Platoon Control of Autonomous Vehicles

Submitted

by

Amisha Verma, 20115012
Archana Singh, 20115019
Ritvik Mahajan, 20117101
Sameer Talwar, 20115122
Shreshth Mehrotra, 20115136

Supervisor
Prof. Arnab Dey

Department of Electrical Engineering
Indian Institute of Technology Roorkee
2022-2023

Declaration

I hereby declare that the work which is presented here, entitled **Platoon Control of Autonomous Vehicles**, submitted for the completion of course EEN 300: Industry Oriented Problem. I also declare that I have been doing my work under the supervision and guidance of **Prof. Arnab Dey, Electrical Engineering Department, Indian Institute of Technology Roorkee**. The matter presented in this report is not submitted for award of any other degree of institute or any other institutes.

Date: 18-04-2023 Signature

Name

Enrolment Number

Certificate

This is to certify that the above statement made by the candidate is true to the best of my knowledge and belief.

Signature

Prof. Arnab Dey

Assistant Professor

Department of Electrical Engineering

Indian Institute of Technology Roorkee

Abstract

With the rapid increase in the deployment of vehicles on limited road infrastructure, there is a need for novel methods to increase traffic density, ensure safety for passengers, and reduce the environmental impact of vehicles. In this regard, autonomous vehicle platoons have recently gauged the interest of academia and industry. In a platoon, multiple vehicles travel closely while maintaining a consistent inter-vehicle distance policy. Controlling vehicular platoons involves various challenges, which include, but are not limited to, disturbances in the leader agent's velocity, lane changing, and the presence of other vehicles near the platoon.

In this work, we tackle the problem of longitudinal and lateral control of platoons during single-lane movements and lane-change manoevre. Initially, we consider a basic bicycle model for the vehicle in MATLAB and develop controllers based on Model Predictive Control for longitudinal distance tracking and PID for yaw control. We then propose a PID-based controller for longitudinal distance tracking and prove that both the distance and yaw error dynamics are stable for all platoon members. We also simulate our proposed method with full vehicle dynamics considered in the CARLA simulator and show through simulations that the platoon remains intact under various adverse conditions. A basic path-planner is also developed for the members of the platoons to ensure that the safety of all the vehicles is ensured during the lane change manoevre.

The report is organized as follows – the first chapter introduces the relevant concepts and definitions associated with the control of vehicular platoons. The second chapter discusses mathematical modeling, and the third chapter moves on to controller design and analysis in MATLAB and Simulink. The fourth chapter presents the path-planning algorithm, and the fifth chapter presents the experiments conducted in both MATLAB and CARLA and the corresponding results. The sixth and final chapter concludes the report.

Table of Contents

List of Figures

Chapter 1

Introduction

Many problems faced on the roads can be solved or reduced in intensity if platoon formations are utilized. It has been proven that autonomous platoons improve traffic flow density, fuel efficiency, and road safety compared to human-driven vehicles. Moreover, as autonomous or self-driven vehicles have started making a presence on roads worldwide, their advanced applications have also surfaced and serve as research opportunities. With this motivation, we have attempted to solve the platoon control problem.

A lot of research has been done in the fields of autonomous vehicles' perception, decision-making, motion control, and motion planning. However, complex and challenging manoeuvres, like cooperative overtaking of autonomous vehicles, still remain unsolved. [1] uses Finite State Machines to attempt the problem of lane-change and overtkaking for a platoon. In addition to maintaining the required distance between platoon members, vehicles also need to interact with the external environment. In this regard, lane keeping and obstacle avoidance also become important. Artificial Potential Fields have been explored in [2], [3] to drive the vehicles to their respective waypoints while avoiding obstacles. Recently, learning-based approaches have also been exploited for planning due to the complex nature of the problem in terms of sample sizes and uncertainties. [4] uses Deep Reinforcement Learning for path planning of individual vehicles.

The following terms are associated with platoons and are defined as such during this research:

Platoon: When defined with respect to automobiles, a platoon is a group of vehicles that travel very closely together, ensuring safety even at high speeds.

Each vehicle communicates with the others in the platoon, but not necessarily with all.

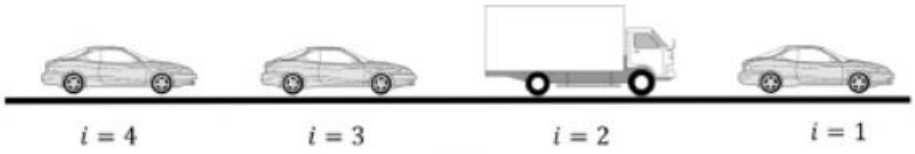

Fig. 1: A 4-member heterogenous platoon

Platoons can be homogeneous or heterogeneous depending on the variety of automobiles in the system. We have only considered homogenous platoons in this work.

Leader: As suggested by the name, it is the vehicle in the front of the platoon that initiates most of the path-planning decisions according to the data obtained from sensors.

Follower: A vehicle in the platoon ranging from the second to last position, essentially following the one ahead of it.

String Stability: A parameter to judge the stability of the platoon at all times, given some controller. An intuitive way of understanding string stability is that any disturbance should not be amplified as we move down the platoon. This can be seen in Fig. 2.

Many formal definitions of String Stability have been proposed by different researchers, which include SFSS (Strong Frequency-domain String Stability), TSS (Time-domain String Stability), LPSS (Lyapunov), etc. The Original String Stability (OSS) criterion adapted from [5] states the following:

A string of vehicles is stable if, for any set of bounded initial disturbances to all the vehicles, the position fluctuations of all the vehicles remain bounded, and these fluctuations approach zeros as $t \rightarrow \infty$.

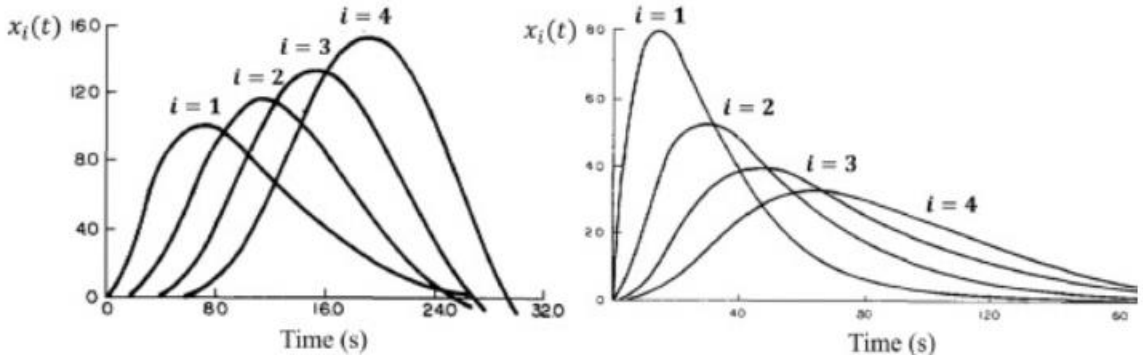

Fig. 2: Graphical interpretation of string stability

Distance Policy: In this work, the desired inter-vehicular distance has been chosen as a function of the velocity of the ego vehicle to ensure safe braking within some permitted value of distance and finite time.

