Einstein Vision: A Safer Autonomy

P Pavan Ganesh
Department of Robotics Engineering
Worcester Polytechnic Institute
Worcester, MA, USA
ppabbineedi@wpi.edu
Using 3 Late days

Manideep Duggi
Department of Robotics Engineering
Worcester Polytechnic Institute
Worcester, MA, USA
mduggi@wpi.edu
Using 3 Late days

***Abstract*—This paper presents a comprehensive Tesla-inspired visualization system designed to enhance human-robot interaction through intuitive visual representations of sensory data. Our implementation encompasses lane detection with classification, vehicle and pedestrian identification, traffic light state recognition, and road sign detection. We employed a combination of traditional computer vision techniques and deep learning approaches, including Mask R-CNN for lane segmentation, YOLOv8x for object detection, and MiDaS for depth estimation. The system successfully converts 2D detections into 3D world coordinates and renders them in Blender to create an intuitive visualization similar to Tesla's dashboard. Additionally, we implemented advanced features such as brake light and indicator detection, distinguishing between parked and moving vehicles, speed bump detection, and collision prediction, further enhancing the system's capabilities. The resulting visualization provides an effective interface for understanding how autonomous systems perceive and interpret their environment, thereby building trust between humans and autonomous machines.**

***Index Terms*—autonomous vehicles, computer vision, visualization, human-robot interaction, deep learning**

I. Introduction

Effective visualization systems are crucial for building trust between humans and autonomous machines, making interfaces more intuitive, and providing insights into how robots perceive and process information. Drawing inspiration from Tesla's dashboard visualization evolution, this project aims to create an improved version of such a system using video data from a 2023 Tesla Model S.

This paper details our implementation of a Tesla-inspired visualization system, focusing on Phase 1 basic features, Phase 2 advanced features, and Phase 3 cognitive abilities. Phase 1 includes lane detection and classification, vehicle detection, pedestrian identification, traffic light state recognition, and road sign detection. Phase 2 extends these capabilities with more granular vehicle classification, advanced road marking detection, traffic light arrow classification, pedestrian pose estimation, and additional object detection. Phase 3 adds cognitive abilities such as brake light and indicator detection, distinguishing between parked and moving vehicles, and extra credit features including speed bump detection and collision prediction.

Our approach combines traditional computer vision techniques with state-of-the-art deep learning models to create accurate and robust visualizations. The remainder of this paper is organized as follows: Section II describes our methodology for implementing the basic features, Section III details our advanced feature implementation, Section IV explains our cognitive ability implementation, Section V presents our results and evaluation, and Section VI concludes with a discussion of challenges faced and potential future improvements.

II. Phase 1: Basic Features Implementation

A. Lane Detection and Classification

Our lane detection system evolved through multiple iterations to achieve robust performance across various lighting conditions and road scenarios.

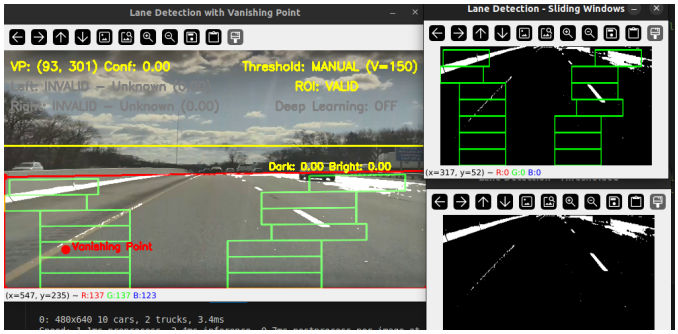

Fig. 1: Lane Detection with Traditional Approach

Initially, we implemented a traditional lane detection pipeline using color thresholding, sliding windows, and vanishing point estimation. This method worked well under optimal lighting conditions, accurately detecting lane markings and classifying them as solid or dashed lines. However, it struggled in challenging scenarios, such as when the sun or bright reflections were present in the frame. These conditions introduced significant noise and false positives, degrading the system's accuracy.

