Project 3 - Einstein Vision

Karthik Mundanad
Robotics Engineering Department
Worcester Polytechnic Institute
Email: krmundanad@wpi.edu

Kushagra Srivastava
Robotics Engineering Department
Worcester Polytechnic Institute
Email: ksrivastava1@wpi.edu

Using 3 LATE days

***Abstract*—The project focuses on visualizing and recreating the front-end dashboard of a self-driving automobile, which is capable of detecting and perceiving different elements on the road, such as lanes, vehicles, pedestrians, traffic lights, signs, and so forth. This aids in determining the next course of action for the vehicle's planning stack. For the task, a variety of methods and deep learning models are used. An overview of the detection and rendering processes for each component is provided in this report. It is primarily split into three stages. The recognition of vehicles, pedestrians (without pose), lanes, and significant road signs is the focus of the first phase. In the second phase, the cars are subclassified and several objects, such as dustbins and traffic cones, are detected in the scene. The third stage involved identifying indicators on the vehicles along with distinguishing moving from parked cars.**

I. Introduction and Blender

The project employs multiple deep-learning models to detect various objects within the scene, which are then simulated in Blender. In this process, objects are spawned at different locations within the scene according to their detections. Information is stored using JSON files for each network, along with blend files for the objects. Scene semantics are then rendered for each frame and saved as a sequence of images. Various Blender functionalities, including Bezier curves, meshes, and object files, are utilized for this purpose. The following sections delve into the detection procedure for each component and the corresponding rendering methods in detail.

II. Phase - 1

A. Identification of Vehicles, Pedestrians and other objects

The project extensively relies on Meta's Detic [1] to detect various objects in the scene, including vehicles, pedestrians, stop signs, dustbins, and traffic cones, among others. Leveraging CLIP integration, this network provides specific object masks and bounding boxes based on custom text prompts. While other models such as Grounded SAM [2], [3], [4] were tested, their high inference time and minimal performance enhancement led to their exclusion from the final rendering process.

To facilitate rendering, a depth estimation network is required to ensure consistency and generalizability of outputs. Various networks, including MiDaS [5] and ZoeDepth [6], were evaluated. While ZoeDepth demonstrated satisfactory results, particularly with lanes as discussed in the subsequent section, it faltered in numerous scenarios. To address these limitations and enhance depth estimation, Marigold was employed. Marigold consistently provided depth estimates along with a fixed scale. An example of the depth map can be seen in Figure 3. By combining the output masks from Detic and depth information from Marigold, objects could be accurately spawned in the simulation environment.

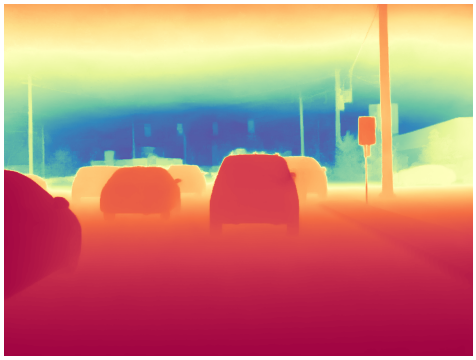


Fig. 1: Depth Map from Marigold

However, despite the acceptability of results obtained through these methods, several drawbacks persist. False detections by Detic remain a challenge, although adjusting confidence levels can mitigate this issue, albeit with varying efficacy across scenes. Additionally, Marigold's depth estimation is less accurate for nearby objects, especially evident in lane scenes where convergence of lines occurs.

B. Traffic Lights

The project utilized a YOLO v3 model pre-trained on the LISA Dataset [7] to detect traffic lights and arrows in the scenes. While Detic provides traffic light masks, which can be processed using various thresholding methods to detect colors, the outputs are heavily influenced by scene conditions. The YOLO v3 model, on the other hand, offers distinct classes such as 'stop', 'warning', and 'go', including variations like 'stop left' and 'go left', facilitating direct rendering of traffic lights based on depth information from Marigold depth maps.

However, the YOLO v3 model exhibits limitations in generalizability, particularly when multiple traffic lights are present. Moreover, it tends to be biased towards detecting 'go left' arrows, leading to false detections in certain cases.

