Modeling and Simulation of Collaborative Highway-merging Vehicles

Design project Systems, Control and Mechatronics (Group 11)

Erik Lind*, Anna Molnö*, Saeed Salih*, Luming Wang*, Qun Zhang*
*Chalmers University of Technology
Email: {erli, molno, saeedsal, luming, qunz}@chalmers.se

***Abstract*—This study investigates the opportunities provided by connected and automated vehicles in the context of road merging, where multiple vehicles collaborate to optimize the overall outcome. This task necessitates the resolution of a complex real-time challenge, which involves the global design of joint trajectories for cars (L3 automation level), which are highly automated, and trucks (L1 automation level), which have a basic level of automation. The objective is to ensure mutual non-interference, culminating in successful road merging. The driverless approach adopted in this research leverages the distinction between desired and planned trajectories, permitting vehicles to exert mutual influence for collective advantage. Each vehicle independently strives to circumvent the desired trajectories of others, thus engendering cooperative behavior. Within this framework, a car is equipped with MPC, optimizing its own trajectory. This innovative, decentralized methodology has been applied in an experiment, involving two full-sized vehicles on a test track in the CARLA environment. The paper presents the outcomes of this experiment, highlighting the validation for feasibility and accuracy.**

I. INTRODUCTION

With the right implementation, automated vehicles can increase the comfort for passengers as well as road safety. Autonomous vehicles use different types of sensors and cameras as well as previously processed data. With the data, the need for a human to drive the vehicle can be partly or totally replaced [1].

The European Union has been funding several projects with a focus on developing autonomous driving with the aim to obtain a higher road safety and develop systems that can work in several countries. The first project *euroFot* started in the year 2012 and aimed to help the driver with detecting hazards, preventing accidents, and making driving more efficient [2]. For further development of autonomous driving the EU funded the project *AdaptiVe* which focused on demonstrating automated driving in a more complex environment as well as the legal framework regarding implementation of automated driving [3]. A third project *L3Pilot* focused on actual road tests with passenger cars and developing a piloting environment [4].

The EU is currently funding the project *Hi-Drive* with the aim to develop more robust self-driving vehicles, extend the operational design domains and increase the usability and functionality across countries and different brands of the vehicles [5]. Decision-making in autonomous vehicles can rely on onboard information as well as data obtained and processed from other autonomous vehicles, which also gather environmental information. By using Vehicle-to-Vehicle (V2V) communication, the autonomous system can make decisions based on other vehicles decisions and planned trajectory. With the ability to adjust the driving based on information from surrounding traffic, several conflict situations can be solved among lane merging and highway merging.

As a part of *Hi-Drive*, a collaborative V2V-enabled algorithm for trajectory planning using Model Predictive Control (MPC) has been developed at Chalmers. MPC is a control strategy designed to optimize system performance through the minimization of a predefined cost function. This approach leverages feedback to incorporate a predictive model of the system over a specified time horizon. The algorithm produced at Chalmers is adapted to a highway-merge scenario where one vehicle is driving onto a highway lane where there is a second vehicle. The algorithm has been tested in a developed Python simulation, using a point-mass model to represent the vehicles.

Since the realm of autonomous vehicle research is dynamic and heavily dependent on gathered data, researchers face several obstacles when conducting real-world data tests due to inherent safety concerns as well as logical challenges. Using a simulator to address these concerns, provides a controlled environment for experiments. The software used for such simulations is CARLA, which is an open-source simulator that can be used to prototype, train, and validate different autonomous driving models by providing an advanced 3D environment for testing.

II. THIS WORK

This project aims to further develop the algorithm produced at Chalmers by introducing a more advanced model of the vehicle and the controller to allow for a successful highway merge. It also explores the challenges of road merging in connected and automated vehicle technologies, employing a driverless algorithm to design joint trajectories for vehicles with varying automation levels. The focus is on real-time resolution of traffic scenarios and optimizing flow and safety through collaborative vehicle efforts. A novel aspect is the differentiation between desired and planned trajectories, enabling adaptive, cooperative behavior that enhances both individual performance and overall traffic efficiency.

MPC is a central element, used to optimize individual trajectories while considering collective dynamics. This research

tests these concepts using full-sized vehicles in a simulated CARLA environment, demonstrating the approach's feasibility and effectiveness. The results contribute to understanding collaborative behaviors in automated vehicles during road merging.

III. RELATED WORK

Previous research presents an approach to collision avoidance in Connected and Automated Vehicles (CAVs) using a decentralized method [6]. The study emphasizes the development of automated vehicles to improve safety and comfort by integrating environment perception and electronic actuator controls. The core concept involves vehicles cooperatively avoiding obstacles and each other through a autonomous method, where vehicles independently avoid the "desired" trajectories of others, leading to cooperative behavior. This method uses MPC to optimize vehicle trajectories.

