RBE 549: Project 3
P3: EinsteinVision

Deepak Harshal Nagle
Robotics Engineering
Worcester Polytechnic Institute
Worcester, Massachusetts 01609
Email: dnagle@wpi.edu
Telephone: (774) 519-8335
Using 2 + 1 = 3 late days

Irakli Grigolia
Computer Science
Worcester Polytechnic Institute
Worcester, Massachusetts 01609
Email: igrigolia@wpi.edu
Telephone: (508) 373-3402
Using 2 + 1 = 3 late days

***Abstract*—The importance of beautiful visualizations in any product that involves human interaction cannot be overstated. In the context of autonomous machines, good visualizations not only make the system easier to use but also provide an intuitive understanding of how the machine is thinking. This project aims to build an improved version of Tesla's latest visualization system for autonomous driving using a set of videos recorded from a 2023 Tesla Model S. The project is open-ended, allowing for creativity in approaching the problem and utilizing any approach, whether classical or deep learning-based. The rendered video of the visualization should show the view in front of the car and the car itself, along with any additional information that can be displayed around the car. The success of this project will be measured by its creativity, effectiveness, and aesthetics.**

INTRODUCTION

In recent years, the development of autonomous machines has been rapidly advancing, and one key component in this development is the creation of intuitive and effective visualization systems. Visualizations not only help with debugging software issues but are also an essential aspect of Human Robot Interaction (HRI). The goal of a good visualization system is to take in sensory information and provide intuitive insights into how the robot is thinking, building trust between the user and the machine.

In this project, we aim to build an improved version of Tesla's latest visualization system for autonomous driving. The visualization will be based on a set of videos recorded from a 2023 Tesla Model S, and the output will be a rendered video showing the view in front of the car, the car itself, and any additional information that can be displayed around the car. The project is open-ended, allowing for creativity in approaching the problem and utilizing any approach, whether classical or deep learning-based.

The project is divided into two checkpoints, with the first one focused on implementing basic features, including identifying different kinds of lanes, representing cars and pedestrians, indicating traffic signals and road signs, and applying appropriate textures. The second checkpoint builds on the previous one by enhancing and adding more features, including classifying and subclassifying vehicles, indicating arrows on traffic lights, indicating road signs on the ground, and detecting objects in the scene.

The project will be graded based on creativity, effectiveness, and aesthetics, encouraging students to approach the problem with a unique perspective and come up with innovative solutions. The importance of good visualizations in autonomous machines cannot be overstated, and this project will provide students with valuable experience in building effective visualization systems for autonomous machines.

DATA

Blender is a professional 3D creation software used in various industries. In this project, we will use Blender to render visualizations of self-driving car data. The data provided includes 13 sequences of videos captured from cameras mounted on a Tesla Model S, calibration videos, and Blender assets such as models of different cars, pedestrians, traffic signals, and road signs.

CHECKPOINTS 1, 2, 3:

Checkpoint 1 requires the implementation of basic features essential for a self-driving car. The required features are identifying and showing different types of lanes, identifying and locating pedestrians, indicating the traffic signals and their color, identifying signboards such as stop signs and speed limit signs, and identifying and representing all cars. All these features need to be visualized in a rendered video of the Tesla Model S that is provided in the data package. The implementation involved both deep learning-based as well as classical approaches, and the visualizations rendered on Blender.

Tasks that we could perform successfully in checkpoint 1:

1. Rendered the lanes
2. Rendered all the vehicles (without classifying them, as asked in the checkpoint)
3. Rendered the traffic lights with colors
4. Rendered the Road signs (stop signs and speed limits)
5. Identified, located and Rendered the pedestrians

What we could not perform for checkpoint 1: Nothing!

Checkpoint 2: Implementation of advanced features for the visualization system, building upon the basic features from Checkpoint 1. This may include additional visualizations such as lane markings, road boundaries, obstacles, and other relevant objects in the scene. The goal is to create a more detailed and informative visualization that provides intuitive insights into how the autonomous machine perceives its environment.

Tasks that we could perform successfully in checkpoint 2:
1. Classified different kinds of vehicles such as cars, trucks, bicycles, motorcycles

2. Traffic lights: Could identify and render colors. Implemented a network for traffic lights with arrows but could not get good results.