Platoon Control is an umbrella of tasks, a few of which are:

1. **The Control Problem**: A pure control problem that involves controlling the speeds and direction of an agent by designing a controller which gives the values of inputs like acceleration, steering angle, etc. It consists of constructing a model of the system, designing a controller, and testing the outputs' stability and accuracy.
    a. **Longitudinal Control**: A 1-D control ensuring string stability and safe inter-vehicular distance even with jerks in the system.
    b. **Lateral or Steering Control**: Warranting that the vehicle is driven along the road, inside the lanes, or turns safely, lateral and orientation control is necessary. The following are the applications requiring such a type of controller.
        i. Lane Changing
        ii. Overtaking
        iii. Turning

2. **The Planning Problem**: On roads, often the same lane isn't the most feasible to travel in, for example, when the cars ahead are moving at a slower pace or if an obstacle is right ahead. The entire decision-making process of the instant and magnitude for methods, like slowing down, overtaking, etc., are taken by a planner. It also consists of a path planner weighing the feasibility of different paths.

   This segment also involves taking in data from the sensors and processing it for well-informed decision-making. It gives inputs to the controller, which acts as an enforcer of the manoevre.

3. **Communication Problem**: V2V communication includes the method used for interacting and sharing data between the platoon's agents. The researchers must decide if every agent communicates with the leader or just the adjacent agents. Any combination of both can also be employed. Practical communication protocols also involve jitters, breakage, and delays. New and innovative algorithms are being developed for this segment. In this work, we assume a predecessor-following model for communication.

In this work, we focus on the problem of longitudinal control and lane changing for a platoon of autonomous vehicles. Initially, we used a bicycle kinematics model to approximate the model of the vehicle in MATLAB and Simulink. For longitudinal control, we first exploited a modified version of Adaptive Cruise Control based on Model Predictive Control to maintain a velocity-dependent safe distance between two adjacent members of the platoon. Then, a PID-based longitudinal controller is developed from scratch for longitudinal control. We also designed another independent PID-based controller for orientation control. Orientation control is required during a lane change and also helps to keep the vehicles aligned in case of small disturbances. We then move to the CARLA simulator to test with high-fidelity vehicle models and real-world conditions. We modify the default path planner and controller provided by CARLA for platooning applications and show that the platoon remains intact even with various disturbances and other vehicles' presence.

The next chapter starts discussing in more detail our contributions by presenting the mathematical modeling of the platoon and its member vehicles.

Chapter 2

Mathematical Modelling

We use the bicycle kinematics model to approximate the model of all the vehicles. The inputs to the system are the longitudinal velocity *v* and steering angle ψ.

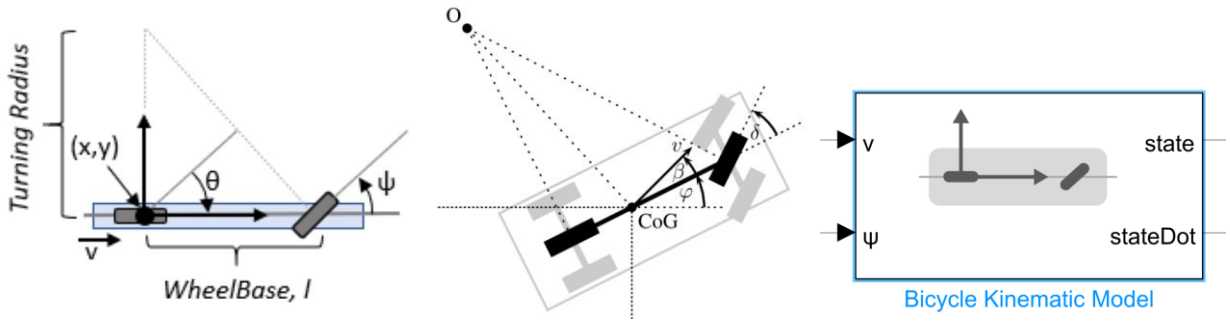

Fig. 3: Bicycle model

The system state is a vector containing its *x, y* coordinates, and orientation θ, with respect to a global inertial frame. The relation between ψ and θ is given by:

$$\dot{\theta} = \frac{V \tan \psi}{l}$$

The bicycle model helps simplify the complex nonlinear dynamics of 4-wheeled vehicles, making the analysis more convenient. However, the vehicle controller (or even human users) generally outputs longitudinal acceleration. The acceleration can be converted to the longitudinal velocity by:

$$V(s) = \frac{a(s)}{s(0.5s+1)}$$

We make the reasonable assumption that the vehicle is able to estimate its own state with decent accuracy.

We are considering a homogenous vehicle platoon where the inter-vehicle distance policy is given by:

$$d_{safe} = d_{default} + T_{gap} v_{ego}$$

Here, d_{safe} is the desired distance between the vehicles, $d_{default}, and\ T_{gap}$ are constants, and v_{ego} is the velocity of the current (ego) vehicle.

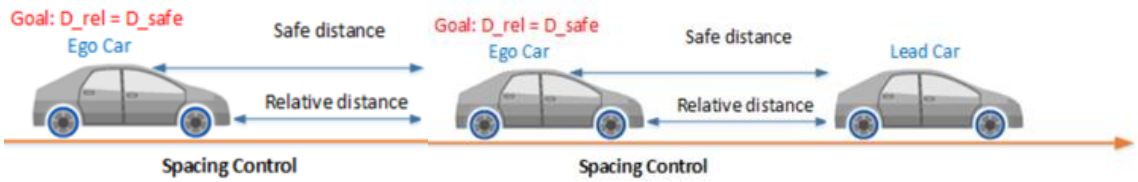

Fig. 4: A three-vehicle platoon

The actual longitudinal distance between two adjacent vehicles is denoted by d_{rel}. Our objective is to maintain the relative distance equal to the safe distance. The controller for achieving this is discussed in the next chapter.

In the case of platoons capable of lane change manoevres, we also need to consider the orientation of member vehicles and ensure that the vehicles are always aligned.

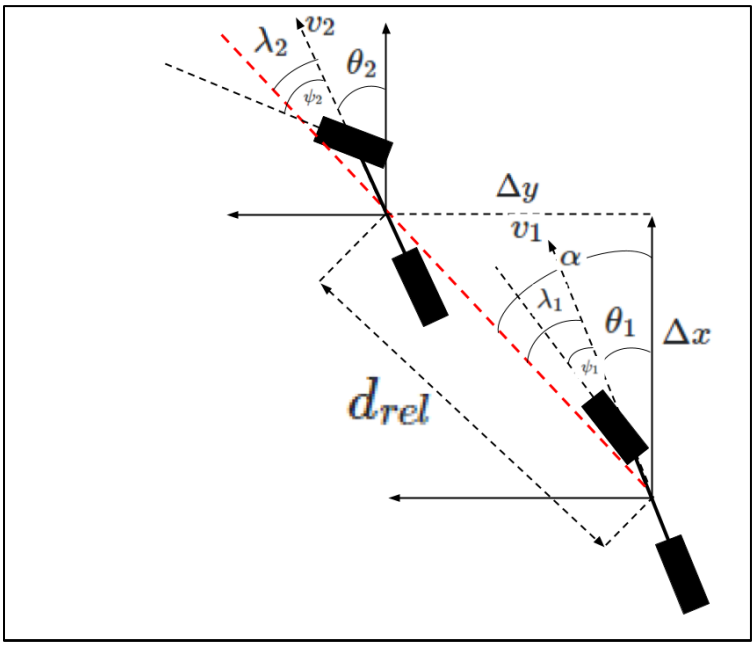

Fig. 5: Vehicles during lane change

Fig. 5 illustrates various quantities of interest for orientation and longitudinal control. Here, vehicle 2 is the leader, and Vehicle 1 is the follower. As defined earlier, d_{rel} is the relative distance between the two vehicles measured along the line joining the centers (We assume that the center of gravity lies on the center of mass for each vehicle) of the two vehicles. $\theta_1 and\ \theta_2$ are the orientations of the vehicles, and $\psi_1 and \psi_2$ are their respective steering angles. The vehicle velocities make angles $\lambda_1 and\ \lambda_2$ with the line joining the two vehicles. To make the follower vehicle track the leader's yaw profile, we need to drive both its orientation and steering angle to that of the leader's.