IV. VEHICLE MODEL AND CONTROL SYSTEM

A. Assumption

It is assumed that the lane merging scenario includes an L3 car and an L1 truck. Among them, the car follows the control of MPC and needs to turn and merge to the main lane, while the truck only needs to follow the acceleration and deceleration control of the Intelligent Driver Model (IDM) [7]. This mathematical model describes how a vehicle modulates its speed, either by accelerating or decelerating, in response to the distance from the vehicle ahead and the relative speed difference between them.

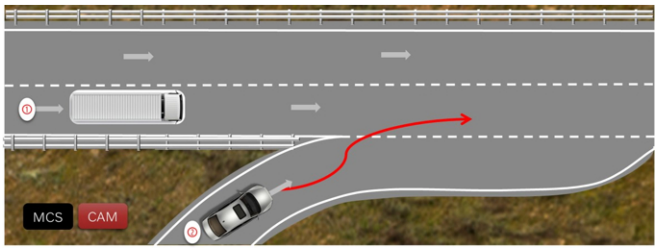

Fig. 1: The lane merging scenario.

The IDM is formulated as follows:

$$\dot{v} = a^* \left[1 - \left(\frac{v}{v_0} \right)^{\sigma} - \left(\frac{s^*(v, \Delta v)}{s} \right)^2 \right] \tag{1}$$

where,

- $\dot{v}$ is the acceleration of the vehicle,
- v is the current velocity of the vehicle,
- v_0 is the desired velocity in free traffic,
- σ is the acceleration exponent,
- s is the current gap to the leading vehicle,
- Δv is the difference in velocity for the leading vehicle,
- $s^*(v, \Delta v)$ is desired minimum gap to the leading vehicle.

$s^*(v, \Delta v)$ is given by the equation:

$$s^*(v, \Delta v) = s_0 + s_1 \sqrt{\frac{v}{v_0}} + \frac{v \Delta v}{2\sqrt{a^* b^*}} \tag{2}$$

where,

- s_0 is the minimum stationary gap,
- s_1 is the minimum dynamic gap,
- a^* is the maximum acceleration,
- b^* is the comfortable deceleration.

B. Kinematic Vehicle Model

The car used in this scenario is modeled using a kinematic bicycle model, as is shown in Fig. 2.

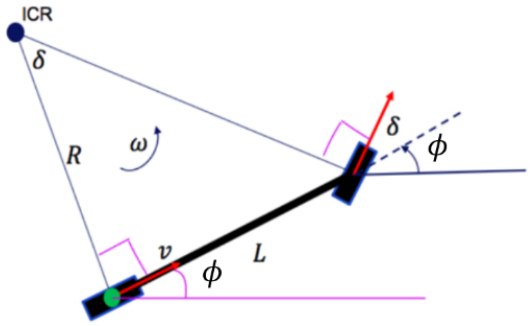

Fig. 2: The vehicle kinematic bicycle model.

The state space representation of the vehicle model includes longitudinal speed, lateral speed, heading angle, time, and velocity as states. The state space in continuous form is represented as:

$$\begin{aligned} \dot{p}_x &= v\cos(\phi) \\ \dot{p}_y &= v\sin(\phi) \\ \dot{v} &= a \\ \dot{\phi} &= -\frac{v}{L}\tan(\delta) \end{aligned} \qquad \text{State: } \mathbf{x} = \begin{bmatrix} p_x \\ p_y \\ \phi \\ v \end{bmatrix} \qquad \text{Input: } \mathbf{u} = \begin{bmatrix} a \\ \delta \end{bmatrix} \tag{3}$$

where,

- p_x, p_y is the Cartesian position of the rear wheel,
- ϕ is the vehicle orientation,
- L is the vehicle length,
- v is the velocity of the rear wheel,
- a is the longitudinal acceleration,
- δ is the steering input.

C. Linear Time-Varying MPC

The control of a vehicle using a bicycle model presents a complex, nonlinear challenge. To address this, two primary approaches are typically considered: the Linear Time-Varying (LTV) model and control through nonlinear models using methods like the Runge-Kutta algorithm.

1) Nonlinear Model Control: For controlling nonlinear models, the system's dynamics are typically represented by the following differential equation:

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}) \tag{4}$$

This equation characterizes how the system's state x changes over time in response to control inputs $\mathbf{u}$. However,

due to the nonlinear nature of $f(\mathbf{x}, \mathbf{u})$, analytical solutions are often not feasible, and numerical methods such as Runge-Kutta are employed for approximations.

