Chapter

**Object Tracking**

**AIM:**

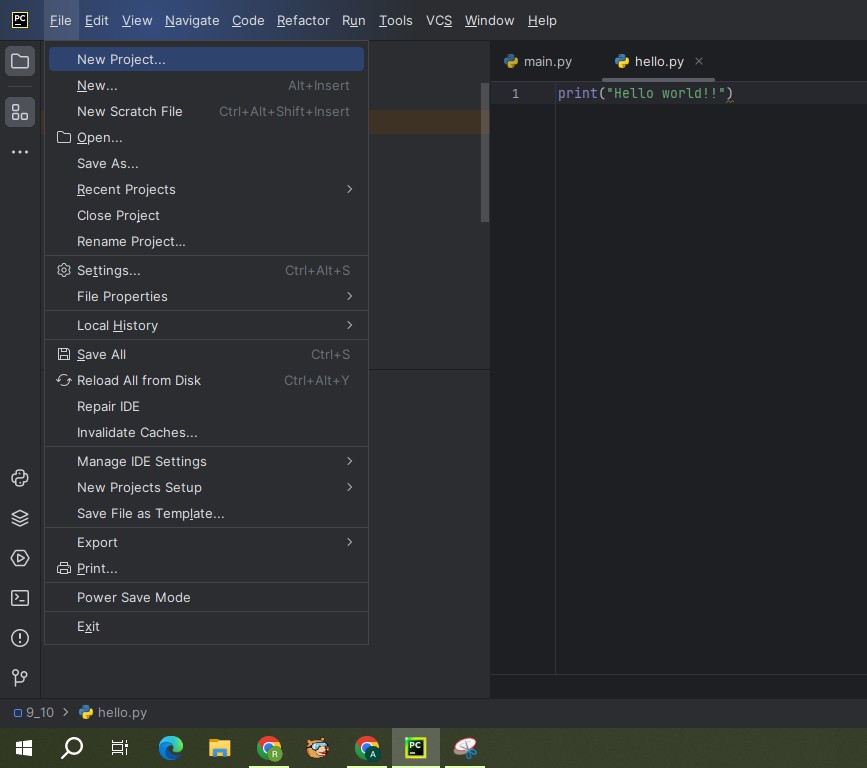
object tracking AI project is to develop a robust and accurate AI model capable of detecting and tracking objects in real-time across various conditions. The system should handle multiple objects simultaneously with low latency, ensuring smooth and continuous monitoring. It will include a user-friendly interface providing clear visualizations and actionable insights. The project also focuses on scalability and flexibility to adapt to different types of objects and environments. Extensive testing will be conducted to ensure the system's robustness and reliability under real-world challenges.

**Introduction:**

Object tracking using AI is a powerful technique in computer vision that involves detecting and following objects of interest within video sequences. This technology has a wide range of applications, including surveillance, autonomous driving, and interactive systems. By leveraging advanced algorithms and deep learning models, AI-powered object tracking can achieve high accuracy and real-time performance. The goal of this project is to develop an efficient and reliable object tracking system that can handle various environmental conditions and provide robust results. Through this project, we aim to demonstrate the potential of AI in enhancing automated visual monitoring and analysis.

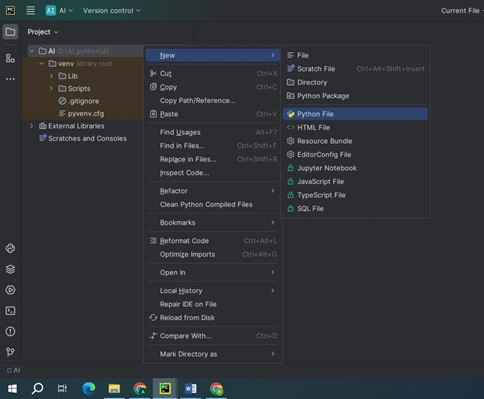
**Procedure:**

* 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: Object Tracking.

A screenshot of a computer program

Description automatically generated

* Once the file is created, copy below given code to Object Tracking.

**Object Tracking code:**

from collections import deque

from imutils.video import VideoStream

import numpy as np

import argparse

import imutils

import cv2

import time

ap = argparse.ArgumentParser()

ap.add\_argument("-v", "--video", help="path to the (optional) video file")

ap.add\_argument("-b", "--buffer", type=int, default=64,

help="max buffer size")

args = vars(ap.parse\_args())

greenLower = (29, 86, 6)

greenUpper = (64, 255, 255)

pts = deque(maxlen=args["buffer"])

if not args.get("video", False):

camera = cv2.VideoCapture(0)

else:

camera = cv2.VideoCapture(args["video"])

# allow the camera or video file to warm up

time.sleep(2.0)

while True:

(grabbed, frame) = camera.read()

if args.get("video") and not grabbed:

break

frame = imutils.resize(frame, width=600)

blurred = cv2.GaussianBlur(frame, (11, 11), 0)

hsv = cv2.cvtColor(frame, cv2.COLOR\_BGR2HSV)

mask = cv2.inRange(hsv, greenLower, greenUpper)

mask = cv2.erode(mask, None, iterations=2)

mask = cv2.dilate(mask, None, iterations=2)

cnts = cv2.findContours(mask.copy(), cv2.RETR\_EXTERNAL,

cv2.CHAIN\_APPROX\_SIMPLE)

cnts = imutils.grab\_contours(cnts)

center = None

if len(cnts) > 0:

c = max(cnts, key=cv2.contourArea)

((x, y), radius) = cv2.minEnclosingCircle(c)

M = cv2.moments(c)

center = (int(M["m10"] / M["m00"]), int(M["m01"] / M["m00"]))

if radius > 10:

cv2.circle(frame, (int(x), int(y)), int(radius), (0, 255, 255), 2)

cv2.circle(frame, center, 5, (0, 0, 255), -1)

pts.appendleft(center)

# loop over the set of tracked points

for i in range(1, len(pts)):

# if either of the tracked points are None, ignore

# them

if pts[i - 1] is None or pts[i] is None:

continue

# otherwise, compute the thickness of the line and

# draw the connecting lines

thickness = int(np.sqrt(args["buffer"] / float(i + 1)) \* 2.5)

cv2.line(frame, pts[i - 1], pts[i], (0, 0, 255), thickness)

cv2.imshow("Frame", frame)

cv2.imshow("Mask", mask)

key = cv2.waitKey(1) & 0xFF

if key == ord("q"):

break

camera.release()

cv2.destroyAllWindows()

* Give the name to the python file: Object Finding.

A screenshot of a computer program

Description automatically generated

**Object Finding code:**

from collections import deque

import numpy as np

import argparse

import imutils

import cv2

ap = argparse.ArgumentParser()

ap.add\_argument("-v", "--video", help="path to the (optional) video file")

args = vars(ap.parse\_args())

greenLower = (29, 86, 6)

greenUpper = (64, 255, 255)

if not args.get("video", False):

camera = cv2.VideoCapture(1)

else:

camera = cv2.VideoCapture(args["video"])

while True:

(grabbed, frame) = camera.read()

if args.get("video") and not grabbed:

break

frame = imutils.resize(frame, width=600)

hsv = cv2.cvtColor(frame, cv2.COLOR\_BGR2HSV)

mask = cv2.inRange(hsv, greenLower, greenUpper)

mask = cv2.erode(mask, None, iterations=2)

mask = cv2.dilate(mask, None, iterations=2)

cnts = cv2.findContours(mask.copy(), cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)[-2]

center = None

if len(cnts) > 0:

c = max(cnts, key=cv2.contourArea)

((x, y), radius) = cv2.minEnclosingCircle(c)

if radius > 10:

cv2.circle(frame, (int(x), int(y)), int(radius), (0, 255, 255), 2)

cv2.imshow("Frame", frame)

cv2.imshow("Mask", mask)

key = cv2.waitKey(1) & 0xFF

if key == ord("q"):

break

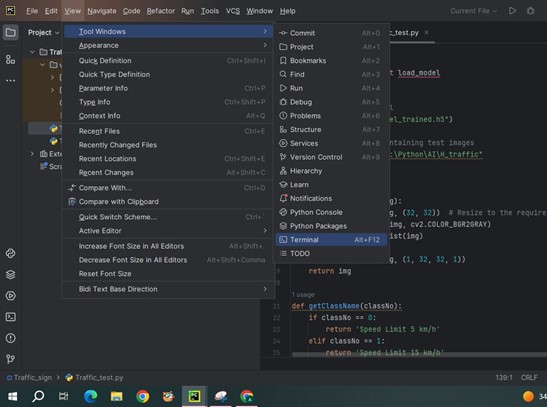
camera.release()

cv2.destroyAllWindows()

**Libraries to install:**

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

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



* **NumPy:**

NumPy (Numerical Python) is a powerful open-source library for numerical computing in Python. It provides support for arrays, matrices, and a wide range of mathematical functions to operate on these data structures. NumPy is a fundamental package for scientific computing and is widely used in data science, machine learning, engineering, and other technical fields.library, type the below command in terminal:

pip install NumPy

A screenshot of a computer

Description automatically generated

* **Imutils**:  
   Install to type the below command in terminal:

pip install imutils

* **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

**Running:**

* Run the Object Tracking code

**Output:**

A screenshot of a computer

Description automatically generated

**Conclusion:**

This guide covers the basic setup for creating a virtual mouse using AI. You can further improve the system by refining the gesture recognition and adding more features as needed.