A

REAL-TIME RESEARCH PROJECT REPORT

ON

AI VIRTUAL MOUSE USING HAND GESTURES

Submitted in partial fulfillment of the requirement for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted by

B. BLESSY PRIYA (23R91A0539**)**

Under the Guidance

Of

Mrs. SWETHA .G

Asst. Professor

Department of CSE

Department of Computer Science and Engineering
TEEGALA KRISHNA REDDY ENGINEERING COLLEGE
(An Autonomous Institution)
Medbowli, Meerpet, Saroornagar, Hyderabad – 500097

(Affiliated to JNTUH, Approved by AICTE, Accredited by NBA & NAAC)
(2024-2025)

TEEGALA KRISHNA REDDY ENGINEERING COLLEGE

(Sponsored by TKR Educational Society)

Approved by AICTE, Affiliated by JNTUH, Accredited by NBA & NAAC)

Medbowli, Meerpet, Saroornagar, Hyderabad – 500 097.

Phone: 040-24092838 Fax: +91-040-24092555

E-mail: tkrec@rediffmail.com Website: www.tkrec.ac.in

Department of Computer Science & Engineering

College code: R9

CERTIFICATE

This is to certify that the Real-Time Research Project report on “ Ai Virtual Mous Using Hand Gestures” is a bonafide work carried out by Students’ B. Blessy Priya 23R91A0539 nos in partial fulfillment for the requirement of the award of B.Tech degree in Computer Science and Engineering, Teegala Krishna Reddy Engineering College, Hyderabad, affiliated to Jawaharlal Nehru Technological University, Hyderabad under my guidance and supervision.

The result of investigation enclosed in this report have been verified and found satisfactory. The results embodied in the project work have not been submitted to any other University for the award of any degree.

INTERNAL GUIDE	**HEAD OF DEPARTMENT**
Mrs. G. Swetha	Dr.CH.V.Phani Krishna
Asst. Professor	Professor
…………………………………..	………………………………..

College code: R9

DECLARATION

We hereby declare that the Real-Time Research Project report entitled **"Ai Virtual Mouse Using Hand Gestures"** is done under the guidance of **Mrs. G. Swetha Asst. Professor**, Department of Computer Science and Engineering, Teegala Krishna Reddy Engineering College, is submitted in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science and Engineering** from **Jawaharlal Nehru Technological University**, Hyderabad.

This is a record of bonafide work carried out by us in **Teegala Krishna Reddy Engineering College** and the results embodied in this project have not been reproduced or copied from any source.

Submitted by

B. Blessy Priya (23R91A0539)

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompanies the successful completion of any task would be incomplete without the mention of the people who made it possible and whose encouragement and guidance have crowned our efforts with success.

We extend my deep sense of gratitude to **Dr.K. Venkata Murali Mohan**, **Principal** Teegala Krishna Reddy Engineering College, Meerpet, for permitting me to undertake this project.

We are indebted to **Dr.CH.V. Phani Krishna, Professor & Head of the Department**, Computer Science and Engineering, Teegala Krishna Reddy Engineering College, Meerpet for his support and guidance throughout our project.

We are indebted to our guide **Mrs G. SWETHA** , ASSISTANT PROFESSOR, Computer Science Engineering, Teegala Krishna Reddy Engineering College, Meerpet for his/her support and guidance throughout our project.

We are indebted to the project coordinator **Mrs P SWETHA,** Assistant professor, Computer Science and Engineering, Teegala Krishna Reddy Engineering College, Meerpet for her support and guidance throughout our project.

Finally, we express thanks to one and all that have helped me in successfully completing this Realtime research project. Further I would like to thank my family and friends for their moral support and encouragement.

By

B. Blessy Priya (23R91A0539)

ABSTRACT

The project titled **“AI Virtual Mouse Using Hand Gestures”** enhances human-computer interaction by offering a contactless, intuitive alternative to traditional input devices. It allows users to control the mouse pointer and perform actions such as cursor movement, left click, right click, double click, scrolling, and selection using only hand gestures. A webcam captures real-time video, which is processed using computer vision techniques to recognize gestures.The system uses MediaPipe for hand landmark detection, OpenCV for image processing, and PyAutoGUI to simulate mouse actions. By analyzing fingertip positions and spatial relationships, the system maps specific gestures to corresponding mouse functions. Cursor movement is smoothed using interpolation to improve responsiveness and precision.This touch-free approach is ideal for hygienic environments and smart applications, reducing hardware dependency by eliminating the need for a physical mouse. It is particularly useful in presentations, education, medical fields, and home entertainment setups. The project showcases how gesture recognition and automation can significantly enhance modern human-computer interaction through computer vision.

CONTENTS

1. INTRODUCTION

1.1 Description

The project titled **"AI Virtual Mouse Using Hand Gestures"** focuses on transforming the traditional methods of computer interaction by introducing a vision-based, contactless interface that enables users to control mouse functions using simple hand gestures. The objective of this system is to create a more natural, intuitive, and hygienic mode of interaction that does not rely on physical devices like the mouse or keyboard. This approach is especially relevant in environments where touchless interaction is preferred—such as medical facilities, public information kiosks, and assistive technologies for users with physical limitations. The motivation behind this project stems from the growing need for smarter, more accessible technologies that align with modern computing standards and user-centric design. Traditional systems that used color-coded markers or gloves for gesture recognition faced issues such as lighting sensitivity, lack of precision, and the need for additional hardware, making them impractical for widespread use.

To address these issues, this project proposes a robust, markerless system that leverages **MediaPipe's hand tracking solution** to detect 21 hand landmarks in real time using a basic webcam. These hand landmarks are analyzed through gesture recognition algorithms to map user movements to mouse functions such as cursor navigation, left click, right click, double click, and vertical scrolling. The system integrates **OpenCV** for video processing and **PyAutoGUI** for executing mouse events, ensuring smooth and accurate control. Unlike previous systems, it does not depend on external devices or sensors, enhancing both portability and ease of use. This research ultimately contributes to the development of intelligent human-computer interfaces by promoting accessibility, flexibility, and real-time responsiveness, opening doors to future innovations in gesture-controlled computing.

1.2 Existing System

Traditional cursor control mechanisms have long depended on physical hardware such as mice and touchpads. Early advancements attempted to introduce gesture-based systems, primarily relying on external accessories or color markers for gesture recognition. These systems aimed to move towards contactless interaction but suffered from several limitations.

Gesture Techniques in Existing Systems

Several gesture recognition techniques were previously explored using external aids like colored markers or gloves to improve tracking. The following are commonly used gestures and their corresponding mouse actions in existing systems:

Gesture Used	Mapped Function in Older Systems
Index Finger Tracked	Cursor Movement
Thumb + Index Finger Touched	Left Click
Thumb + Middle Finger	Right Click
Thumb + Ring Finger	Double Click
Pinky Finger Raised	Stop Cursor Movement

These systems primarily relied on:

- **Color-coded fingertips or gloves** to assist the camera in identifying hand parts.
- **Threshold-based distances** between fingers for basic click simulations.

