

VOLUME CONTROL WITH HAND GESTURE USING PYTHON OPENCV

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Computer Science

By
SHARMILA.P

REG NO. 40290091



DEPARTMENT OF COMPUTER SCIENCE
SCHOOL OF SCIENCE AND HUMANITIES

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A" by NAAC | 12B Status

By UGC | Approved by AICTE

JEPPIAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI 600119

MAY 2023



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOG

(DEEMED TO BE UNIVERSITY)

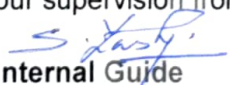
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the Bonafide work of **SHARMILA.P (40290091)** who carried out the Project entitled "**VOLUME CONTROL WITH HAND GESTURE USING PYTHON OPENCV.**" under our supervision from January 2023 to May 2023.


Internal Guide

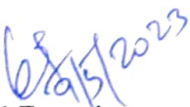
MS. YUVA SNEHA S, M.Sc., Assistant Professor

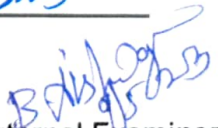

Dean of arts and science

Dr. REKHA CHAKRAVARTHI M.E., Ph.D.

Submitted for Viva-voce Examination held on

09/05/2023


Internal Examiner


External Examiner

DECLARATION

I am, **SHARMILA.P (40290091)** hereby declare that the Project Report entitled **“VOLUME CONTROL WITH HAND GESTURE USING PYTHON OPENCV.”** done by me under the guidance of **Ms.YUVA SNEHA S, M.Sc., Assistant Professor** Department of Computer Science at **SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY**, Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai-600119 is submitted in partial fulfillment of the requirements for the award of Bachelor of degree in Computer Science.

DATE:

PLACE: Chennai



SHARMILA.P

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to the **Board of Management** of **SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY** for their kind encouragement in doing this Project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. REKHA CHAKRAVARTHI M.E., Ph.D.**, Dean of arts and science for providing me with necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guides, of **Ms. YUVA SNEHA S, M.Sc.**, Assistant Professor, her valuable guidance, suggestions, and constant encouragement paved the way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the Department of Computer Science who were helpful in many ways for the completion of the Project.

TABLE OF CONTENTS

CHAPTER	TITLE	PAGE NO.
	ABSTRACT	VII
	LIST OF FIGURES	VI
1	INTRODUCTION	1
	1.1 PROJECT INTRODUCTION	1
	1.2 LITERATURE SURVEY	2
2	WORKING ENVIRONMENT	5
	2.1 HARDWARE REQUIREMENT	5
	2.2 SOFTWARE REQUIREMENT	5
	2.3 SYSTEM SOFTWARE	5
3	SYSTEM ANALYSIS	23
	3.1 FEASIBILITY STUDY	23
	3.3 EXISTING SYSTEM	24
	3.4 PROPOSED SYSTEM	24
	3.5 BENEFITS OF PROPOSEDSYSTEM	24
	3.6 METHODOLOGY/APPROACH	24
	3.7 SCOPE OF THE PROJECT	25
4	SYSTEM DESIGN	26
	4.1 DATA FLOW DIAGRAM	26
	4.2 USE CASE DIAGRAM	27
	4.3 CLASS DIAGRAM	28
	4.4 ARCHITECTURE DIAGRAM	29

	4.5 SEQUENCE DIAGRAM	30
5	PROJECT DESCRIPTION	31
	5.1 OBJECTIVE	31
	5.2 MODULE DESCRIPTION	31
	5.3 IMPLEMENTATION	33
6	SYSTEM TESTING	34
	6.1 TESTING AND ITS TYPES	34
7	CONCLUSION	37
	7.1 CONCLUSION	37
8	APPENDIX	38
	8.1 SAMPLE CODING	38
	8.2 SCREENSHOTS	44
9	BIBLIOGRAPHY AND REFERENCES	48

LIST OF FIGURES

Figure No.	Figures Name
4.1	Data Flow Diagram
4.2	Use Case Diagram
4.3	Class Diagram
4.4	Architecture Diagram
4.5	Sequence Diagram
8.2.1	Web Access
8.2.2	Hand Tracking
8.2.3	Volume Getting Decreased
8.2.4	Volume Getting Increased

ABSTRACT

A hand gesture recognition system provides a natural, innovative and modern way of non-verbal communication. It has a wide area of application in human computer interaction and sign language. The intention of this implementation is to discuss a novel approach of hand gesture recognition based on detection of some shape based features. The setup consists of a single camera to capture the gesture formed by the user and take this as input to the system. A primary goal of gesture recognition is to create a system which can identify specific human gestures and use them to convey information for device control and by implementing real time gestures recognition a user can control a computer by doing a specific gesture in front of a video camera linked to a computer. In this project we will develop a hand gesture volume control system with the help of OpenCV module. Here the system can be operated using hand gestures without using keyboard and mouse. The Aim of the project is to control the volume of a system using Python Open CV Using the concept of Motion tracking we can control the volume of system in this project. So, using the motion tracking method we track the motion of our FINGERS to increase or decrease the volume of a system.

CHAPTER-1

INTRODUCTION

1.1 PROJECT INTRODUCTION

In this generation with the development of technologies in the areas of augmented reality, virtual reality, etc and devices that we use in our daily life are becoming compact in the form of Bluetooth or wireless technologies. This report proposes an idea that makes use of the hand gestures, movements for performing volume control functions in the users computer using computer vision. The main objective of the proposed system is to perform volume control functions using a web camera or a built in camera in the computer instead of using a handheld mouse device. With the use of the 2 important modules, we can track the fingertip of the hand by using a built-in camera or web camera we can perform the volume control functions. While using a wired or a wireless mouse, some disadvantage such as the dongle to connect to the PC, and a battery to power the mouse to operate are used, but in this project, the user uses their built in camera or a webcam and uses hand gestures for volume control operations. Gesture recognition helps computers to understand human body language. This helps to build a more potent link between humans and machines, rather than just the basic text user interfaces or graphical user interfaces (GUIs). In this project for gesture recognition, the human body's motions are read by computer camera. The computer then makes use of this data as input to handle applications. The objective of this project is to develop an interface which will capture human hand gesture dynamically and will control the volume level. NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level

mathematical functions to operate on these arrays. Pycaw : Python Audio Control Library Media pipe is an open-source machine learning library of Google, which has some solutions for face recognition and gesture recognition, and provides encapsulation of python, is and other languages.

1.2 LITERATURE SURVEY

A non-local algorithm for hand gestures was proposed by A. Buades, B. Coll, and J. Morel. At the moment, finding finger movement algorithms remains a valid task. Functional analysis and statistics collide. Despite the fact that most recently presented approaches have a high level of sophistication, Algorithms have not yet reached a satisfactory degree of performance applicability. All work admirably when the model matches the algorithm assumptions, but they all fail in general, producing defects in analysing the pixels through the camera. The primary goal of this study is to define a generic mathematical and experimental technique for comparing and classifying conventional hand movement recognition algorithms.