((a)) Traffic Cone Detection using Detic

((b)) Dustbin Detection using Detic

Fig. 2: Object Detection example using Detic

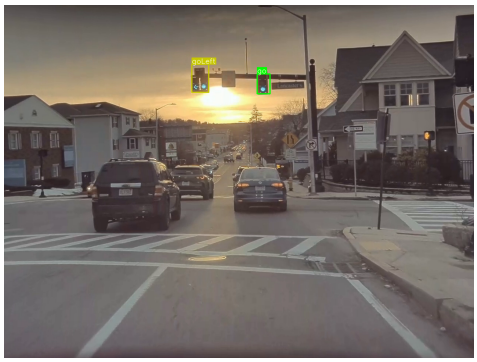


Fig. 3: Traffic light mask using YOLO v3

C. Lanes

Various deep learning frameworks were evaluated for lane detection, each with its own limitations. ClRNet and its variants [8] demonstrated good performance in highway scenarios, but their output of separate instances for different lanes made lane classification challenging. Classical methods, while providing separate instances, struggled with curved lanes.

Ultimately, a MASK RCNN with ResNet50 backbone, utilizing pre-trained weights [9], was tested. This model successfully detected various classes such as 'Solid line', 'Divider line', 'Dashed line', and 'Road Sign lines'. The distinct masks for dashed lines allowed the direct fitting of lanes in Blender using their endpoints. However, when lanes were too close to the camera, sampling points on contoured lanes and fitting Bezier curves did not yield satisfactory results. As a workaround, a row-wise average was taken along the image to obtain aligned points.

Despite its effectiveness, the Mask RCNN network exhibits shortcomings in certain scenarios, such as the detection of curbs and curbside lanes. Moreover, its reliability diminishes when lanes are near the vehicle, such as in city road scenarios.

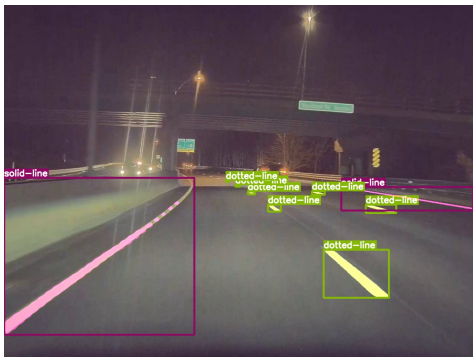


Fig. 4: Road Sign detection using Mask RCNN

III. Phase-2

A. Vehicle Subclassification and other objects

As outlined in Section II-A, Detic boasts an extensive custom vocabulary encompassing various objects, including different subtypes of vehicles such as sedans, SUVs, pickup trucks, trucks, motorcycles, bicycles, and more. Achieving acceptable results necessitated fine-tuning the confidence levels associated with each class. While classes like motorcycles and bicycles were detected straightforwardly, the presence of multiple prompts per image required a systematic sorting of subclasses to ensure accurate detection.

This method effectively filters out many outliers, yet occasional misclassifications may occur, particularly depending on the depth of the scene. Despite this, the systematic approach employed significantly improves the overall accuracy of vehicle detection within the scene.

B. Vehicle Pose Estimation

In the task of obtaining 3D bounding boxes from Detic's 2D bounding box outputs, we faced challenges due to the limited availability of open-source networks. To address this, we turned to a YOLO3D network. However, the pre-trained weights proved to be inaccurate, often resulting in significant offsets.

To mitigate this issue, we employed a workaround by passing the 2D bounding box from Detic to the regressor of the YOLO3D network. The output, specifically the yaw, was then utilized to spawn the vehicles accordingly. The results of this approach are depicted in Figure **??**.

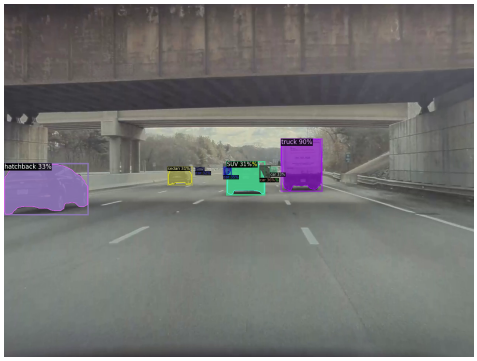


Fig. 5: Detic Vehicle detection and subclassification of vehicles

Despite the effectiveness of this method, certain inaccuracies were observed, particularly with trucks and pickup trucks. Additionally, orientation errors became apparent when objects were occluded or close to the camera. While this approach provided satisfactory results in many cases, further refinement may be necessary to enhance accuracy, particularly for complex objects and challenging scenarios.