Chapter 3

Controller Design and Analysis

Control of Leader

To test the platoon under different conditions, we give arbitrary trajectories to the lead vehicle, like a constant velocity profile, an impulse disturbance in the velocity, etc.

Control of Followers

The objective of designing a controller for such a system is to ensure the platoon follows the path the path planner gives. The errors in such a case are defined by-

Error in distance: The difference between the instantaneous value of the inter-vehicular distance between adjacent agents and the safe distance, which is a function of their instantaneous velocities.

Error in Lateral Position: As agents in the platoon should line up one behind the other, this error is defined as the lateral distance between the body line of the leading vehicle (in front of the vehicle of concern) and the instantaneous position of the latter.

Error in Orientation: Similar to the error in the lateral position, it is defined as the difference in the angle of body line with respect to global reference frame.

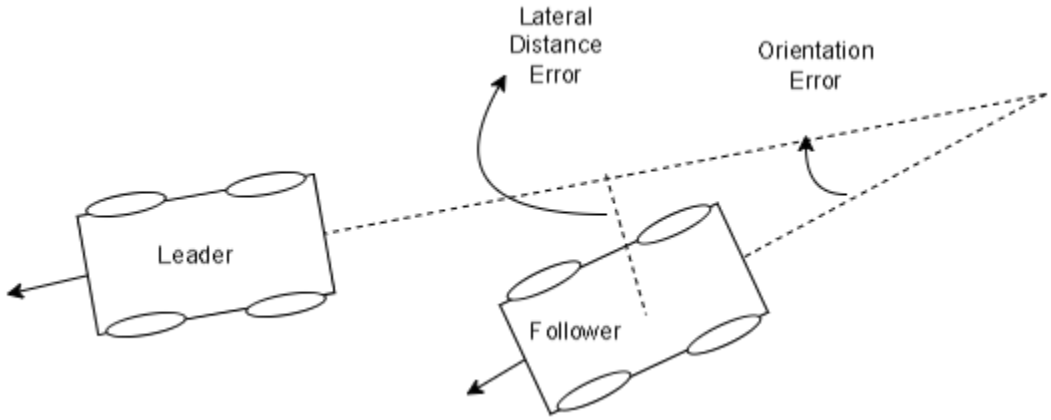

Fig. 6: Two adjacent members of the platoon during lane change

Error in Steering Angle: This is the difference between steering angle inputs given to the leader and the follower. Controlling it would make sure that the platoon reacts faster and simultaneously to lane change and turning initiations.

For longitudinal control, we first explore the existing Adaptive Cruise Control block available in MATLAB and modify it for platooning applications. This method is based on Model Predictive Control [7]. The motivation behind building the control structure around an existing method is that since ACC is usually available off-the-shelf in many real vehicles, it makes our approach easier to implement on practical systems. However, the default ACC module tracks the reference distance only if the relative distance is less than the safe distance. Otherwise, a constant velocity (v_{set}) is tracked.

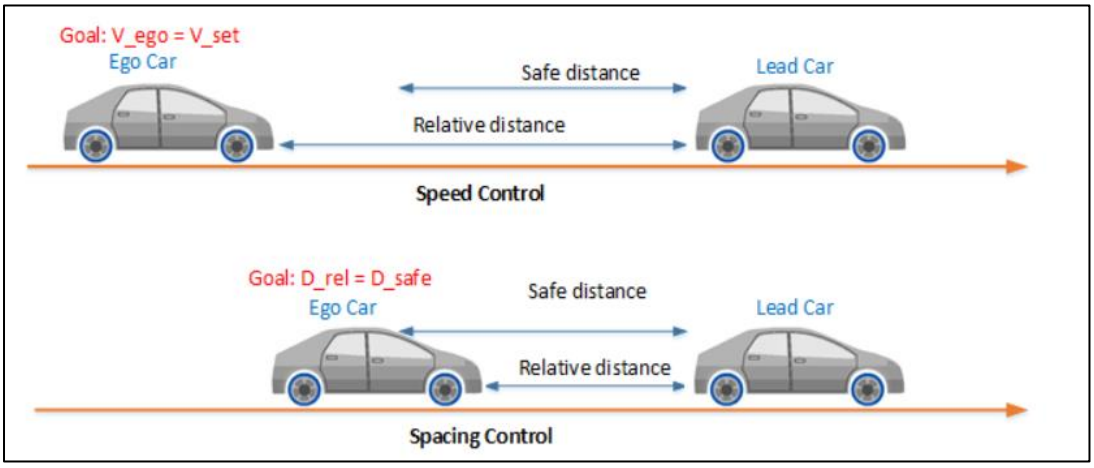

Fig. 7: Default Adaptive Cruise Controller

To make it so that the controller always tried to track d_{safe}, we modify the expression of v_{set} as:

$$V_{set} = V_{lead}\cos\lambda_2 + K_p e_d + K_d\frac{de_d}{dt} + K_I\int_0^t e_d dt$$

Here, $e_d = d_{rel} - d_{safe}$ and $V_{lead}cos\lambda_2$ is the preceding vehicle's velocity along the line joining the two vehicles. This way, the modified ACC module always tries to track d_{safe}.

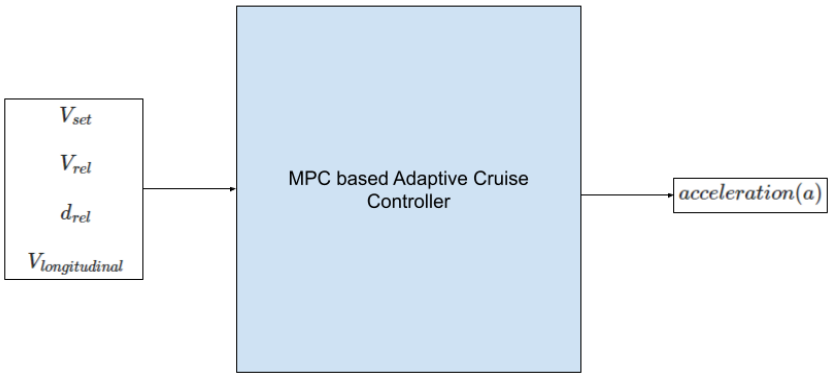

Fig. 8: Modified Adaptive Cruise Controller

The modified ACC module takes as input $V_{set},, d_{rel}, V_{rel}$ (relative velocity between the two vehicles), and $V_{longitudinal}$ (longitudinal velocity of the ego vehicle).. It calculates a, the component of vehicle acceleration along the line joining the two vehicles. Thus, the acceleration input to the vehicle will be $a_{longitudinal} = a\ sec\lambda_{\text{ego}}$.

