A Project Report on

"VIRTUAL MOUSE USING HAND GESTURES"

Submitted in partial fulfillment of the requirements of

the degree of

Bachelor of Technology

Submitted by

B.Sarath -(O180081)

B.Renuka Devi -(O180873)

T.Krishnaveni -(O180366)

P.Naveen -(O180892)

Under the Supervision of

Mr.G.Malakondaiah,M.Tech

Assistant Professor

Department Of ECE

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

RAJIV GANDHI UNIVERSITY OF KNOWLEDGE TECHNOLOGIES

ONGOLE CAMPUS

2023-2024

APPROVAL SHEET

This report entitled **"Virtual Mouse Using Hand Gestures"** by B.Sarath -(O180081), B.Renuka Devi -(O180873), T.Krishna Veni - (O180366), P. Naveen -(O180892) is approved for the degree of Bachelor of Technology in Electronics and Communication Engineering.

Examiner(s)

_____

_____

_____

Supervisior(s)

_____

_____

_____

_____

Chairman

_____

_____

Date: _____

Place:_____

CANDIDATE'S DECLARATION

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included. I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Signature

B.Sarath -(O180081) _____

B.Renuka Devi -(O180873) _____

T.Krishnaveni -(O180366) _____

P.Naveen -(O180892) _____

Date: _____

RAJIV GANDHI UNIVERSITY OF KNOWLEDGE TECHNOLOGIES

ONGOLE CAMPUS

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

Certificate

This is to certify that the report entitled **"Virtual Mouse Using Hand Gestures"** submitted by **B.Sarath - (O180081), B.Renuka Devi- (O180873) T.Krishna Veni- (O180366) and P.Naveen- (O180892)** in partial fulfilment of the requirements for the award of Bachelor of Technology in Electronics and Communication Engineering is a bonafide work carried by them under my supervision and guidance.

Project Internal Guide	**Head Of The Department**
Mr.G.Malakondaiah,M.Tech	Mrs.N.Padmavathi,M.Tech
Assistant Professor	Assistant Professor
Department Of ECE	Department Of ECE

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to **Mr.G.Malakondaiah** , my project guide for valuable suggestions and keen interest throughout the progress of my course and research.

I am grateful to **Mrs.N.Padmavathi**, HOD of **Electronics & Communication Engineering**,**MR.G.Bala Nagi Reddy**,former HOD of Electronics&Communication Engineering,**MR.M Rupas Kumar**,Dean of Academics, **Dr. Bhaskar Patel** director of RGUKT-ONGOLE,**Prof.B.Jayarami Reddy**,former Director of RGUKT-ONGOLE for providing excellent computing facilities and a congenial atmosphere for progressing with my project.

I would like to thank **RAJIV GANDHI UNIVERSITY OF KNOWLEDGE TECHNOLOGIES, ONGOLE** for providing all the necessary resources for the successful completion of my course work. At last, but not the least I thank my teammates and other students for their physical and moral support.

With Sincere Regards

B.Sarath -O180081

B.Renuka Devi -O180873

T.Krishnaveni -O180366

P.Naveen -O180892

Date:_____

ABSTRACT

With new changes seen in computer technology day by day, it has become quite essential for us to find specific new ways of interaction with computer systems as its need is increasing in society every day. Today, every device is making the use of touch screen technology on its systems, which isn't affordable to be used in all applications. A specific interactive module like a virtual mouse that makes use of Object Tracking and Gestures that will help us to interact can be an alternative way for the traditional touch screen and the physical mouse.

The objective is to create an Object Tracking application that interacts with the system. HSV Colour detection technique and Computer Vision libraries and algorithms which are previously used.

Now we proposed a new modules OpenCV,which establish a connection to webcam/camera and helps to observe the data,Media Pipe framework is used to analyze video/images by using graphs and PyAutoGUI allows users to intimate keyboard button pushes as well as mouse cursor movements and clicks.Finally,a distinct hand gesture movements were perform different operations in system.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER-1

INTRODUCTION

1. INTRODUCTION

1.1 Introduction

A virtual mouse is a computer program or system that simulates the functionality of a physical mouse using software techniques. It allows users to control the mouse cursor, perform clicks, and interact with graphical user interfaces (GUIs) without physically moving a physical mouse device. Virtual mice are often used in situations where traditional mouse input is impractical, such as in hands-free or gesture-based interactions, accessibility enhancements, or automation tasks.

Developing a virtual mouse using a combination of OpenCV, MediaPipe, and PyAutoGUI leverages the capabilities of these powerful libraries to create a hands-free or gesture-controlled mouse system. Here's a brief introduction to each component and how they contribute to the development of a virtual mouse:

1. **OpenCV (Open Source Computer Vision):**

    - OpenCV is a popular open-source library for computer vision and image processing tasks.
    - It provides a wide range of functionalities for tasks such as image/video capture, object detection and tracking, image manipulation, and more.
    - In the context of a virtual mouse, OpenCV can be used for capturing live video from a camera, detecting hand gestures or movements, and processing the video frames to extract relevant information for controlling the mouse cursor.

2. **MediaPipe:**

    - MediaPipe is a comprehensive cross-platform framework developed by Google that offers ready-to-use machine learning solutions for various perception tasks, including hand tracking.
    - It provides pre-trained models and tools for building pipelines for tasks such as hand detection and tracking.
    - In the context of a virtual mouse, MediaPipe can be used for detecting and tracking the user's hand movements in real-time, allowing for gesture-based control of the mouse cursor.

3. **PyAutoGUI:**

- PyAutoGUI is a Python library for GUI automation tasks such as controlling the mouse, keyboard, and screen.
- It provides functionalities for simulating mouse movements, clicks, keystrokes, and more.
- In the context of a virtual mouse, PyAutoGUI can be used to translate the detected hand movements into corresponding mouse movements and clicks on the screen, effectively creating a virtual mouse control system.

By integrating OpenCV for video processing, MediaPipe for hand tracking, and PyAutoGUI for mouse control, developers can create a virtual mouse system that allows users to interact with graphical user interfaces using hand gestures or movements captured by a camera. This can enable hands-free or gesture-based interactions in various applications, such as accessibility tools, gaming, presentations, and more.

1.2 Incidents where it is used:

The virtual mouse project, developed using OpenCV, MediaPipe, and PyAutoGUI, can be applied in various scenarios and incidents where traditional mouse input may be impractical or inaccessible. Here are some situations where the virtual mouse project could be particularly useful:

1. **Accessibility Enhancement:** Individuals with physical disabilities or mobility impairments may find it challenging to use a traditional mouse. A virtual mouse system that can be controlled via hand gestures or movements captured by a camera can provide an accessible alternative for interacting with computers and assistive technologies.

2. Hands-Free Interaction: In environments where users cannot physically manipulate a mouse, such as in sterile environments like operating rooms or clean rooms, a virtual mouse controlled by hand gestures can enable hands-free interaction with computers and displays.

3. Gesture-Based Interfaces: Virtual mouse technology can be integrated into gesture-based interfaces for interactive displays, kiosks, or presentations. Users can navigate menus, select options, and interact with content using intuitive hand gestures, enhancing user experience and engagement.

4. Gaming and Entertainment: Virtual mouse systems can be used in gaming applications to provide immersive and interactive experiences. Players can control game characters or navigate virtual environments using hand gestures, adding a new dimension to gaming interactions.

5. Education and Training: Virtual mouse technology can be utilized in educational settings for interactive learning experiences. Students can engage with educational content, simulations, and virtual labs using hand gestures to control the virtual mouse, promoting active learning and exploration.

