2020-01-0718

Creating Driving Scenarios from Recorded Vehicle Data for Validating Lane Centering System in Highway Traffic

Seo-Wook Park, Kunal Patil, Will Wilson and Mark Corless
MathWorks
Gabriel Choi and Paul Adam
General Motors

Abstract

The adoption of simulation is critical to reducing development time and enhancing system robustness for Advanced Driver Assistance Systems (ADAS). Automotive companies typically have an abundance of real data recorded from a vehicle which is suitable for open-loop simulations. However, recorded data is often not suitable to test closed-loop control systems since the recorded data cannot react to changes in vehicle movement. This paper introduces a methodology to create virtual driving scenarios from recorded vehicle data to enable closed-loop simulation. This methodology is applied to test a lane centering application. A lane centering application helps a driver control steering to stay in the current lane and control acceleration and braking to maintain a set speed or to follow a preceding vehicle. The driver's vehicle is referred to as the ego vehicle. Other vehicles on the road are referred to as target vehicles. To test the lane centering system in simulation, engineers must model the ego vehicle (sensors and dynamics) as well as the scenario (roads and target vehicles). A virtual driving scenario is created by reconstructing roads and target vehicles using GPS, camera-based lane detections, radar-based vehicle detections, and map data. The virtual driving scenario is integrated into a closed-loop simulation to assess the behavior of a lane centering system.

Introduction

A lane centering system requires longitudinal and lateral control to guarantee appropriate lane-following behavior while maintaining a safety distance from the leading vehicle [1]. The driving automation system executes controls for steering, acceleration and brake. However, for this level of automation (L2 or L3), human driver is still required to supervise the system and ready to take over the control in case the system is malfunctioning [2]. Considering the complexity of driving automation system and intrinsic uncertainty of operational environments, its performance needs to be thoroughly tested against standard requirements (e.g. ISO 15622/22178) and real-world driving scenarios.

The vehicle equipped with the driving automation system would have to be driven tens of thousands of miles to evaluate the performance and reliability. The adoption of a virtual simulation tool is key to reducing development time and enhancing system robustness [3,4].

F. Martinelli proposed a methodology for design and verification of a traffic jam assist system via simulation using MATLAB and Simulink [5]. Virtual driving scenarios representing typical real-world driving situations were used to verify appropriate behavior of the control system and evaluate its performance. One of main concerns of their approach is how well the virtual driving scenarios can represent the real-world traffic situation in terms of accuracy of road network and vehicle motions.

In this paper, we introduce a methodology to create driving scenarios from recorded vehicle data such as GPS, vehicle speed, on-board radar and camera sensors. When test engineers observe unwanted system behavior from the real traffic, they want to identify the root causes by reproducing the driving scenario with simulation.

The methodology to create a virtual driving scenario consists of the following steps:

1. Record and select data
2. Reconstruct road network
3. Localize ego trajectory
4. Reconstruct target vehicles
5. Compare with recorded video

The virtual driving scenario can then be used to test an ADAS system using simulation. This paper demonstrates applying this methodology to test a lane centering application.

Record and select data

To enable reproducing a real-world traffic situation with a virtual driving scenario in order to test the lane centering system, our test engineer drove on highways and collected data over the CAN bus. Data collected from the CAN bus included:

- Global Positioning System (GPS)
- Vehicle speed
- Object detections from radar sensor
- Lane marking detections from camera sensor

In addition to the CAN data, the test engineer logged video to an MP4 file using a separate camera. This recording would be used at the verification stage for visual comparison against the open-loop virtual driving scenario.

We read the CAN messages and MP4 data into MATLAB and visualized the results. We then identified scenarios with near

collisions that would be interesting to reproduce in closed loop simulation.

Reconstruct road network

We identified the general area where the data was collected based on the recorded GPS data. To create a road model, we needed to correlate the recorded GPS location data to a map data. Today, many digital map databases (e.g., Google Map, OpenStreetMap, HERE Map, etc.) are available to create the driving scenario using road attributes extracted from the map data [6]. Yang Zheng and Michael Zilske used OpenStreetMap to generate the virtual traffic scenarios [6,7].

We used a high-definition (HD) map as the source of road network information because it has extremely high resolution at centimeter-level. The precise geometry of the HD map makes it suitable for automated driving workflows.

