Fast Collision-Free Multi-Vehicle Lane Change Motion Planning and Control Framework in Uncertain Environments

Tianhao Liu, *Student Member, IEEE,* Runqi Chai, *Senior Member, IEEE,* Senchun Chai, *Senior Member, IEEE,* Farshad Arvin, *Senior Member, IEEE,* Jinning Zhang, and Barry Lennox, *Senior Member, IEEE*

***Abstract*—In this work, we focus on the design, test and validation of a hierarchical control framework capable of optimizing lane change trajectories and steering the motion of multiple automated guided vehicles (AGVs) in an uncertain environment. In the upper-level maneuver planning phase, a convex feasible set-based real-time optimization algorithm is adopted to plan the optimal motion trajectories for AGVs. The main novelty of this approach lies in its optimization process, where a sequence of convex feasible sets around the current solution is iteratively constructed such that the non-convex collision avoidance constraints can be approximated. Subsequently, an improved sequential convex programming (SCP) algorithm is designed and applied to reshape the current maneuver trajectory in the pre-constructed convex feasible sets and reduce the error caused by successive linearization of vehicle kinematics and constraints. The planned lane change trajectories are then provided to the lower-level motion controller, where a deep reinforcement learning (DRL)-based collision-free tracking control method is established and applied onboard to produce the control commands. This approach has the capability to deal with unexpected obstacles (e.g., those that suddenly appear around the vehicle). The proposed training method integrates a consensus algorithm with actor-critic deep reinforcement learning to allow multi-agent training to achieve faster training speed and improved performance compared with single-agent training. The feasibility and effectiveness of the proposed design are verified by carrying out simulation case studies. Moreover, the validity of the designed hierarchical control framework is further confirmed by executing hardware-in-the-loop tests.**

***Index Terms*—Multi-vehicle lane change, automated guided vehicles, convex feasible sets, sequential convex programming, deep reinforcement learning, unexpected obstacles.**

I. Introduction

AUTOMATED guided vehicles refer to one kind of intelligent ground vehicle capable of sensing the environment, planning the path of movement, and maneuvering without human input [1]. They have been widely researched during the last decade due to their extensive applications in commercial, industry, and military sectors. The core aim of the technology is to use it to replace manned vehicles with a lower risk and more economically attractive alternative. Among different automatic driving/maneuvering modes, one of the most typical behaviors is lane change movement. Serious on-road accidents may occur if a lane change maneuver is performed in an unsafe way. Consequently, a promising decision-making unit, responsible for planning a safe lane change motion trajectory, is of particular importance to enhance mobility, alleviate traffic congestion, and guarantee road safety.

T. Liu, R. Chai, and S. Chai are with the school of Automation, Beijing Institute of Technology, Beijing, 100081, China. R. Chai is also with the School of Aerospace, Transport and Manufacturing, Cranfield University, Cranfield MK43 0AL, U.K. e-mail: (tianhao.liu@bit.edu.cn), (r.chai@bit.edu.cn), (chaisc97@bit.edu.cn).

F. Arvin is with the Department of Computer Science, Durham University, DH1 3LE Durham, U.K. e-mail: (farshad.arvin@durham.ac.uk).

J. Zhang is with the School of Engineering, University of Leicester, Leicester, LE1 7RH, UK. e-mail: (jz388@leicester.ac.uk).

B. Lennox is with the Department of Electrical and Electronic Engineering, The University of Manchester, M13 9PL Manchester, U.K. e-mail: (barry.lennox@manchester.ac.uk). (*Corresponding Author: Runqi Chai*)

A. Related works

In recent years, researchers have devoted significant effort to developing intelligent vehicle lane change motion planners for single vehicles [2], [3]. For example, the authors of [2] developed a simultaneous trajectory planning and tracking control scheme able to change lanes in the presence of static on-road obstacles. A low-complexity lane change maneuver algorithm, capable of determining an optimal transfer time and producing reliable control actions, was introduced in [3]. However, these contributions cannot be easily extended to address the multi-vehicle cases. This is due to the fact that by taking into account the motion of surrounding vehicles, the maneuver corridor for each vehicle is likely to be significantly narrowed. As a result, lane change behaviors should be planned cooperatively, thus stimulating the development of motion planning for multi-vehicle platforms.