Fig. 6: 3D pose detection of vehicles

C. Pedestrian Pose Estimation

Initially, pedestrian poses were detected using a YOLO V8 and YOLO NAS model [10] with pre-trained weights. The strategy involved extracting keypoints and projecting them in 3D, then feeding this information to Blender's armature function. However, the proximity of key points led to similar depth estimates for joints, complicating the armature function's comprehension. To streamline the process, we opted for networks capable of directly providing poses in file formats supported by Blender. One such network is OSX [11], which detects human poses and generates meshes in .obj format using SMPL models [12]. However, the meshes generated were in a coordinate frame different from ours, necessitating modification of the provided code to align the mesh with Blender's world coordinate system using depth masks generated earlier. The results showcased in Figure **??** and 7 demonstrate accurate estimates, eliminating the need for separate methods for pose estimation and rendering.

Fig. 7: Pedestrian Pose visualization using OSX

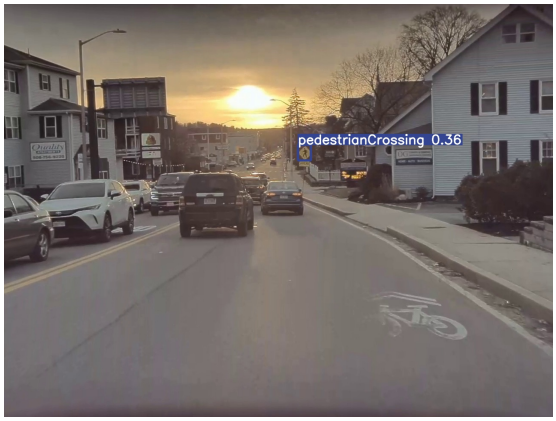
D. Traffic Sign Detection

A custom YOLO v8 model, specifically trained on the LISA Traffic Sign dataset, was meticulously developed to detect an array of signs commonly encountered on roads. These included crucial indicators such as stops, pedestrian crossings, lane ends, speed limits, and several others. Through rigorous training over 25 epochs, the model achieved an impressive validation mAP of 0.96, indicating its robust performance in accurately identifying traffic signs within diverse scenes. The results, as showcased in Figures 12 and **??**, highlight the model's proficiency in detecting these signs, thus affirming its efficacy in enhancing road safety and navigation. Notably, the model was engineered to provide detection for a wide range of sign classes, totaling 46, which encompassed various speed limits and speed breaker signs commonly encountered on roadways.

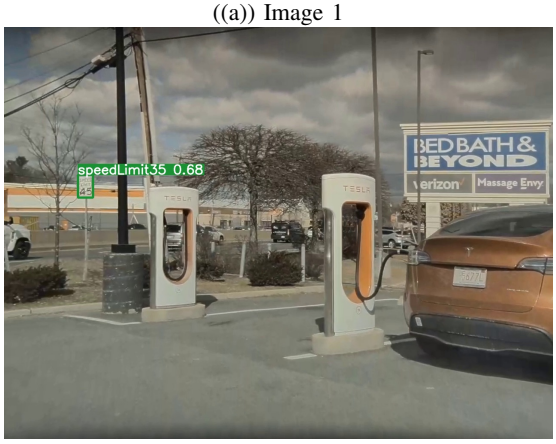
However, an inherent bias towards the "35mph" sign was observed in the model's detections pertaining to speed limits. Despite efforts to mitigate this bias through the implementation of classical methods, the endeavor was hindered by time constraints, resulting in less-than-optimal outcomes. Nonetheless, the model's overall performance and its ability to accurately identify a multitude of traffic signs underscore its significance in enhancing traffic management and safety measures on roads.