The HD map data is composed of tiled mapping layers that provide access to accurate geometry and robust attributes of a road network. The layers are grouped into the following models:

- **Road Centerline Model**: Provides road topology (specified as nodes and links in a graph), shape geometry, and other road-level attributes.
- **HD Lane Model**: Provides lane topology (as lane groups and lane group connectors), highly accurate geometry, and lane-level attributes.
- **HD Localization Model**: Provides features to support vehicle localization strategies.

We began by plotting the recorded GPS sequence on the HD map as shown in Figure 1. This allowed us to identify the section of road corresponding to the test drive.

Figure 1. HD Map and driving route with recorded GPS data

We then extracted road center coordinates and lane information from the Road Centerline and HD Lane Models. We used this information to programmatically create a model of the road network as shown in Figure 2. We used the driving scenario object from Automated Driving Toolbox [8] to create the model the road network.

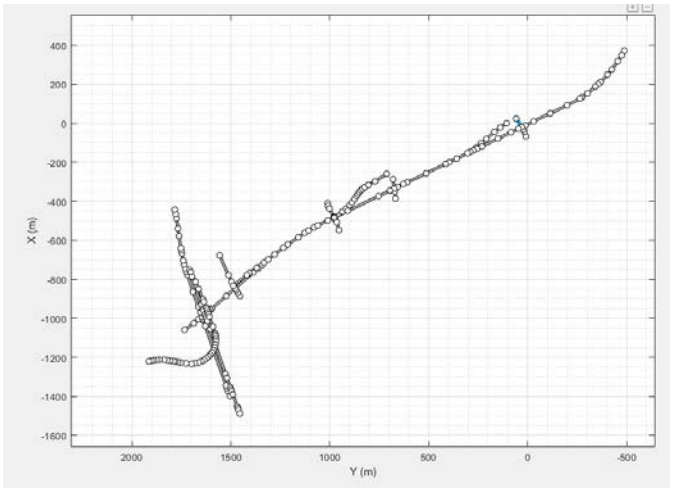

Figure 2. Road network constructed from HD Map

The road network imported from the HD map contained more road network information than required to reproduce the test route. To simplify the network, we removed unnecessary roads elements such as bridges and cross roads. The edited road network is shown in Figure 3.

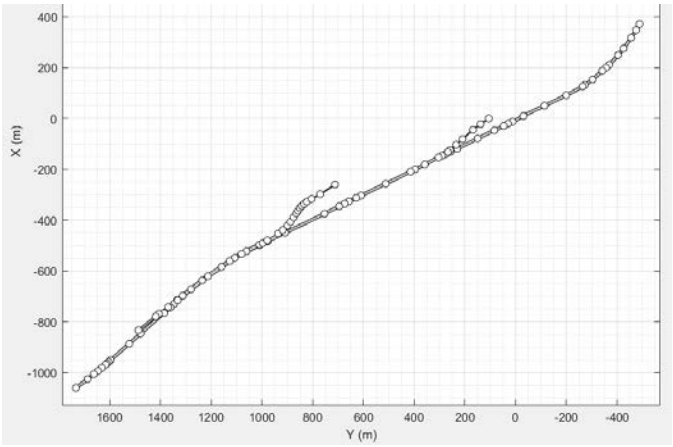

Figure 3. Edited road network

Localize ego trajectory

Once the road network was reconstructed, we needed to add the surrounding detected objects to the scenario. Since the recorded data was from a highway drive, the only detected objects we needed to represent were target vehicles. The target vehicle detections are based on recorded detections from the radar. These detections are measured with respect to the ego vehicle. To construct equivalent target vehicles in the virtual driving scenario, we needed to transform the detections from ego coordinates to world coordinates. This transform depends on accurate position information for the ego vehicle.

To assess how suitable the recorded GPS data was to represent the ego trajectory, we plotted GPS data atop the constructed road network as shown in Figure 4.

