

VIRTUAL KEYBOARD USING PYTHON OPENCV

Submitted in partial fulfilment of the required for the award of

Bachelor of Computer Science

by

**ARUN KUMAR R
(REGISTER NO 40290013)**



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 600 199

MAY 2023



SATHYABAMA

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

Accredited "A" Grade by NAAC | 12B Status by UGC I 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 **ARUN KUMAR R (40290013)** who carried out the project entitled **“VIRTUAL KEYBOARD USING PYTHON OPENCV”** under our supervision from JANUARY 2023 TO MAY 2023.

Internal Guide

Ms .YUVASNEHA.S M.Sc.,ASSISTANT PROFESSOR

DEAN AND HEAD OF THE DEPARTMENT

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

Submitted for Viva voce examination held on 08.05.2023

Internal Examiner

External Examiner

DECLARATION

I am, **ARUN KUMAR R (40290013)** here by declare that the Project Report entitled “**VIRTUAL KEYBOARD USING PYTHON OPENCV**” done by us under the guidance of **Ms. YUVASNEHA S , M.Sc., Assistant Professor,** Department of Computer Science at **SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY,** Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai-600 119 is submitted in partial fulfillment of the requirements for the award of Bachelor of Science degree in Computer Science.



DATE:

ARUN KUMAR R

PLACE: CHENNAI

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to 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 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 guide **Ms. YUVASNEHA.S, M.Sc., Assistant Professor,** Department of Computer Science for their valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

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



ARUN KUMAR R

TABLE CONTENT

CHAPTER NO.	TITLE	PAGE NO.
1	INTRODUCTION	
	1.1 OBJECTIVE OF THE PROJRCT	4
	1.2 PROJECT INTRODUCTION	4
	1.3 LITERATURE SURVEY	6
2	SYSTEM ANALYSIS	
	2.1 FEASIBILITY STUDY	9
	2.2 TYPES OF FEASIBILITY	9
	2.3 EXISTING SYSTEM	9
	2.4 PROPOSED SYSTEM	9
3	SYSTEM CONFIGURATION	
	3.1 HARDWARE REQUIREMENTS	10
	3.2 SOFTWARE REQUIREMENTS	10
	3.3 REQUIRE PACKAGES	10
	3.4 PACKAGES EXPLANATION	10
	3.5 PYTHON INTRODUCTION	18
	3.6 MACHINE LEARNING	20
	3.7 MACHNE LEARNING TYPES	21
	3.8 COMPUTER VISION TECHNIQUES	25
	3.9 IMAGE PROCESSING	27
	3.10 PYCHARM	28
4	SYSTEM DESIGN	
	4.1 ARCHITECTURE DIAGRAM	31
	4.2 SEQUENTIAL DIAGRAM	31

	4.3 WORKING MODEL	32
5	IMPLEMENTATION	
	5.1 HAAR CASCADE ALGORITHM	33
	5.2 MODULE DESCRIPTION	35
6	RESULT AND DISCUSSION	36
7	CONCLUSION ANF FUTURE WORK	37
8	SOURCE CODE AND OUTPUT	38
9	REFERENCE	53

ABSTRACT

Since the development of computer technology, human interaction with computers has evolved. Gestures are an effective method to interact also the era of Covid-19 impacted on us. The keyboard devices that interact with computers. Here we have tried to make the keyboard feature interaction using hand gestures. Eventually discarding the electronic equipment. Hence, the typing on a virtual keyboard. The typing data will be carried out using various hand gestures. The IOT device needed to achieve this is a webcam. The camera's output will be presented on the system's screen so that the user can further calibrate it. We use technologies like Open-CV, Media-Pipe, PyAutoGUI, Python. The Media-Pipe library comes in very handy in AI projects and provides features that help the model's efficiency. The user will be able to type using the virtual keyboard. In this paper, we propose a hand gesture recognition system to typing on the virtual mouse and a virtual keyboard for natural human computer interface.

CHAPTER 1

1.1 OBJECTIVE OF THE PROJECT

The objective for creating a virtual keyboard using OpenCV and Python is to develop a computer vision-based system that allows users to input text and commands using an on-screen keyboard. The virtual keyboard should be capable of recognizing user input from a camera feed, tracking the user's hand movements, and mapping those movements to specific keys on the keyboard. The system should also be robust, accurate, and efficient, enabling users to enter text and commands with ease and speed. Ultimately, the objective is to create a reliable and intuitive input method that can be used in a variety of applications, including gaming, accessibility, and human-computer interaction..

1.2 PROJECT INTRODUCTION

A virtual keyboard is a computer-based input device that allows users to enter text and other characters without the need for a physical keyboard. It is a software-based solution that can be used on any device with a display, such as a computer, tablet, or smartphone. The virtual keyboard is particularly useful for touch screen devices, where it can be difficult to type using a physical keyboard.

In this project, we aim to create a virtual keyboard using OpenCV, an open-source computer vision library. OpenCV provides a set of functions and algorithms for image and video processing, making it an ideal tool for creating a virtual keyboard that can recognize hand gestures and movements.

The virtual keyboard will use the camera on the device to capture images of the user's hand and track its movement in real-time. The hand movements will be analyzed using image processing techniques, and the keyboard will respond accordingly, allowing the user to type and interact with the virtual environment.

The virtual keyboard will consist of a set of buttons, each corresponding to a letter, number, or other character. The user will be able to interact with the keyboard by moving their hand over the buttons and making gestures to select the desired character.

To implement the virtual keyboard, we will first need to detect and track the user's hand in real-time using OpenCV. This will involve using various image processing techniques, such as color segmentation, edge detection, and contour analysis, to extract the hand region from the image.

Once the hand region has been extracted, we will need to identify the location of the fingers and other hand landmarks. This can be done using machine learning algorithms, such as convolutional neural networks, or using a predefined set of rules based on hand anatomy.

With the location of the hand landmarks identified, we can then map each button on the virtual keyboard to a specific hand gesture or movement. For example, the user could select a button by pointing their index finger towards it, or by making a fist and releasing it over the desired button.

To improve the accuracy and performance of the virtual keyboard, we may need to implement additional features, such as gesture recognition, noise reduction, and hand pose estimation. These features will enable the virtual keyboard to detect more complex hand movements and gestures, making it more intuitive and user-friendly.

The virtual keyboard using OpenCV is an exciting project with many potential applications in various industries, such as gaming, virtual and augmented reality, and assistive technology. By creating a virtual keyboard that can recognize hand gestures and movements, we can provide a more natural and intuitive interface for users, particularly those who have difficulty typing using a physical keyboard.

Furthermore, the virtual keyboard project can be extended to support multiple languages and keyboard layouts, making it accessible to a wider audience. Additionally, the project can be integrated with other computer vision solutions, such as face recognition and gaze tracking, to create a more immersive and interactive experience for users.

The virtual keyboard using OpenCV is also a valuable tool for individuals with disabilities or mobility impairments. For example, individuals with conditions such as cerebral palsy or spinal cord injuries may have difficulty using a physical keyboard.

Overall, the virtual keyboard project using OpenCV has significant potential for various applications, from gaming and entertainment to assistive technology and research. The project combines the power of computer vision and image processing with the convenience and accessibility of a virtual keyboard, making it a promising tool for improving human-computer interaction and accessibility.

1.3 LITERATURE SURVEY

REVIEW ON LITERATURE WORK

1.Paper 1: Hand gestures are replacing other computer input systems.

Author Name: Ray, A.

Year of Publish:2022

Abstract:

Computers, over the years, have gone through incredible phases of development. They started with being bulky, heavy, and extremely expensive devices used only for research purposes in the early 1900s. From there, they started getting miniaturised and developed into instruments for personal use in the 1960s. Now they come in different sizes and with varying modes of input (touch or external devices).

However, researchers are constantly developing new methods to make computers even more interactive and operational with minimal physical contact. Using different AI/ML models for this purpose has become a go-to method for all data scientists.

2.Paper 2: Finger Recognition and Gesture based Virtual Keyboard

Author Name : Nikhil, C. D. S., Rao, C. U. S.,
Brumancia, E., Indira, K., Anandhi, T., &Ajitha, P.

Year of Publish : 2020

Abstract :

Hand motion acknowledgment is critical for human-PC connection. Right now, present a novel constant strategy for hand motion recognition. Vision based signal acknowledgment following and motion acknowledgment in our structure, the hand area is separated from the foundation with the foundation subtraction technique. At that point, fingers are portioned in order to identify and perceive the fingers. At long last, a standard classifier is applied to anticipate the names of hand motions. The examinations on the informational index of 1300 pictures show that our strategy performs well and is exceptionally productive. Besides, our technique shows preferred execution over a condition of-workmanship strategy on another informational collection of hand motions.

3.Paper 3 : Virtual Hands: Real Time Keyboard, Desktop & Application

Navigation using Gestures

Author Name : Harshit Sharma, Richa Saxena, Satendra

Kumar, Ashish Saini ,Mohammad Saad, Mohammad Faraz

Year of Publish : 2022

Abstract :

In computer system keyboard is the most prominent input medium of all time. But lately, human community is living in an era of global pandemic being afraid of suffering from Coronavirus (Covid-19) and hence each and every person avoids touching anything. This is because of the fear of contracting this contagious virus and their mutants. So, to mitigate this issue, we present a method "webcam based virtual keyboard interface" to interact with a computer system. The code of this method is written using pre- built modules like OpenCV, MediaPipe, PyAutoGUI etc. and Python 3.9. This approach uses matching the index finger and middle finger on the specific key. After that the virtual desktop switching mechanism is done by PyAutoGUI. The Pygame library plays the sound whenever any key is pressed or when a desktop switch is initiated, corresponding to the key pressed or the desktop switched. In this approach no additional hardware device other than the webcam, that is already available in the system, is required. This approach is also useful for those persons who wants the access the system, even when their hands are dirty.

4.Paper 4: Survey on 3D Hand Gesture Recognition

Author Name : Cheng, H., Yang, L., & Liu, Z.