2) Linear Time-Varying Model: An alternative to direct nonlinear control is the LTV approach. The LTV model linearizes the nonlinear system around a series of operating points, which can vary over time. This linearization is represented as:

$$\dot{\mathbf{x}} \approx f(\bar{\mathbf{x}}, \bar{\mathbf{u}}) + \left.\frac{\partial f}{\partial \mathbf{x}}\right|_{\bar{\mathbf{x}}, \bar{\mathbf{u}}} (\mathbf{x} - \bar{\mathbf{x}}) + \left.\frac{\partial f}{\partial \mathbf{u}}\right|_{\bar{\mathbf{x}}, \bar{\mathbf{u}}} (\mathbf{u} - \bar{\mathbf{u}}) \tag{5}$$

Subsequently, this can be expressed in a state-space form:

$$\dot{\mathbf{x}} = \mathbf{A}_c \mathbf{x} + \mathbf{B}_c \mathbf{u} + \mathbf{g}_c \tag{6}$$

The specific state-space model is shown below:

$$\begin{aligned} \begin{bmatrix} \dot{p}_x \\ \dot{p}_y \\ \dot{v} \\ \dot{\phi} \end{bmatrix} = & \underbrace{\begin{bmatrix} 0 & 0 & -\bar{v}\sin\bar{\phi} & \cos\bar{\phi} \\ 0 & 0 & \bar{v}\cos\bar{\phi} & \sin\bar{\phi} \\ 0 & 0 & 0 & \frac{\tan\bar{\delta}}{L} \\ 0 & 0 & 0 & 0 \end{bmatrix}}_{\mathbf{A}_c} \begin{bmatrix} p_x \\ p_y \\ v \\ \phi \end{bmatrix} \\ & + \underbrace{\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \frac{\bar{v}}{L\cos^2\bar{\delta}} & 1 \\ 1 & 0 \end{bmatrix}}_{\mathbf{B}_c} \begin{bmatrix} \alpha \\ \delta \end{bmatrix} + \underbrace{\begin{bmatrix} \bar{v}\bar{\phi}\sin\bar{\phi} \\ -\bar{v}\bar{\phi}\cos\bar{\phi} \\ 0 \\ \frac{\bar{v}\bar{\delta}}{L\cos^2\bar{\delta}} \end{bmatrix}}_{\mathbf{g}_c} \end{aligned} \tag{7}$$

where,

- $\mathbf{A}_c$ is continuous-time state matrix,
- $\mathbf{B}_c$ is continuous-time control matrix,
- $\mathbf{g}_c$ is continuous-time linearization remainder,
- $\bar{v}, \bar{\delta}$ and $\bar{\phi}$ are states or inputs at each time step.

To implement the LTV model in a discrete-time control system, the forward Euler method is used:

$$\frac{\mathbf{x}_{k+1} - \mathbf{x}_k}{T_s} = \mathbf{A}_c \mathbf{x}_k + \mathbf{B}_c \mathbf{u}_k + \mathbf{g}_c \tag{8}$$

This leads to the discrete-time state-space model:

$$\mathbf{x}_{k+1} = \underbrace{(\mathbf{I} + T_s \mathbf{A}_c)}_{\mathbf{A}_d} \mathbf{x}_k + \underbrace{T_s \mathbf{B}_c}_{\mathbf{B}_d} \mathbf{u}_k + \underbrace{T_s \mathbf{g}_c}_{\mathbf{g}_d} \tag{9}$$

$$\mathbf{x}_{k+1} = \mathbf{A}_d \mathbf{x}_k + \mathbf{B}_d \mathbf{u}_k + \mathbf{g}_d \tag{10}$$

where,

- $\mathbf{A}_d$ is discrete-time state matrix,
- $\mathbf{B}_d$ is discrete time control matrix,
- $\mathbf{g}_d$ is discrete-time linearization remainder,
- T_s is the sampling time for the discretization.

3) Reasons for Choosing LTV: The decision to select the LTV model is primarily due to its capability to simplify complex nonlinear dynamics into a more manageable linear framework. This simplification enhances the model's analytical tractability, simplifying the design of control systems. The adaptability of the LTV system to fluctuating operating conditions, coupled with its compatibility with conventional linear control methodologies, further validates its selection. Although linearization introduces some approximation error, the overall benefits in terms of implementation simplicity and robust control performance outweigh these concerns, making LTV a preferred choice for the bicycle model in vehicle control.

D. MPC Objective Function and Constraints

The MPC objective function is formulated as a quadratic cost function to be minimized. The objective function is given by:

$$J_N(\mathbf{x}_k, \mathbf{u}_k) = \sum_{k=1}^{N} \left\| \mathbf{x}_k - \mathbf{x}_k^{\text{ref}} \right\|_{\mathbf{Q}_p}^2 + \sum_{k=0}^{N-1} \left\| \mathbf{u}_k \right\|_{\mathbf{Q}_u}^2 \tag{11}$$

where,

- $\mathbf{x}_k$ is the states of the system at time k,
- $\mathbf{u}_k$ is the control input at time k,
- $\mathbf{x}_k^{\text{ref}}$ is the reference of trajectory k,
- N is the prediction horizon of the model,
- $\mathbf{Q}_p$ and $\mathbf{Q}_u$ are weight matrices for the state and control, respectively.

