Behaviour Planning for Autonomous Vehicles

Nagarjun Vinukonda
nvinukonda@wpi.edu

***Abstract*—This paper proposes an approach of safe navigation while Lane changing in road case scenario. The approach is to build a cost based decision making module for lane changing task with multi agents for avoiding collisions. The cost libraries used in this model helps to make decision of when to change a Lane. The project is built on modelling exiting Lane Following and Highway Lane change MATLAB models. The paper describes three models of the project. Model-1 is working method of simulation implementation. Model-2 is method of creating decision making module using safe zones and Model-3 is proposed method of creating decision making module using cost functions.**

***Index Terms*—Simulink, MATLAB, Lanes, sensors, EGO vehicles, MIOs.**

I. Introduction

The safe navigation for autonomous vehicles depends on many factors when comes to planning. In general planning framework can be divided into Motion planning, Mission planning, behaviour planning. Motion planning generates desired trajectory of the vehicle considering the dynamic parameters and output of steering and throttle. Mission planning is to optimizes the path to achieve different checkpoints considering the arrival time, distance or different required maneuvers. Behaviors planning makes tactical driving decisions about such things as distance keeping, Lane changing and neighbouring vehicle interactions [1].

The aim of this project is to build a decision making module based on cost functions[2]. The project is modified based on reference Highway Lane change [3].

In order to create a Autonomous Lane change model. First we need to understand how Lateral[4] and Longitudinal control [5] is built, which is required to create Traffic Jam assist [6]. The reference Model created by MathWorks for Traffic Jam assist is also known as Lane Following Control with Sensor Fusion and Lane Detection [7]. Which is modelled further with Lane change controller [8] to design a highway Lane change [9].

The paper uses words like EGO which are vehicles that are sensor equipped, MIO(Most Important Object) which are other EGO vehicles around our EGO vehicle. The MIO motion prediction is based on constant speed model. Both the Models 1&2 require to create a driving a case scenario using MATLAB driving scenario app [10]. The framework of Model 2 is as shown in Fig.(1).

My work in Model-1 is related to Vehicle&Environment, sensor fusion and behaviour planning.

II. Literature Survey

Over past years, ADAS systems and autonomous vehicles have been used to support human driving tasks, especially in order to improve safety and convenience in navigation.

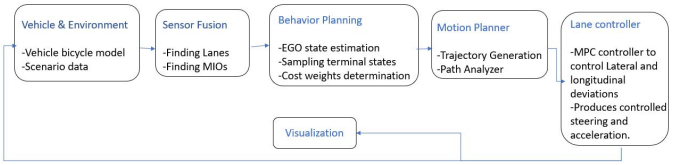

Fig. 1: Frame work of Model-1

Many researchers have proposed different architectures for planning and controls. [11] has proposed an offline learning mechanism to emulate human driver behaviours. This approach is on prediction based cost functions which is trained with different traffic scenarios to determine optimal cost weights that predict human behaviour cost functions.

Learning algorithms such as Artificial Neural Networks (ANN) have also been implemented in lane keeping and adaptive cruise control. The major constraint of learning algorithms is that they depend too much on training. Therefore, no safety criteria can be verified. They also restrict the autonomous vehicle's ability to exceed a human driver's performance.

The paper [12] has proposed a real time decision making module based on human driving behaviour characteristics, which depends on lawfulness, fear and patience. These parameters used to evaluate when to change a selected target lane and mode selection of maneuvering on lane.

Another author [13] has proposed a hierarchical architecture to enable autonomous vehicles to finish long-term missions and reduce the workload of motion planning. For each layer of the architecture, the input higher-level mission is decomposed into sub-missions and passed on to the next-lower level. One of the shortcoming is that the planner problem becomes very complicated and computationally expensive as most information is processed in the motion planner, including road geometry, vehicle dynamics, and surrounding moving objects and static obstacles. This makes planner to sacrifice its performance to work on real-time constraints.

Similarly, [14] proposed a novel approach of hierarchical architecture that considers social cooperation between the autonomous vehicle and surrounding cars. It introduced a layer called reference planning which generates kinematically and dynamically feasible paths assuming no obstacles on the road, then a behavioral planning layer takes static and dynamic obstacles into account. The paper has provided experimental results which has improved both simulation and real autonomous vehicle platform driving quality considerably.

