# RBE 549 Project 3: Einstein Vision

Amrit Krishna Dayanand, Venkata Sai Krishna Bodda
MS Robotics Engineering
WPI
Email: adayanand@wpi.edu, vbodda@wpi.edu
using 1 late day for phase 3

## I. Introduction

Perceiving the environment around a vehicle is crucial for autonomous navigation planning. Autonomous vehicles use various sensors like a monocular camera, inertial measurement unit (IMU), light detection and ranging (LIDAR), or radar to achieve this. Tesla recently pivoted its strategy to fully rely on an array of monocular cameras and deep-learning-based computer vision algorithms for scene estimation. In this project, we use the camera feed from the front camera of a Tesla Model S and estimate the scene around the car in 3D, using Blender. This is broken down into three subtasks. First, we want to estimate the spatiotemporal states of objects, then visually distinguish different classes of objects and finally display information in an aesthetically pleasing way.

In particular, we are interested in the following objects, and states.

1) Vehicles
   a) classified as sedan, SUV, truck, bicycle, etc.
   b) oriented in space
   c) with brake lights and turn signals
   d) distinguished as parked or moving, where moving vehicles have moving direction displayed
2) Road signs
   a) stop signs
   b) traffic lights with colors and arrows
   c) lanes of different styles (dashed, solid), with color
   d) arrows on the ground
   e) speed signs
3) Pedestrian pose
4) Road objects like trash cans, cones, poles, etc.
5) Optional
   a) Speed bumps
   b) Collision prediction with road objects, including cars, pedestrians

## II. Proposed approach

Our approach uses classical and deep learning approaches to infer the states of objects in the scene. First, we estimate the locations and classes of objects in space by combining information from object detection and relative depth estimation, using a deep-learning approach. We find lower-level states like the color of lights by cropping regions of interest and applying an HSV segmentation and thresholding approach. To estimate human poses, we use Meta PIFuHD to go from crops of pedestrians to the 3D pose. We identified lanes using CLRerNet and using control points with bezier curves in Blender.

### A. Pixel to world coordinates

To convert coordinates from image coordinates to world coordinates, we use the projection equation and relative depth. This is done by emitting a ray from each pixel and picking a value along the ray scaled by the relative depth value, as shown below

$$\tilde{x}_{proj} = K\tilde{u}$$
$$\tilde{x}_{cam} = Z\tilde{x}_{proj}$$
$$\tilde{x}_{world} = t_c^w + R_c^w\tilde{x}_{cam}$$

where $\tilde{x}_{proj}$ is the ray projected from the image, $Z$ is the depth, $\tilde{x}_{cam}$ is the 3D point in the camera frame, $\tilde{x}_{world}$ is the 3D point in the world frame, and $t_c^w$ and $R_c^w$ are the pose of the camera in the world frame.

MMDetection is an open source object detection toolbox based on PyTorch. It is a part of the OpenMMLab project.

Detect object positions and classes - MMDetection

Find relative inverse depth - MiDaS (DPT large architecture)

Crop people, traffic lights for further processing

Traffic light color detected - HSV segmentation and thresholding

Convert 2D person image into mesh - Meta PIFuHD Lanes painted by finding control points - CLRNet and Blender bezier curves

The detection of whether brake lights were applied to cars was conducted using conventional methodologies. Each frame of the vehicle underwent a comparative analysis with its preceding frame to ascertain the presence of increased redness, indicative of brake activation.

The determination of frame redness involved applying a color threshold utilizing the adaptive threshold method provided by the OpenCV library. This process facilitated the identification and quantification of red pixels within the frame.

Moreover, the Connected Components with Stats method from the OpenCV library was employed to obtain bounding boxes delineating the regions corresponding to the vehicle's tail lights. Subsequently, the redness analysis was specifically

conducted within these bounded regions to infer the activation status of the brakes.