Year of Publish : 2015

Abstract :

Three-dimensional hand gesture recognition has attracted increasing research interests in computer vision, pattern recognition, and human-computer interaction. The emerging depth sensors greatly inspired various hand gesture recognition approaches and applications, which were severely limited in the 2D domain with conventional cameras. This paper presents a survey of some recent works on hand gesture recognition using 3D depth sensors. We first review the commercial depth sensors and public data sets that are widely used in this field. Then, we review the state-of-the-art research for 3D hand gesture recognition in four aspects: 1) 3D hand modeling; 2) static hand gesture recognition; 3) hand trajectory gesture recognition; and 4) continuous hand gesture

recognition. While the emphasis is on 3D hand gesture recognition approaches, the related applications and typical systems are also briefly summarized for practitioners.

5.Paper 5: Virtual Keyboard Design with Finger Gesture Recognition

Author Name: Tae-Ho Lee & Hyuk-Jae Lee

Year of Publish: 2018

Abstract:

This paper proposes a virtual keyboard that is used to type in characters by recognizing finger gestures with a low-cost camera. To increase the type-in speed, the proposed keyboard takes advantage of the freedom in hand motion. For type-in action with a single hand, the keyboard layer is selected at the same time as the type-in action. When both left and right hands are used for typing, an ambidextrous keyboard layout is proposed. In addition, an efficient type-in action is defined to speed up character input with both hands. An objective function is defined to formulate the type-in time and an algorithm is proposed to design a keyboard layout that minimizes the input time. Simulation results show that the proposed ambidextrous keyboard increases the type-in speed by 56.49% when compared with the right-handed keyboard.

CHAPTER 2

SYSTEM ANALYSIS

2.1 FEASIBILITY STUDY

The feasibility of the project is analysed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

2.2 TYPES OF FEASIBILITY

Economic Feasibility

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus, the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

Technical Feasibility

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

Social Feasibility

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

2.3 EXISTING SYSTEM

The existing system typically uses an old method tracking algorithm to detect the user's handgesture and delay in typing. The existing system have only alphabet and number and does not have any functional keys.

2.4 PROPOSED SYSTEM

The proposed system works in latest hand tracking method and it has functional keys like volume control, play pause, caps lock and keypress sound. This proposed system accuracy of typing is increased.

CHAPTER 3

SYSTEM CONFIGURATION

3.1 Hardware Requirements:

- Laptop or Desktop
- Web cam
- Processor: i3 or above
- Ram: minimum 4gb ram require

3.2 Software Requirements:

- PyCharm latest version
- Python

3.3 Require Packages:

- OpenCV
- Mediapipe
- PyAutoGUI
- Pygame

3.4 PACKAGES EXPLANATION

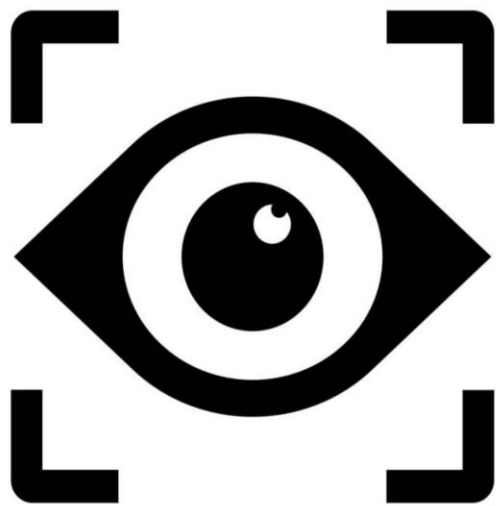
3.4.1 OPENCV

OpenCV stands for Open-Source Computer Vision. To put it simply, it is a library used OpenCV (Open-Source Computer Vision Library) is an open-source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being an Apache 2 licensed product, OpenCV makes it easy for businesses to utilize and modify the code.

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc. OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 18 million. The library is used extensively in companies, research groups and by governmental bodies.

Along with well-established companies like Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota that employ the library, there are many startups such as Applied Minds, VideoSurf, and Zeitera, that make extensive use of OpenCV. OpenCV's deployed uses span the range from stitching street view images together, detecting intrusions in surveillance video in Israel, monitoring mine equipment in China, helping robots navigate and pick up objects at Willow Garage, detection of swimming pool drowning accidents in Europe, running interactive art in Spain and New York, checking runways for debris in Turkey, inspecting labels on products in factories around the world on to rapid face detection in Japan.

It has C++, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS. OpenCV leans mostly towards real-time vision applications and takes advantage of MMX and SSE instructions when available. A full-featured CUDA and OpenCL interfaces are being actively developed right now. There are over 500 algorithms and about 10 times as many functions that compose or support those algorithms. OpenCV is written natively in C++ and has a templated interface that works seamlessly with STL containers.



Features of OpenCV Library

Using OpenCV library, you can

- Read and write images.
- Capture and save videos.
- Process images (filter, transform)
- Perform feature detection.
- Detect specific objects such as faces, eyes, cars, in the videos or images.
- Analyse the video, i.e., estimate the motion in it, subtract the background, and track objects in it.

OpenCV Library Modules

Following are the main library modules of the OpenCV library.

Core Functionality

This module covers the basic data structures such as Scalar, Point, Range, etc., that are used to build OpenCV applications. In addition to these, it also includes the multidimensional array Mat, which is used to store the images. In the Java library of OpenCV, this module is included as a package with the name `org.opencv.core`.

Video

This module covers the video analysis concepts such as motion estimation, background subtraction, and object tracking. In the Java library of OpenCV, this module is included as a package with the name `org.opencv.video`.

Video I/O

This module explains the video capturing and video codecs using OpenCV library. In the Java library of OpenCV, this module is included as a package with the name `org.opencv.videoio`.

Calib3d

This module includes algorithms regarding basic multiple-view geometry algorithms, single and stereo camera calibration, object pose estimation, stereo correspondence, and elements of 3D reconstruction. In the Java library of OpenCV, this module is included as a package with the name `org.opencv.calib3d`.

features2d

This module includes the concepts of feature detection and description. In the Java library of OpenCV, this module is included as a package with the name `org.opencv.features2d`.

Object detect

This module includes the detection of objects and instances of the predefined classes such as faces, eyes, mugs, people, cars, etc. In the Java library of OpenCV, this module is included as a package with the name `org.opencv.objdetect`.

Highgui

This is an easy-to-use interface with simple UI capabilities. In the Java library of OpenCV, the feature of this module is included in two different packages namely, `org.opencv.imgcodecs` and `org.opencv.videoio`.

3.4.2 Mediapipe

MediaPipe Hand Landmarks is a computer vision solution provided by Google's MediaPipe framework for detecting and tracking hand landmarks in real-time. It provides a set of 21 3D landmarks that represent various points on a hand's surface, including fingertips, knuckles, wrist, and palm.

The MediaPipe Hand Landmarks model uses a deep neural network (DNN) architecture, specifically a convolutional neural network (CNN), to accurately detect and track hand landmarks. It takes as input an image or video frame containing a human hand and outputs the coordinates of the 21 hand landmarks.

The hand landmarks are numbered from 0 to 20, where each number represents a different point on the hand's surface. Landmark 0 represents the wrist, while landmarks 1-4 represent the tips of the thumb and four fingers, respectively. Landmarks 5-8 represent the knuckles of the four fingers, and landmarks 9-12 represent the joints between the fingers and the palm. Landmarks 13-16 represent the joints between the fingers, and landmarks 17-20 represent the tips of the four fingers.

The MediaPipe Hand Landmarks model works by first detecting the hand in the input image or video frame using a separate hand detection model. Once the hand is detected, the hand landmarks model is applied to the detected hand region to identify the 21 landmarks. The model

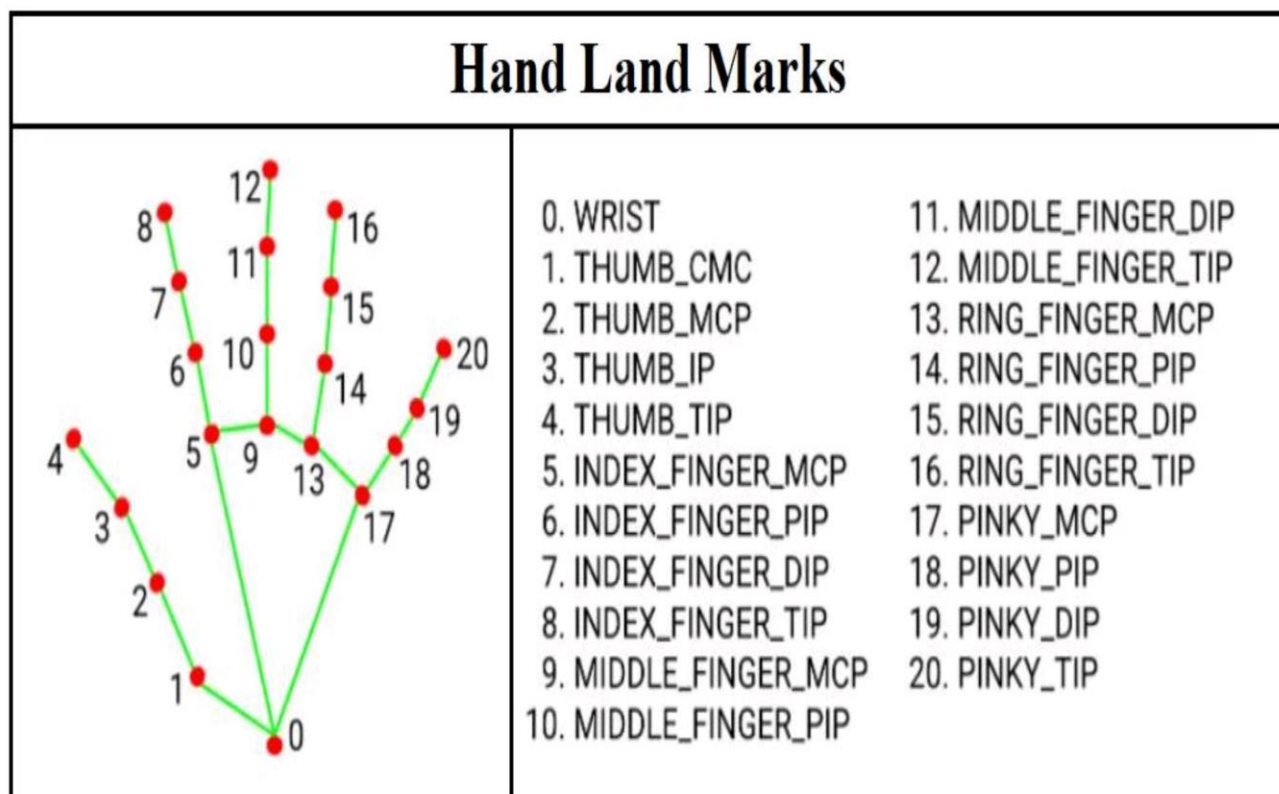
uses a heatmap regression technique to predict the coordinates of each landmark, where each landmark is represented by a peak in the heatmap.