Further, few authors [15] , [16] have introduced a robust

prediction-cost function based algorithm for autonomous freeway driving. The prediction engine is built so that the autonomous vehicle is able to estimate human drivers' intentions. And the cost function library is used to help behavior planners generate the best strategies of maneuvering on road. My proposed work is based on the following authors to create the cost library that helps to make decisions for Lane changing.

III. Methodology

These are the common methods that can be used for all three Models.

A. Driving Scenario:

The driving scenario is created with 8 radars and 1 camera sensor. The radar sensors are named as: 1) FrontLRR 2)FrontMRR 3)LeftFrontSRR 4)LeftRearMRR 5)LeftRearSRR 6)RightFrontSRR 7)RightRearMRR 8)RightRearSRR. The SRR stands for Short-Range, MRR for Mid-Ranged, LRR for Long-Range. Each has its properties and the complete sensor addition is as shown in Fig.(2).

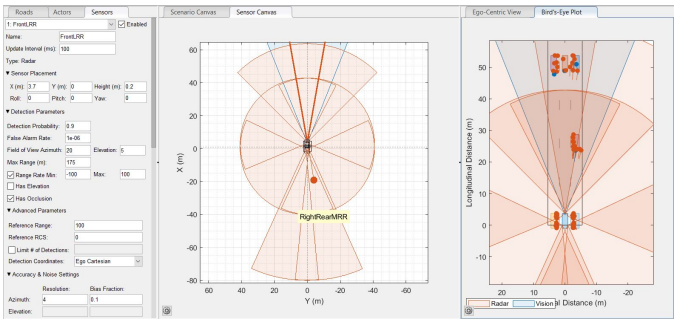

Fig. 2: Sensors block

After Addition of sensors we create a the road case scenario by adding reference waypoints to our model and other actors. Which results in 3D simulation in birds eye plot as shown in Fig.(3).

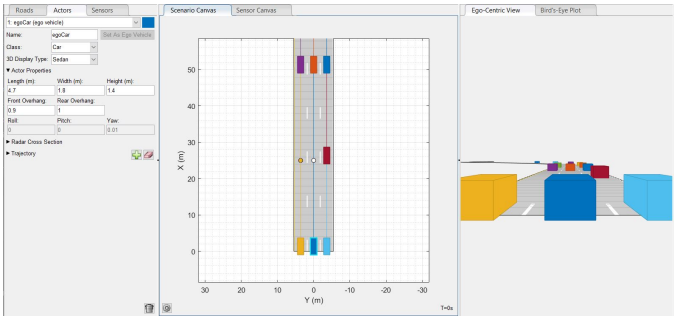

Fig. 3: Sensors block

After creating the scenario we need to export it into MATLAB function. The driving case scenario cant provide reference signals. It provides actors profiles which includes pose, velocity, index. We can generate our reference signals which will be required for MPC controller further using "helperCreateReferencePath.m" file provided by MATLAB. Later, we need to send our reference signals from MATLAB editor to Simulink. In our I am importing signals as a constant called "globalplanpoints" as shown in Fig.(5).

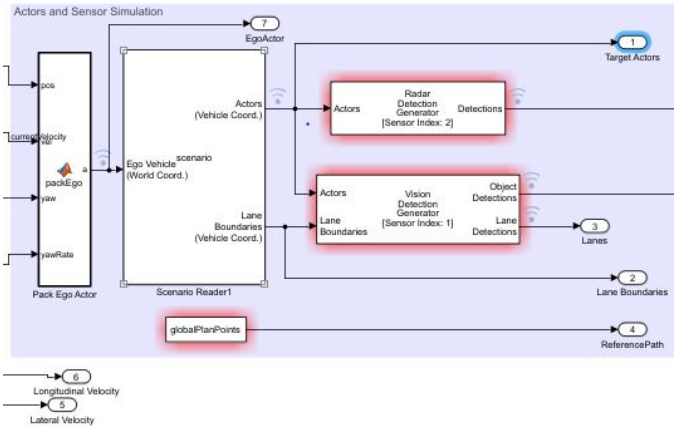

Fig. 4: Vehicle and Environment module

B. Finding MIOs:

In order to find MIOs, we first need to find other lanes around EGO lane. For which we use MATLAB function called "PackLanes.m" which provides curvatures of EGO lane and its next lane. Next we need to build a function to find MIOs. This can be either done by calculating Safe Zones or calculating time of contact(TTC). In this model I have used Safe Zone Calculations. The safe zone is distance for which our EGO vehicle wouldn't be in contact with other actors. This is calculated with the formula as stated in eq(1). v_f is relative velocity of forward vehicle, ρ is response time of EGO vechile to perform any activity, a_b is brake deceleration of our EGO car. As stated previously MIOs are considered to move in constant velocities hence, we take deceleration as 0.4*a_g i.e. a_g is 9.8 m/s^2.