For the no required elements in the video frame, Golam Moktader Daiyan et al. (2012) suggested a high performance decision based median filter. This technique detects noise pixels iteratively over numerous phases before replacing them with the median value. Noise detection is accomplished by enlarging the field of view. Mask till 77% to keep the extraction of local data going. Furthermore, if the algorithm fails to find a noise-free pixel at 7 7, the processing pixel is replaced by the last processed pixel. If the noise-free median value isn't available in the 7th processing window, the last processed pixel is used to determine if it is noise-free. The method chooses a window size if the last processed pixel is noisy. Calculate the number of 0s and 255s in the processing window using the 1515 dimension. Then, in

the selected window, replace the processed pixel with 0 or 255, whichever is higher in number.

Rajeshwari Kumar Dewangan et.al accurate object information and obtain a location using a deep learning object recognition technique. Object recognition algorithms are designed based on the Single Shot MultiBox Detector (SSD) structure, an object recognition deep learning model, to detect objects using a camera.

H. Jabnoun et, al suggested the system that restores a central function of the visual system which is the identification of surrounding objects which is based on the local features extraction concept. Using SFIT algorithm and keypoints matching showed good accuracy for detecting objects.

Košale U, Žnidaršic P, Stopar K suggested that Detection of obstacles is performed by Time of Flight (ToF) sensors, whose ranging data is then processed with an on-board microcontroller and send via Bluetooth connection to the belt. The belt is equipped with a second microcontroller, which interprets the data and presents it to the wearer with 15 vibration motors arranged in a square grid. The glasses are worn on the head, whereas the belt is attached around the stomach area. But the number of sensors detecting the obstacle decreased with the distance. Circle and square were detected better than triangle. This suggests that different shapes trigger different responses of sensors on glasses. A. Jaiswal et al.

Proposed an approach that used user generated picture denoising. The remaining task is broken down into four steps. The first image is denoised using a filtering process, and the second image is denoised using a different method. Wavelet-based approaches are used to denoise pictures, filtering, third hard thresholding, and thresholding. Finally, the approach was applied to a noisy image concurrently. PSNR

output results are calculated by comparing all cases, the MSE (mean square error) is obtained. On the basis of PSNR, MSE, and image visual quality, experiments are conducted on 512 X 512 noisy images with noise variance of 0.04, output of median filter, Wiener filter, hard thresholding, and hard thresholding with median filtering. When the filtering and wavelet thresholding techniques are combined, they produce excellent results.

CHAPTER-2

WORKING ENVIRONMENT

2.1 HARDWARE REQUIREMENT

PROCESSOR	:	Core i5Processor
HARD DISK	:	256 GB
WEBCAM	:	5MP
RAM	:	4GB

2.2 SOFTWARE REQUIREMENT

Operating system	:	Windows 10
Simulation Tool	:	OPENCV PYTHON/MEDIA PIPE
Coding Language	:	Python 3.10.1
Development IDE	:	Pycharm 2021.3 (Community Edition)

2.3 SYSTEM SOFTWARE

2.3.1. PYTHON

Python is a high-level, interpreted, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small- and large-scale projects. Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly procedural), object-oriented and functional programming. It is often described as a "batteries included" language due

to its comprehensive standard library.

Guido van Rossum began working on Python in the late 1980s as a successor to the ABC programming language and first released it in 1991 as Python 0.9.0.[33] Python 2.0 was released in 2000 and introduced new features such as list comprehensions, cycle-detecting garbage collection, reference counting, and Unicode support. Python 3.0, released in 2008, was a major revision that is not completely backward-compatible with earlier versions. Python 2 was discontinued with version 2.7.18 in 2020.

Python consistently ranks as one of the most popular programming languages. Python was conceived in the late 1980s by Guido van Rossum at Centrum Wiskunde & Informatica (CWI) in the Netherlands as a successor to the ABC programming language, which was inspired by SETL, capable of exception handling and interfacing with the Amoeba operating system. Its implementation began in December 1989. Van Rossum shouldered sole responsibility for the project, as the lead developer, until 12 July 2018, when he announced his "permanent vacation" from his responsibilities as Python's "benevolent dictator for life", a title the Python community bestowed upon him to reflect his long-term commitment as the project's chief decision-maker. In January 2019, active Python core developers elected a five-member Steering Council to lead the project.

Python 2.0 was released on 16 October 2000, with many major new features.[45] Python 3.0, released on 3 December 2008, with many of its major features backported to Python 2.6.x and 2.7.x. Releases of Python 3 include the 2to3 utility, which automates the translation of Python 2 code to Python 3.[47] Python 2.7's end-of-life was initially set for 2015, then postponed to 2020 out of concern that a large body of existing code could not easily be forward-ported to Python 3.[48][49]

No further security patches or other improvements will be released for it. With Python 2's end-of-life, only Python 3.6 and later were supported, and later support for 3.6 also dropped.

PYTHON VERSION 2.0

A new release of Python, version 2.0, was released on October 16, 2000. This article covers the exciting new features in 2.0, highlights some other useful changes, and points out a few incompatible changes that may require rewriting code.

Python's development never completely stops between releases, and a steady flow of bug fixes and improvements are always being submitted. A host of minor fixes, a few optimizations, additional docstrings, and better error messages went into 2.0; to list them all would be impossible, but they're certainly significant. Consult the publicly available CVS logs if you want to see the full list. This progress is due to the five developers working for PythonLabs are now getting paid to spend their days fixing bugs, and also due to the improved communication resulting from moving to Source Forge.

Python 1.6 can be thought of as the Contractual Obligations Python release. After the core development team left CNRI in May 2000, CNRI requested that a 1.6 release be created, containing all the work on Python that had been performed at CNRI. Python 1.6 therefore represents the state of the CVS tree as of May 2000, with the most significant new feature being Unicode support. Development continued after May, of course, so the 1.6 tree received a few fixes to ensure that it's forward-compatible with Python 2.0. 1.6 is therefore part of Python's evolution, and not a side branch.

So, should you take much interest in Python 1.6? Probably not. The 1.6final and 2.0beta1 releases were made on the same day (September 5, 2000), the plan

being to finalize Python 2.0 within a month or so. If you have applications to maintain, there seems little point in breaking things by moving to 1.6, fixing them, and then having another round of breakage within a month by moving to 2.0; you're better off just going straight to 2.0. Most of the really interesting features described in this document are only in 2.0, because a lot of work was done between May and September.

