RBE/CS 549 Computer Vision
Project 3
Einstein Vision

Puneet Shetty
MS in Robotics
Worcester Polytechnic Institute
Email: ppshetty@wpi.edu

Edwin Clement
MS in Robotics
Worcester Polytechnic Institute
Email: eclement@wpi.edu

***Abstract*—Using camera data from Tesla automobiles, this project creates immersive environments by detecting objects such as cars, pedestrians, traffic lights, traffic signs, road signs, and car poses. The objective is to smoothly include these detections into movies that are created with Blender, offering an accurate portrayal of actual situations. Accurate identification and precise scene reconstruction are made possible by sophisticated computer vision algorithms, which also make it possible to create high-fidelity movies that are necessary for a variety of applications, including traffic simulation settings and autonomous driving research.**

Index Terms*— *YOLO, Detic, Optical Flow, Human pose estimation, Traffic sign, Traffic signal, Mask RCNN

I. Pipeline Overview

In order to obtain intrinsic parameters and adjust the raw camera output for distortion correction, our pipeline started with the camera's calibration. We then used the corrected footage to apply sub-sampling, at a rate of 10fps. These crucial frames were taken out of every driving clip and fed into the models that were used to render the semantics of the scene. After that, we used a Marigold depth estimation network to extract the necessary feature's 3D world coordinates from the network models. After that, frame by frame, the features were placed at their assigned coordinates, and Blender was used to generate the video.

II. Feature Extraction using Pre-trained Models

A. Lane & Ground Sign Detection

Using a mask RCNN model, we were able to precisely extract road sign outlines and lane lines from every frame in our method. Detecting and segmenting lanes is a difficult task because of the nature of the objects that need to be handled. For example, lane lines are generally thin objects that are located at significant distances inside the image. However, we are still dedicated to enhancing the performance of our model in order to obtain the most precise lane recognition results via instance segmentation. In our quest for optimization, we made use of the state-of-the-art Mask RCNN ResNet50 FPN V2, which has excellent pretrained weights. This improved model

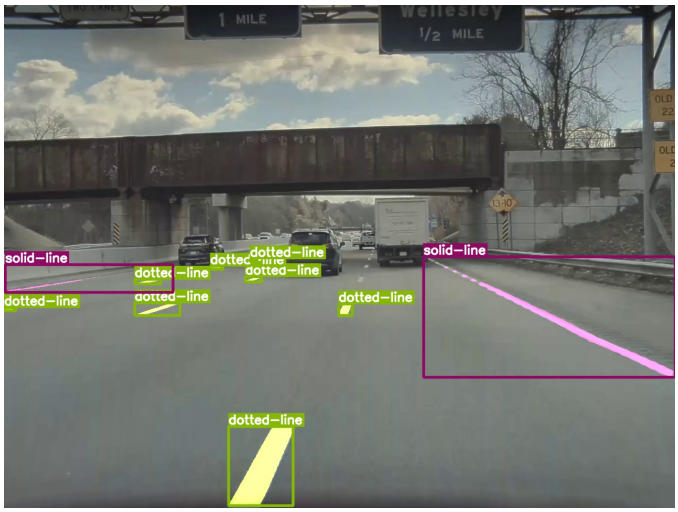

Fig. 1. Lane Detection

improves our performance metrics and strengthens our segmentation capabilities. With a mask mean Average Precision (mAP) of 41.8% and a box mAP of 47.4%, we are equipped with a robust framework poised to deliver exceptional results in lane detection using the pre-trained model. The classes detected by this network were:

- Dashed line
- Solid White lines
- Double yellow lines
- Road Sign lines

To assist the mapping procedure, we made two important assumptions in order to translate the 2D lane detection outputs from our network into a 3D metric space. We assumed that all lanes had zero height, which simplified the model. By making use of these presumptions, we were able to provide a fixed z-coordinate of 0 meters to each predicted lane pixel, giving it an exact 3D position in metric units. After obtaining the 3D positions of every lane, the last step we took was to use cubic bezier curve fitting to draw the lanes. The control points

Fig. 2. Ground Sign

obtained from this fitting procedure were very helpful inputs to the Blender framework's lane geometry node.