6. Remote Control Systems: In scenarios where traditional input devices are not available or practical, such as remote monitoring or control systems, a virtual mouse controlled by hand gestures can provide a convenient way to interact with remote computers or devices.

7. Assistive Technologies: Virtual mouse systems can be integrated into assistive technologies and communication devices for individuals with disabilities. Users can control computers, communicate, and access digital resources using hand gestures, empowering them to participate more fully in daily activities and social interactions.

8. Artistic and Creative Applications: Artists, designers, and creators can use virtual mouse technology to explore new forms of digital expression and creativity. Hand gestures can be used to manipulate digital art tools, create interactive installations, or control multimedia presentations.

Overall, the virtual mouse project offers a versatile and accessible solution for interacting with computers and digital interfaces, with potential applications across various domains including accessibility, entertainment, education, healthcare, and more.

CHAPTER-2

LITERATURE

2.LITERATURE

2.1 Literature

Title: Real-Time Hand Gesture Recognition Using Python and OpenCV

Description: This paper presents a real-time hand gesture recognition system implemented using Python and OpenCV. The system enables the recognition of hand gestures in real-time, laying the foundation for applications such as virtual mouse control and interactive interfaces.

Title: Implementation of Virtual Mouse Pointer Control System Based on Hand Gesture Recognition

Description: This paper describes the implementation of a virtual mouse pointer control system that utilizes hand gesture recognition techniques. By recognizing specific hand gestures, the system enables users to control the mouse pointer in a virtual environment, offering an alternative interaction method for computing devices.

Title: Hand Gesture Recognition for Virtual Mouse Control

Description: This paper discusses the design and implementation of a hand gesture recognition system tailored for virtual mouse control applications. By accurately detecting and interpreting hand gestures, the system facilitates intuitive and hands-free interaction with computing devices, enhancing user experience and accessibility.

Title: Virtual Mouse Control Using Hand Gesture Recognition

Description: This paper presents a virtual mouse control system that harnesses hand gesture recognition technology. By tracking and analyzing hand movements and gestures, the system translates these inputs into mouse cursor movements and actions, offering an alternative and potentially more natural way of interacting with computers.

Title: A Survey on Hand Gesture Recognition for Human-Computer Interaction

Description: This survey paper provides a comprehensive overview of hand gesture recognition techniques and their applications in human-computer interaction. It covers various methodologies, algorithms, and technologies used for hand gesture recognition, shedding light on the advancements and challenges in this field.

CHAPTER-3

EXISTING METHOD

3. EXISTING METHOD

The Existing system uses Computer Vision libraries and algorithms to determine the object, its movements, and act as the movement using Real-time tracking. But our primary focus is on pointing the mouse and different actions by hand tracking and the movements performed by the hand and the fingers.

3.1 HSV Color detection technique:

The HSV (Hue, Saturation, Value) color space is a representation of colors commonly used in computer vision and image processing tasks like color detection, segmentation, and object tracking. Here's how the HSV color detection technique works:

- **Hue (H):** Represents the type of color, such as red, green, blue, etc. It is represented as an angle around a color wheel, with red at 0°, green at 120°, and blue at 240°.
- **Saturation (S):** Represents the intensity or purity of the color. A higher saturation value means the color is more vivid, while lower values indicate a more washed-out or gray version of the color.
- **Value (V):** Represents the brightness of the color. A higher value indicates a brighter color, while lower values represent darker shades.

Here's a basic approach to HSV color detection:

- **Convert the Image to HSV Color Space:** Convert the input image from the RGB color space to the HSV color space. This can be done using various libraries such as OpenCV in Python.
- **Define the Color Range:** Define the lower and upper bounds of the HSV values for the color you want to detect. This can be done by specifying a range of values for H, S, and V.
- **Thresholding:** Apply thresholding to the HSV image to create a binary mask where the pixels within the specified color range are set to white (255) and all other pixels are set to black (0).
- **Object Detection:** Use the binary mask to detect the objects or regions in the image that match the specified color range. This can be done by finding contours, connected components, or using other segmentation techniques.

- **Filtering and Post-Processing:** Optionally, apply filtering or post-processing techniques to refine the detected regions, remove noise, or improve the accuracy of the detection results.

- **Visualization:** Visualize the detected objects or regions by overlaying them on the original image or displaying them separately.

Overall, the HSV color detection technique provides a flexible and intuitive way to detect specific colors in images and is widely used in various computer vision applications.