Constraints:

Constraints are applied to guarantee that the optimization problem adheres to plausible operational limits. Specifically, to mitigate against excessive acceleration and oversteering, the permissible acceleration and steering angles for the vehicle are limited between $-1\,\text{m/s}^2$ and $1\,\text{m/s}^2$, and $-\frac{\pi}{6}$ to $\frac{\pi}{6}$ radians, respectively. Consequently, the cost function governing the MPC is formulated as follows:

$$\begin{aligned} \underset{\mathbf{x}_k, \mathbf{u}_k \; \forall k \in [1,N]}{\text{minimize}} \quad & J_N(\mathbf{x}_k, \mathbf{u}_k) \\ \text{subject to} \quad & -\frac{\pi}{6} \leq \delta[k] \leq \frac{\pi}{6}, \; \forall k \in [1, N] \\ & -1 \leq a[k] \leq 1, \; \forall k \in [1, N] \\ & \mathbf{x}_{k+1} = \mathbf{A}_d \mathbf{x}_k + \mathbf{B}_d \mathbf{u}_k + \mathbf{g}_d, \; \forall k \in [1, N] \end{aligned} \tag{12}$$

V. OPTIMIZATION PROBLEM AND OVERALL CONTROL ARCHITECTURE

For the results of highway V2V lane merging, the construction of a path planner is the first necessity, and road conditions and vehicle positions need to be considered to solve this complex quadratic programming (QP) problem, as it requires repeated calculations every time instance. Secondly, an overall control architecture needs to be constructed to achieve road merging for real-time communication on highways.

A. Cost Function Design

To ensure an optimal balance between dynamic response and safety, the cost function J_p is formulated, which is minimized to determine the vehicle's optimal trajectory. The cost function J_p is a weighted sum of the components highlighted in the following subsections:

1) Penalty on Acceleration and Velocity: To ensure a comfortable driving experience, it is crucial to regulate the vehicle's lateral and longitudinal accelerations, preventing abrupt variations. Moreover, at the end of the driving phase, achieving a lateral velocity close to zero and a longitudinal velocity approximating the desired velocity is essential for safety and comfort. Consequently, the penalties for deviations in velocity and acceleration are quantified in Equation 13.

This cost function, J_1, is formulated to minimize discomfort by penalizing sharp accelerations and deviations from the desired velocities, thereby enhancing the overall driving experience.

$$\begin{aligned} J_1 =& p_{\text{a}} \sum_{k=0}^{M} (a_s[k])^2 + p_{\text{b}} \sum_{k=0}^{M} (a_n[k])^2 \\ &+ p_{\text{c}} \sum_{k=0}^{M} (v_s[k] - v_{\text{nominal}})^2 + p_{\text{d}} (v_n[M])^2 \end{aligned} \tag{13}$$

where,

- J_1 denotes the cost function of acceleration and velocity,
- a_s denotes the longitudinal acceleration,
- a_n denotes the lateral acceleration,
- v_s denotes the longitudinal velocity,
- v_n denotes the lateral velocity,
- M denotes the prediction horizon of the planner,
- v_{nominal} denotes the target terminal velocity,
- p_a, p_b, p_c, p_d are coefficients for penalizing deviations in longitudinal acceleration, lateral acceleration, longitudinal velocity, and lateral velocity, respectively.

2) Penalty for Deviating from the Centerline: When a vehicle is in motion, it is crucial not only to monitor its velocity but also to ensure that it remains centered within its lane or promptly realigns to the center after a lane change. Therefore, it is necessary to implement a penalization mechanism for any deviation between the vehicle's lateral position and the lane's centerline.:

$$J_2 = p_e (p_y[M] - n_{\text{center}})^2 \tag{14}$$

where,

- J_2 is the cost function for centerline deviation,
- p_e is the deviation penalty coefficient,
- n_{center} is the lane centerline's lateral position.

3) Penalty for Proximity to Vehicles and Road Edges: As the scenario involves highway lane merging, the cost function must account for vehicle-to-vehicle distances to prevent collisions with other vehicles. Furthermore, vehicles must also avoid collisions with the road's edge. To achieve this, a nonlinear function, $tanh$, is chosen to progressively increase penalties for both inter-vehicle distances and distances between vehicles and the road.

