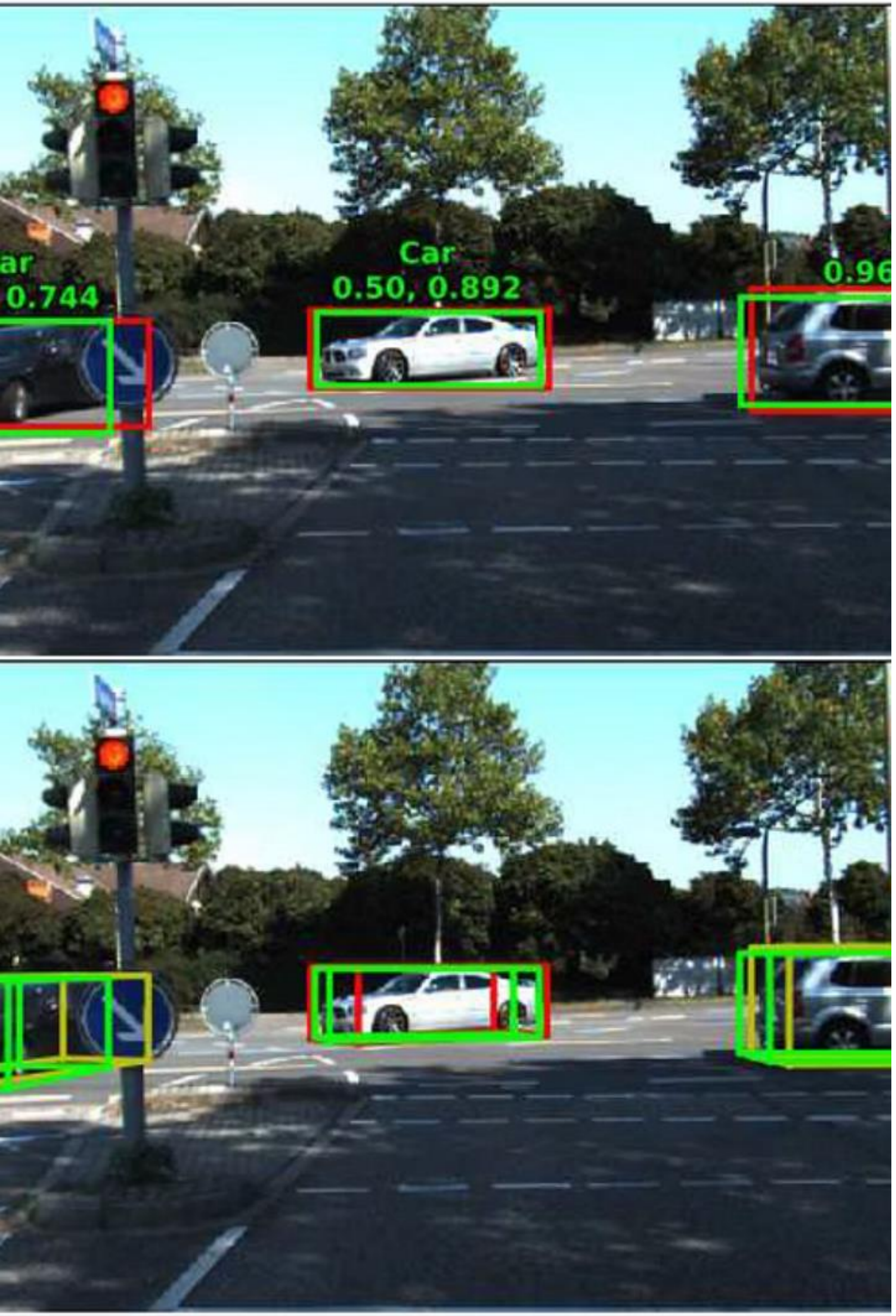
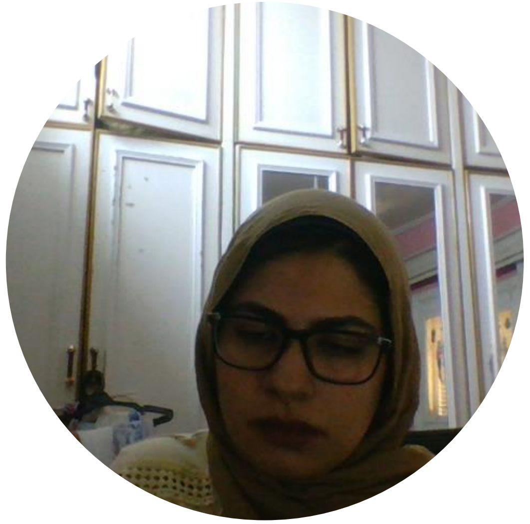


Automated Vehicle-Level Feature Annotation Tool

Name: Fareeda Saad

Major: Computer Science and Engineering

Supervised By: Dr Catherine Malak Elias

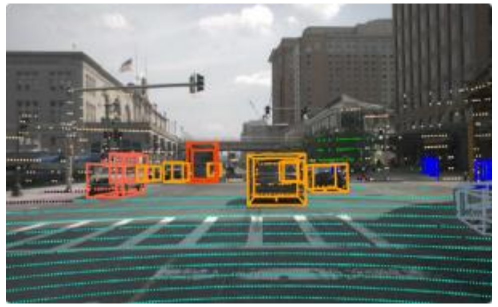


Project Outline

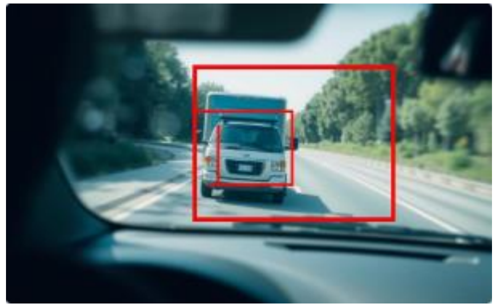
1. Introduction
2. My Problem Formulation
3. Literature Review
4. Research Gap
5. Thesis Objective
6. Methodology Design & Implementation
    1. Explore **2D automatic annotation** using YOLOv3, YOLOv11 (combined with Depth Anything v2) and **3D manual annotation** using OpenCV, CVAT, and Mindkosh.
    2. Demos
7. Results and Discussion : Present samples from YOLOv3, CVAT/Mindkosh, and annotated video frames and outputs of the models .
8. Conclusion
9. Future Recommendations
10. Questions

Introduction

In the field of Autonomous driving and intelligent transportation systems ,high-quality annotated datasets are crucial for training and evaluating perception models



These datasets typically include detailed information about vehicles—such as position, speed, orientation, and dimensions—extracted from sensor data or visual input.



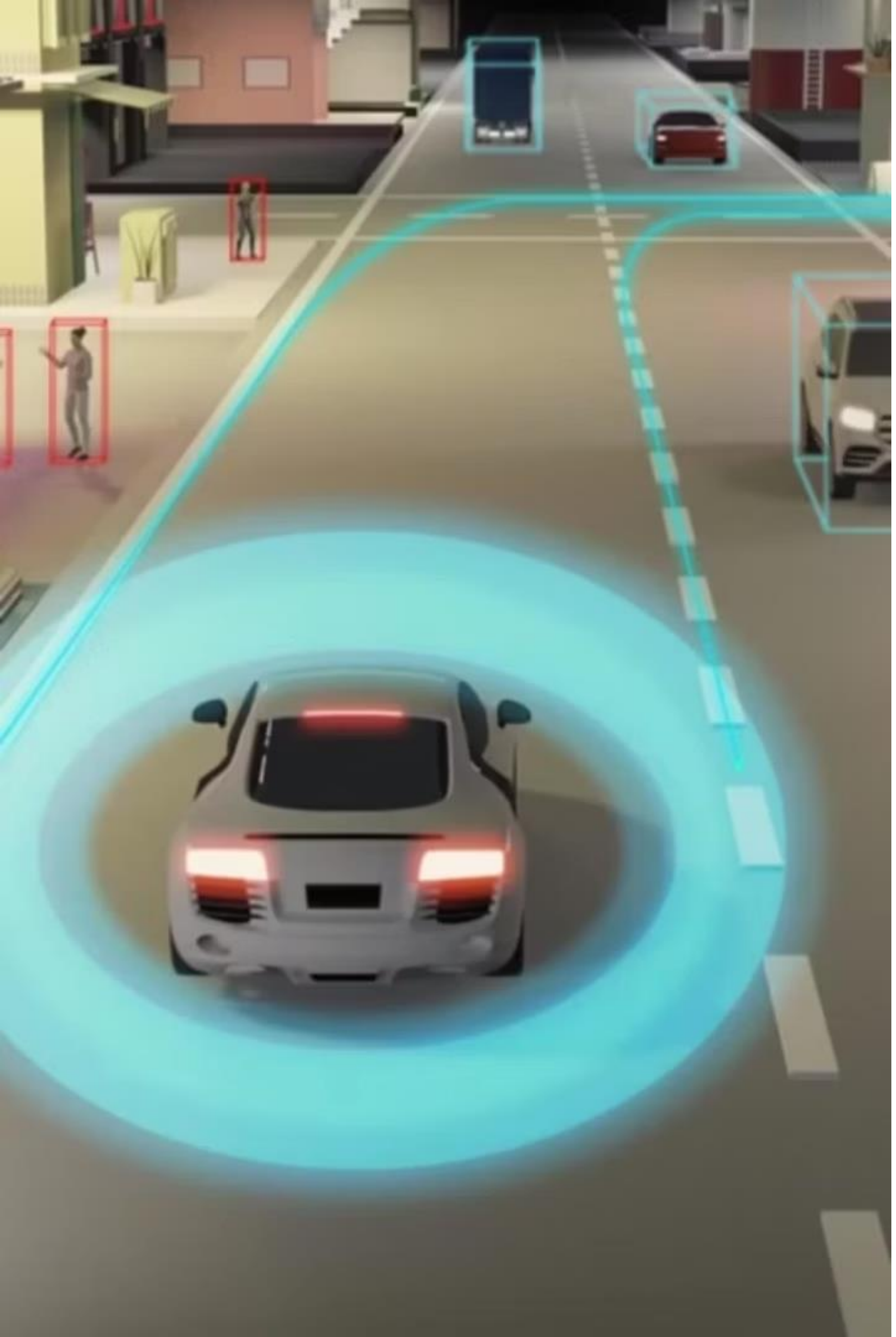
While 2D annotations are widely used, they offer limited spatial understanding.