Fig:3.1.1 RGB to HSV Conversion

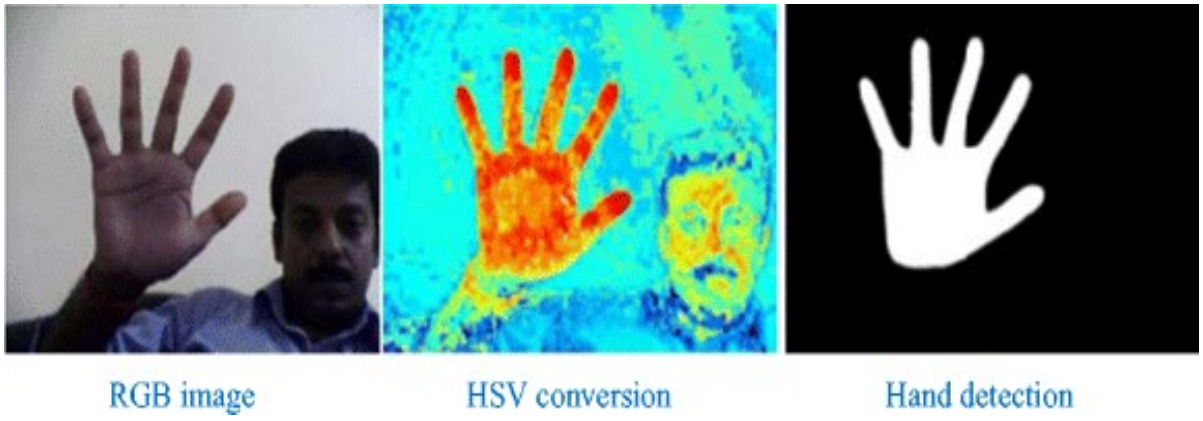


Fig:3.1.2 Detecting hand using HSV technique

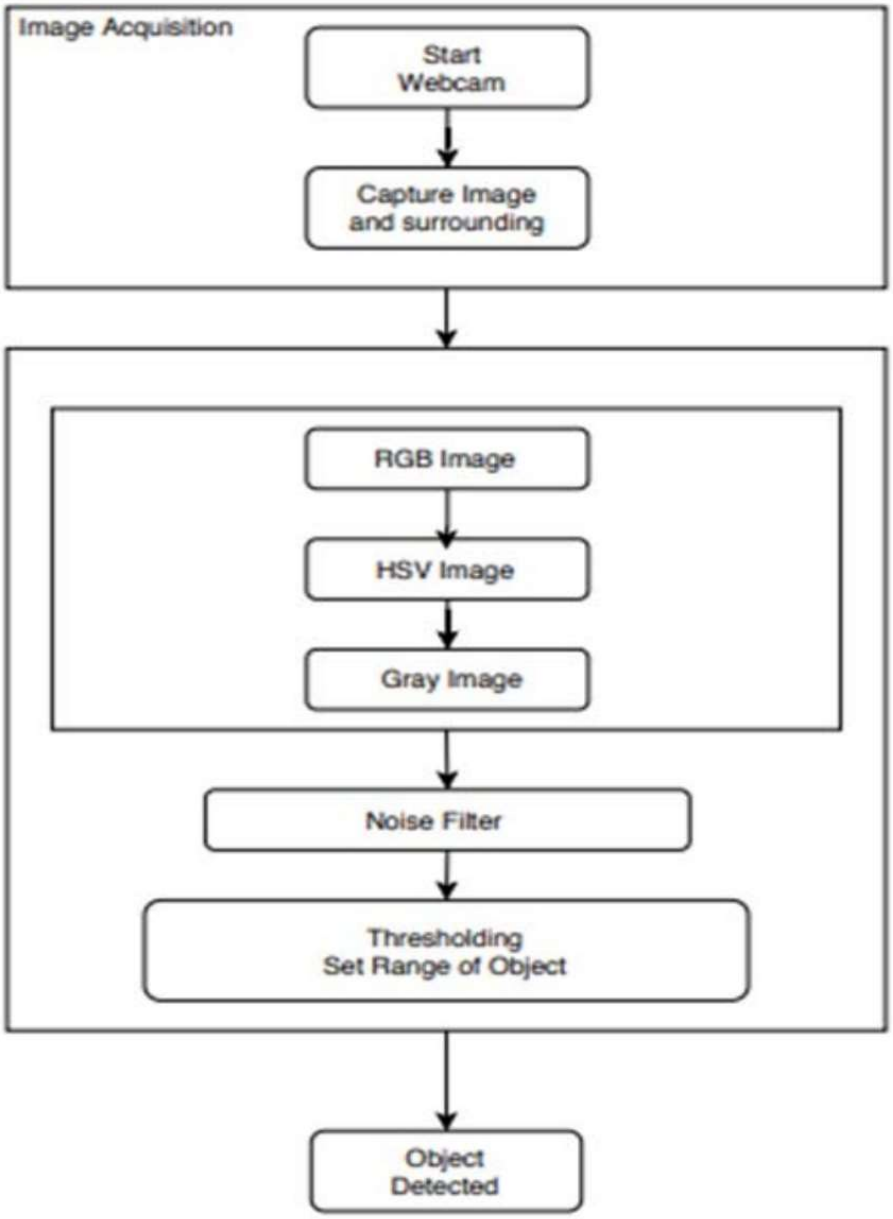


Fig:3.1.3 Flow chart of HSV Conversion

CHAPTER-4

PROPOSED METHOD

4. PROPOSED METHOD

4.1 OpenCV2

OpenCV, short for Open Source Computer Vision Library, is an open-source computer vision and machine learning software library. Originally developed by Intel, it is now maintained by a community of developers under the OpenCV Foundation.

Introduction

Opencv is a huge open-source library for computer vision, machine learning, and image processing. Now, it plays a major role in real-time operation which is very important in today's systems. By using it, one can process images and videos to identify objects, faces, or even the handwriting of a human.

When it is integrated with various libraries, such as NumPy, python is capable of processing the opencv array structure for analysis. To identify an image pattern and its various features we use vector space and perform mathematical operations on these features.

The first OpenCV version was 1.0. OpenCV is released under a BSD license and hence it's free for both academic and commercial use. It has C++, C, Python, and Java interfaces and supports Windows, Linux, Mac OS, iOS and Android. When OpenCV was designed the main focus was real-time applications for computational efficiency. All things are written in optimized C/C++ to take advantage of multi-core processing.

Features of OpenCV2:

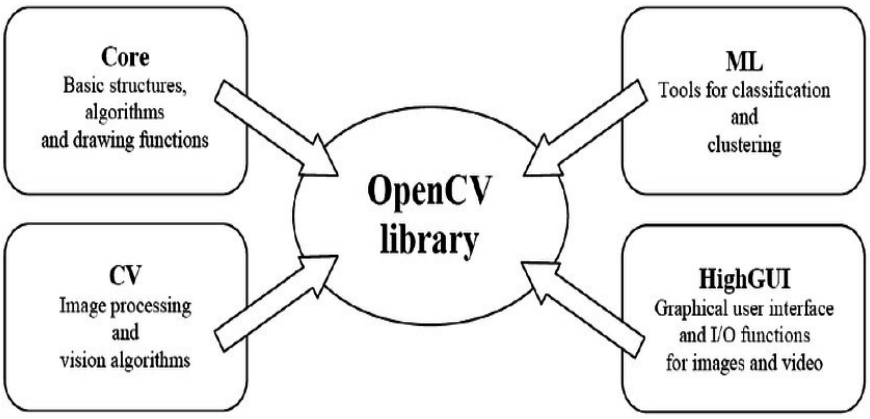


Fig: 4.1.1 OpenCV Library

- **Open Source**: OpenCV is freely available to use, study, modify, and distribute. Its open-source nature fosters collaboration and innovation among developers, researchers, and enthusiasts.

- **Fast Speed:** OpenCV is optimized for speed and efficiency, making it suitable for real-time applications. It utilizes hardware acceleration and efficient algorithms to process images and videos quickly.

- **Easy to Integrate:** OpenCV can be easily integrated with other libraries and frameworks, such as NumPy, SciPy, and machine learning libraries like TensorFlow and PyTorch, enhancing its capabilities and flexibility.

- **Easy Coding:** OpenCV provides a simple and intuitive API (Application Programming Interface) with extensive documentation and examples, making it easy for developers to write code for various computer vision tasks.

- **Fast Prototyping:** OpenCV enables rapid prototyping of computer vision applications due to its extensive collection of pre-built functions, algorithms, and tools. This allows developers to quickly test ideas and iterate on their projects.

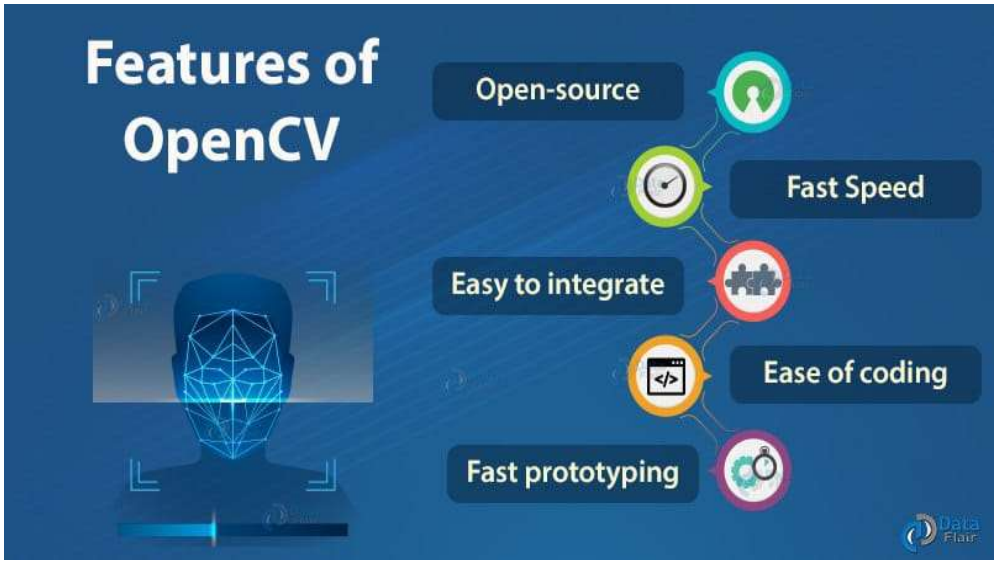


Fig.4.1.2 Features of OpenCV

OpenCV functionality:

- Image/video I/O, processing, display (core, imgproc, highgui)
- Object/feature detection (objdetect, features2d, nonfree)