This controller ensures that the longitudinal distance error is driven to zero. Next, we develop a PID-based controller from scratch to replace the ACC block.

The longitudinal PID controller tries to make the relative distance between adjacent vehicles equal to the required safe distance.

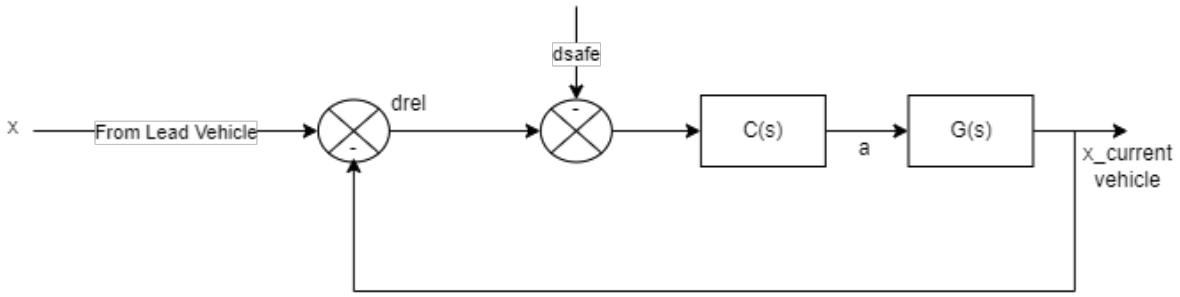

Fig. 9: PID controller block diagram for longitudinal control

We use another independent PID-based controller for tracking the desired yaw values for orientation control.

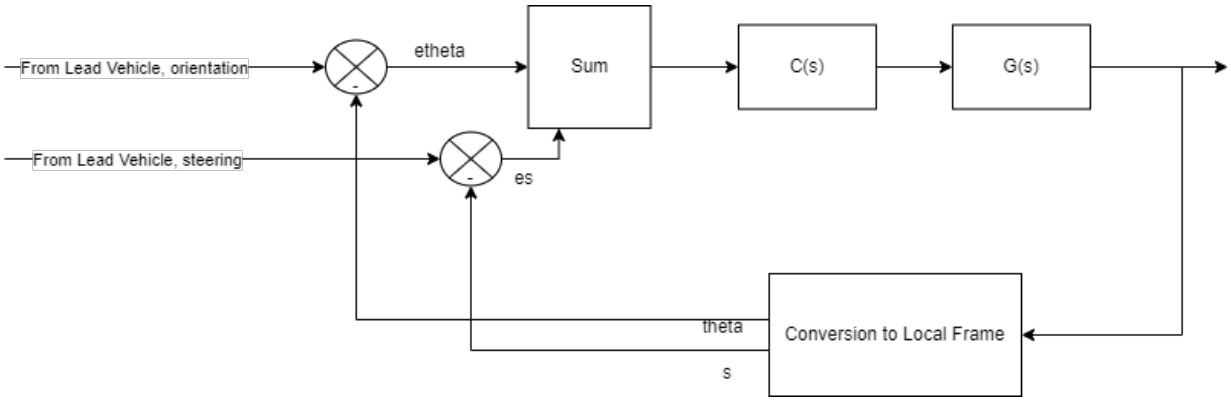

Fig.10: PID controller block diagram for orientation control

Proof of convergence of the distance error:

We define the Transfer function from acceleration to velocity as $G(s) = \frac{1}{s(0.5s+1)}$, , and our proposed PD controller as $C(s) = k_p + k_d s$. Then for the i[th] vehicle:

$$a_i(s) = C(s)E_i(s) \tag{1}$$

$$V_i = sX_i = a_i G = CE_i G \implies X_i = \frac{CE_i G}{s} \tag{2}$$

The safe, and relative distances between the i[th] and the (i-1) [th] vehicle (with the (i-1) [th] vehicle being in front) are defined as:

$$D_{rel} = \mathcal{L}(x_{i-1} - x_i) = \frac{X_{i-1}}{s} - \frac{X_i}{s} \tag{3}$$

$$D_{safe} = \mathcal{L}(10 + T_{gap} v_i) = \frac{10}{s} + T_{gap} s X_i \tag{4}$$

The distance error between this pair of vehicles is defined as $E_i = D_{rel} - D_{safe}$. Using (1)-(4), this simplifies to:

$$E_i = D_{rel} - D_{safe} = X_{i-1} - \frac{10}{s} - (1 + sT_{gap})X_i = X_{i-1} - \frac{10}{s} - (1 + sT_{gap})(\frac{CE_i G}{s}) \tag{5}$$

$$E_i(1 + \frac{CG}{s} + T_{gap} CE_i G) = X_{i-1} - \frac{10}{s} \tag{6}$$

$$E_i = \frac{X_{i-1} - \frac{10}{s}}{1 + \frac{CG}{s} + T_{gap} CG} = \frac{\frac{CE_{i-1} G}{s} - \frac{10}{s}}{1 + \frac{CG}{s} + T_{gap} CG} = \frac{E_{i-1} - \frac{10}{CG}}{1 + \frac{s}{CG} + sT_{gap}} = \frac{E_{i-1} - a}{b} \tag{7}$$

Where:

$$a = \frac{10}{CG} = \frac{10s(0.5s+1)}{k_p + k_d s} \tag{8}$$

$$b = 1 + \frac{s}{CG} + sT_{gap} = 1 + sT_{gap} + \frac{s^2(0.5s+1)}{k_p + k_d s} \tag{9}$$

(7) is a recursive equation involving E_i and E_{i-1}. for i=2, 3...n (note that i=0 is the lead vehicle). We can observe the first few terms:

$$E_2 = \frac{E_1 - a}{b} \tag{10}$$

$$E_3 = \frac{E_2 - a}{b} = \frac{E_1 - a(1+b)}{b^2} \tag{11}$$

$$E_4 = \frac{E_3 - a}{b} = \frac{E_1 - a(1+b+b^2)}{b^3} \tag{12}$$

From (10)-(12) we write a closed-form solution for E_n as:

$$E_n = \frac{E_1 - a(1+b+b^2..b^{n-2})}{b^{n-1}} \tag{13}$$

From the final value theorem, we can calculate the steady state error between the nth pair of vehicles as:

$$e_{ss} = \lim_{s\to 0} sE_n \tag{14}$$

From (8) and (9):

$$\lim_{s\to 0} a = 0 \tag{15}$$

$$\lim_{s\to 0} b = 1 \tag{16}$$

The error between the leader and the first follower E_1 can be expressed using (6) as:

$$E_1 = \frac{X_L - \frac{10}{s}}{1 + \frac{CG}{s} + T_{gap}CG} = \frac{s(V_L - 10)}{s^2 + (1 + sT_{gap})\frac{k_p + k_d s}{0.5s+1}} \tag{17}$$

Using (13) to (17):

$$e_{ss} = \lim_{s\to 0} sE_n = \lim_{s\to 0} sE_1 = \frac{1}{k_p}\lim_{s\to 0} s^2 V_L(s) = \lim_{s\to 0} \frac{s^2 G}{k_p} a_L(s) \tag{18}$$

Since $G(s) = \frac{1}{s(0.5s+1)}$, we finally get :

$$e_{ss} = \lim_{s \to 0} \frac{s}{k_p} a_L(s) \tag{19}$$

For an impulse disturbance, $e_{ss} = 0$ .. For a step disturbance $a_L(t) = ku(t)$: $e_{ss} = \frac{k}{k_p}$:

a constant, bounded steady state error. These results are validated experimentally in chapter (5).