This project focuses on developing a tool for automated vehicle-level feature annotation

Using monocular camera data, with the goal of enhancing scene understanding in a low-cost, sensor-free environment.

problem formulation

- Current vehicle-level annotation tools often rely on 2D bounding boxes or sensor fusion (e.g., LiDAR, radar) to capture features like speed, orientation, and dimensions.
- However, these approaches either lack spatial depth or require expensive hardware.
- Moreover, manual annotation—especially for 3D bounding boxes—is time-consuming and error-prone, leading to inconsistencies in training data.
- This project addresses the dual challenge of (1) achieving accurate, sensor-free vehicle feature extraction using only monocular camera input and 3D bounding box annotations, and (2) improving the efficiency and consistency of the annotation process for rich feature labeling (e.g., velocity, orientation, inter-vehicle distance, time-to-collision).



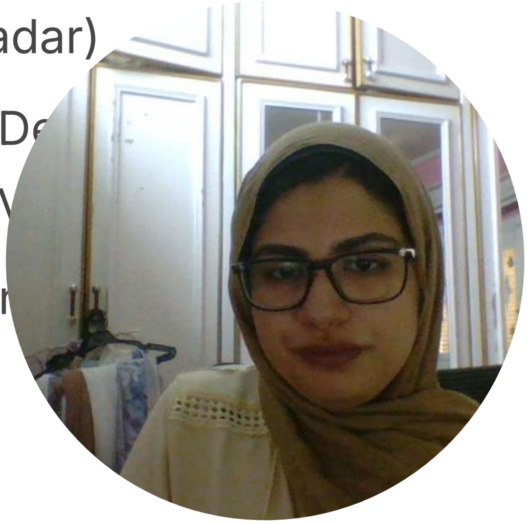
Literature Review Summary

Most Commonly Detected Vehicle Features

- Vehicle Type
- Trajectory & Speed
- 3D Bounding Box (Position, Orientation, Dimensions)
- License Plate / ID
- Object Motion Status (Moving / Stationary)

Popular Detection Methods

- YOLO / CNN-based Object Detection
- LiDAR Point Cloud Segmentation
- Sensor Fusion (Camera + LiDAR + Radar)
- 3D Object De (PointNet, V
- Manual Ann LabelImg



Extended Literature Review: Monocular 3D Object Detection

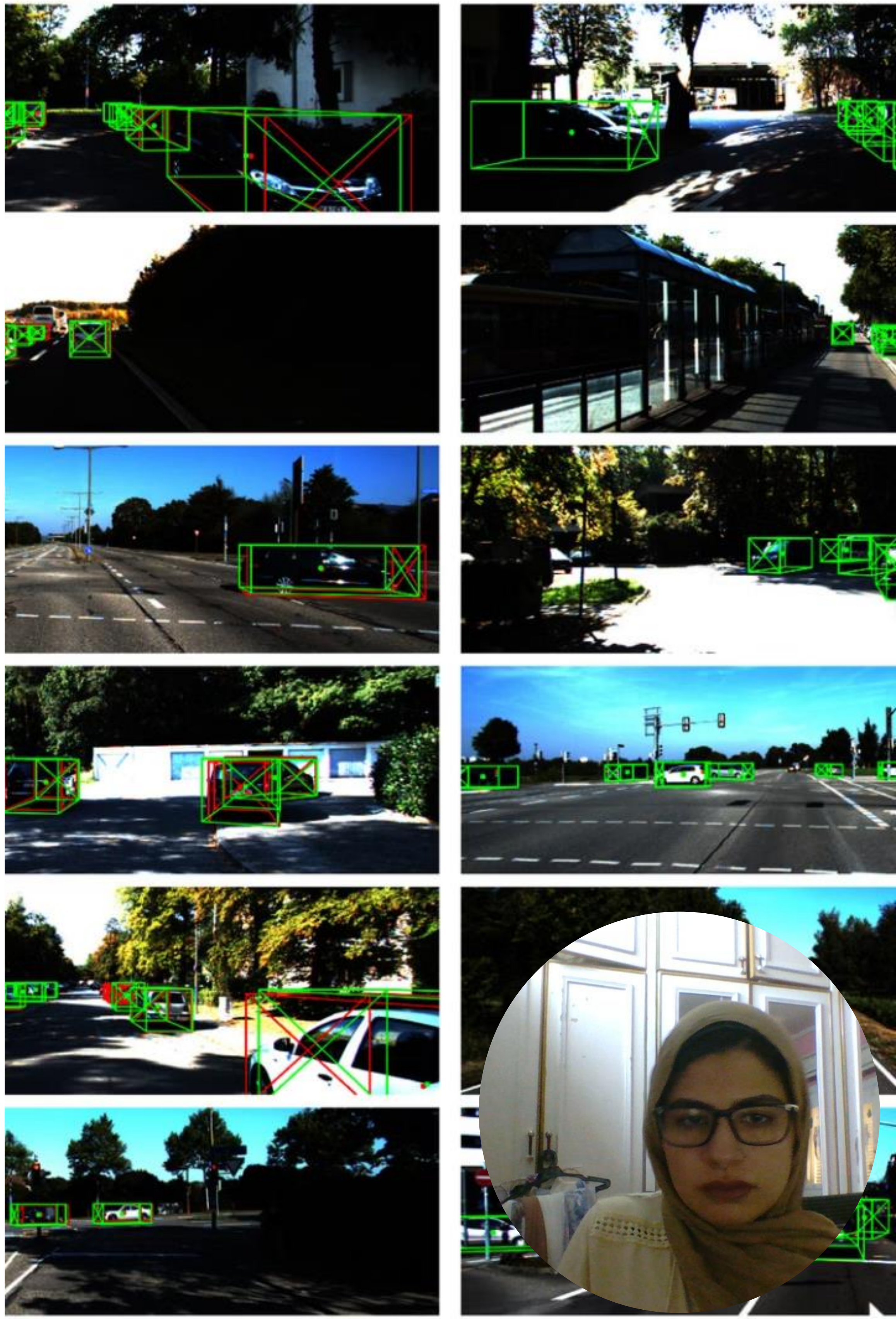
In the last two weeks, I expanded my literature review by exploring recent monocular 3D object detection models using Papers with Code and Google Scholar. I studied papers indexed from 21 to 27 in my spreadsheet:, which provided a variety of models suitable for automatic annotation tasks.

Key insights from these papers:

- Rapid advancements in monocular 3D detection without the need for expensive LiDAR sensors.
- Diverse model architectures focusing on depth estimation, 3D localization, and bounding box refinement.
- Available models offer different trade-offs between accuracy, speed, and computational efficiency.
- Several models showed strong potential for vehicle feature extraction directly from monocular images, enabling efficient dataset annotation.