Using the mask produced by our dedicated ground sign identification network, we implemented a painstaking contour extraction process in our ground sign handling methodology. Through the process of separating the non-zero components of this mask, we were able to accurately draw the outlines of the ground signs and maintain their originality. We then used these contours to easily import the extracted shapes into Blender, which improved the precision and realism of our produced scenes. We used the same mapping procedure we had used for the lane lines

However, the model's usage of heavier RPN, box, and mask heads is a drawback. As a result, the model operates more slowly than in its prior iteration.

B. Traffic Light Detection

We discovered a traffic light identification model that used the YOLOv3 framework and the LISA traffic light dataset for training. With this model, we can immediately identify the color and arrow of traffic lights. Incorporating the identified traffic lights into Blender involved a multi-step process aimed at seamless integration. Initially, we leveraged the depth estimation results to derive precise 3D coordinates for each detected feature. Subsequently, utilizing this spatial information, we meticulously crafted separate color elements within Blender to accurately represent the detected traffic signals. By aligning these color elements with the corresponding elements of the traffic signal model, we ensured faithful portrayal of the detections made by our network within the rendered scenes. This integration methodology not only enhances the visual realism of our simulations but also underscores the effectiveness of our traffic light identification system within a 3D environment.

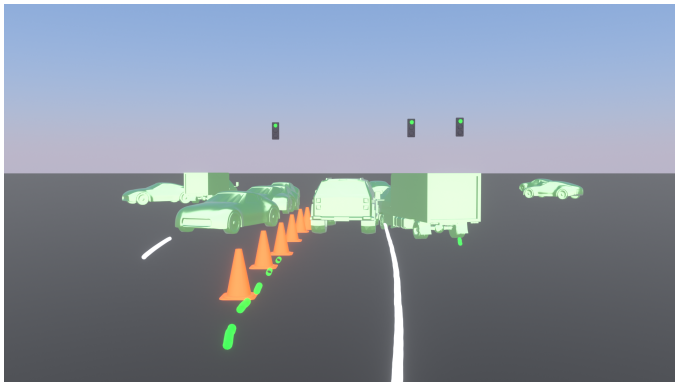

Fig. 3. Traffic Lights

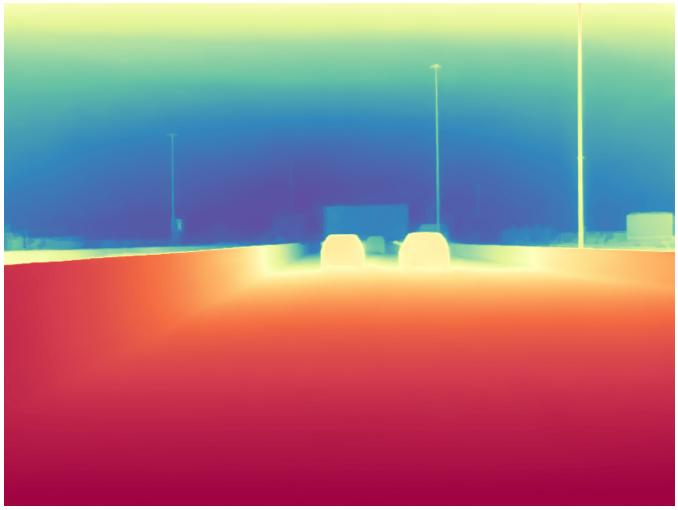

Fig. 4. Marigold Depth

C. Depth Estimation

The Marigold network, a state-of-the-art diffusion model with an advanced fine-tuning strategy designed specifically for monocular depth estimation applications, drove our depth estimate. Fundamentally, Marigold makes use of the abundance of visual information included in modern generative image models. Based on Stable Diffusion and optimized with synthetic data, the model has exceptional flexibility, able to automatically adjust to new data via zero-shot transfer learning. This special capacity guarantees the delivery of cutting-edge monocular depth estimation outcomes. We prepared ourselves with rich spatial information essential for our 2D-to-3D point conversion process—a core component integrated across all aspects of our project—by extracting depth values for each pixel in the range of 0 to 255 from every frame.

