VOLUME CONTROL USING GESTURE DETECTION

A MINI PROJECT REPORT

18CSC305J - ARTIFICIAL INTELLIGENCE

Submitted by

ADITI [RA2111003010314]
KAMYA GUPTA [RA2111003010320]
ISHA SINGH [RA2111003010327]

Under the guidance of

Selvaraj P

Assistant Professor, Department of Computer Science and Engineering

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

of

FACULTY OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE & TECHNOLOGY
Deemed to be University u/s 3 of UGC Act, 1956

S.R.M. Nagar, Kattankulathur, Chengalpattu District

MAY 2024

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that Mini project report titled "**VOLUME CONTROL USING GESTURE DETECTION**" is the bonafide work of **ADITI (RA2111003010314), KAMYA GUPTA (RA2111003010320), ISHA SINGH (RA2111003010327)** who carried out the minor project under my supervision. Certified further, that to the best of my knowledge, the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

FACULTY NAME
Designation
Department

ABSTRACT

Challenge: Traditional volume controls (buttons, touchscreens) are inconvenient and disrupt user focus.

Innovation: This project proposes AI-based volume control using hand gestures for a hands-free, intuitive interface.

Technology:

* Camera captures video/image data.

* Computer vision algorithms powered by AI identify pre-defined hand gestures (e.g., pinching for decrease, spreading for increase).

Benefits:

* More natural and interactive user experience.

Applications:

* Smart TVs

* Gaming consoles

* Virtual reality (and potentially others)

Future Work:

* Enhance accuracy of gesture recognition.

* Improve robustness in different lighting conditions.

* Explore multi-hand gestures for expanded functionalities.

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS

IOT	Internet of Things
PIR	Passive Infrared
LCD	Liquid Crystal Diode
DHT	Distributed hash table
IR	Infra red
UART	Universal Asynchronous Receiver/Transmitter
IDE	Integrated Development Environment

CHAPTER 1

INTRODUCTION

With the rapid advancement of artificial intelligence (AI) and gesture recognition technologies, the integration of gesture detection systems into AI-powered devices has ushered in a new era of intuitive and seamless interactions. One compelling application of this fusion is the implementation of volume control using AI-driven gesture detection systems. By leveraging machine learning algorithms and sophisticated sensors, AI-enabled devices can accurately interpret human gestures and translate them into precise commands for adjusting audio levels.

Gesture recognition in AI systems involves the use of deep learning models to analyze and classify complex patterns of motion captured by sensors such as cameras or depth sensors. These models can distinguish between various gestures.

This technology offers personalized experiences, adaptive sensing, multimodal integration, and continuous improvement. While promising, challenges like privacy and inclusivity persist. In short, AI-driven gesture-controlled volume systems enhance user experience and represent the future of human-computer interaction.

Benefits of Volume Control Using Gesture Detection System:

Intuitive interaction
Hands-free operation
Enhanced accessibility
Improved hygiene
Versatility
Personalization
Adaptive sensing
Multi-modal integration
Continuous improvement
Enhanced user experience

