IMAGE PROCESSING - CS4045E

ASSIGNMENT 2, TASK 2

BATCH - 1

<u>PROJECT TITLE</u> : - CROSS-MODAL OCCLUSION REASONING USING DEPTH-FROM-MONOCULAR VISION

TEAM MEMBERS

V DEVANARAYANAN	**B230600CS**	**CS04**
VAISHNAVI V.M.	**B230603CS**	**CS04**
SIVAPRABHA P	**B230569CS**	**CS04**
ATHUL V NARAYANAN	**B230569CS**	**CS03**

1. <u>INTRODUCTION</u>

There are frequently several cars in traffic scenes that partially overlap one another. Because of occlusion, which occurs when one object obscures another, localization and detection become challenging. In order to address this, we combine depth estimation (MiDaS) and object detection (YOLOv8) from a single RGB image using cross-modal reasoning.

The primary concept is to estimate depth maps using MiDaS and detect vehicles using YOLOv8. We can identify which vehicles are occluded (behind) and which are visible (in front) by comparing the average depth of overlapping bounding boxes.

Real-world motivation: In congested environments like roads and intersections, this method aids in autonomous driving, traffic analysis, and occlusion reasoning.

2. <u>OBJECTIVE DESCRIPTION</u>

This project's goal is to combine object detection and depth estimation to make traffic scenes easier to comprehend. Because of their combination, the system is able to distinguish between vehicles that are in plain sight and those that are obscured by other vehicles.

(i) Detecting vehicles and creating depth maps

First, the YOLOv8 model detects vehicles in each image. It is fast and works reliably for multiple cars in a single frame. Then, a depth map is generated from the same image using the MiDaS model.This helps create a 3D understanding of the scene and provides a sense of how far away each vehicle is from the camera.

(ii) To identify which vehicles are visible or occluded based on relative depth

The second objective focuses on occlusion reasoning. By analyzing overlapping bounding boxes from YOLOv8 detections and comparing their corresponding median depth values obtained from MiDaS, the system determines the visibility status of each vehicle. Vehicles with smaller depth values (closer to the camera) are marked as visible, while those with larger depth values are classified as occluded.

(iii) To visualize occlusion results and summarize findings

The third objective is to produce clear visual representations of the results. Each detected vehicle is highlighted using color-coded bounding boxes — green for visible and red for occluded — allowing for easy interpretation of the occlusion order. The visual and tabular outputs are analyzed to summarize how effectively depth estimation aids in distinguishing overlapping vehicles.

(iv) To enhance understanding of computer vision model integration and performance analysis

This is the final objective.it focuses on developing a deeper understanding of how multiple image processing and deep learning models can be integrated to solve real-world vision problems. It also aims to strengthen practical skills in implementing, testing, and interpreting model outputs for meaningful analysis.

3. METHODOLOGY

Step I : Input Images

Traffic scene images are used as the input to the system. These can be either real-world traffic images captured by a camera or samples taken from benchmark datasets such as Cityscapes. Images are loaded in standard RGB format and passed to the object detection and depth estimation modules.

Step II : Object Detection (YOLOv8)

The YOLOv8 model is employed to detect vehicles in the image. Only traffic-related classes (car, bus, truck, motorcycle) are retained.For each detected object, the model returns:

Bounding box coordinates (x1, y1, x2, y2)

Class label

Confidence score

Step III : Depth Estimation (MiDaS)

The same input image is passed through the MiDaS model, which generates a relative depth map. Since MiDaS outputs inverse depth, higher values correspond to objects that are closer to the camera.

The depth map is resized to match the original image resolution, allowing pixel-wise comparison with bounding box regions from YOLO.

Step IV : Occlusion Reasoning

To determine visible vs. occluded objects, the following logic is applied:

(i): Median depth calculation

For each detected bounding box, the median depth of the enclosed pixels is computed.

(ii): Bounding box overlap check

Intersection-over-Union (IoU) is calculated between all object pairs. If IoU > 0.2, the objects are considered overlapping.