The model is trained on a large dataset of hand images and videos, using a combination of supervised and unsupervised learning techniques. It is capable of detecting and tracking hand landmarks with high accuracy in real-time, even when the hand is in motion or partially occluded.

The MediaPipe Hand Landmarks model has a wide range of applications, including hand gesture recognition, hand pose estimation, hand tracking, and sign language recognition. It can be used in various industries, such as gaming, virtual and augmented reality, robotics, and healthcare.

In addition to the hand landmarks model, MediaPipe also provides a hand tracking model that can track the movement of the hand over time. The hand tracking model uses the output of the hand landmarks model to track the hand's position, orientation, and velocity in 3D space. This allows for more complex hand interactions, such as grasping and manipulating objects, to be detected and tracked in real-time.

Overall, MediaPipe Hand Landmarks is a powerful computer vision solution that provides accurate and efficient detection and tracking of hand landmarks in real-time. Its applications are broad, and its potential impact on various industries is significant.



3.4.3 PyAutoGUI

PyAutoGUI is a cross-platform Python module that enables users to automate mouse movements, keyboard inputs, and other GUI-related tasks on their computer. The module can simulate human input and interactions, allowing users to automate repetitive tasks or perform actions that would otherwise be difficult or time-consuming.

With PyAutoGUI, users can automate tasks such as opening applications, filling out forms, navigating menus, and clicking buttons. The module can also be used to capture and manipulate screenshots, allowing users to automate tasks that involve visual feedback.

PyAutoGUI works by using the Python programming language to control the mouse and keyboard inputs. The module provides functions and methods for moving the mouse cursor, clicking on specific locations, and typing keyboard inputs. Users can also control the speed and duration of mouse and keyboard inputs, allowing for precise and controlled automation.

One of the key features of PyAutoGUI is its cross-platform compatibility. The module can run on Windows, macOS, and Linux, and it supports both Python 2 and Python 3. This makes it a versatile tool for automating tasks on different operating systems and environments.

PyAutoGUI also provides support for hotkeys and keyboard shortcuts, allowing users to automate tasks that involve multiple keyboard inputs. The module can also send virtual keypresses, allowing users to simulate keypresses for keys that are not present on their physical keyboard.

In addition to automating tasks, PyAutoGUI can also be used for testing and debugging software applications. The module can simulate user inputs and interactions, allowing developers to test their applications under different scenarios and conditions.

Overall, PyAutoGUI is a powerful and versatile module for automating GUI-related tasks in Python. With its cross-platform compatibility, support for hotkeys and virtual keypresses, and extensive documentation, PyAutoGUI is an essential tool for anyone looking to automate repetitive or complex tasks on their computer.

Moreover, PyAutoGUI can be used in combination with other Python libraries and frameworks to build more advanced automation workflows. For example, PyAutoGUI can be used with the Selenium WebDriver library to automate web browser interactions and test web applications.

PyAutoGUI can also be used with other automation frameworks, such as Robot Framework and Appium, to automate tasks on different platforms and devices. This makes PyAutoGUI a versatile tool for automating tasks in various domains, such as software development, testing, data entry, and system administration.

Another useful feature of PyAutoGUI is its ability to perform image recognition and manipulation. The module can locate and click on specific pixels or images on the screen, making it useful for automating tasks that involve visual feedback or complex user interfaces.

However, it is important to note that PyAutoGUI should be used with caution when automating tasks, especially when running on production systems or critical infrastructure. Improper use of the module can result in unintended consequences, such as data loss, system crashes, or security breaches.

In summary, PyAutoGUI is a powerful and flexible Python module for automating GUI-related tasks on different operating systems and environments. With its cross-platform compatibility, support for hotkeys and virtual keypresses, and extensive documentation, PyAutoGUI is an essential tool for anyone looking to automate repetitive or complex tasks on their computer.

3.4.4 Pygame

Pygame is a popular Python library that enables users to create games and multimedia applications. The library provides a range of modules and functions for handling graphics, sound, input, and other aspects of game development.

With Pygame, users can create 2D and 3D games, animations, and interactive applications. The library provides functions for drawing graphics, playing sound effects and music, handling user input from a variety of devices, and managing game states and events.

One of the key features of Pygame is its ease of use and simplicity. The library provides a straightforward interface that allows users to quickly get started with game development, without requiring advanced knowledge of programming or graphics.

Pygame also provides extensive documentation and examples, making it easy for users to learn how to use the library and build their own games and applications. Additionally, Pygame has an active

community of developers and users who contribute to the library and provide support for beginners and advanced users alike.

Pygame is cross-platform and can run on Windows, macOS, and Linux, making it accessible to a wide range of users. The library is also compatible with Python 2 and Python 3, providing flexibility for users who prefer one version of the language over the other.

Pygame can be used for a wide range of game development projects, from simple arcade-style games to complex role-playing games (RPGs) and simulations. The library can also be used for non-game applications, such as multimedia presentations and interactive installations.

Furthermore, Pygame can be extended and customized with third-party modules and libraries, allowing users to add new features and functionality to their games and applications. For example, users can use the PyOpenGL library to add 3D graphics to their games or the Pygame GUI library to create custom user interfaces.

In conclusion, Pygame is a powerful and versatile Python library for game development and multimedia applications. With its ease of use, cross-platform compatibility, and extensive documentation, Pygame is an excellent choice for beginners and experienced developers alike who want to create fun and engaging games and applications.

3.5 PYTHON INTRODUCTION

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics developed by Guido van Rossum. It was originally released in 1991. Designed to be easy as well as fun, the name "Python" is a nod to the British comedy group Monty Python. Python has a reputation as a beginner-friendly language, replacing Java as the most widely used introductory language because it handles much of the complexity for the user, allowing beginners to focus on fully grasping programming concepts rather than minute.

Python is used for server-side web development, software development, mathematics, and system scripting, and is popular for Rapid Application Development and as a scripting or glue language to tie existing components because of its high-level, built-in data structures, dynamic typing, and dynamic binding. Program maintenance costs are reduced with Python due to the easily learned syntax and emphasis on readability. Additionally, Python's support of modules and packages facilitates modular programs and reuse of code. Python is open-source community language so numerous independent programmers are continually building libraries.

Python Use Cases

- Creating web applications on a server
- Building workflows that can be used in conjunction with software
- Connecting to database systems
- Reading and modifying files
- Performing complex mathematics
- Processing BIG DATA
- Fast prototyping
- Developing production-ready software

Professionally, Python is great for backend web development, data analysis, artificial intelligence, and scientific computing. Developers also use Python to build productivity tools, game and desktop apps.

Features and Benefits of Python

- Compatible with a variety of platforms including Windows, Mac, Linux, Raspberry Pi, and others
- Uses a simple syntax comparable to the English language that lets developers use fewer lines than other programming languages
- Operates on an interpreter system that allows code to be executed immediately, fast-tracking prototyping
- Can be handled in a procedural, object-orientated, or functional way

Python Syntax

- Somewhat similar to the English language, with a mathematical influence, Python is built for readability
- Unlike other languages that use semicolons and/or parentheses to complete a command, Python uses new lines for the same function
- Defines scope (i.e., loops, functions, classes) by relying indentation, using whitespace.

Python Flexibility

Python, a dynamically typed language, is especially flexible, eliminating hard rules for

building features and offering more problem-solving flexibility with a variety of methods. It also allows users to compile and run programs right up to a problematic area because it uses run-time type checking rather than compile-time checking.

The Less Great Parts of Python

On the down side, Python isn't easy to maintain. One command can have multiple meanings depending on context because Python is a dynamically typed language. And, maintaining a Python app as it grows in size and complexity can be increasingly difficult, especially finding and fixing errors. Users will need experience to design code or write unit tests that make maintenance easier.

Speed is another weakness in Python. Its flexibility, because it is dynamically typed, requires a significant amount of referencing to land on a correct definition, slowing performance. This can be mitigated by using alternative implementation of Python (e.g., PyPy).

Python and AI

AI researchers are fans of Python. Google TensorFlow, as well as other libraries (scikit-learn, Keras), establish a foundation for AI development because of the usability and flexibility it offers Python users. These libraries, and their availability, are critical because they enable developers to focus on growth and building.

Good to Know

The Python Package Index (PyPI) is a repository of software for the Python programming language. PyPI helps users find and install software developed and shared by the Python community.

3.6 Machine learning

Machine learning is a growing technology which enables computers to learn automatically from past data. Machine learning uses various algorithms for building mathematical models and making predictions using historical data or information. Currently, it is being used for various tasks such as image recognition, speech recognition, email filtering, Facebook auto-tagging, recommender system, and many more.

Machine Learning is said as a subset of artificial intelligence that is mainly concerned with the

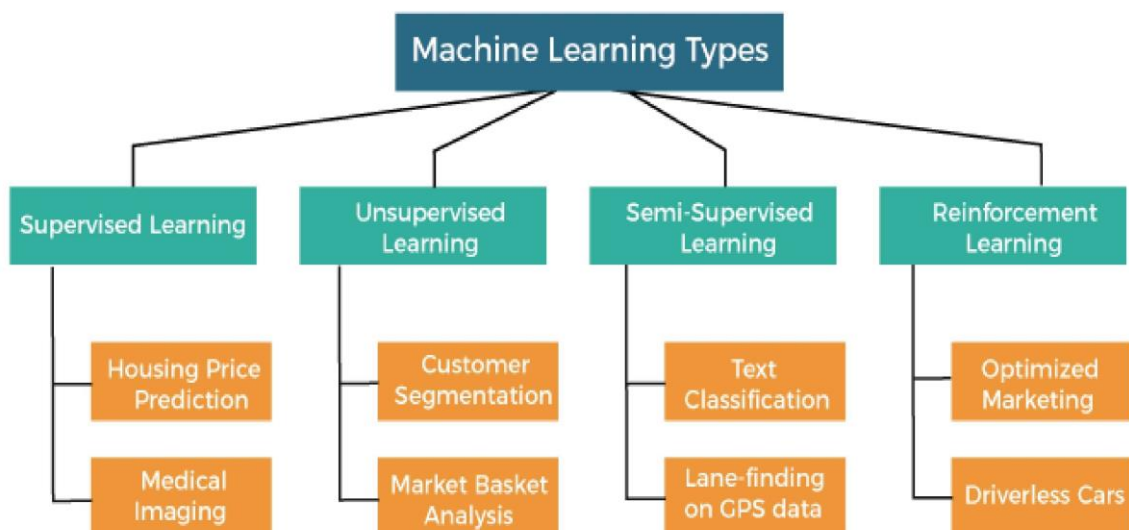
development of algorithms which allow a computer to learn from the data and past experiences on their own. The term machine learning was first introduced by Arthur Samuel in 1959. We can define it in a summarized way as: “Machine learning enables a machine to automatically learn from data, improve performance from experiences, and predict things without being explicitly programmed”. Suppose we have a complex problem, where we need to perform some predictions, so instead of writing a code for it, we just need to feed the data to generic algorithms, and with the help of these algorithms, machine builds the logic as per the data and predict the output. Machine learning has changed our way of thinking about the problem

3.7 Types of Machine learning

These ML algorithms help to solve different business problems like Regression, Classification, Forecasting, Clustering, and Associations, etc.