- Geometry-based monocular or stereo computer vision (calib3d, stitching, videostab)
- Computational photography (photo, video, superres)
- Machine learning & clustering (ml, flann)
- CUDA acceleration (gpu)

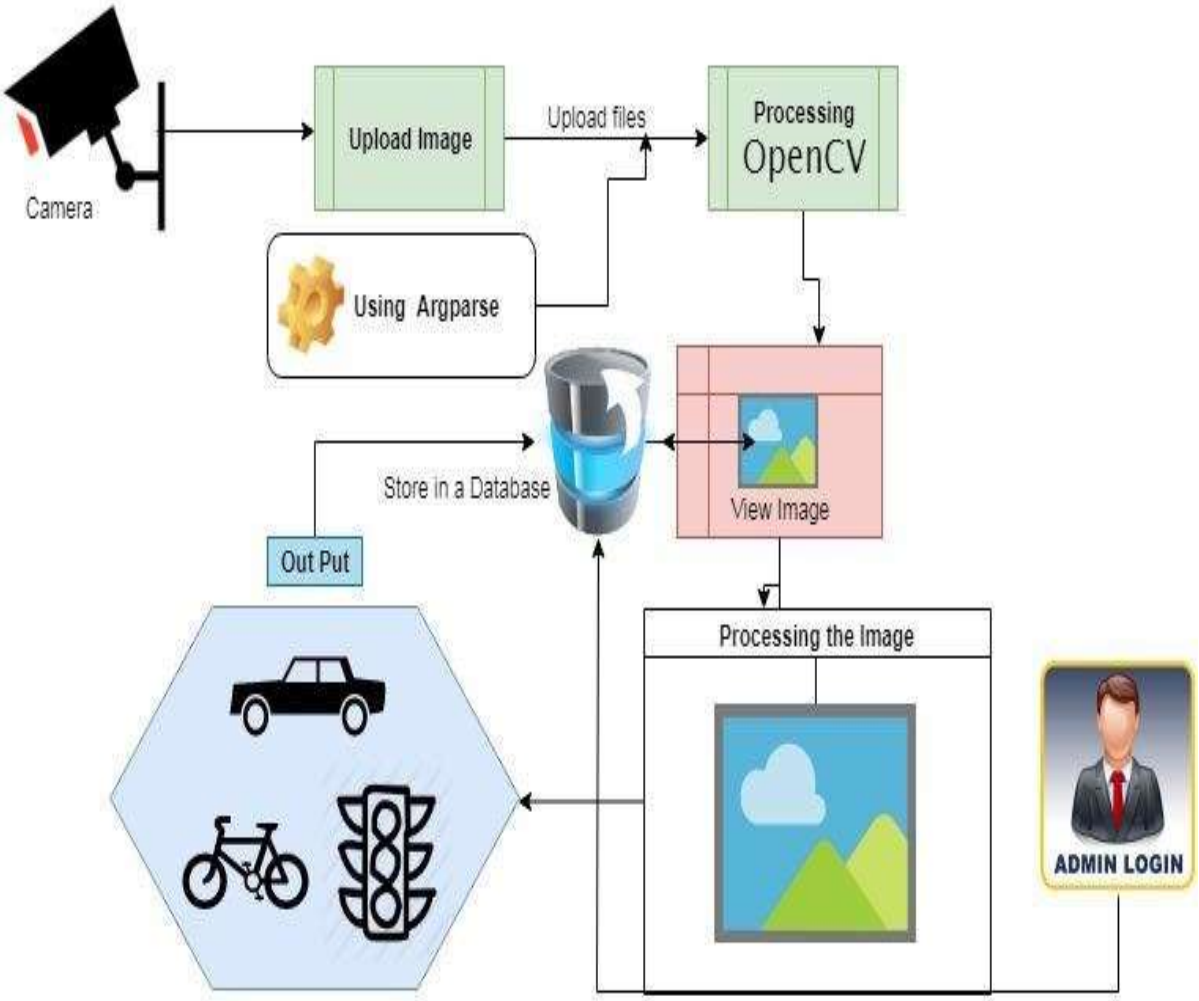


Fig:4.1.3 OpenCV Functionality

Applications of OpenCV

There are lots of applications which are solved using OpenCV, some of them are listed below:

- Face recognition
- Automated inspection and surveillance
- number of people – count (foot traffic in a mall, etc)
- Vehicle counting on highways along with their speeds
- Interactive art installations
- Anomaly (defect) detection in the manufacturing process (the odd defective products)
- Street view image stitching
- Video/image search and retrieval
- Robot and driver-less car navigation and control
- object recognition
- Medical image analysis
- Movies – 3D structure from motion
- TV Channels advertisement recognition.

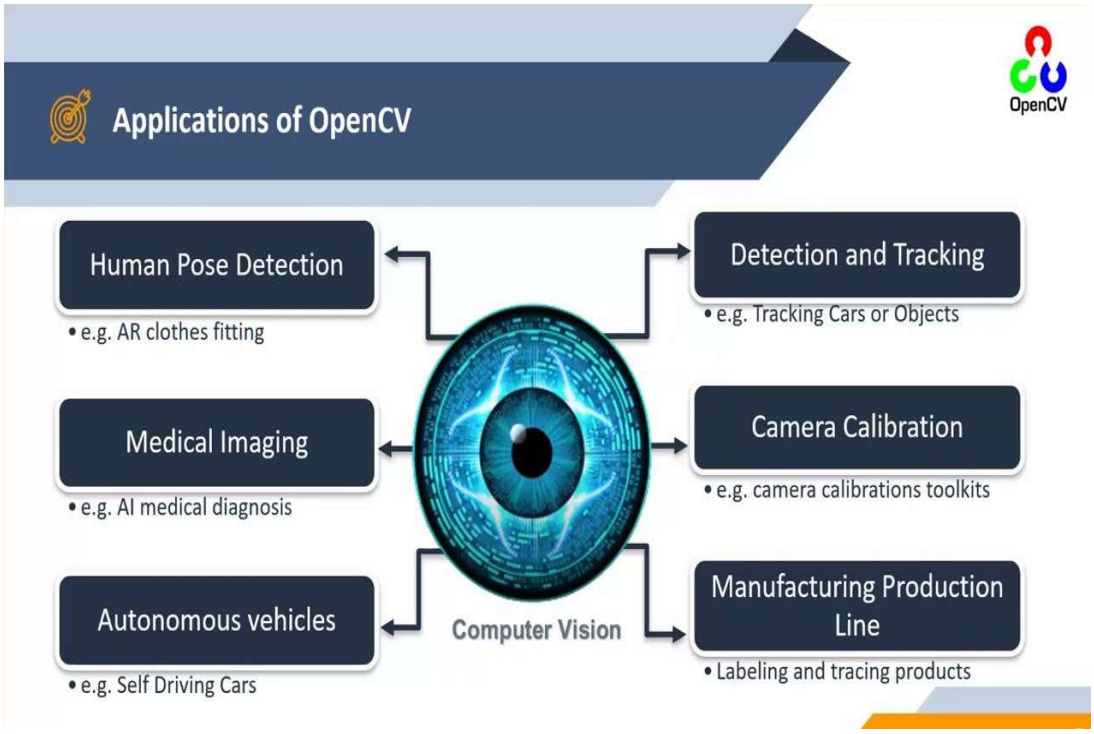


Fig:4.1.4 Applications of OpenCV

4.2 Mediapipe:

MediaPipe is a comprehensive cross-platform framework developed by Google that offers ready-to-use, high-quality ML solutions for various tasks such as object detection, pose estimation, hand tracking, and face detection. It provides easy-to-use Python APIs for integrating machine learning models into real-world applications.