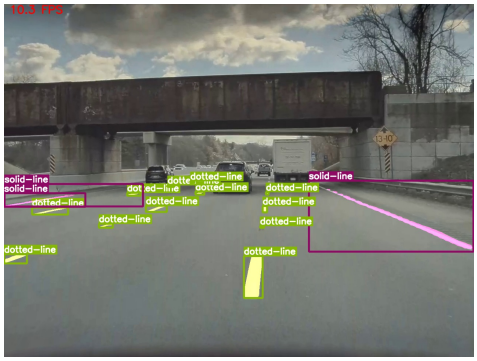

Fig. 2: Lane Detection and Classification with Deep learning

To address these limitations, we transitioned to a more robust approach using Mask R-CNN, a deep learning-based model capable of segmenting lane markings with higher precision. In this approach, after detecting lane masks using Mask R-CNN, the system extracts 10 equidistant points along each lane by fitting a second-order polynomial (quadratic curve) to the detected mask. This polynomial, of the form: provides a smooth approximation of the lane's curvature. If polynomial fitting fails due to insufficient points, the system falls back to contour extraction for robustness. These 10 points are then projected into 3D space using depth estimation, enabling accurate spatial representation of the lanes. This approach ensures a structured and mathematically consistent representation of lane geometry, improving stability in real-world driving scenarios.

$$f(y) = ay^2 + by + c \tag{1}$$

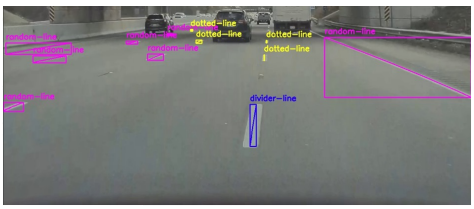

Fig. 3: Second order polynomial fitting for lane detection

B. Vehicle Detection

For vehicle detection, we employed the YOLOv8x model, which offers an excellent balance of accuracy and inference speed. The model was pre-trained on a diverse dataset that includes various vehicle types in different lighting conditions and orientations.

Fig. 4: Vehicle Detection

Our implementation processes each frame to detect vehicles, extracting their bounding box coordinates, confidence scores, and class labels. We applied a confidence threshold of 0.5 to filter out low-confidence detections, reducing false positives while maintaining high recall rates. The detected vehicles are then represented in 3D space using the estimated depth at the bottom center of each bounding box, providing a realistic representation of their position relative to the ego vehicle. This approach allows for accurate visualization of surrounding traffic in the Blender environment.

C. Pedestrian Detection

Similar to vehicle detection, we utilized YOLOv8x for pedestrian identification. The model effectively detects pedestrians across various poses, clothing styles, and lighting conditions.

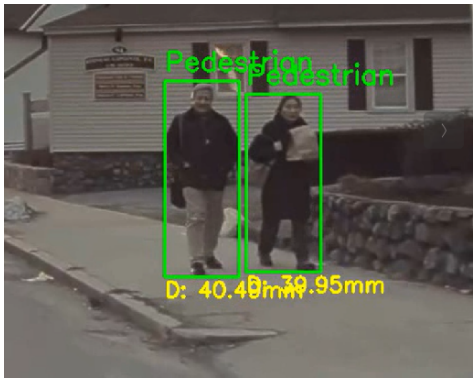

Fig. 5: Pedestrian Detection

Our system processes each frame to identify pedestrians, extracting their bounding box coordinates and confidence scores. We applied the same confidence threshold of 0.5 to ensure reliable detections while minimizing false positives.

The detected pedestrians are positioned in 3D space using depth estimation, allowing for accurate representation of their location relative to the ego vehicle. This spatial awareness is crucial for safety-critical applications and provides valuable context for human operators monitoring the system.

D. Traffic Light Detection

Traffic light detection presents unique challenges due to their relatively small size in the frame and the importance of accurately identifying their state (red, yellow, or green).

We implemented traffic light detection using YOLOv8x, which was trained to recognize not only the presence of traffic lights but also their current state. The model outputs bounding box coordinates, confidence scores, and state classifications for each detected traffic light.