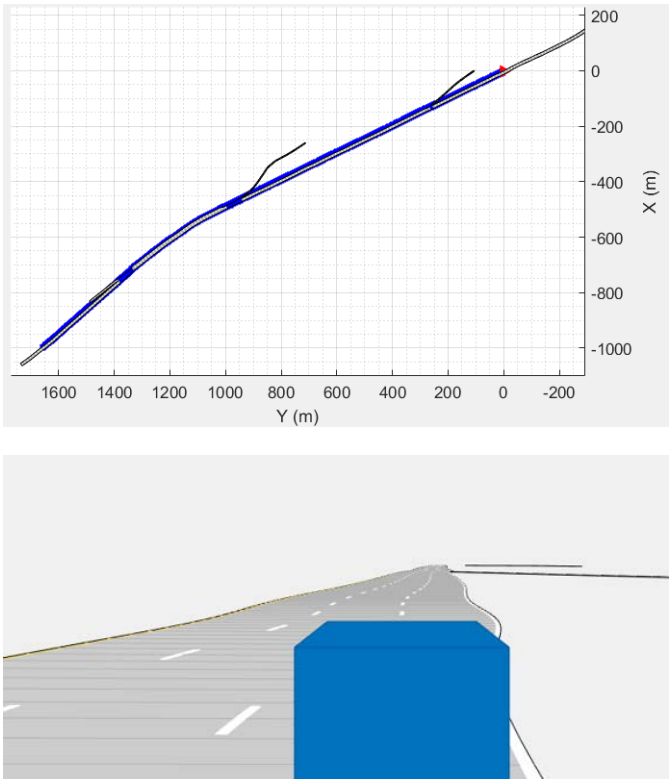

Figure 4. Ego vehicle trajectory on road network (top: scenario top view, bottom: ego-centric projective perspective view)

We noticed that the plotted position deviated further from the lane center than we expected. To explore this further, we compared the ego lateral deviations from lane center by the GPS based trajectory and camera lane sensor as shown in Figure 5. From this we assessed that the GPS based trajectory was not accurate enough for us to base our target vehicle transforms.

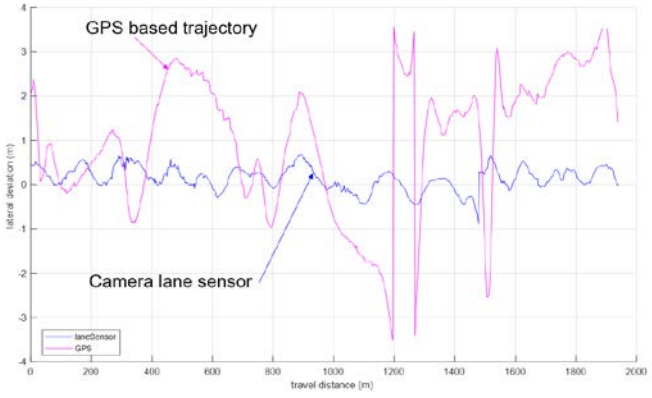

Figure 5. Lateral deviation of ego vehicle from lane center

We improved the lateral position accuracy of the GPS based trajectory by localizing it against the lane sensor data. The improved localized trajectory is show in Figure 6.

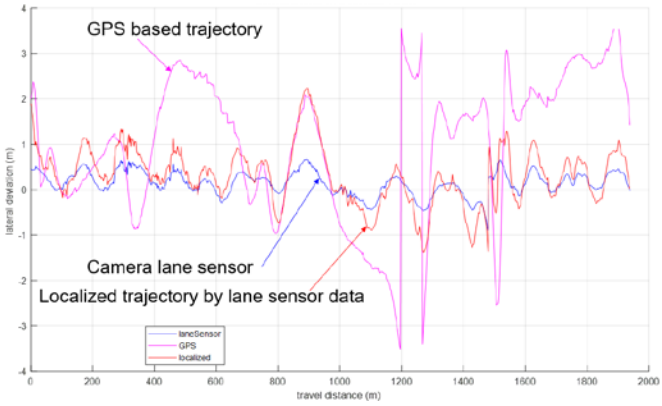

Figure 6. Localized trajectory by lane sensor data

We determined that the trajectory localized by the lane sensor was suitable for use in the coordinate transforms to reconstruct the target vehicle trajectories.