D. Traffic Sign Detection

Our approach to traffic sign detection relied on the robust capabilities of the YOLOv8 pre-trained model, renowned for its nuanced discernment among various classes of traffic signs. With a comprehensive classification spectrum encompassing stop signs, speed limit indicators ranging from 35 to 60,

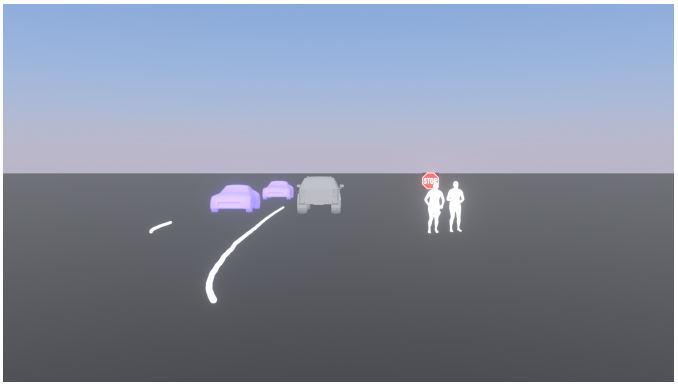

Fig. 5. Initial Blender Detections

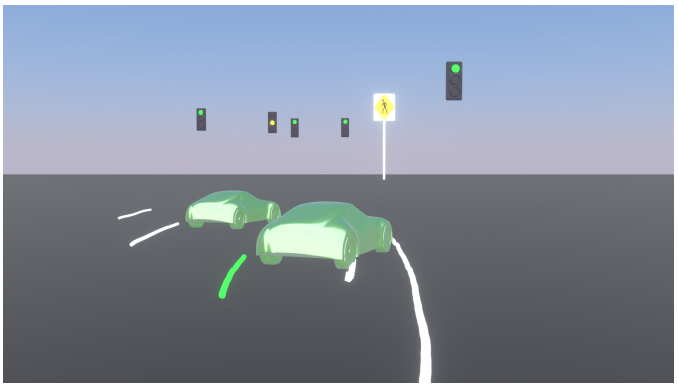

Fig. 6. Traffic Sign

pedestrian crossing symbols, lane merge warnings, and alerts for upcoming speed bumps, the model provided a versatile solution for real-time traffic sign identification. Upon detection, the model generated masks delineating the precise locations of these signs within the image. These masks were then meticulously processed, extracting the pertinent sign details and converting them into accurate 3D coordinates. This conversion facilitated the seamless integration of the detected traffic signs into our Blender environment, enriching the visual fidelity and functional realism of our simulated world.