This exploration helped me select potential candidates for automatic annotation, including YOLOv8+Depth, YOLOv11+Depth Anything v2, and MonoLSS.

Spreadsheetlink:https://1drv.ms/x/c/5962268546b6d7e5/ETV2CZ7LUBNNt_BeV_03efQB4MlPjgUBlb6lZJhG02Iyxg?e=gk3DVA&nav=MTVfezNEREFEODZDLTkwRUUtNDU3RC05MUYyLTZCOEY5QzQwMkMyMH0



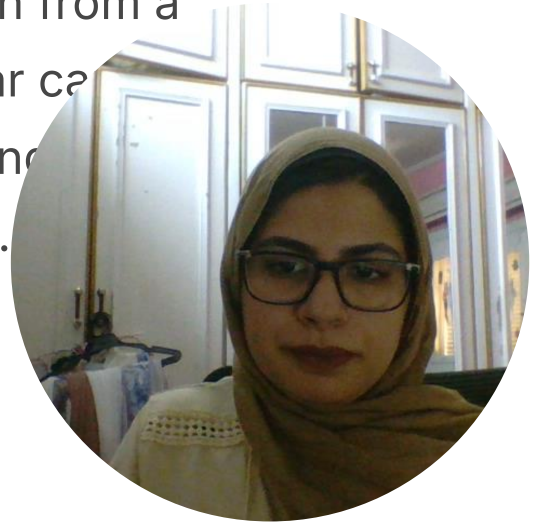
Research Gap

Most existing datasets and annotation tools either rely heavily on LiDAR/radar or offer only 2D bounding box annotations.

Monocular 3D detection models are improving, but their integration into annotation tools for vehicle-level feature extraction is still limited.

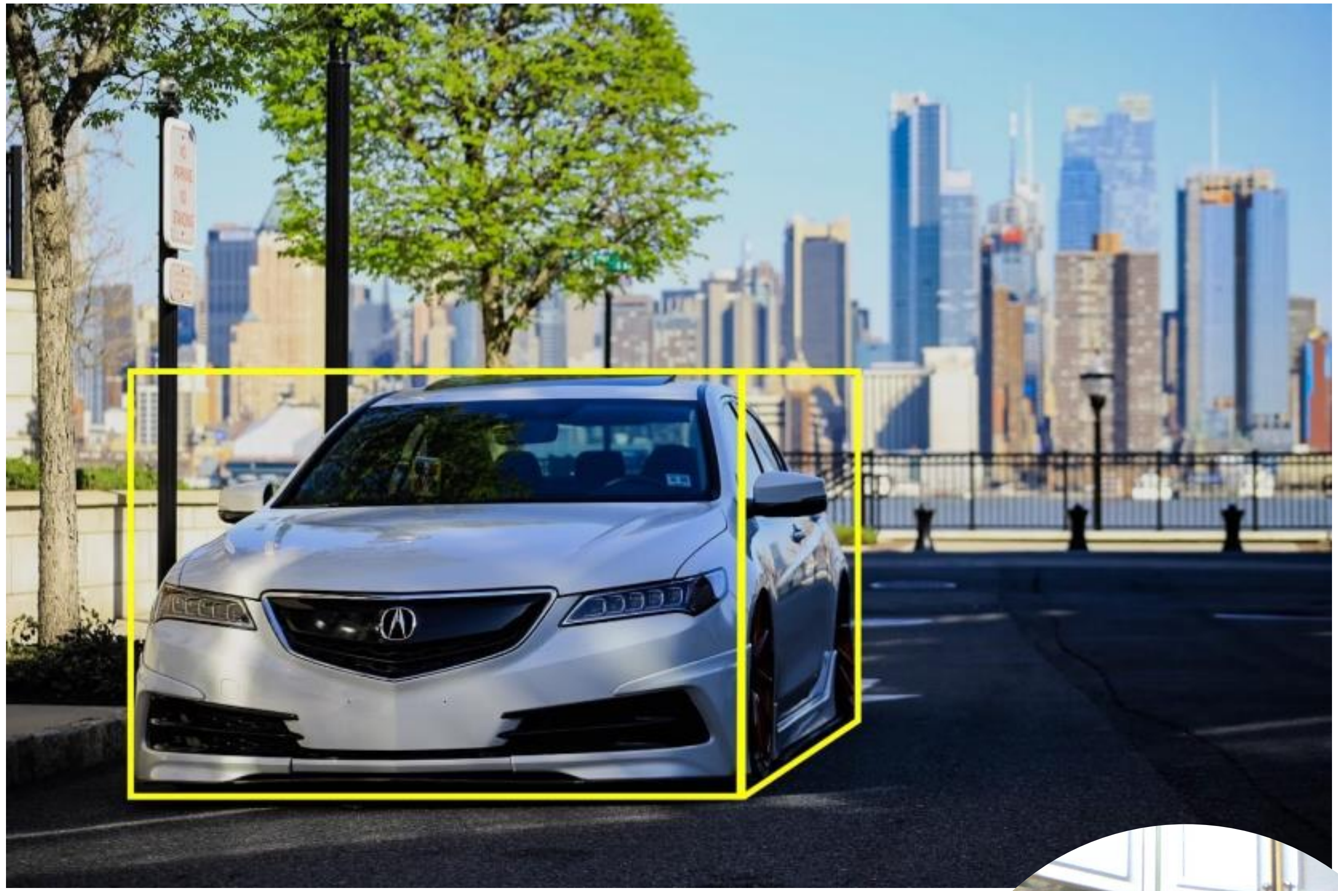
Limited work is available on extracting detailed dynamic features (velocity, acceleration, inter-vehicle distance) from monocular 3D outputs without sensor fusion.

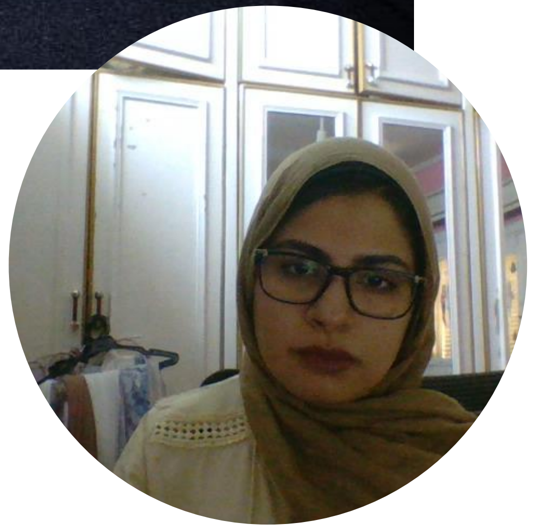
Lack of accessible and standardized pipelines for fully automatic 3D annotation from a monocular ca training an purposes.



My topic Objectives

- Develop a 3D annotation tool for vehicles using only monocular camera data.
- Extract detailed vehicle-level features such as velocity, acceleration, dimensions, and orientation.
- Compare different monocular 3D detection models for annotation efficiency and accuracy.
- Reduce reliance on expensive sensors while maintaining rich scene understanding.
- Propose a pipeline for efficient vehicle feature annotation and validate it on real-world datasets.

