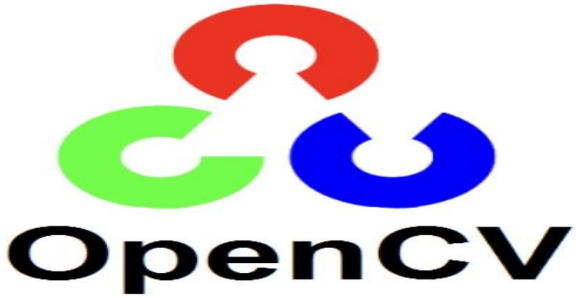
OBJECT IDENTIFICATION COMPUTER VISION

INTRODUCTION

This project demonstrates the integration of advanced computer vision techniques to solve practical problems in real-time object detection, tracking, and counting. It provides a foundation for various applications requiring real-time monitoring and analysis of objects in video streams.

LEARNING IN THIS PROJECT



YOLOv8

It can detect multiple classes of objects in a single pass through the network, making it suitable for real-time applications .

PYCHARM

Integrated Development Environment (IDE)

Steps :

- Download python in your device
- Download pycharm in your device
- Create a project (venv)
- Install the libraries listed in the next page

List of Libraries

- Cvzone
- Ultralytics
- Hydra-core
- Opencv-python
- Matplotlib
- Pillow
- Pyyaml
- Torch
- Torchvision
- Requests
- Scipy
- Tqdm
- Filterpy
- Scikit-image
- lap

Steps :

- Maintain two python files main.py and objectidentification.py
- sort.py you will be needed for SORT (Simple Online and Realtime Tracking) and robust tracking algorithm that links detected objects between frames.
- It assigns unique IDs to each detected object and tracks their movements over time, allowing for the counting of distinct objects.
- sort.py will be you third file in directory.

Object_identification.py

```python
from ultralytics import YOLO
import cv2
import cvzone
import math
import time

cap = cv2.VideoCapture(0)  # For Webcam
cap.set(3, 1280)
cap.set(4, 720)

if not cap.isOpened():
    print("Error: Could not open webcam.")
    exit()

model = YOLO("../Yolo-Weights/yolov8l.pt")

classNames = ["person", "bicycle", "car", "motorbike", "aeroplane", "bus", "train", "truck", "boat",
              "traffic light", "fire hydrant", "stop sign", "parking meter", "bench", "bird", "cat",
              "dog", "horse", "sheep", "cow", "elephant", "bear", "zebra", "giraffe", "backpack", "umbrella",
              "handbag", "tie", "suitcase", "frisbee", "skis", "snowboard", "sports ball", "kite", "baseball bat",
              "baseball glove", "skateboard", "surfboard", "tennis racket", "bottle", "wine glass", "cup",
              "fork", "knife", "spoon", "bowl", "banana", "apple", "sandwich", "orange", "broccoli",
              "carrot", "hot dog", "pizza", "donut", "cake", "chair", "sofa", "pottedplant", "bed",
              "diningtable", "toilet", "tvmonitor", "laptop", "mouse", "remote", "keyboard", "cell phone",
              "microwave", "oven", "toaster", "sink", "refrigerator", "book", "clock", "vase", "scissors",
              "teddy bear", "hair drier", "toothbrush"
```

```
prev_frame_time = 0
new_frame_time = 0

while True:
    new_frame_time = time.time()
    success, img = cap.read()
    if not success:
        print("Error: Failed to capture image from webcam.")
        continue

    print(f"Image shape: {img.shape}")

    try:
        results = model(img, stream=True)
        for r in results:
            boxes = r.boxes
            for box in boxes:
                # Bounding Box
                x1, y1, x2, y2 = box.xyxy[0]
                x1, y1, x2, y2 = int(x1), int(y1), int(x2), int(y2)
                w, h = x2 - x1, y2 - y1
                cvzone.cornerRect(img, (x1, y1, w, h))
                # Confidence
                conf = math.ceil((box.conf[0] * 100)) / 100
                # Class Name
                cls = int(box.cls[0])
```

```
            cvzone.putTextRect(img, f'{classNames[cls]} {conf}', (max(0, x1), max(35, y1)), scale=1, thickness=1)
    except Exception as e:
        print(f"Error during model prediction: {e}")
        continue

    fps = 1 / (new_frame_time - prev_frame_time)
    prev_frame_time = new_frame_time
    print(fps)

    cv2.imshow("image", img)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

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

main.py