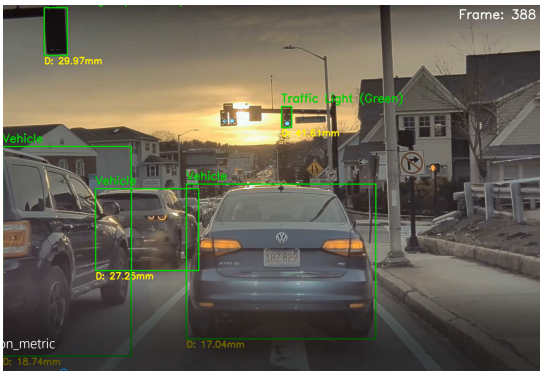

Fig. 6: Traffic Light Detection (Green)

To improve reliability, we incorporated temporal consistency checks, tracking traffic lights across consecutive frames and using a voting mechanism to determine the most likely state when there are conflicting detections. This approach significantly reduces flickering and improves the stability of traffic light state visualization.

E. Road Sign Detection

Road sign detection is essential for understanding traffic rules and regulations. We employed YOLOv8x to detect various road signs, with particular emphasis on stop signs as specified in the project requirements.

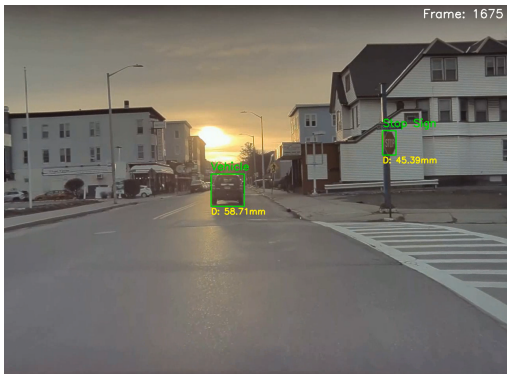

Fig. 7: Stop Sign Detection

The model identifies road signs in each frame, providing bounding box coordinates, confidence scores, and sign type classifications. We applied post-processing to filter out low-confidence detections and resolve conflicts when multiple sign types are detected in close proximity.

The detected signs are positioned in 3D space using depth estimation, allowing for accurate representation in the visualization. For stop signs, we utilized the provided 3D models and textures to create realistic representations in the Blender environment.

III. Phase 2: Advanced Features Implementation

A. Depth Estimation

The depth estimation in our system serves a crucial role in converting 2D detections into 3D world coordinates. We used Intel's DPT-Hybrid-MiDaS model through the Hugging Face pipeline to estimate relative depth from a single RGB frame.

Fig. 8: Depth Estimation

The raw depth output from MiDaS is converted to metric depth using the formula:

$$truedepth = 1.0/(A + (B * midas_output)) \tag{2}$$

Here, the scale factor (3000) and shift (0.1) empirically adjust the relative depth values to approximate real-world distances (in millimeters). The depth map is then combined with the camera intrinsics (focal lengths f_x, f_y and optical center c_x, c_y) to project 2D points into 3D using perspective projection:

$$X = \frac{(x - c_x) \cdot Z}{f_x}, \quad Y = \frac{(y - c_y) \cdot Z}{f_y}, \quad Z = \text{depth}(y, x) \tag{3}$$

This approach assumes a pinhole camera model, where depth (Z) is directly derived from the estimated depth map at each pixel. While efficient, the method's accuracy depends on the depth model's robustness to varying lighting and road textures. The fallback to contour extraction when polynomial fitting fails ensures continuity, but depth errors may propagate if the depth map is noisy, particularly in low-texture or highly reflective regions (e.g., wet roads or direct sunlight).

B. Enhanced Vehicle Classification

Building upon our Phase 1 vehicle detection system, we implemented a more granular vehicle classification approach in Phase 2. Instead of simply detecting vehicles as a generic

class, we trained our YOLOv8x model to distinguish between multiple vehicle types:

- Cars: Sedan, SUV, hatchback, pickup trucks
- Trucks
- Bicycles
- Motorcycles

This finer-grained classification enables more realistic visualization by representing each vehicle with its appropriate 3D model in the Blender environment. Additionally, we implemented vehicle orientation detection by analyzing the aspect ratio and orientation of the bounding boxes. By calculating the angle of the vehicle relative to the camera's viewing direction, we can accurately position and rotate the 3D models to match the actual orientation of the vehicles in the scene.