E. Road Signs

Several networks designed for road sign detection are primarily trained on European datasets, making them less accurate for our specific scenario. To circumvent this limitation, we opted for a more streamlined approach by leveraging the Mask RCNN discussed in Section II-C. This versatile model allowed us to directly create a mesh from its output, eliminating the need for additional sampling or curve fitting, as required for lanes. This elegant method not only simplified the process but also ensured accurate detection of road signs tailored to our environment. The results obtained through this approach are depicted in Figure 9 and **??**, showcasing the effectiveness and precision of the method in rendering road signs within our scene.



((a)) Image 1



((b)) Image 2

Fig. 8: Traffic Sign Detection using YOLO v8. Image 2 indicates speed limit bias.

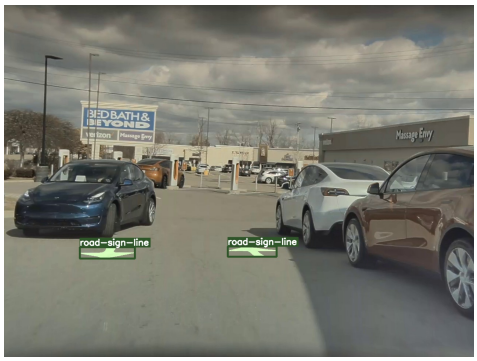


Fig. 9: Road Sign detection using Mask RCNN

IV. Phase-3

A. Brake and Turning Lights

The Detic model is employed for car taillight detection, with subsequent bounding box acquisition. Following BGR to YCrCb conversion, erosion is applied using a 5x5 kernel to enhance image clarity. Adaptive thresholding is then utilized to retain pixels surpassing a specified luminance threshold. A pixel count threshold of 220 is set to classify taillights as ”on”. In each frame, bounding boxes of all cars are retrieved, and centroid points for detected taillights within each box are established as pairs. The signaling behavior of vehicles is determined through conditional checks. Specifically, if the right taillight is classified as ”on” while the left is ”off”, the car is interpreted as indicating a right turn, and vice versa. Conversely, if both taillights are ”on”, the vehicle is considered to be braking, while both being ”off” suggests the absence of braking signals. Despite its effectiveness in precise taillight detection and accurate inference of vehicle signaling behavior, this methodology exhibits limitations in generalizability between day and night scenes, primarily due to variations in taillight color intensity caused by sunlight.

B. Classification of Parked and Moving Vehicles

Distinguishing between parked and moving vehicles presents numerous challenges, encompassing various edge cases. Our approach relies on utilizing the optical flow estimate provided by RAFT [13]. Initially, we employed a straightforward method of thresholding the flow image based on the Sampson distance to distinguish dynamic obstacles from relatively stationary objects. However, this method has several limitations. Firstly, vehicles in front of us or those at a distance exhibit low flow rates, leading to potential misclassification. Additionally, the threshold for distinguishing between parked and moving vehicles varies from frame to frame and scene to scene.

To address these shortcomings, we augmented the original flow map with certain enhancements. To account for the flow of our own vehicle, we created a 1D Gaussian mask for each image, with the variance of the Gaussian proportional to the variance of flow in the image. This adjustment acknowledges that flow towards the edges tends to be greater than in the center. Furthermore, we computed and compared the flow within the bounding box of the vehicle with the flow in the surrounding region. We set a threshold on this net flow, as well as an adaptive threshold based on the flow of the entire image, for the absolute flow within the bounding box. This adaptive threshold considers that the absolute flow of moving vehicles in front may be low and thus should be disregarded. These refinements aim to enhance the accuracy and robustness of distinguishing between parked and moving vehicles. The results are presented in **??** and 11

C. Extra Credit - Speed Breakers

Speed breakers were detected based on 3 factors. The speed breaker sign which is detected using the YOLO v8 Traffic sign network, the road sign which is present on the speed breaker, and the brake lights of cars while on a speed breaker. A combination of these methods is necessary for the detection of a speed breaker as only one or more factors may be absent or inconsistent depending on the scene.