$$v_f * \rho + v_f^2/(2 * a_b) \tag{1}$$

This formula is applied for every MIO which is available in our lane or next Lane. While calculating safe zones, we require to state whether calculated distance is safe or not. For which we introduce a parameter called FCW(Forward Collision Warning). While detecting MIOs we need to state FCW=3 if vechile ahead is having more than safe distance from our EGO vehicle. FCW=2, if relative speed is decreasing and FCW=1, if a MIO is detected inside our safe zone.

C. Pseudo code:

The Algorithm-1 is pseudocode to calculate MIOs. Instead of safe zones we can also calculate TTC in similar way through taking relative distance with speed.

IV. Model-1

This is the model of working simulation.

Algorithm 1: General implementation of finding MIOs

Identify the lane curvatures of our lane and next lane.
Initialize EGO parameters like Min and Max position EGO can reach.
Initialize MIOs.
for Every Target actors found from sensor data: **do**
  Find its lateral and longitudinal positions relative to our EGO vehicle.
  Find lateral positions of left and right EGO lane.
  if MIO lateral and longitudinal positions lay within Lane boundaries **then**
    Store that MIO track index.
  end if
end for
if The MIO track index>0 (i.e. MIO is detected) **then**
  Store its position and velocity
  Calculate Safe Zone or Time of contact
end if
Store the data in MIO class.

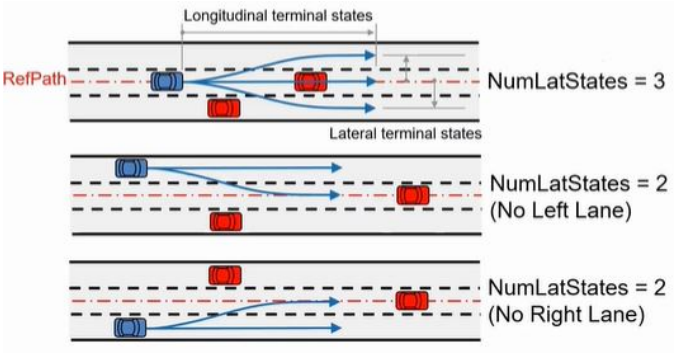

Fig. 5: EGO vehicle while changing the Lane

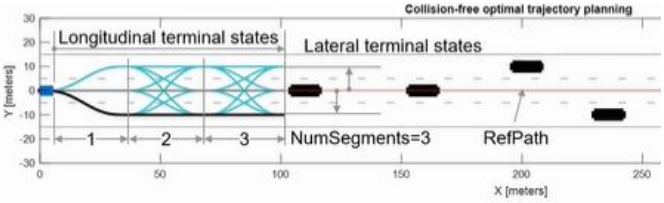

Fig. 6: Lateral and Longitudinal samples

A. Behaviour Planner

:

The behaviour planner consists of estimating EGO vehicle state, sampling terminal states and defining cost weights for EGO vehicle.

The terminal state will define sampling behaviour for generating multiple candidate trajectories. These terminal states will be used to divide the longitudinal state into more complex trajectory. The feasibility parameters are required to avoid excessive curvatures and accelerations. First we need to sample multiple candidate trajectories for each pair of state (start, terminal states). Evaluate cost for each of these trajectories which depends upon time, Arc length, deviation, Lateral/Longitudinal smoothness[17].

Followed by feasibility check for all these trajectories and then collision check for every trajectory using state validator. Finally, with these results we can choose feasible collision free trajectory with minimal cost. Where these two tasks are done in Motion Planner.

In order to do this first we need to create occupancy map using state validator. The state validator creates a "validatorOccupancyMap" built by MATLAB. In my scenario, I have provided binaryOccupancyMap as 100*75 which is different from default. I also require to provide my Lanewidth which is 4.5m. The occupancy map is useful to calculate if there is left and right lane available after lane change and finding no. of lanes available to change based on the trajectory generated by reference path as shown in Fig.(5)

We also require to calculate deviation of our EGO vehicle both lateral and longitudinal from reference path. Next we need to calculate no.of lateral and longitudinal terminal states samples as shown in Fig.(6).

