RBE/CS549: P3: Einstein Vision

Shrishailya Chavan
WPI Robotics Engineering
Worcester Polytechnic Institute
schavan@wpi.edu

Using 1 late day

Sreejani Chatterjee
WPI Robotics Engineering
Worcester Polytechnic Institute
schatterjee@wpi.edu

Using 1 late day

***Abstract*—In this project, we have built a visualization inspired and bettered version of Tesla's dashboard. Here we are provided with a set of videos recorded from cameras of a 2023 Tesla Model S. As a result we are required to output a rendered video of our visualizations or atleast we need to show the view in front of the car and of our car, and if possible we can also show everything around the car. Here we have used both deep learning based and classical based approached to create a rendered video. We have used Blender software to create the rendered images of the data provided from the Tesla car. Additionally, Project is divided in three Checkpoints where in each checkpoint we are supposed to work on different perspectives.**

***Index Terms*—Blender, Rendered Video, visualization**

A. Dataset:

The data given for this project contains the following:

1) Assets (Blender models in the Assets folder) for various things like Cars: Sedan, SUV, Pickup truck, Bicycle, motorcycle, Truck, Traffic signal, Stop Sign, Traffic Cone, Traffic Pole, Speed Sign and Pedestrian. We also include texture images for stop sign and a blank speed sign (add your own speed as text here.

2) Videos (Undistorted and Raw in the Sequences folder) for 13 sequences under various conditions with what scenarios are encountered in the respective markdown files in each folder.

3) Calibration Videos (in the Calib folder) used to calibrate the cameras.

Fig. 1. Tesla Model S used to capture data.

When all the steps mentioned above were followed then we began our Three objectives which were performed in Blender using python scripts and dataset given.

B. Checkpoint 1: Basic Features

In this checkpoint, we implemented basic features which are absolutely essential for a self-driving car. The features we implemented are:

1) **Lanes**: Here we identified and displayed the different kinds of lanes on the road. Each lane has significance and are essential to identify such as green lanes are drivable lanes.

2) **Vehicles**: Here, we identified different kinds of vehicles such as Cars, Trucks, Jeeps, Bicycles, Motorcycles and SUV's.

3) **Pedestrians**: Here, we identified and located the pedestrians that were walking or standing.

4) **Traffic lights**: We identified and located the traffic lights at the various positions we came across them.

5)**Road signs:** The Road signs such as STOP and Speed limit signs were identified and located at certain places.

C. Object Detection

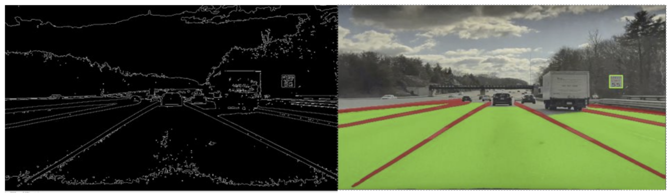

Fig. 2. Speed Limit Recognition

The following steps were done for object detection on the given dataset:-

- 1) We used Yolov5l pretrained model for detection of main objects identified on the road, Cars, trucks, Traffic Lights and Pedestrians.
- 2) Here, Traffic Sign Detection was done using Pytorch and Pretrained Faster RCNN model with Resnet50 backbone. The dataset used in this model was Ger=man Trafic Sign Dataset so we were not able to detect Speed Limit from this model
- 3) For speed limit we did image segmentation where a polygon for a Traffic signal is first segmented and then we used pytesseract OCR library from python detect the

text and Speed limit. Fig. 2 is a demonstration of the process.

- 4) We estimated the Depth from Camera center (camaera used in the car's dashboard) and identified objects using intel Midas pretrained models. Midpas Pretrained model.
- 5) We saved the bounding boxes details and the confidence scores of all the objects detected along with th depth estimation in a CSV file.
- 6) In Blender we wrote a script that will identify the object class name from the csv file and then estimate a scaled position of that object relative to the driving car as stated in the depth estimation. Then we place the .blend object in that position in the scene
- 7) We used YolopV2 model to identify the lanes, and drivable regions which we segmented with Red and Green Masks, then we idemtified the red and green masks in the frames in Blender scripting to place the lanes in desired positions

D. Checkpoint 2: Advanced Features

In this checkpoint, we built on top of the previous checkpoint by enhancing and adding more features. Here, we add more granularity to the vision system which can aid in algorithmic decisions in navigation modules.

