

Project 9 (MATLAB Based) Report

Submitted for

VEHICLE DYNAMICS (UME525)

Submitted by:

Priyanshu Bist (102208010)

Ansh Gupta (102208004)

Tushar (102208031)

Sugam (102208030)

BE Third Year Batch – 3MEE

Submitted to:

Dr. Jai Prakash Tripathi



THAPAR INSTITUTE
OF ENGINEERING & TECHNOLOGY
(Deemed to be University)

Department of Mechanical Engineering

**Thapar Institute of Engineering & Technology,
Patiala, Punjab**

July-December 2024

DECLARATION

I assert that the statements made and conclusions drawn are an outcome of my own research work.

I further certify that the work contained in this report is original and has been done by me under the general supervision of our supervisor.

I have followed the guidelines provided by the University in writing this report.

I also declare that this report is the outcome of my own effort and has not been submitted to any other university for the award of any degree.

Highway Lane Change

Objectives:

- Provide and explain all the mathematical derivations used in the project with relevant sketches.
- Show the steps involved in the MATLAB script/program algorithm in a flowchart.
- For the analysis, choose small, medium, and large values of each internal parameter of the vehicle (i.e. track width, wheelbase, CG position), or choose small, medium, and large values of each externally governed parameter of the vehicle (i.e. grade, friction coefficient, vehicle mass, steering angle, radius of curvature of turning, bump type/size).

Project Description: Highway Lane Change is a project developing an autonomous vehicle system that will enable lane-changing maneuvers on highways. Using the toolkits of MATLAB - particularly the Automated Driving Toolbox, Navigation Toolbox, and Model Predictive Control Toolbox - the system will integrate the planning, control, and real-time simulation methodologies to ensure the safe and efficient operation of the autonomous vehicle in dynamic traffic [1]. This report's prime reference is the instructional video on the highway lane change example that elucidates everything on system design, simulation, and evaluation.

System Overview:

The project system includes the following:

- **Scenario and Environment Subsystem:** The subsystem creates highway scenarios with dynamic traffic, which includes test vehicles and other vehicles in its surroundings. Supports virtual testing on various scenarios including straight roads, curved roads, and scenarios imported from HD sources.
- **Lane Change Planner Subsystem:** Calculates the terminal Frenet states for the cruise control, lead car following, and lane change maneuvering. This is evaluated using a cost-based assessment to find a collision-free optimal trajectory.
- **Lane Change Controller Subsystem:** Adaptive Model Predictive Control (AMPC) implementation to track the trajectory. Lateral and longitudinal control are combined for both steering and optimal acceleration.
- **Vehicle Dynamics Subsystem:** The subsystem models Longitudinal, lateral, and yaw motion using a 3-DOF bicycle. It outputs all parameters related to longitudinal velocity, lateral velocity, and the dynamics of the Ego vehicle.
- **Metrics and Visualization Subsystems:** Evaluate metrics related to simulations, such as jerk, acceleration, and collision status.

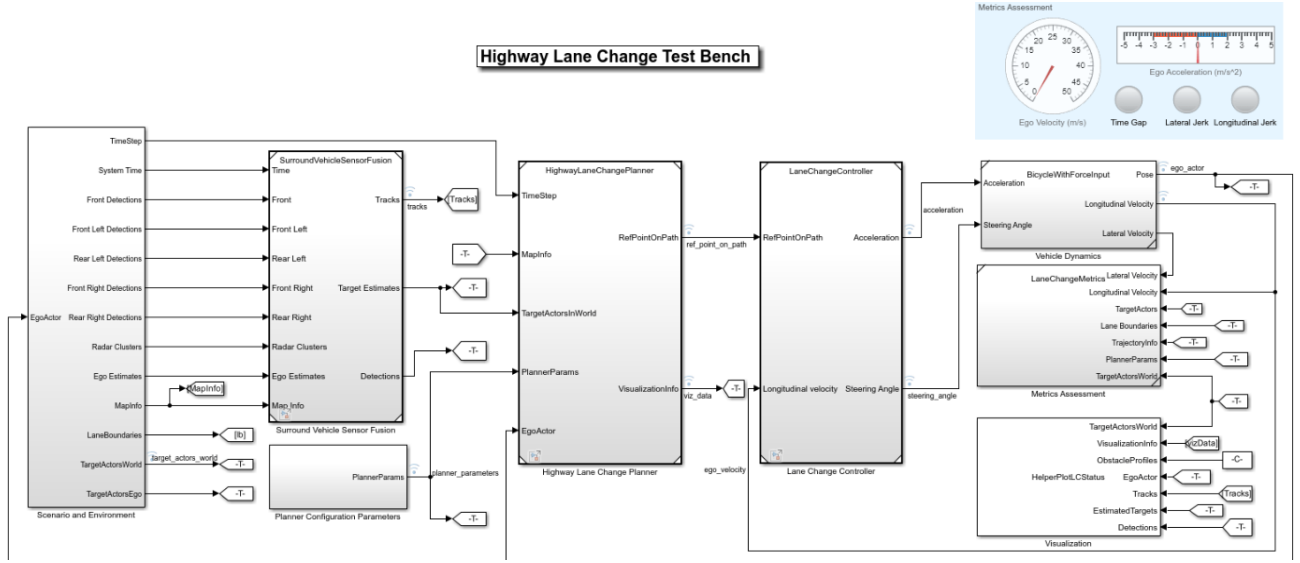


Fig 1: Scenario and Environment created using Predefined Toolkits from Mathworks Inc.

Mathematical Derivations:

1. Frenet Coordinate System: The Frenet coordinate system is one of the most fundamental tools in autonomous driving on urban roads, highways, and other traffic scenarios. Unlike the Cartesian coordinate system, the Frenet system simplifies vehicle motion analysis by using a predefined reference path as its basis. Since the system aligns itself with the tangent and normal vectors of this reference path, this essentially transforms the two-dimensional motion of the vehicle into two independent one-dimensional components. The present position of the vehicle, referred to as point M in the Cartesian coordinate system, is characterized by variables $[x, y, \theta, \kappa, v, a]$, which stand for the position, orientation, curvature, velocity, and acceleration. The corresponding reference point N is determined by projecting point M onto the reference path with its properties in the Cartesian system represented as $[x_r, y_r, \theta_r, \kappa_r, \kappa_r', s_r]$ [2]. When transitioning to the Frenet coordinate system, the vehicle's motion is expressed relative to the reference path. The state variables become $[s, s', s'', l, l', l'']$ where: s is longitudinal displacement along the reference path; s' is the velocity in the tangent direction of the reference path; s'' is acceleration in the tangent direction; l is lateral displacement perpendicular to the reference path; l' is the rate of change of lateral displacement relative to s and finally l'' rate of change of l' relative to s [3].