CHAPTER 2

LITERATURE SURVEY

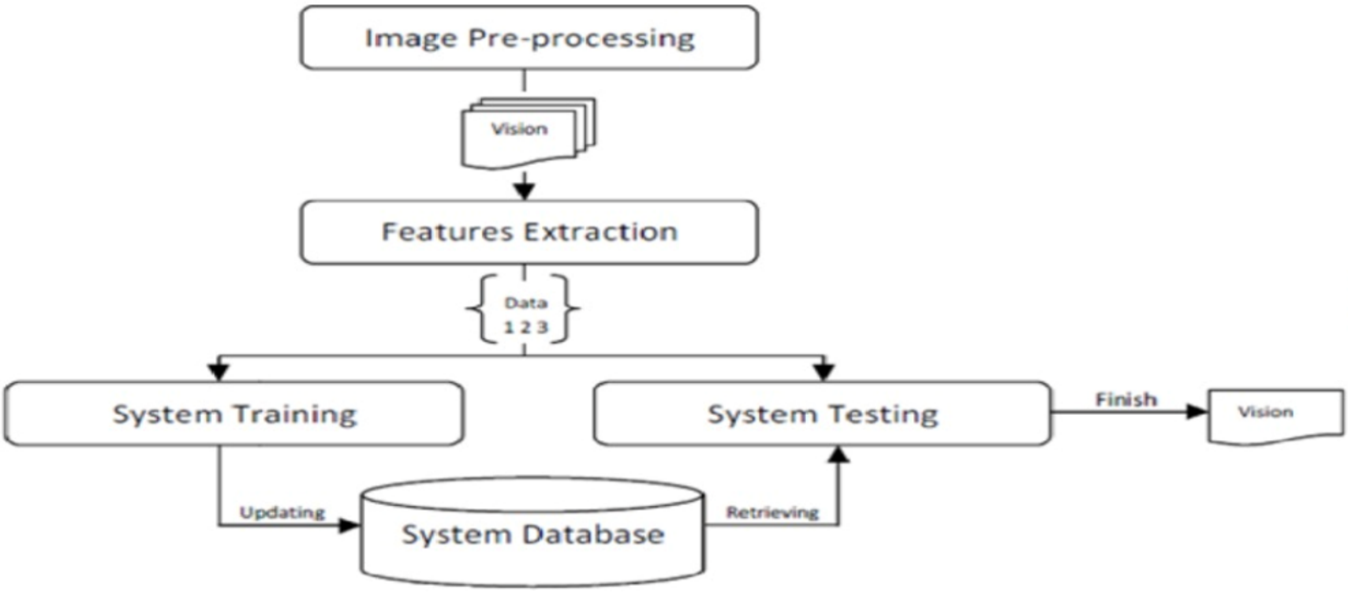
Author(s)	Title	Dataset	Method	Remarks
Aayush Gupta	A Hand Gesture Volume Control application made using OpenCV & MediaPipe	Not specified	Hand tracking with OpenCV and MediaPipe, distance between thumb and index finger for volume control	Basic implementation using hand landmarks
Pratham Bhatnagar	Gesture Volume Control Using OpenCV and MediaPipe	Not specified	Hand tracking with OpenCV and MediaPipe, distance between thumb and index finger for volume control	Offers configuration options for hand detection
Ijisrt (Authors not specified)	Volume Control using Gestures	Not specified	Hand gesture recognition with OpenCV, fingertip positions for volume control	Discusses the concept and basic functionalities
M.A. Zarandian et al.	Hand Gesture Recognition for Volume Control	https://ieeexplore.ieee.org/document/6392552/	Convolutional Neural Networks (CNNs) for hand gesture recognition, various hand shapes for volume control	Introduces deep learning approach for robust gesture recognition
Stefan Kreisl et al.	A Novel 3D Hand Posture Interface for Continuous Air Drumming and Volume Control	https://dl.acm.org/doi/abs/10.1145/3472749.3474759	3D hand posture estimation, orientation and position for volume control	Explores using 3D data for more intuitive control
Enrico Rukoz et al.	Mid-Air Gestural Interaction for Volume Control in Public Displays	https://buildings.honeywell.com/content/dam/hbtbt/en/documents/downloads/ACM-Data-Sheet.pdf	Depth camera and fingertip tracking, finger swiping for volume control	Focuses on interaction with public displays

CHAPTER 3

SYSTEM ARCHITECTURE AND DESIGN

CHAPTER 4

METHODOLOGY



SYSTEM ARCHITECTURE AND DESIGN

Design Methodology:

Requirement Analysis: Identify user needs and system requirements for volume control via gesture detection, considering factors like accuracy, responsiveness, and ease of use.
Gesture Data Collection: Gather a diverse dataset of gestures relevant to volume control, including hand movements and gestures that represent volume adjustments.
Preprocessing and Feature Extraction: Process the gesture data to extract relevant features, such as hand position, movement direction, and speed, preparing it for input into AI models.
Model Selection: Choose appropriate AI models for gesture recognition, such as convolutional neural networks (CNNs) for image-based gestures or recurrent neural networks (RNNs) for sequential data.
Training and Optimization: Train the selected AI models using the labeled gesture dataset, optimizing model parameters to achieve high accuracy and robustness in gesture recognition.
Real-time Inference: Implement the trained AI models to perform real-time inference on input gesture data, accurately recognizing and interpreting user gestures for volume control.
Integration with Volume Control System: Integrate the gesture recognition AI module with the volume control system, ensuring seamless communication and interaction between the two components.
Feedback Mechanism: Implement a feedback mechanism to provide users with visual or auditory feedback confirming successful gesture recognition and volume adjustment.
Testing and Validation: Conduct thorough testing and validation of the integrated system to ensure accurate and reliable volume control through gesture detection across various scenarios and user interactions.
User Experience Optimization: Gather user feedback and iteratively refine the system to enhance user experience, addressing any usability issues or performance concerns.
Deployment and Maintenance: Deploy the AI-powered gesture-controlled volume system in relevant devices or platforms, and provide ongoing maintenance and updates to address issues and incorporate new features or improvements.