In the calculation of Longitudinal Samples I have taken decreased my Planning Horizon to 45m as my scenario is having more curved roads and the longitudinal planning requires the road to be straight. I have increased no.of longitudinal sample to 2 as my scenario has less no.of cars and I don't require to maneuver more to get complex trajectories.

Similarly, in lateral sampling I require to change my Lane width. As no. of Lanes in my scenario are three, I dont require to change the sampling process. But, if there is a scenario where no. of Lanes more than 3 or we want to deviate two lanes at once, we require to calculate lateral states.

The velocity sampler is intended to consider lane speed velocity profile generated for the reference path while generating trajectory. I have changed the ego set velocity to $20m/s$. There is no use of using acceleration sampler since, we are using constant velocity model, so I have discarded it and provided an empty array to avoid index error. Further I have changed the weights of the costs for time, ARClength, lateralsmoothness to 0.2, 0.2, 0.4 respectively. The following is the Module diagram of behaviour planner as shown in Fig.(7).

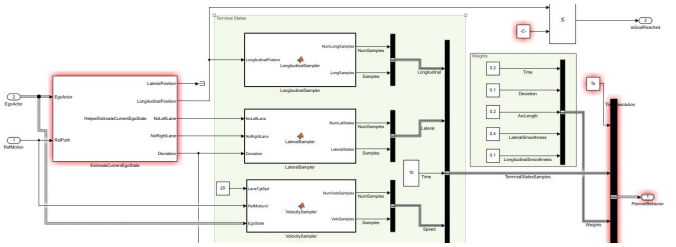

Fig. 7: Behaviour Planner Module

For my working model I have used Motion planner and Lane change controller provided by MATLAB. And the following is the result of my scenario as shown in Fig.(8).

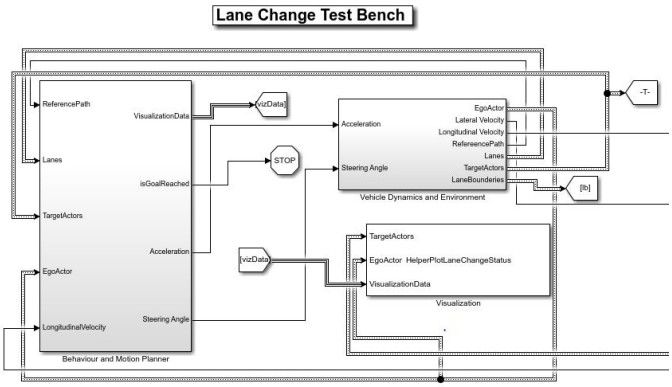

Fig. 8: Model-1 result and Module

B. Model-1 Results:

The following are the results of simulation Model-1 Fig.(11). The Fig.(9) shows lateral offset provided by our reference vs lateral offset after tuned by MPC controller. Fig.(10) shows the heading angle deviation. Heading angle is angle between EGO heading direction and Longitudinal axis. Although in video there are unexpected sharp turns the result are good. The reason for these sharp turns might be due to cost weights provided to lateral and longitudinal smoothness, planning horizon and no.of samples created.

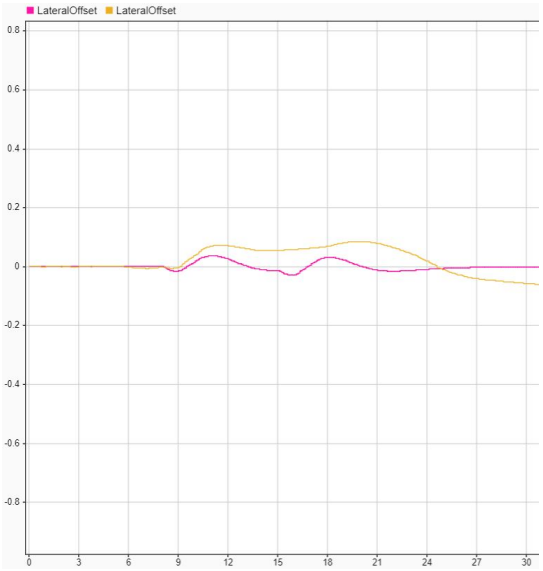

Fig. 9: Later offset, yellow: reference provided and pink: MPC tuned lateral offset

V. Model-2

This model is made from Lane Following with sensor model from MATLAB, which is been modified to our customs to create a state flow diagram for decision making module [18]. The following is the Model-2 Fig.(12).