Based on the methods and way of learning, machine learning is divided into mainly four types, which are:

1. **Supervised Machine Learning**
2. **Unsupervised Machine Learning**
3. **Semi-Supervised Machine Learning**
4. **Reinforcement Learning**



1. Supervised Machine Learning

As its name suggests, Supervised machine learning is based on supervision. It means in the supervised learning technique, we train the machines using the "labelled" dataset, and based on the training, the machine predicts the output. Here, the labelled data specifies that some of the inputs are already mapped to the output. More precisely, we can say; first, we train the machine with the input and corresponding output, and then we ask the machine to predict the output using the test dataset.

Let's understand supervised learning with an example. Suppose we have an input dataset of cats and dog images. So, first, we will provide the training to the machine to understand the images, such as the **shape & size of the tail of cat and dog, Shape of eyes, colour, height (dogs are taller, cats are smaller), etc.** After completion of training, we input the picture of a cat and ask the machine to identify the object and predict the output. Now, the machine is well trained, so it will check all the features of the object, such as height, shape, colour, eyes, ears, tail, etc., and find that it's a cat. So, it will put it in the Cat category. This is the process of how the machine identifies the objects in Supervised Learning.

The main goal of the supervised learning technique is to map the input variable(x) with the output variable(y). Some real-world applications of supervised learning are Risk Assessment, Fraud Detection, Spam filtering, etc.

Categories of Supervised Machine Learning

Supervised machine learning can be classified into two types of problems, which are given below:

- **Classification**
- **Regression**

2. Unsupervised Machine Learning

Unsupervised learning is different from the Supervised learning technique; as its name suggests, there is no need for supervision. It means, in unsupervised machine learning, the machine is trained using the unlabelled dataset, and the machine predicts the output without any supervision.

In unsupervised learning, the models are trained with the data that is neither classified nor labelled, and the model acts on that data without any supervision.

The main aim of the unsupervised learning algorithm is to group or categories the unsorted dataset according to the similarities, patterns, and differences. Machines are instructed to find the hidden patterns from the input dataset.

Let's take an example to understand it more preciously; suppose there is a basket of fruit images, and we input it into the machine learning model. The images are totally unknown to the model, and the task of the machine is to find the patterns and categories of the objects.

So, now the machine will discover its patterns and differences, such as colour difference, shape difference, and predict the output when it is tested with the test dataset.

Categories of Unsupervised Machine Learning

Unsupervised Learning can be further classified into two types, which are given below:

- Clustering
- Association

3. Semi-Supervised Learning

Semi-Supervised learning is a type of Machine Learning algorithm that lies between Supervised and Unsupervised machine learning. It represents the intermediate ground between Supervised (With Labelled training data) and Unsupervised learning (with no labelled training data) algorithms and uses the combination of labelled and unlabelled datasets during the training period.

Although Semi-supervised learning is the middle ground between supervised and unsupervised learning and operates on the data that consists of a few labels, it mostly consists of unlabelled data. As labels are costly, but for corporate purposes, they may have few labels. It is completely different from supervised and unsupervised learning as they are based on the presence & absence of labels.

To overcome the drawbacks of supervised learning and unsupervised learning algorithms, the concept of Semi-supervised learning is introduced. The main aim of semi-supervised learning is to effectively use all the available data, rather than only labelled data like in supervised learning. Initially, similar data is clustered along with an unsupervised learning algorithm, and further, it helps to label the unlabelled data into labelled data. It is because labelled data is a comparatively more expensive acquisition than unlabelled data.

We can imagine these algorithms with an example. Supervised learning is where a student is under the supervision of an instructor at home and college. Further, if that student is self-analysing the same concept without any help from the instructor, it comes under unsupervised learning. Under semi-supervised learning, the student has to revise himself after analyzing the same concept under the guidance of an instructor at college.

4. Reinforcement Learning

Reinforcement learning works on a feedback-based process, in which an AI agent (A software component) automatically explore its surrounding by hitting & trail, taking action, learning from experiences, and improving its performance. Agent gets rewarded for each good action and get punished for each bad action; hence the goal of reinforcement learning agent is to maximize the rewards.

In reinforcement learning, there is no labelled data like supervised learning, and agents learn from their experiences only.

The reinforcement learning process is similar to a human being; for example, a child learns various things by experiences in his day-to-day life. An example of reinforcement learning is to play a game, where the Game is the environment, moves of an agent at each step define states, and the goal of the agent is to get a high score. Agent receives feedback in terms of punishment and rewards.

Due to its way of working, reinforcement learning is employed in different fields such as **Game theory, Operation Research, Information theory, multi-agent systems.**

A reinforcement learning problem can be formalized using **Markov Decision Process(MDP)**. In MDP, the agent constantly interacts with the environment and performs actions; at each action, the environment responds and generates a new state.

Categories of Reinforcement Learning

Reinforcement learning is categorized mainly into two types of methods/algorithms:

- **Positive Reinforcement Learning:** Positive reinforcement learning specifies increasing the tendency that the required behaviour would occur again by adding something. It enhances the strength of the behaviour of the agent and positively impacts it.
- **Negative Reinforcement Learning:** Negative reinforcement learning works exactly opposite to the positive RL. It increases the tendency that the specific behaviour would occur again by avoiding the negative condition.

3.8 COMPUTER VISION TECHNIQUES

Computer vision techniques refer to a set of algorithms and methods used to extract information and knowledge from digital images or video data. These techniques are essential for many applications, including object detection, tracking, segmentation, recognition, and analysis. Here are some of the most common computer vision techniques

Image Filtering:

Image filtering refers to a set of methods used to enhance or suppress certain features of an image. For example, low-pass filters can be used to remove high-frequency noise, while high-pass filters can be used to enhance the edges and details in an image.

Edge Detection:

Edge detection refers to a set of techniques used to identify the boundaries between objects in an image. Edge detection methods use mathematical operators, such as the Sobel operator or the Canny edge detector, to locate the edges and boundaries in an image.

Feature Extraction:

Feature extraction refers to a set of methods used to extract meaningful information from an image. For example, features such as HOG, LBP, or SIFT can be used to represent the shape, texture, or colour information of an object in an image.

Object Detection:

Object detection refers to the process of locating and classifying objects in an image. Object detection methods use techniques such as Haar cascades, RCNN, or YOLO to identify the objects and classify them based on their features.

Tracking:

Tracking refers to the process of following an object's motion across multiple frames in a video sequence. Tracking methods use techniques such as KCF, MOSSE, or Particle filter to estimate the object's position and velocity in each frame.

Segmentation:

Segmentation refers to the process of dividing an image into regions or segments based on their properties or features. Segmentation methods use techniques such as thresholding, clustering, or graph-cut to separate the image into meaningful regions. In summary, computer vision techniques play a crucial role in extracting information and knowledge from digital images and video data. These techniques enable many applications, such as object detection, tracking, segmentation, recognition, and analysis.

TRACKING ALGORITHM

The tracking algorithm is responsible for locating the iris in the video frames and estimating its position accurately. The algorithm consists of several components, including pre-processing, feature extraction, filtering, and estimation.

Here's a detailed overview of each component:

Pre-processing:

The input video frames are pre-processed to enhance the iris region's visibility. The frames are converted to the appropriate colour space, such as HSV or BGR, and filtered using techniques such as low-pass filtering or high-pass filtering. This step aims to reduce noise and improve the iris region's contrast, making it easier to identify in the subsequent steps.

Feature extraction:

The iris region is identified using techniques such as thresholding, edge detection, or template matching. Once the iris region is identified, features such as HOG, LBP, or PCA are extracted to train the KCF tracker. These features provide a compact representation of the iris region, which can be used to track the iris's motion across the video frames.

Filtering:

The KCF tracker is trained using the extracted features to estimate the iris's position in the subsequent frames. The KCF tracker uses a machine learning algorithm to learn a correlation filter that correlates with the iris features. This filter is then applied to the subsequent frames to estimate the iris's position. The tracker's output is refined using filtering techniques such as low pass filtering or Kalman filter to improve the position estimates' accuracy.

Estimation:

The resulting position estimates are used to predict the iris's future position using the Kalman filter. The Kalman filter takes into account the motion dynamics of the iris, such as its velocity and acceleration, to refine the position estimates and predict the iris's future position accurately. and machine learning algorithms to locate the iris in the video frames and estimate its position accurately. The algorithm can handle various challenges, such as occlusion, illumination changes, and motion blur, making it suitable for various applications, such as eye-tracking and gaze estimation.

