Chapter

**Virtual Mouse**

**AIM:**

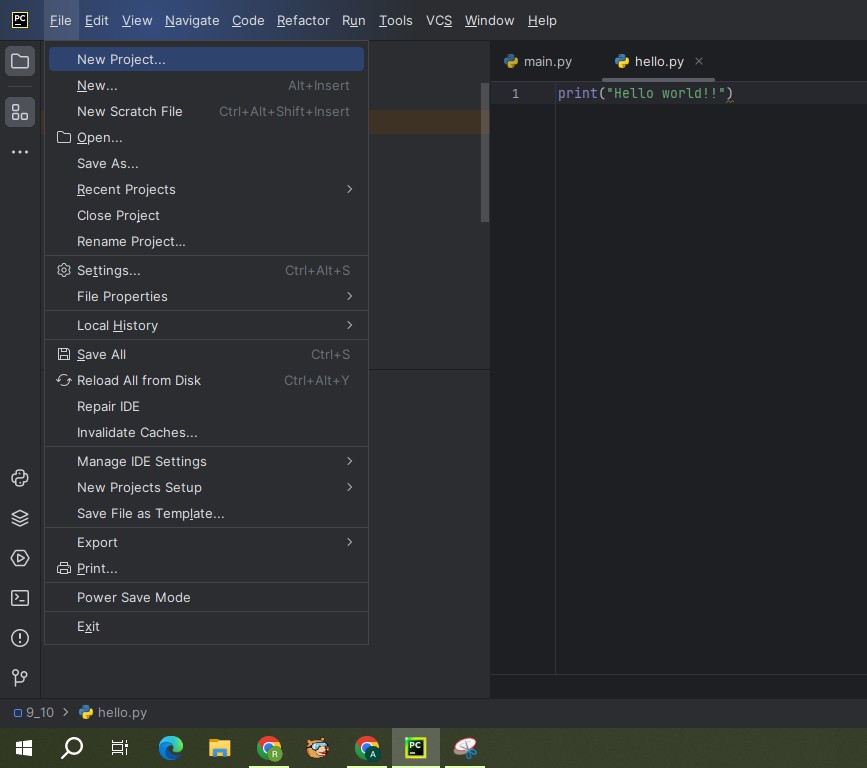
The aim of the Virtual Mouse Using AI project is to enable users to control their computer’s mouse using hand gestures, eliminating the need for a physical mouse. By utilizing computer vision and machine learning, the system can detect and track hand movements in real-time. These movements are then mapped to the screen, allowing the user to move the cursor naturally. Specific gestures, such as pinching, can be recognized to simulate mouse clicks. This technology enhances accessibility, particularly for individuals with physical disabilities, and provides a hygienic, touch-free interface suitable for various environments. Overall, it offers a more intuitive and innovative way to interact with computers.

**Introduction:**

The Virtual Mouse Using AI project leverages advanced computer vision and machine learning techniques to provide a touch-free method of controlling a computer’s mouse. By tracking hand gestures in real-time using a webcam, this system enables users to move the mouse cursor and perform clicks without needing a physical mouse. This innovative approach offers numerous benefits, including enhanced accessibility for individuals with physical disabilities and a hygienic alternative in environments where minimizing physical contact is crucial. The technology translates hand movements into cursor actions and recognizes specific gestures for clicking, making computer interaction more intuitive and seamless. Through this project, we aim to demonstrate the potential of AI in creating more user-friendly and accessible computing experiences.

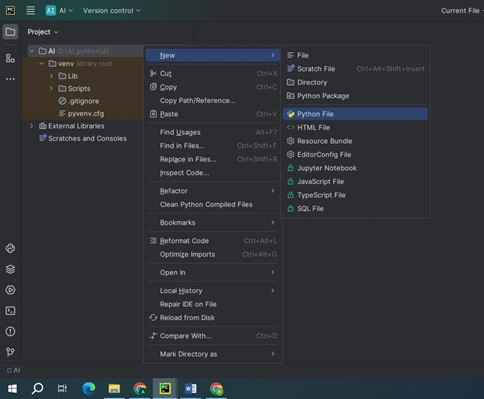
**Procedure:**

* Open PyCharm IDE software.
* Go to Menu —> File —> New Project.