The story of MediaPipe in Python begins with Google's commitment to democratizing machine learning and computer vision technologies. Recognizing the growing need for accessible and efficient tools in these domains, Google engineers embarked on a journey to create a framework that simplifies the development and deployment of ML-based applications.

It provides ready-to-use ML solutions with pre-trained models, as well as tools for building custom pipelines tailored to specific applications.

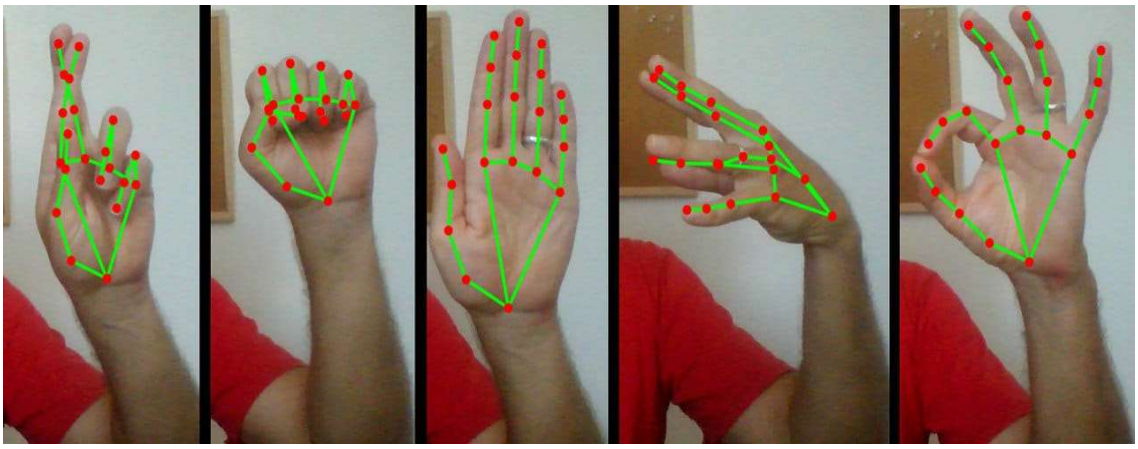


Fig:4.2.1 Hand landmarks

Key features of the MediaPipe library include:

- **Modular Design**: MediaPipe adopts a modular design, allowing developers to construct ML pipelines by connecting reusable building blocks called "calculators." These calculators encapsulate specific ML models or processing functions, enabling easy composition and customization of pipelines.

- **Ready-to-Use Solutions:** MediaPipe provides a collection of pre-trained models and pipelines for common perception tasks such as face detection, hand tracking, pose estimation, object detection, and more. These solutions are designed to work out-of-the-box, enabling rapid development of applications without the need for training new models.

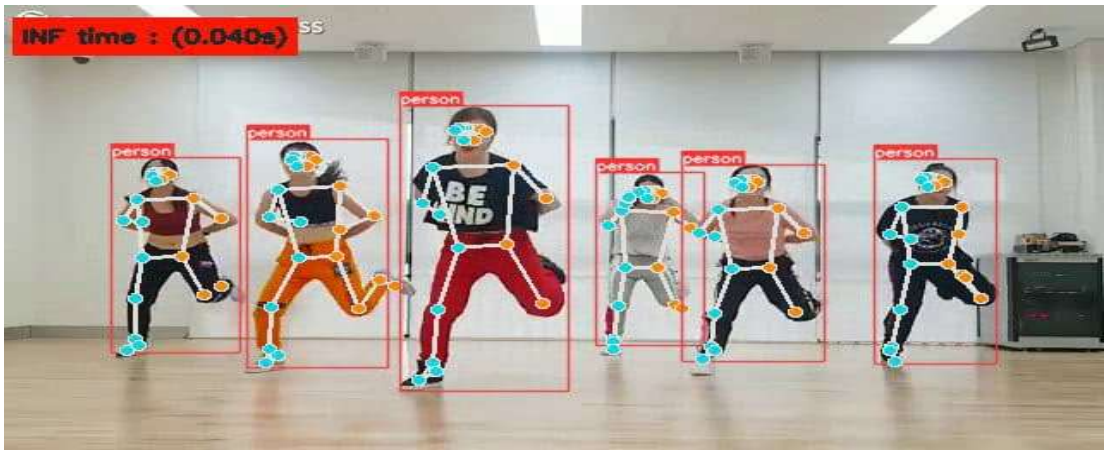


Fig:4.2.2 Detecting full body landmarks

- **Cross-Platform Support:** MediaPipe is designed to be cross-platform, supporting deployment on various devices and operating systems, including desktops, mobile devices, and embedded systems. It offers APIs for Python as well as C++, making it accessible to a wide range of developers.

- **Real-Time Performance**: MediaPipe is optimized for real-time performance, making it suitable for applications that require low-latency processing of multimedia data, such as augmented reality (AR) and gesture recognition systems.

- **Integration with TensorFlow**: MediaPipe seamlessly integrates with TensorFlow, Google's open-source ML framework, allowing developers to leverage TensorFlow models within MediaPipe pipelines. This integration enables the use of custom ML models and facilitates interoperability with other TensorFlow-based tools and workflows.

- **Extensibility and Customization:** Developers can extend MediaPipe's functionality by creating custom calculators and pipelines tailored to their specific requirements. MediaPipe's modular architecture and flexible APIs make it easy to integrate custom ML models, algorithms, and processing techniques into existing pipelines.

Overall, MediaPipe provides a powerful and flexible platform for building ML-based perception applications in Python, empowering developers to create innovative solutions for a wide range of domains, including computer vision, multimedia processing, robotics, healthcare, and more.