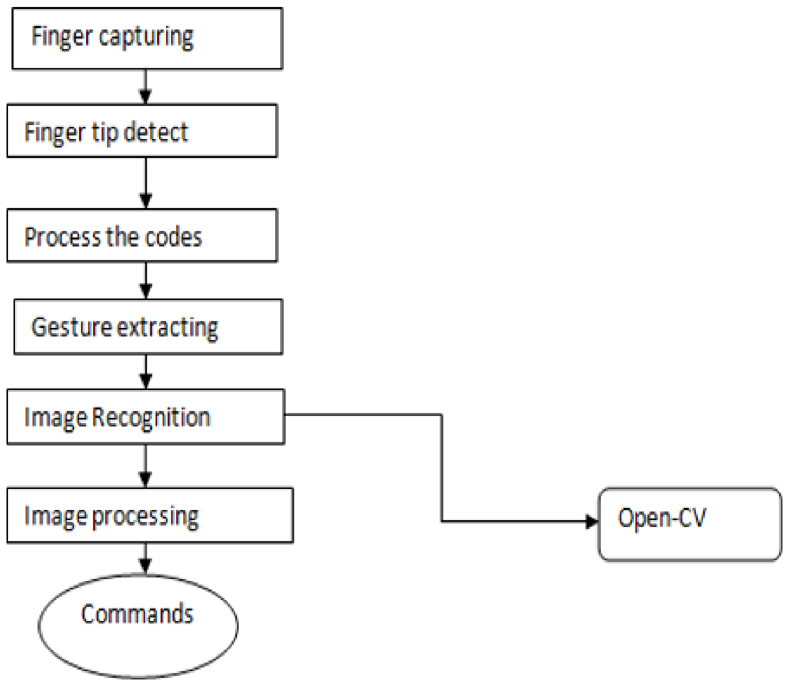


Fig 4: flow chart

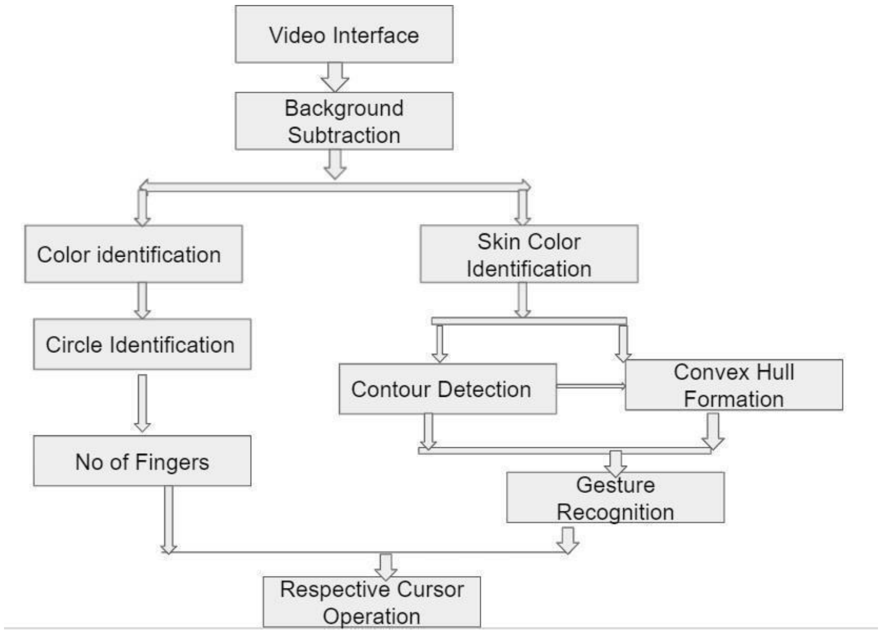
Disadvantages of Existing Systems :

- **Accessory Dependency**: Required color markers, gloves, or other wearable aids for detection.
- **Lighting Sensitivity**: Performance dropped drastically under variable lighting or busy backgrounds.
- **Low Accuracy**: Gesture recognition was often inaccurate due to simple contour or skin-tone tracking.
- **Limited Interaction Range**: Users needed to be within a fixed range of the camera.
- **Lack of Gesture Diversity**: Systems were unable to support complex or fine-grained gestures (e.g., scrolling).

1.3 Proposed System

The proposed **"AI Virtual Mouse Using Hand Gestures"** system addresses the limitations of conventional input devices by offering a contactless, gesture-based interface. Key components and functionalities of the system include:

- **Hand Gesture Recognition**: Utilizing a webcam, the system captures real-time video of the user's hand. **MediaPipe** is employed to detect and track 21 hand landmarks, enabling the recognition of specific gestures associated with mouse functions.
- **Cursor Control**: The position of the index finger is mapped to the on-screen cursor, allowing users to move the cursor by simply moving their hand within the camera's field of view.
- **Mouse Click Operations**: Distinct hand gestures are designated for mouse clicks. For instance:
  - **Left Click**: Bringing the thumb and index finger together.
  - **Right Click**: Bringing the thumb and middle finger together.
  - **Double Click**: Rapid repetition of the left-click gesture.
- **Scrolling Functions**: Vertical movements of the hand with specific finger configurations can simulate scroll up and scroll down actions, facilitating navigation through documents and web pages
- **Gesture Processing and Command Execution**: The system processes the detected gestures using **OpenCV** and translates them into corresponding mouse events via **PyAutoGUI**, effectively emulating traditional mouse behavior.
- **Smooth and Responsive Interaction**: To ensure a fluid user experience, the system incorporates algorithms that minimize cursor jitter and lag, providing responsive and precise control.



Advantages of Proposed Systems :

- **Enhanced Hygiene**: By eliminating the need for physical contact, the system is ideal for use in environments where cleanliness is paramount, such as medical facilities and public kiosks.
- **Improved Accessibility**: Provides an alternative input method for individuals with mobility impairments, enabling them to interact with computers more comfortably.
- **Flexibility and Convenience**: Users can control the computer from a distance, offering greater flexibility in various scenarios, including presentations and home entertainment setups.
- **Cost-Effective Implementation**: Leveraging existing hardware (standard webcams) and open-source software libraries makes the system an economical solution without the need for specialized equipment.

2. SYSTEM ANALYSIS

The "AI Virtual Mouse Using Hand Gestures" project is a novel approach to human-computer interaction, aiming to replace conventional hardware-based mouse systems with an intuitive, gesture-based interface. The system functions by capturing hand gestures through a webcam and translating them into corresponding mouse actions. By utilizing advanced computer vision technologies and gesture recognition algorithms, the project promotes hygienic, contactless interaction with computing devices. This is especially relevant in environments where minimizing physical contact with shared devices is a priority, such as healthcare institutions, smart classrooms, and public information kiosks.