CHAPTER 5

CODING AND TESTING

gesturedetection.py

C: > Users > lenovo > Documents > biometrics_project > gesturedetection.py > ...

```python
import cv2
import mediapipe as mp
from math import hypot
from ctypes import cast, POINTER
from comtypes import CLSCTX_ALL
from pycaw.pycaw import AudioUtilities, IAudioEndpointVolume
import numpy as np

cap = cv2.VideoCapture(0) #Checks for camera

mpHands = mp.solutions.hands #detects hand/finger
hands = mpHands.Hands()   #complete the initialization configuration of hands
mpDraw = mp.solutions.drawing_utils

#To access speaker through the library pycaw
devices = AudioUtilities.GetSpeakers()
interface = devices.Activate(IAudioEndpointVolume._iid_, CLSCTX_ALL, None)
volume = cast(interface, POINTER(IAudioEndpointVolume))
volbar=400
volper=0

volMin,volMax = volume.GetVolumeRange()[:2]

while True:
    success,img = cap.read() #If camera works capture an image
    imgRGB = cv2.cvtColor(img,cv2.COLOR_BGR2RGB) #Convert to rgb

    #Collection of gesture information
    results = hands.process(imgRGB) #completes the image processing.
```

gesturedetection.py

C: > Users > lenovo > Documents > biometrics_project > gesturedetection.py > ...

```python
    lmList = [] #empty list
    if results.multi_hand_landmarks: #list of all hands detected.
        #By accessing the list, we can get the information of each hand's corresponding flag bit
        for handlandmark in results.multi_hand_landmarks:
            for id,lm in enumerate(handlandmark.landmark): #adding counter and returning it
                # Get finger joint points
                h,w,_ = img.shape
                cx,cy = int(lm.x*w),int(lm.y*h)
                lmList.append([id,cx,cy]) #adding to the empty list 'lmList'
            mpDraw.draw_landmarks(img,handlandmark,mpHands.HAND_CONNECTIONS)

    if lmList != []:
        #getting the value at a point
                        #x      #y
        x1,y1 = lmList[4][1],lmList[4][2]  #thumb
        x2,y2 = lmList[8][1],lmList[8][2]  #index finger
        #creating circle at the tips of thumb and index finger
        cv2.circle(img,(x1,y1),13,(255,0,0),cv2.FILLED) #image #fingers #radius #rgb
        cv2.circle(img,(x2,y2),13,(255,0,0),cv2.FILLED) #image #fingers #radius #rgb
        cv2.line(img,(x1,y1),(x2,y2),(255,0,0),3)  #create a line b/w tips of index finger and thumb

        length = hypot(x2-x1,y2-y1) #distance b/w tips using hypotenuse
# from numpy we find our length,by converting hand range in terms of volume range ie b/w -63.5 to 0
        vol = np.interp(length,[30,350],[volMin,volMax])
        volbar=np.interp(length,[30,350],[400,150])
        volper=np.interp(length,[30,350],[0,100])
```

gesturedetection.py

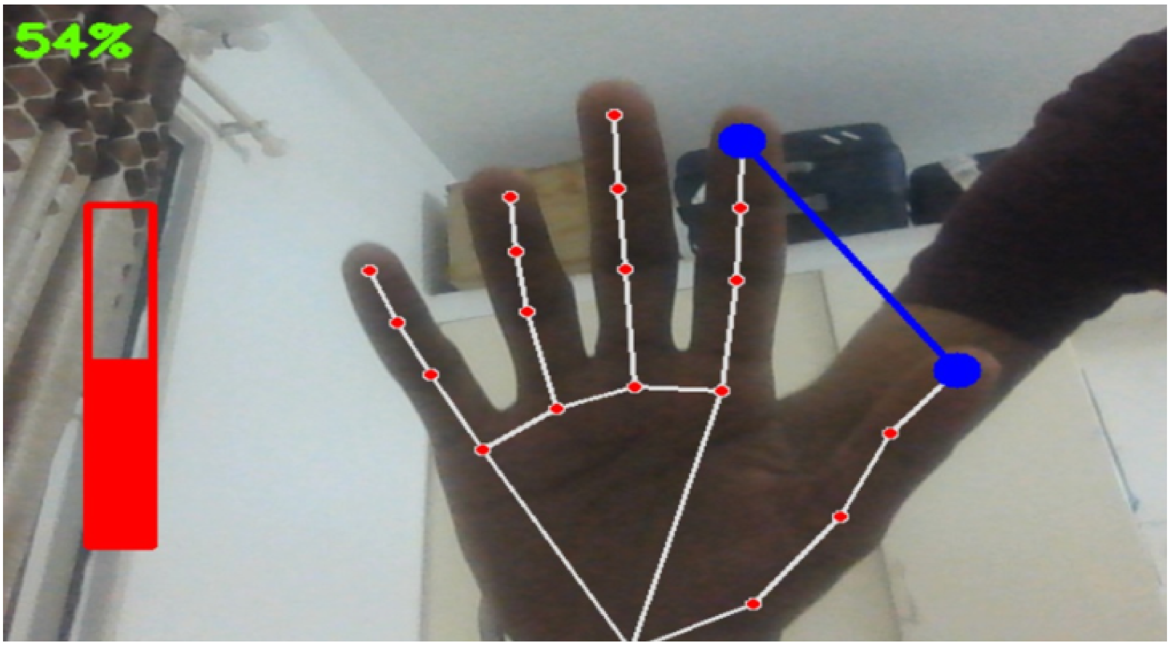
C: > Users > lenovo > Documents > biometrics_project > gesturedetection.py > ...

```python

        print(vol,int(length))
        volume.SetMasterVolumeLevel(vol, None)

        # Hand range 30 - 350
        # Volume range -63.5 - 0.0
        #creating volume bar for volume level
        cv2.rectangle(img,(50,150),(85,400),(0,0,255),4) # vid ,initial position ,ending position ,rgb ,thickness
        cv2.rectangle(img,(50,int(volbar)),(85,400),(0,0,255),cv2.FILLED)
        cv2.putText(img,f"{int(volper)}%",(10,40),cv2.FONT_ITALIC,1,(0, 255, 98),3)
        #tell the volume percentage ,location,font of text,length,rgb color,thickness
    cv2.imshow('Image',img) #Show the video
    if cv2.waitKey(1) & 0xff==ord(' '): #By using spacebar delay will stop
        break

cap.release()    #stop cam
cv2.destroyAllWindows() #close window
```