NEW DEVELOPMENT PROCESS

The most important change in Python 2.0 may not be to the code at all, but to how Python is developed: in May 2000 the Python developers began using the tools made available by SourceForge for storing source code, tracking bug reports, and managing the queue of patch submissions. To report bugs or submit patches for Python 2.0, use the bug tracking and patch manager tools available from Python's project page, located at [The most important of the services now hosted at SourceForge is the Python CVS tree](#), the version-controlled repository containing the source code for Python. Previously, there were roughly 7 or so people who had write access to the CVS tree, and all patches had to be inspected and checked in by one of the people on this short list. Obviously, this wasn't very scalable. By moving the CVS tree to SourceForge, it became possible to grant write access to more people; as of September 2000 there were 27 people able to check in changes, a fourfold increase. This makes possible large-scale changes that wouldn't be attempted if they'd have to be filtered through the small group of core developers. For example, one day Peter Schneider-Kamp took it into his head to drop K&R C compatibility and convert the C source for Python to ANSI C. After getting approval on the python-dev mailing list, he launched into a flurry of checkins that lasted about a week, other developers joined in to help, and the job was done. If there were only 5 people with

write access, probably that task would have been viewed as “nice, but not worth the time and effort needed” and it would never have gotten done.

The shift to using Source Forge’s services has resulted in a remarkable increase in the speed of development. Patches now get submitted, commented on, revised by people other than the original submitter, and bounced back and forth between people until the patch is deemed worth checking in. Bugs are tracked in one central location and can be assigned to a specific person for fixing, and we can count the number of open bugs to measure progress. This didn’t come without a cost: developers now have more e-mail to deal with, more mailing lists to follow, and special tools had to be written for the new environment. For example, SourceForge sends default patch and bug notification e-mail messages that are completely unhelpful, so Ka-Ping Yee wrote an HTML screen-scraper that sends more useful messages.

The ease of adding code caused a few initial growing pains, such as code was checked in before it was ready or without getting clear agreement from the developer group. The approval process that has emerged is somewhat similar to that used by the Apache group. Developers can vote +1, +0, -0, or -1 on a patch; +1 and -1 denote acceptance or rejection, while +0 and -0 mean the developer is mostly indifferent to the change, though with a slight positive or negative slant. The most significant change from the Apache model is that the voting is essentially advisory, letting Guido van Rossum, who has Benevolent Dictator For Life status, know what the general opinion is. He can still ignore the result of a vote, and approve or reject a change even if the community disagrees with him.

Producing an actual patch is the last step in adding a new feature, and is usually easy compared to the earlier task of coming up with a good design.

Discussions of new features can often explode into lengthy mailing list threads, making the discussion hard to follow, and no one can read every posting to python-dev. Therefore, a relatively formal process has been set up to write Python Enhancement Proposals (PEPs), modelled on the internet RFC process. PEPs are draft documents that describe a proposed new feature, and are continually revised until the community reaches a consensus, either accepting or rejecting the proposal. Quoting from the introduction to PEP 1, "PEP Purpose and Guidelines":

PEP stands for Python Enhancement Proposal. A PEP is a design document providing information to the Python community, or describing a new feature for Python. The PEP should provide a concise technical specification of the feature and a rationale for the feature.

We intend PEPs to be the primary mechanisms for proposing new features, for collecting community input on an issue, and for documenting the design decisions that have gone into Python. The PEP author is responsible for building consensus within the community and documenting dissenting opinions.

ADVANTAGES OF PYTHON

1. Easy to Read, Learn and Write

Python is a high-level programming language that has English-like syntax.

This makes it easier to read and understand the code.

Python is really easy to pick up and learn, that is why a lot of people recommend Python to beginners. You need less lines of code to perform the same task as compared to other major languages like C/C++ and Java.

2. Improved Productivity

Python is a very productive language. Due to the simplicity of Python, developers can focus on solving the problem. They don't need to spend too much

time in understanding the syntax or behavior of the programming language. You write less code and get more things done.

3. Interpreted Language

Python is an interpreted language which means that Python directly executes the code line by line. In case of any error, it stops further execution and reports back the error which has occurred. Python shows only one error even if the program has multiple errors. This makes debugging easier.

4. Dynamically Typed

Python doesn't know the type of variable until we run the code. It automatically assigns the data type during execution. The programmer doesn't need to worry about declaring variables and their data types.

5. Free and Open-Source

Python comes under the OSI approved open-source license. This makes it free to use and distribute. You can download the source code, modify it and even distribute your version of Python. This is useful for organizations that want to modify some specific behavior and use their version for development.

6. Vast Libraries Support

The standard library of Python is huge, you can find almost all the functions needed for your task. So, you don't have to depend on external libraries. Portability In many languages like C/C++, you need to change your code to run the program on different platforms. That is not the same with Python. You only write once and run it anywhere. However, you should be careful not to include any system-dependent features.

DISADVANTAGES OF PYTHON

1. Slow Speed

We discussed above that Python is an interpreted language and dynamically-typed language. The line by line execution of code often leads to slow execution.

The dynamic nature of Python is also responsible for the slow speed of Python because it has to do the extra work while executing code. So, Python is not used for purposes where speed is an important aspect of the project.

2. Not Memory Efficient

To provide simplicity to the developer, Python has to do a little tradeoff. The Python programming language uses a large amount of memory. This can be a disadvantage while building applications when we prefer memory optimization.

3. Weak in Mobile Computing

Python is generally used in server-side programming. We don't get to see Python on the client-side or mobile applications because of the following reasons. Python is not memory efficient and it has slow processing power as compared to other languages.

4. Database Access

Programming in Python is easy and stress-free. But when we are interacting with the database, it lacks behind.

The Python's database access layer is primitive and underdeveloped in comparison to the popular technologies like JDBC and ODBC.

Huge enterprises need smooth interaction of complex legacy data and Python is thus rarely used in enterprises.

5. Runtime Errors

As we know Python is a dynamically typed language so the data type of a

variable can change anytime. A variable containing integer number may hold a string in the future, which can lead to Runtime Errors.

Therefore Python programmers need to perform thorough testing of the applications.

In 2022, Python 3.10.4 and 3.9.12 were expedited and so were older releases including 3.8.13, and 3.7.13 because of many security issues in 2022. In 2021, Python 3.9.2 and 3.8.8 were also expedited as all versions of Python (including 2.7) had security issues leading to possible remote code execution and web cache poisoning. Python 3.6 (and every older release), is no longer supported as of 2021.

What can Python do?

Python can be used on a server to create web applications. Python can be used alongside software to create work flows. Python can connect to database systems. It can also read and modify files. Python can be used to handle big data and perform complex mathematics. Python can be used for rapid prototyping, or for production-ready software development.

Why Python?

Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc). Python has a simple syntax similar to the English language. Python has syntax that allows developers to write programs with fewer lines than some other programming languages. Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick. Python can be treated in a procedural way, an object-oriented way or a functional way.

Python Syntax compared to other programming languages

Python was designed for readability, and has some similarities to the English language with influence from mathematics. Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses. Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpose.

About Python Programming

Free and open-source - You can freely use and distribute Python, even for commercial use.