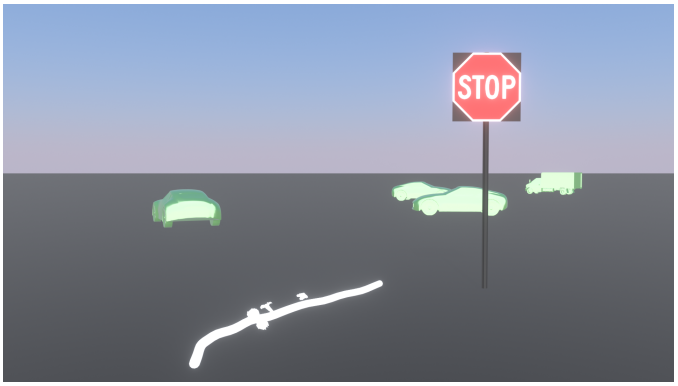

Fig. 7. Stop Sign

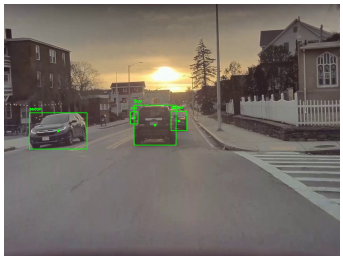

Fig. 8. Car Detections

Fig. 9. Garbage Can Detections

E. Identification of cars, garbage cans, traffic cones

We were able to identify a wide range of classes using Meta's Detic model, in addition to traffic signs. These classes included things like garbage cans, traffic cones, fire hydrants, and different kinds of vehicles, like trucks, pickup trucks, SUVs, sedans, hatchbacks, motorcycles, and bicycles. One of Detic's unique features is its capacity to identify any class using the class names that are supplied, utilizing CLIP technology to improve accuracy. Detic has demonstrated remarkable cross-dataset generalization abilities, showcasing its detection prowess on datasets such as OpenImages and Objects365 without the need for fine-tuning. It was trained on the ImageNet-21K dataset, which contains a wide variety of 21,000 classes. Because of its cutting-edge performance on both the open-vocabulary LVIS and COCO datasets, it can be used as a flexible solution for a variety of detection applications, including ones that need DETR-style detectors. We were able to classify these objects with ease by providing Detic with a bespoke vocabulary that was particular to the classes of interest. We then used the bounding box outputs to make it easier to apply each class in subsequent projects. This all-encompassing strategy not only expedited the classification procedure but also established a strong framework for future project advancements and improvements.

Fig. 10. Traffic Cones

Fig. 11. Vehicle Pose

F. 3D Vehicle Pose Estimation

We used Detic to do vehicle detection duties, as was previously mentioned in the section on vehicle identification of the first checkpoint of our project. Detic's powerful features went beyond simple identification and allowed cars to be divided into several groups. Its extensive training routine on numerous datasets, each painstakingly annotated to cover a broad range of vehicle types and classifications, produced this capabilities. By utilizing the extensive knowledge present in these datasets, Detic was able to differentiate between a wide range of vehicle classifications and provide a level of accuracy and granularity in vehicle identification that was previously unmatched.

We discovered a functional GitHub repository that effectively implements the methodology proposed in a 3D pose estimation paper. This comprehensive framework comprises two distinct networks, each serving a specific purpose. The first network focuses on 2D bounding box detection, accurately identifying the spatial boundaries of objects within an image. Meanwhile, the second network specializes in dimension and orientation estimation, providing crucial information necessary for inferring the 3D pose and bounding box dimensions. By integrating the outputs from both networks, the framework enables precise estimation of 3D poses and bounding boxes, enhancing the overall depth and accuracy of object detection in three-dimensional space.

We improved our implementation's capabilities to achieve precise 3D bounding box estimation. First, by swapping out the projection matrix for our camera's intrinsic matrix and guaranteeing ground plane alignment, we fixed the bounding box tilt problem. Second, we estimated the real-world dimensions of the objects to extend 3D pose estimation capabilities to untrained objects, allowing the model to successfully infer their poses. Together, these changes optimize performance and address issues for reliable 3D object detection and posture estimation.

Even with our changes, there are still issues. Due to the large amount of data in the KITTI dataset, orientation estimations for vehicles are still reliable, but for other objects—like trucks—especially those with fewer instances available for training, the dependability is lower. Larger datasets may be required in order to solve this problem and enhance model performance. Challenges also arise from the method's fundamental assumptions, like uniform stance within object categories. Inaccuracies in depth estimation can result from misclassifications and variations in object size, particularly when objects are obscured. These problems might be addressed by subclassifying cars and improving classification accuracy, which would provide more precise approximations for a range of item sizes and kinds. Nonetheless, difficulties continue to arise when objects are obscured, jeopardizing estimates of both dimensions and orientation.

G. Pedestrian Pose Estimation

To precisely estimate pedestrian positions within our frames, we utilized the One-Stage 3D Whole-Body Mesh Recovery with Component Aware Transformer (OSX). We were able to extract the intricate meshes of every pedestrian in the picture thanks to our sophisticated framework. By using these models, we were able to obtain all of the 2D coordinates of pedestrians, accounting for situations in which the centroid might not line up with the center because of things like people riding motorcycles. We used YOLOv3 to apply a voting mechanism to the mesh coordinates in order to find the centroid. We then used the centroid that we had located to extract the exact 3D coordinates of the pedestrians and project their meshes onto the appropriate spots in the scene. This all-encompassing method ensured the authenticity of our scene reconstructions by enabling precise pedestrian location and portrayal.