3.9 Image processing

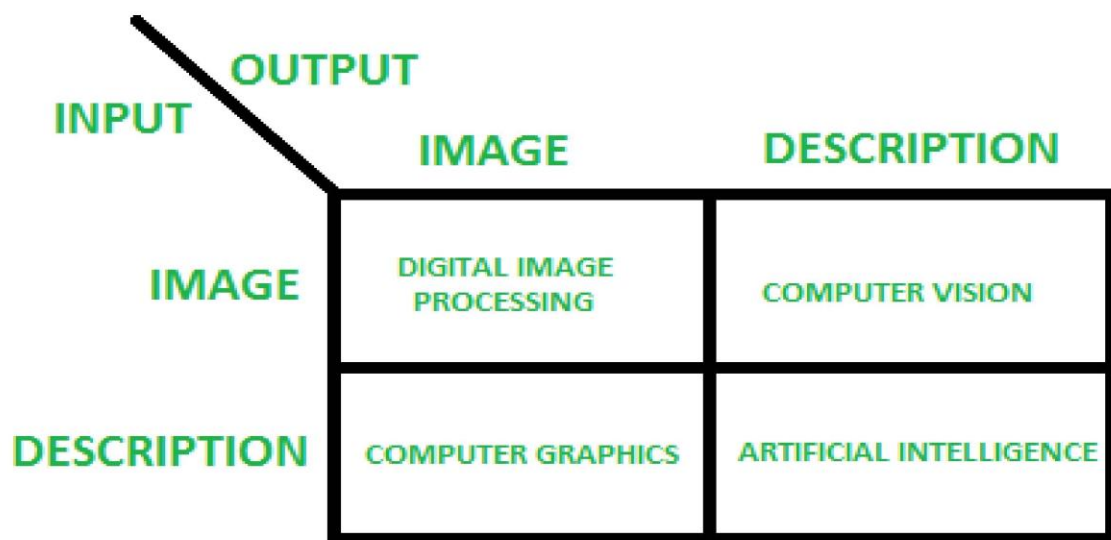
Image processing is the process of transforming an image into a digital form and performing certain operations to get some useful information from it. The image processing system usually treats all images as 2D signals when applying certain predetermined signal processing methods.

Types of Image Processing

There are five main types of image processing:

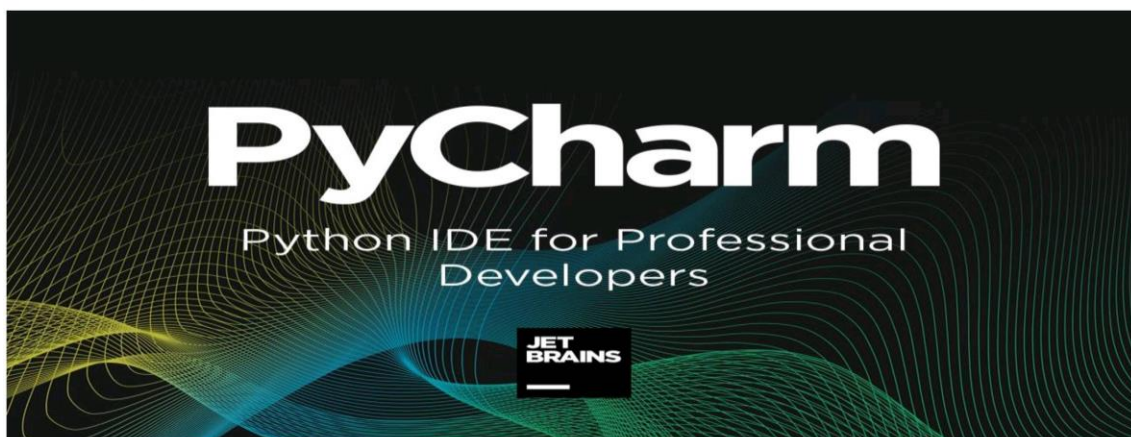
- Visualization - Find objects that are not visible in the image
- Recognition - Distinguish or detect objects in the image
- Sharpening and restoration - Create an enhanced image from the original image

- Pattern recognition - Measure the various patterns around the objects in the image
- Retrieval - Browse and search images from a large database of digital images that are similar to the original image



3.10 PYCHARM

PyCharm is a dedicated Python Integrated Development Environment (IDE) providing a wide range of essential tools for Python developers, tightly integrated to create a convenient environment for productive Python, web, and data science development. It is cross-platform, working on Microsoft Windows, macOS and Linux. PyCharm has a Professional Edition, released under a proprietary license and a Community Edition released under the Apache License. PyCharm Community Edition is less extensive than the Professional Edition.



FEATURES:

1.Intelligent Code Editor

- It helps us write high-quality codes!
- It consists of colour schemes for keywords, classes, and functions. This helps increase the readability and understanding of the code.
- It helps identify errors easily.
- It provides the autocomplete feature and instructions for the completion of the code.

2.Code Navigation:

- It helps developers in editing and enhancing the code with less effort.
- With code navigation, a developer can easily navigate to a function, class, or file.
- A programmer can locate an element, a symbol, or a variable in the source code within no time.

3.Refactoring

- It has the advantage of making efficient and quick changes to both local and global variables.
- Refactoring in PyCharm enables developers to improve the internal structure without changing the external performance of the code.
- It also helps split up more extended classes and functions with the help of the extract method.

4.Assistance for Many Other Web Technologies:

- It helps developers create web applications in Python.
- It supports popular web technologies such as HTML, CSS, and JavaScript.
- Developers have the choice of live editing with this IDE. At the same time, they can preview the created/updated web page.
- The developers can follow the changes directly on a web browser.
- PyCharm also supports AngularJS and NodeJS for developing web applications.

5.Support for Popular Python Web Frameworks

- PyCharm supports web frameworks such as Django.
- It provides the autocomplete feature and suggestions for the parameters of Django.
- It helps in debugging the codes of Django.
- It also assists web2py and Pyramid, the other popular web frameworks.

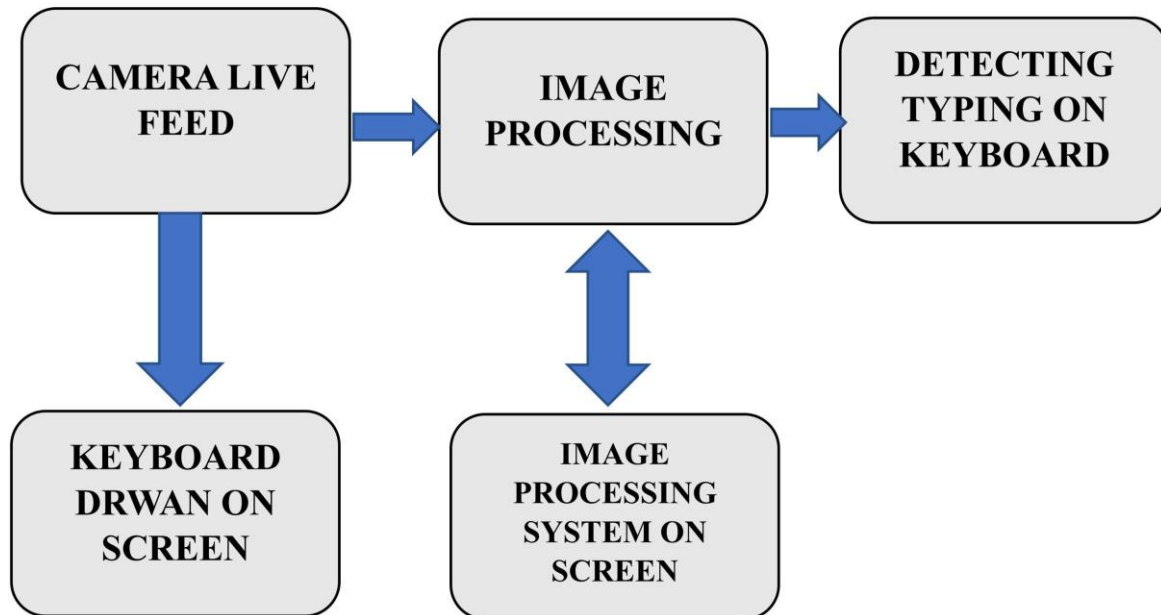
6. Assistance for Python Scientific Libraries

- PyCharm supports Python's scientific libraries such as Matplotlib, NumPy, and Anaconda.
- These scientific libraries help in building projects of Data Science and Machine Learning.
- It consists of interactive graphs that help developers understand data.
- It can integrate with various tools such as IPython, Django, and Pytest. This integration helps innovate unique solutions.

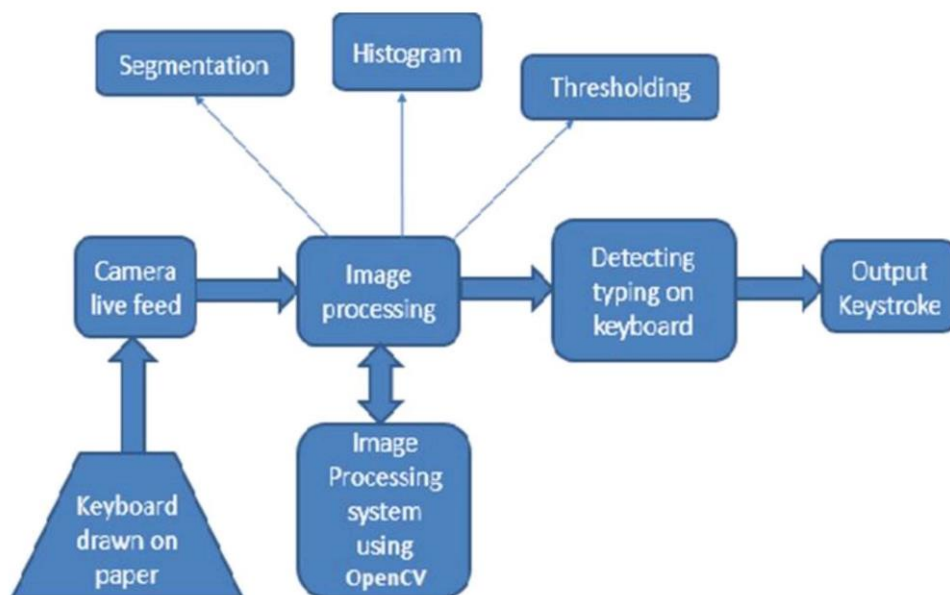
CHAPTER 4

SYSTEM DESIGN

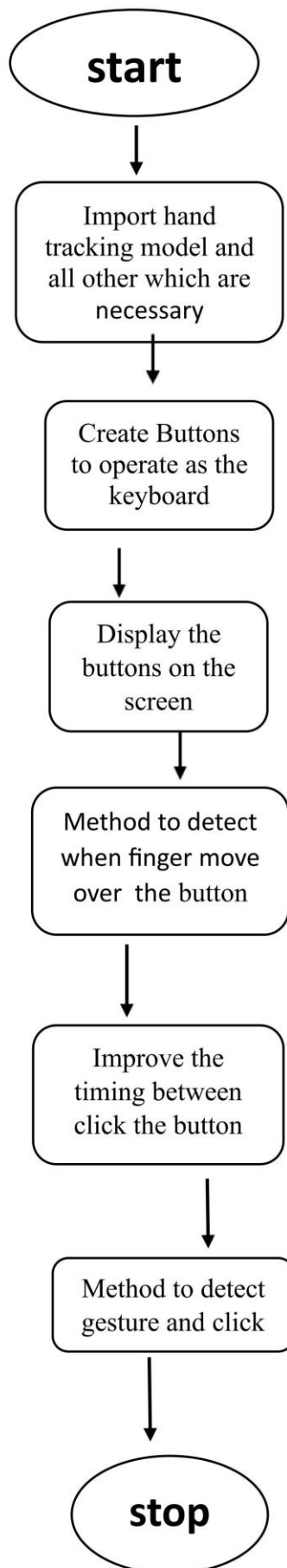
4.1 ARCHITECTURE DIAGRAM:



4.2 SEQUENTIAL DIAGRAM:



4.3 WORKING MODEL:



CHAPTER 5

SYSTEM ALGORITHM

5.1 Haar cascade algorithm

Haar Cascade algorithm can be used for hand detection in computer vision applications. The algorithm works by training a classifier on a set of positive and negative images to detect the presence of a hand in a given image or video stream. Here's how Haar Cascade algorithm can be applied for hand detection:

1. Gathering and labelling images:

The first step in training the Haar Cascade classifier is to gather a set of images containing hands and label them as "positive". This set of images should contain a variety of hand shapes and positions to train the classifier to detect hands under different conditions. Negative images, which do not contain any hands, should also be gathered and labelled as "negative".

2. Preprocessing images:

The gathered images need to be pre-processed to extract features that can be used for training the classifier. In Haar Cascade algorithm, the images are first converted to grayscale and then the Haar wavelet transform is applied to them to extract features. These features are then used to train the classifier.

3. Training the classifier:

The pre-processed positive and negative images are used to train the classifier. The classifier is typically a machine learning algorithm, such as Support Vector Machine (SVM), that is trained to distinguish between positive and negative images based on the extracted features. The training process involves adjusting the parameters of the algorithm until it can accurately distinguish between positive and negative images.

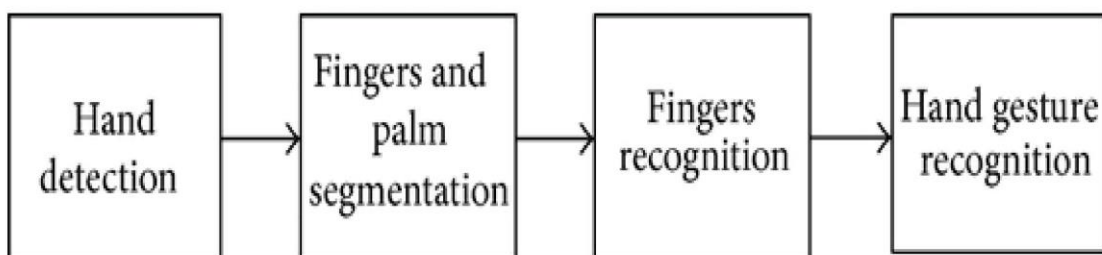
4. Object detection:

Once the classifier is trained, it can be used for object detection on a test image or video stream. The image or video stream is analyzed using a sliding window approach to detect potential hand regions. The classifier is then applied to each potential region to determine if it contains a hand. Non-maximum suppression is used to remove duplicate detections.

5. Limitations and improvements:

Haar Cascade algorithm has some limitations when it comes to hand detection, such as sensitivity to lighting conditions and variations in hand size and orientation. Techniques to improve the accuracy and robustness of the algorithm, such as using multiple classifiers, incorporating other features like Local Binary Patterns (LBP), and applying deep learning techniques can be used to overcome these limitations.

In conclusion, Haar Cascade algorithm can be used for hand detection in computer vision applications. By training a classifier on a set of positive and negative images, the algorithm can detect hands in real-time applications, such as gesture recognition and human-computer interaction. However, the algorithm has limitations and requires careful tuning and preprocessing to achieve optimal performance.



Haar cascade algorithm uses:

Haar cascade is an algorithm that can detect objects in images, irrespective of their scale in image and location. This algorithm is not so complex and can run in real-time. We can train a haar-cascade detector to detect various objects like cars, bikes, buildings, fruits, etc.

5.2 MODEL DESCRIPTION

Module 1

Live Camera: A camera is used to capture the image of the hand gestures made by the user. This image is processed by the computer vision algorithm to recognize the gestures and translate them into keyboard input.

Module 2

Hand Gesture Recognition Module: This module is responsible for recognizing different hand gestures made by the user. The OpenCV library provides several image processing techniques such as background subtraction, skin color detection, contour detection, and convex hull that can be used to recognize different hand gestures.

Module 3

Gesture-to-Text Translation Module: This module is responsible for translating the recognized hand gestures into text. This can be achieved by mapping the gestures to specific keyboard keys or by using machine learning algorithms to recognize different words or phrases.

Module 4

Text Display Module: This module is responsible for displaying the typed text on the screen. It can be implemented using GUI frameworks like PyQt or Tkinter.

Module 5

Keyboard Emulation Module: This module is responsible for simulating the keyboard input based on the recognized hand gestures.

Module 5

Evaluation: Virtual keyboard using a real-time camera as well as hand detection and hand tracking, we have attempted to perform virtual keyboard. Hand gesture recognition module worked smoothly with a minimal latency but we were able to generate our desired output.

CHAPTER 6

RESULT AND DISCUSSION

Typing without touching the keys is fulfilled to ignore obstructions covering the paper keyboard. The experiment results indicate that the overall recognition rate of the proposed virtual keyboard is 94.62%. The proposed virtual keyboard can be used for a smartphone in the future. Furthermore, as a blueprint, it can be applied to computers after it is improved to allow ten-finger recognition. Moreover machine- learning can be potentially embedded into virtual keyboard to greatly improve its figure recognition and performance.

End to end testing took place when webcam has been used for testing whether it's taking the image from outside or not, this was the most crucial step. If user has virtual keyboard printed in the camera, it surely be handled carefully. Without the proper input to the application, it won't result in detecting the alphabets, symbols, number.

1. Our VIRTUAL KEYBOARD is based on working of sensors and actuators.
2. There is no specific need of any kind of knowledge to run this software. It can be easily implemented with the basic idea of rules set to run the program.
3. It is also used in 6th sense technology device in which it is not depends on surface.
4. We are able to access the keyboard layout drawn on a screen aligned and designed for camera window.
5. Further the images are pre-processed before the camera pop up window gets activated which will sharp or blur the image as per the needs of the situation.
6. Then the fingertips of user are detected as soon as there is a contact made with any of the character in the keyboard.

CHAPTER 7

CONCLUSION AND FUTURE WORK

CONCLUSION

We have successfully built and implemented VIRTUAL KEYBOARD using a real-time camera as well as hand detection and tracking, we have attempted to perform virtual keyboard. Hand gesture recognition module worked smoothly with a minimal latency but we were able to generate our desired output. Artificial Intelligence and Digital World is a vast topic and there is a lot we can explore in it.

With the help of sensors, actuators and programming it was possible to bring our hardware keyboard on air and experience something new in the area of computer science and technology with some cool method. The keyboard is simulated as an actual keyboard as well as the alphabets are well aligned. The system designed can we worked in all conditions just requirement is a bright room and hand as well as a HD webcam for fast detection and better result. The system efficiency will help us in saving space at work, reduce plastic waste by eliminating the keyboard. It will also an innovation leading to brighter prospects.

FUTURE WORK

Our future plan includes adding more functionalities such as expanding and reducing windows, closing windows, and so on, by utilizing the palm and multiple fingers. We aim to integrate voice recognition into the keyboard as well. The project's upcoming work will focus on enhancing the Fingertip Detection module to be insensitive to variations in lighting and determining the 3D posture of the panel for the purpose of augmenting 3D objects in reality. Future will be adding more functional hotkeys like ctrl, alt, shift and more functional keys.

CHAPTER 8

SOURCE CODE

```
import cv2

import pyautogui as pg

import mediapipe as mp

import pygame

pygame.init()

pygame.mixer.init()

mc = pygame.mixer.Sound("kc.wav")


fps = 30

window_width = 1280

window_height = 720


w = 1280

h = 720

cam = cv2.VideoCapture(0, cv2.CAP_DSHOW)

cam.set(cv2.CAP_PROP_FRAME_WIDTH, window_width)

cam.set(cv2.CAP_PROP_FRAME_HEIGHT, window_height)

cam.set(cv2.CAP_PROP_FPS, fps)

cam.set(cv2.CAP_PROP_FOURCC, cv2.VideoWriter_fourcc(*'MJPG'))


alph = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V',
```

'W', 'X', 'Y', 'Z']

alph1 = ['~', '1', '2', '3', '4', '5', '6', '7', '8', '9', '0', '-', '=', 'back']

R1x1 = 225

R1x2 = 275

R1y1 = 100 # 10 400

R1y2 = 150 # 60 450

t1x = 232

t1y = 145 # 55 445

alph2 = ['V+', 'Q', 'W', 'E', 'R', 'T', 'Y', 'U', 'I', 'O', 'P', '[', ']', '|']

R2x1 = 240 + 60

R2x2 = 290 + 60

R2y1 = R1y1 + 55

R2y2 = R1y2 + 55

t2x = 247 + 60

t2y = t1y + 50

alph3 = ['caps', 'A', 'S', 'D', 'F', 'G', 'H', 'J', 'K', 'L', ':', "'", 'enter']