Easy to learn - Python has a very simple and elegant syntax. It's much easier to read and write Python programs compared to other languages like C++, Java, and C #.

Portable - You can move Python programs from one platform to another, and run it without any changes.

Design philosophy and features

Python is a multi-paradigm programming language. Object-oriented programming and structured programming are fully supported, and many of its features support functional programming and aspect-oriented programming (including by metaprogramming and metaobjects [magic methods]). Many other paradigms are supported via extensions, including design by contract and logic programming.

Python uses dynamic typing, and a combination of reference counting and a cycle-detecting garbage collector for memory management. It uses dynamic name resolution (late binding), which binds method and variable names during program execution.

Its design offers some support for functional programming in the Lisp tradition. It has filter, map and reduce functions; list comprehensions, dictionaries, sets, and generator expressions. The standard library has two modules (itertools and functools) that implement functional tools borrowed from Haskell and Standard ML.

Its core philosophy is summarized in the document The Zen of Python (PEP 20), which includes aphorisms such as:

- Beautiful is better than ugly.
- Explicit is better than implicit.
- Simple is better than complex.
- Complex is better than complicated.
- Readability counts.

Typing

Python uses duck typing and has typed objects but untyped variable names. Type constraints are not checked at compile time; rather, operations on an object may fail, signifying that it is not of a suitable type. Despite being dynamically-typed, Python is strongly-typed, forbidding operations that are not well-defined (for example, adding a number to a string) rather than silently attempting to make sense of them.

Python allows programmers to define their own types using classes, most often used for object-oriented programming. New instances of classes are

constructed by calling the class (for example, `Spam Class` or `EggsClass()`), and the classes are instances of the metaclass type (itself an instance of itself), allowing metaprogramming and reflection.

Before version 3.0, Python had two kinds of classes (both using same syntax): old-style and new-style. Current Python versions only support the semantics new style. The long-term plan is to support gradual typing. Python's syntax allows specifying static types, but they are not checked in the default implementation, CPython. An experimental optional static type-checker, `mypy`, supports compile-time type checking.

Development

Python's development is conducted largely through the Python Enhancement Proposal (PEP) process, the primary mechanism for proposing major new features, collecting community input on issues and documenting Python design decisions.[156] Python coding style is covered in PEP 8. Outstanding PEPs are reviewed and commented on by the Python community and the steering council. Enhancement of the language corresponds with development of the C Python reference implementation.

2.3.2 Open CV

Open CV (Open Source Computer Vision Library) is a library of programming functions mainly aimed at real-time computer vision.[1] Originally developed by Intel, it was later supported by Willow Garage then Itseez (which was later acquired by Intel[2]). The library is cross-platform and free for use under the open-source Apache 2 License. Starting with 2011, OpenCV features GPU

acceleration for real-time operations

Programming language

Open CV is written in C++ and its primary interface is in C++, but it still retains a less comprehensive though extensive older C interface. All of the new developments and algorithms appear in the C++ interface. There are bindings in Python, Java and MATLAB/OCTAVE. The API for these interfaces can be found in the online documentation.[12] Wrappers in several programming languages have been developed to encourage adoption by a wider audience. In version 3.4, JavaScript bindings for a selected subset of Open CV functions was released as OpenCV.js, to be used for web platforms

MEDIAPIPE

MediaPipe powers revolutionary products and services we use daily. Unlike power-hungry machine learning Frameworks, MediaPipe requires minimal resources. It is so tiny and efficient that even embedded IoT devices can run it. In 2019, MediaPipe opened up a whole new world of opportunity for researchers and developers following its public release. If you have just started with MediaPipe and this is one of the first articles you are going through, congratulations, you are in the right place. This article will cover the basics of MediaPipe, the difference between Solutions, and the Framework. The official documentation states that inferencing is real-time, and it takes just a few lines of code to create a perception pipeline.

Graphs

The MediaPipe perception pipeline is called a **Graph**. Let us take the example of the first solution, Hands. We feed a stream of images as input which comes out with hand landmarks rendered on the images. The flow chart below represents the

MP (Abbr. MediaPipe) hand solution graph. In computer science jargon, a graph consists of Nodes connected by Edges. Inside the MediaPipe Graph, the nodes are called **Calculators**, and edges are called **Streams**. Every stream carries a sequence of **Packets** that have ascending time stamps.

Calculators

These are specific computation units written in C++ with assigned tasks to process. The packets of data (Video frame or audio segment) enter and leave through the **ports** in a calculator. When initializing a calculator, it declares the packet payload type that will traverse the port. Every time a graph runs, the Framework implements **Open**, **Process**, and **Close** methods in the calculators. **Open** initiates the calculator; the **process** repeatedly runs when a packet enters. The process is **closed** after an entire graph run. As an example, consider the first calculator shown in the above graph. The calculator, **Image Transform**, takes an image at the input port and returns a transformed image in the output port. On the other hand, the second calculator, **ImageToTensor**, takes an image as input and outputs a tensor.

Calculator Types in MediaPipe

All the calculators shown above are built-in into Media Pipe. We can group them into four categories.

1. **Pre-processing calculators** are family of image and media processing calculators. The **Image Transform** and **Image to Sensors** in the graph above fall in this category.
2. **Inference calculators** allow native integration with Tensorflow and Tensorflow Lite for ML inference.

3. **Post-processing calculators** perform ML post-processing tasks such as detection, segmentation, and classification. **TensorToLandmark** is a post-processing calculator.
4. **Utility calculators** are a family of calculators performing final tasks such as image annotation.

Media Pipe in Hand Detection

The ability to perceive the shape and motion of hands can be a vital component in improving the user experience across a variety of technological domains and platforms. For example, it can form the basis for sign language understanding and hand gesture control, and can also enable the overlay of digital content and information on top of the physical world in augmented reality. While coming naturally to people, robust real-time hand perception is a decidedly challenging computer vision task, as hands often occlude themselves or each other (e.g. finger/palm occlusions and handshakes) and lack high contrast patterns.

ML Pipeline

MediaPipe Hands utilizes an ML pipeline consisting of multiple models working together: A palm detection model that operates on the full image and returns an oriented hand bounding box. A hand landmark model that operates on the cropped image region defined by the palm detector and returns high-fidelity 3D hand keypoints. This strategy is similar to that employed in our MediaPipe Face Mesh solution, which uses a face detector together with a face landmark model.