Recently, various trajectory planning and control strategies have been proposed to tackle different maneuver scenarios for multiple vehicles. A common class of strategies is based on roadmap search, such as rapidly exploring random tree-based method [4], probabilistic roadmap-based method [5], and A*-based method [6], etc. However, such strategies can only provide feasible paths for the vehicles and are not suitable when certain metrics need to be optimized.

Alternatively, the entire maneuver process can be formulated as an optimization model and well-developed optimization methods can be applied to search for the optimal maneuver solutions [7]. Optimization-based algorithms not only seek a feasible solution to trajectory planning problems but also optimize user-defined objectives. In [8], the authors developed an interior point-based decision-making scheme for a multi-AGV system. In their work, the optimal motion trajectory was obtained by solving a large-scale nonlinear programming (NLP) problem directly. Moreover, the lateral vehicle guidance task was formulated as a linear time-varying model predictive control (MPC) problem in [9], where the vehicle's velocity is treated as a time-varying parameter. A distributed maneuver planning framework is proposed in [10]. This framework utilizes distributed MPC to calculate a collision-free trajectory and is capable of improving traffic throughput. However, either solving the NLP problem directly or formulating it in an MPC approach suffers from a high computational burden and there is no guarantee that a solution can be found. On the other hand, convex programming can be effectively solved in polynomial time [11] and has had many successful applications in solving trajectory optimization problems [12]. The authors in [13] designed a convex quadratic programming-based model predictive control scheme for the collision-free navigation problem. It was verified that this scheme improves the real-time

performance of the motion planner. However, this approach is only applicable when the mismatch between the convexified model and the original one is negligible.

Moreover, the traffic environment might be uncertain in real-world scenarios. For example, an object may suddenly appear in the environment and create a collision hazard. With the development of perception technology, the key obstacle information can be extracted using LiDAR [14] or camera [15] data. Then well-developed obstacle avoidance methods such as dynamic window approach (DWA) [16] and artificial potential field [17] can be applied. Recently, the application of deep reinforcement learning (DRL)-based techniques in AGV motion planning and control has received increasing attention [18]. DRL is model-free and no map information is required. It produces actions that maximize task-level rewards based on sensor data, thus making it an end-to-end approach [19], [20].

B. Motivation, Goals, and Contributions

The aforementioned NLP-based motion planners/controllers can be computationally intractable when addressing complex multi-vehicle maneuver tasks. While convex programming-based methods are effective, vehicle equations of motion and collision avoidance constraints render the trajectory optimization problem nonconvex, and therefore cannot be solved in typical convex optimization frameworks. On the other hand, existing DRL-based methods for single vehicle local planning and control may consume lengthy training time when applied to multi-vehicle tasks. Specifically, each DRL controller is trained for individual vehicles and no information is shared between any of them, which limits the training performance.

To address the aforementioned open issues and enhance road safety, we design, test, and validate a hierarchical motion planning and control framework capable of producing collision-free maneuvers for multi-AGV systems with consideration of unmodeled obstacles. To summarize, the main contributions and novelties of the proposed research are highlighted in the following three aspects:

1) The non-convex collision avoidance constraints are made convex by the convex feasible set algorithm [21] and the original nonlinear optimization model is reformulated into a convex one, which requires less time to be solved. A sequential convex programming (SCP) algorithm with adaptive trust region radii which successively solves the convex problem is designed to reduce the approximation error. To ensure the algorithm starts from an initial collision-free reference, an extended A* algorithm is proposed to generate an initial feasible trajectory to start the optimization process from a collision-free trajectory.
2) Unlike existing SCP algorithms that accept the optimal solution from the previous iteration as the reference to the current iteration, a line search process between each iteration is introduced to accelerate convergence and further reduce the approximation error. A merit function is designed to measure how much the optimal solution to a convex subproblem violates the constraints of the original problem.
3) To deal with the sudden appearance of obstacles and steer the AGVs along the pre-optimized trajectories, a consensus-based DRL training algorithm that updates both actor and critic parameters simultaneously is proposed. Compared to the Deep Deterministic Policy Gradient (DDPG) algorithm, this approach achieves faster convergence speed and higher reward value.