3. Road signs: Were successful in detecting, rendering the stop signs accurately

4. Objects: Could detect fire hydrants and poles

Checkpoint 3: Tasks that we could perform successfully in checkpoint 3:

1. Implemented classical methods of color threshholding for Break lights and indicators of the other vehicles. However, could not get good results

2. Pedestrian pose: Identify pedestrian pose accurately with keypoint detection. However, rendering it on blender in time was really tough task.

3. We could only propose methods to detect Parked and Moving Vehicles, but due to time constrains, could never implement them.

Lanes:

In the first attempt at finding lanes for checkpoint one, we utilized the "YOLO Pv2" network for lane detection. We started by taking the lowest point on the lane on the image and projecting it into 3-D using Blender. We then drew lines parallel to the Blender Y-axis, passing through the projected points, to represent the lanes.

However, we soon realized that this approach would not work for a multiple-framed video, as the camera on the car was moving along with the car relative to the lanes. Additionally, passing the lane through a single point resulted in higher error as compared to passing it through multiple points. Furthermore, the "YOLO Pv2" network classified all types of lanes into a single category, which was not accurate for our purpose. As a result, we decided to switch to a different network called LaneNet.

We found that LaneNet[1], although promising, had lower accuracy as it was trained on only 25 epochs. Therefore, we decided to resort to a classical method for lane design. We estimated the location of the camera and took a roughly estimated distance between the two parallel lanes to render a good lane design. This approach proved to be more accurate and reliable for our project. Overall, we learned that choosing the right lane detection method for our specific project requirements was crucial in achieving the desired results.

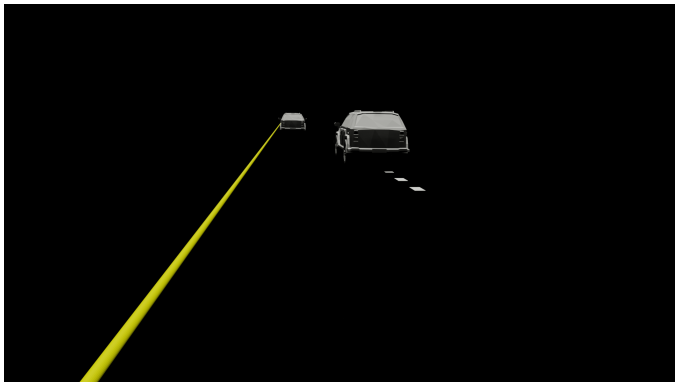

Fig. 1. Lanes detected

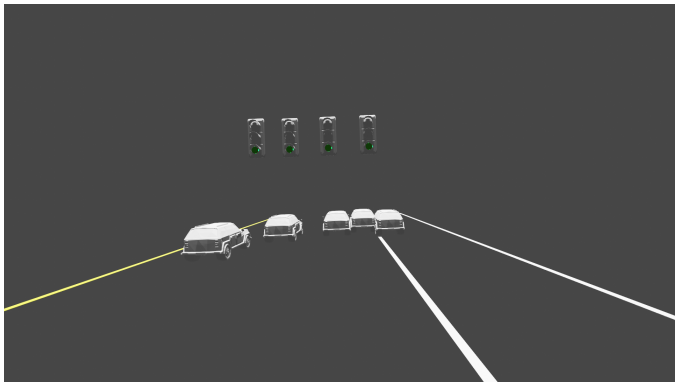

Fig. 2. Lanes detected

Monocular Depth Estimation:

We experimented with different methods to estimate depths for the individual images. We initially used a basic depth estimation model called MIDAS, which provided moderate accuracy. However, after consulting with our professor, we decided to explore transformer-based depth estimation networks[3], which are known for retaining higher frequency information despite being computationally heavy. This approach yielded high accuracy results, especially for closer distances.

However, we encountered challenges with accuracy at higher depths, where objects with were far were appearing to have almost similar depth values.. To address this issue, we implemented a modification by fitting an exponential function into the depth map. Thus we modified the depth map by fitting an exponential function into it, so that the objects at the farther distances, even though having similar intensities in depth map, will have significant depth difference. This adjustment helped to differentiate objects at farther distances and improved the accuracy of our depth estimation, as observed in the images.