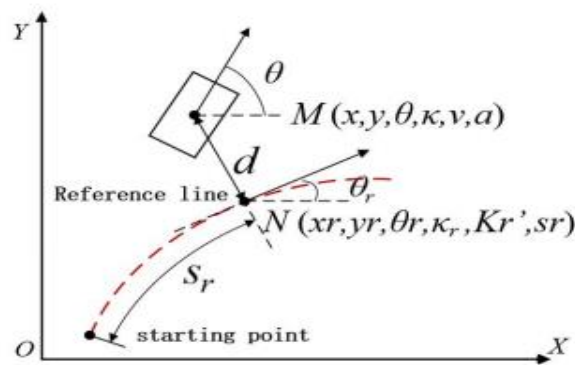


Fig 2: Coordinate System Conversion

- 2. Cost Function:** When trajectories are created, but no optimal selection exists. To determine which is the best one, a cost function is utilized to analyze each trajectory with the points of concern being comfort, safety, and centerline proximity. The trajectory with the lowest cost is therefore the optimal choice [4]. The cost function $J[i]$ is:

$$J[i] = w_0 J_{\text{jerk}}[i] + w_1 J_{\text{safety}}[i] + w_2 J_{\text{offset}}[i]$$

Here J_{jerk} , J_{safety} , and J_{offset} represent the costs associated with trajectory smoothness, safety, and deviation from the centreline, respectively. The weights w_0 , w_1 and w_2 reflect the driving style preferences.

2.1. Jerk Cost Function: Jerk, the rate of change of acceleration, is a key indicator of trajectory smoothness and passenger comfort. By decoupling longitudinal and lateral motions in the Frenet frame, the total jerk cost for a trajectory can be computed as:

$$J_{\text{jerk}}[i](t) = \int_{t_0}^{t_1} \ddot{d}^2(t) dt + \int_{t_0}^{t_1} \ddot{s}^2(t) dt$$

Where $\ddot{d}(t)$ and $\ddot{s}(t)$ are the second derivatives of lateral and longitudinal accelerations, respectively.

2.2. Safety Cost Function: Although all candidate trajectories are collision-free, their proximity to obstacles affects safety. A safer trajectory maintains greater distance from obstacles, particularly larger ones. This is modeled using a convolution approach inspired by the Gauss-Laplace operator [5]:

$$J_{\text{safety}}[i] = f * \text{colli_V}[i]$$

$$f(x) = g(x) + \min(|g(x)|)$$

Where $f(x)$ is the convolution kernel, colli_V is collision value and $g(x)$ is a variation of the Gauss-Laplace operator:

$$g(x) = \frac{1}{2\pi\sigma} \cdot e^{-\frac{(x-u)^2}{2\sigma^2}} \cdot \frac{\sigma^2 - x^2}{\sigma^4}$$

2.3. Deviation Cost Function: Staying close to the road's centreline minimizes steering corrections and reduces the likelihood of triggering lane departure warnings, thus improving maneuverability and safety. The deviation cost function measures how much a trajectory diverges from the centerline.

In a discrete time frame, this can be simplified where N is the number of trajectory points, at point i.

$$J_{\text{offset}}[i] = \sum_{j=0}^N d_{ij}^2 / \sum_{j=0}^N s_{ij}^2$$

3. Jerk Calculation: Jerk measures abrupt changes in acceleration, impacting passenger comfort.

\mathbf{r} Position

$\frac{d\mathbf{r}}{dt} = \dot{\mathbf{r}}$ Velocity (speed)

$\frac{d^2\mathbf{r}}{dt^2} = \ddot{\mathbf{r}}$ Acceleration

$\frac{d^3\mathbf{r}}{dt^3} = \dddot{\mathbf{r}}$ Jerk

$\frac{d^4\mathbf{r}}{dt^4} = \mathbf{r}^{(4)}$ Snap (jounce)

$\frac{d^5\mathbf{r}}{dt^5} = \mathbf{r}^{(5)}$ Crackle

$\frac{d^6\mathbf{r}}{dt^6} = \mathbf{r}^{(6)}$ Pop

4. Time to Collision (TTC): Ensures safety by calculating the time remaining before a collision:

$$TTC = \frac{d_{\text{gap}}}{\Delta v}$$

Where d_{gap} is the distance between the Ego and lead vehicle and Δv is relative velocity.

Steps Involved (Basic Algorithm):

☐ **Input Acquisition:**

- Load scenario (road, traffic, Ego vehicle).
- Initialize parameters: friction, mass, etc.

☐ **Trajectory Generation:**

- Generate multiple candidate paths.
- Evaluate using the cost function.

□ **Collision Checking:**

- Check for safe trajectories (no collisions).
- Select the optimal path.

□ **Control Implementation:**

- Use AMPC to follow the selected trajectory.

□ **Metrics Monitoring:**

- Calculate lateral/longitudinal jerk, TTC, and acceleration.

□ **Output Visualization:**

- Render simulation results and dashboards.

[**Note:** The *Highway Lane Change* project leverages several pre-defined MATLAB toolboxes essential for automated driving systems. Key toolboxes include the **Automated Driving Toolbox** for designing, simulating, and testing ADAS features, and the **Vehicle Dynamics Blockset** for modeling vehicle motion. The **Model Predictive Control Toolbox** assists in implementing advanced control strategies for smooth lane changes. Other important toolboxes are **Simulink**, which provides a dynamic environment for system simulation, and the **Optimization Toolbox**, used for solving trajectory cost functions. For sensor integration and obstacle detection, the **Sensor Fusion and Tracking Toolbox** is applied, while the **MATLAB Coder** enables deployment by generating C/C++ code from simulations. These toolboxes streamline system design and testing for this complex application.]