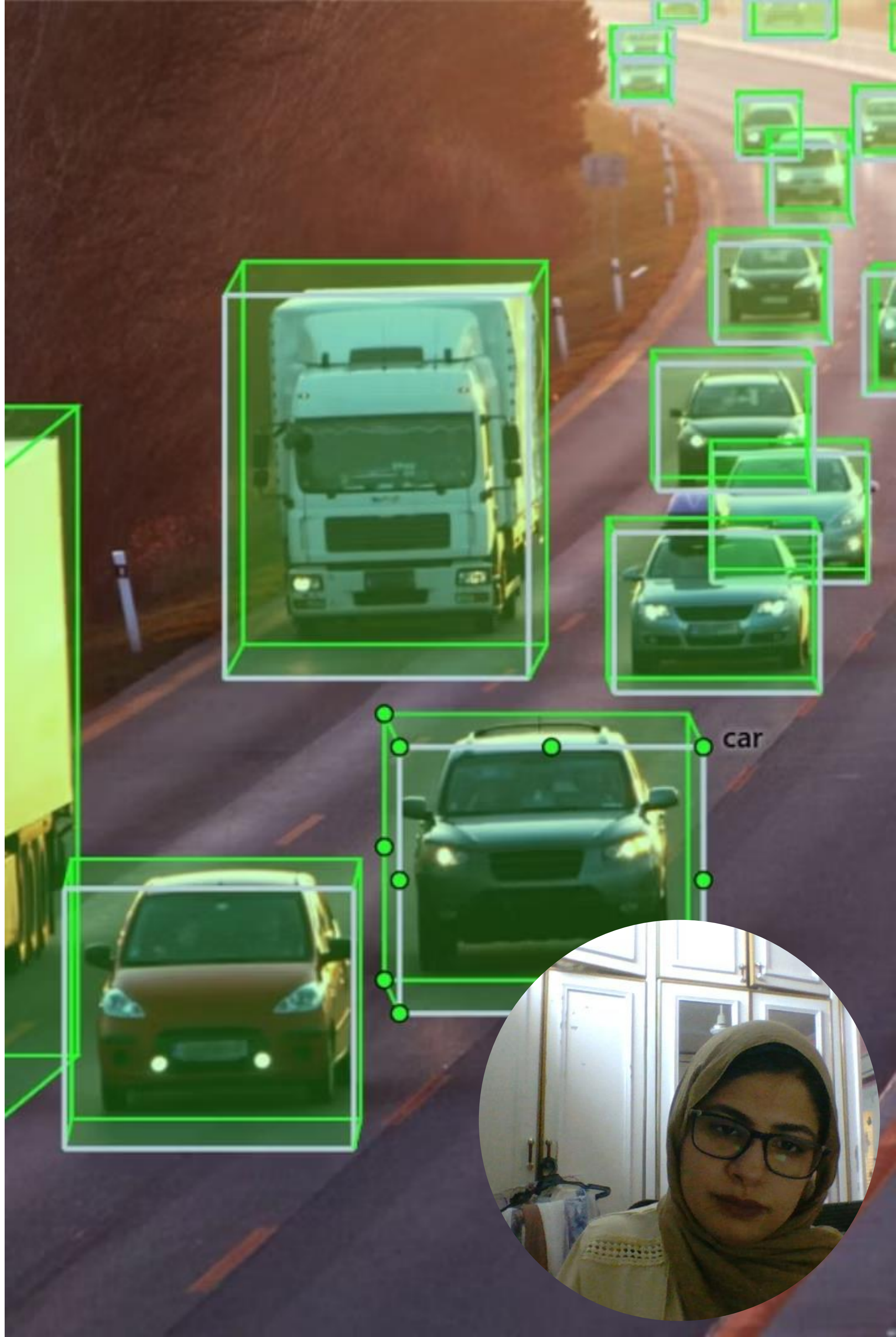




Methodology & Implementation

1. **Early Stage:**
   - Automated 2D vehicle detection using YOLOv3.
   - Manual 3D annotation using OpenCV, CVAT, and Mindkosh platforms.
2. **Mid Stage:**
   - Semi-automated annotation using YOLOv11 with Depth Anything v2.
   - Extracted vehicle-level features: 3D bounding box dimensions, centroids, velocities, accelerations, inter-vehicle distances.
3. **Current Stage:**
   - Training and evaluating MonoLSS on KITTI dataset.
   - Comparing performance with YOLOv11 + Depth Anything v2 outputs.
   - Feature extraction improvements: occlusion handling, time-to-collision estimation.
4. **Challenges:**
   - Code unavailability for some models (e.g., YOLOBU).
   - Selection of best monocular 3D model based on practical considerations.

.



Why You Chose MonoLSS over Others

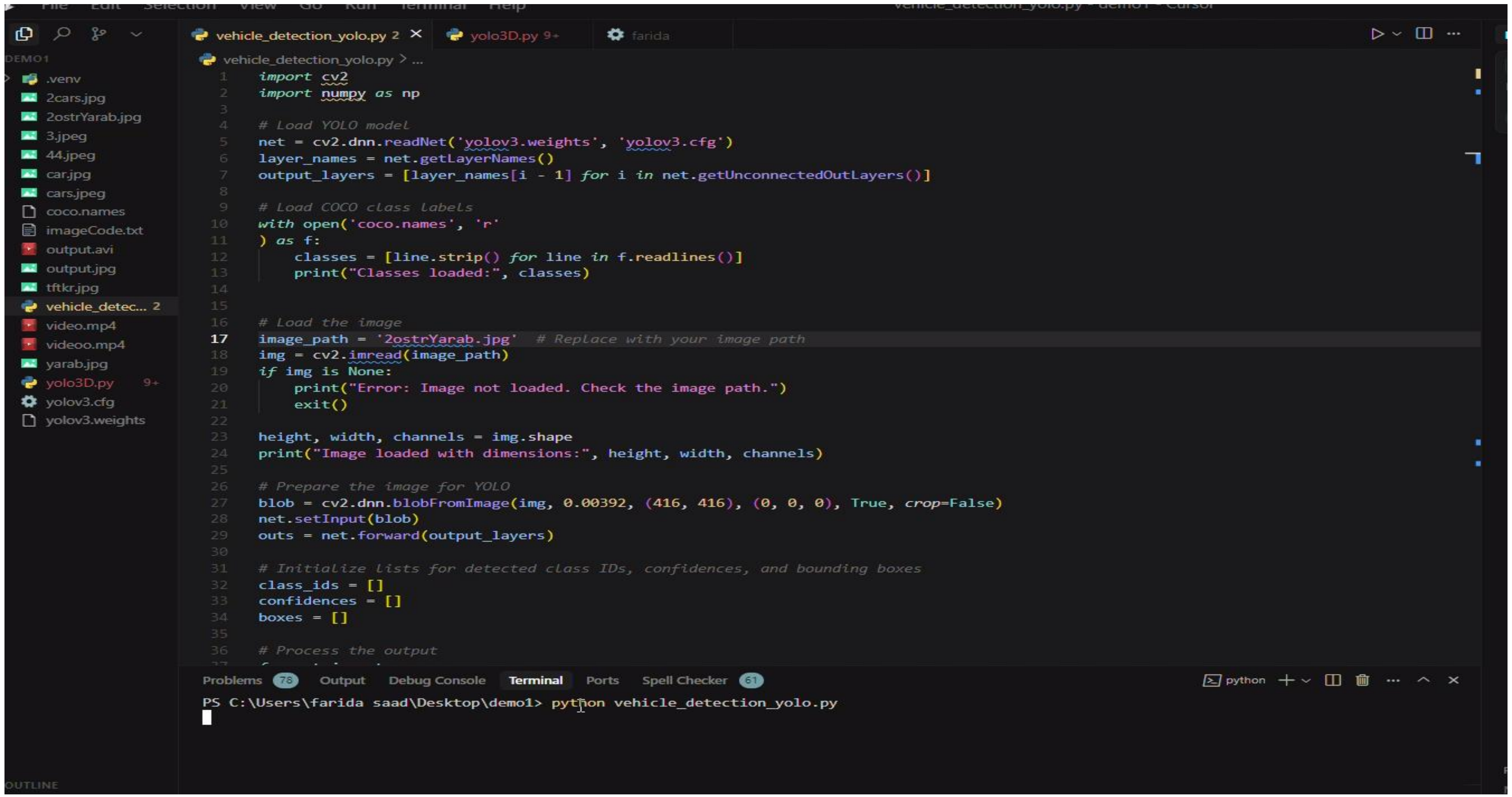
Why MonoLSS?

- **Compared to MoGDE:**
- MonoLSS better models geometric depth explicitly rather than only depth prediction.
- **Compared to MonoCAN:**
- MonoCAN uses uncertainty-aware attention but underperforms in spatial accuracy compared to MonoLSS.
- **Compared to Implicit3DUnderstanding:**
- MonoLSS directly supervises depth and geometry, making it easier to train and generalize without complex implicit learning techniques.
- **Compared to SMOKE:**
- SMOKE predicts 3D boxes from 2D heatmaps but struggles with depth ambiguity. MonoLSS addresses depth supervision better, leading to more reliable 3D positions.
- **Overall:**
- MonoLSS strikes a strong balance between detection performance, training stability, and implementation simplicity—making it ideal for an annotation tool.