## III. Results - for each case in each phase

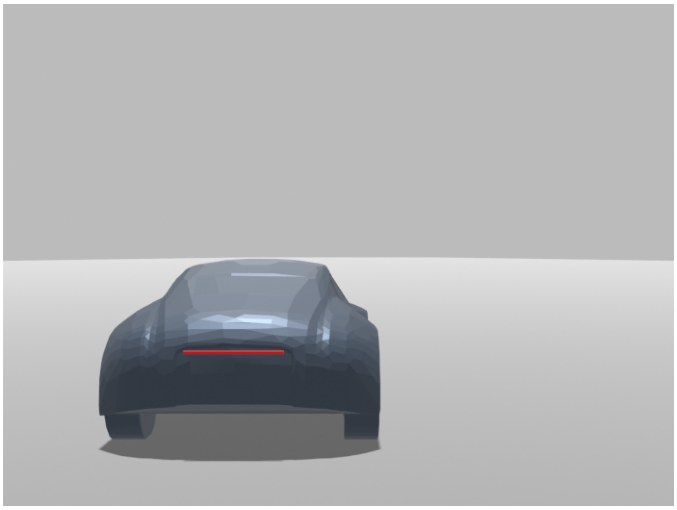


Fig. 1. Car braking action in blender

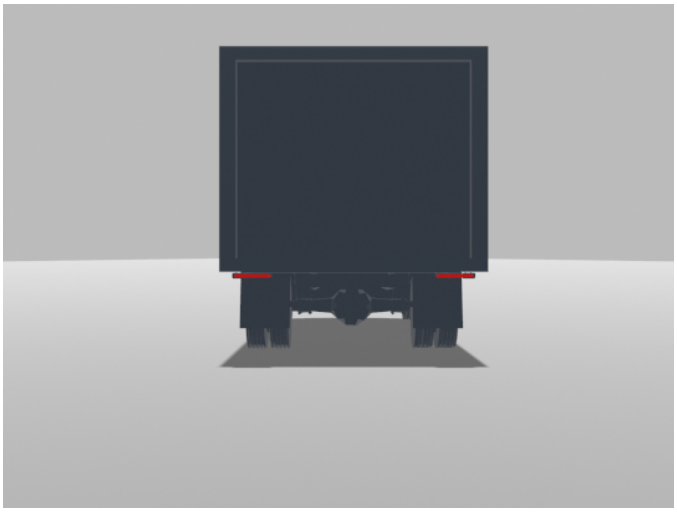


Fig. 2. Truck braking action in blender

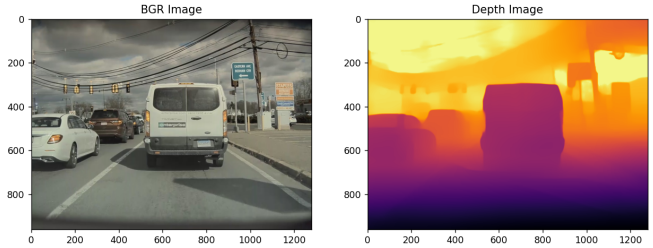



Fig. 3. depth detection at day

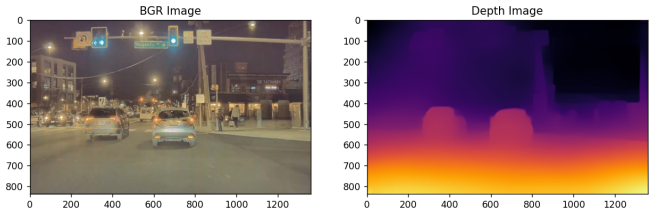



Fig. 4. depth detection at night

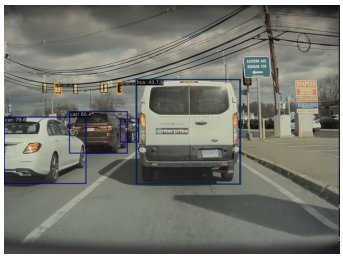


Fig. 5. Bounding box at day

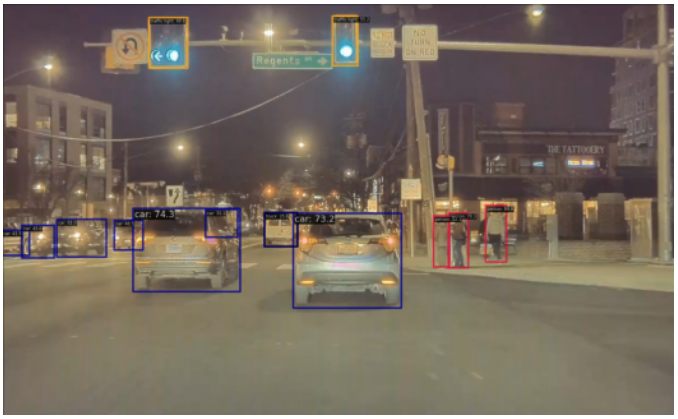


Fig. 6. bounding box at night

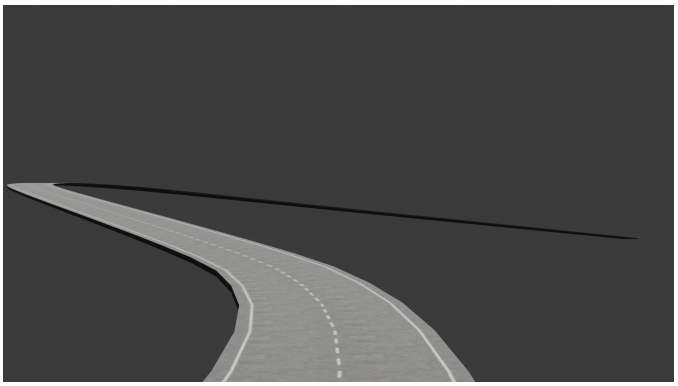


Fig. 7. Lane generation using the blender array and Bezier curve modifiers

## IV. Challenges and Resolutions

### A. Relative depth estimation

The relative depth estimation from MiDaS provides information on the relative depth between objects, but not the absolute distance between them, or from the camera. To convert from relative to absolute depth, we scaled the depth value using the height of known objects in the environment like stop signs, and tuning. This provided good results when objects were close to the camera, but became poor the further away they were.

This is because at larger distances, the relative depth between objects becomes less apparent as shown in 8. This led to objects at a distance spawning at the same depth, which did not match the ground truth.