R3x1 = 240 + 80

R3x2 = 290 + 80

R3y1 = R2y1 + 55

R3y2 = R2y2 + 55

t3x = 247 + 80

```
t3y = t2y + 50
```

```
alph4 = ['Mute', 'Z', 'X', 'C', 'V', 'B', 'N', 'M', ',', '.', '?', 'playpause']
```

```
R4x1 = 240 + 110
```

```
R4x2 = 290 + 110
```

```
R4y1 = R3y1 + 55
```

```
R4y2 = R3y2 + 55
```

```
t4x = 247 + 110
```

```
t4y = t3y + 55
```

```
mphands = mp.solutions.hands
```

```
hands = mphands.Hands()
```

```
clk = 1
```

```
while True:
```

```
    ignore, frame = cam.read()
```

```
    frame = cv2.flip(frame, 1)
```

```
    frame = cv2.resize(frame, (1280, 720))
```

```
    frameRGB = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
```

```
    result = hands.process(frameRGB)
```

```
    for i in range(0, len(alph1), 1):
```

```
        if i < len(alph1) - 1:
```

```

x1 = R1x1

y1 = R1y1

x2 = R1x2

y2 = R1y2

cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 0, 0), -1)

cv2.putText(frame, alph1[i], (t1x, t1y), cv2.FONT_HERSHEY_SIMPLEX, 1, (255,
255, 255), 2)


if (result.multi_hand_landmarks):

    lmlist = []

    for handsLms in result.multi_hand_landmarks:

        for id, landmarks in enumerate(handsLms.landmark):

            x, y = int(landmarks.x * w), int(landmarks.y * h)

            lmlist.append([id, x, y])


    X = lmlist[8][1]

    Y = lmlist[8][2]

    cv2.circle(frame, (X, Y), 9, (0, 0, 255), cv2.FILLED)


    if x1 < X < x2 and y1 < Y < y2:

        cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 255, 0), -1)

        cv2.putText(frame, alph1[i], (t1x, t1y), cv2.FONT_HERSHEY_SIMPLEX, 1,
(255, 255, 255), 2)


    if (lmlist[12][2]) > (lmlist[11][2]) and ((clk > 0)):

```

```

        if x1 < X < x2 and y1 < Y < y2:

            b = alph1[i]

            pygame.mixer.Sound.play(mc)

            pg.press(b)

            cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 0, 255), -1)

            cv2.putText(frame, alph1[i], (t1x, t1y), cv2.FONT_HERSHEY_SIMPLEX,
1, (255, 255, 255), 2)

            clk = -1

        elif (lmlist[12][2] < lmlist[11][2]):

            clk = 1


R1x1 += 5

R1x2 += 5

R1x1 += 50

R1x2 += 50

t1x = R1x1 + 7


R1x1 = 225

R1x2 = 275

t1x = 232


for i in range(0, len(alph2), 1):

    if i > 0 and i < len(alph2) - 1:

        x1 = R2x1

```

```

y1 = R2y1

x2 = R2x2

y2 = R2y2

cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 0, 0), -1)

cv2.putText(frame, alph2[i], (t2x, t2y), cv2.FONT_HERSHEY_SIMPLEX, 1, (255,
255, 255), 2)

if (result.multi_hand_landmarks):

    lmlist = []

    for handsLms in result.multi_hand_landmarks:

        for id, landmarks in enumerate(handsLms.landmark):

            x, y = int(landmarks.x * w), int(landmarks.y * h)

            lmlist.append([id, x, y])

    X = lmlist[8][1]

    Y = lmlist[8][2]

    cv2.circle(frame, (X, Y), 9, (0, 0, 255), cv2.FILLED)

    if x1 < X < x2 and y1 < Y < y2:

        cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 255, 0), -1)

        cv2.putText(frame, alph2[i], (t2x, t2y), cv2.FONT_HERSHEY_SIMPLEX, 1,
(255, 255, 255), 2)

    if (lmlist[12][2] > lmlist[11][2]) and ((clk > 0)):

        if x1 < X < x2 and y1 < Y < y2:

            b = alph2[i]

```

```

pygame.mixer.Sound.play(mc)

pg.press(b)

cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 0, 255), -1)

cv2.putText(frame, alph2[i], (t2x, t2y), cv2.FONT_HERSHEY_SIMPLEX,
1, (255, 255, 255), 2)

clk = -1

elif (lmlist[12][2] < lmlist[11][2]):

    clk = 1


R2x1 += 5

R2x2 += 5

R2x1 += 50

R2x2 += 50

t2x = R2x1 + 7


if i >= len(alph2) - 1:

    x1 = R2x1

    y1 = R2y1

    x2 = R2x1 + 80

    y2 = R2y2

    cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 0, 0), -1)

    cv2.putText(frame, alph2[-1], (t2x, t2y), cv2.FONT_HERSHEY_SIMPLEX, 1, (255,
255, 255), 2)

if (result.multi_hand_landmarks):

    lmlist = []

```

```

for handsLms in result.multi_hand_landmarks:

    for id, landmarks in enumerate(handsLms.landmark):

        x, y = int(landmarks.x * w), int(landmarks.y * h)

        lmlist.append([id, x, y])

X = lmlist[8][1]

Y = lmlist[8][2]

cv2.circle(frame, (X, Y), 9, (0, 0, 255), cv2.FILLED)

if x1 < X < x2 and y1 < Y < y2:

    cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 255, 0), -1)

    cv2.putText(frame, alph2[-1], (t2x, t2y), cv2.FONT_HERSHEY_SIMPLEX, 1,
(255, 255, 255), 2)

if (lmlist[12][2] > lmlist[11][2]) and ((clk > 0)):

    if x1 < X < x2 and y1 < Y < y2:

        b = alph2[-1]

        pygame.mixer.Sound.play(mc)

        pg.press(b)

        cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 0, 255), -1)

        cv2.putText(frame, alph2[-1], (t2x, t2y),
cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 2)

        clk = -1

elif (lmlist[12][2] < lmlist[11][2]):

```

clk = 1

R2x1 = 240 + 60

R2x2 = 290 + 60

t2x = 247 + 60

for i in range(0, len(alph3), 1):

if i > 0 and i < len(alph3) - 1:

x1 = R3x1

y1 = R3y1

x2 = R3x2

y2 = R3y2

cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 0, 0), -1)

cv2.putText(frame, alph3[i], (t3x, t3y), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 2)

if (result.multi_hand_landmarks):

lmlist = []

for handsLms in result.multi_hand_landmarks:

for id, landmarks in enumerate(handsLms.landmark):

x, y = int(landmarks.x * w), int(landmarks.y * h)

lmlist.append([id, x, y])

X = lmlist[8][1]

Y = lmlist[8][2]

```

cv2.circle(frame, (X, Y), 9, (0, 0, 255), cv2.FILLED)

if x1 < X < x2 and y1 < Y < y2:

    cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 255, 0), -1)

    cv2.putText(frame, alph3[i], (t3x, t3y), cv2.FONT_HERSHEY_SIMPLEX, 1,
(255, 255, 255), 2)

if (lmlist[12][2] > lmlist[11][2]) and ((clk > 0)):

    if x1 < X < x2 and y1 < Y < y2:

        b = alph3[i]

        pygame.mixer.Sound.play(mc)

        pg.press(b)

        cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 0, 255), -1)

        cv2.putText(frame, alph3[i], (t3x, t3y), cv2.FONT_HERSHEY_SIMPLEX,
1, (255, 255, 255), 2)

        clk = -1

elif (lmlist[12][2] < lmlist[11][2]):

    clk = 1

R3x1 += 5

R3x2 += 5

R3x1 += 50

R3x2 += 50

t3x = R3x1 + 7

```

$R_{3x1} = 240 + 80$

$R_{3x2} = 290 + 80$

$t_{3x} = 247 + 80$

for i in range(0, len(alph4), 1):

 if i > 0 and i < len(alph4) - 1:

 x1 = R4x1

 y1 = R4y1

 x2 = R4x2

 y2 = R4y2

 cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 0, 0), -1)

 cv2.putText(frame, alph4[i], (t4x, t4y), cv2.FONT_HERSHEY_SIMPLEX,
1,(255,255,255),2)

if (result.multi_hand_landmarks):

 lmlist = []

 for handsLms in result.multi_hand_landmarks:

 for id, landmarks in enumerate(handsLms.landmark):

 x, y = int(landmarks.x * w), int(landmarks.y * h)

 lmlist.append([id, x, y])

 X = lmlist[8][1]

```

Y = lmlist[8][2]

cv2.circle(frame, (X, Y), 9, (0, 0, 255), cv2.FILLED)

if x1 < X < x2 and y1 < Y < y2:

    cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 255, 0), -1)

    cv2.putText(frame, alph4[i], (t4x, t4y), cv2.FONT_HERSHEY_SIMPLEX, 1,
(255, 255, 255), 2)


if (lmlist[12][2] > lmlist[11][2]) and ((clk > 0)):

    if x1 < X < x2 and y1 < Y < y2:

        b = alph4[i]

        pygame.mixer.Sound.play(mc)

        pg.press(b)

        cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 0, 255), -1)

        cv2.putText(frame, alph4[i], (t4x, t4y), cv2.FONT_HERSHEY_SIMPLEX,
1, (255, 255, 255), 2)

        clk = -1


elif (lmlist[12][2] < lmlist[11][2]):

    clk = 1

    R4x1 += 5

    R4x2 += 5

    R4x1 += 50

    R4x2 += 50

    t4x = R4x1 + 7

```

$$R_{4x1} = 240 + 110$$

$$R_{4x2} = 290 + 110$$

$$t_{4x} = 247 + 110$$

$$R_{1x1} = 225 + 70$$

$$R_{1x2} = 275 + 70$$

$$t_{1x} = 232 + 70$$

$$R_{1x1} = 225$$

$$R_{1x2} = 275$$

$$t_{1x} = 232$$

$$R_{5x1} = 240 + 220$$

$$R_{5x2} = 240 + 220 + 330 + 50$$

$$t_{5x} = 247 + 220 + 330$$

```
cv2.imshow('Virtual Keyboard', frame)
```

```
cv2.moveWindow('Virtual Keyboard', 0, 0)
```