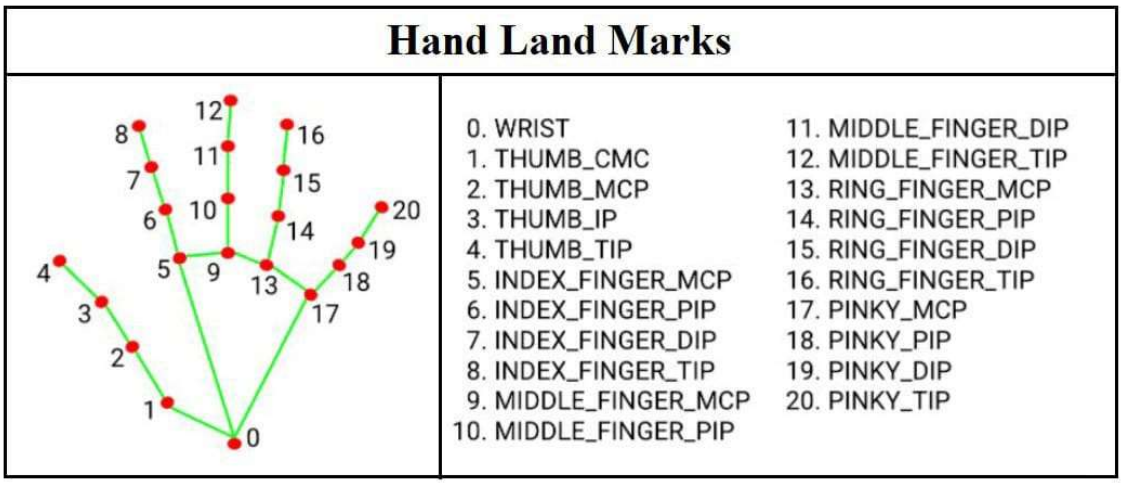


Fig:4.2.3 Hand land marks analysis

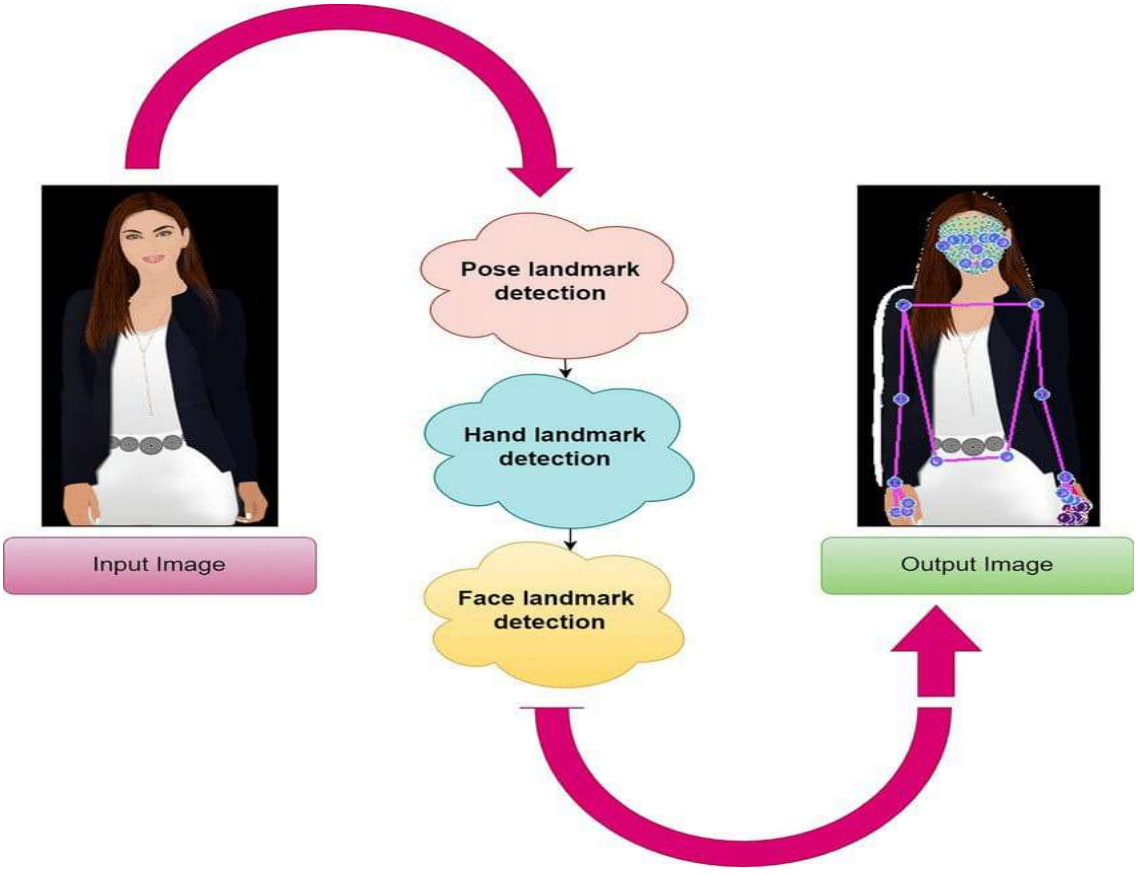


Fig:4.2.4 Process of detecting landmarks

4.3 PyAutoGUI :

PyAutoGUI is a Python library that provides cross-platform support for controlling the mouse, keyboard, and screen of a computer through code. It allows automation of GUI (Graphical User Interface) interactions and provides functionalities for tasks such as simulating mouse clicks and movements, typing keystrokes, taking screenshots, and more.

Here are some of the main features and functionalities of PyAutoGUI:

- **Mouse Control:** PyAutoGUI allows you to programmatically control the mouse, including moving the cursor to specific coordinates, clicking mouse buttons, dragging, scrolling, and more.
- **Keyboard Control:** With PyAutoGUI, you can simulate keyboard input by sending keystrokes, key combinations (e.g., Ctrl+C), and keyboard shortcuts to the active window or application.
- **Screen Interaction:** PyAutoGUI provides functions for capturing screenshots of the entire screen or specific regions, locating images on the screen, and performing image recognition tasks.
- **Window Management**: PyAutoGUI offers basic window management functionalities such as minimizing, maximizing, and focusing on windows, as well as getting information about window titles and sizes.
- **Multiplatform Support:** PyAutoGUI is designed to work on multiple operating systems, including Windows, macOS, and Linux, making it suitable for cross-platform automation tasks.
- **Ease of Use**: PyAutoGUI aims to be user-friendly and easy to use, with a simple and intuitive API that allows developers to automate GUI interactions with minimal effort.
- **Customization**: While PyAutoGUI provides high-level functions for common automation tasks, it also offers flexibility and customization options for more advanced users, allowing fine-tuning and adaptation to specific use cases.

PyAutoGUI can be particularly useful for tasks such as GUI testing, automation of repetitive tasks, creating bots or scripts for games or applications, accessibility enhancements, and more. However, it's essential to use PyAutoGUI responsibly and ethically, as it can potentially be used for unintended purposes such as automating malicious activities or violating terms of service agreements.