The orientation estimation uses the following approach: for each detected vehicle, we analyze the width-to-height ratio of the bounding box and combine it with the position of the vehicle relative to the vanishing point. This information, along with the depth estimation, allows us to determine the approximate heading of the vehicle in 3D space. The resulting orientation information is crucial for creating realistic visualizations that accurately represent the traffic flow in the scene.

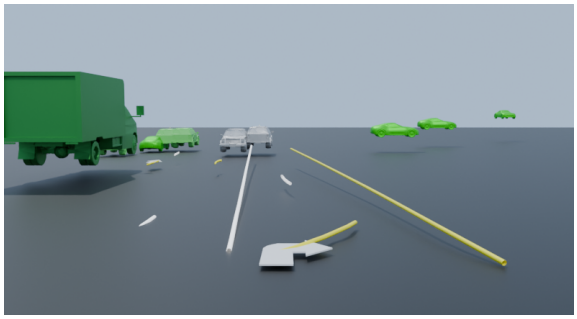

Fig. 9: Vehicle types, orientation and Road markings

C. Advanced Road Marking Detection

In Phase 2, we significantly expanded our road marking detection capabilities beyond basic lane lines. We implemented a 3D road marking detection system using a custom-trained Mask R-CNN model combined with our MiDaS-based depth estimation pipeline. This system processes video frames to detect various road markings, including:

- Directional arrows
- Lane indicators
- Pedestrian crossings
- Speed limit markings

The Mask R-CNN model was trained on a custom dataset of road markings to achieve high accuracy in segmenting these features. After detection, the system estimates the 3D coordinates of each marking using camera intrinsics and the depth map, enabling precise spatial positioning in the visualization.

For speed sign detection, we trained YOLOv8x on a custom dataset of speed limit signs with various numerical values. The model not only detects the presence of speed limit signs but also recognizes the specific speed value displayed. This information is then used to apply the appropriate texture to the speed sign 3D models in the visualization.

The system outputs both annotated video and per-frame JSON files containing 3D data, facilitating precise spatial understanding of road markings for applications in autonomous driving and advanced driver-assistance systems (ADAS).

D. Traffic Light Arrow Classification

Extending our traffic light detection system from Phase 1, we implemented the detection and classification of directional arrows on traffic lights. This feature is crucial for understanding lane-specific traffic signals at intersections.

We fine-tuned our YOLOv8x model on a dataset that includes traffic lights with various arrow configurations (left, right, straight, and combinations). The model was trained to recognize not only the color state of the traffic light but also the presence and direction of any arrows.

This enhanced traffic light detection system enables our visualization to accurately represent complex intersection signaling, providing a more comprehensive understanding of the traffic rules governing each lane.

E. Additional Object Detection

To create a more complete representation of the urban environment, we expanded our object detection capabilities to include various street furniture and infrastructure elements:

- Dustbins
- Traffic poles
- Traffic cones
- Traffic cylinders

These objects are detected using our YOLOv8x model, which was trained to recognize these additional classes. The detected objects are positioned in 3D space using depth estimation, allowing for accurate representation in the visualization.

The inclusion of these additional objects enhances the realism of the visualization and provides valuable context for understanding the complete urban environment surrounding the vehicle.

Fig. 10: Traffic Poles and Cylinders

F. Pedestrian Pose Estimation

In Phase 2, we significantly enhanced our pedestrian detection system by implementing full-body pose estimation. Instead of simply detecting pedestrians as bounding boxes, we now extract detailed skeletal information that represents their body posture and movement.

To identify pedestrians and their poses in each video frame, we implemented a two-stage process. First, we applied a YOLOv8 pose estimation model specifically trained to detect people and extract their body keypoints. For every frame, the model predicts bounding boxes around detected persons and provides the coordinates and confidence scores for a set of anatomical keypoints (such as nose, shoulders, elbows, wrists, hips, knees, and ankles).