Overall, our team utilized a combination of different depth estimation methods, starting with MIDAS and then transitioning to a transformer-based network with further modifications, to achieve more accurate depth estimation results for our project. This iterative approach allowed us to refine our depth estimation process and improve the accuracy of our results.

Fig. 3. Depth Estimation Map

Fig. 4. Depth Estimation rendered

Absolute scale estimation using road markers:

Estimating absolute scales was a challenging task due to the lack of information about the distance of objects from the camera. Although we knew the sizes of real-life cars, we didn't have accurate knowledge of their distances relative to the camera. We quickly realized that scales would differ for objects parallel to the image plane compared to objects at different depths. Therefore, we had to experiment with trial and error to improve the scale estimation.

In fact, this is what inspired us to come up with exponentially increasing depth and finally we obtained the scales that we accurate enough for a good rendering. By fitting an exponential function to the depth map, we were able to create depth differences between objects at farther distances. This approach resulted in more accurate scale estimation, which in turn improved the rendering of the scene. It was a creative solution that enabled us to obtain satisfactory results in estimating absolute scales for our scene.

Vehicle Classification, Object detection, and Classification:

We initially used an MS COCO-based YOLOv3 model[4] for object detection, which performed fairly well in detecting vehicles but struggled with other objects like traffic lights, road signs, and bicycles. It also had some confusion between trucks and cars.

Much later (halfway through the final deadline of checkpoint-2,3) we found a more recent MS COCO-based YOLOv7 model[5], which was more accurate than the previous one. However, using bounding boxes for object localization may not be accurate, as it could result in averaging the depths, including from parts of the bounding box where the object is not technically present. We wanted to perform instance segmentation and find the centroid of the object for better localization, but couldn't find a suitable network for this task due to time constraints.

As a result, we tried implementing a 3-D bounding box based network for better localization, but it performed poorly in detecting different instances of cars as well as other objects. Further, if we were to use the 3-D bounding box based model, just for pose, we will not be able to tell which car it corresponds to, based on the cars detected in 2-D object detection. We faced challenges in finding a network that could perform instance or panoptic segmentation and distinguish between different instances of the same object.

Despite our extensive efforts, we were unable to find a reliable model that could accurately detect and classify cars into different types such as SUVs, sedans, and hatches. However, we were able to use YOLOv3 and YOLOv7 for detecting and rendering different types of vehicles including cars, trucks, and bicycles, as mentioned earlier. As an **additional approach**, we considered using a 3-D bounding box-based object detection network[8] to estimate the **aspect ratio** of cars and compare it with the ratios of actual types of cars. This could potentially help with classifying the cars into different types. We further attempted to find the **orientation** of the vehicles for better rendering, even though it was **NOT** a requirement of the project. However, after spending significant time on finding and debugging such a network to obtain "fairly accurate" **poses**, we realized that it performed poorly in detecting different instances of cars and other objects. Moreover, using the 3-D bounding box-based model for pose estimation alone would not allow us to determine which car it corresponds to, based on the 2-D object detection results. This posed a challenge in accurately classifying different types of cars in our project.

Despite these challenges, we continued to explore different approaches and models and **successfully performed vehicle classification** (including Cars, Trucks, Bicycles, Motorcycle) in our project.

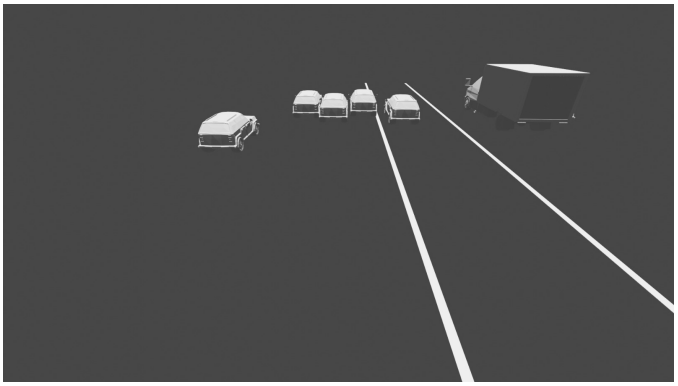

Fig. 5. Cars, Truck detected separately

Road Signs:

In the beginning, we experimented with the MS COCO-based YOLOv3 model[4], but it yielded unsatisfactory results. After much effort, we switched to the MS COCO-based YOLOv7 model[5], which showed slight improvements. However, we faced challenges in accurately rendering different assets for Road signs, especially for rendering different assets of different speed limits, as well as in pasting the .png image on top of the asset (this is because technically the road sign includes the pole as well, thus the image would get pasted on the flat surface as well as the pole). Finally, we were able to find and debug a YOLOv5 model[6] that was specifically trained on different street sign data and provided satisfactory results.