Vehicles: Here, we classified (identified different vehicles) and subclassified them (identify different kinds of a type of vehicle). We displayed these detections as the respective 3D model in our renders. More particularly, we rendered Cars: like *Sedan, SUV, hatchback*, **Trucks** like Trucks and Pickups, Bicyle, Motorcycle, TRaffic Lights, Road Signs and Objects like Traffic Cones, Fire Hydrants. We were able to identify all these items using previous YoloV5 model and also placed them in the relavant position in Blender using the logic explained in -C

E. Checkpoint 3: Bells and Whistles

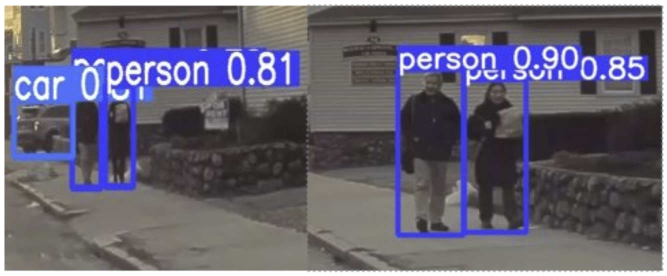

Fig. 3. Identifying Pose of a Person

In this checkpoint, we try to add further cognitive abilities for better decision making in our planning stage.

1) Break lights and indicators of the other vehicles: Identified and displayed the vehicle break lights and indicator signals. This helps the navigation module in making better lane changing decisions.

2)Pedestrian pose: We identified pedestrians and their pose for each frame instead of just classifying and displaying them.

3)Parked and Moving Vehicles: Distinguished between parked and moving vehicles and displayed them accordingly.

F. Methodology for Checkpoint 3:

We were not able to paint the breaklights on cars body in blender. Neither were we able to convincingly detect parked vehicle. However we used the following logic to figure out the vehicle and pedestrian pose. We looked into corresponding frames for vehicles position, and if the vehicles is beyond the drivable region and also the relative distance between the camera center and the center of the car's or pedestrian decreases, we place them posing facing the vehicle in Blender. Fig. 3 demonstrates two frames where the relative distance from the detected persons are decreasing.

G. Results

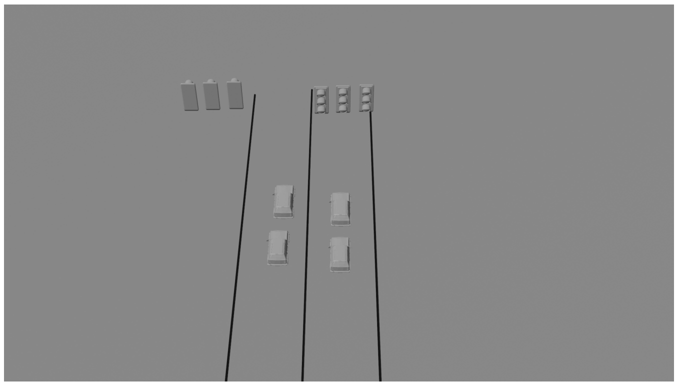

Fig. 4. Scene 11 Rendered Image

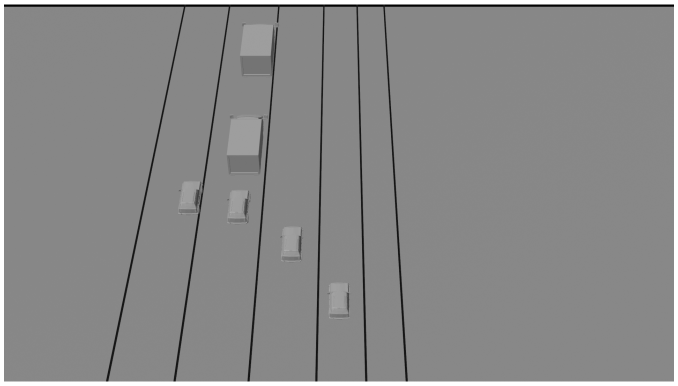

Fig. 5. Scene 13 Rendered Image

Above are the results that we got after Rendering the frames from the videos.

After finding the Rendered images of all the scenes from 1 to 13 we did Rendered Visualization of those images. This we achieved using the dataset provided.

I. Conclusion

In the project we performed Visualization of Tesla dashboard where we were provided the Dataset from Tesla Model

S. We used various object detection methods and Pretrained models to detect objects from the data. We also used image segmentation for certain detection tasks due to lack of pre-trained models. After detecting objects we rendered those objects in Blender and performed final visualization of the objects. For this project due to interest of time we had to rely on existing pre-trained models, however for future use we need to train our own model from larger datasets.

References

[1] https://github.com/matlab-deep-learning/Object-Detection-Using-Pretrained-YOLO-v2/blob/main/README.md
[2] https://github.com/ultralytics/yolov5
[3] https://krbnite.github.io/Intel-at-the-Edge-Leveraging-Pre-Trained-Models/
[4] https://debuggercafe.com/traffic-sign-detection-using-pytorch-and-pretrained-faster-rcnn-model/