V. Conclusion

To sum up, our project is a thorough undertaking meant to progress the domain of scene interpretation and computer

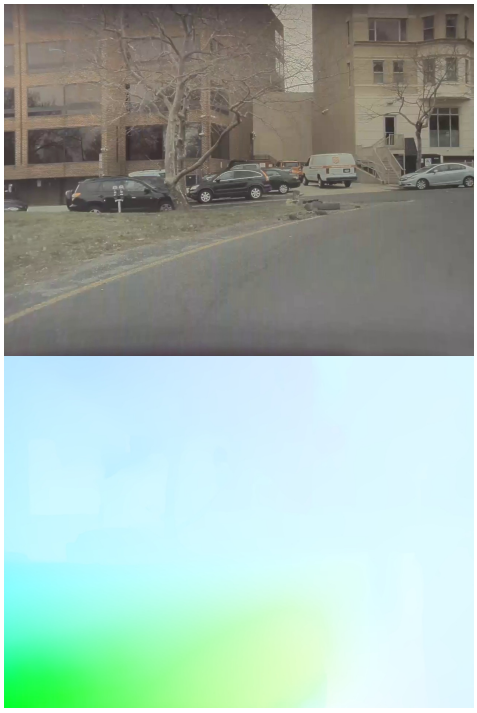


Fig. 10: Optical Flow image

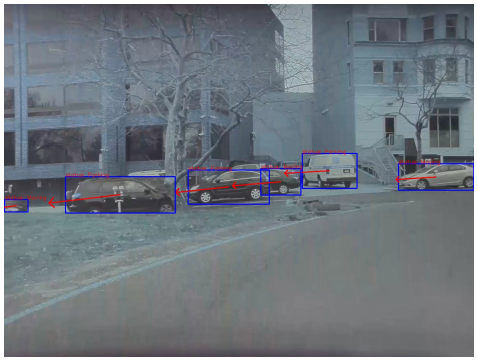


Fig. 11: Classifying based on Parked and moving cars - Arrows indicate the flow rate

vision for traffic control applications. By combining cutting-edge models and techniques, we have created a strong framework that can recognize and comprehend a wide range of objects in urban settings, such as lane markings, traffic signs, pedestrian positions, and car actions. Utilizing state-of-the-art methods including object detection, signal interpretation, and 3D position estimation, we have accomplished impressive results in traffic analysis and scene reconstruction. Although we faced obstacles and constraints in the process, our work demonstrates how cutting-edge computer vision techniques might improve traffic efficiency, road safety, and urban mobility. Further development and improvement of our system may eventually lead to even more sophisticated and impactful solutions in the realm of intelligent transportation systems.

((a)) Speed Breaker traffic sign

((b)) Speed Breaker Road Sign

Fig. 12: Speed Breaker Detection

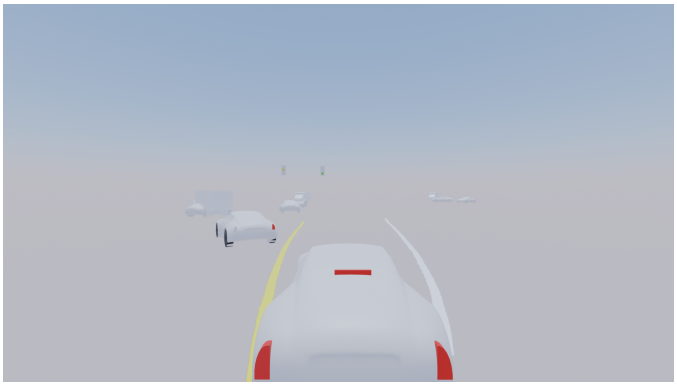


Fig. 13: Blender Output 1

REFERENCES

[1] X. Zhou, R. Girdhar, A. Joulin, P. Krähenbühl, and I. Misra, "Detecting twenty-thousand classes using image-level supervision," in *ECCV*, 2022.

[2] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollár, and R. Girshick, "Segment anything," *arXiv:2304.02643*, 2023.

[3] S. Liu, Z. Zeng, T. Ren, F. Li, H. Zhang, J. Yang, C. Li, J. Yang, H. Su, J. Zhu *et al.*, "Grounding dino: Marrying dino with grounded pre-training for open-set object detection," *arXiv preprint arXiv:2303.05499*, 2023.