(iii): Depth comparison

For overlapping boxes, the object with lower median depth (closer to the camera) is marked visible, and the one with higher median depth is marked occluded.

Step V : Visualization

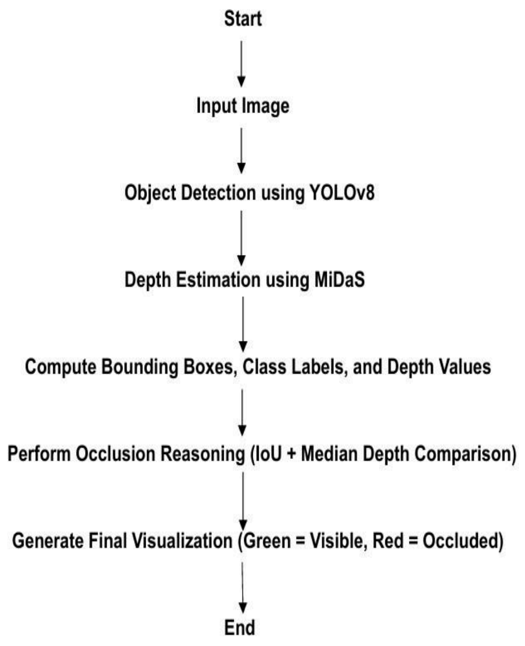
The final output image is generated using OpenCV:

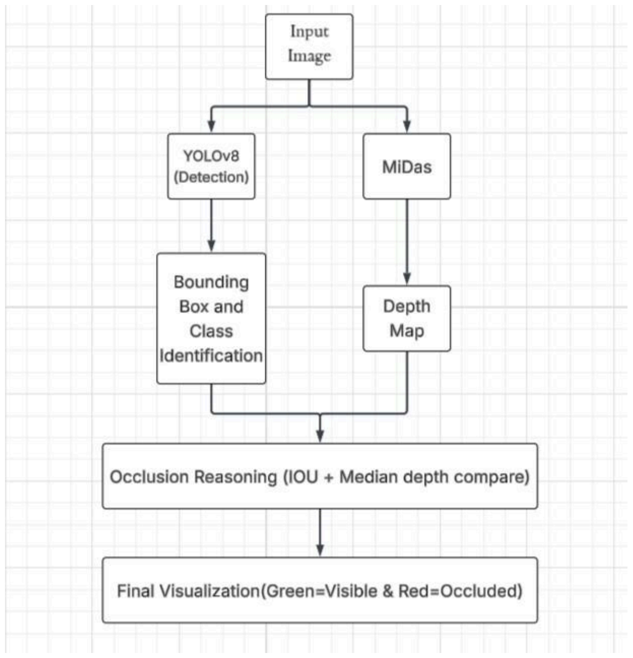
Green bounding box → visible vehicle

Red bounding box → occluded vehicle

Each box is annotated with the class label and median depth value for interpretability. A color-coded depth map is also displayed separately.

A summary table of detection results (ID, class, depth, status) is printed for analysis.


Start
Input Image
Object Detection using YOLOv8
Depth Estimation using MiDaS
Compute Bounding Boxes, Class Labels, and Depth Values
Perform Occlusion Reasoning (IoU + Median Depth Comparison)
Generate Final Visualization (Green = Visible, Red = Occluded)
End


Input Image
YOLOv8 (Detection)
MiDas
Bounding Box and Class Identification
Depth Map
Occlusion Reasoning (IOU + Median depth compare)
Final Visualization(Green=Visible & Red=Occluded)

4. IMPLEMENTATION DETAILS

The entire implementation was carried out using **Python** and **PyTorch**, integrating two pretrained deep learning models — **YOLOv8n** for object detection and **MiDaS** for monocular depth estimation. The workflow follows the four main steps defined in the task: detection, depth generation, occlusion reasoning, and visualization.

(i) Environment Setup

Language: Python 3.10

Frameworks/Libraries: torch, ultralytics, opencv-python, matplotlib, numpy, os

Hardware: Runs on GPU (CUDA) if available, else defaults to CPU.