Fig. 8. Objects further away from the camera appear to have similar relative depth (same color)

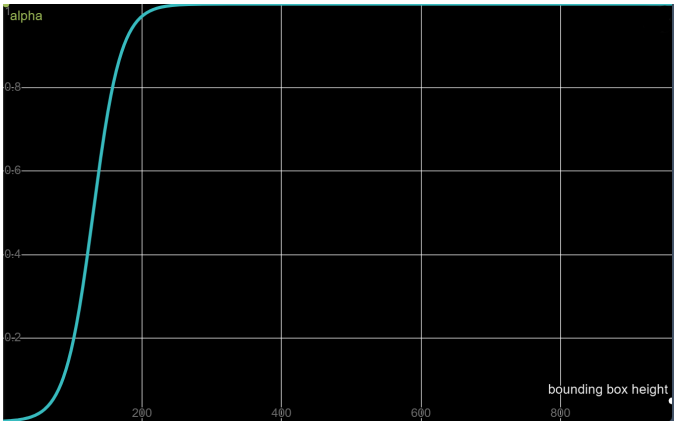
To resolve this issue, we can use information from the bounding box of objects, focal length, and their known size.

$$\frac{h_w}{h_i} = \frac{d_w}{f}$$
$$d_w = kf\frac{h_w}{h_i}$$

where $k$ is a scalar tuning value, and $h_i$ is the height of the bounding box.

We compared the bounding box approach alone, and MiDaS approach alone, and found that a blended approach was better. At close distances, we used the MiDaS approach and at greater distances, we used the bounding box approach. The height of the bounding box is used as a proxy for the distance from the camera. This is achieved by creating a closeness bias, $\alpha$, a sigmoid function:

$$\alpha = \frac{1}{1 + \exp\left(-(k_1(x - x_0))\right)}$$




where $k_1$ is the horizontal scaling factor ($k_1 = 0.05$) and $x_0$ is the horizontal shift ($x_0 = 130$). The horizontal shift represents the height of the bounding box at which the bias becomes 0.5.

The final depth value is thus a function of how close an object is to the camera, by proxy of bounding box height, given by:

$$Z = \alpha f_1(u, v) + (1 - \alpha) f_2(h_i, h_w, f)$$

where $\alpha$ is the closeness bias, $f_1$ is the MiDaS depth, and $f_2$ is the bounding box depth.