C. Layout

The remaining sections of this article are organized as follows. Section II illustrates the original mathematical formulation of the multi-vehicle lane change problem. In Section III, the translation of the original optimization problem and the proposed sequential convex programming algorithm are presented. The DRL-based trajectory tracking scheme that is capable of avoiding unexpected obstacles in the lane change process is presented in Section IV. In Section V, simulation and HiL tests are conducted to demonstrate the effectiveness of the proposed hierarchical collision-free control framework. Finally, conclusions are summarised in Section VI.

II. Multi-AGV Lane Change Problem Description

In this section, the multi-vehicle lane change maneuver is characterized as a finite-horizon nonlinear trajectory optimization problem. The problem is to transfer multiple vehicles from their current lanes to the target lanes while removing collision risks between vehicles and minimizing the transfer time. Before presenting the optimization problem in detail, the vehicle variables, equations of motion, and different types of constraints are introduced.

TABLE I: Nomenclature

$p_i = (p_{ix}, p_{iy})$:	The position of rear axle midpoint of vehicle i
v_i, a_i:	The velocity and acceleration of vehicle i
θ_i, δ_i, η_i:	Heading angle, steering angle, and steering rate of vehicle i
L_r, L_w, L:	Vehicle's rear overhang, wheelbase, and total length
$I_a = \{1, \dots, N_a\}$:	Set of vehicle
$I_c = \{1, \dots, N_c\}$:	Set of circle that approximate the shape of vehicle
$C_{ik}, i \in I_a, k \in I_c$:	Center coordinate of the kth circle in vehicle i
$O_{ik}, i \in I_a, k \in I_c$:	Region occupied by the kth circle in vehicle i

A. Vehicle Kinematics

Suppose that there are N_a vehicles in the lane change maneuver process, the kinematic equations of motion for the i'th vehicle ($i \in I_a$) can be described as [22]

$$\begin{cases} \dot{p}_{ix}(t) = v_i(t)\cos(\theta_i(t)) \\ \dot{p}_{iy}(t) = v_i(t)\sin(\theta_i(t)) \\ \dot{\theta}_i(t) = \dfrac{v_i(t)\tan(\delta_i(t))}{L_w} \\ \dot{\delta}_i(t) = \eta_i(t) \\ \dot{v}_i(t) = a_i(t) \end{cases} \tag{1}$$

where the definition of the variables can be found in Table I. Denoting the state and control variables of the controlled vehicles as $x_i(t) = [p_{ix}(t), p_{iy}(t), \theta_i(t), \delta_i(t), v_i(t)]^T$ and $u_i = [\eta_i(t), a_i(t)]^T$, respectively. Equation (1) can be abbreviated as $\dot{x}_i(t) = f(x_i(t), u_i(t))$. During the lane change maneuver, the state and control variables should vary within certain regions and can be expressed as

$$x_i(t) \in [x^{\min}, x^{\max}], u_i(t) \in [u^{\min}, u^{\max}], \forall i \in I_a. \tag{2}$$

B. Boundary Constraints

The initial and terminal conditions of the lane change problem are as follows

$$\begin{gathered} p_{ix}(t_0) = p_{ix0},\ p_{iy}(t_0) = \mathrm{Init}_i, p_{iy}(t_f) = \mathrm{Des}_i. \\ [\theta_i(t_f), v_i(t_f), \eta_i(t_f), a_i(t_f)] = [\theta_i(t_0), v_i(t_0), \eta_i(t_0), a_i(t_0)] \end{gathered} \tag{3}$$

where the initial conditions of all state and control variables are known. Let $\{Y_1, Y_2, \dots\}$ be the coordinates of the line centers, the initial and the desired lanes are denoted by Init_i, $\mathrm{Des}_i \in \{Y_1, Y_2, \dots\}$, respectively. Equation (3) can be written compactly as $x_i(t_0) = x_{i0}$, $x_i(t_f) = x_{if}$, and $u_i(t_f) = u_i(t_0)$.