Reconstruct target vehicles

The radar detections provide the position and velocities of the surrounding target vehicles with respect to the ego vehicle. We transformed the target vehicle positions to the world coordinate and estimated the orientation angle based on vehicle motion using the following equations 1 and 2. Figure 7 shows the relationships between the variables used in these equations.

$$(X_t, Y_t) = (X_{ego}, Y_{ego}) + \mathbb{R}(\Psi_{ego}) \cdot (x_t, y_t)^T \quad (1)$$

where $\mathbb{R}$: rotation matrix, (x, y): position in ego coordinate, (X, Y): position in world coordinate

$$\Psi_{target} = \tan^{-1}\left(\frac{Y_t - Y_{t-1}}{X_t - X_{t-1}}\right) \quad (2)$$

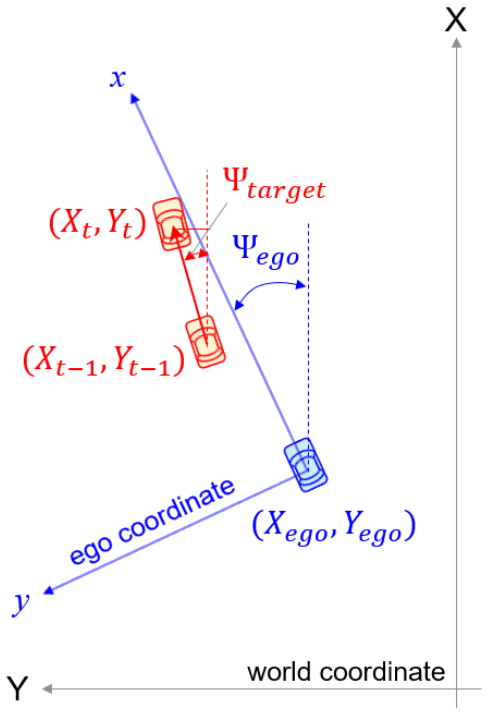

Figure 7. Estimation for orientation angle of target vehicle

We explored the target vehicle trajectories and noticed that after the coordinate transforms, some of the yaw angles were noisy. We smoothed the estimated trajectory using Savitzky-Golay filter [9]. The original and smoothed trajectories are shown in Figure 8.

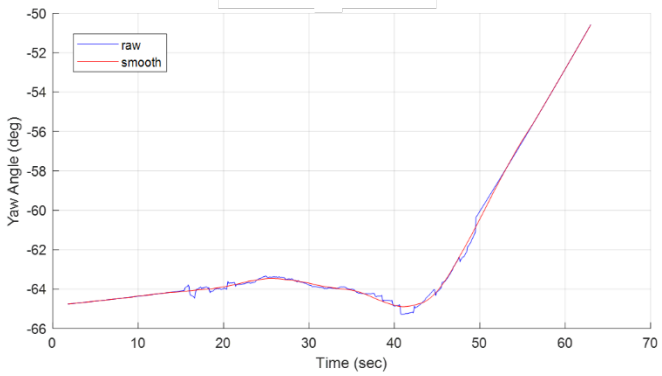

Figure 8. Estimated yaw angle trajectory

Once we determined the trajectories for the target vehicles were reasonable, we used the driving scenario in Automated Driving Toolbox to add vehicle actors based on the trajectories calculated in world coordinates.

Compare with recorded video

To verify the virtual driving scenario, we ran the scenario in open-loop and visually compared to the corresponding video we recorded from the vehicle. We looked at road width, road markings, location of the ego vehicle, and location of surrounding vehicle as shown in Figure 9. Note that only vehicles detected by the radar were reconstructed. Vehicles that were not detected by the radar were not reconstructed. For example, in Figure 9, the vehicle in the first lane from the left of the video was not detected by the radar, so it is not included in the virtual scenario.