Next, we estimated the depth of each frame using our transformer-based depth estimation model, which generates a depth map corresponding to the RGB image. By combining the 2D keypoint positions from the pose model with the depth values at those locations, we converted each keypoint into 3D world coordinates using the camera's intrinsic parameters.

This approach allowed us to robustly detect pedestrians and reconstruct their full-body poses in three dimensions for every frame, rather than simply classifying their presence. The resulting 3D skeletal representations provide a much more detailed understanding of pedestrian behavior, enabling more realistic visualization and potentially supporting advanced safety features such as pedestrian intent prediction.

IV. Phase 3: Bells and Whistles

A. Brake Light and Indicator Detection

In Phase 3, we implemented a sophisticated system for detecting and visualizing vehicle brake lights and indicator signals. This feature provides crucial information about the intentions of other road users, enhancing the safety and predictability of autonomous driving systems.

Our brake light and indicator detection system employs a multi-stage pipeline. First, vehicles are detected using a YOLOv8 model trained on the COCO dataset, focusing specifically on the car class. For each detected vehicle, the system determines if it's showing a rear view by analyzing color patterns and symmetrical bright spots in the lower portion of the vehicle ROI.

When a rear view is confirmed, a custom-trained tail light detection model identifies the tail light regions within the vehicle. The system then performs detailed analysis of each tail light by evaluating brightness levels, red color presence, and illumination patterns to determine if the light is ON or OFF. This analysis includes checking for:

- Significant bright red presence
- High brightness areas
- Moderate brightness with red components
- Very bright spots

The detection results are visually displayed with color-coded overlays and text annotations, distinguishing between left and right lights. In the Blender visualization, vehicles with active brake lights are represented with illuminated rear lights, providing an intuitive understanding of their braking status.

For indicator signals, the system analyzes temporal patterns in the tail light illumination, detecting the characteristic blinking pattern of turn signals. This information is then used to display appropriate indicator animations on the vehicle models in the visualization.

B. Parked vs. Moving Vehicle Classification

Another important cognitive ability implemented in Phase 3 is the distinction between parked and moving vehicles. This feature is essential for understanding the dynamic elements of the traffic scene and predicting potential interactions between road users.

Our system distinguishes between parked and moving vehicles using a sophisticated optical flow-based approach. First, YOLOv8 detects vehicles in each frame, then the RAFT neural network computes dense optical flow between consecutive frames. To account for camera movement, the system estimates ego-motion using homography-based flow and subtracts it from the total flow to isolate vehicle-specific motion.

A VehicleTracker class maintains vehicle identities across frames, storing position history and flow magnitudes. The system determines vehicle status using adaptive thresholds based on ego speed, where higher ego speeds require larger motion magnitudes to classify a vehicle as moving. For each tracked vehicle, the system calculates residual flow direction and magnitude, applying temporal filtering to stabilize measurements.

V. Extra Credit Features

A. Speed Bump Detection

As an extra credit feature, we implemented a robust speed bump detection system to enhance the safety and comfort of autonomous driving.

We trained a custom YOLOv8x model on a dataset of speed bumps captured under various lighting conditions, weather scenarios, and road types. The model was fine-tuned to recognize the distinctive visual patterns of speed bumps, including their characteristic shapes, textures, and road markings.

Fig. 11: Speed-Bump Detection

The detection process involves analyzing each frame for potential speed bumps, extracting their bounding box coordinates, and estimating their 3D position using our depth estimation pipeline. The detected speed bumps are then represented in the Blender visualization with appropriate 3D models, providing clear visual cues about their presence and location.

This feature significantly enhances the system's ability to navigate urban environments safely and comfortably, as it allows for appropriate speed adjustments before encountering speed bumps.