Overall Implementation

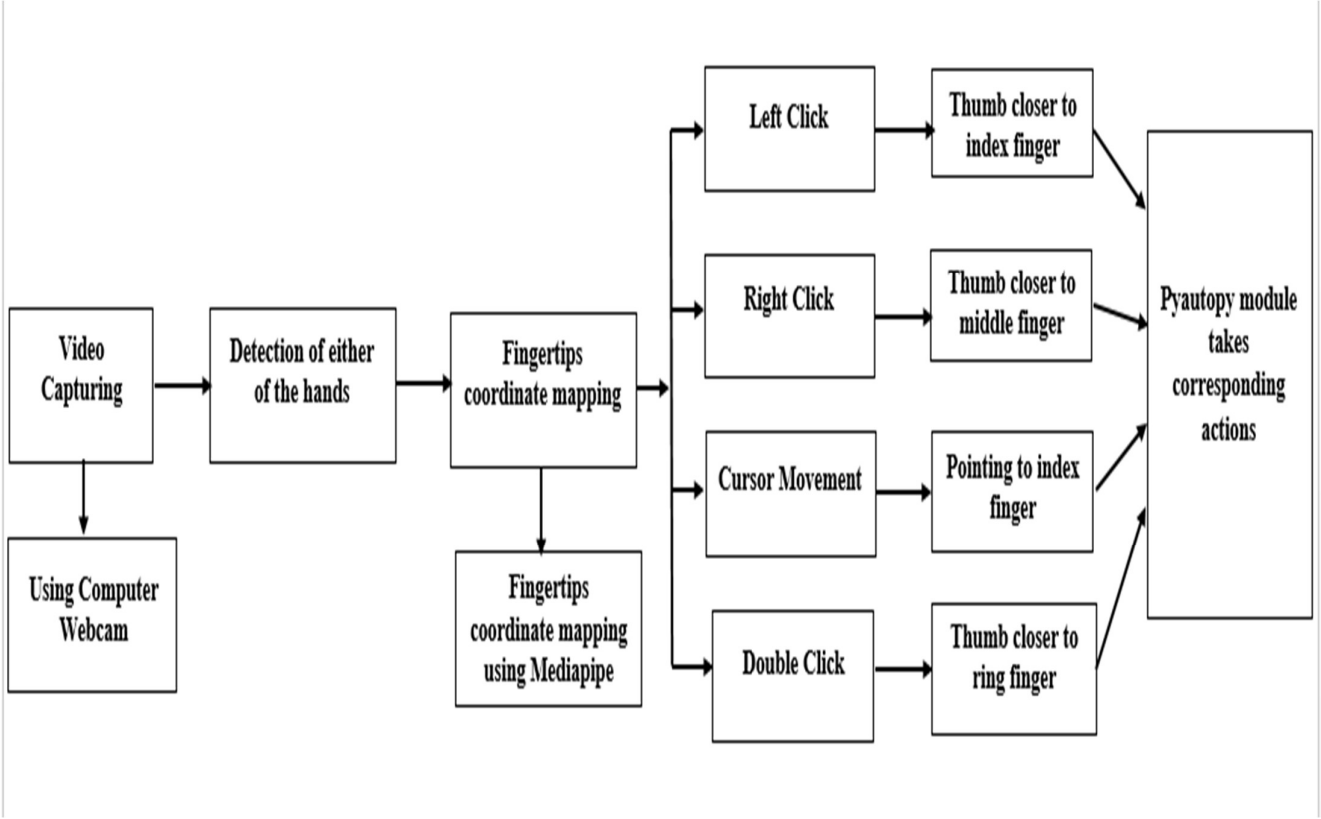


Fig:4.3.1.Overall Implementation

CHAPTER-5

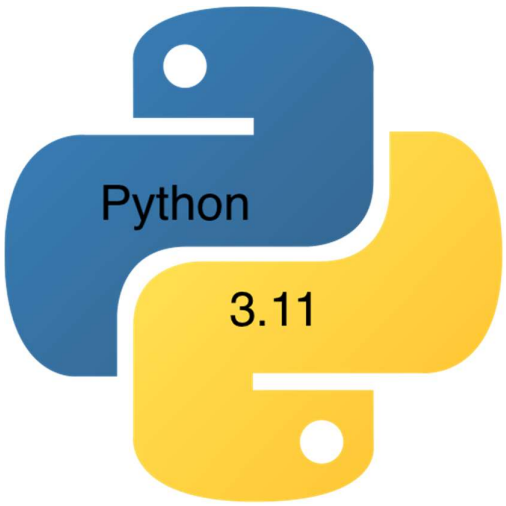
SOFTWARE REQUIREMENTS

5.SOFTWARE REQUIREMENTS :

Python 3.11.0:

Every version of Python has developments and variations, and 3.11 is no exception. Python releases a new version every year. A feature-locked beta release is done in the first half of the year and the final release is done in the second half of the year.

The release of Python 3.11 occurred on October 24, 2022. Python's latest version is faster and easier to use. It has been in development for 17 months and is now ready for widespread use.



Here are some anticipated uses and potential features for Python 3.11.0:

- **Performance Improvements:** Python 3.11.0 may introduce optimizations and performance improvements to the language runtime and standard library modules. These improvements could lead to faster execution times and better resource utilization for Python programs.

- **Language Enhancements:** Python 3.11.0 might introduce new language syntax, features, or enhancements to existing language constructs. These changes could make Python code more expressive, readable, and efficient.

- **Standard Library Updates:** The new release could include updates, additions, or deprecations to the Python standard library modules. These changes might address new use cases, improve existing functionalities, or remove outdated features.

- **Security Enhancements**: Python 3.11.0 may include security enhancements to protect against common vulnerabilities and address security issues in the language or standard library.

- **Developer Experience Improvements:** The release might introduce improvements to developer tools, debugging capabilities, error reporting, and other aspects of the Python development experience. These enhancements could make it easier for developers to write, debug, and maintain Python code.

- **Compatibility and Migration:** Python 3.11.0 could focus on improving compatibility with previous Python versions and easing the migration path for projects still using Python 2.x. This might involve deprecating or removing features that are no longer recommended or supported.

- **Community Contributions:** The release will likely incorporate contributions from the Python community, including bug fixes, patches, and new features proposed and implemented by Python developers worldwide.

- **Documentation and Educational Resources:** Python 3.11.0 might include updates to the official Python documentation, tutorials, and educational resources to help users learn about new features and best practices in Python programming.