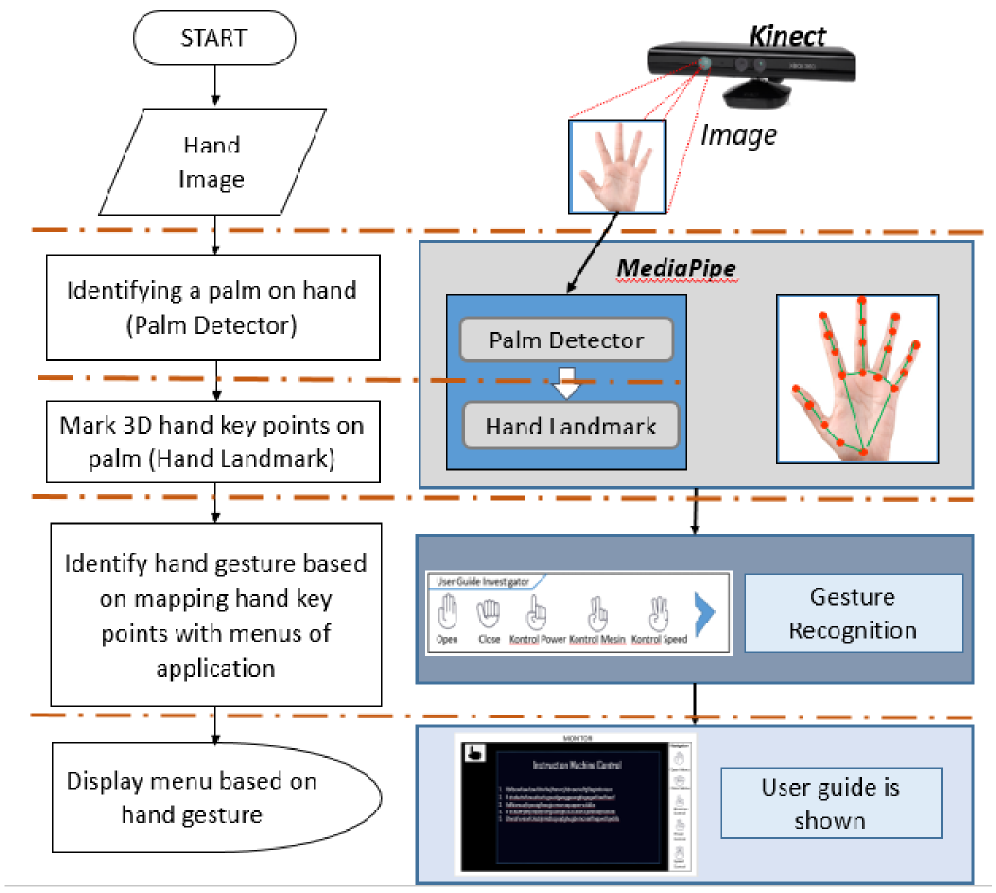
The system is implemented using real-time video processing through a webcam, where hand landmarks are detected using MediaPipe, and the frames are processed via OpenCV to extract spatial information. Gestures are interpreted based on the relationship between finger positions and their movement dynamics. These interpreted gestures are mapped to actions like cursor movement, left/right click, double-click, scrolling, and item selection, all simulated using PyAutoGUI. The system is designed to operate efficiently on standard computing hardware, requiring no external sensors or additional peripherals beyond a webcam.

By replacing the traditional mouse with a virtual interface controlled by natural hand gestures, the system not only reduces hardware dependency but also enhances user experience with its seamless, touch-free interaction. The underlying architecture ensures modularity and extensibility, enabling future enhancements such as support for additional gestures, voice commands, or integration into augmented reality environments

2.1 Architecture

The architecture of the AI Virtual Mouse Using Hand Gestures project is structured into multiple conceptual and functional layers that ensure smooth and real-time human-computer interaction. This layered approach enhances modularity and supports efficient gesture recognition, processing, and mouse action execution.

The system processes live video input from a webcam and translates recognized hand gestures into actionable mouse functions using advanced computer vision and automation techniques.


START
Hand Image
Identifying a palm on hand (Palm Detector)
Mark 3D hand key points on palm (Hand Landmark)
Identify hand gesture based on mapping hand key points with menus of application
Display menu based on hand gesture
Kinect
Image
MediaPipe
Palm Detector
Hand Landmark
Gesture Recognition
User guide is shown

Layered Architecture Overview

1. **Interface Layer**:
   This is the first layer of the system, responsible for capturing real-time video input from the user's environment through a webcam. The frame-by-frame video feed acts as the core input source for all further processing.
2. **Processing Layer**:
   In this layer, the video frames are processed for gesture recognition. The hand region is identified, and MediaPipe's hand tracking model is applied to detect key landmarks on the palm and fingers.

3. **Data Manipulation Layer**:
   After recognizing hand gestures, the system interprets the spatial configurations and dynamic movements of fingers. It maps the detected gestures to specific mouse functions (e.g., cursor movement, left-click, right-click).
4. **Data Layer**:
   Finally, the mapped gestures are translated into system-level mouse actions using PyAutoGUI. Actions like cursor navigation, clicking, and scrolling are executed, allowing the user to control the system without physical contact.

Stepwise Flow of the System:

1. Image Acquisition and Resizing

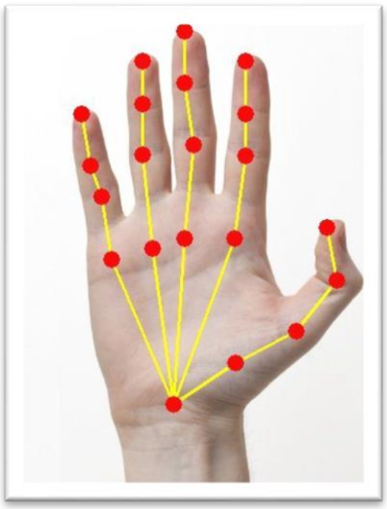
The system starts by capturing an image frame from the webcam, which may contain one or both hands. The frame is resized and preprocessed to suit the input dimensions required by the model.

2. Image Segmentation

Segmentation is used to isolate each hand from the image, helping the system focus only on relevant hand regions. This is essential for analyzing each hand independently and avoiding false detection.

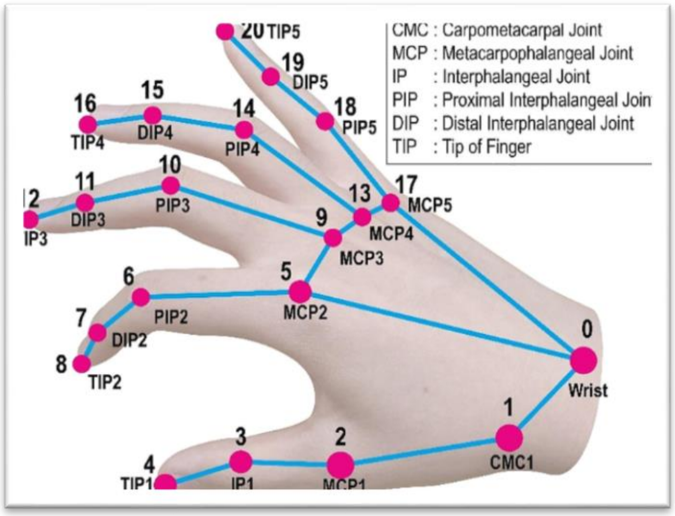
3. Denoising and Background Elimination

To improve gesture recognition accuracy, the system removes noise and separates the hand from complex backgrounds. The palm area is extracted and cropped to retain only the necessary information.



