures. This interactive feedback can create a more immersive and intuitive control experience for the user.

By implementing real-time feedback, the media player control system can enhance user engagement, improve gesture recognition accuracy, and provide a more intuitive and interactive interface for controlling media playback.

For example, as the user raises their hand, a virtual representation of the hand could be displayed on the screen, allowing the user to see how their gestures are being interpreted. This visual feedback can help users understand if their gestures are being recognized accurately and adjust their hand movements accordingly.

REFERENCES

CHAPTER 12

12. REFERENCES

[1] A Vision based Hand Gesture Interface for Controlling VLC Media Player, Siddharth Swarup Rautaray, Anupam Agrawal,(2022).

[2] Gestures based audio/video players,Indrajeet Vadgama, Yash Khot, Yash Thaker, Pranali, Jourasand Yogita Mane, 2021.

[3] A Comparative Analysis on Hand Gesture Recognition using Deep Learning, Preeti Nutipall, Surya Pavan Kumar Gudla, B.Yogith, Gujjala Rajesh, 2021.

[4] Media player with face detection and hand gestures, Mukesh Vishwakarma, Akshay navratna, Sneha ghorpade, Saket thombre, Trupti kumbhare, 2021.

[5] Look Based Media Player with Hand Gesture Recognition, Saritha M , Soniya N , Sharanya, Skandashree H M, Veena B, 2021.

[6] N. Krishna Chaitanya and R. Janardhan Rao, "Controlling of windows media player using hand recognition system", 2020.

[7] Swapnil D. Badgujar, Gourab Talukdar, Omkar Gondhalekar and Mrs. S.Y. Kulkarni, "Hand Gesture Recognition System", 2020.

[8] Viraj Shinde, Tushar Bacchav, Jitendra Pawar and Mangesh Sanap, "Hand Gesture Recognition System Using Camera", 2019

[9] M. Heikkilä and M. Pietikäinen, "A Multiview Approach to Gesture Recognition with Invariant Features," Pattern Recognition, vol. 33, no. 5, pp. 915-928, 2018