The use of the HSV color space for traffic light color detection posed significant challenges, primarily due to the orange casing of the traffic lights. This presented a major obstacle since detecting the traffic light signal color relied on counting the number of pixels within the ranges of red, orange, and green. However, due to the dominance of orange pixels, the color detection algorithm consistently identified orange as the dominant color, regardless of the actual signal color.

To address this issue, a novel approach was devised involving clever color thresholding. Specifically, a crop of the traffic light facing side was obtained, allowing for the extraction of color ranges present within the image. As the majority of pixels in this cropped region corresponded to the orange casing, it facilitated the accurate determination of the range of orange hues. Notably, this particular shade of orange appeared muted in the cropped image, thus distinguishing it from the orange light color emitted by the traffic light itself.

Through this innovative technique, the challenges posed by the orange casing were effectively mitigated, ensuring more accurate and reliable traffic light color detection.

As previously discussed, the detection of brake lights entailed a comparison of redness levels between the current and past bounding box crops. Initially, this method only provided the brake status at the moment when brakes were engaged, but did not sustain detection while brakes were in the process of activation.

To address this limitation, an additional condition was introduced to detect a decline in redness levels. Consequently, the brake status remains classified as "brake applied" until the redness diminishes, thus aligning with the original scenario. This enhancement ensures a more comprehensive and accurate assessment of brake activation throughout its duration.

As previously mentioned, employing PIFuHD provides detailed meshes and poses of humans within the scene. However, inherent inaccuracies in pose estimation can introduce noticeable jittering effects in the video for human subjects. To address this issue, a smoothing technique was implemented, leveraging poses extracted from preceding frames. By averaging the poses across these previous frames, a smoother transition in pose is achieved, offering a more faithful representation of reality in the video footage.

### *B. Observations*

The depth estimation worked well mostly, with the tuned depth scale generalizing well to all scenes. Since depth estimation primarily relied on MiDaS, inaccuracies propagated in all axes during scene estimation. The render worked best in highway scenes where vehicles are facing forward. On the contrary, in scene 8, vehicles turning are rendered facing forward due to the lack of pose tracking.

The MMDetection object detection library was reasonably good at classifying vehicles and objects during day time but struggled in poor lighting conditions. This was observed in scene 12 (night time) and scene 2 (tunnel) where vehicles would sometimes not be recognized. Furthermore, there was ambiguity in classifying trucks, SUVs, pickup trucks and buses. This led to objects flickering between vehicle classes.

An oscillation was observed when vehicles were close to the camera, like in scene 9. This is likely due to the rapidly changing gradient in MiDaS depth at the bottom of the image. This stabilized as the vehicle moved further away.

Traffic light detection works predicts way better during night due to the high contrast between lights. It confuses a bit between red and orange because of camera color science and but the accuracy doesn't take a huge dip.

The efficacy of traffic light detection notably improves during nighttime operations, primarily owing to the heightened contrast between the illuminated lights. Despite encountering occasional challenges, such as confusion between red and orange signals due to nuances in camera color science, the overall accuracy of the detection mechanism remains consistently robust.