4. Hand Landmark Detection

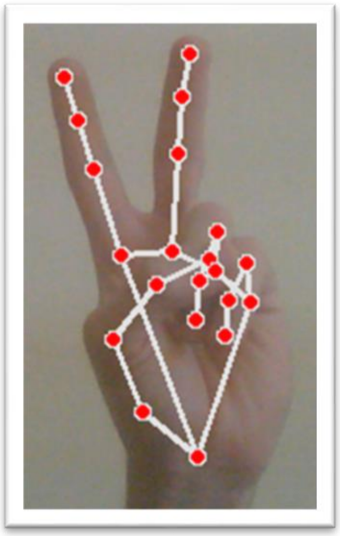
Using MediaPipe, 21 hand landmark points are detected on each hand. These landmarks include the tips, joints, and base of each finger, enabling detailed hand pose estimation.



5. Gesture Interpretation and Mouse Control

Based on the distance and orientation between specific landmarks, various gestures are identified. These gestures are mapped to mouse controls:

- Index finger extended → **Cursor Movement**
- Index + Thumb touching → **Left Click**
- Middle + Thumb touching → **Right Click**
- Thumb + Ring touching → **Double Click**
- Thumb + Pinky upward → **Scroll Down**
- Thumb + Index upward → **Scroll Up**



2.2 System Requirements

The implementation of the AI Virtual Mouse system necessitates specific hardware and software components to ensure optimal performance and seamless user experience. The following subsections delineate these requirements.

2.2.1 Hardware Requirements

To achieve efficient operation, the system requires the following hardware specifications:

- **Processor**: Intel Core i3 or higher
- **RAM**: Minimum 4 GB
- **Hard Disk**: At least 500 GB
- **Webcam**: Built-in or external webcam for real-time hand gesture capture

2.2.2 Software Requirements

The software environment should include:

- **Operating System**: Windows 7
- **Programming Language**: Python
- **Integrated Development Environment (IDE)**: Python IDE 3.8 or VS Code

3. IMPLEMENTATION

The development of the **AI Virtual Mouse** system was carried out using the Python programming language due to its simplicity, flexibility, and strong ecosystem of open-source libraries suitable for image processing, gesture recognition, and automation. The implementation follows a modular and layered architecture to process real-time webcam input and translate hand gestures into mouse events.

3.1 Libraries

The system operates by capturing real-time video through a webcam, detecting hand landmarks using MediaPipe, identifying specific finger gestures, and mapping these gestures to control mouse movements and clicks. Below is the step-by-step workflow:

1) **NumPy :**

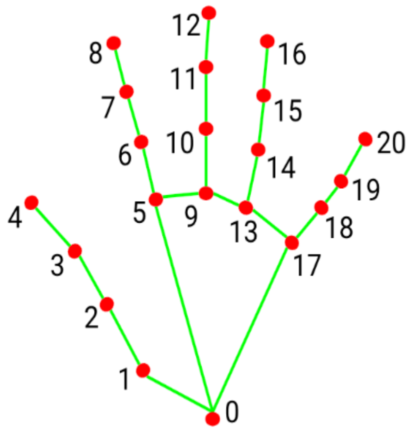
   NumPy is a Python library used for numerical operations. It helps work with arrays and perform mathematical functions quickly. In this project, NumPy is used to handle and process the data received from the hand landmarks. It helps in storing finger positions and calculating distances between points, which are needed to detect gestures.

2) **OpenCV**

   OpenCV is an open-source computer vision library. It is mainly used to process images and videos. Here, OpenCV is used to capture the live video from the webcam and to draw shapes like lines and circles to show hand landmarks on the screen. It also helps in converting the video to the correct color format for other tools to use.

3) **MediaPipe**

MediaPipe is a framework developed by Google for building machine learning pipelines. It provides ready-to-use models for face detection, hand tracking, and more.



0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

- **21 Landmark Points**: It detects and returns 21 key hand landmark coordinates for a single or both hands. These include the tips, intermediate joints, and base of each finger and the palm.
- **Real-Time Performance**: Optimized for mobile and desktop environments, enabling smooth and efficient tracking.
- **Markerless Tracking**: No gloves or sensors are required. Detection works using just a standard webcam.
- **Cross-Platform Support**: Works on Android, iOS, Web (via WebAssembly), desktop, and embedded systems.

4) **AutoPy**

AutoPy is a Python library used to control the keyboard and mouse. In this project, AutoPy is used to move the mouse pointer on the screen and perform clicks based on the detected hand gestures. It makes the computer respond to the gestures as if they were real mouse actions.

3.2 Algorithm

```
Step 1: Initialize webcam and set resolution.
Step 2: Load MediaPipe hand tracking model.
Step 3: While the webcam is active:
        → Capture frame from the camera.
        → Process the frame to detect hand landmarks.
        → If hand is detected:
            ↳ Extract coordinates of finger tips.
            ↳ Determine which fingers are raised or touching.
            ↳ Match gesture to defined actions.
            ↳ Map gesture to corresponding mouse control.
            ↳ Execute mouse function (move, click, double click).
Step 4: Release webcam and close all windows.
```

3.3 Gesture-to-Action Mapping

Gesture Type	**Fingers Involved**	**Action Performed**
Cursor Movement	Index Finger Up	Move mouse cursor
Left Click	Index + Thumb Touch	Perform left click
Right Click	Index + Middle Finger	Perform right click
Double Click	Thumb + Ring Finger Touch	Perform double click
Scroll (Optional)	Index + Middle Finger Up/Down	Scroll the page