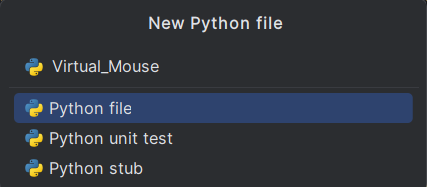


* A screenshot of a program

  Description automatically generatedA window will be appeared as below. Change the Name and Location of the project as per our requirement. Select custom environment for interpreter type. Select Generate new for Environment. Select Virtualenv for Type. Select latest version of python for Base python i.e., Python 3.11.6. And click on create
* Once our project is created, right click on project. Go to —> New —> Python File.



* Give the name to the python file: Virtual\_Mouse.



* Once the file is created, copy below given code to Virtual Mouse.

**Virtual Mouse** **code:**

import cv2

import mediapipe as mp

import pyautogui

import random

import util

from pynput.mouse import Button, Controller

mouse = Controller()

screen\_width, screen\_height = pyautogui.size()

mpHands = mp.solutions.hands

hands = mpHands.Hands(

static\_image\_mode=False,

model\_complexity=1,

min\_detection\_confidence=0.7,

min\_tracking\_confidence=0.7,

max\_num\_hands=1

)

def find\_finger\_tip(processed):

if processed.multi\_hand\_landmarks:

hand\_landmarks = processed.multi\_hand\_landmarks[0] # Assuming only one hand is detected

index\_finger\_tip = hand\_landmarks.landmark[mpHands.HandLandmark.INDEX\_FINGER\_TIP]

return index\_finger\_tip

return None

def move\_mouse(index\_finger\_tip):

if index\_finger\_tip is not None:

x = int(index\_finger\_tip.x \* screen\_width)

y = int(index\_finger\_tip.y \* screen\_height)

pyautogui.moveTo(x, y)

def is\_left\_click(landmark\_list, thumb\_index\_dist):

return (

util.get\_angle(landmark\_list[5], landmark\_list[6], landmark\_list[8]) < 50 and

util.get\_angle(landmark\_list[9], landmark\_list[10], landmark\_list[12]) > 90 and

thumb\_index\_dist > 50

)

def is\_right\_click(landmark\_list, thumb\_index\_dist):

return (

util.get\_angle(landmark\_list[9], landmark\_list[10], landmark\_list[12]) < 50 and

util.get\_angle(landmark\_list[5], landmark\_list[6], landmark\_list[8]) > 90 and

thumb\_index\_dist > 50

)

def is\_double\_click(landmark\_list, thumb\_index\_dist):

return (

util.get\_angle(landmark\_list[5], landmark\_list[6], landmark\_list[8]) < 50 and

util.get\_angle(landmark\_list[9], landmark\_list[10], landmark\_list[12]) < 50 and

thumb\_index\_dist > 50

)

def is\_screenshot(landmark\_list, thumb\_index\_dist):

return (

util.get\_angle(landmark\_list[5], landmark\_list[6], landmark\_list[8]) < 50 and

util.get\_angle(landmark\_list[9], landmark\_list[10], landmark\_list[12]) < 50 and

thumb\_index\_dist < 50

)

def detect\_gesture(frame, landmark\_list, processed):

if len(landmark\_list) >= 21:

index\_finger\_tip = find\_finger\_tip(processed)

thumb\_index\_dist = util.get\_distance([landmark\_list[4], landmark\_list[5]])

if util.get\_distance([landmark\_list[4], landmark\_list[5]]) < 50 and util.get\_angle(landmark\_list[5], landmark\_list[6], landmark\_list[8]) > 90:

move\_mouse(index\_finger\_tip)

elif is\_left\_click(landmark\_list, thumb\_index\_dist):

mouse.press(Button.left)

mouse.release(Button.left)

cv2.putText(frame, "Left Click", (50, 50), cv2.FONT\_HERSHEY\_SIMPLEX, 1, (0, 255, 0), 2)

elif is\_right\_click(landmark\_list, thumb\_index\_dist):

mouse.press(Button.right)

mouse.release(Button.right)

cv2.putText(frame, "Right Click", (50, 50), cv2.FONT\_HERSHEY\_SIMPLEX, 1, (0, 0, 255), 2)

elif is\_double\_click(landmark\_list, thumb\_index\_dist):

pyautogui.doubleClick()

cv2.putText(frame, "Double Click", (50, 50), cv2.FONT\_HERSHEY\_SIMPLEX, 1, (255, 255, 0), 2)

elif is\_screenshot(landmark\_list, thumb\_index\_dist):

im1 = pyautogui.screenshot()