Fig. 6. Stop Sign (Original Image)

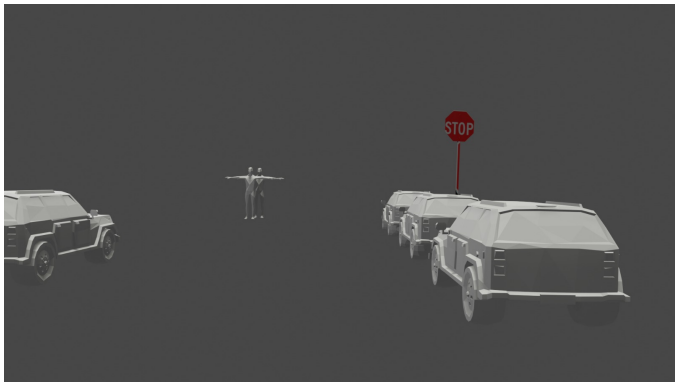

Fig. 7. Stop Sign (Rendered Image)

Traffic Lights:

Initially we used MS COCO-based YOLOv3 model[4], however, it would give out very poor results. Much later, we could move on to MS COCO-based YOLOv7 model[5], to get slightly better results. Finally we could find and debug a YOLOv3 model[6] which was specifically trained on traffic light data such as different traffic lights with arrows and gave satisfactory results. Rendering different assets for different lights turned on, is also a tough task.

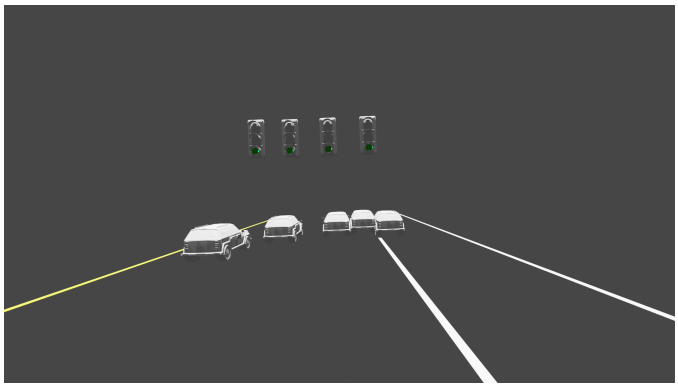

Fig. 8. Traffic lights

Other objects:

Initially we used MS COCO-based YOLOv3 model[4], as we could move on to MS COCO-based YOLOv7[5], we were able to detect fire hydrants and poles.

Break lights and indicators of the other vehicles:

After detecting the cars, we used classical approach for this: We took the 2-D bounding box in which the car is detected and found out if the specific shades of red/orange light are present. However, our approach was giving us, numerous false positives.

Pedestrians and their Pose

: We detected the pedestrians very well using YOLOv7 model[5] as seen in the image below.

Fig. 9. Pedestrian Pose (Original Image)

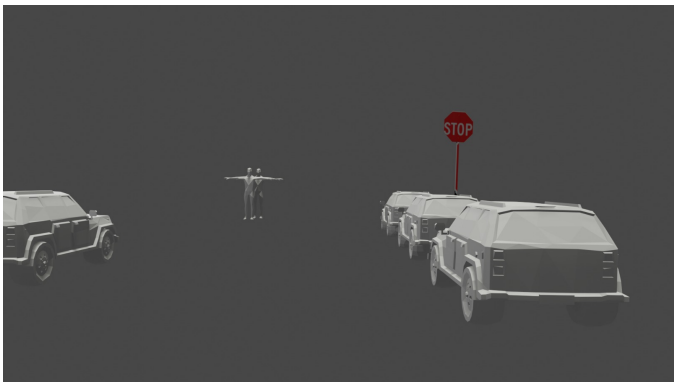

Fig. 10. Pedestrian Pose (Rendered Image)

We were able to get really good pedestrian pose estimation as well, using the network implemented by MS COCO 2016 challenge winners[9]. However, we could not render it on blender.