4. CODING

Ai_Virtual_Mouse. py:

```
import cv2
import mediapipe as mp
import pyautogui
from pynput.mouse import Controller

# Initialize mouse control
mouse = Controller()
screen_width, screen_height = pyautogui.size()

# Initialize MediaPipe Hands
mp_hands = mp.solutions.hands
hands = mp_hands.Hands(
    min_detection_confidence=0.7,
    min_tracking_confidence=0.7,
    max_num_hands=1
)
draw = mp.solutions.drawing_utils

# Cursor smoothing variables
prev_x, prev_y = 0, 0
alpha = 0.7  # Smoothing factor

def get_distance(p1, p2):
    """Calculate Euclidean distance between two points."""
    return ((p1[0] - p2[0]) ** 2 + (p1[1] - p2[1]) ** 2) ** 0.5

def detect_gesture(frame, landmarks):
    """Detect gestures and control mouse accordingly."""
    global prev_x, prev_y
```

```
if len(landmarks) < 21:
    return
# Finger tip coordinates
thumb_tip = landmarks[4]
index_tip = landmarks[8]
middle_tip = landmarks[12]
ring_tip = landmarks[16]
pinky_tip = landmarks[20]

# Stop cursor movement if pinky is raised
if pinky_tip[1] < landmarks[18][1]:
    cv2.putText(frame, "Cursor Stopped", (50, 50),
                cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 255), 2)
    return

# Cursor movement using index finger
x = int(index_tip[0] * screen_width)
y = int(index_tip[1] * screen_height)
x = int(alpha * prev_x + (1 - alpha) * x)
y = int(alpha * prev_y + (1 - alpha) * y)
pyautogui.moveTo(x, y)
prev_x, prev_y = x, y

# Left Click: Thumb close to Index finger
if get_distance(thumb_tip, index_tip) < 0.05:
    pyautogui.click()
    cv2.putText(frame, "Left Click", (50, 100),
                cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)

# Right Click: Thumb close to Middle finger
elif get_distance(thumb_tip, middle_tip) < 0.05:
```

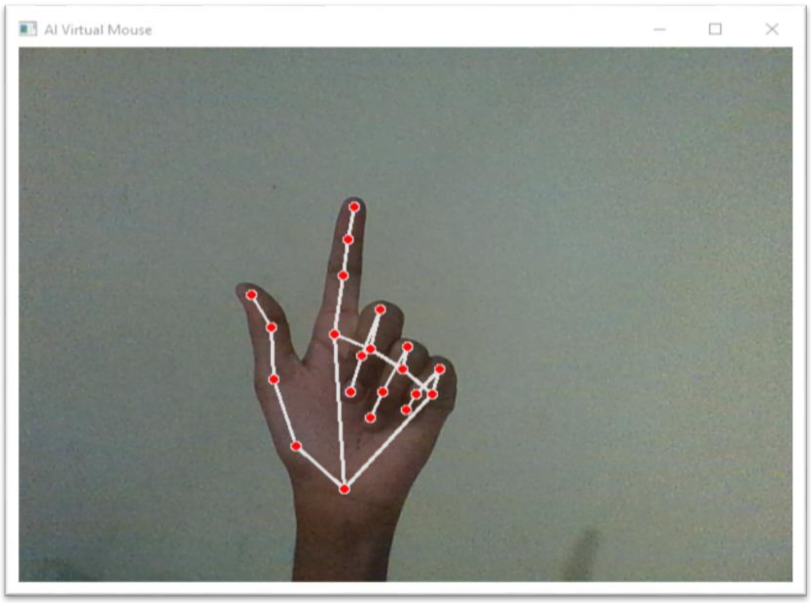
```
            pyautogui.click(button='right')
            cv2.putText(frame, "Right Click", (50, 150),
                        cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)

        # Double Click: Thumb close to Ring finger
        elif get_distance(thumb_tip, ring_tip) < 0.05:
            pyautogui.doubleClick()
            cv2.putText(frame, "Double Click", (50, 200),
                        cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 0, 0), 2)

        # Scroll Up: Index and Middle fingers raised
        elif index_tip[1] < landmarks[6][1] and middle_tip[1] < landmarks[10][1]:
            pyautogui.scroll(10)
            cv2.putText(frame, "Scroll Up", (50, 250),
                        cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 0, 255), 2)

        # Scroll Down: Index and Middle fingers lowered
        elif index_tip[1] > landmarks[5][1] and middle_tip[1] > landmarks[9][1]:
            pyautogui.scroll(-10)
            cv2.putText(frame, "Scroll Down", (50, 300),
                        cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 0), 2)

def main():
    cap = cv2.VideoCapture(0)
    if not cap.isOpened():
        print("Camera not detected. Please check your webcam.")
        return

    try:
        while True:
            ret, frame = cap.read()
            if not ret:
                break

            frame = cv2.flip(frame, 1)
            frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
            results = hands.process(frame_rgb)

            landmark_list = []
            if results.multi_hand_landmarks:
                hand_landmarks = results.multi_hand_landmarks[0]
                draw.draw_landmarks(frame, hand_landmarks,
mp_hands.HAND_CONNECTIONS)
                for lm in hand_landmarks.landmark:
                    landmark_list.append((lm.x, lm.y))

            detect_gesture(frame, landmark_list)
            cv2.imshow("AI Virtual Mouse", frame)
            if cv2.waitKey(1) & 0xFF == ord('q'):
                break
    finally:
        cap.release()
        cv2.destroyAllWindows()

if __name__ == "__main__":
    main()
```

5. Output Screens

A. Cursor Movement

➤ **Gesture**: Only index finger is raised and moved freely.

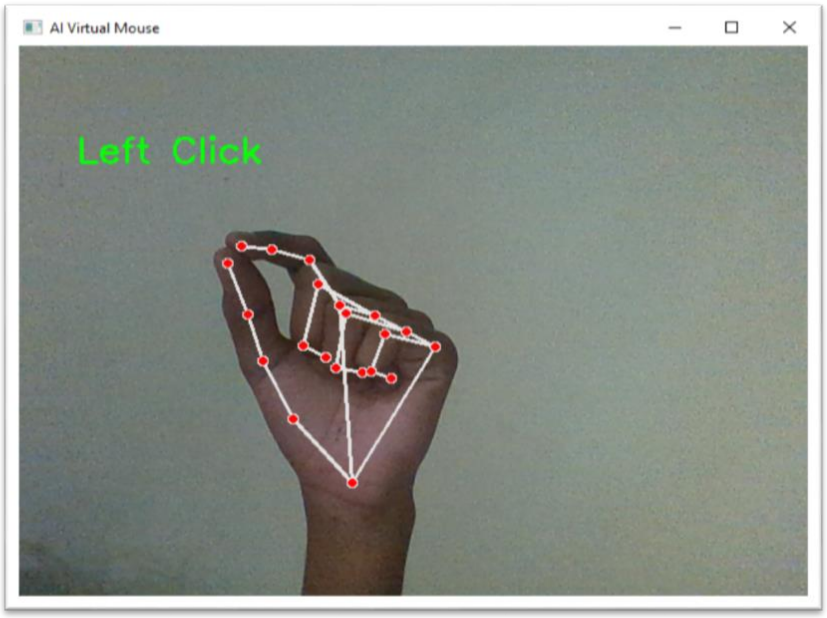
➤ **Action**: Controls the mouse cursor in real-time across the screen.



B. Left Click

➤ **Gesture**: Thumb touches the tip of the index finger.