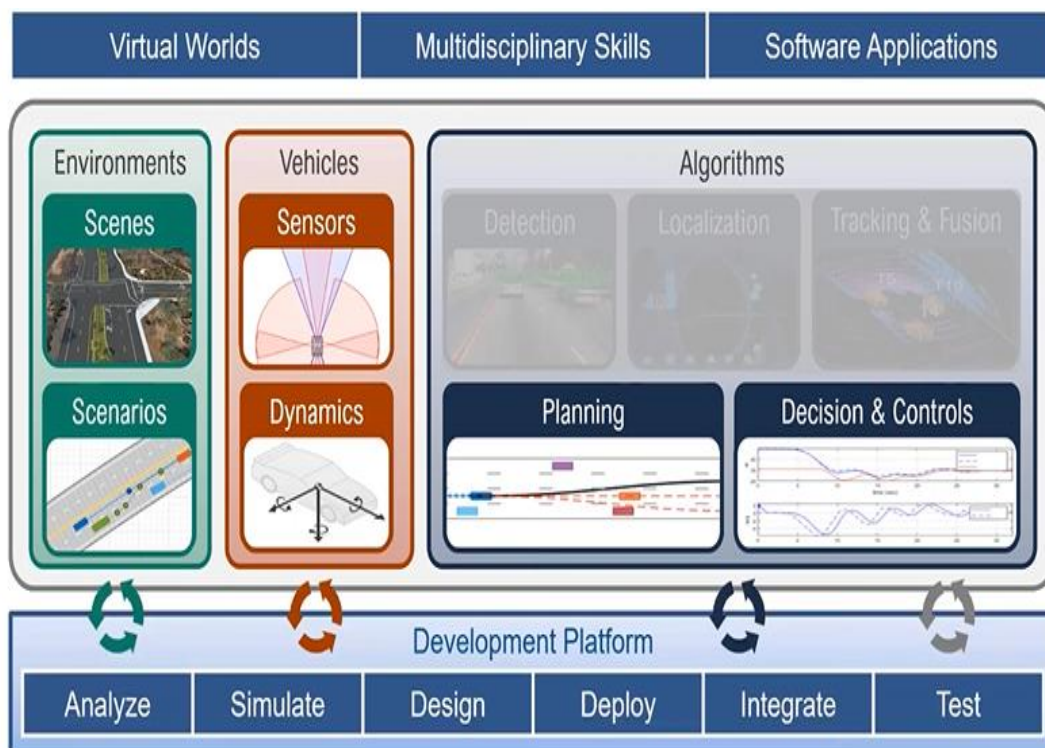


Fig 3: Toolboxes and Development platforms



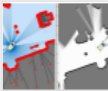





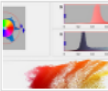



Name	Author
 Sensor Fusion and Tracking Toolbox version 2.5	 MathWorks
 Navigation Toolbox version 2.4	 MathWorks
 Model Predictive Control Toolbox version 8.1	 MathWorks
 Computer Vision Toolbox version 10.4	 MathWorks
 Image Processing Toolbox version 11.7	 MathWorks
 Automated Driving Toolbox version 3.7	 MathWorks

Fig 4: Inbuilt Toolboxes systems installed in MATLAB Software

The predefined toolboxes installed have inbuilt algorithms which helped to develop:

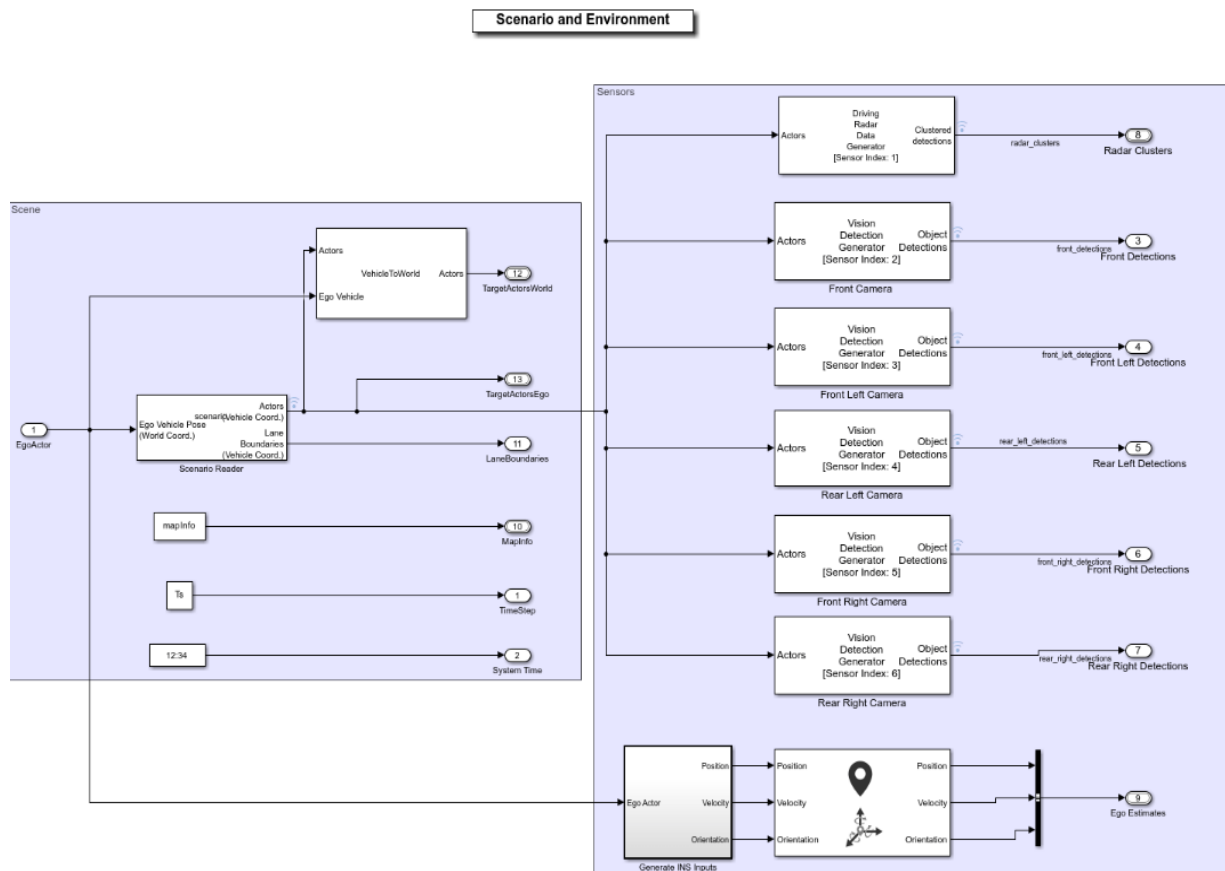
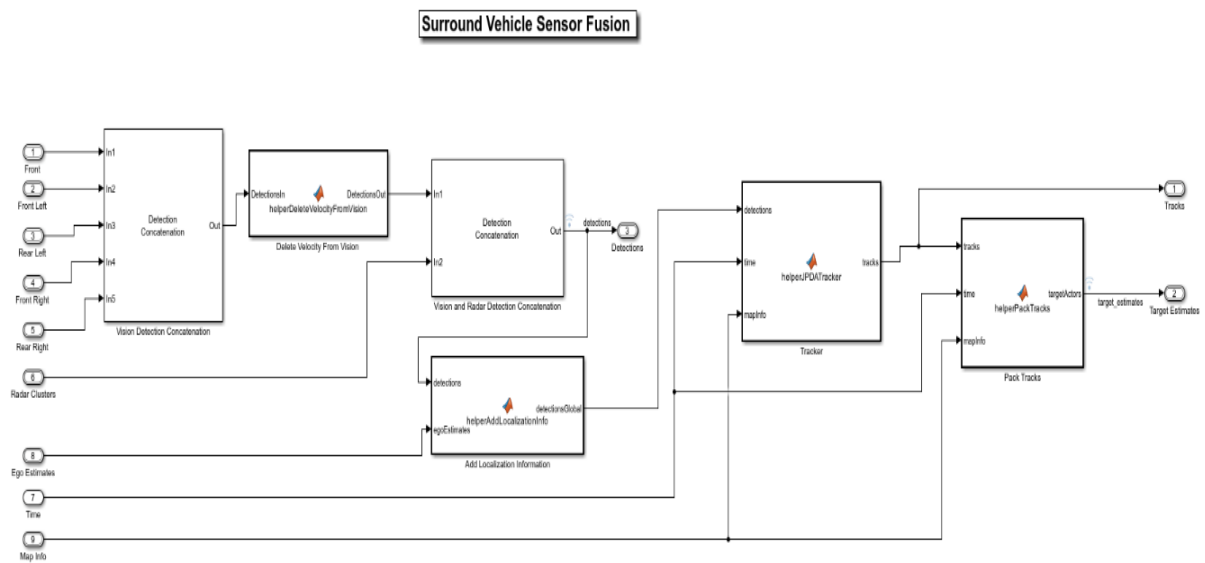
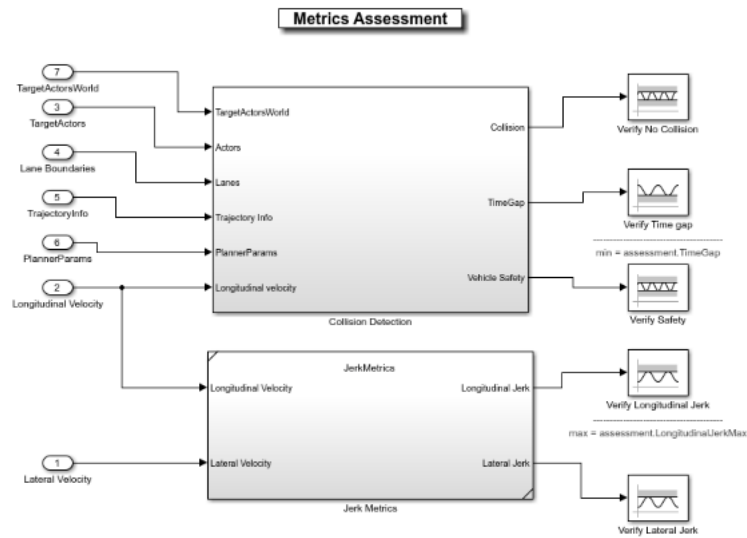
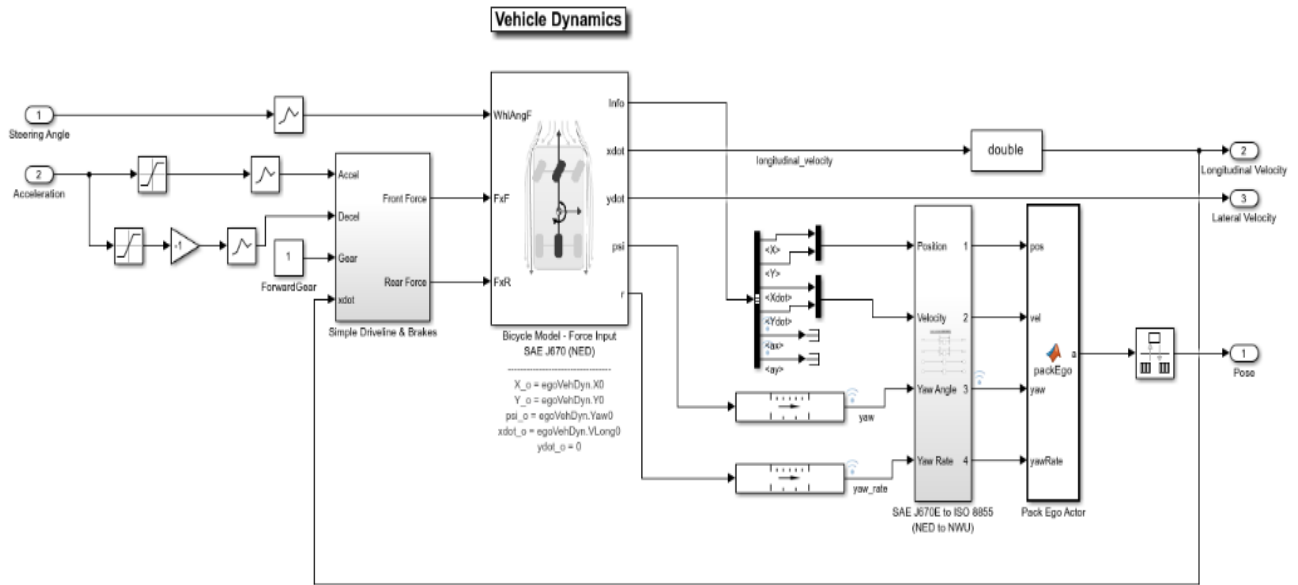
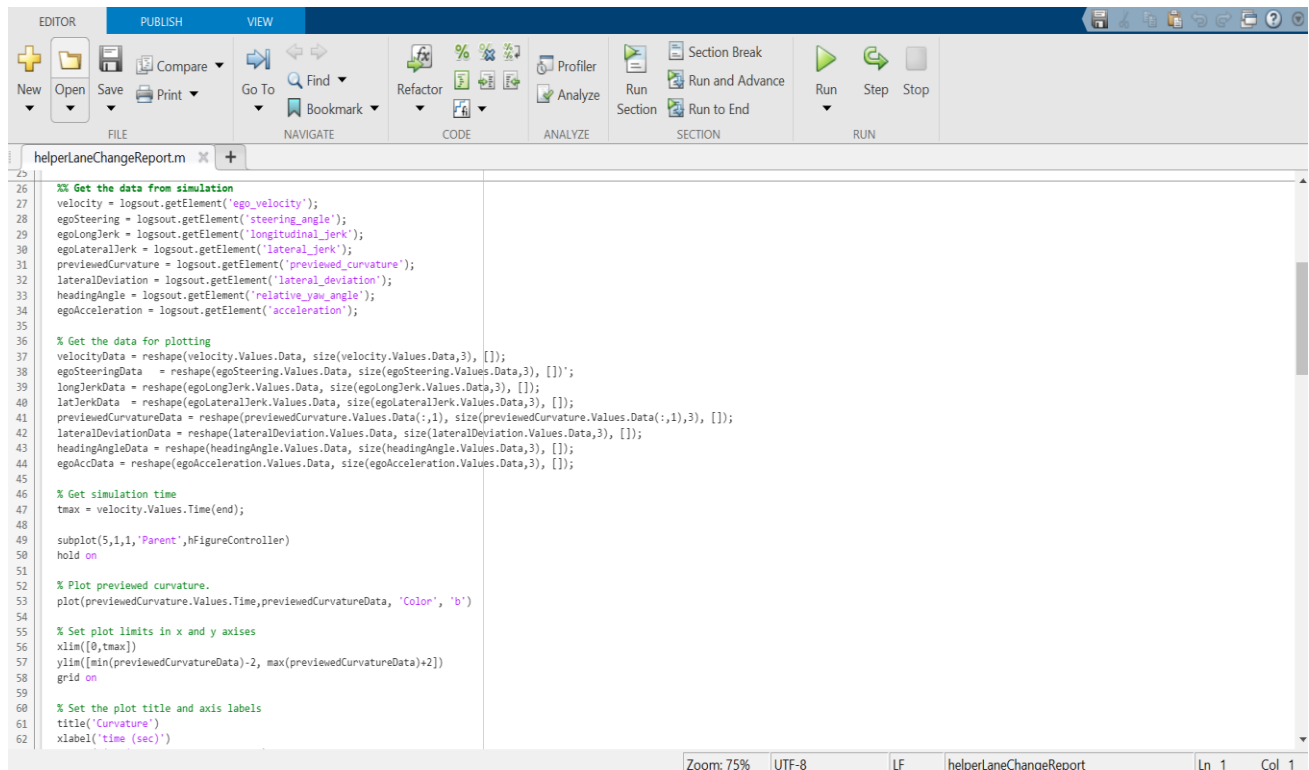