```python
#install library
import numpy as np
from ultralytics import YOLO
import cv2
import cvzone
import math
from sort import *

# Initialize video capture from webcam (change parameter to use different camera)
cap = cv2.VideoCapture(0)
# Load YOLO model with pre-trained weights
model = YOLO("../Yolo-Weights/yolov8l.pt")

# Define class names for object detection
classNames = ["person", "bicycle", "car", "motorbike", "aeroplane", "bus", "train", "truck", "boat",
              "traffic light", "fire hydrant", "stop sign", "parking meter", "bench", "bird", "cat",
              "dog", "horse", "sheep", "cow", "elephant", "bear", "zebra", "giraffe", "backpack", "umbrella",
              "handbag", "tie", "suitcase", "frisbee", "skis", "snowboard", "sports ball", "kite", "baseball bat",
              "baseball glove", "skateboard", "surfboard", "tennis racket", "bottle", "wine glass", "cup",
              "fork", "knife", "spoon", "bowl", "banana", "apple", "sandwich", "orange", "broccoli",
              "carrot", "hot dog", "pizza", "donut", "cake", "chair", "sofa", "pottedplant", "bed",
              "diningtable", "toilet", "tvmonitor", "laptop", "mouse", "remote", "keyboard", "cell phone",
              "microwave", "oven", "toaster", "sink", "refrigerator", "book", "clock", "vase", "scissors",
              "teddy bear", "hair drier", "toothbrush"
              ]
```

```python
# Initialize SORT tracker with parameters
tracker = Sort(max_age=20, min_hits=3, iou_threshold=0.3)

# Define the coordinates for a line to count objects crossing
limits = [200, 500, 1000, 500]  # [x1, y1, x2, y2] to draw a line from (200, 500) to (1000, 500)

# Initialize list to store IDs of counted objects
totalCount = []

# Main loop to process video frames
while True:
    # Capture a frame from the video
    success, img = cap.read()

    # Break the loop if video capture failed
    if not success:
        break

    # Perform object detection using YOLO on the current frame
    results = model(img, stream=True)

    # Initialize an empty array to store detections (x1, y1, x2, y2, confidence)
    detections = np.empty((0, 5))
```

```python
# Process each detected object in the results
    for r in results:
        boxes = r.boxes
        for box in boxes:
            # Extract bounding box coordinates
            x1, y1, x2, y2 = box.xyxy[0]
            x1, y1, x2, y2 = int(x1), int(y1), int(x2), int(y2)
            w, h = x2 - x1, y2 - y1

            # Extract confidence and class name
            conf = math.ceil((box.conf[0] * 100)) / 100
            cls = int(box.cls[0])
            currentClass = classNames[cls]

            # Filter objects of interest (e.g., cell phone, remote, laptop, apple) based on confidence
            if (
                    currentClass == "cell phone" or currentClass == "remote" or currentClass == "laptop" or currentClass == "apple"):
                currentArray = np.array([x1, y1, x2, y2, conf])
                detections = np.vstack((detections, currentArray))

    # Update SORT tracker with current detections
    resultsTracker = tracker.update(detections)
```

```python
# Draw a line on the image for counting objects
cv2.line(img, (limits[0], limits[1]), (limits[2], limits[3]), (0, 0, 255), 5)

# Process each tracked object result
for result in resultsTracker:
    x1, y1, x2, y2, id = result
    x1, y1, x2, y2 = int(x1), int(y1), int(x2), int(y2)

    # Draw a rectangle around the tracked object
    w, h = x2 - x1, y2 - y1
    cvzone.cornerRect(img, (x1, y1, w, h), l=9, rt=2, colorR=(255, 0, 255))

    # Display the object ID near the object
    cvzone.putTextRect(img, f' {int(id)}', (max(0, x1), max(35, y1)),
                       scale=2, thickness=3, offset=10)