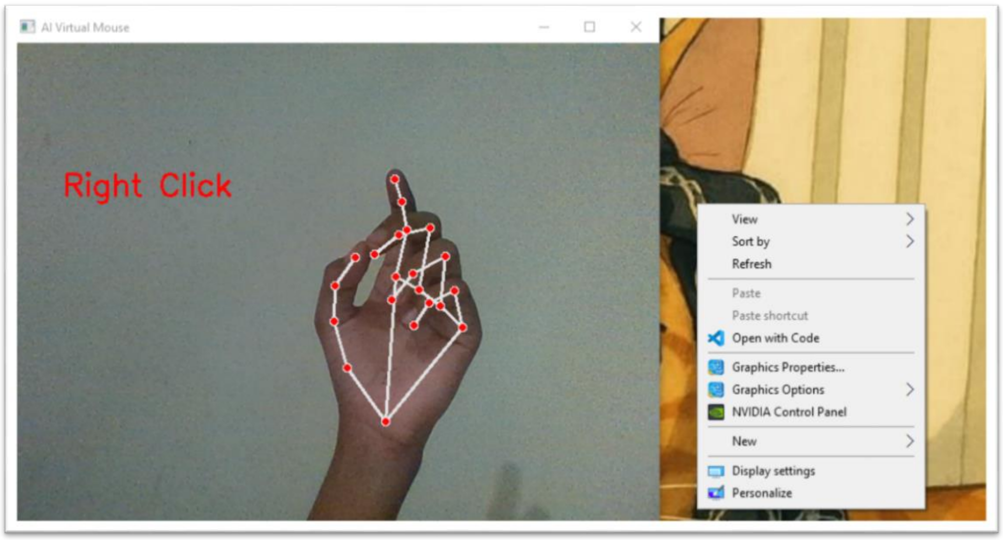
➤ **Action**: Performs a left-click operation on the screen.



C. Right Click

➤ **Gesture**: Thumb touches the tip of the middle finger.

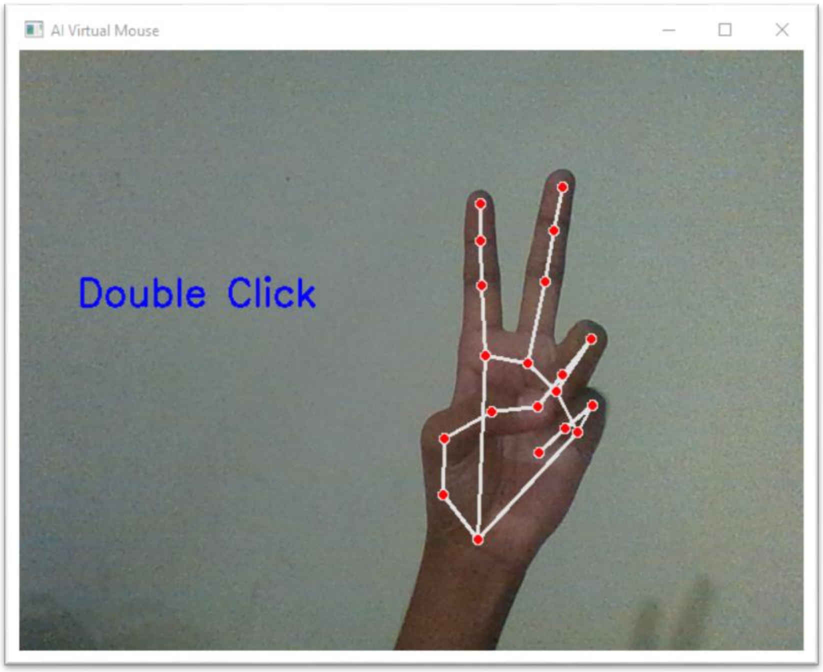
➤ **Action**: Triggers a right-click operation.



D. Double Click

➤ **Gesture**: Thumb touches the tip of the ring finger.

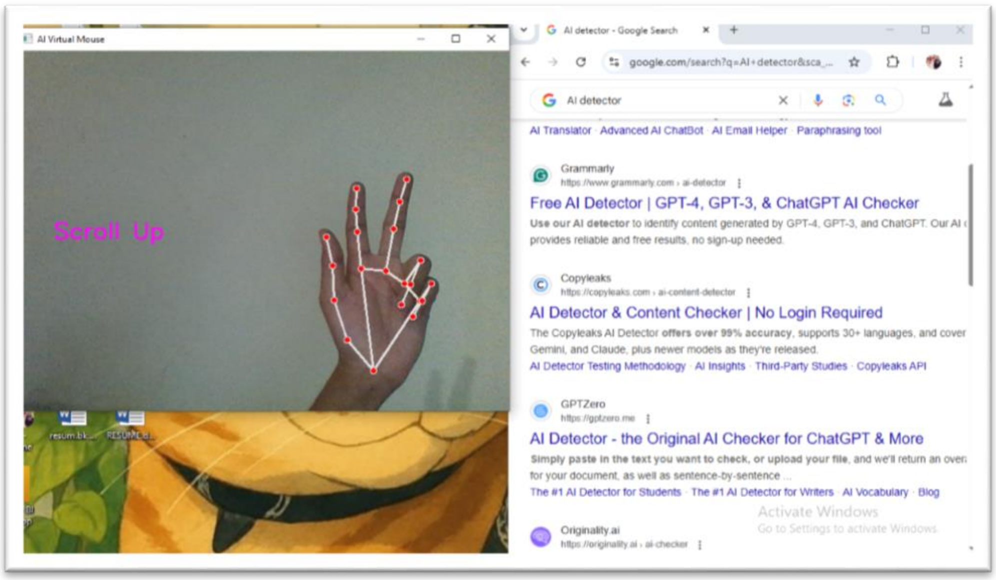
➤ **Action**: Executes a double-click, similar to opening files or apps.



E. Scroll Up

➤ **Gesture**: Both index and middle fingers raised upright.

➤ **Action**: Scrolls the screen upward, simulating mouse wheel up.


AI Virtual Mouse
Scroll Up
AI detector - Google Search
google.com/search?q=AI+detector&sca_...
AI detector

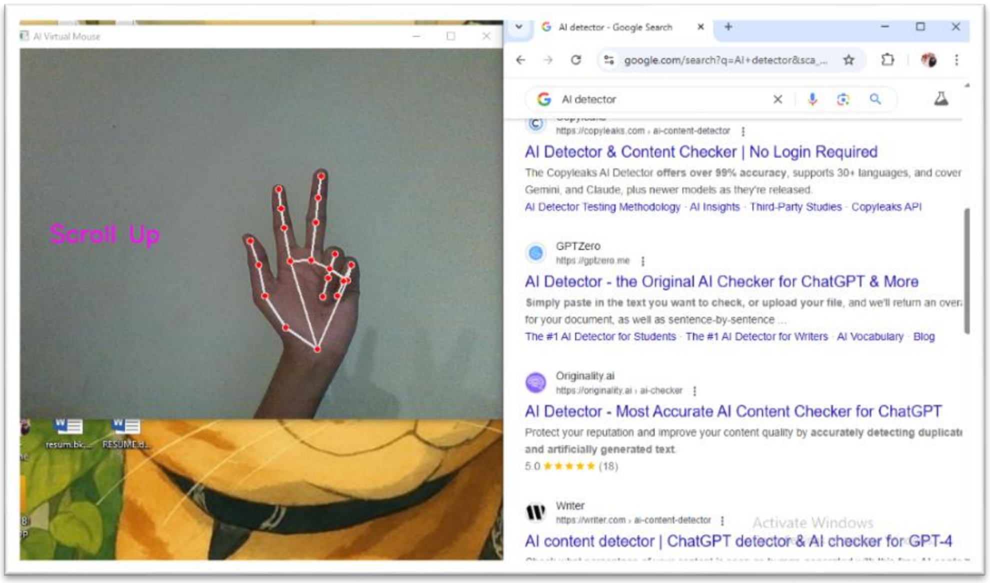
AI Translator · Advanced AI ChatBot · AI Email Helper · Paraphrasing tool