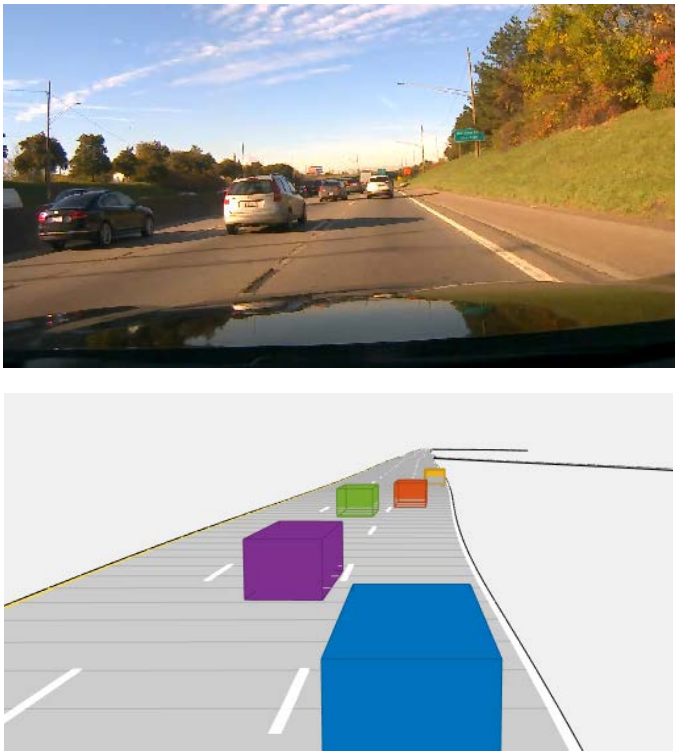

Figure 9. Recorded video (top) vs. reconstructed driving scenario (bottom)

Simulate closed-loop scenario

Once we gained confidence that the virtual scenario behaved as expected in open-loop, we integrated it with a lane centering control algorithm for closed-loop simulation. We used the same Simulink model described in [5]. The key components of the algorithm are shown in Figure 10.

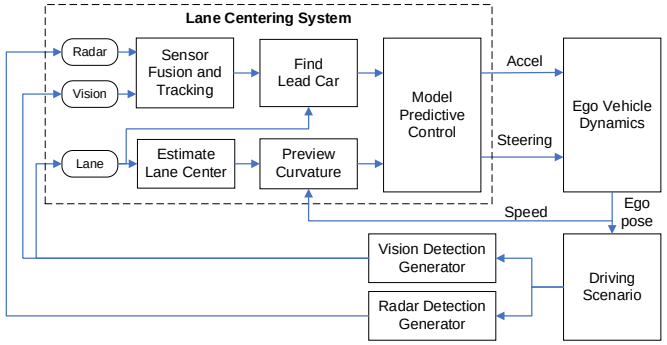

Figure 10. Schematic of closed-loop model for lane centering system

The lane centering system consists of components for:

- **Sensor fusion and tracking**: Fuses object detections from radar and camera sensors to track other vehicles on the road
- **Find lead car**: Decides if one of the fused detections represents a lead vehicle in the same lane as the ego.
- **Estimate lane center**: Estimates lane center based on detections from the camera sensor and current state of the ego vehicle.
- **Preview curvature**: Estimates lane centers for a time horizon into the future.
- **Model predictive control**: Performs lateral and longitudinal control.

Testing the lane centering system in simulation requires additional components for:

- **Vision detection generator**: Models object detections and lane detection generated by a camera-based sensing system.
- **Radar detection generator**: Models object detections generated by a radar-based sensing system.
- **Ego vehicle dynamics**: Models dynamics of the vehicle under test. Reacts to commanded acceleration and steering. Generates pose information (position and speed).
- **Driving scenario**: This is the virtual driving scenario constructed from recorded data. Models the road and other vehicles on the road with respect to the ego pose. Provides ground truth information used by the sensor models.

We ran the simulation and visualized the results at run time as shown in Figure 11. Overall the results demonstrated that the ego vehicle showed appropriate lane-following behavior while maintaining a safety distance from the leading vehicle.

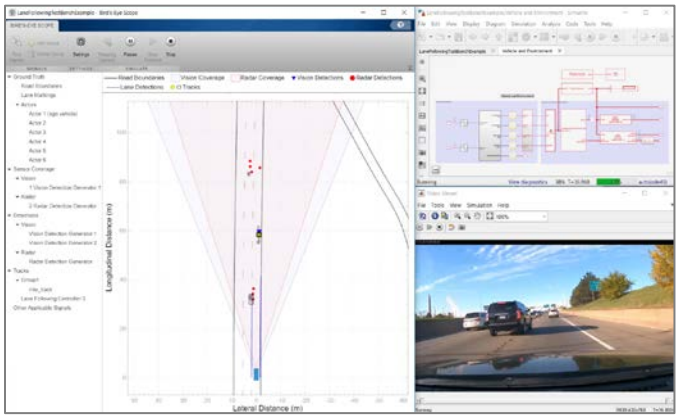

Figure 11. Closed-loop simulation using reconstructed virtual driving scenario

The simulation also provided us insight into areas where the algorithm under test could be improved or encountered edge conditions. For one reconstructed scenario we noticed that the distance from the leading vehicle dropped below the pre-defined safety three times as shown in Figure 12. This log shows three interesting cases labeled a, b, and c.