C. Collision Avoidance Constraints

To achieve the lane change maneuver, two types of collisions need to be avoided, namely i) collisions between the AGVs and the road barrier, and ii) collisions among the vehicles.

The collisions between the AGVs and the road barrier can be avoided by bounding the four corner points of the AGVs in the y axis,

which can be determined by the state variables $[p_{ix}(t), p_{iy}(t), \theta_i(t)]$ and the length and width of the vehicles. Denoting the coordinate of the corner points of AGV i in the y axis as y_{i1}, y_{i2}, y_{i3} and y_{i4}, the first type of collision avoidance can be formulated as

$$y_{i1}, y_{i2}, y_{i3}, y_{i4} \in [y_l, y_u], \forall i \in I_a \tag{4}$$

where y_l, y_u are the coordinates of the two road barriers.

To formulate the second type of collision avoidance, this article uses N_c circles, with the same radius r, placed evenly to approximate the rectangular vehicles [23], as shown in Fig. 1. The coordinate of the kth circle center in vehicle i at time t is

$$C_{ik}(t) = \begin{bmatrix} p_{ix}(t) \\ p_{iy}(t) \end{bmatrix} + \left(\frac{(k-\frac{1}{2})L}{N_c} - L_r \right) \begin{bmatrix} \cos(\theta_i(t)) \\ \sin(\theta_i(t)) \end{bmatrix}. \tag{5}$$

The collision avoidance constraint is given by

$$d(C_{ik}(t), O_{jl}(t)) \geq \underline{r} + r, \quad i < j, \ \forall i, j \in I_a, \ \forall k, l \in I_c \tag{6}$$

where $\underline{r} > 0$ is safety margin. The collision avoidance for each vehicle in $\mathbb{R}^2$ is achieved by ensuring that each circle located within it keeps a safe distance, i.e. $\underline{r} + r$, from any circles approximating the location of other vehicles.

D. Overall Optimization Formulation

With the AGV dynamic model and different constraints established, a collision-free optimal trajectory for the AGVs can be obtained by addressing the following optimization problem:

$$\begin{aligned} \min_{x_i, u_i, t_f} \quad & J_0 = t_f \\ \text{s.t.} \quad & \dot{x}_i(t) = f(x_i(t), u_i(t)), \forall i \in I_a, \forall t \in [0, t_f] \\ & x_i(t) \in [x^{\min}, x^{\max}], u_i(t) \in [u^{\min}, u^{\max}], \forall t \in [0, t_f] \\ & x_i(0) = x_{i0}, \ x_i(t_f) = x_{if}, \ u_i(t_f) = u_i(0), \forall i \in I_a \\ & C_{ik}(t) = p_i(t) + \left(\frac{(k-\frac{1}{2})L}{N_c} - L_r \right) \begin{bmatrix} \cos(\theta_i(t)) \\ \sin(\theta_i(t)) \end{bmatrix}, \\ & \quad \forall i \in I_a, \forall k \in I_c, \forall t \in [0, t_f] \\ & d(C_{ik}(t), O_{jl}(t)) \geq \underline{r} + r, \forall i < j \in I_a, \forall k, l \in I_c, \forall t \in [0, t_f] \end{aligned} \tag{7}$$

where J_0 is the objective function consisting of stage cost L and terminal cost Φ of all the AGVs. The decision variables are $[x_1^T, u_1^T, \ldots, x_{Na}^T, u_{Na}^T, t_f]^T$. This formulation allows us to fulfill different mission requirements including minimum lane changing time and ensure the smoothness of the obtained trajectory.