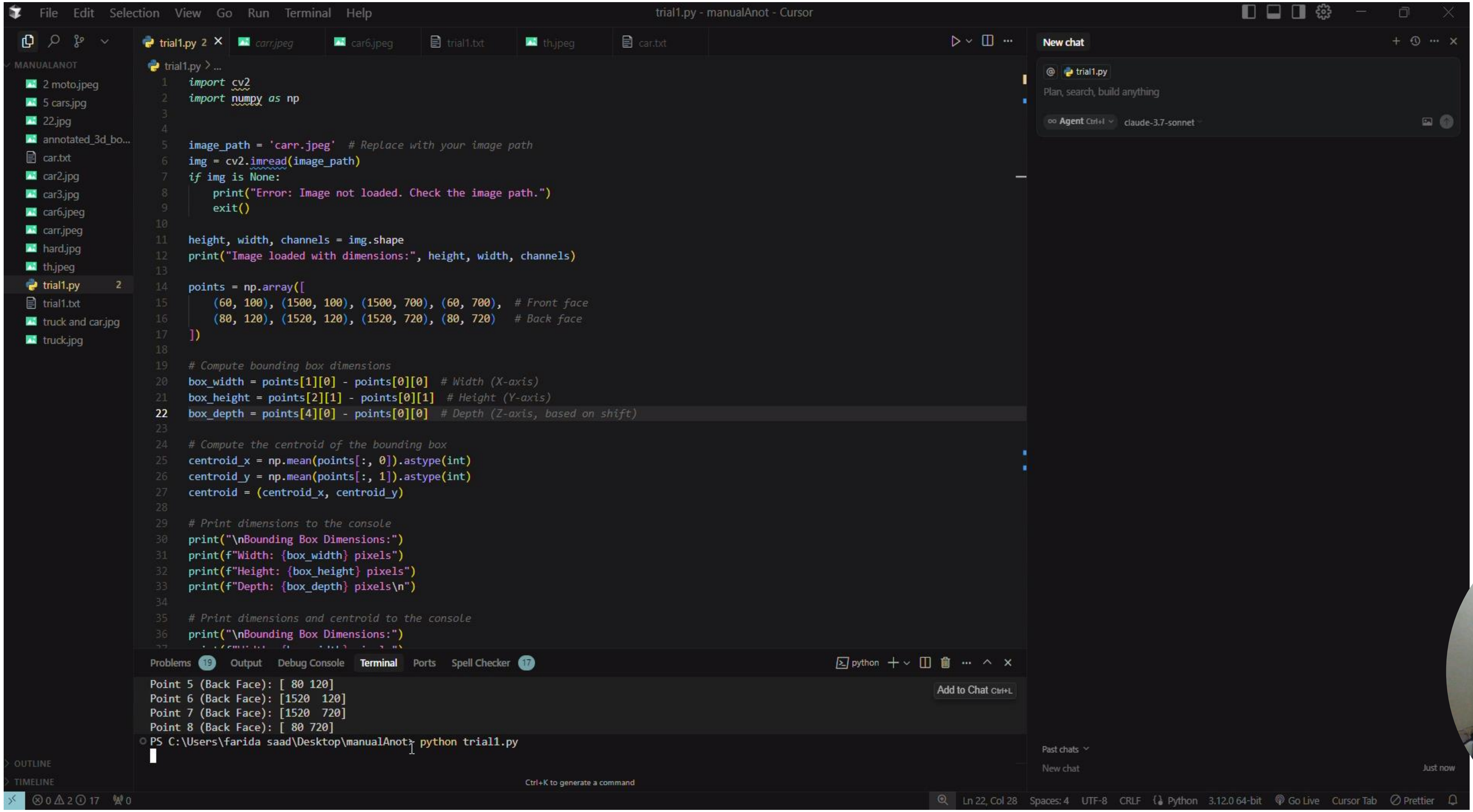
Results

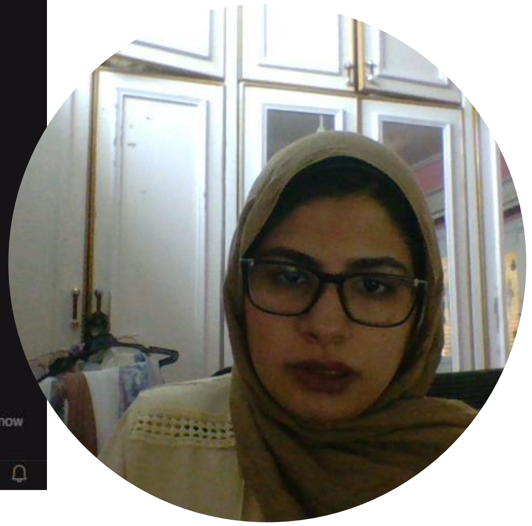
Here are some sample outputs showcasing both our automatic and manual annotation methods.

Yolo Detection (2D bounding boxes)

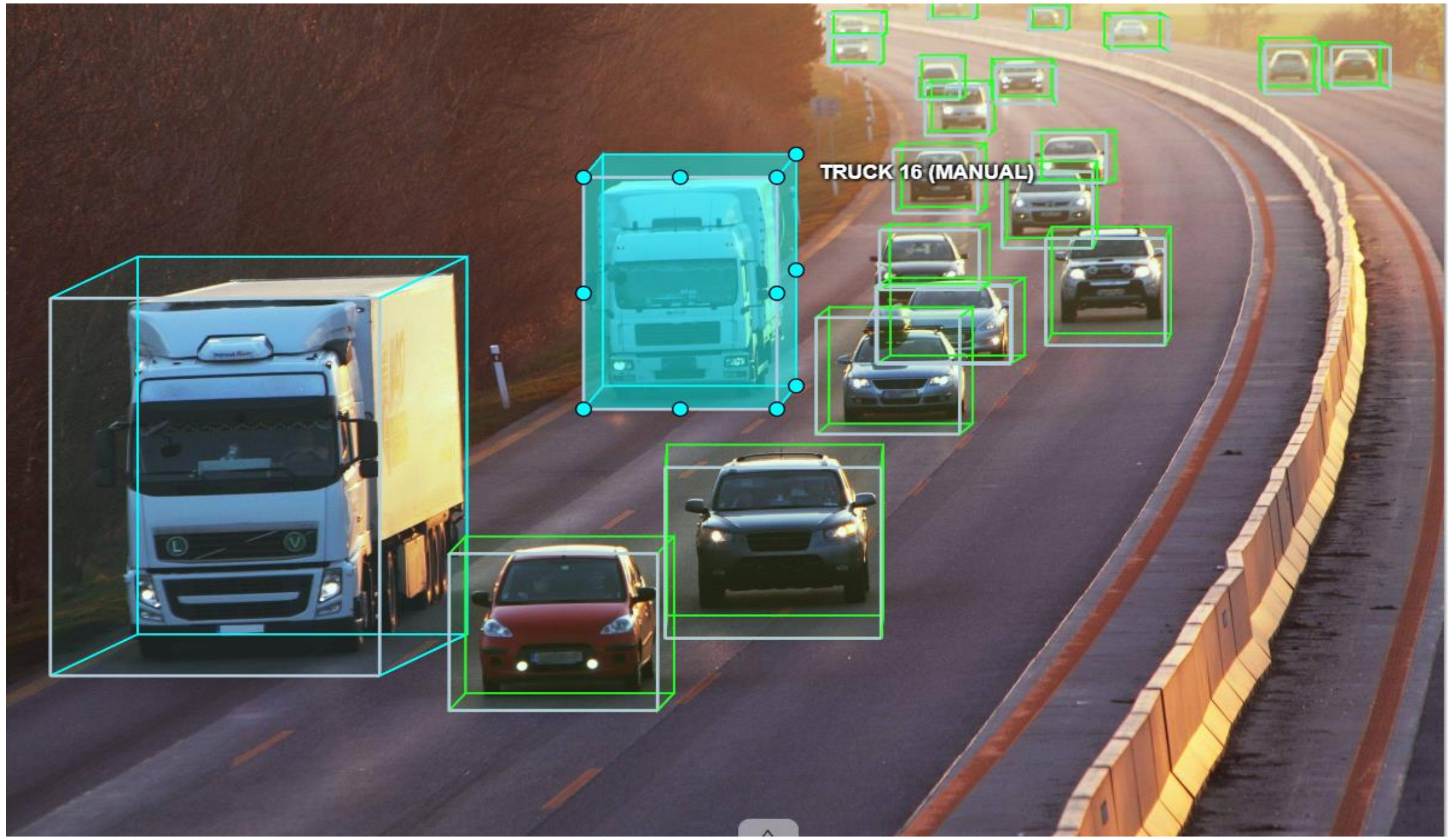


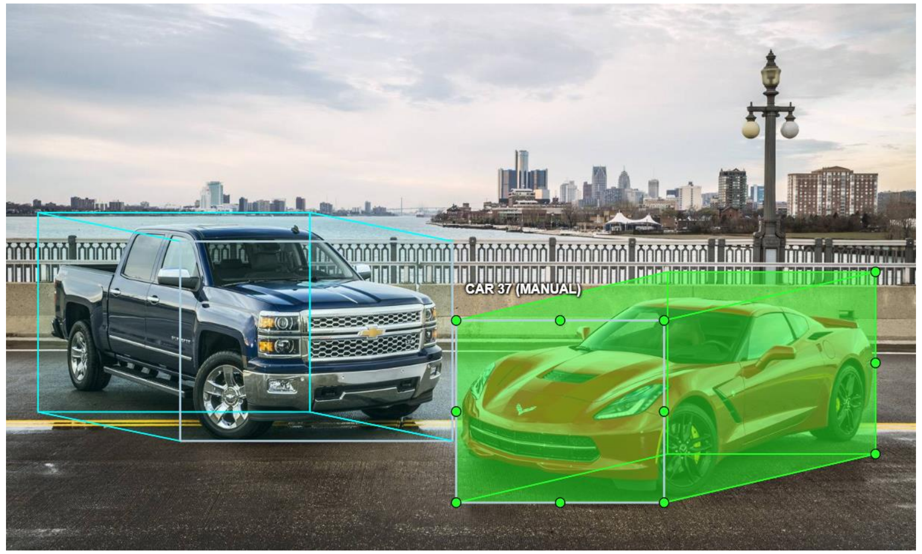
3D bounding boxes manual annotation using OpenCV