We Follow the same steps for building vehicle Environment, sensor fusion and finding MIOs. After finding MIOs we create

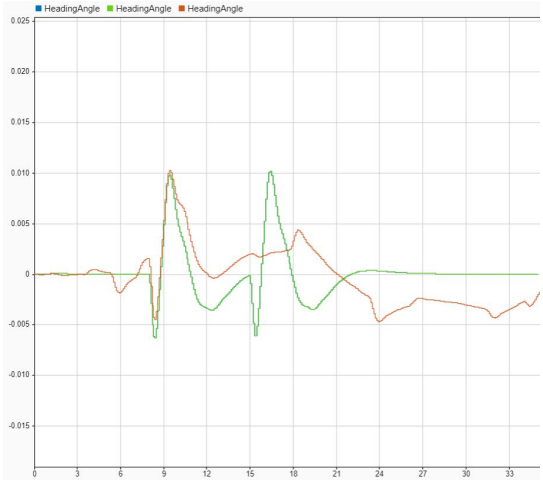

Fig. 10: Heading, red: reference provided and green and blue: MPC tuned heading angle and its next iteration respectively.

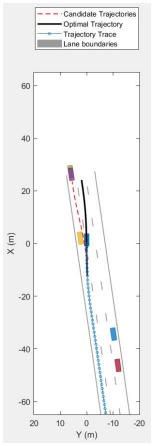

Fig. 11: Lanechange result

a Lane change planner which requires a Decision making module as shown in Fig.(13) state flow diagram.

Inside the state flow diagram we have created the decision making Logic as follows Fig(15). Here the data, data1, data2, data3, data4, data5 are MIO.Rear, MIO.LeftFront, MIO.LeftRear, MIO.RightFront, MIO.RightRear respectively. LCActive is state of Lane change mode whether it is active or inactive. This is the input from Lane change Planner which consists trajectory generator. The duration is Time of contact. Duration(FCW) is a function which determines TTC with MIO in current EGO Lane. LCParam is either 1 or 2 for left and right lane respectively.

First we check if there are vehicles around us. If there are no MIOs available in LeftFront and LeftRear then we change Left else we change Right. A trigger message is set. When LCTrigger is false then we start to change the lane. Finally, the LCstatus and LCParam are provided as output for the Lane change Planner as shown in Fig.(14). This planner calculates reference trajectory for changing lane, Lane centers and velocity of EGO after Lane change. Which are further fed

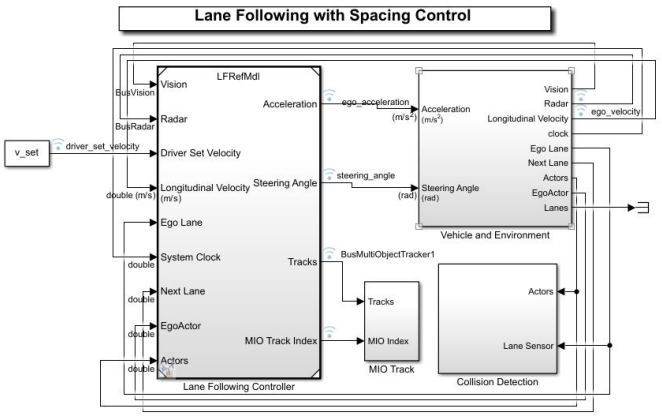

Fig. 12: Model-2 framework overview

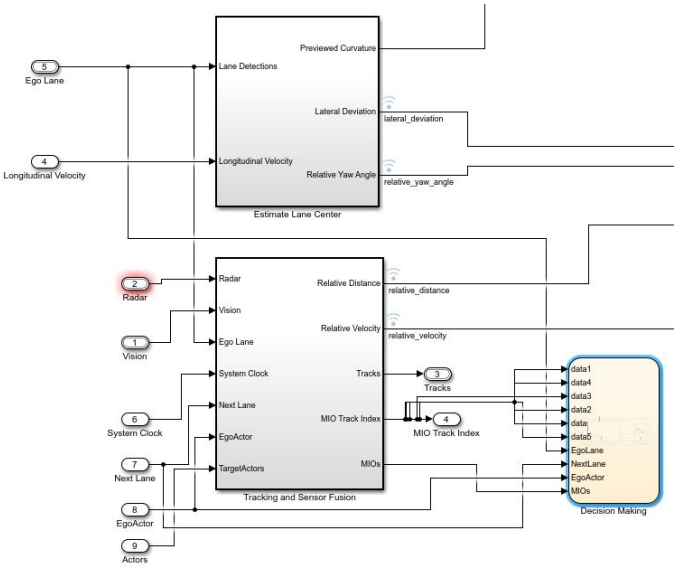

Fig. 13: State Flow diagram framework

into Lane change MPC controller to provide steering angle and acceleration.