The results obtained hold true for any pair of vehicles and are thus independent of the platoon size.

Proof of convergence of the orientation error:

Let ψ_i denote the steering angle and θ_i denote the orientation (yaw) angle of the i[th] vehicle (with i=0 being the leader, i.e $\theta_0 = \theta_L$. The proposed controller is:

$$C(s) = k_p + k_d s \tag{20}$$

The orientation error is defined as:

$$E_i = \theta_{i-1} - \theta_i \tag{21}$$

From the bicycle model kinematics, and assuming small steering angles,

$$\begin{aligned} \dot{\theta} &= \frac{v \tan \psi}{l} = k \tan \psi \approx k\psi \\ s\theta(s) &= k\psi(s) \implies \psi = \frac{s\theta}{k} \end{aligned} \tag{22}$$

To ensure that the vehicles in the platoon follow their predecessor's yaw profile, we need to drive both the steering angle and orientation of the ego vehicle to that of the predecessor's. Thus, the steering command to the ego vehicle considers the steering angle of the preceding vehicle also in addition to the yaw error between the two vehicles. The control law is given by:

$$\psi_i = k_1 \psi_{i-1} + CE_i \implies \frac{s\theta_i}{k} = k_1 \frac{s\theta_{i-1}}{k} + CE_i \tag{23}$$

Using (21):

$$E_i = (1-k_1)\theta_{i-1} - \frac{kCE_i}{s} \implies E_i = \frac{s(1-k_1)}{s+kC}\theta_{i-1} = \alpha\theta_{i-1} \tag{24}$$

Where:

$$\alpha = \frac{s(1-k_1)}{s+kC} \tag{25}$$

Now, from (21) and (24) , we get:

$$\theta_{i-1} = \theta_{i-2} - E_{i-1} \implies E_i = \alpha(\theta_{i-2} - E_{i-1}) = E_{i-1} - \alpha E_{i-1} = E_{i-1}(1-\alpha) \tag{26}$$

Since $E_1 = \alpha\theta_0 = \alpha\theta_L$, (26) simplifies to :

$$E_n = E_1(1-\alpha)^{n-1} = \alpha(1-\alpha)^{n-1}\theta_L \tag{27}$$

Now, we observe that: $\lim_{s\to 0}(1-\alpha)^{n-1} = 1$

Thus, by the final value theorem, we can calculate the steady state error:

$$e_{ss} = \lim_{s\to 0} sE_n = \lim_{s\to 0} s\alpha\theta_L \tag{28}$$

Finally, using (22) and (28),

$$e_{ss} = \frac{1}{k_p k}\lim_{s\to 0} s^2\theta_L(s) = \frac{1}{k_p}\lim_{s\to 0} s\psi_L(s) \tag{29}$$

For an impulse disturbance, $e_{ss} = 0$ and for a step disturbance $\theta_l = cu(t)$: $e_{ss} = \frac{c}{k_p}$ which is constant and thus, bounded.

Again, the results obtained hold true for any pair of vehicles and are independent of the platoon size.

To further improve the performance of the controller, we added an extra term that accounts for the lateral deviation: to try to ensure that the vehicles end up in the middle of the lane when the lane change manouvre is finished. Also, we added an exponential term to speed up the process when the error is large. The additional term goes to 0 as the lateral error goes to 0. The modified control law in the time domain is given by:

$$\psi_i = k_1\psi_{i-1} + k_p e_\theta + k_d\frac{de_\theta}{dt} + k_y e_y e^{k_{exp}|e_y|} \tag{30}$$

Where:

$$e_\theta = \theta_{i-1} - \theta_i \tag{31}$$

$$e_y = y_{i-1} - y_i \tag{32}$$

The overshoot was reduced considerably which improved the transient performance considerably.

Chapter 4

Path Planning

The job of any path planner is to generate feasible and safe paths to drive the vehicle to its goal location in the most efficient manner. In this work, in addition to our developed logic, we are also using the default path planner provided by the CARLA simulator [6], called the Traffic Manager (TM). The architecture of TM is shown in Fig. 11.

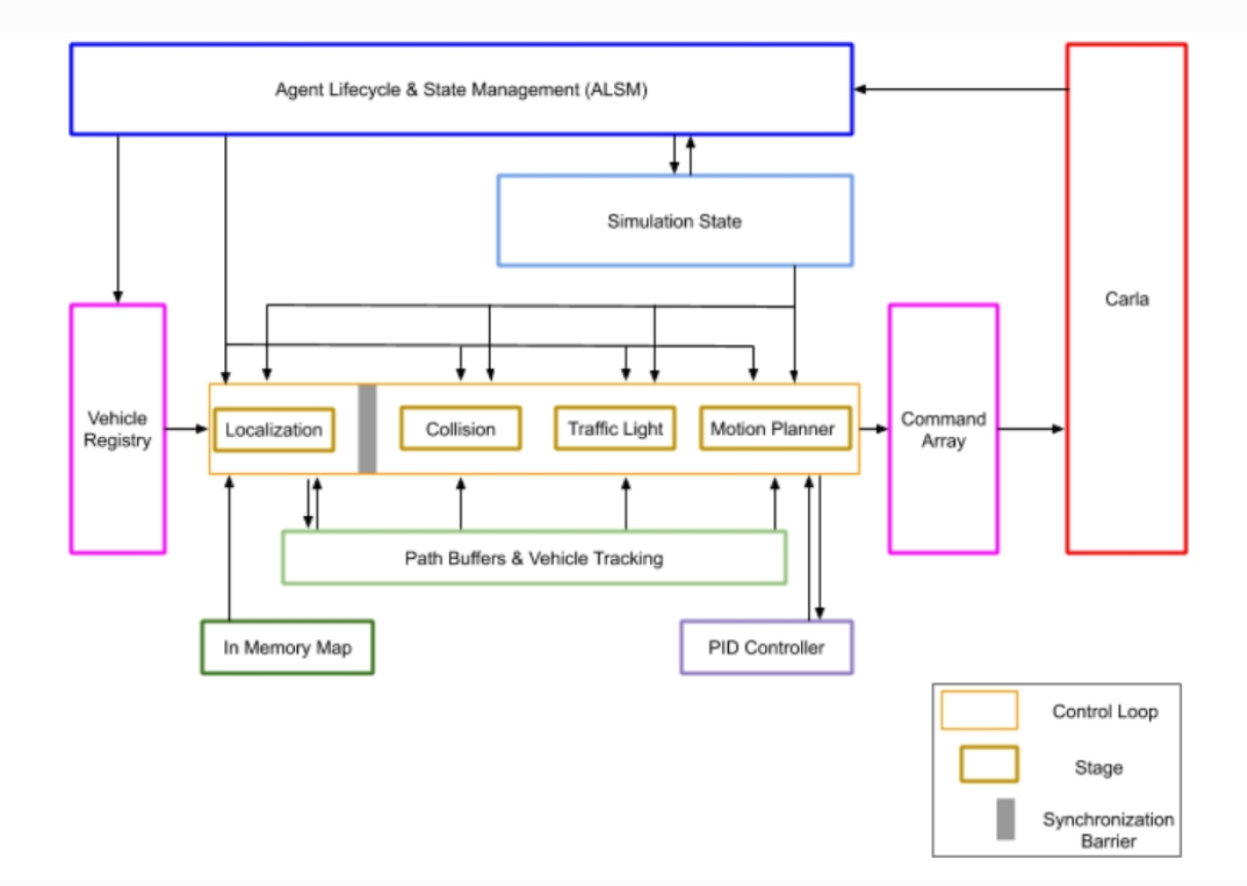

Fig. 11: Architecture of CARLA's Traffic Manager

The planner has various components, which are explained as follows:

- Agent Lifecycle and State Management (ALSM): This component keeps track of all the vehicles and pedestrians present in the world and their states (positions and velocities).