[4] T. Ren, S. Liu, A. Zeng, J. Lin, K. Li, H. Cao, J. Chen, X. Huang, Y. Chen, F. Yan, Z. Zeng, H. Zhang, F. Li, J. Yang, H. Li, Q. Jiang, and

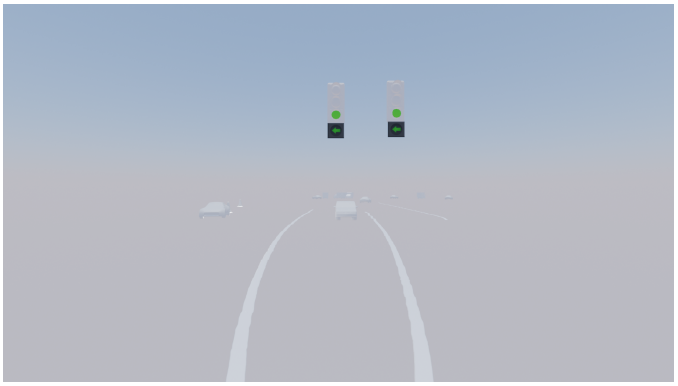


Fig. 14: Bledner Output 2

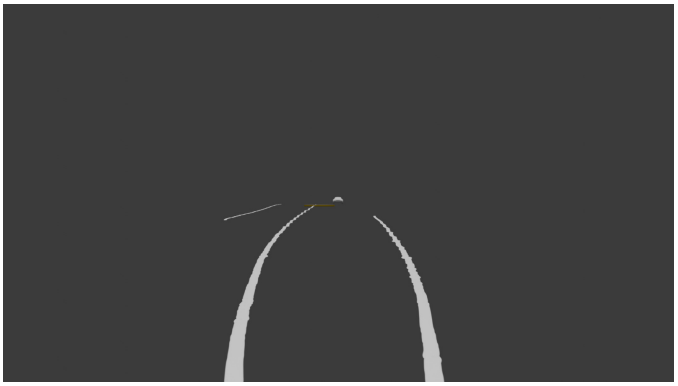


Fig. 15: Bledner Output of Dustbin

L. Zhang, "Grounded sam: Assembling open-world models for diverse visual tasks," 2024.

[5] R. Birkl, D. Wofk, and M. Müller, "Midas v3.1 – a model zoo for robust monocular relative depth estimation," *arXiv preprint arXiv:2307.14460*, 2023.

[6] S. F. Bhat, R. Birkl, D. Wofk, P. Wonka, and M. Müller, "Zoedepth: Zero-shot transfer by combining relative and metric depth," *arXiv preprint arXiv:2302.12288*, 2023.

[7] sovit 123, "Sovit-123/traffic-light-detection-using-yolov3: Traffic light detection using deep learning with the yolov3 framework. pytorch =¿ yolov3." [Online]. Available: https://github.com/sovit-123/Traffic-Light-Detection-Using-YOLOv3?tab=readme-ov-file

[8] T. Zheng, Y. Huang, Y. Liu, W. Tang, Z. Yang, D. Cai, and X. He, "Clrnet: Cross layer refinement network for lane detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022, pp. 898–907.

[9] sovit 123, "Sovit-123/traffic-light-detection-using-yolov3: Traffic light detection using deep learning with the yolov3 framework. pytorch =¿ yolov3." [Online]. Available: https://github.com/sovit-123/Traffic-Light-Detection-Using-YOLOv3?tab=readme-ov-file

[10] S. Aharon, Louis-Dupont, Ofri Masad, K. Yurkova, Lotem Fridman, Lkdci, E. Khvedchenya, R. Rubin, N. Bagrov, B. Tymchenko, T. Keren, A. Zhilko, and Eran-Deci, "Super-gradients," 2021. [Online]. Available: https://zenodo.org/record/7789328

[11] J. Lin, A. Zeng, H. Wang, L. Zhang, and Y. Li, "One-stage 3d whole-body mesh recovery with component aware transformer," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 21 159–21 168.

[12] J. Williams. [Online]. Available: https://smpl.is.tue.mpg.de/

[13] Z. Teed and J. Deng, "RAFT: recurrent all-pairs field transforms for optical flow," *CoRR*, vol. abs/2003.12039, 2020. [Online]. Available: https://arxiv.org/abs/2003.12039