The function T_k is designed to impose a penalty proportional to the distance between the subject vehicle and other vehicles on either side. In contrast, R_k and L_k calculate two critical distances: the gap between the vehicle and the edge of its current lane, and the distance to the edge of the target lane where it intends to change lanes. By incorporating T_k, R_k, and L_k into the cost function J_3 and evaluating these parameters, the system effectively minimizes the risk of collisions during lane-changing maneuvers.

$$J_3 = \sum_{k=0}^{M} \Big[T_k + L_k + R_k \Big] \tag{15}$$

T_k, L_k and R_k are defined as:

$$\begin{aligned} T_k =& (\tanh(\frac{p_{\text{slope\_x}}}{2}(p_{\text{min\_s}}^2 - (p_x[k] - p_{\text{paths}}[k + (M+1)])^2)) \\ &+ 1) p_{\text{paths\_weight}} \cdot \frac{1}{2}(\tanh(p_x[k] - p_{\text{switch}}[0]) + 1) \end{aligned} \tag{16}$$

$$\begin{aligned} R_k =& 10 \cdot \sum (\max(\frac{P_{\text{road}}[0]}{2}(\tanh(P_{\text{road}}[1] \cdot (p_x[k] - P_{\text{road}}[2])) + 1) \\ &+ P_{\text{road}}[3] - p_y[k] + \text{n\_margin\_bottom}, 0)^2) \end{aligned} \tag{17}$$

$$\begin{aligned} L_k =& 10 \cdot \sum (\max(p_y[k] - (\frac{P_{\text{road}}[4]}{2}(\tanh(P_{\text{road}}[5] \cdot (p_x[k] - P_{\text{road}}[6])) \\ &+ 1) + P_{\text{road}}[7]) + \text{n\_margin\_top}, 0)^2) \end{aligned} \tag{18}$$

where,

- J_3 is the cost function for proximity to vehicles and road Edges,
- T_k is the penalty function for proximity to vehicles,
- L_k is the penalty function for proximity to left road Edges,
- R_k is the penalty function for proximity to right road Edges,
- $P_{\text{road}}[i]$ represents coefficients of the road,
- $P_{\text{path}}[i]$ represents the interpolated trajectory,
- $p_{\text{paths\_weight}}$ represents the path penalty coefficient,
- $p_{\text{slope\_x}}, p_{\text{min\_s}}$ define minimum distance between two vehicles,
- n_margin_top and n_margin_bottom indicates the distance to the road edge.

The final cost function is derived by integrating Equations 13, 14, and 15 into Equation 19. This cost function takes the form of a QP function, characterized by its convex nature. By solving this cost function within the constraints, the optimal trajectory can be obtained.

$$\begin{aligned} \min_{\mathbf{x}_k,\, \mathbf{u}_k, \forall k \in [1,M]} \quad & J_p(\mathbf{x}_k, \mathbf{u}_k) = J_1 + J_2 + J_3 \\ \text{subject to} \quad & -1 \le a_n[k] \le 1,\ \forall k \in [1, M] \\ & -1 \le a_s[k] \le 1,\ \forall k \in [1, M] \\ & 0 \le v_s[k] \le 35,\ \forall k \in [1, M] \\ & Road_{left} \le p_y[k] \le Road_{right},\ \forall k \in [1, M] \end{aligned} \tag{19}$$

where,

- a_n is the lateral acceleration for vehicle i,
- a_s is the longitudinal acceleration for vehicle i,
- v_s is the longitudinal velocity for vehicle i,
- $Road_{left}, Road_{right}$ are boundaries of the road.

B. Algorithm and Control Architecture

As illustrated in Fig. 3 and explained in Algorithm 1, the algorithm begins with an essential predictive analysis. This analysis enables vehicles in the CARLA simulation to foresee potential collisions over a 30-step horizon (M=30). The algorithm's design is particularly focused on the unique requirements of cars and trucks, ensuring tailored responses to various traffic scenarios.

For cars, a prediction of potential collision leads to an immediate strategy shift. They swiftly recalibrate their paths, formulating alternative trajectories through interpolation techniques. These new trajectories are crucial for avoiding collisions without significantly deviating from the original routes and are managed by the MPC system.

Trucks, in contrast, continue on their set paths but adjust their velocities in response to traffic conditions, guided by the IDM. This differentiation in response between cars and trucks is a key feature of the algorithm, facilitating effective traffic management in the simulation.

The modified paths and velocities are then executed within the CARLA environment. This stage is crucial for translating planned adjustments into actual vehicle movements, taking into account the dynamic nature of driving environments. The algorithm maintains a continuous feedback loop, where the real-time operational states of

the vehicles inform ongoing adaptive planning and control adjustments. This ensures the smooth integration of the vehicles within the simulation, balancing safety and efficiency.