Grammarly
https://www.grammarly.com › ai-detector
Free AI Detector | GPT-4, GPT-3, & ChatGPT AI Checker
Use our AI detector to identify content generated by GPT-4, GPT-3, and ChatGPT. Our AI provides reliable and free results, no sign-up needed.

Copyleaks
https://copyleaks.com › ai-content-detector
AI Detector & Content Checker | No Login Required
The Copyleaks AI Detector offers over 99% accuracy, supports 30+ languages, and cover Gemini, and Claude, plus newer models as they're released.
AI Detector Testing Methodology · AI Insights · Third-Party Studies · Copyleaks API

GPTZero
https://gptzero.me
AI Detector - the Original AI Checker for ChatGPT & More
Simply paste in the text you want to check, or upload your file, and we'll return an over for your document, as well as sentence-by-sentence ...
The #1 AI Detector for Students · The #1 AI Detector for Writers · AI Vocabulary · Blog

Originality.ai
https://originality.ai › ai-checker


AI Virtual Mouse
Scroll Up
AI detector - Google Search
google.com/search?q=AI+detector&sca_...
AI detector

https://copyleaks.com › ai-content-detector
AI Detector & Content Checker | No Login Required
The Copyleaks AI Detector offers over 99% accuracy, supports 30+ languages, and cover Gemini, and Claude, plus newer models as they're released.
AI Detector Testing Methodology · AI Insights · Third-Party Studies · Copyleaks API

GPTZero
https://gptzero.me
AI Detector - the Original AI Checker for ChatGPT & More
Simply paste in the text you want to check, or upload your file, and we'll return an over for your document, as well as sentence-by-sentence ...
The #1 AI Detector for Students · The #1 AI Detector for Writers · AI Vocabulary · Blog

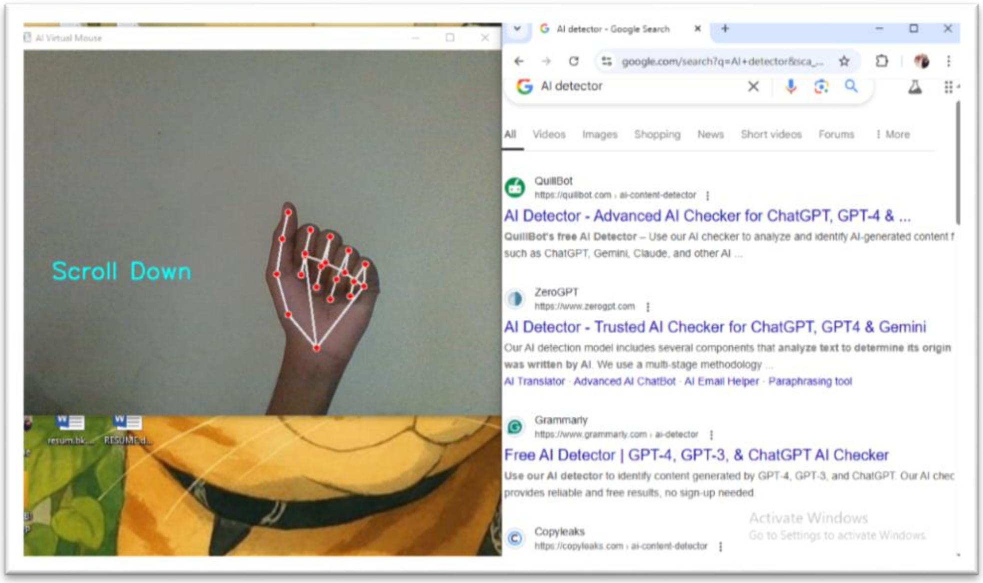
Originality.ai
https://originality.ai › ai-checker
AI Detector - Most Accurate AI Content Checker for ChatGPT
Protect your reputation and improve your content quality by accurately detecting duplicate and artificially generated text.
5.0 ★★★★★ (18)

Writer
https://writer.com › ai-content-detector
AI content detector | ChatGPT detector & AI checker for GPT-4

F. Scroll Down

➤ **Gesture**: Index and middle fingers bent downward.

➤ **Action**: Scrolls the screen downward, like mouse wheel down.

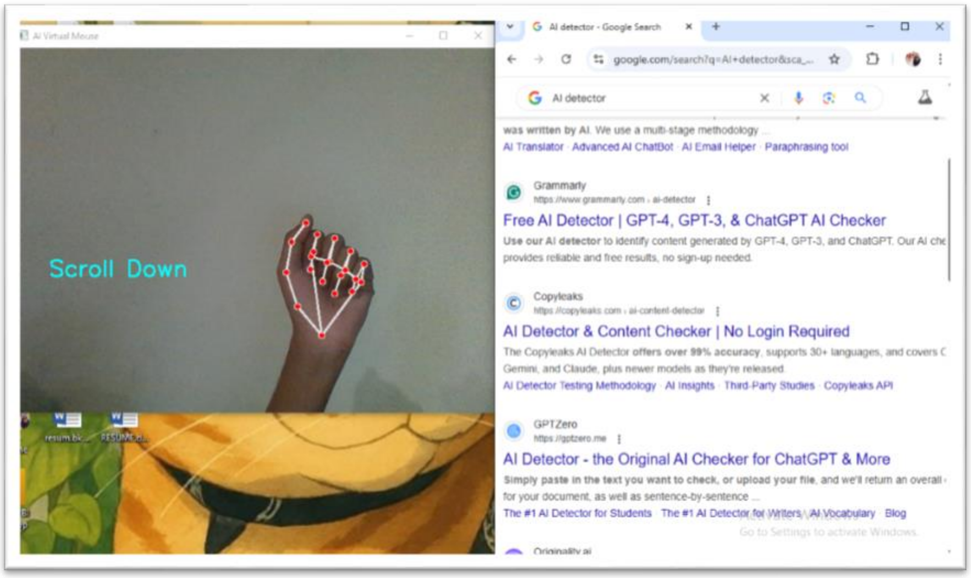


QuillBot
https://quillbot.com › ai-content-detector
AI Detector - Advanced AI Checker for ChatGPT, GPT-4 & ...
QuillBot's free AI Detector – Use our AI checker to analyze and identify AI-generated content f
such as ChatGPT, Gemini, Claude, and other AI ...