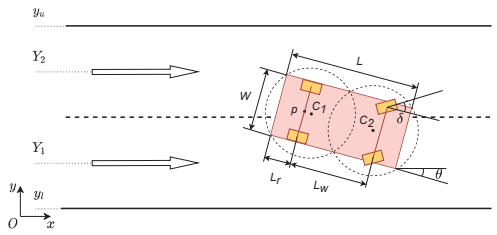


Fig. 1: Illustration of vehicle and task-related parameters

E. Communication Topology

The interaction topology of communication among N_a vehicles can be illustrated using an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ represents a vertex set $\mathcal{V} = \{1, 2, \cdots, N_a\}$ and $\mathcal{E}$ stands for an edge set $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$. The edge $(i, j) \in \mathcal{E}$, if the i^{th} and j^{th} agents are connected. The weighted connectivity matrix w also satisfies the following condition: (i) $w_{ij} \in [0, 1), \ \forall (i, j)$; (ii) $w_{ij} = w_{ji}, \ \forall (i, j)$; (iii)$\sum_{j=1}^{N} w_{ij} = 1, \ \forall i$.

III. Convex Feasible Set-based Algorithm

Currently, there are many off-the-shelf numerical optimizers available [7]. However, the existence of nonlinear dynamic constraints (1) and large scale non-convex constraints, (6) leads to low solution-finding efficiency when using such methods. In this section, the original formulation is converted into a convex optimization problem to reduce the computing demand in the optimization process. Note that the constraints (2), (4) are convex and (3) is linear, which can be directly used in the convex programming problem. Therefore, converting the problem to be convex includes linearizing the nonlinear equality constraints (1), (5), and relaxing the collision avoidance constraints (6) to convex constraints. After the process of making the problem convex, an SCP algorithm is introduced to address the converted lane change problem.

A. Linearization of Equality Constraints

The process of convexification begins with the linearization of system dynamics (1). Given the reference state and control variables $(\hat{x}_i, \hat{u}_i)$, the linearized system model can be written as

$$\dot{x}_i = A_i(\hat{x}_i, \hat{u}_i)x_i + B_i(\hat{x}_i, \hat{u}_i)u_i + c_i(\hat{x}_i, \hat{u}_i), \ \forall i \in I_a \tag{8}$$

where $c_i(\hat{x}_i, \hat{u}_i) = f(\hat{x}_i, \hat{u}_i) - A_i(\hat{x}_i, \hat{u}_i)\hat{x}_i - B_i(\hat{x}_i, \hat{u}_i)\hat{u}_i$ and the coefficients $A_i(\hat{x}_i, \hat{u}_i)$ and $B_i(\hat{x}_i, \hat{u}_i)$ are defined as

$$A_i(\hat{x}_i, \hat{u}_i) = \frac{\partial f(x,u)}{\partial x}\Big|_{\substack{x=\hat{x}_i \\ u=\hat{u}_i}}, \ B_i(\hat{x}_i, \hat{u}_i) = \frac{\partial f(x,u)}{\partial u}\Big|_{\substack{x=\hat{x}_i \\ u=\hat{u}_i}}. \tag{9}$$

The linearized system dynamics should also be discretized to formulate a static convex function. Define $A_i(n) := A_i(\hat{x}_i(n), \hat{u}_i(n))$, the discretized system model can be written as

$$\begin{aligned} & x_i(n+1) = x_i(n) + [A_i(n)x_i(n) + B_i(n)u_i(n) + c_i(n)] \times \Delta t, \\ & x_i(n) \in [x^{\min}, x^{\max}], u_i(n) \in [u^{\min}, u^{\max}], n = 0, 1, \ldots, N_t, \ \forall i \in I_a \end{aligned} \tag{10}$$

where $N_t + 1$ temporal nodes are used to split the time interval $[0, t_f]$ into equal small intervals of length Δt. The variables (x_i, u_i) at time t_n are denoted as $(x_i(n), u_i(n))$ with $x_i(0) = x_i(t_0)$ and $x_i(N_t) = x_i(t_f)$. Note that so far the bounds on state and control variables are imposed at each time instant n. Although only time-invariant bounds are considered in this article, time-varying variable bounds can be easily handled by imposing different bounding values at different time instants. This extension makes the proposed method applicable to a wider range of scenarios, such as varying road width.