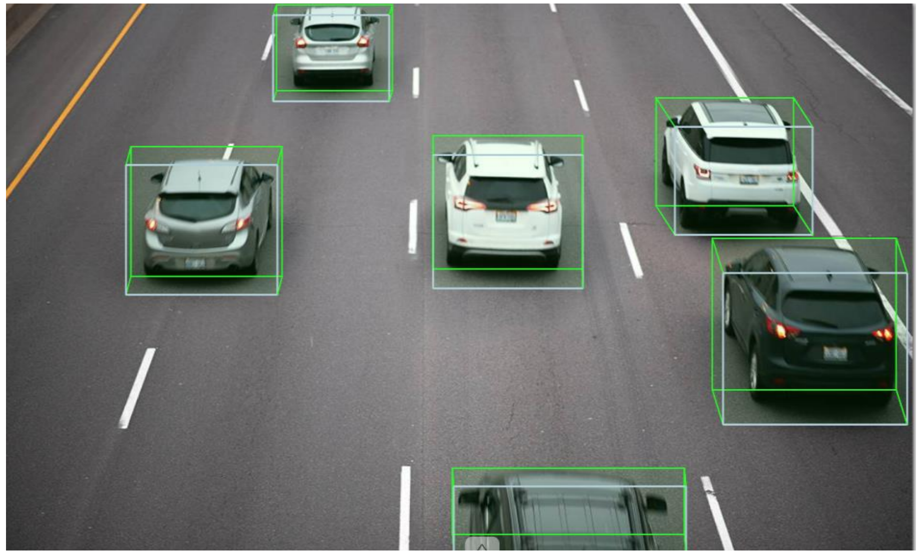


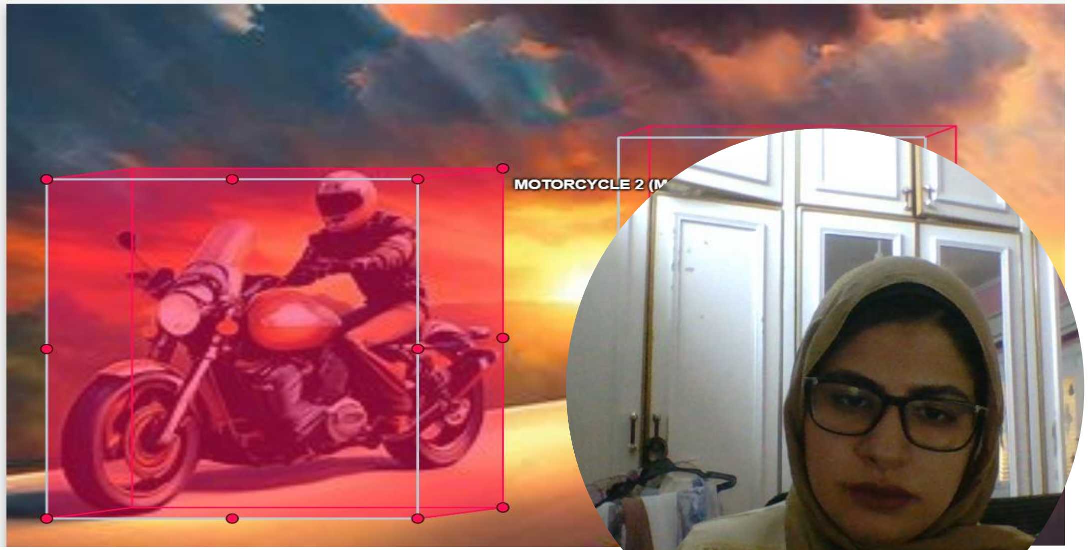


Annotated CVAT/Mindkosh Outputs



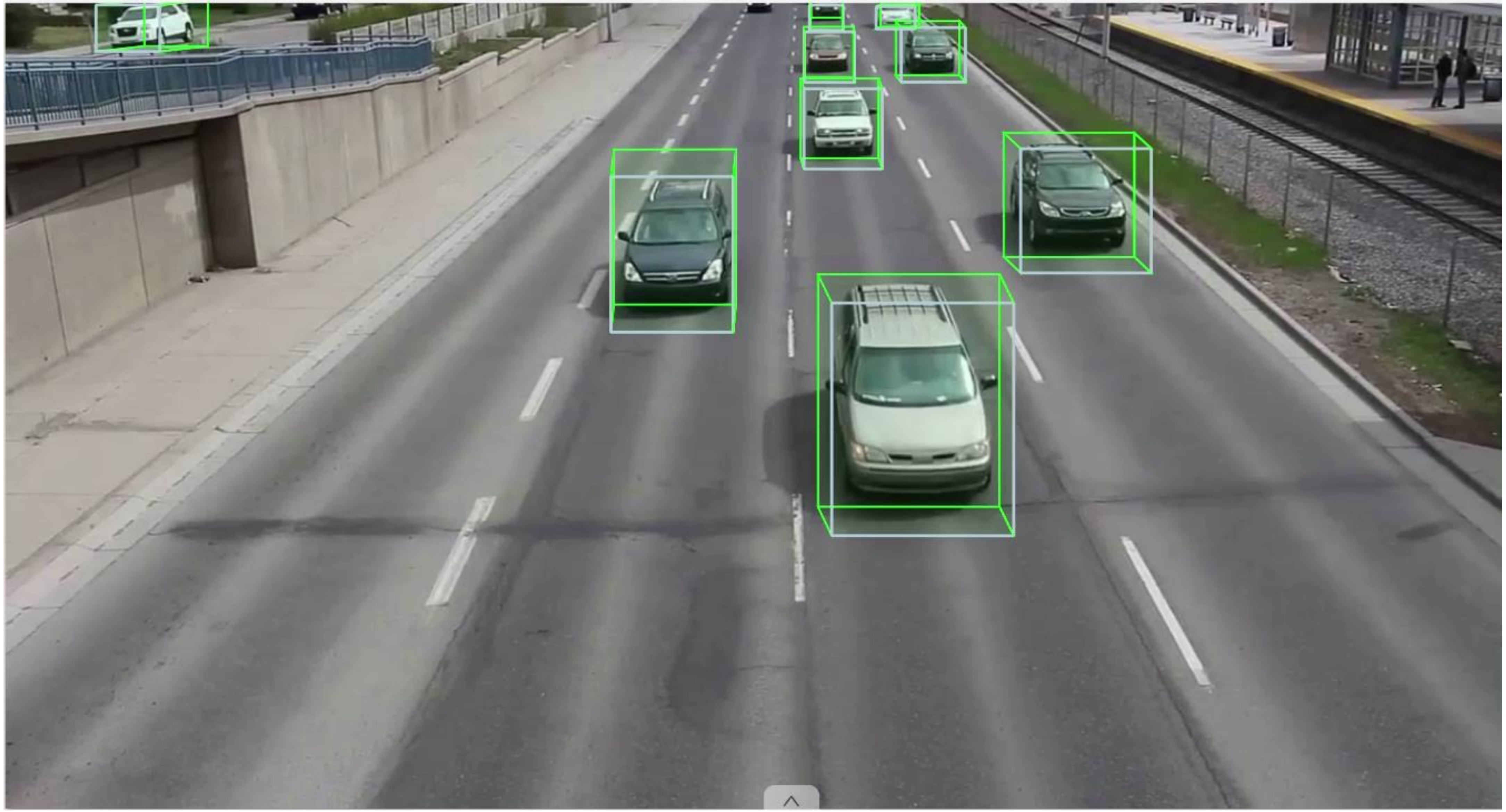






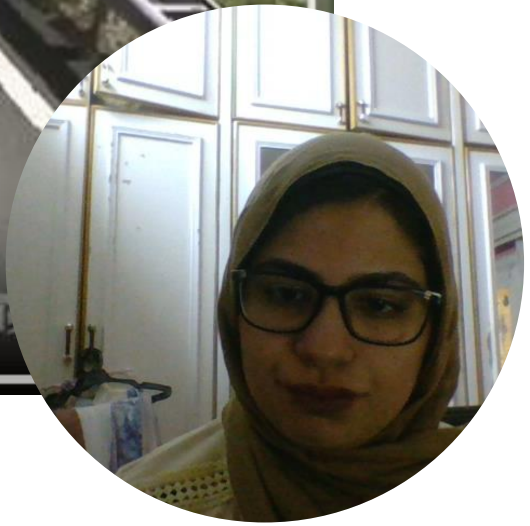
Annotated Video Results

Here is the **60-frame video** I annotated. It shows the progress of labeling over 60 frames. Doing this frame-by-frame using the track features helps create high-quality training data for vehicle detection models.