Fig 5: Scenario and Environment pre-processed using Inbuilt Toolbox features



Fig(s) 6,7,8: Simulink Modals Generated using Toolboxes features

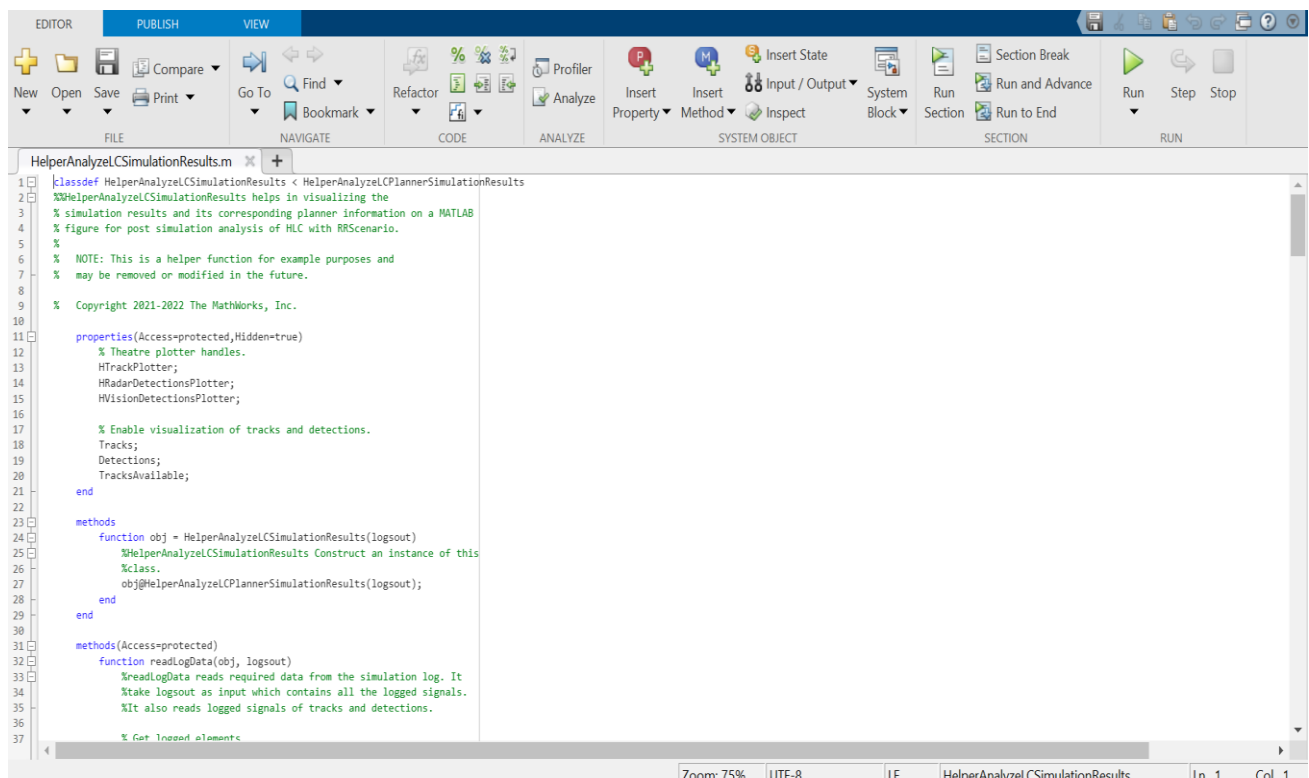
PORTION OF MATLAB CODE USED FOR PROJECT SCREENSHOT SNIPPETS:



The screenshot shows the MATLAB Editor window with the file `helperLaneChangeReport.m` open. The code is as follows:

```
26 % Get the data from simulation
27 velocity = logouts.getElement('ego_velocity');
28 egoSteering = logouts.getElement('steering_angle');
29 egoLongJerk = logouts.getElement('longitudinal_jerk');
30 egoLateralJerk = logouts.getElement('lateral_jerk');
31 previewedCurvature = logouts.getElement('previewed_curvature');
32 lateralDeviation = logouts.getElement('lateral_deviation');
33 headingAngle = logouts.getElement('relative_yaw_angle');
34 egoAcceleration = logouts.getElement('acceleration');
35
36 % Get the data for plotting
37 velocityData = reshape(velocity.Values.Data, size(velocity.Values.Data,3), []);
38 egoSteeringData = reshape(egoSteering.Values.Data, size(egoSteering.Values.Data,3), []);
39 longJerkData = reshape(egoLongJerk.Values.Data, size(egoLongJerk.Values.Data,3), []);
40 latJerkData = reshape(egoLateralJerk.Values.Data, size(egoLateralJerk.Values.Data,3), []);
41 previewedCurvatureData = reshape(previewedCurvature.Values.Data(:,1), size(previewedCurvature.Values.Data(:,1),3), []);
42 lateralDeviationData = reshape(lateralDeviation.Values.Data, size(lateralDeviation.Values.Data,3), []);
43 headingAngleData = reshape(headingAngle.Values.Data, size(headingAngle.Values.Data,3), []);
44 egoAccData = reshape(egoAcceleration.Values.Data, size(egoAcceleration.Values.Data,3), []);
45
46 % Get simulation time
47 tmax = velocity.Values.Time(end);
48
49 subplot(5,1,1,'Parent',hFigureController)
50 hold on
51
52 % Plot previewed curvature.
53 plot(previewedCurvature.Values.Time,previewedCurvatureData, 'Color', 'b')
54
55 % Set plot limits in x and y axes
56 xlim([0,tmax])
57 ylim([min(previewedCurvatureData)-2, max(previewedCurvatureData)+2])
58 grid on
59
60 % Set the plot title and axis labels
61 title('Curvature')
62 xlabel('time (sec)')
```

The status bar at the bottom indicates: Zoom: 75% | UTF-8 | LF | helperLaneChangeReport | Ln 1 Col 1.



The screenshot shows the MATLAB Editor window with the file `HelperAnalyzeLCSimulationResults.m` open. The code is as follows:

```
1 classdef HelperAnalyzeLCSimulationResults < HelperAnalyzeCPlannerSimulationResults
2 %HelperAnalyzeLCSimulationResults helps in visualizing the
3 % simulation results and its corresponding planner information on a MATLAB
4 % figure for post simulation analysis of HLC with RRScenario.
5 %
6 % NOTE: This is a helper function for example purposes and
7 % may be removed or modified in the future.
8 %
9 % Copyright 2021-2022 The MathWorks, Inc.
10
11 properties(Access=protected,Hidden=true)
12 % Theatre plotter handles.
13 HTrackPlotter;
14 HRadarDetectionsPlotter;
15 HVisionDetectionsPlotter;
16
17 % Enable visualization of tracks and detections.
18 Tracks;
19 Detections;
20 TracksAvailable;
21 end
22
23 methods
24 function obj = HelperAnalyzeLCSimulationResults(logouts)
25 %HelperAnalyzeLCSimulationResults Construct an instance of this
26 %class.
27 obj@HelperAnalyzeCPlannerSimulationResults(logouts);
28 end
29 end
30
31 methods(Access=protected)
32 function readLogData(obj, logouts)
33 %readLogData reads required data from the simulation log. It
34 %take logouts as input which contains all the logged signals.
35 %It also reads logged signals of tracks and detections.
36
37 % Get Inroad elements
```

The status bar at the bottom indicates: Zoom: 75% | UTF-8 | LF | HelperAnalyzeLCSimulationResults | Ln 1 Col 1.

[NOTE: Some of the files are related to automated testing of the Highway Lane Change test bench model. Other files would need paid access to Simulink Requirements and Simulink Test, then these files can be used to run the test scenarios in an automated way.]

```

300 function [posScene] = calculatePositionInScenarioFrame(obj,detections, egoVehicle)
301
302 % Calculate Cartesian positions for all detections in the "sensor"
303 % coordinate frame
304 allDets = detections;
305 meas = horzcat(allDets.Measurement);
306
307 if strcmpi(allDets(1).MeasurementParameters(1).Frame,'Spherical')
308     az = meas(1,:);
309     r = meas(2,:);
310     e1 = zeros(1,numel(az));
311     [x, y, z] = sph2cart(deg2rad(az),deg2rad(e1),r);
312     posSensor = [x;y;z];
313     rr = meas(3,:);
314     rVec = posSensor./sqrt(dot(posSensor,posSensor,1));
315     velSensor = rr.*rVec;
316 else
317     posSensor = meas;
318     velSensor = zeros(3,size(meas,2));
319 end
320
321 % Transform parameters
322 sensorToEgo = detections(1).MeasurementParameters(1);
323 R = sensorToEgo.Orientation;
324 T = sensorToEgo.OriginPosition;
325 if isfield(sensorToEgo,'OriginVelocity')
326     Tdot = sensorToEgo.OriginVelocity;
327 else
328     Tdot = zeros(3,1);
329 end
330
331 if isfield(sensorToEgo,'IsParentToChild') && sensorToEgo.IsParentToChild
332     R = R';
333 end
334
335 % Position, velocity in ego frame
336 posEgo = T + R*posSensor;
337 velEgo = Tdot + R*velSensor;