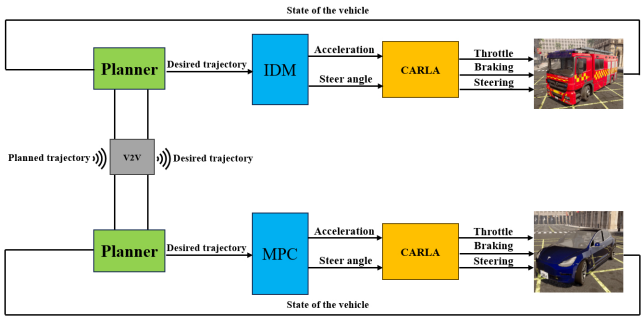

Fig. 3: The Control Architecture

Algorithm 1: Multi-Car Cooperative Collision Avoidance

Data: Initial positions and velocities of all vehicles
Result: Planned trajectories for each vehicle

Procedure `PlanTrajectory(`*vehicle*`)`
  Determine the current trajectory of the vehicle
  Communicate with other vehicles to share trajectory information
  if *vehicle is a car and collision detected with another vehicle* **then**
    Re-plan trajectory for the car using the cost function
    Assign re-planned trajectory as *planned trajectory*
  else if *vehicle is a truck* **then**
    `AdjustSpeedIDM(`*truck*`)`
    Maintain the current trajectory for the truck
  else
    Continue on Planned Trajectory

Procedure `AdjustSpeedIDM(`*truck*`)`
  Adjust truck's velocity based on Intelligent Driver Model (IDM)

Procedure `TrackTrajectory(`*vehicle*`)`
  if *vehicle is a car* **then**
    Track trajectory using Model Predictive Control
  else if *vehicle is a truck* **then**
    Track trajectory using velocity from IDM
  Send real-time status back to the planner

while *vehicle in the system* **do**
  `PlanTrajectory(`*vehicle*`)`
  `TrackTrajectory(`*vehicle*`)`

VI. Evaluation of Implementation

A. Setup in CARLA

In the initial phase of this study, an appropriate map that accurately represents the intended scenario is crucial. The CARLA simulator, equipped with a variety of pre-designed maps featuring diverse layouts and traffic conditions, offered several options. Among these, *Town06* is particularly noteworthy for its extended stretch of road, closely aligning with the project's requirements. Consequently, this map, specifically the mentioned road section, is selected for the lane merging scenario, as depicted in Fig. 4.

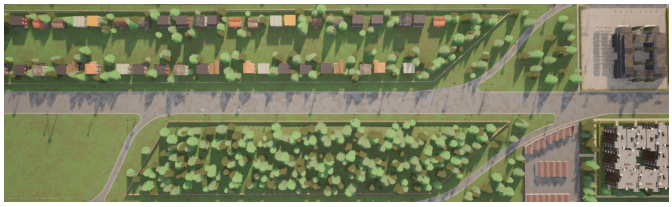

Fig. 4: The test scenario under evaluation.

Further, the study required the selection of specific vehicle models to represent the car and the truck in the simulation. The CARLA simulator offers an array of pre-configured vehicle models. For this study, a Tesla Model 3 is chosen to represent the car, and a firetruck is selected for the truck role. These vehicles are illustrated in Fig. 5.

Fig. 5: Vehicles utilized in the study.

The scenario depicted in Fig. 6 represents the initial setup for tuning and testing the Model Predictive Control (MPC) algorithm within the CARLA simulation environment. This arrangement, with the truck located in the lower right straight road and the car positioned on the ramp, was specifically chosen to facilitate a comprehensive evaluation of the MPC's performance in complex driving scenarios, such as lane merging.

This setup is a crucial component of the study, as it provides a platform for refining the MPC parameters and assessing its responsiveness and effectiveness in real-time vehicle control scenarios. The selected configuration facilitates in-depth analysis of the MPC's interaction within the CARLA simulation framework, enabling precise adjustments to the controller to further enhance the overall performance of the algorithm.

Fig. 6: MPC simulation in CARLA

B. Fine Tunning

The Q_p matrix in Equation 11 is critical in MPC as it directly influences the weightage of states in the cost function, thereby affecting the control performance.

1) Tuning Methodology: The tuning of the Q_p matrix was performed iteratively, using a combination of manual tuning and performance-based adjustments.

2) Evaluation Criteria: The effectiveness of the MPC was evaluated based on the following key criteria:

- **Trajectory Accuracy**: Measured by the deviation from the predefined path, assessing how precisely the vehicle follows the desired trajectory under varying Q_p configurations.
- **Stability and Overshoot**: Analyzing the vehicle's stability and tendency to overshoot, particularly in scenarios involving rapid direction or velocity changes.

3) Tuning Process and Results: During the initial trials, adjustments to the weightings of state positions p_x and p_y in the Q_p matrix were found to significantly influence the vehicle's trajectory accuracy and stability. Higher weights led to a more responsive control but also increased the tendency for overshoots. The outcomes of various tuning configurations are depicted in Fig. 7, which illustrates four different scenarios labeled from (a) to (d). These scenarios correspond to Q_p values set as $q_{a1} = q_{a2} = 0.01$, $q_{b1} = q_{b2} = 0.1$, $q_{c1} = q_{c2} = 1$, and $q_{d1} = q_{d2} = 100$, alongside constant values of $q_{x3} = 0.5$ and $q_{x4} = 0.5$.