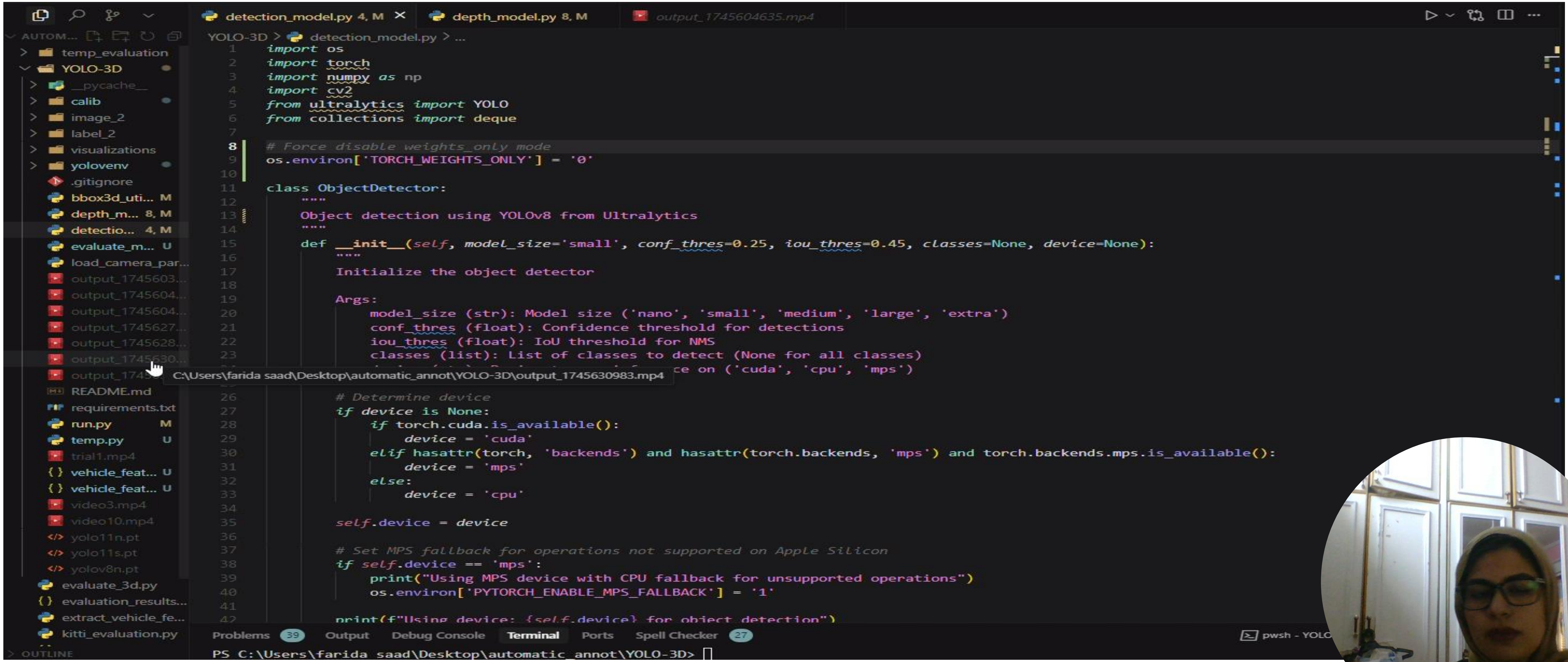


YOLOv11 and Depth Anything v2

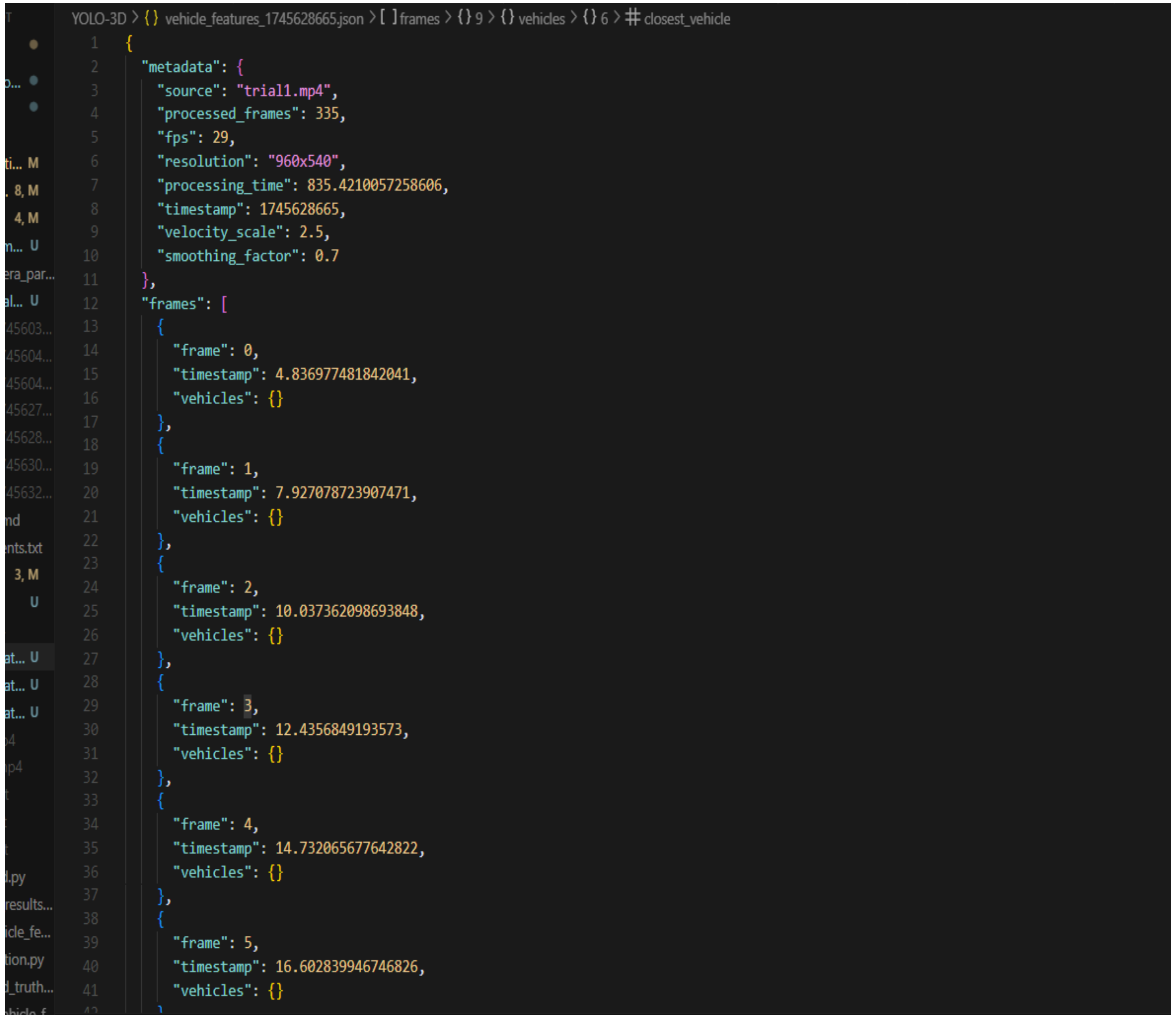
YOLOv11 and Depth Anything v2

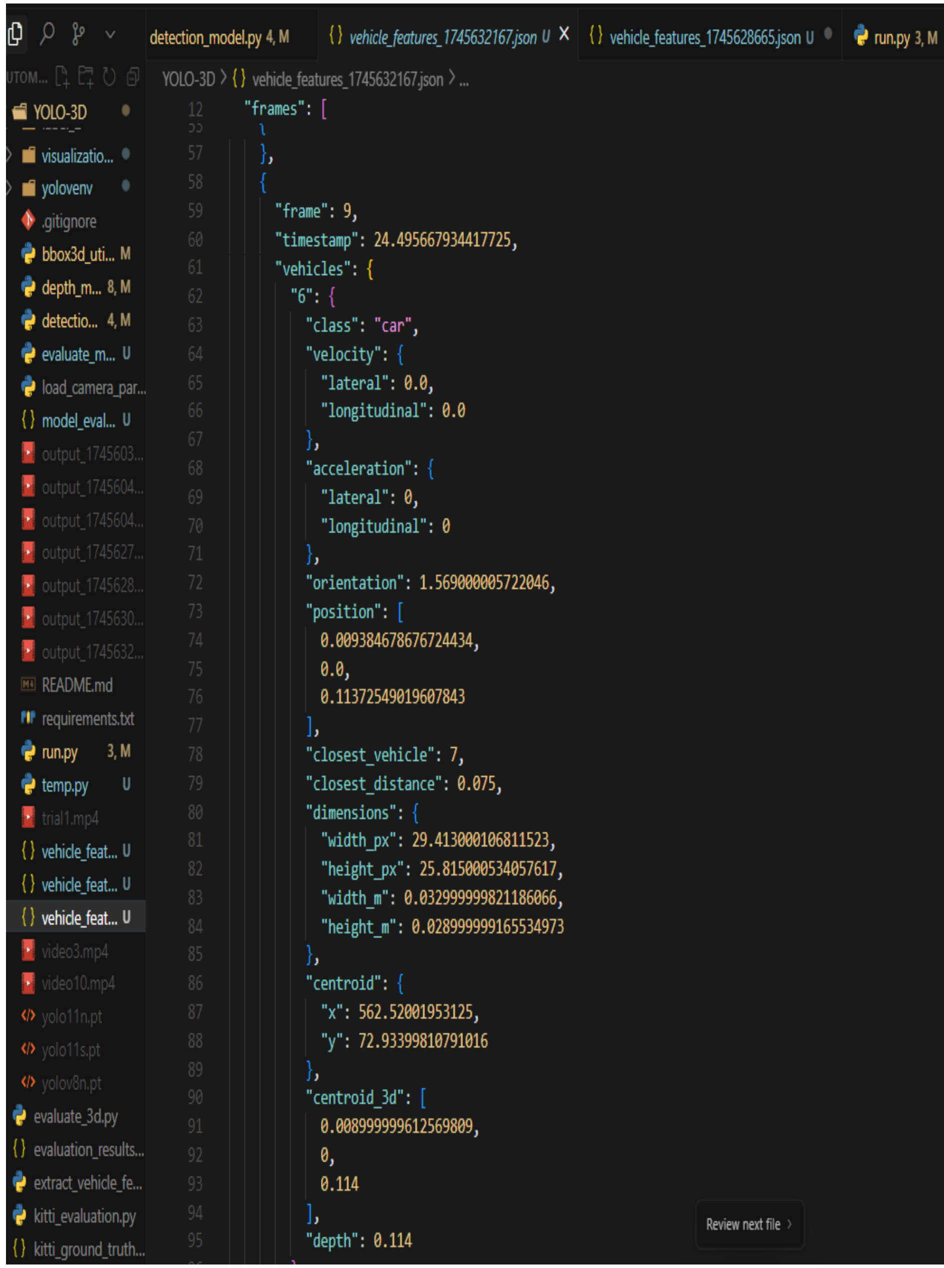


YOLOv11 and Depth Anything v2 with annotations



Features extracted





Evaluation of YOLOv11 + Depth Anything v2

Evaluation Setup:

- Tested on a sampled subset of the KITTI dataset.
- Metrics used: Precision, Recall, Average Precision (AP).
- Focused classes: Cars, Trucks, Buses, and Other Road Users.

Detection Performance:

- Overall mAP: **6.8%**
- Car class:
  - **Precision:** 50.5%
  - **Recall:** 50.5%
  - **AP (car): 25.5%** (better performance achieved compared to Cursor baseline)

YOLO-3D > detection_model.py > ...

- YOLO-3D
  - visualizatio...
    - vis3d_000... U
    - vis3d_000... U
    - vis3d_000... U
    - vis3d_000... U
    - vis3d_000... U
    - vis3d_000... U
    - vis3d_000... U
    - vis3d_000... U
    - vis3d_000... U
    - vis3d_000... U
    - vis3d_000... U
    - vis3d_000... U
    - vis3d_000... U
  - yolovenv
  - .gitignore
  - bbox3d_uti... M
  - depth_m... 8, M
  - detectio... 4, M
  - evaluate_m... U
  - load_camera_par...
  - output_1745603...
  - output_1745604...
  - output_1745604...
  - output_1745627...
  - output_1745628...
  - output_1745630...
  - output_1745632...
  - README.md
  - requirements.txt
  - run.py M
  - temp.py U
  - trial1.mp4
  - vehicle_feat... U
  - vehicle_feat... U
  - video3.mp4
  - video10.mp4

OUTLINE

```python
import os
import torch
import numpy as np
import cv2
from ultralytics import YOLO
from collections import deque

# Force disable weights_only mode
os.environ['TORCH_WEIGHTS_ONLY'] = '0'

class ObjectDetector:
    """
    Object detection using YOLOv8 from Ultralytics
    """
    def __init__(self, model_size='small', conf_thres=0.25, iou_thres=0.45, classes=None, device=None):
        """
        Initialize the object detector

        Args:
            model_size (str): Model size ('nano', 'small', 'medium', 'large', 'extra')
            conf_thres (float): Confidence threshold for detections
            iou_thres (float): IoU threshold for NMS
            classes (list): List of classes to detect (None for all classes)
            device (str): Device to run inference on ('cuda', 'cpu', 'mps')
        """
        # Determine device
        if device is None:
            if torch.cuda.is_available():
                device = 'cuda'