Case (a) shows the transient behavior before the controller is fully stabilized. This is likely due to the high initial speed with high set-velocity (98.6 kph).

Case (b) was caused by a cut-in vehicle at low speed. This can be seen in the from the logged video image in Figure 13. the slow-moving vehicle from right lane cuts in the ego vehicle, the ego vehicle reduces longitudinal speed as the headway distance drops below the safety distance.

Case (c) was caused by a vehicle from the left lane suddenly cutting in front of the ego vehicle with very close distance. This can be seen in the from the video recording in Figure 14.

Having a closed loop simulation which reproduces these edge conditions allows us to explore design alternatives which can improve overall performance.

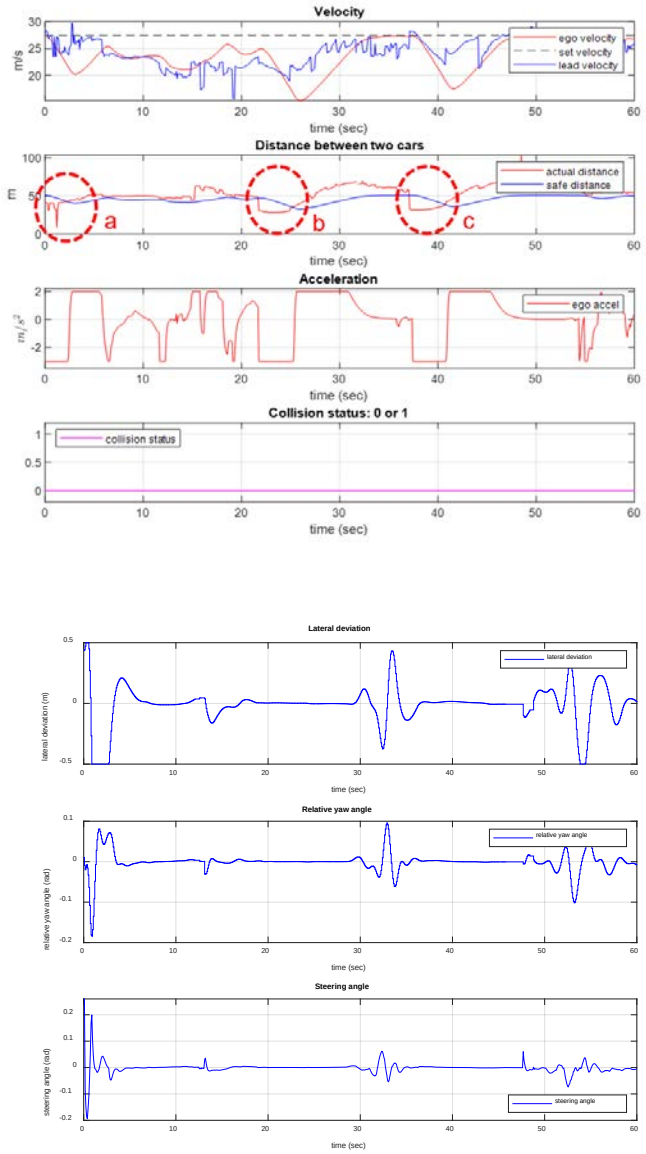

Figure 12. Simulation result (top: longitudinal control performance, bottom: lateral control performance)

