YOLO Object Detection with OpenCV and Mediapipe

Roll No : 2022MC58 Haseeb ul Hassan

January 9, 2025

Abstract

This report outlines the implementation of a YOLO-based object detection system using OpenCV and Mediapipe. The system processes video frames captured from a webcam, detects objects using the YOLO model, and displays the detection results in real time. The implementation details and code are provided below.

Introduction

Object detection is a crucial component of computer vision applications. YOLO (You Only Look Once) is a state-of-the-art deep learning model for object detection, known for its speed and accuracy. This report demonstrates the integration of YOLO with OpenCV and Mediapipe to create a real-time object detection system.

Implementation

The Python code for implementing YOLO-based object detection is provided in Listing 1. This system captures video frames, processes them using the YOLO model, and displays the results with detected objects highlighted.

Listing 1: YOLO Object Detection Code

```
import cv2 as cv import mediapipe as mp from ultralytics import YOLO Load YOLO model model =

YOLO('yolo11n.pt') Initialize video capture video = cv.VideoCapture(0)

video.set(3, 1000) # Set width video.set(4, 1000) # Set

height Process video frames

while(True):
ret, frame = video.read() frame = cv.flip(frame, 1) # Flip the frame horizontally results = model(frame,
show=True, conf=0.6, save=True) # Run YOLO
        model
if cv.waitKey(1) & 0xFF == ord('q'): # Exit on 'q' key press break

Release video and close windows

video.release() cv.destroyAllWindows()
```

Flowchart

The flowchart for the YOLO-based object detection process is shown below:

Results

The system successfully detects objects in real time and displays them on the video stream. The output includes bounding boxes and labels for the detected objects. Screenshots of the detection results are shown below:

Conclusion

This report demonstrated the implementation of a YOLO-based object detection system using OpenCV and Mediapipe. The system is capable of detecting objects in real time with high accuracy. Future work could involve optimizing the system for specific use cases or integrating additional features such as tracking or analytics.

References

1. Redmon, J., Farhadi, A. (2018). YOLOv3: An Incremental Improvement. arXiv preprint arXiv:1804.02767.

2. OpenCV Documentation: https://docs.opencv.org/

3. Mediapipe Documentation: https://mediapipe.dev/

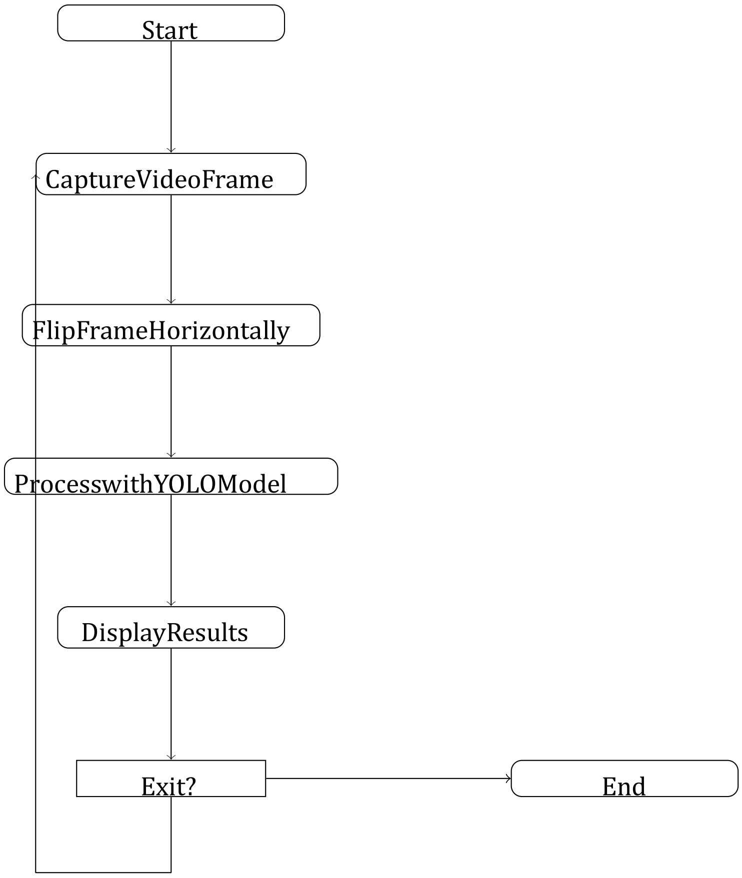

Figure 1: Flowchart of YOLO Object Detection Process example_output.png

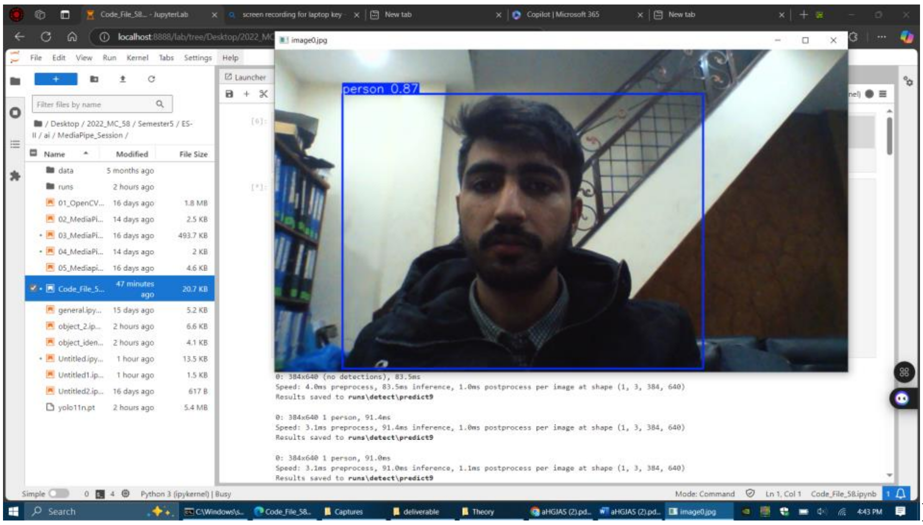

Figure 2: Example of YOLO object detection output.

Instructions to install the dependencies ;

- Install python 3.9from this site: https://www.python.org/downloads/
- For Jupyter Lab Visit: https://jupyter.org/install
- Open cmd and type Jupyter lab then type : pip install opencv-python mediapipe ultralytics
- Then Make a new file and paste the code and start
- Meanwhile am using Opencv On GPU as my system support but installation is a bit different like Opencv is build using Cmake

How to run the code and record results

- Open cmd and type Jupyter lab then A browser window will open On your local host usually on 8080 port but may differ.
- Then Make a new file and paste the code and start
- Execution Status wull be shown and no of persons etc will be shown and a Seprate folder will be made if you enabled save option in yolo run line where we select the camera option and the sensitivity