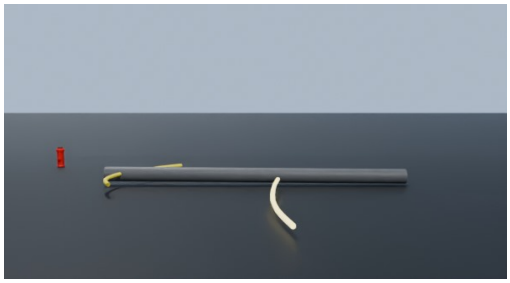

Fig. 12: Speed-Bump Detection in Blender

B. Collision Prediction

Another extra credit feature we implemented is collision prediction for pedestrians and vehicles. This safety-critical feature analyzes the spatial relationships between the ego vehicle and other road users to identify potential collision risks.

Our collision prediction system calculates the distance from the camera to each detected car or pedestrian using the depth estimation pipeline. If an object is detected within a critical distance threshold, it is classified as a potential collision risk and highlighted in red in the visualization.

This feature provides valuable safety information to the driver or autonomous system, allowing for proactive collision avoidance maneuvers. In the visualization, potential collision risks are immediately apparent, enhancing situational awareness and safety.

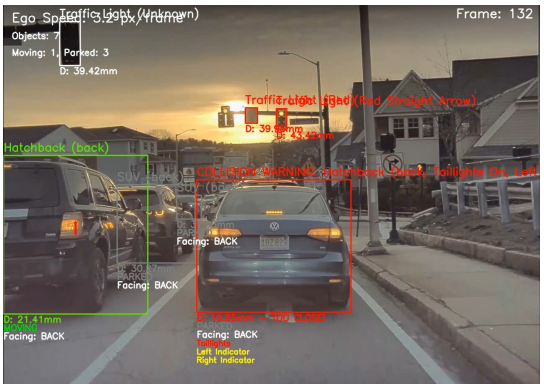

Fig. 13: Collision Warning (RED), Brake and indicator light detection, Moving Parked Vehicles detection, Traffic Light Arrow.

VI. Blender Visualization

To create the final visualization, we developed a pipeline that converts the detected objects and their 3D coordinates into a format compatible with Blender. The process involves several key steps:

First, we export all detection results to a structured JSON file containing frame-by-frame information about lanes, vehicles, pedestrians, traffic lights, road signs, and other detected features. Each object includes its 3D coordinates, classification, and any relevant state information (such as brake light status, movement status, etc.).

In Blender, we developed a custom Python script that imports this JSON data and creates corresponding 3D objects for each detected element. Lanes are represented as curved paths using Blender's curve objects, vehicles and pedestrians are visualized using the provided 3D models, and traffic lights and road signs are represented with their respective models and textures.

The script animates these objects over time based on the frame-by-frame data, creating a smooth and continuous visualization of the scene. We also implemented camera controls that allow viewing the scene from different perspectives, including a third-person view similar to Tesla's dashboard visualization.

For Phase 3 features, we added special visual effects to represent brake lights, indicators, and vehicle motion. Brake lights are visualized as illuminated red lights on the rear of vehicle models, while indicators are shown as blinking yellow lights. Moving vehicles are displayed with directional arrows indicating their movement vector, while parked vehicles are shown without such indicators.

For the extra credit features, speed bumps are represented with appropriate 3D models, and potential collision risks are highlighted in red with warning indicators. These visual cues provide immediate and intuitive information about safety-critical aspects of the driving environment.

The final output is rendered as a video showing the ego vehicle's perspective along with the detected objects in their appropriate 3D positions, providing an intuitive understanding of how the system perceives its environment.