- Vehicle Registry: It receives an updated list of all the vehicles and pedestrians from the ALSM and stores the vehicles which are controlled by the TM.

- Simulation State: This receives data from the ALSM and stores information like actor state, traffic light states, etc., in a cache for faster control loop implementation.

- Control Loop: This is one of the primary components of the TM, and it manages the calculations of the next command for all autopilot vehicles. The control loop has five stages:

    - Localization: After obtaining the vehicle state from the Simulation State component, it relates every vehicle with a near-future path according to its trajectory. Naturally, this path is affected by vehicle speed and high-level decisions such as lane change. This stage also compares paths with each other to estimate potential collisions.
    - Collision: The Collision stage receives a list of vehicle pairs that can potentially collide, i.e., their paths overlap. After evaluating if the vehicles will actually collide by considering the bounding boxes of the vehicles, it sends all the possible hazardous paths to the Motion Planner stage for modification.
    - Traffic Light: This stage manages the traffic regulators like traffic lights and stop signs. At unsignaled intersections, a First-In-First-Out (FIFO) order is followed.
    - Motion Planner: This stage takes as input the vehicle state, path, and possible hazards. It is responsible for making high-level behavioral decisions like calculating the brake command for preventing collisions. These commands are sent to the Command Array component for implementation. Under the hood, this component runs a PID controller.
    - Vehicle Lights: As the name suggests, this component is responsible for controlling the lights based on vehicle movement. Its tasks include turning on indicators while turning and controlling headlights, stop lights, and fog lights.

- In-Memory Map: Its job is to convert the map into a grid of discrete waypoints. This component also identifies the vehicles located nearby to these waypoints.

- Path Buffer and Vehicle Tracking: This component contains the expected path for all the vehicles and stores the In-Memory Map to relate the vehicles with their nearby waypoints.

- PID Controller: It is responsible for calculating the low-level commands, like throttle, brake, and steering, for the vehicles once the target waypoint is received from the Motion Planner.

- Command Array: It is the final stage of the Traffic Manager. It receives all the commands and applies them in the simulator.

The multiple stages in the planner ensure that the generated paths are safe and collisions are avoided. However, for platoon path planning, we must consider all the members' safety while additionally ensuring that the platoon remains intact. This means that the decisions of the individual members cannot be independent of each other.

We use the default automatic planner and controller based on TM provided by CARLA for the lead vehicle. This is better than using a custom planner and controller for the leader since using

the existing planner provides an opportunity to test the follower vehicles' behaviors for various leader behaviors. The leader is given a goal location, and its planner and controller are responsible for generating and tracking the path toward the goal. During its journey, the planner may direct the leader to change lanes, increase or decrease speeds or stop. The followers are also given a goal location in accordance with the distance policy.

For the followers, we create a rectangular bounding box around the vehicle to check whether a lane change manoevre is safe. The steering commands given by the planner of the leader are checked continuously to check if the leader wants to change lanes. If the steering command exceeds a pre-defined threshold, all the follower vehicles' bounding boxes (in the direction of the proposed lane) are checked.

- If all of them are clear, the platoon undertakes the lane change manoevre with the longitudinal and lateral controllers, as discussed in the previous chapter.
- If any vehicle contains another vehicle not part of the platoon in its bounding box, the lane change manoevre is classified as unsafe and aborted. In this case, the leader's steering commands are overwritten, and the whole platoon remains in its lane.

This way, it is ensured that the platoon only changes lanes if all the vehicles are safe and there is no splitting of the platoon. As an added safety measure, if the platoon is broken in any case, we deploy the default automatic controller on all the individual members of the platoon. The simulation is stopped once all the platoon members reach the final destination.

Chapter 5

Experiments and Results

Modified Adaptive Cruise Control and PID Yaw Controller

The first experiment was carried out by simulating the model of the platoon in MATLAB for the CACC system which used MPC blocks. We simulated various conditions to verify the appropriate working of the controller.

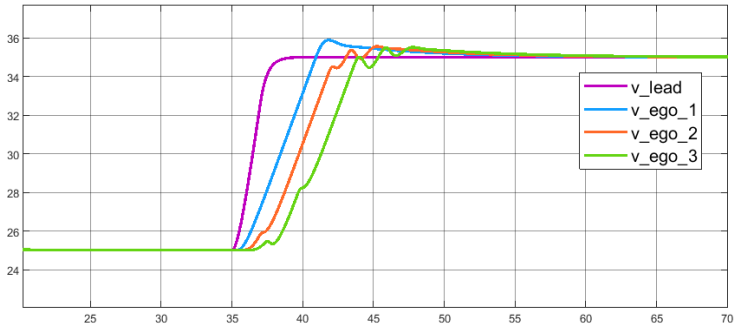

Fig.12: velocity vs time for 4-vehicle platoon

Fig. 12 represents the output of a velocity vs time curve when a short-duration pulse of acceleration is given to the leader of a 4-vehicle platoon as input. The result is that all the vehicles quickly readjust their velocity within 15 seconds. We can see that the steady state error is near to zero.

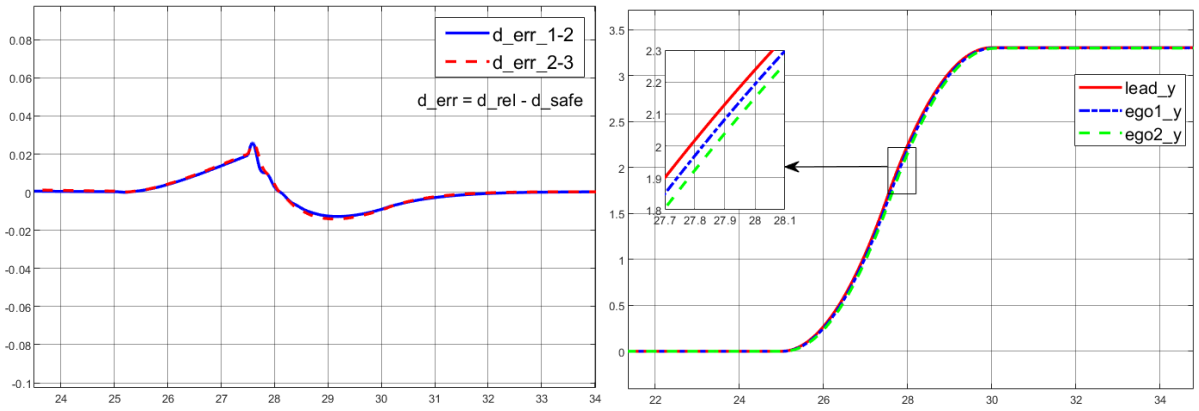

Fig. 13(a): error in distance vs time; *Fig. 13(b): y-position vs time;*
for a 3-vehicle platoon performing lane change manoevre.