            elif hasattr(torch, 'backends') and hasattr(torch.backends, 'mps') and torch.backends.mps.is_available():
                device = 'mps'
            else:
                device = 'cpu'

        self.device = device

        # Set MPS fallback for operations not supported on Apple Silicon
        if self.device == 'mps':
            print("Using MPS device with CPU fallback for unsupported operations")
            os.environ['PYTORCH_ENABLE_MPS_FALLBACK'] = '1'

        print(f"Using device: {self.device} for object detection")
```

Problems 39 | Output | Debug Console | Terminal | Ports | Spell Checker 27 | pwsh - YOLO-3D

```
PS C:\Users\farida saad\Desktop\automatic_annot\YOLO-3D>
```

Current Pipeline Status

- **Automatic Annotation:**
  - Using YOLOv11 combined with Depth Anything v2 for real-time pseudo-3D bounding box generation.
- **•Vehicle Feature Extraction:**
  - 3D bounding box dimensions and centroids
  - Orientation (yaw angle)
  - Lateral and longitudinal velocity and acceleration
  - Inter-vehicle distances
  - Occlusion levels (still working on it)
  - Time to Collision (TTC) (still working on it)
- **Automatic Annotation : (I'm Here)**
  - Training MonoLSS on the KITTI dataset for monocular 3D object detection.
  - Evaluating model performance using Average Precision 3D (AP3D) and Bird's Eye View Average Precision (APBEV).
- **•Next Steps:**
  - Compare YOLOv11 + Depth Anything v2 with MonoLSS to select the best model for automatic annotation.

Conclusion

Developed an efficient vehicle-level feature annotation tool using monocular camera input.

Extracted critical vehicle features including velocity, acceleration, dimensions, and time-to-collision without using LiDAR or radar.

Demonstrated the feasibility of using real-time monocular models for detailed 3D object de

Future Recommendations

- **Enhance Feature Extraction:**
  - Refine extraction of occlusion and collision risk metrics.
- **Expand Dataset Size:**
  - Annotate longer video sequences and diverse traffic scenes to strengthen training data.
- **Explore Advanced Models:**
  - Investigate emerging monocular 3D detection models like YOLOBU or MonoFlex once accessible.
- **Automation Improvements:**
  - Develop a fully automatic pipeline minimizing manual intervention.
- **Generalization Testing:**
  - Test the trained pipeline across different datasets to ensure model robustness.

Any Questions?

Thank you for your time. I welcome any questions you may have.