H. Brake lights & Turning Signals

In our process of detecting car taillights, we utilize the Detic model to identify these crucial components. Once the

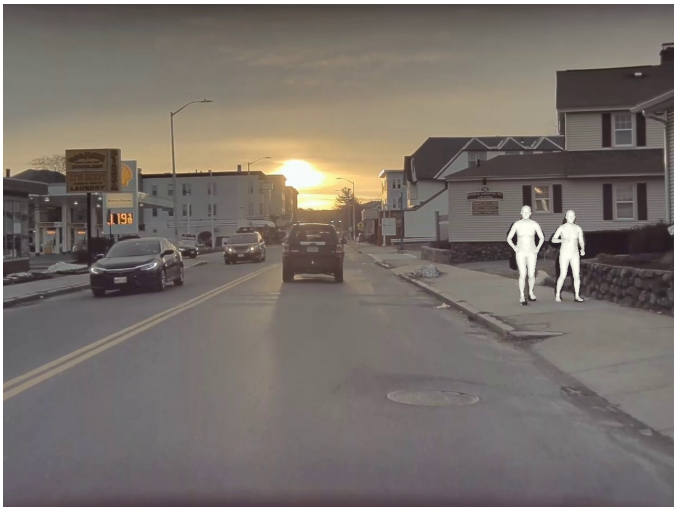

Fig. 12. Pedestrian Pose

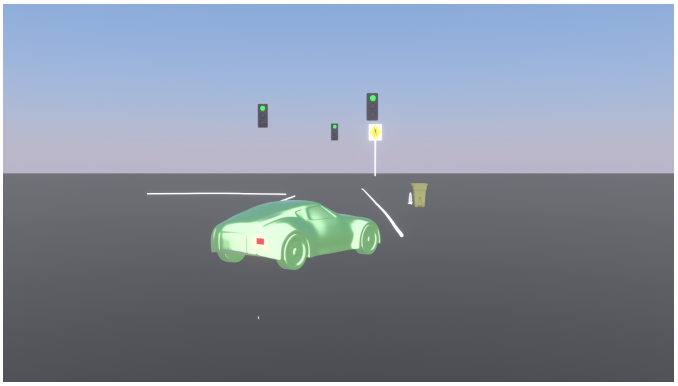

Fig. 13. Turning Signal

bounding boxes of the taillights are obtained, we perform color space conversion from BGR to YCrCb. Subsequently, we apply erosion using a 5x5 kernel to enhance the clarity of the image. Employing adaptive thresholding on the transformed image, we retain only those pixels whose luminance surpasses a specified threshold value. We then assess the pixel count in the resultant image, with a threshold of 220 serving as a criterion. If the pixel count exceeds this threshold, we classify the taillight as ”on”.

For each frame, we retrieve the bounding boxes of all cars and identify centroid points for the two detected taillights within each car bounding box, establishing them as pairs. Employing a series of conditional checks, we determine the signaling behavior of the vehicle. 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.

This methodology not only facilitates precise taillight detection but also enables accurate inference of vehicle signaling behavior, enhancing the overall effectiveness of our system in understanding and interpreting traffic dynamics. However, the model seems to not be generalisable between night and day scenes since the color of the taillights in night scenes is not washed out by the sunlight

I. Classification between Still & Moving Cars

The methodology of the study combines optical flow analysis to distinguish between moving and stationary autos in a scene. For this objective, traditional techniques such as putting a threshold on the Sampson distance were insufficient. As a result, a completely new approach was developed: using Gaussian masking along the flow field’s horizontal axis. By taking use of the spatial properties of flow vectors, this novel method reveals that flow subtraction is generally more pronounced along the borders and less pronounced in the center. Interestingly, the variance of the Sampson distance or flow magnitude directly affects the variance of this Gaussian distribution. This improved process makes it possible to identify parked cars by using the unique flow patterns that are present in the scene, which improves the analysis’s accuracy and dependability.