Fig. 13(b) depicts the y-position vs t curve for a 3-vehicle platoon, minimum error is observed in this case. Also, through Fig. 13(a) it can be observed that the CACC works almost flawlessly during the lane change manoevre as the relative distance error between the vehicles does not exceed a few millimeters in magnitude.

After concluding this experiment, further tests were carried out on the newly formed MATLAB model with custom PID controllers in place of the MPC blocks.

PID-based Longitudinal and Yaw Control

Fig. 14 and Fig. 15 are the observed results of an overtaking manoevre carried out by the platoon using this PID controller. A very small constant displacement can be observed with this controller, however, during an overtaking manoevre, the vehicles converge back to their original position in the original lane. The longitudinal error still remains miniscule in magnitude implying the CACC with PID also works very well.

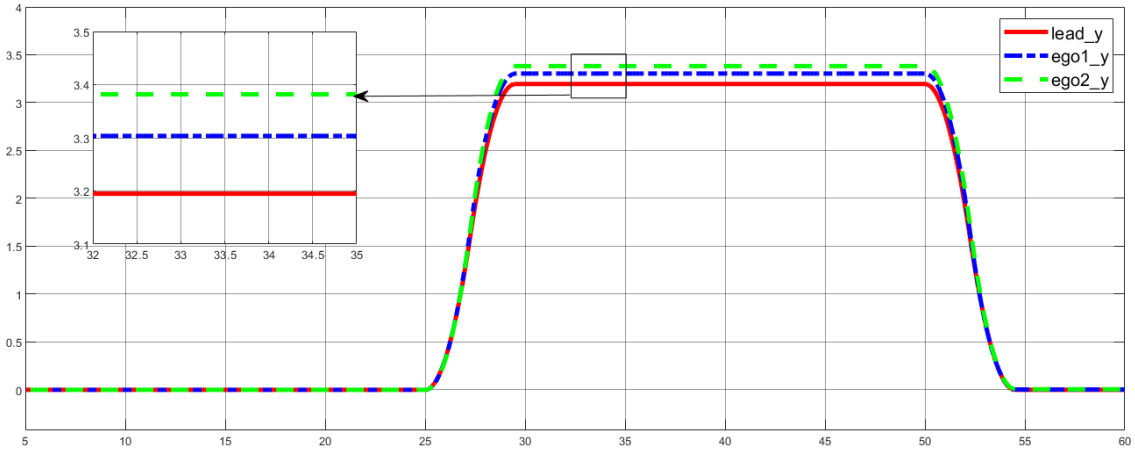

Fig. 14: y-position vs time for overtaking manoevre using PID controller

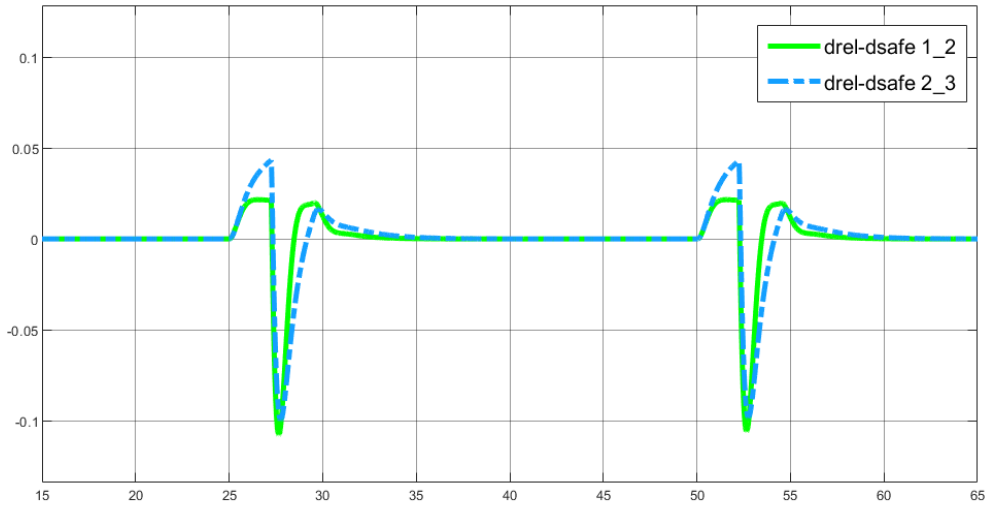

Fig. 15: error in distance vs time for overtaking manoevre using PID controller

The next experiment in MATLAB was to test the stability of the model by verifying the convergence of the errors in the model for different types of inputs (acceleration). It was observed that for an impulse acceleration, the steady state errors indicated by the difference between relative distance of errors and safe distance, was zero. The error for constantly accelerating cars was also a constant finite value.

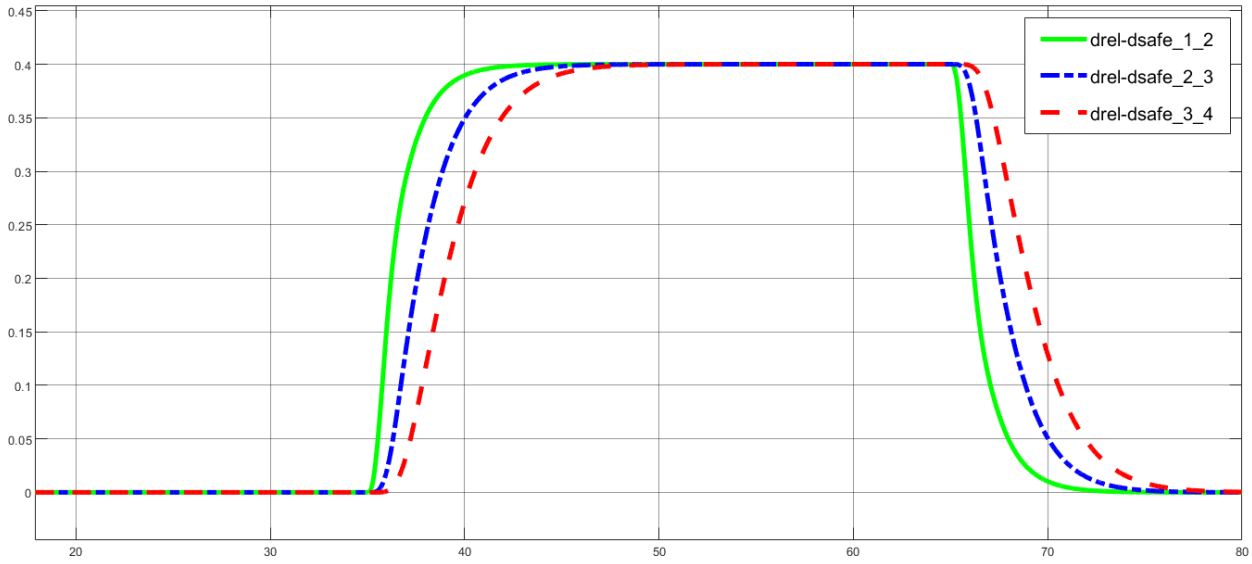

Fig. 16: error in distance vs time for a 4 vehicle platoon depicting finite stead-state error

High Fidelity Simulations in CARLA

MATLAB and Simulink were powerful tools for simulating the pure control behaviour when manually given initiating commands. In order to test out scenarios of decision-making, we used CARLA, which has provisions for simulating an entire town's urban mobility. We have used Python API for CARLA and programmed the vehicle behaviors using Python.