Models Used:
(i) YOLOv8n.pt – pretrained object detector from the Ultralytics library.

(ii) MiDaS_small – pretrained monocular depth estimation model from Intel ISL.

Input Data: Custom traffic image (e.g., Traffic_Road_Image_10.jpg).

(ii) Object Detection (YOLOv8)

The YOLOv8n model detects only vehicle classes (car, bus, truck, motorcycle) by filtering COCO labels [2, 3, 5, 7].
Each detection returns bounding box coordinates (x1, y1, x2, y2), class label, and confidence score.
All detected vehicles are stored in a structured list of dictionaries for later processing.

(iii) Depth Estimation (MiDaS)

The input image is converted from BGR to RGB and passed through the MiDaS _small model.
A corresponding depth map is generated and resized to match the original image dimensions using bicubic interpolation.
Since MiDaS outputs inverse depth, higher pixel values indicate objects closer to the camera.

(iv) Occlusion Reasoning

To determine which vehicles are visible or occluded:

1. Median depth inside each bounding box is calculated from the depth map.

2. Intersection-over-Union (IoU) is computed between every pair of detected boxes using a custom calculate_iou() function.

3. If IoU > 0.2 (indicating overlap), the bounding box with the higher median depth is marked as visible (closer), and the other as occluded.

This logic is implemented through nested loops comparing all detection pairs.

(iv) Visualization and Output

The final image is drawn using OpenCV:

- Green boxes → visible vehicles
- Red boxes → occluded vehicles

Each box also displays the class name and median depth value.

Two visual outputs are displayed using Matplotlib:

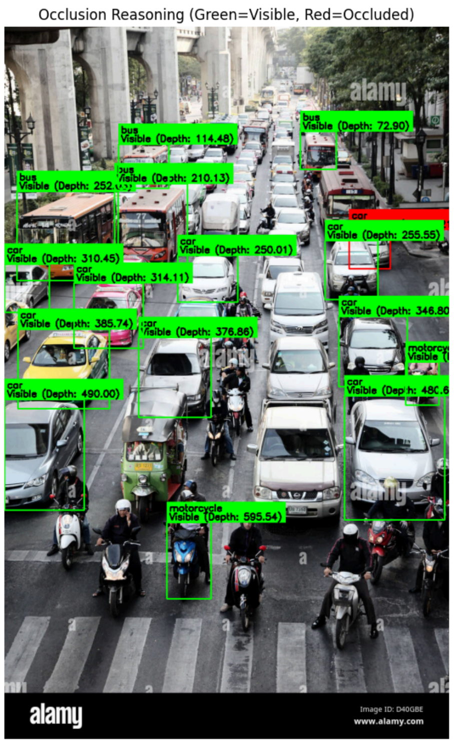
1. Occlusion Visualization: Image with visible (green) and occluded (red) boxes.
2. Depth Map: Color-coded representation where lighter areas denote nearer objects.

Additionally, the script prints a detection summary table in the terminal, listing each vehicle's class, median depth, and occlusion status.