Providing the accurately cropped hand image to the hand landmark model drastically reduces the need for data augmentation (e.g. rotations, translation and scale) and instead allows the network to dedicate most of its capacity towards

coordinate prediction accuracy. In addition, in our pipeline the crops can also be generated based on the hand landmarks identified in the previous frame, and only when the landmark model could no longer identify hand presence is palm detection

Models

Palm Detection Model

To detect initial hand locations, we designed a single-shot detector model optimized for mobile real-time uses in a manner similar to the face detection model in MediaPipe Face Mesh. Detecting hands is a decidedly complex task: our lite model and full model have to work across a variety of hand sizes with a large scale span ($\sim 20\times$) relative to the image frame and be able to detect occluded and self-occluded hands

Our method addresses the above challenges using different strategies. First, we train a palm detector instead of a hand detector, since estimating bounding boxes of rigid objects like palms and fists is significantly simpler than detecting hands with articulated fingers. In addition, as palms are smaller objects, the non-maximum suppression algorithm works well even for two-hand self-occlusion cases, like handshakes. Moreover, palms can be modelled using square bounding boxes (anchors in ML terminology) ignoring other aspect ratios, and therefore reducing the number of anchors by a factor of 3-5. Second, an encoder-decoder feature extractor is used for bigger scene context awareness even for small objects (similar to the RetinaNet approach). Lastly, we minimize the focal loss during training to support a large amount of anchors resulting from the high scale variance. With the above

CHAPTER-3

SYSTEM ANALYSIS

3.1 FEASIBILITY STUDY

The Feasibility study is a test of system proposal according to its workability, impact on the organization, ability to meet user needs and effective use of resources. The object of Feasibility study is not to solve the problem, but to acquire the sense of its scope. During the study the problem definition is crystallized and aspects of the problem to be included in the system are determined, consequently costs and benefits are estimated with greater detail at this stage. The result of Feasibility study is the system formal proposal. This is simply a form of documenting or detailing the nature and scope of proposed solutions. The proposal summarizes what is known and what is going to be done. Two key considerations involved in Feasibility analyses,

- I. Economic Feasibility
- II. Technical Feasibility

3.1.1 Economic Feasibility

As observed the current hardware functionality can be damaged easily. The method of Configuring a volume using a remote can be complex for a user. Nowadays, Majority of the screening devices has an in built web-cam in it which includes a Television too, so In our proposed system there wont be a need to get a separate webcam for this.

3.1.2 Technical Feasibility

The feasibility center on the existing computer system (application, hardware)

and to what extent it can support the proposed addition. In the proposal system gestures can be easily recognized by the in built-web cam. It gives immediate and real-time results. Therefore, the system is technically feasible.

3.3 Existing System

In the existing system the use of hardware is higher which results in the peak of e-wastes. For example EPA (ENVIRONMENTAL PROTECTION AGENCY) has found that in 2012 we generated 3.42 million tons of e-waste.

3.4 PROPOSED SYSTEM

The proposed system is quite simple to use. It is not complex in its functionalities. It is easy for a new or regular user to use it. It can vastly reduce the amount of e waste produced in a year too.

3.5 Benefits of Proposed System

- Users can interact with artificial environments in a safe and controlled setting.
- It gives immediate and real time result.
- It is easy to capture and faster than the traditional complex KEY movements.

3.6 Methodology/Approach

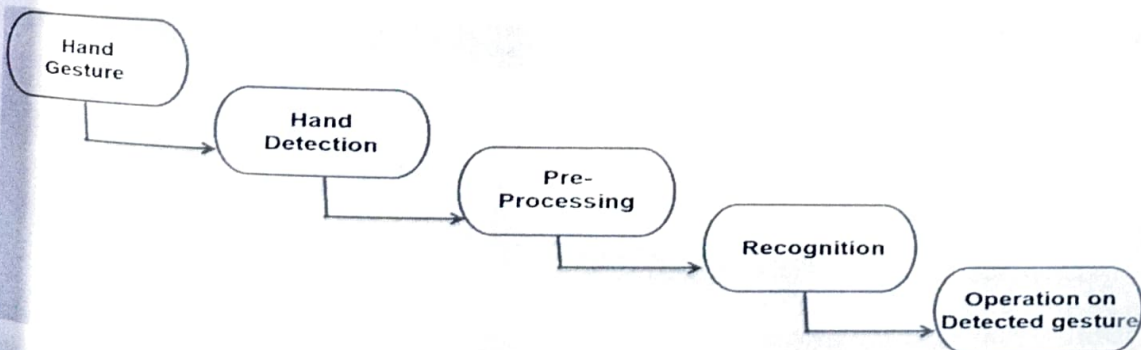
- Detect hand landmarks
- Calculate the distance between thumb tip and index finger tip.
- Map the distance of thumb tip and index finger tip with volume range. For my case, distance between thumb tip and index finger tip was within the range of 30 – 350 and the volume range was from -63.5 – 0.0.
- In order to exit press 'Spacebar'

CHAPTER-4

SYSTEM DESIGN

4.1 DATA FLOW DIAGRAM

The DFD takes an input-process-output view of a system i.e. data objects flow into the software, are transformed by processing elements, and resultant data objects flow out of the software. Data objects represented by labeled arrows and transformation are represented by circles also called as bubbles. DFD is presented in a hierarchical fashion i.e. the first data flow model represents the system as a whole. Subsequent DFD refine the context diagram (level 0 DFD), providing increasing details with each subsequent level.



4.1 Data Flow Diagram Figure

4.3 CLASS DIAGRAM

Class is nothing but a structure that contains both variables and methods.

The Class Diagram shows a set of classes, interfaces, and collaborations and their relating ships. There is most common diagram in modeling the object-oriented systems and are used to give the static view of a system. It shows the dependency between the classes that can be used in our system.

The interactions between the modules or classes of our projects are shown below. Each block contains Class Name, Variables and Methods.