## V. Result

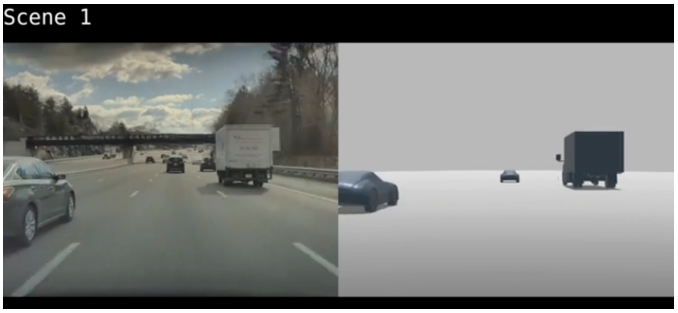



Fig. 9. scene 1

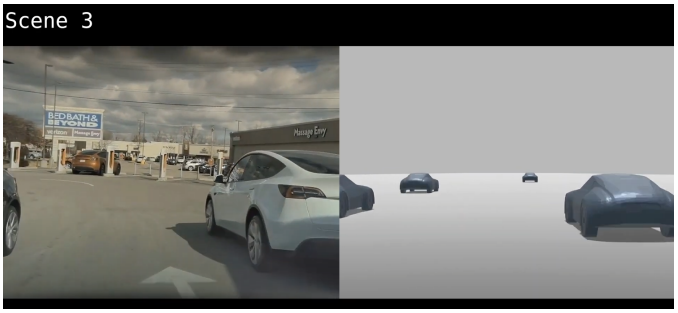



Fig. 10. scene3

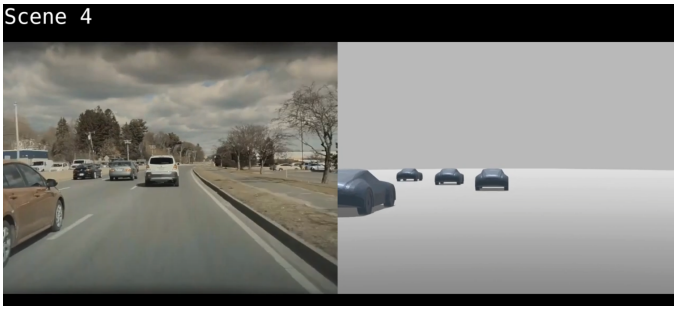



Fig. 11. scene4


Fig. 12. scene6

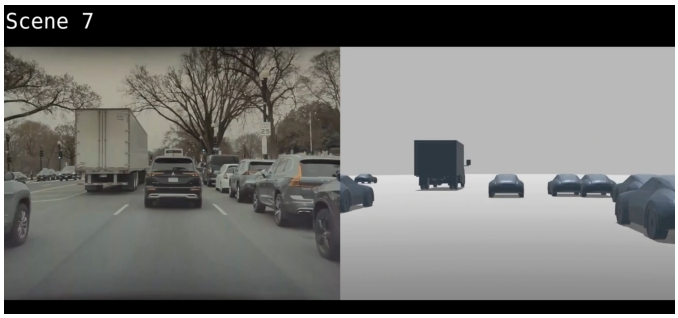



Fig. 13. scene7


Fig. 14. scene9

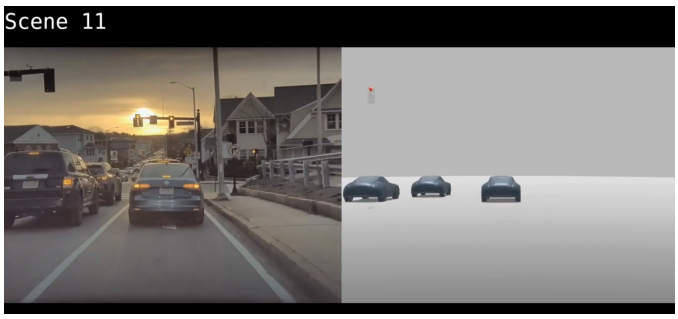



Fig. 15. scene11

## VI. References


1) https://pytorch.org/hub/intelisl/midas_ v2

2) https://github.com/TechToker/Vehicle-rear-lights-analyser/tree/master

3) Mauricio Casares, Akhan Almagambetov, Senem Velipasalar, A Robust Algorithm for the Detection of Vehicle Turn Signals and Brake Lights
4) https://mmdetection.readthedocs.io/
5) https://github.com/hirotomusiker/CLRerNet
6) Hiroto Honda, Yusuke Uchida, CLRerNet: Improving Confidence of Lane Detection with LaneIoU
7) https://shunsukesaito.github.io/PIFuHD/

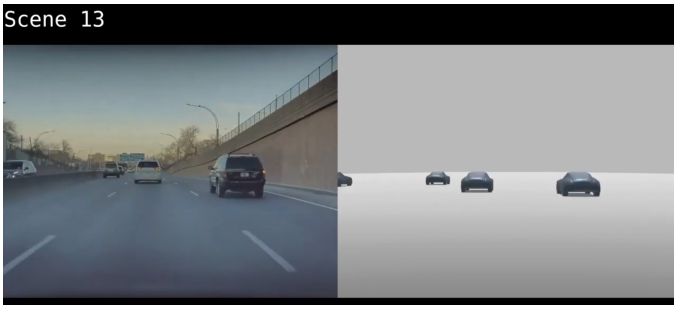



Fig. 16. scene 13