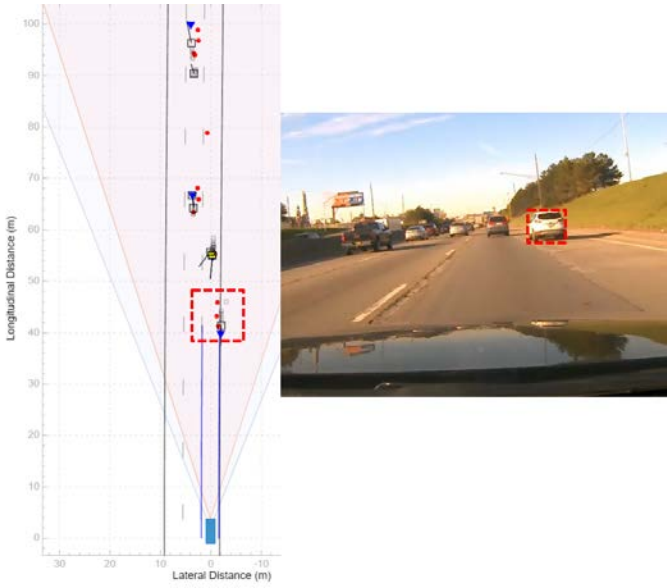

Figure 13. Driving case (b): cut-in vehicle at low speed

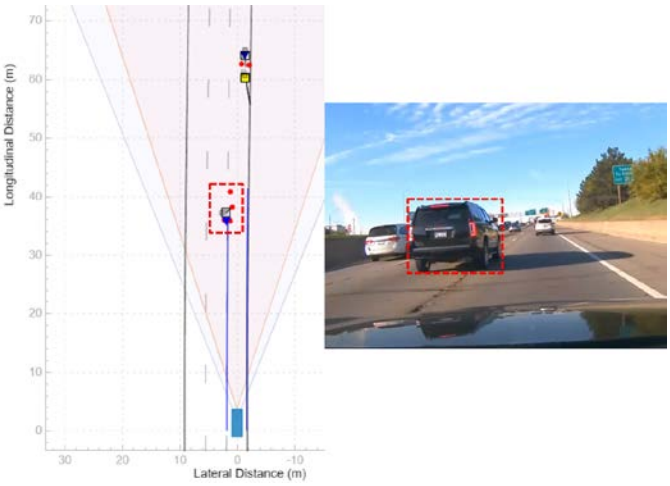

Figure 14. Driving case (c): cut-in vehicle with too close distance

Conclusion

We presented a methodology to create virtual driving scenarios from recorded vehicle GPS, vehicle speed, on-board radar and camera sensors, and HD map. With this approach, real-world driving scenarios were successfully reproduced in a closed-loop simulation environment.

Although we demonstrated this methodology used specific sensors and map information, the methodology can be scaled to different sensors and map providers. For example, we used radar detections due to the configuration of our test vehicle, but the same methodology could be used with lidar data. Similarly, instead of HD map, road network information could be imported from other map databases like OpenStreetMap.

With this approach, a real-world driving scenario can be reproduced in a virtual simulation environment. This enables engineer to assess functional behavior and gain insight while reducing time spent in the vehicle especially when the test cases have any hazardous scenarios.

References

1. National Highway Traffic Safety Administration, "Functional Safety Assessment of an Automated Lane Centering System," DOT HS 812 573, Aug. 2018
2. SAE J 3016-2018, Taxonomy And Definitions For Terms Related To Driving Automation Systems For On-Road Motor Vehicles, 2018-06
3. Marc René Zofka, et al, "Testing and Validating High Level Components for Automated Driving: Simulation Framework for Traffic Scenarios," IEEE Intelligent Vehicles Symposium, 2016
4. Till Menzel, et al., "Scenarios for Development, Test and Validation of Automated Vehicles," IEEE Intelligent Vehicles Symposium, 2018
5. F. Martinelli, F. Stevenin, S.-W. Park, M. Corless, "Design and Verification of a Traffic Jam Assist System," SIA Simulation Numérique, April 3-4, 2019, St-Quentin-en-Yvelines, France
6. Yang Zheng, et al., "Exploring OpenStreetMap Availability for Driving Environment Understanding," IEEE Intelligent Vehicles Symposium, 2018
7. Michael Zilske, et al., "OpenStreetMap for traffic simulation," Proceedings of the 1st European state of the map : OpenStreetMap conference, 2011
8. Automated Driving Toolbox (https://www.mathworks.com/products/automated-driving.html)
9. Orfanidis, Sophocles J. Introduction to Signal Processing. Englewood Cliffs, NJ: Prentice-Hall, 1996.