```

Zoom: 75% UTF-8 LF HelperPlotLCStatus Ln 1 Col 1

```

21 'BusDrivingRadarDataGeneratorDetections',...
22 'BusDrivingRadarDataGeneratorDetectionsMeasurementParameters',...
23 'BusEgoPose',...
24 'BusObjectDetections1',...
25 'BusObjectDetections1Detections',...
26 'BusRadarDetectionGeneratorDetectionsObjectAttributes',...
27 'BusTracker',...
28 'BusTrackerTracks',...
29 'BusTrackerJPDA',...
30 'BusTrackerJPDATracks',...
31 'BusTrackerTracks',...
32 'BusTruth',...
33 'BusVision',...
34 'BusVisionDetections',...
35 'BusVisionDetectionsObjectAttributes',...
36 'BusVisionDetectionsMeasurementParameters');
37
38 clear actorProfiles;
39 clear camera;
40 clear egoVehicle;
41 clear M;
42 clear N;
43 clear numCoasts;
44 clear numSensors;
45 clear numTargets;
46 clear numTracks;
47 clear P;
48 clear Q;
49 clear radar;
50 clear tout;
51 clear assignThreshold;
52 clear trackAssignmentThreshold;
53 clear trackRegisterThreshold;
54
55 function clearBuses(buses)
56     matlabshared_tracking_internal_dynamicBusUtilities.removeDefinition(buses);
57 end
58

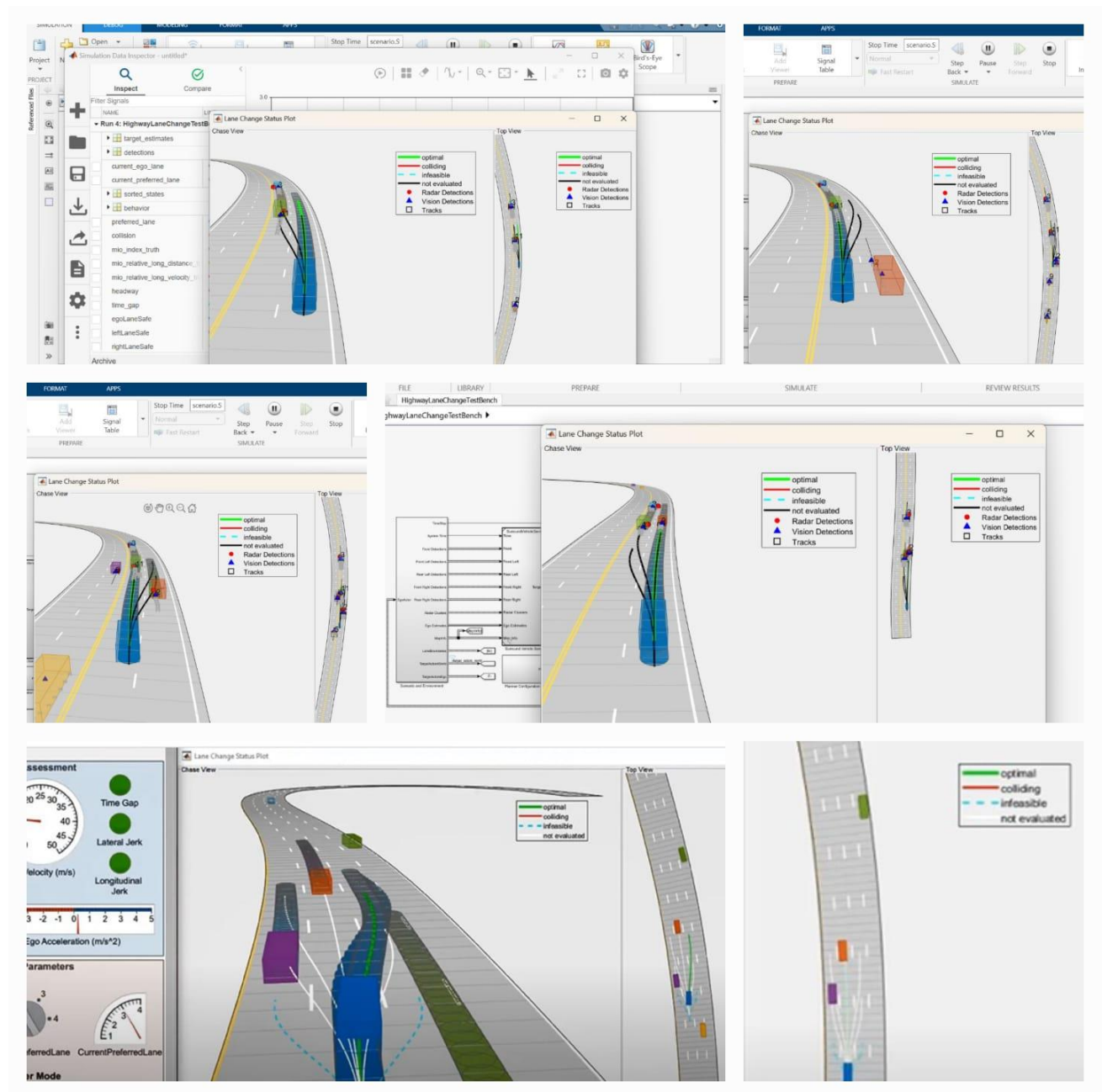
```