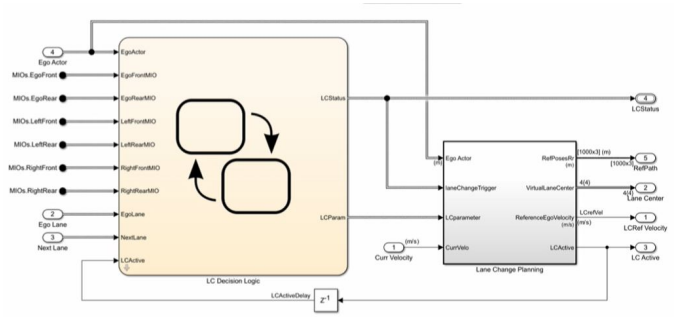

Fig. 14: Overview of Lane Change planner

VI. Model-3

The Model-3 is my proposed work for adding the cost functions to make a decision making module. The cost library

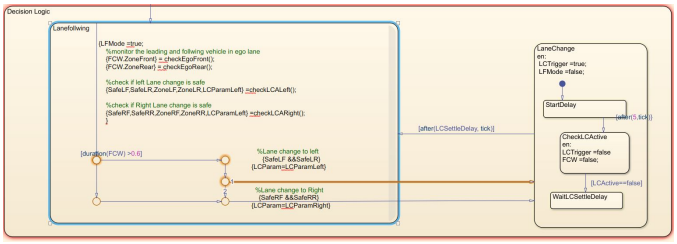

Fig. 15: Decision Making Logic

consists of Gap error, Clear Distance, Relative Velocity. In this model not only we want to lane change by avoiding the MIOs around us but also maintain the gap with them. In this model instead of calculating safety zones we calculate TTC to determine the relative speeds between EGO vehicles. We assume our desired distance for EGO to maintain with MIO is $8m$.

1) ***Gap Error:*** The gap error cost as shown in Fig.16 is mainly used in keeping a desired distance while following a lead vehicle. When the gap error is smaller than zero, which means the current distance to the lead vehicle is smaller than desired, the cost increases significantly.
2) ***Clear Distance:*** The clear distance cost as shown in Fig.17 penalizes moving too close to surrounding vehicles. It is set to zero when all other vehicles are above safe distances.
3) ***Velocity Differences:*** Since, we are taking constant velocity MIOs, if the detected MIO relative velocity is lower than our EGO, the cost increases as shown in Fig.18.

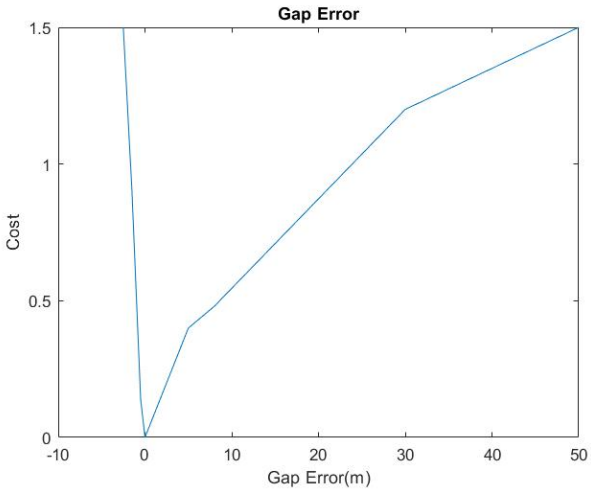

Fig. 16: Gap Error

The equations for the cost functions are as follows 2, 3:

$$C_scen = \mu_1 * C_Gap + \mu_2 * (C_Clear) + \mu_3 * (C_RelVel) \quad (2)$$

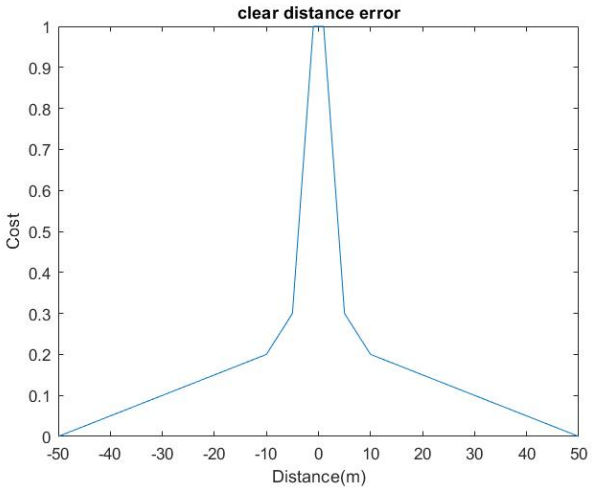

Fig. 17: Clear Distance

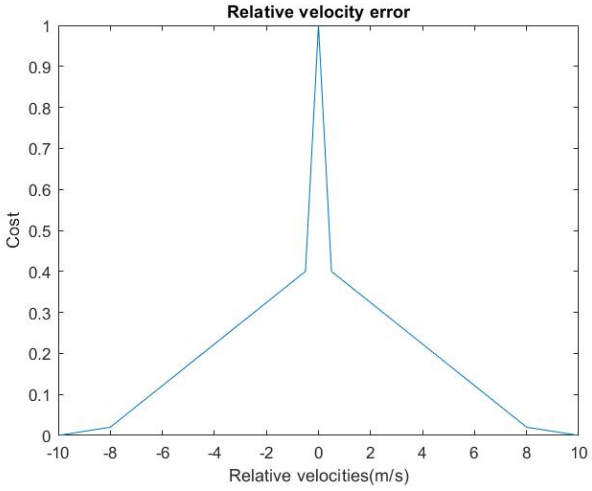

Fig. 18: Relative Velocities

$$C_total = \sum C_scen/n \tag{3}$$

C_scen is total cost of a scenario, μ_1,μ_2, μ_3 cost are weights, n is no. of scenarios, C_total is average total cost evaluated for all scenarios.

We are require to tune our cost weights and for calculated C_scen if the cost is greater than threshold C_total then we trigger the Lane change in the decision Module.

The cost weights are taken based on trail and error method. The following table shows the values taken during experiment Table.I.

The experiment 1,2 lead to collision of EGO vehicle with MIO and exp 3 lead to No lane change even though its necessary. Exp 4 provided Lane change without collision but maintaining the gap between EGO and MIO very small. Exp5 is able to change lane without collision but its leads to collision at end of scenario, which is neglected. We considered threshold for scenario as 65.

Exp.No	μ_1	μ_2	μ_3	scenario
1	20	60	40	Collision
2	10	30	20	Collision
3	6	30	70	No Lane change
4	5	40	10	Success
5	30	15	20	Unsuccessful

TABLE I: The cost weights considered

A. Decision Making Module:

The decision making module is similar to Model-2 where we use Stateflow chart with modifications of adding cost functions, weights and Motion metrics. The cost functions imported into simulink using variables for workspace and weights are provides as constants in simulink model. The motion metrics used are to check the collision when running the model. The decision making module in state flow chart is designed as shown in pseudo code. The sub pseudo codes are available in Appendix.

Algorithm 2: Cost-Based decision making module

```
Initialise isMIOs, isLCActive, isFCW,isCollsion as False
Get Lane Boundaries, centers and index of current Lane
Check if MIO is detected in current Lane
if isMIO=True then
    CheckFCW i.e. check forward collision
    if isFCW=True then
        Check if Left Lane change is safe using TTC
        if If safeLF & safeLR then
            Trigger_Lane_change()
            Check if Right Lane change is safe using TTC
        else if safeRF & safeRR then
            Trigger_Lane_change()
        else
            LCTrigger =True
            isLCActive = False
            Check_Collision()
        end if
    else
        return to check if MIO detected
    end if
end if
Send LCParam and LCstatus outputs
```

VII. Future Scope and Conclusion

In this project, I have implemented simulation setup for lane changing task for my created scenario(Model-1). The project requires comprehensive understanding of MATLAB and Simulink. The Model-2 requires to create a Lane change Planner Module as described previously which consists of Trajectory Generator, Path Analyser, Virtual Lane centers and Reference velocity of EGO calculator. In given amount of time I was able to create the state flow diagram for decision making logic. In future, to further continue Model-2 we require to create Lane change planner.