ZeroGPT
https://www.zerogpt.com
AI Detector - Trusted AI Checker for ChatGPT, GPT4 & Gemini
Our AI detection model includes several components that analyze text to determine its origin
was written by AI. We use a multi-stage methodology ...
AI Translator · Advanced AI ChatBot · AI Email Helper · Paraphrasing tool

Grammarly
https://www.grammarly.com › ai-detector
Free AI Detector | GPT-4, GPT-3, & ChatGPT AI Checker
Use our AI detector to identify content generated by GPT-4, GPT-3, and ChatGPT. Our AI chec
provides reliable and free results, no sign-up needed.

Copyleaks
https://copyleaks.com › ai-content-detector


AI Virtual Mouse
Scroll Down
AI detector - Google Search
google.com/search?q=AI+detector&sca_...
AI detector
Activate Windows
Go to Settings to activate Windows.

was written by AI. We use a multi-stage methodology ...
AI Translator · Advanced AI ChatBot · AI Email Helper · Paraphrasing tool

Grammarly
https://www.grammarly.com › ai-detector
Free AI Detector | GPT-4, GPT-3, & ChatGPT AI Checker
Use our AI detector to identify content generated by GPT-4, GPT-3, and ChatGPT. Our AI che
provides reliable and free results, no sign-up needed.

Copyleaks
https://copyleaks.com › ai-content-detector
AI Detector & Content Checker | No Login Required
The Copyleaks AI Detector offers over 99% accuracy, supports 30+ languages, and covers C
Gemini, and Claude, plus newer models as they're released.
AI Detector Testing Methodology · AI Insights · Third-Party Studies · Copyleaks API

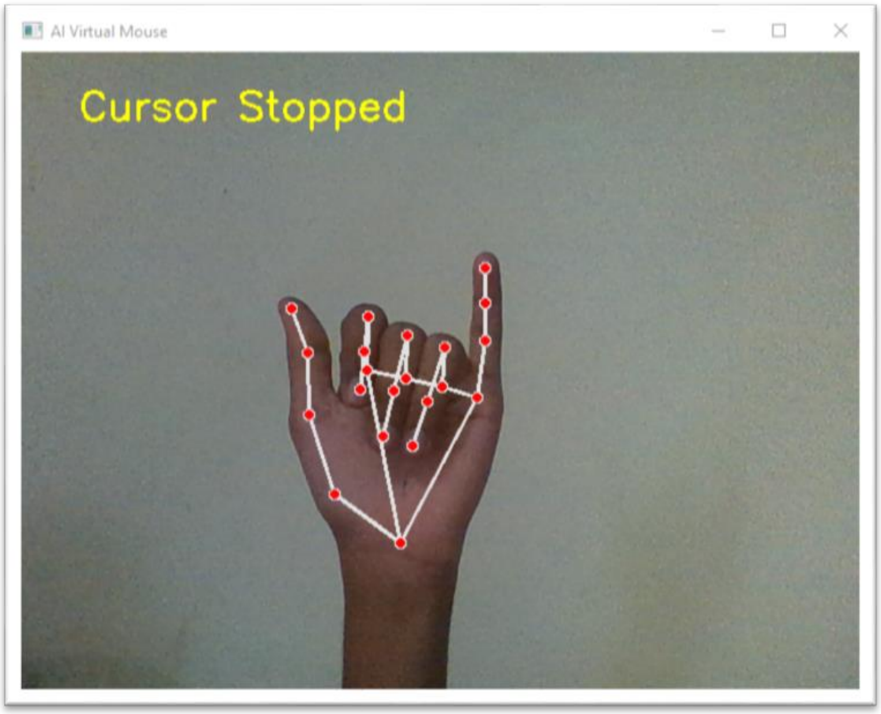
GPTZero
https://gptzero.me
AI Detector - the Original AI Checker for ChatGPT & More
Simply paste in the text you want to check, or upload your file, and we'll return an overall
for your document, as well as sentence-by-sentence ...
The #1 AI Detector for Students · The #1 AI Detector for Writers · AI Vocabulary · Blog

Originality.ai

G. Stop Cursor Movement

➤ **Gesture**: Pinky finger raised while other fingers remain low.

➤ **Action**: Halts cursor movement, pausing interaction temporarily.



6. Conclusion

The project titled **"AI Virtual Mouse Using Hand Gestures"** presents a novel and effective approach to human-computer interaction by enabling mouse control using only hand gestures and a standard webcam. Through the integration of computer vision tools such as MediaPipe, OpenCV, and PyAutoGUI, the system accurately detects and interprets hand landmarks to perform real-time mouse functions including cursor movement, left and right clicks, double click, and scrolling. Unlike traditional systems that rely on physical contact or accessory-based gesture recognition, this solution provides a contactless, hygienic, and accessible alternative that is both cost-effective and user-friendly. The implementation demonstrates smooth and responsive control without requiring special hardware, making it suitable for use in various domains such as healthcare, education, and smart environments. This project highlights the practical application of artificial intelligence in enhancing everyday computing, and sets the foundation for future developments in touchless, gesture-driven interfaces

7. Bibliography

1. Amardip Ghodichor, Binitha Chirakattu. “Virtual Mouse Using Hand Gesture and Color Detection.” International Journal of Computer Applications, Vol. 128, No. 11, October 2015.
2. Chhoriya, P., Paliwal, G., Badhan, P. “Image Processing Based Color Detection.” International Journal of Emerging Technology and Advanced Engineering, Vol. 3, Issue 4, 2013.
3. Rhitivij Parasher, Preksha Pareek. “Event Triggering Using Hand Gesture with OpenCV.” International Research Journal of Engineering and Technology (IRJET), Vol. 2, February 2016.
4. Ahmed Siddique, Abhishek Kommera, Divya Verma. “Simulation of Mouse Using Image Processing via Convex Hull Method.” International Journal of Engineering Research and Applications (IJERA), Vol. 4, Issue 3, March 2016.
5. Kalyani Pendke, Prasanna Khuje, Smita Narnaware, Shweta Thool, Sachin Nimje. “Virtual Mouse Using Hand Gesture.” International Journal of Computer Science and Mobile Computing (IJCSMC), Vol. 4, Issue 3, March 2015.
6. Abhilash S. S., Lisho Thomas, Naveen Wilson, Chaithanya C. “Virtual Mouse Using Hand Gesture.” International Research Journal of Engineering and Technology (IRJET), Vol. 5, Issue 4, April 2018.
7. Abdul Khaliq, A. Shahid Khan. “Virtual Mouse Implementation Using Color Pointer Detection.” International Journal of Electrical Electronics & Computer Science Engineering (IJEECSE), Vol. 2, Issue 4, August 2015.
8. Erdem E., Yardimci Y., Atalay V., Cetin A. E. “Computer Vision Based Mouse.” IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASS), 2002.