The results showcase that the configuration in Fig. 7.c, with $Q_p = \text{diag}(1.0, 1.0, 0.5, 0.5)$, strikes an optimal balance between trajectory accuracy and stability. On the other hand, Fig. 7.a reveals the drawbacks of excessively low Q_p values, leading to a significant deviation from the intended trajectory due to inadequate control response. In contrast, overly high Q_p values, as observed in Fig. 7.d, result in increased instability and overshooting, causing the vehicle to stray from the desired path.

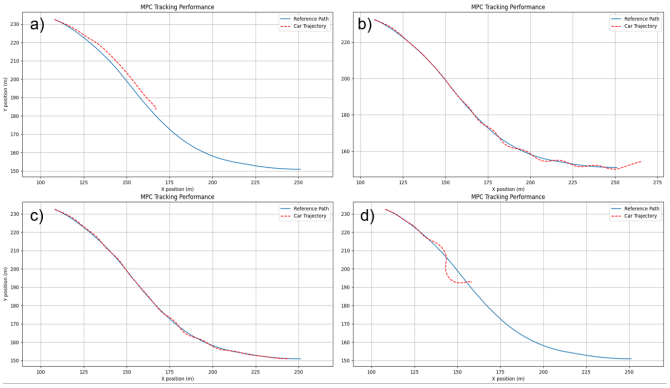

Fig. 7: The performance of trajectory tracking utilizing different Q_p.

C. Final Simulation

To accurately simulate the intended scenario, a simulation was done inside of Python completely isolated from the CARLA simulator. The scenario involves lane merging on a highway ramp, featuring a truck and a car. In this simulation, the truck's initial velocity is set at 25 m/s, while the car's initial velocity is 10 m/s. Fig. 8 illustrates the results of this simulation. Notably, the car's movement is controlled by MPC using a kinematic model, whereas the truck's behavior is governed by the IDM.

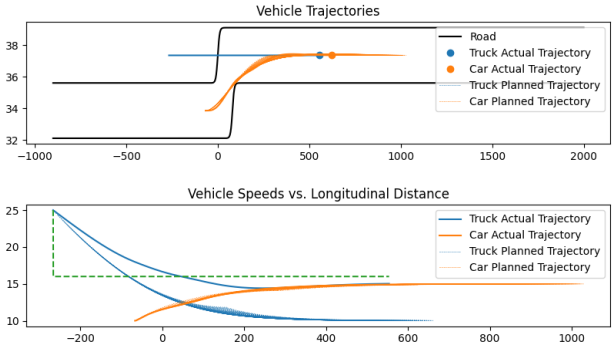

Fig. 8: Python Simulation for lane merging

Lane merging necessitates real-time solutions, enforcing strict solution time requirements for both MPC and trajectory planning. In this scenario, the control interval is limited to 0.02 seconds. Calculations indicate that the MPC (N=6) occupies approximately 0.008 seconds, and the trajectory planner requires about 0.02 seconds. It's important to note that, in the worst-case scenario, the planner may need up to 0.04 seconds for computation. Consequently, this results in a total solution time of roughly 0.048 seconds, already exceeding the specified control interval without even accounting for the operational time of the main program. This implies that, under certain conditions, the computation time of the trajectory planner and controller could exceed the length of the entire control interval.

This discrepancy in timing presents significant challenges for obtaining timely and accurate control instructions. Delays in control response can lead to a loss of control over the steering angle, a critical aspect of lane merging maneuvers. When coupled with high operational velocities, such delays directly contribute to failures in the CARLA simulation, as illustrated in Fig. 9.

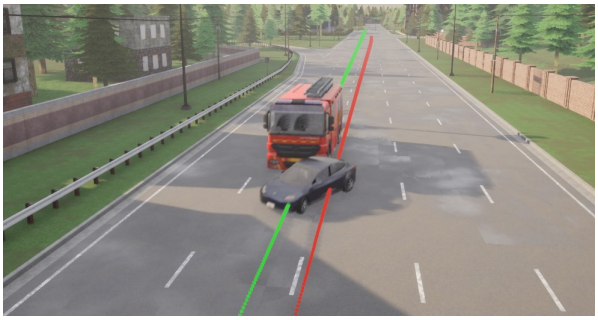

Fig. 9: Failure caused by real-time solving.

To address the issue of control interval in real-time computational demands, the study initially considered increasing the sampling time to 0.1 s to allow the controller more time for system management. However, this adjustment resulted in larger linearization errors, leading to significant discrepancies between the state-space model and the actual system behavior. These errors were substantial enough to cause collisions in simulations.