CARLA also offers various libraries for including sensors, like Camera, LiDAR, GNSS, etc. Making use of them, we simulated an autonomously driven vehicle given a set of coordinates as the destination. We then implemented the existing autopilot provided by CARLA, with some modifications (discussed in the previous chapter) for the lead vehicle.

Carla considers the exact nonlinear vehicle dynamics, and our controllers still ensure convergence to 0 steady state error for both distance and orientation (Fig 17). Fig 18 shows that the vehicles end up in the middle of the lane (the lane width was chosen as 4 meters), at the end of the lane change manouvre, with minmal overshoot.

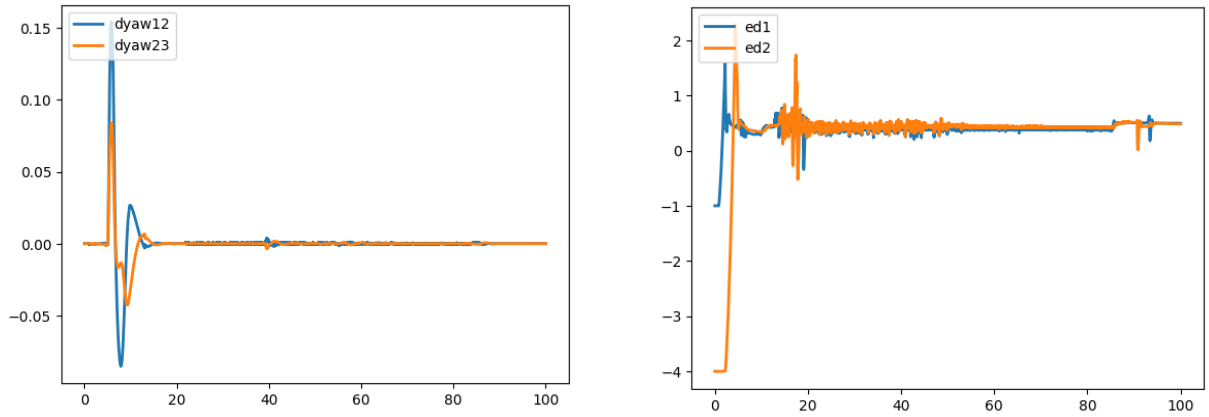

Fig. 17: Orientation and distance errors for a 3 membered platoon.

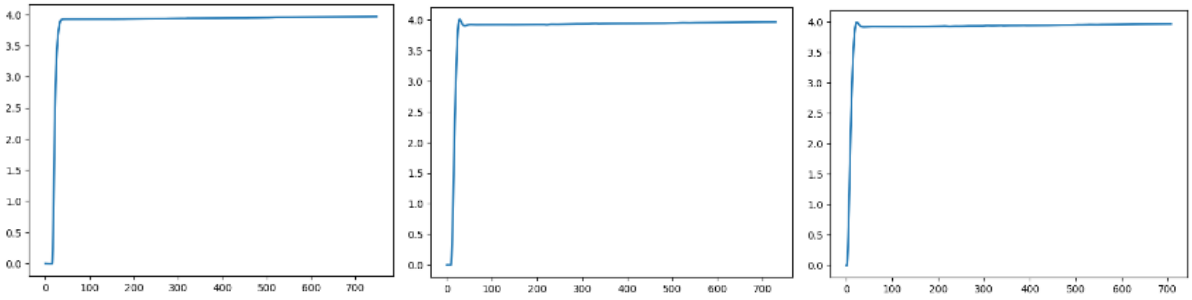

Fig. 18: y vs x plots for a 3 membered platoon (leader, follower 1, follower 2).

Chapter 6

Conclusion and Future Works

In this report, we discussed the control of autonomous vehicle platoons. We proposed a controller based on the existing ACC infrastructure usually available in CAVs for longitudinal control and developed a PID-based lateral controller for yaw tracking. We also presented a PID-based longitudinal controller and proved that both the distance and yaw error dynamics are stable for the bicycle kinematics model. We then moved on to the path planning algorithm that ensures safety for all members of the platoon. Finally, all the proposed methods are combined and tested using the high-fidelity vehicle model and general traffic conditions in the CARLA simulator.

One of the main advantages of our approach is its ability to be readily implemented in practical systems. Instead of selecting an overengineered method, we propose a simple-to-understand and easy-to-implement control method that still guarantees convergence. Further, we showed through simulations that our method is able to handle complex and unmodelled dynamics in CARLA and is robust to variations in the lead vehicle's velocity. During lane change, applying both longitudinal and lateral control parallelly ensures that the platoon remains intact. Further, the path planner continuously checks if a lane change is feasible, ensuring no unsafe manoevres are possible.

A future research direction can include developing more robust path planners that may incorporate learning-based approaches. Further, formal guarantees on string stability, in addition to error convergence, can be explored. Overtaking is a more complex manoevre than lane change, and more sophisticated control and planning approaches should be developed for this relatively unsolved problem.

References

[1] M. Strunz, J. Heinovski and F. Dressler, “CoOP: V2V-based Cooperative Overtaking for Platoons on Freeways,” 2021 IEEE International Intelligent Transportation Systems Conference (ITSC), Indianapolis, IN, USA, 2021, pp. 1090-1097, doi: 10.1109/ITSC48978.2021.9565122.

[2] Y. Rasekhipour, A. Khajepour, S. -K. Chen and B. Litkouhi, “A Potential Field-Based Model Predictive Path-Planning Controller for Autonomous Road Vehicles,” in IEEE Transactions on Intelligent Transportation Systems, vol. 18, no. 5, pp. 1255-1267, May 2017, doi: 10.1109/TITS.2016.2604240.

[3] S. Xie, J. Hu, P. Bhowmick, Z. Ding and F. Arvin, “Distributed Motion Planning for Safe Autonomous Vehicle Overtaking via Artificial Potential Field,” in IEEE Transactions on Intelligent Transportation Systems, vol. 23, no. 11, pp. 21531-21547, Nov. 2022, doi: 10.1109/TITS.2022.3189741.

[4] Changxi You, Jianbo Lu, Dimitar Filev, Panagiotis Tsiotras,”Advanced planning for autonomous vehicles using reinforcement learning and deep inverse reinforcement learning, Robotics and Autonomous Systems”,Volume 114, 2019.

[5] Shuo Feng, Yi Zhang, Shengbo Eben Li, Zhong Cao, Henry X. Liu, Li Li, String stability for vehicular platoon control: Definitions and analysis methods, Annual Reviews in Control, Volume 47, 2019, Pages 81-97, ISSN 1367-5788, https://doi.org/10.1016/j.arcontrol.2019.03.001

[6] Dosovitskiy, Alexey & Ros, German & Codevilla, Felipe & López, Antonio & Koltun, Vladlen. (2017). CARLA: An Open Urban Driving Simulator.

[7] Al-Gabalawy, M., Hosny, N.S. & Aborisha, Ah.S. Model predictive control for a basic adaptive cruise control. Int. J. Dynam. Control 9, 1132–1143 (2021). https://doi.org/10.1007/s40435-020-00732-w