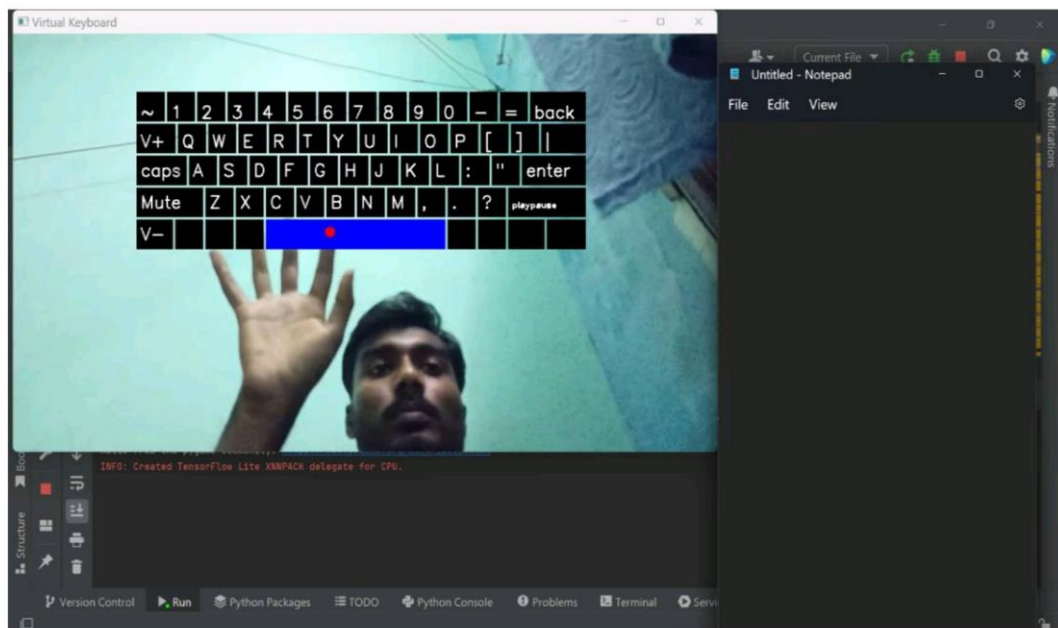
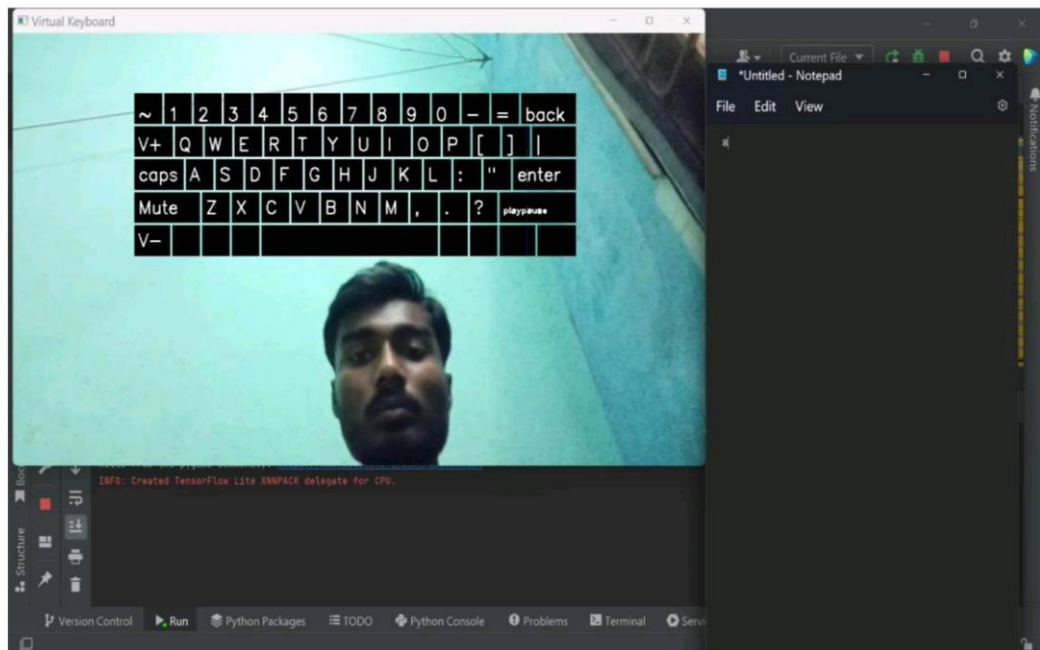
```
if cv2.waitKey(1) & 0xff == ord('q'):
```

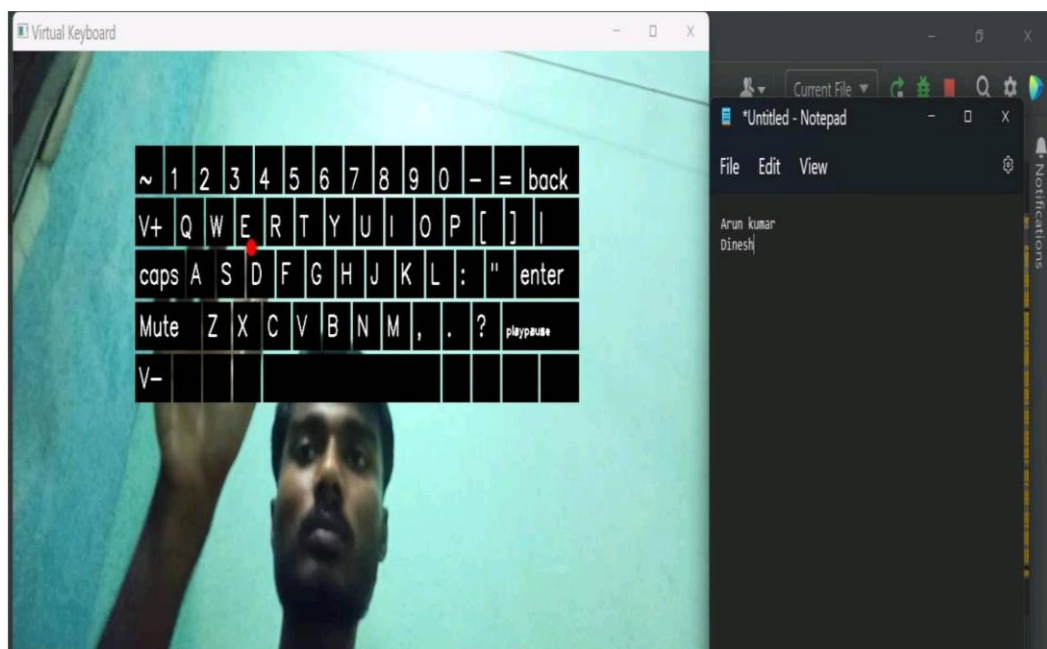
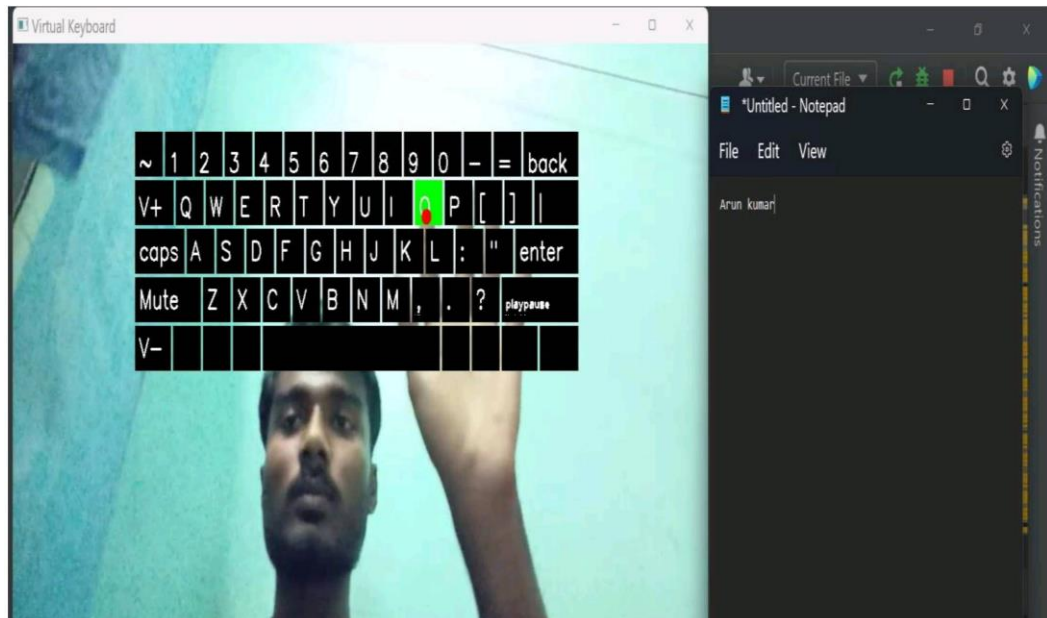
```
    cv2.destroyAllWindows()
```

```
    break
```

```
cam.release()
```

Output Screenshot:





CHAPTER 9

REFERENCES

- [1] Ray, A. (2022). How Hand gestures are replacing other computer input systems. Analytics India
- [2] Nikhil, C. D. S., Rao, C. U. S., Brumancia, E., Indira, K., Anandhi, T., & Ajitha, P. (2020, June). Finger Recognition and Gesture based Virtual Keyboard. In 2020 5th International Conference on Communication and Electronics Systems (ICCES) (pp. 1321- 1324). IEEE.
- [3] Harshit Sharma, Richa Saxena, Satendra Kumar, Ashish Saini ,Mohammad Saad, Mohammad Faraz(2022) Virtual Hands: Real Time Keyboard Desktop & Application Navigation using Gestures.IEEE
- [4] Shah, P., Pandya, K., Shah, H., & Gandhi, J. (2019). Survey on vision-based hand gesture recognition. International Journal of Computer Sciences and Engineering.
- [5] Cheng, H., Yang, L., & Liu, Z. (2015). Survey on 3D hand gesture recognition. IEEE transactions on circuits and systems for video technology.
- [6] Naganandhini K., Sowmya B., Pavish S., Nitesh J., Karthika R., and Prabhu E.(2022) Hand Tracking Based Human-Computer Interaction Teaching System. 3rd International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT).

- [7] Tae-Ho Lee & Hyuk-Jae Lee.(2018) Virtual Keyboard Design with Finger Gesture Recognition. IEEE International Symposium on Circuits and Systems (ISCAS).
- [8] Hagara M, Pucik J. “Accuracy of 3D camera based virtual keyboard”. 2014 24th International Conference on Radioelektronika .,IEEE.
- [9] AlKassim, Z., "Virtual laser keyboards: A giant leap towards human-computer interaction," 2012 International Conference on Computer Systems and Industrial Informatics (ICCSII).
- [10] Eckert, M. Lopez, M. ; Lazaro, C. ; Meneses, J. ; Martinez Ortega J.F2015 Mokey - A motion-based keyboard interpreter .Tech. Univ. of Madrid,Spain.