III. Conclusion

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

References

[1] Jing Lin, Ailing Zeng, Haoqian Wang, Lei Zhang, and Yu Li, ”One-Stage 3D Whole-Body Mesh Recovery with Component Aware Transformer,” arXiv:2303.16160 [cs.CV], 2023.

[2] Zachary Teed and Jia Deng, ”RAFT: Recurrent All-Pairs Field Transforms for Optical Flow,” in *European Conference on Computer Vision (ECCV)*, 2020, pp. 296-313.

[3] Arsalan Mousavian, Dragomir Anguelov, John Flynn, and Jana Kosecka, ”3D Bounding Box Estimation Using Deep Learning and Geometry,” arXiv:1612.00496 [cs.CV], 2017.

[4] Xingyi Zhou, Rohit Girdhar, Armand Joulin, Philipp Krähenbühl, and Ishan Misra, ”Detecting Twenty-thousand Classes using Image-level Supervision,” arXiv:2201.02605 [cs.CV], 2022.

[5] Bingxin Ke, Anton Obukhov, Shengyu Huang, Nando Metzger, Rodrigo Caye Daudt, and Konrad Schindler, ”Repurposing Diffusion-Based Image Generators for Monocular Depth Estimation,” arXiv:2312.02145 [cs.CV], 2024.

[6] Debugger Cafe, ”Lane Detection using Mask R-CNN,” https://debuggercafe.com/lane-detection-using-mask-rcnn/.

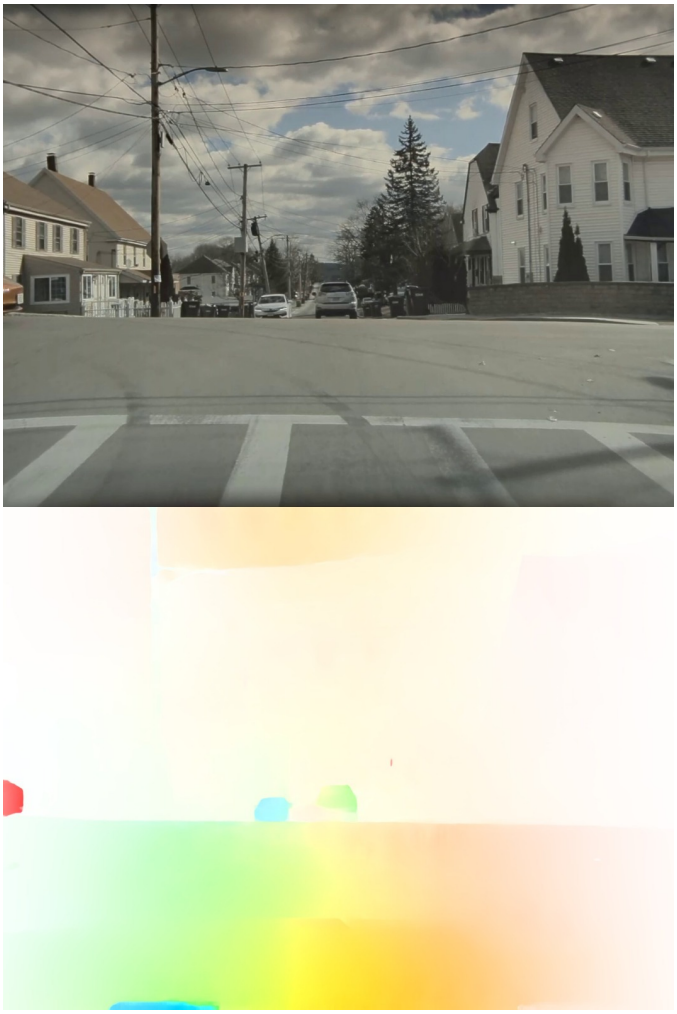

Fig. 14. Optical Flow

[7] Sovit Ranjan Rath, "Traffic Light Detection Using YOLOv3," GitHub repository, 2024. [Online]. Available: https://github.com/sovit-123/Traffic-Light-Detection-Using-YOLOv3.