label = random.randint(1, 1000)

im1.save(f'my\_screenshot\_{label}.png')

cv2.putText(frame, "Screenshot Taken", (50, 50), cv2.FONT\_HERSHEY\_SIMPLEX, 1, (255, 255, 0), 2)

def main():

draw = mp.solutions.drawing\_utils

cap = cv2.VideoCapture(0)

try:

while cap.isOpened():

ret, frame = cap.read()

if not ret:

break

frame = cv2.flip(frame, 1)

frameRGB = cv2.cvtColor(frame, cv2.COLOR\_BGR2RGB)

processed = hands.process(frameRGB)

landmark\_list = []

if processed.multi\_hand\_landmarks:

hand\_landmarks = processed.multi\_hand\_landmarks[0] # Assuming only one hand is detected

draw.draw\_landmarks(frame, hand\_landmarks, mpHands.HAND\_CONNECTIONS)

for lm in hand\_landmarks.landmark:

landmark\_list.append((lm.x, lm.y))

detect\_gesture(frame, landmark\_list, processed)

cv2.imshow('Frame', frame)

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

break

finally:

cap.release()

cv2.destroyAllWindows()

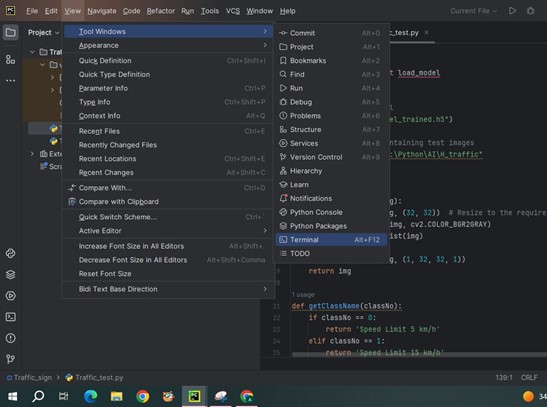
if \_\_name\_\_ == '\_\_main\_\_':

main()

**Libraries to install:**

Ensure you have the following libraries installed before running the code:

Go to Menu —> View —> Tool Windows —> Terminal



* **Pyautogui:**

PyAutoGUI is a Python library for automating mouse and keyboard actions. It lets you move the mouse cursor, click, and type text through code. Key functions include moveTo() for moving the mouse to specific screen coordinates and click() for simulating mouse clicks. Install Pyautogui .library, type the below command in terminal:

pip install Pyautogui

A screen shot of a computer program

Description automatically generated

* **Random**:  
   Actually, random is not a library that you install with pip; it's a built-in module in Python. The random module provides functions for generating random numbers and making random selections. Install to type the below command in terminal:

pip install random

* **OpenCV**:

OpenCV (cv2) is a Python library for computer vision tasks, offering tools for image and video processing. It supports reading, writing, and manipulating images and videos, along with features like object detection and facial recognition. OpenCV's versatility makes it suitable for a wide range of applications, from basic image operations to advanced machine learning integration. Its cross-platform nature and extensive documentation make it accessible for developers aiming to incorporate vision-based functionalities into their projects efficiently.Install to type the below command in terminal:

pip install opencv

* **MediaPipe**:

**MediaPipe** is a comprehensive open-source framework developed by Google for building machine learning pipelines to process perceptual data, such as video and audio. It offers a wide range of pre-built components and tools to facilitate the development of complex machine learning models and applications, particularly in the fields of computer vision, augmented reality, and audio processing. MediaPipe provides efficient, scalable solutions for tasks like object detection, facial recognition, hand tracking, pose estimation, and more, making it suitable for both research and commercial applications requiring real-time data processing and analysis.Install to type the below command in terminal:

pip install **mediapipe**

**Running:**

* Run the Virtual Mouse code

**Output:**

A screenshot of a computer

Description automatically generated

**Conclusion:**

Object tracking using AI represents a pivotal advancement in computer vision, enabling robust real-time detection and monitoring of objects across various applications like surveillance, autonomous vehicles, and augmented reality. By harnessing deep learning models and sophisticated algorithms, AI-driven tracking systems can accurately handle diverse environmental conditions and complex scenarios. The technology's ability to track objects seamlessly and efficiently enhances safety, automation, and user interaction in modern digital environments. As AI continues to evolve, integrating object tracking capabilities promises to further revolutionize industries reliant on real-time spatial awareness and intelligent decision-making based on visual data.