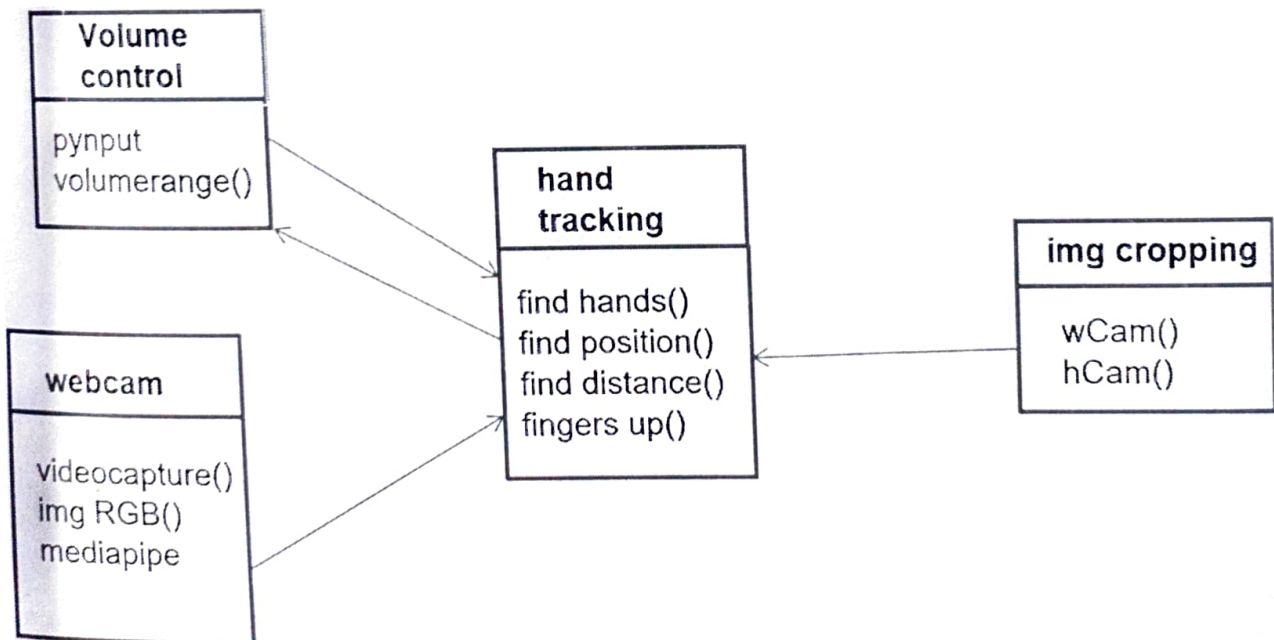


Figure 4.3 – Class Diagram

4.4 ARCHITECTURE DIAGRAM

An architecture diagram is a graphical representation of a set of concepts that are part of architecture, including their principles, elements and components.

There are many kinds of architecture diagrams, like a software architecture diagram, system architecture diagram, application architecture diagram, security architecture diagram, etc.

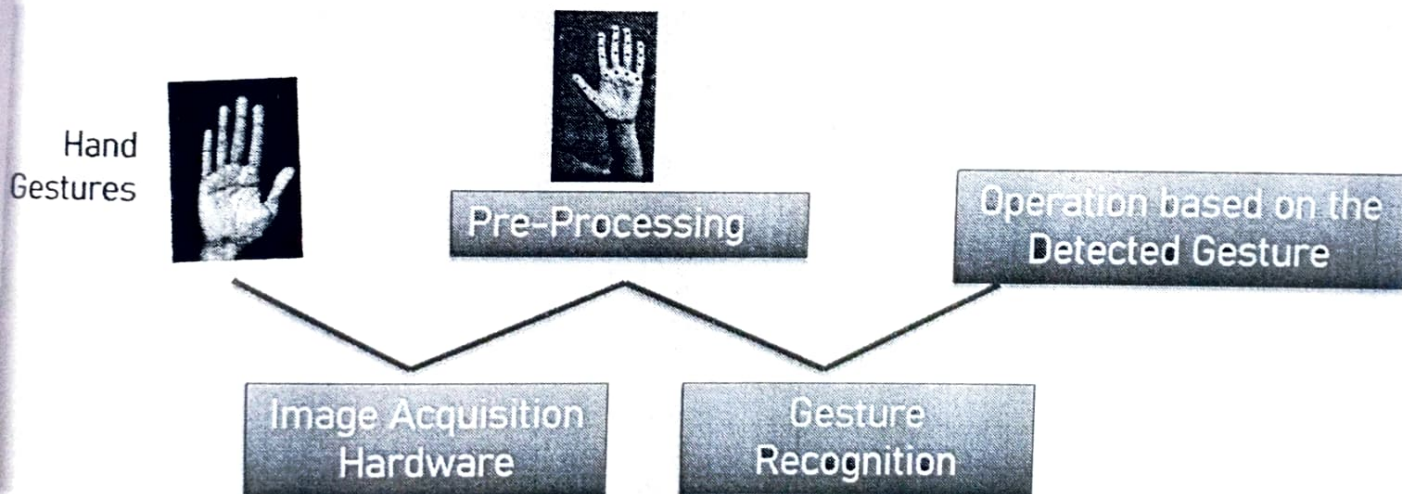


Figure 4.4 – Architecture diagram

CHAPTER-5

PROJECT DESCRIPTION

5.1 OBJECTIVE

- The main objective of our project is to configure between volume by using a hand gesture module.
- To control the large production of plastic E-Wastes which supports the ecological balance.

5.2 MODULE

- Hand Tracking Module
- Volume Control Module

5.2.1 Hand Tracking Module

We first import the Python libraries that we need in our project. This will enable us to use its dependencies. we create a class that we will use for tracking. We then key in the basic parameters that we require for the hands function to work. MediaPipe provides these parameters in the hands function.

Afterwards, we provide all the initializations that we need for our class. These are the parameters above and the MediaPipe initializations.

We put self before every object to allow access to the methods and the attributes of that object. This in turn allows each object to possess its own attributes and methods. we created a method that we will use to specifically track the hands in our input image. The code that goes in this method is the one that converts the image to RGB and processes the RGB image to locate the hands. It also draws the hand landmarks on the image and finally draws the hand connections. we created a

method that we will use to find the x and y coordinates of each of the 20 hand points. We also created a list that we will use to store the values of these coordinates. The code that goes in this method is the one we use to find the id and hand landmark information of each hand point. We also put in the code that we will use to circle the hand-point that we want to use.

5.2.2 Volume Control Module

We start reading frame from our webcam, one frame at a time, and then send the frame image to our find HandLandMarks(). then extract the x and y coordinates of the tips of the thumb and index finger

The [4] means we are referring to the 4th landmark which is the tip of our thumb. The [1] means we want the x coordinate which resides at the 1st index of our response returned by findHandLandMarks(). You can figure out the rest by yourself.

We then calculate the length of the line which joins landmark 4 and landmark 8:

When the tips of our index finger and thumb are touching each other, there would be a value of length. When you open your index finger and thumb wide there would be another value of length.

So the next line of code converts our length(L1, L2) to proportionate for our volume levels(V1, V2) Once you obtain the correct volume value, you just use it to set your system volume.

The remaining part is just showing the tips of the index finger and thumb, drawing a line between those two points for a better visualization

DESCRIPTION

There are 2 different modules and each module have their own functionalities. At first we need a built in cam or web-cam. to track the hand gesture After installing a web cam we will run our code which will open the web cam of our system. Once after opening the web cam the user will show their hand for tracking the hand.

After showing the hand gesture the web cam starts to track the hand gesture for controlling the volume. After the hand has been tracked the user shows a particular motion to increase or decrease the volume Seeing the input given by the user the volume will be controlled according to the input given by the user.

Hand tracking is the process in which a computer uses computer vision to detect a hand efrom an input image and keeps focus on the hand's movement and orientation. Hand tracking allows us to develop numerous programs that use hand movement and orientation as their input. We tend to write the same code in different projects to perform hand tracking as part of our program.

Creating a hand tracking module solves this problem since we write the code once. We then convert this piece of code into a module. We can import this module into any python project that we are working on and it will perform hand tracking.

5.3 IMPLEMENTATION

Implementation is the stage of a project where the theoretical design is translated into a working system, giving users confidence in the new system's efficiency and effectiveness. It entails meticulous planning, research of the current system and its implementation restrictions, creation of changeover methods, and evaluation of change over methods.