Conclusion

In conclusion, this project aimed to build an improved version of Tesla's latest visualization system for autonomous driving using videos recorded from a 2023 Tesla Model S. The project was divided into three checkpoints, with the first checkpoint focusing on implementing basic features, the second checkpoint enhancing and adding more features, and the third checkpoint proposing methods for detecting parked and moving vehicles, among other tasks.

Throughout the project, a combination of classical and deep learning-based approaches were utilized to implement various features such as lane detection, vehicle classification, traffic signal detection, road sign recognition, and pedestrian pose estimation. Overall, the project was successful in implementing most of the required features, including rendering lanes, vehicles, traffic lights, road signs, and pedestrians in Blender. However, some tasks, such as classifying types of cars accurately, and rendering pedestrian poses in Blender, proved to be challenging and did not yield satisfactory results.

Problems Faced

Accuracy of Classification: Despite utilizing deep learning-based approaches for vehicle classification, achieving accurate results proved to be challenging. This could be due to the complexity and variability of vehicle types, lighting conditions, and occlusions in the videos, which impacted the accuracy of the classification models.

Time Constraints: The project faced time constraints, especially in the third checkpoint. This resulted in incomplete implementation and limited evaluation of these proposed methods.

Blender Rendering: Rendering visualizations in Blender proved to be challenging, particularly in terms of accurately rendering pedestrian poses and implementing complex features such as traffic lights with arrows. This required extensive experimentation and tweaking of parameters, which consumed a significant amount of time and effort.

Lack of Data: The availability of limited data/models, especially for specific tasks such 6-D pose estimation for vehicles, detection of traffic cylidners, garbage cans, poles, etc. posed a challenge. This led to limitations in the performance of these tasks.

Complexity of the Problem: Building an improved visualization system for autonomous driving is a complex task that requires a deep understanding of computer vision, deep learning, and 3D rendering. The team encountered various technical challenges, such as selecting appropriate techniques, integrating different components, and fine-tuning parameters, which required a significant amount of effort and expertise.

Despite these challenges, the project was successful in implementing many of the required features and provided valuable insights and experience in building effective visualization systems for autonomous machines. The project highlighted the importance of creativity, effectiveness, and aesthetics in designing visualizations for autonomous driving and the need for further research and development in this area to overcome existing challenges and improve the usability and intuitiveness of such systems. Future work could focus on refining the accuracy of classification models, exploring alternative rendering techniques, and utilizing larger and more diverse datasets to train and evaluate the models. Additionally, addressing the limitations of time constraints and data availability could lead to more comprehensive and robust visualization systems for autonomous driving. Overall, this project has provided valuable insights into the field of autonomous machine visualizations and the challenges and opportunities associated with building effective and intuitive visualization systems for human-robot interaction. So, the project's success can be measured in terms of its contributions to the field, creativity in approaching the problem, effectiveness in implementing features, and aesthetics of the rendered visualizations.

REFERENCES

[1] Lane Detection: https://github.com/CAIC-AD/YOLOPv2

[2] Lane Detection: https://github.com/IrohXu/lanenet-lane-detection-pytorch
[3] Monocular Depth Estimation: https://github.com/isl-org/MiDaS
[4] Object Detection: Cars, Trucks, Traffic Lights, Road Signs: https://github.com/xiaogangLi/tensorflow-MobilenetV1-SSD
[5] Object Detection: Cars, Trucks, Traffic Lights, Road Signs: https://github.com/WongKinYiu/yolov7
[6] Object Detection: Traffic Lights: https://github.com/sovit-123/Traffic-Light-Detection-Using-YOLOv3
[7] Object Detection: Road Signs: https://github.com/Anant-mishra1729/Road-sign-detection
[8] YOLO 3-D bounding boxes: https://github.com/ruhyadi/YOLO3D
[9] Pedestrian keypoint detection: https://github.com/ZheC/Realtime_Multi-Person_Pose_Estimation