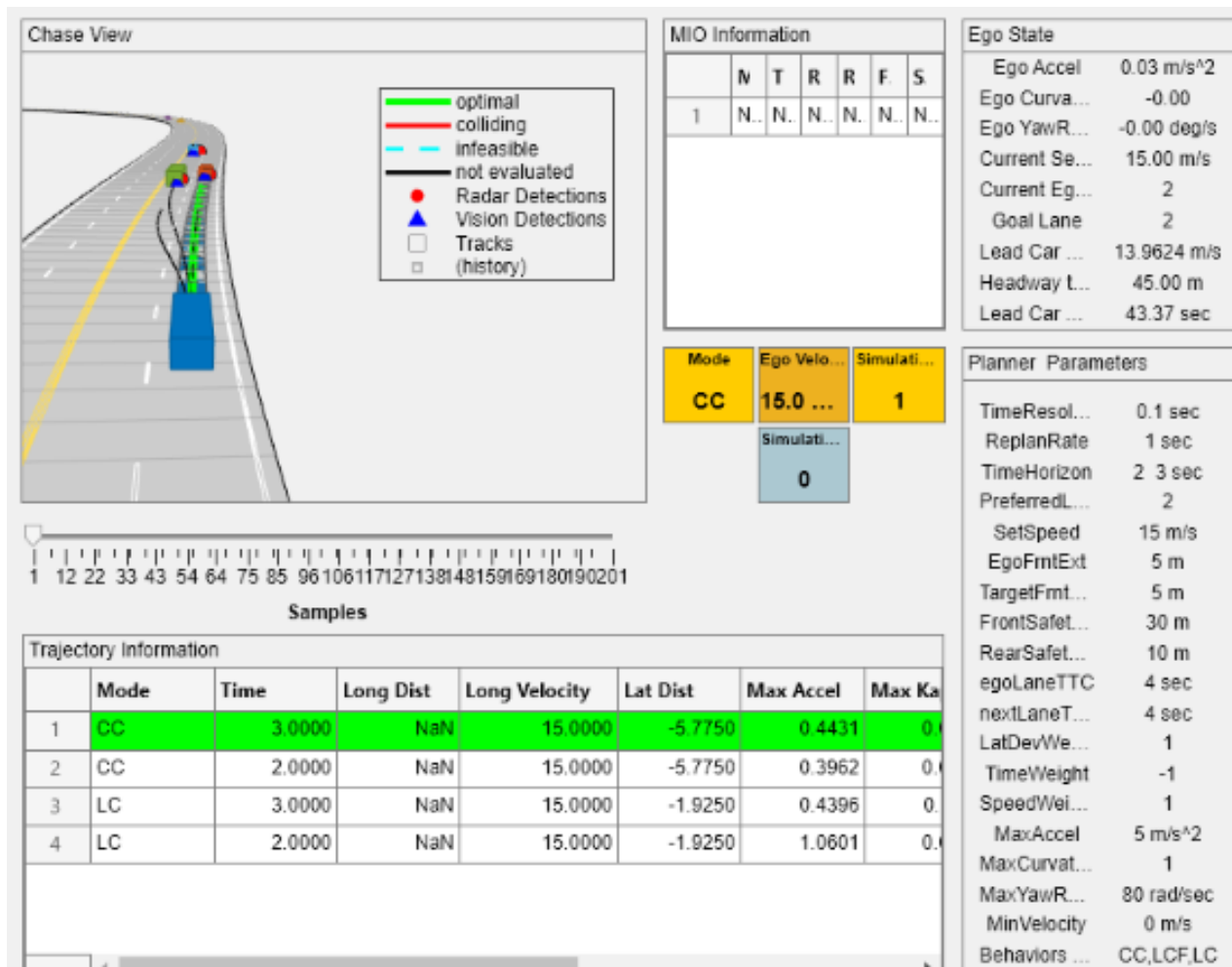
Zoom: 75% UTF-8 LF script Ln 1 Col 1

Fig(s) 9,10,11,12: MATLAB Code snippets of files used for Project

TESTING(S) AND SIMULATION: Here are some screenshots of simulations I ran in Interface:

Fig 13, 14: Snippets of Simulation and Visualization Output





Analysis of Parameter Variations:

1. Internal Parameters

- Track Width: Wider vehicles may face difficulties in sharp turns.
 - Observation: Larger track widths increase lateral jerk.
- Wheelbase: Affects turning radius and stability.
 - Observation: Longer wheelbases reduce lateral acceleration.
- Center of Gravity (CG): Influences roll and yaw stability.
 - Observation: Higher CGs lead to increased rollover risk.

2. External Parameters

- Friction Coefficient: Affects tire grip and braking.
 - Observation: Low friction causes longer braking distances and unstable turns.
- Vehicle Mass: Impacts acceleration and stopping distance.
 - Observation: Heavier vehicles exhibit slower acceleration.
- Steering Angle: Determines the turning radius.
 - Observation: Larger angles can induce oversteer.
- Radius of Curvature: Affects turn sharpness.
 - Observation: Smaller radii increase lateral acceleration and jerk.

Simulation Insights:

Metrics and Dashboard Outputs

- Collision Detection: Halts simulation upon detecting a collision.
- Time-to-Collision (TTC): Visualized during lane changes.
- Lateral and Longitudinal Jerk: Monitored for passenger comfort.
- Trajectory Visualization: Displays reference and actual paths.

Scenario Variations

- Straight Roads: Baseline for lane change performance.
- Curved Roads: Test lateral control and curvature following.
- Obstacle Detection: Includes parked and moving vehicles.

Conclusion: This project demonstrates a robust Highway Lane Change system, emphasizing the synergy between simulation, planning, and control. The insights derived from parameter variations and metrics assessment ensure safety and efficiency. By leveraging MATLAB and Simulink, the system serves as a blueprint for further development in automated driving systems.

References:

[1] <https://www.youtube.com/watch?v=DWlyVmIfiRQ>

[2] Wang, X., Yu, H. and Liu, J., 2022, October. Research on Local Path Planning Algorithm Based on Frenet Coordinate System. In 2022 4th International Conference on Machine Learning, Big Data and Business Intelligence (MLBDBI) (pp. 249-252). IEEE.

[3] Silenko, A.J., 2015. Comparison of spin dynamics in the cylindrical and Frenet-Serret coordinate systems. Physics of Particles and Nuclei Letters, 12, pp.8-10.

[4] Huang, J., He, Z., Arakawa, Y. and Dawton, B., 2023. Trajectory planning in Frenet frame via multi-objective optimization. IEEE Access.

[5] Wei, M., Teng, D. and Wu, S., 2021. Trajectory planning and optimization algorithm for automated driving based on Frenet coordinate system. Control Decis, 36(4), pp.815-824.

Signature of the faculty member