CHAPTER-6

SYSTEM TESTING

6.1 TESTING

6.1.1 UNIT TESTING:

Unit tests are typically automated tests written and run by software developers to ensure that a section of an application (known as the "unit") meets its design and behaves as intended.[2] In procedural programming, a unit could be an entire module, but it is more commonly an individual function or procedure. In object-oriented programming, a unit is often an entire interface, such as a class, or an individual method.[3] By writing tests first for the smallest testable units, then the compound behaviors between those, one can build up comprehensive tests for complex applications.[2]

During development, a software developer may code criteria, or results that are known to be good, into the test to verify the unit's correctness. During test case execution, frameworks log tests that fail any criterion and report them in a summary. For this, the most commonly used approach is test - function - expected value. Writing and maintaining unit tests can be made faster by using parameterized tests. These allow the execution of one test multiple times with different input sets, thus reducing test code duplication. Unlike traditional unit tests, which are usually closed methods and test invariant conditions, parameterized tests take any set of parameters. Parameterized tests are supported by Testing, JUnit and its .Net counterpart, XUnit.

Advantages

The goal of unit testing is to isolate each part of the program and show that the individual parts are correct. A unit test provides a strict, written contract that the piece of code must satisfy. As a result, it affords several benefits.

Unit testing finds problems early in the development cycle. This includes both bugs in the programmer's implementation and flaws or missing parts of the specification for the unit. The process of writing a thorough set of tests forces the author to think through inputs, outputs, and error conditions, and thus more crisply define the unit's desired behavior. The cost of finding a bug before coding begins or when the code is first written is considerably lower than the cost of detecting, identifying, and correcting the bug later.

Limitations and disadvantages

Testing will not catch every error in the program, because it cannot evaluate every execution path in any but the most trivial programs. This problem is a superset of the halting problem, which is undecidable. The same is true for unit testing. Additionally, unit testing by definition only tests the functionality of the units themselves. Therefore, it will not catch integration errors or broader system-level errors (such as functions performed across multiple units, or non-functional test areas such as performance). Unit testing should be done in conjunction with other software testing activities, as they can only show the presence or absence of particular errors; they cannot prove a complete absence of errors. To guarantee correct behavior for every execution path and every possible input, and ensure the absence of errors, other techniques are required, namely the application of formal methods to proving that a software component has no unexpected behavior.

CHAPTER-7

CONCLUSION

7.1 CONCLUSION

The main objective of the project is to control the volume control functions by using the hand gestures instead of using a physical mouse. The proposed system can be achieved by using a webcam or a built-in camera which detects the hand gestures and hand tip and processes these frames to perform the volume control functions.

7.2 FUTURE ENHANCEMENTS

- For the future enhancement we will bring up new features by using the same Hand Gesture control in various forms.
- The changes to be made will be done briefly in future.

CHAPTER-8

APPENDIX

8.1 SAMPLE CODING

HAND TRACKING CODE:

```
import cv2

import mediapipe as mp

import time

class handDetector():

    def __init__(self, mode=False, maxHands=2, detectionCon=0.5,
trackCon=0.5):

        self.mode = mode

        self.maxHands = maxHands

        self.detectionCon = detectionCon

        self.trackCon = trackCon

        self.mpHands = mp.solutions.hands

        self.hands = self.mpHands.Hands(self.mode, self.maxHands, self.mode,
self.detectionCon, self.trackCon)

        self.mpDraw = mp.solutions.drawing_utils

    def findHands(self, img, draw=True):
```

```
lmList.append([id, cx, cy])
```

```
if draw:
```

```
cv2.circle(img, (cx, cy), 15, (255, 0, 255), cv2.FILLED)
```

```
return lmList
```

```
def main():
```

```
    pTime = 0
```

```
    cTime = 0
```

```
    cap = cv2.VideoCapture(1)
```

```
    detector = handDetector()
```

```
    while True:
```

```
        success, img = cap.read()
```

```
        img = detector.findHands(img)
```

```
        lmList = detector.findPosition(img)
```

```
        if len(lmList) != 0:
```

```
            print(lmList[4])
```

```
            cTime = time.time()
```

```
            fps = 1 / (cTime - pTime)
```

```
            pTime = cTime
```

```
cv2.putText(img, str(int(fps)), (10, 70), cv2.FONT_HERSHEY_PLAIN, 3,
```

```
(255, 0, 255), 3)
```

```
cv2.imshow("Image", img)
```

```
cv2.waitKey(1)
```

```
if __name__ == "__main__":
```

```
    main()
```

VOLUME CONTROL CODE:

```
import cv2
```

```
import mediapipe as mp
```

```
from math import hypot
```

```
from ctypes import cast, POINTER
```

```
from comtypes import CLSCTX_ALL
```

```
from pyaw.pycaw import AudioUtilities,
```

```
IAudioEndpointVolume
```

```
import numpy as np
```

```
cap = cv2.VideoCapture(0)
```

```
mpHands = mp.solutions.hands
```

```
hands = mpHands.Hands()
```

```
mpDraw = mp.solutions.drawing_utils
```

```
devices = AudioUtilities.GetSpeakers()
```

```
cv2.circle(img,(x2,y2),4,(255,0,0),cv2.FILLED)
```

```
cv2.line(img,(x1,y1),(x2,y2),(255,0,0),3)
```

```
length = hypot(x2-x1,y2-y1)
```

```
vol = np.interp(length,[15,220],[volMin,volMax])
```

```
print(vol,length)
```

```
volume.SetMasterVolumeLevel(vol, None)
```

```
# Hand range 15 - 220
```

```
# Volume range -63.5 - 0.0
```

```
cv2.imshow('Image',img)
```

```
if cv2.waitKey(1) & 0xff==ord('q'):
```

```
    break
```