Further, Model-3 can also be implemented once Model-2 is completed to create a decision making module based on cost functions. For this project we used MATLAB Motion Planner for experimentation, which require further tuning for better results. We can further different costs like distance to goal,etc.

Later, the extension of this project is to export the MATLAB module into C++ and test it CARLA simulator with different scenarios.

REFERENCES

[1] J. Kim, K. Jo, D. Kim, K. Chu, and M. Sunwoo, "Behavior and path planning algorithm of autonomous vehicle a1 in structured environments," *IFAC Proceedings Volumes*, vol. 46, no. 10, pp. 36–41, 2013.

[2] J. Wei and J. M. Dolan, "A robust autonomous freeway driving algorithm," in *2009 IEEE Intelligent Vehicles Symposium*. IEEE, 2009, pp. 1015–1020.

[3] MathWorks, *Highway Lane Change*, https://www.mathworks.com/help/driving/ug/highway-lane-change.html.

[4] ——, *Lateral controller*, https://www.mathworks.com/help/driving/ug/lateral-control-tutorial.html.

[5] ——, *ACC controller*, https://www.mathworks.com/help/driving/ug/adaptive-cruise-control-with-sensor-fusion.html.

[6] M. Seo Woo Park, *Traffic-Jam assist*, https://www.mathworks.com/videos/design-and-test-traffic-jam-assist-a-case-study-1527496724717.html.

[7] MathWorks, *Lane Following Control with Sensor Fusion and Lane Detection*, https://www.mathworks.com/help/driving/ug/lane-following-control-with-sensor-fusion-and-lane-detection.html.

[8] ——, *Lane Change Assist Using Nonlinear Model Predictive Control*, https://www.mathworks.com/help/mpc/ug/lane-change-assist-using-nonlinear-model-predictive-control.html.

[9] M. Seo-Wook Park, *Design and Test Decision-Making, Path-Planning, and Control Modules in Traffic Scenarios*, https://www.mathworks.com/videos/design-and-test-decision-making-path-planning-and-control-modules-in-traffic-scenarios-1558962225433.html.

[10] MathWorks, *Driving Scenario Designer*, https://www.mathworks.com/videos/driving-scenario-designer-1529302116471.html.

[11] J. Wei, J. M. Dolan, and B. Litkouhi, "A learning-based autonomous driver: emulate human driver's intelligence in low-speed car following," in *Unattended Ground, Sea, and Air Sensor Technologies and Applications XII*, vol. 7693. International Society for Optics and Photonics, 2010, p. 76930L.

[12] H. Naseri, A. Nahvi, and F. S. N. Karan, "A real-time lane changing and line changing algorithm for driving simulators based on virtual driver behavior," *Journal of simulation*, vol. 11, no. 4, pp. 357–368, 2017.

[13] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke *et al.*, "Junior: The stanford entry in the urban challenge," *Journal of field Robotics*, vol. 25, no. 9, pp. 569–597, 2008.

[14] J. Wei, J. M. Snider, T. Gu, J. M. Dolan, and B. Litkouhi, "A behavioral planning framework for autonomous driving," in *2014 IEEE Intelligent Vehicles Symposium Proceedings*. IEEE, 2014, pp. 458–464.

[15] J. Wei and J. M. Dolan, "A robust autonomous freeway driving algorithm," in *2009 IEEE Intelligent Vehicles Symposium*. IEEE, 2009, pp. 1015–1020.

[16] J. Wei, J. M. Dolan, and B. Litkouhi, "A prediction-and cost function-based algorithm for robust autonomous freeway driving," in *2010 IEEE Intelligent Vehicles Symposium*. IEEE, 2010, pp. 512–517.

[17] M. Seo-Wook Park, *Developing Planning and Controls for Highway Lane Change Maneuvers*, https://www.mathworks.com/videos/developing-planning-and-controls-for-highway-lane-change-maneuvers-1592820244862.html.

[18] MathWorks, *Introducing the New Stateflow Editor*, https://www.mathworks.com/videos/introducing-the-new-stateflow-editor-70056.html.

VIII. APPENDIX

Algorithm 3: Trigger_Lane_change()

Calculate the cost
if $C_total < C_scen$ **then**
  LCTrigger =False
  isLCActive = True
end if

Algorithm 4: Check_Collision()

if isCollsion =True **then**
  CollisionCount++
else
  return to check if MIO detected
end if