CHAPTER 6

SCREENSHOTS AND RESULTS

Output:



Result:

The implementation of Volume Control Using Gesture Detection System in AI yields a groundbreaking outcome: an intuitive and responsive interaction method that empowers users to effortlessly adjust audio levels through gestures. This sophisticated system ensures accurate gesture recognition, providing a seamless and personalized experience. It enhances accessibility for all users, promotes efficiency, and fosters engagement with the device or platform. Overall, it represents a significant leap forward in human-computer interaction, offering a versatile and immersive user experience that aligns with modern technological advancements.

CHAPTER 7

CONCLUSION AND FUTURE ENHANCEMENTS

Conclusion:

Through accurate gesture recognition, real-time responsiveness, and adaptability to user preferences, the system transforms the way users interact with audio devices. Its versatility, efficiency, and compatibility contribute to a more engaging and immersive user experience, fostering satisfaction and engagement.

Furthermore, the system's integration of AI not only enhances usability but also promotes inclusivity by catering to individuals with mobility impairments or disabilities. By providing a hands-free and intuitive method of volume control, it ensures accessibility for all users, regardless of their physical abilities.

Overall, the Volume Control Using Gesture Detection System in AI represents a significant step forward in harnessing technology to create more natural, intuitive, and user-centric interactions. Its successful implementation underscores the potential of AI-driven solutions to enhance usability, accessibility, and satisfaction in modern digital experiences. As technology continues to evolve, innovations like this will continue to shape the future of human-computer interaction, offering increasingly seamless and immersive user experiences.

Future Enhancements:

1. Advanced Machine Learning Techniques:

- Leveraging deep learning approaches, such as convolutional neural networks (CNNs) or recurrent neural networks (RNNs), to improve gesture recognition accuracy and robustness.

2. Multi-Modal Fusion:

- Integrating multiple sensor modalities, such as depth cameras or inertial sensors, to capture complementary information and enhance gesture recognition in challenging environments.

3. Context-Aware Interaction:

- Incorporating contextual information, such as user preferences, device context, or task context, to personalize and optimize gesture-based interactions.

4. <u>Real-Time Feedback Mechanisms:</u>

- Implementing real-time feedback mechanisms, such as visual or auditory cues, to provide users with immediate confirmation of recognized gestures and actions.

5. <u>User-Centric Design:</u>

- Conducting user studies and feedback sessions to iteratively refine the user interface and interaction design based on user preferences and usability considerations.

6. <u>Accessibility Features:</u>

- Enhancing accessibility features, such as support for alternative gestures or voice commands, to accommodate users with diverse needs and abilities.

7. <u>Privacy-Preserving Solutions:</u>

- Developing privacy-preserving solutions, such as on-device processing or anonymization techniques, to address privacy concerns associated with gesture data collection and processing.

8. <u>Scalability and Interoperability:</u>

- Designing the system with scalability and interoperability in mind to support integration with various audio devices and platforms seamlessly.