5. RESULTS AND DISCUSSION

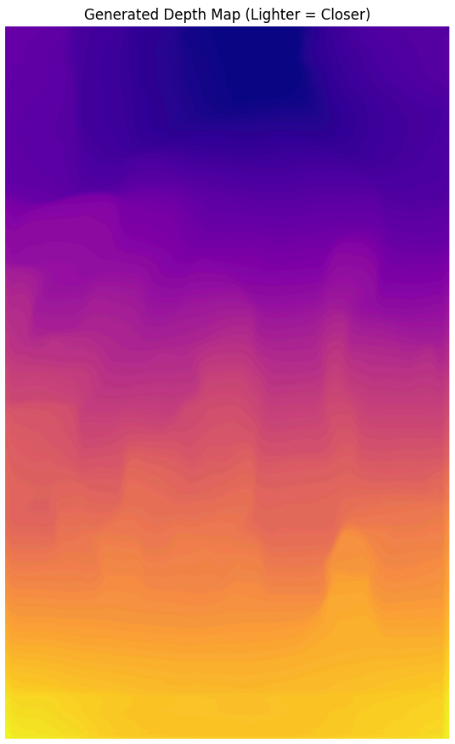
Sample Visualisation

Original Image



Occlusion Reasoning (Green=Visible, Red=Occluded)
bus Visible (Depth: 114.48)
bus Visible (Depth: 72.90)
bus Visible (Depth: 252.0)
bus Visible (Depth: 210.13)
car
car Visible (Depth: 255.55)
car Visible (Depth: 250.01)
car Visible (Depth: 310.45)
car Visible (Depth: 314.11)
car Visible (Depth: 346.80
car Visible (Depth: 385.74)
car Visible (Depth: 376.86)
motorcyc Visible
car Visible (Depth: 480.6
car Visible (Depth: 490.00)
motorcycle Visible (Depth: 595.54)
alamy
Image ID: D40GBE
www.alamy.com

Occlusion Visualisation



Depth Map

Table for the above image

Detection Results Table			
ID	Class	Median Depth	Occlusion Status
0	bus	252.03	Visible
1	bus	210.13	Visible
2	car	385.74	Visible
3	car	490.00	Visible
4	car	346.80	Visible
5	car	480.66	Visible
6	car	213.54	Occluded
7	car	376.86	Visible
8	car	255.55	Visible
9	bus	114.48	Visible
10	car	250.01	Visible
11	car	314.11	Visible
12	car	310.45	Visible
13	bus	72.90	Visible
14	motorcycle	385.37	Visible
15	motorcycle	595.54	Visible

Depth ordering helps identify occluded vehicles by arranging detected objects based on their distance from the camera. When two vehicles overlap in an image, comparing their depth values reveals which one is in front and which is behind, enabling accurate occlusion detection. It also refines segmentation and supports consistent tracking even when a vehicle is partially hidden. However, depth estimation can become unreliable in **low-light** or **low-texture** conditions, leading to **noisy or inaccurate depth maps**, which may reduce occlusion detection accuracy.

6. CONCLUSIONS

In this project, object detection and depth estimation were combined to understand occlusion in traffic scenes more effectively. The YOLOv8 model successfully identified vehicles in each frame, while the MiDaS model provided relative depth information to estimate which vehicles were in front or hidden behind others. Together, these two methods helped create a clearer picture of how objects interact within a scene.

This method can greatly enhance autonomous vehicles' perception capabilities. the system gains a better understanding of which vehicles are closer and which are hidden behind others by combining object detection with depth estimation.This helps self-driving cars make safer decisions when navigating through crowded roads, overtaking, or maintaining safe distances.

In traffic analysis, depth-based occlusion reasoning allows for more accurate counting and tracking of vehicles, even when some are partially blocked in the camera view. It also helps in analyzing congestion patterns and movement flow in complex intersections or dense urban areas. Overall, this technique enhances both safety and efficiency in intelligent transportation systems

Possible Extensions-

- To improve occlusion reasoning and acquire more precise depth measurements, the project can be expanded by incorporating LiDAR or stereo vision sensors.
- Another extension could involve using real-time video input instead of static images, allowing continuous tracking of vehicles across frames.
- To improve detection accuracy in a variety of lighting and weather scenarios, YOLOv8 and MiDaS could be fine-tuned on larger, domain-specific traffic datasets.
- future improvement might include developing a combined end-to-end model that performs detection and depth estimation simultaneously for faster inference.

7. REFERENCES

1.Ranftl, R., Lasinger, K., Hafner, D., Schindler, K., & Koltun, V. (2020). *Towards Robust Monocular Depth Estimation.* IEEE TPAMI.

2.Jocher, G., Chaurasia, A., & Qiu, J. (2023). *YOLOv8 by Ultralytics.* GitHub repository.

3.Intel ISL. (2024). *MiDaS – Monocular Depth Estimation.* GitHub repository.

4.R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, Pearson Education, 2018.

5.A. K. Jain, *Fundamentals of Digital Image Processing*, Pearson Education India, 2015.