CHAPTER-6

RESULTS AND DISCUSSIONS

6. RESULTS AND DISCUSSIONS

Fig:6.1. Hand gesture to move the cursor

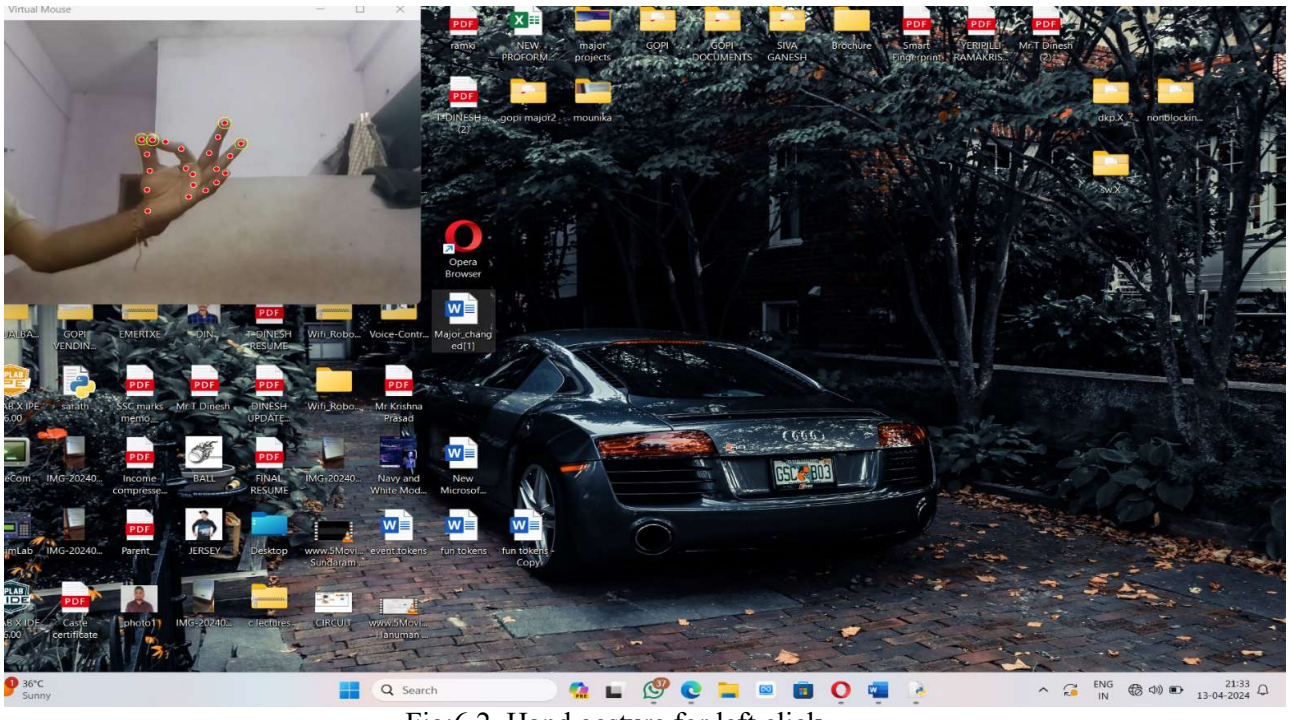


Fig:6.2. Hand gesture for left click

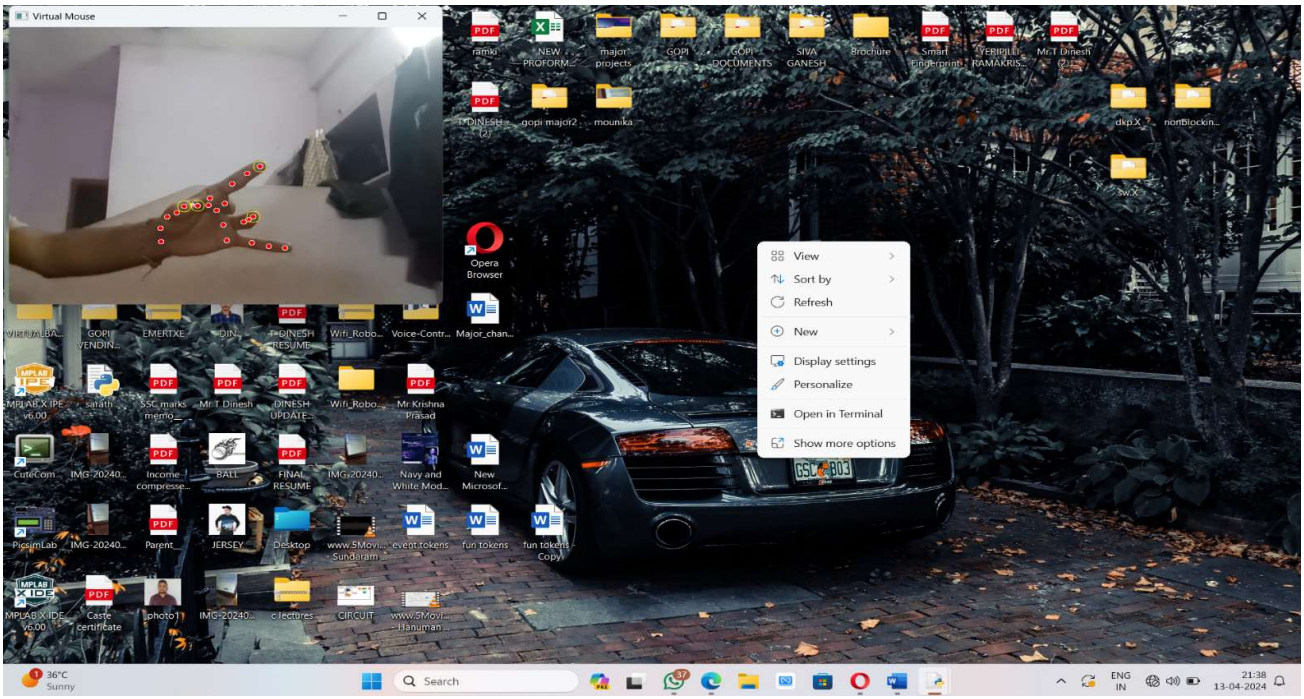


Fig:6.3. Hand gesture for right click

Fig:6.4. Hand gesture for double click

CHAPTER-7
CONCLUSION

7.CONCLUSION

The virtual mouse project harnesses the collective power of PyAutoGUI, MediaPipe, and OpenCV2 to craft an innovative human-computer interaction experience. PyAutoGUI streamlines the automation of mouse and keyboard actions, facilitating effortless control of the computer. MediaPipe's robust hand tracking capabilities enable precise detection and recognition of hand gestures, providing the foundation for intuitive control. Complementing these, OpenCV2 offers a versatile suite of computer vision tools, essential for image processing tasks crucial to gesture recognition.

Together, these modules create a seamless and responsive interface, exemplifying the potential of gesture-based interactions in enhancing user experience. By leveraging these technologies, the project not only showcases the synergy between computer vision and automation but also sets the stage for future advancements in human-computer interaction paradigms.

APPENDIX:

Final code:

```
import cv2
import mediapipe as mp
import pyautogui

cap = cv2.VideoCapture(0)
hand_detector = mp.solutions.hands.Hands()
drawing_utils = mp.solutions.drawing_utils
screen_width, screen_height = pyautogui.size()
index_y = 0

while True:
    _, frame = cap.read()
    frame = cv2.flip(frame, 1)
    frame_height, frame_width, _ = frame.shape
    rgb_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    output = hand_detector.process(rgb_frame)
    hands = output.multi_hand_landmarks

    if hands:
        for hand in hands:
            drawing_utils.draw_landmarks(frame, hand)
            landmarks = hand.landmark

            for id, landmark in enumerate(landmarks):
                x = int(landmark.x * frame_width)
                y = int(landmark.y * frame_height)

                if id == 8:  # Index finger
```

```
        cv2.circle(img=frame, center=(x, y), radius=10, color=(0, 255, 255))
        index_x = screen_width / frame_width * x
        index_y = screen_height / frame_height * y

    if id == 4:  # Thumb
        cv2.circle(img=frame, center=(x, y), radius=10, color=(0, 255, 255))
        thumb_x = screen_width / frame_width * x
        thumb_y = screen_height / frame_height * y

    if id == 12:  # middle finger
        cv2.circle(img=frame, center=(x, y), radius=10, color=(0, 255, 255))
        middle_x = screen_width / frame_width * x
        middle_y = screen_height / frame_height * y

    if id == 16:  # ring finger
        cv2.circle(img=frame, center=(x, y), radius=10, color=(0, 255, 255))
        ring_x = screen_width / frame_width * x
        ring_y = screen_height / frame_height * y



        # left-click when thumb meets index finger
        if abs(index_y - thumb_y) < 5: # Thumb close to index finger
                pyautogui.leftClick()
                pyautogui.sleep(0.5)

        # Right-click when thumb meets middle finger
        elif abs(middle_y - thumb_y) < 10:  # Thumb close to middle finger
                pyautogui.rightClick()
                pyautogui.sleep(0.5)
```

```
            # Move the cursor if thumb not close to index or middle finger
            elif abs(index_y - thumb_y) < 150:
                pyautogui.moveTo(index_x, index_y)

            # Double-click when thumb meets ring finger
            elif abs(ring_y - thumb_y) < 20:   # Thumb close to middle finger
                    pyautogui.click(clicks=2)
                    pyautogui.sleep(1)



    cv2.imshow('Virtual Mouse', frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

REFERENCES:

I. N. Subhash Chandra, T. Venu, P. Srikanth, "A Real-Time Static & Dynamic Hand Gesture Recognition System", International Journal of Engineering Inventions, Volume 4, Issue 12, August 2019.11:20 AM

II. Rashmi Adatkar, Ronak Joshi, Mehul Jani, Prashant Karn, " Virtual Mouse ",Imperial Journal of Interdisciplinary Research (IJIR), Vol-3, Issue- 4, 2020.

III. J.L. Raheja, A. Chaudhary, K. Singal, 2011, "Tracking of Fingertips and Centre of Palm using KINECT ", In proceedings of the 3rd IEEE International Conference on Computational Intelligence, Modelling and Simulation.

IV. K. P. Vinay, "Cursor control using hand gestures," International Journal of Critical Accounting, vol. 0975–8887, 2016. View at: Google Scholar.

V. J. Katona, "A review of human–computer interaction and virtual reality research fields in cognitive Info Communications," Applied Sciences, vol. 11, no. 6, p. 2646, 2021.

VI. https://link.springer.com/chapter/10.1007/978-981-16-8862

VII. https://developers.google.com/mediapipe/solutions/vision/hand_landmarker

VIII. https://pypi.org/project/PyAutoGUI/