8.2 SCREENSHOTS

8.2.1 WEB ACCESS



FIGURE 8.2.1-WEB ACCESS

8.2.2 HANDTRACKING



FIGURE 8.2.2- HAND TRACKING

8.2.3 VOLUME GETTING DECREASED

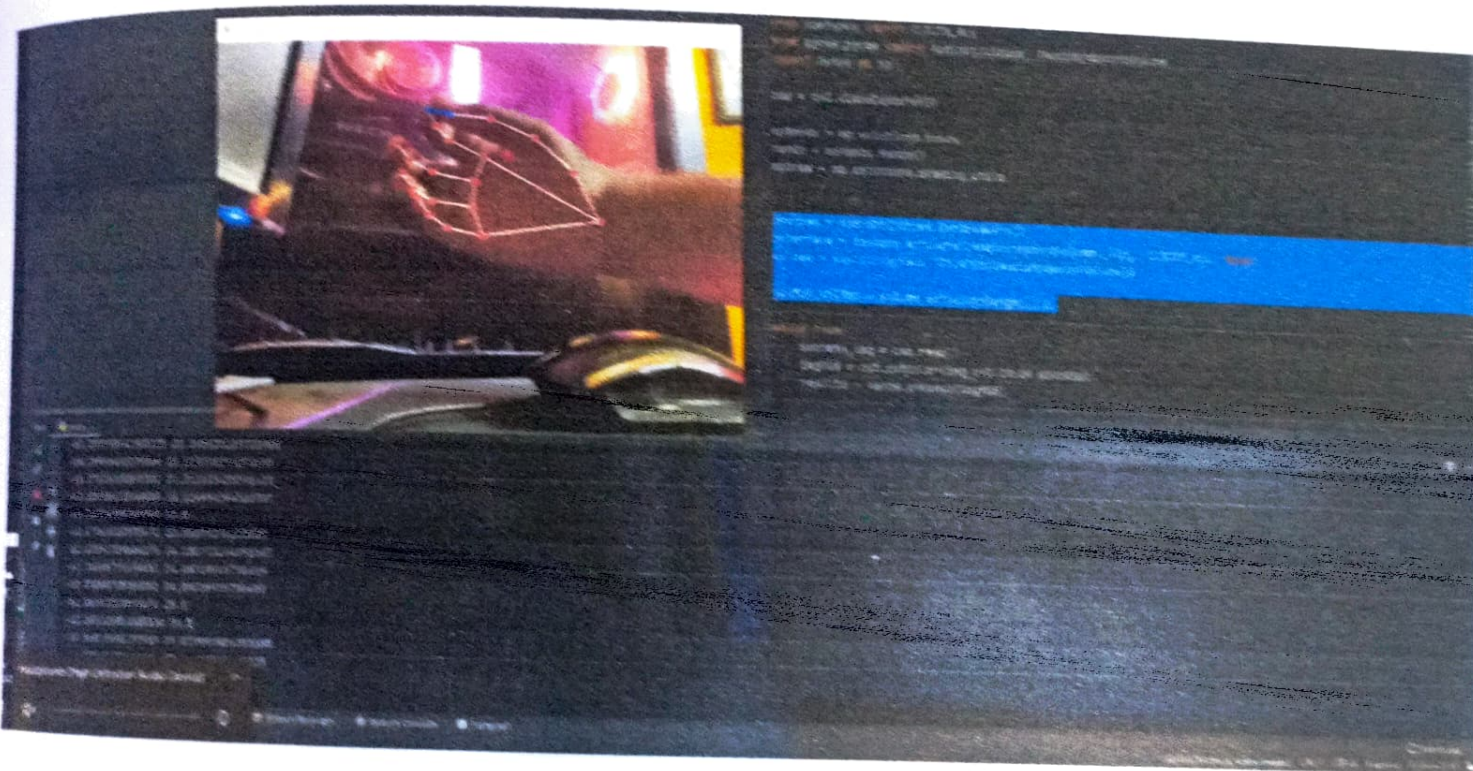


FIGURE 8.2.3-VOLUME GETTING DECREASED

8.2.4 VOLUME GETTING INCREASED

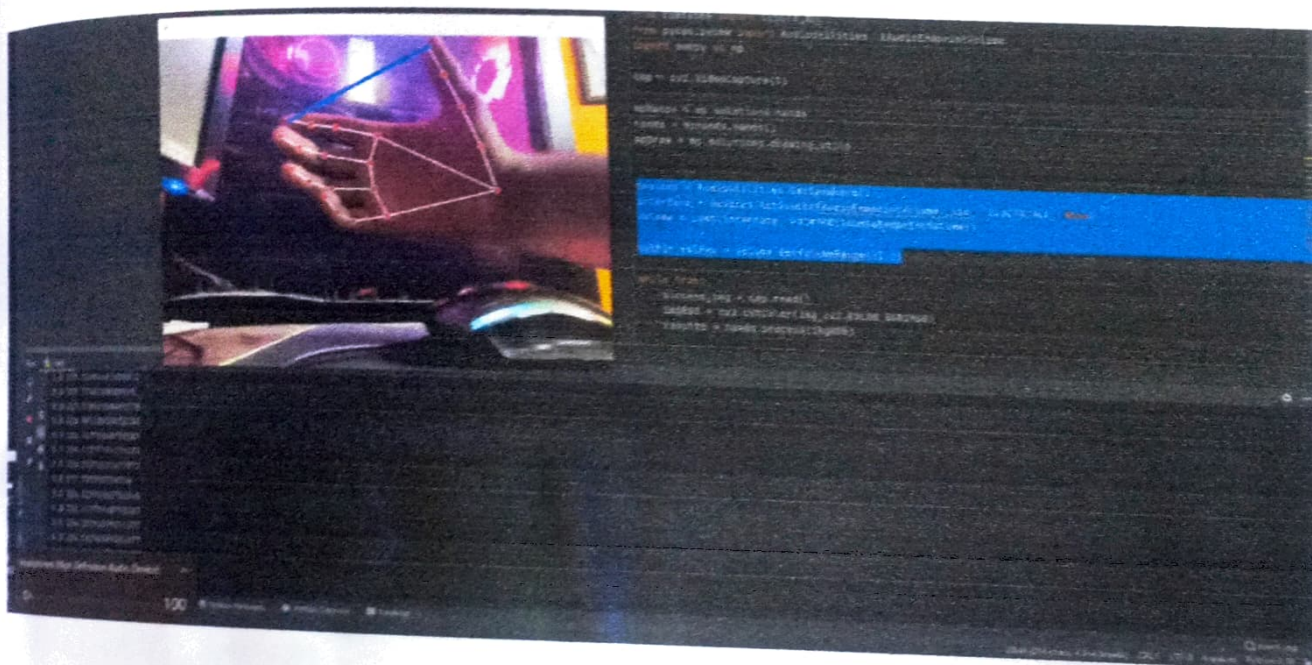


FIGURE 8.2.4- VOLUME GETTING INCREASE

BIOGRAPHY AND REFERENCE

9.1 BIBILOGRAPHY

- [en.wikipedia.org/wiki/Python_\(programming_language\)](https://en.wikipedia.org/wiki/Python_(programming_language))

9.2 REFERENCES

- S. Sadhana Rao," Sixth Sense Technology", Proceedings of the International Conference on Communication and Computational Intelligence-- 2010, pp.336-339.
- Game P. M., Mahajan A.R,"A gestural user interface to Interact with computer system ",
- International Journal of Latest Trends in Engineering and Technology Vol. (7)Issue(4), pp.055-062.
- Imperial Journal of Interdisciplinary Research (IJIR) Vol-3, Issue-4, 2017.
- Christy, A., Vaithyasubramanian, S., Mary, V.A., Naveen Renold, J. (2019)," Artificial intelligence based automatic decelerating vehicle control system to avoid misfortunes International Journal of Advanced Trends in Computer Science and Engineering, Vol. 8, Issue.6, Pp. 3129-3134.