Due to these challenges, the study pivoted to an alternative, offline solution strategy. Results computed in Python are stored in a text file, serving as a predefined trajectory for vehicle tracking within the CARLA simulation environment. This approach mitigates real-time computational limitations. Moreover, considering vehicle kinematic constraints at higher velocities, the maximum velocity for both truck and car in the simulation is capped at 20 m/s, with a cruising velocity of 15 m/s for optimal performance. Fig. 10 shows the CARLA simulation's initialization phase, setting initial velocities at 10 m/s for the car and 20 m/s for the truck, aiding in effective trajectory tracking for lane merging.

Fig. 10: CARLA trajectory tracking initialization

Fig. 11 and Fig. 12 display the outcomes of these maneuvers. Here, the car's actual trajectory is depicted by a red line and the truck's by a green line. This visual representation clearly demonstrates the successful execution of the lane merging operation, as both vehicles follow their respective trajectories while converging into a single lane.

Fig. 11: Lane merging result in CARLA

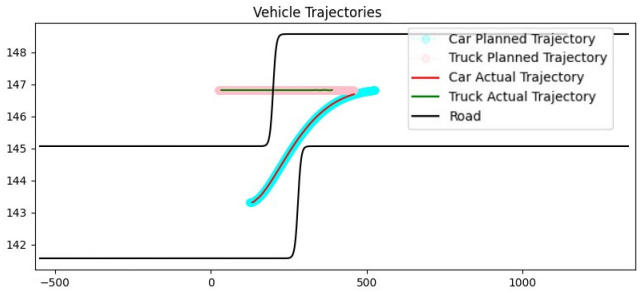

Fig. 12: Comparison of the planned and actual trajectory

VII. Conclusion

The objective of this research is to address the challenge of lane merging on highways. The truck was implemented using IDM, whilst the more advanced bicycle kinematics model was applied to the car. An optimization problem targeting the highway merging issue was formulated and addressed in conjunction with the CARLA simulation environment. However, challenges were encountered due to compatibility issues between the planning of the online road merging and the CARLA control system, resulting in difficulties in vehicle control. These issues occurred since the computations took longer than the time step in CARLA, thus preventing the control input from being applied in time. To overcome this issue, an offline solution was chosen instead. This change in strategy involved pre-computing the merging trajectories using the same optimization framework, but without the real-time constraints imposed by the CARLA control system. The vehicles' movements were then guided by these pre-determined trajectories, effectively achieving road merging through a process of trajectory tracking.

VIII. Further Work

For future tests, integrating a Proportional-Integral (PI) controller following the MPC controller is suggested. The PI controller can be utilized to eliminate steady-state errors, thereby enhancing the robustness and control precision of the overall system. Once finely tuned, this integration could further mitigate the disruptive factors within the CARLA environment that affect the vehicle model.

The cost function should be simplified or softened to reduce the computational time for the Quadratic Programming (QP) optimal problem solver and the MPC controller. This will enable quicker determination of optimal solutions, allowing CARLA to receive instantaneous messages from the controller.

To assess the algorithm's performance in a more realistic simulation, noise could be introduced to challenge the controller. This disturbance could take the form of Gaussian noise, or, for a greater challenge, more complex non-ideal noise, which more closely mimics real-life conditions. In such scenarios, introducing a robust MPC could be advantageous for addressing these uncertainties.

References

[1] Center for Sustainable Systems, University of Michigan, *Autonomous vehicles factsheet*, Pub. No. CSS16-18, 2023. [Online]. Available: https://css.umich.edu/publications/factsheets/mobility/autonomous-vehicles-factsheet.

[2] European Center for Information and Communication Technologies, *Eurofot- bringing intelligent vehicles to the road*, 2012. [Online]. Available: https://www.eurofot-ip.eu/en/welcome_to_eurofot.htm.

[3] European Center for Information and Communication Technologies, *Adaptive - objectives*, 2016. [Online]. Available: https://www.adaptive-ip.eu/index.php/objectives.html.

[4] European Center for Information and Communication Technologies, *About l3pilot - objectives*, 2018. [Online]. Available: https://l3pilot.eu/about.

[5] European Center for Information and Communication Technologies, *Widespread and continuous operational design domains (odds) on european roads*, 2021. [Online]. Available: https://www.hi-drive.eu.

[6] D. Bellan and C. Wartnaby, "Decentralized cooperative collision avoidance for automated vehicles: A real-world implementation," in *2020 IEEE Intelligent Vehicles Symposium (IV)*, 2020, pp. 487–494. DOI: 10.1109/IV47402.2020.9304736.

[7] M. Treiber, A. Hennecke, and D. Helbing, "Congested traffic states in empirical observations and microscopic simulations," *Physical review E*, vol. 62, no. 2, p. 1805, 2000.

Appendix

A YouTube playlist with some demonstrations of the scenario.

The code repository of this project on github