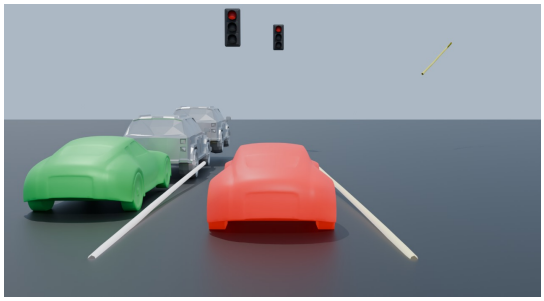

Fig. 14: Collision warning detection (RED),Moving Vehicles (Green) and Parked Vehicles (Gray)

VII. Corner Cases and Limitations

Despite our efforts to create a robust system, several corner cases and limitations remain:

A. Extreme Lighting Conditions

Our system struggles with very bright or very dark scenes, which affect both object detection and depth estimation accuracy. This is a common challenge in vision-based systems, as noted in Tesla's own approach to autonomous driving, which faces similar issues with "low-visibility conditions, weather changes, and diverse driving styles across different regions"[4]. Camera-based sensors have inherent difficulties detecting objects in low visibility conditions like fog, rain, or nighttime[5], which significantly impacts our system's performance in these scenarios.

B. Occlusions

When objects are partially hidden behind others, detection accuracy decreases significantly. This is particularly problematic for pedestrian pose estimation and vehicle orientation detection. Our current implementation lacks sophisticated occlusion handling, which can lead to incomplete or inaccurate 3D reconstructions when parts of objects are not visible.

C. Distant Objects

Both detection and depth estimation become less reliable for distant objects, limiting the effective range of our system. This is a fundamental limitation of monocular depth estimation approaches, which struggle with accurate depth perception at longer distances[8][10]. As noted in research on depth estimation for autonomous vehicles, "the longer the distance between the camera lenses, the more accurate the depth estimation"[10], highlighting the inherent limitations of our camera setup.

D. Unusual Road Configurations

Non-standard road layouts or temporary changes (construction zones, detours) can confuse the lane detection system. This is similar to the challenges faced by other autonomous systems, as illustrated by examples where "people can creatively navigate around a Waymo by simply drawing a circle around it"[4], showing how unusual configurations can confuse even sophisticated autonomous systems.

E. Camera Visibility Limitations

There are inherent blind spots in any camera-based system. Our system faces similar limitations, particularly at intersections where objects might be outside the camera's field of view. As noted in discussions about camera-only self-driving systems, human vision provides "stereo vision on a swivel with higher dynamic range and near instant adaptation to changes in brightness"[4], capabilities that our fixed camera system cannot fully replicate.

F. Pose Rendering Limitations in Blender

Although our system detects poses accurately, we encountered challenges rendering these poses in Blender. The primary issues stem from technical workflow constraints, such as missing bone keyframes, which are essential for proper animation rendering. Additionally, certain Blender versions may revert to the rest position due to bugs, and incorrect visibility or modifier settings can prevent the pose from appearing as intended in the final render. Addressing these issues requires careful management of keyframes, armature properties, and render settings to ensure the detected poses are faithfully visualized.

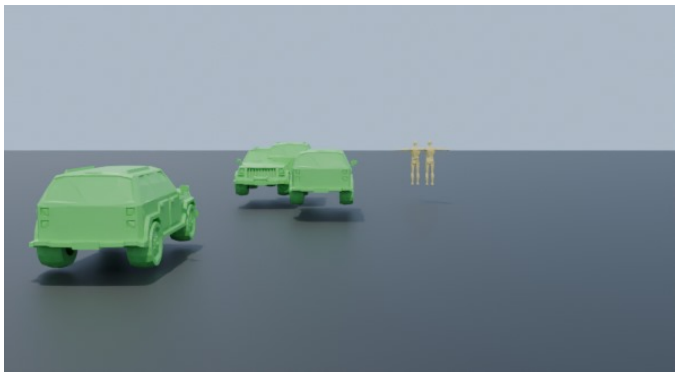

Fig. 15: Failed Case 1

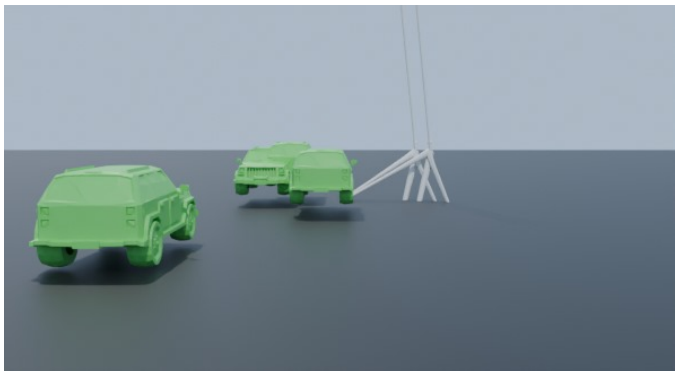

Fig. 16: Failed Case 2

VIII. Potential Improvements

Based on our experience and the limitations identified, several improvements could enhance our system:

A. Multi-frame Analysis

Implementing more sophisticated temporal analysis could improve detection stability and help handle occlusions by tracking objects across frames. This approach would allow our system to maintain object identity and position even when temporarily occluded, improving the continuity and reliability of the visualization.

B. Adaptive Thresholds

Dynamically adjusting detection thresholds based on lighting conditions and scene complexity could improve performance in challenging environments. This would allow the system to adapt to varying conditions, such as bright sunlight or nighttime driving, maintaining consistent performance across different scenarios.

C. Sensor Fusion

Integrating additional sensor data, such as LiDAR or radar, would provide more reliable depth information and improve object detection in poor visibility conditions. As noted in research, "LIDAR provides accurate depth but is sparse in vertical and horizontal resolution; RGB images provide dense texture but lack depth information"[6], suggesting that a fusion approach could leverage the strengths of both sensor types. However, we must be mindful that "sensor integration is an extremely hard problem"[4] and could introduce new failure modes if not implemented carefully.

D. Advanced Simulation Testing

We could develop a more comprehensive testing framework to evaluate our system under a wider range of scenarios. This would allow us to identify and address edge cases before deploying the system in real-world environments.

E. Improved Depth Estimation

Implementing a more sophisticated depth estimation algorithm, possibly combining learning-based methods with geometric constraints, could improve the accuracy of 3D positioning. Recent research has focused on "enhancing depth estimation in adverse lighting scenarios"[8], which could be particularly valuable for improving our system's performance in challenging conditions.

IX. Challenges Faced and Feedback

Throughout this project, we encountered several significant challenges:

A. Computational Resources

Processing high-resolution video frames with multiple deep learning models in real-time required substantial computational resources, limiting our ability to implement more complex algorithms. This is a common challenge in autonomous vehicle systems, where balancing performance with computational efficiency is crucial.

B. Data Variability

The provided video sequences exhibited significant variability in lighting, weather, and road conditions, making it difficult to achieve consistent performance across all scenarios. This highlights the importance of diverse training data and robust algorithms that can handle a wide range of environmental conditions.

C. Integration Complexity

Combining multiple detection systems (lanes, vehicles, pedestrians, etc.) into a coherent visualization required careful coordination and data synchronization. Each component had its own processing pipeline and output format, necessitating a well-designed integration framework to ensure consistent and accurate visualization.

D. Depth Estimation Accuracy

Achieving reliable depth estimation from monocular video remains a challenging problem, particularly for distant objects or in scenes with limited texture. As noted in research, monocular depth estimation has inherent limitations[11], and while our approach achieved reasonable results in many scenarios, there is still significant room for improvement.

For future improvements to the project, we suggest:

- **Standardized Evaluation Metrics:** Providing clear metrics for evaluating the quality of visualizations would help guide development and enable objective comparison between different approaches.
- **Additional Training Data:** Access to a more diverse set of labeled training data would enable better fine-tuning of detection models for specific object classes.
- **Real-time Feedback:** Implementing a system for real-time evaluation during development would accelerate the iteration cycle and help identify issues earlier.
- **Modular Framework:** Developing a more modular framework that allows easy swapping of different detection and visualization components would facilitate experimentation with alternative approaches.

In conclusion, our Tesla-inspired visualization system successfully implements a comprehensive set of features for understanding and representing the driving environment. While challenges remain, particularly in handling edge cases and ensuring consistent performance across all conditions, the system provides a solid foundation for future development and improvement.

Link for Pitch Video:
https://drive.google.com/drive/folders/1iuleoQ68BJFc04H3BPSZ7smhzc