    # Draw a circle at the center of the object
    cx, cy = x1 + w // 2, y1 + h // 2
    cv2.circle(img, (cx, cy), 5, (255, 0, 255), cv2.FILLED)
```

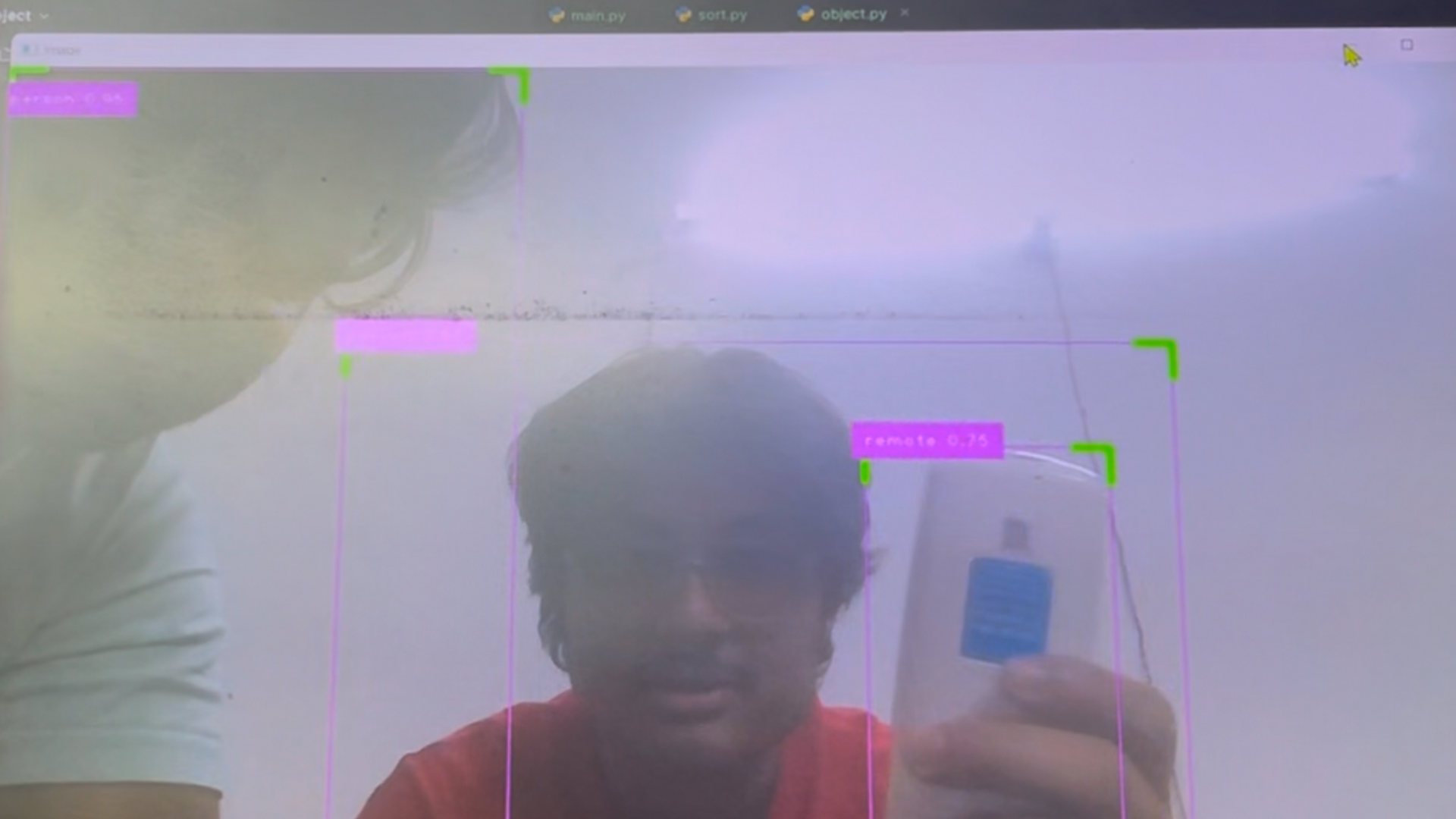
```
        # Check if the object crosses the counting line
        if limits[0] < cx < limits[2] and limits[1] - 15 < cy < limits[1] + 15:
            if totalCount.count(id) == 0:
                totalCount.append(id)
                # Change line color to green when an object is counted
                cv2.line(img, (limits[0], limits[1]), (limits[2], limits[3]), (0, 255, 0), 5)

    # Display the total count of objects on the image
    cv2.putText(img, str(len(totalCount)), (255, 100), cv2.FONT_HERSHEY_PLAIN, 5, (50, 50, 255), 8)

    # Show the processed image with overlays
    cv2.imshow("Image", img)

    # Wait for 'q' key press to exit the loop
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# Release the video capture and close all OpenCV windows
cap.release()
cv2.destroyAllWindows()
```


main.py
sort.py
object.py
remote 0.75