In addition to nonlinear differential equation (1), the circle center equations (5) should also be converted to linear equality constraints. For each vehicle, (5) is equivalent to

$$C_{i1}(n) = \begin{bmatrix} p_{ix}(n) \\ p_{iy}(n) \end{bmatrix} + \left(\frac{L}{2N_c} - L_r \right) \begin{bmatrix} \cos(\theta_i(n)) \\ \sin(\theta_i(n)) \end{bmatrix} \tag{11}$$

$$C_{ik}(n) = C_{i1}(n) + \frac{(k-1)L}{\frac{L}{2} - N_c L_r} \left(C_{i1}(n) - \begin{bmatrix} p_{ix}(n) \\ p_{iy}(n) \end{bmatrix} \right), \ k \in I_c \backslash \{1\}. \tag{12}$$

Appending the first circle center coordinate C_{i1} to the state variables, (12) are linear with respect to the extended state variables $z_i = [x_i^T, C_{i1}^T]^T$. Furthermore, denoting $g(z_i) = C_{i1} - p_i - \left(\frac{L}{2N_c} - L_r \right) [\cos\theta_i, \sin\theta_i]^T$, (11) can be linearized at a reference point $\hat{z}_i$ as

$$g(\hat{z}_i) + \nabla g(\hat{z}_i)^T (z_i - \hat{z}_i) = 0. \tag{13}$$

B. Construction of Feasible Sets

Since collision-free among AGVs is realized by a series of distance constraints between the circle center and the circle in (6), the CFS algorithm proposed in [21] can be applied to calculate the convex feasible sets around a reference trajectory. Given a reference

trajectory $\hat{\boldsymbol{z}}$ obtained from the previous iteration, the circles $\hat{O}_{ik}$ and their centers $\hat{C}_{ik}$ can be determined according to (11), (12). For C_{ik} in AGV i, the region occupied by other vehicles are considered as obstacles. At time step n, define a distance function as

$$\phi_{jl}^n(x) := \|x - \hat{C}_{jl}(n)\|_2 - 2r - \underline{r},\ j > i,\ j \in I_a,\ l \in I_c. \tag{14}$$

$\phi_{jl}^n(C_{ik}(n)) \geq 0$ indicates that there is no collision between $O_{ik}(n)$ and $\hat{O}_{jl}(n)$. The convex feasible set for $C_{ik}(n)$ is defined as

$$F_{ik}(\hat{\boldsymbol{z}}(n)) := \bigcap_{j>i, j\in I_a, l\in I_c} \mathcal{F}_{jl}(\hat{\boldsymbol{z}}(n)) \subset \mathbb{R}^2 \tag{15}$$

where

$$\mathcal{F}_{jl}(\hat{\boldsymbol{z}}(n)) = \{C_{ik} : \phi_{jl}^n(\hat{C}_{ik}(n)) + \nabla\phi_{jl}^n(\hat{C}_{ik}(n))^T(C_{ik} - \hat{C}_{ik}(n)) \geq 0\} \tag{16}$$

is the convex feasible set of $C_{ik}(n)$ when avoiding $\hat{O}_{jl}(n)$. Taking $N_a = 3$ and $N_c = 1$ as an example, Fig. 2 illustrates the process of constructing the CFS. Then the collision avoidance constraint for AGV i along its trajectory can be represented as

$$z_i \in F_i(\hat{\boldsymbol{z}}) := \bigcap_{n=1}^{N_t} \bigcap_{k\in I_c} F_{ik}(\hat{\boldsymbol{z}}(n)) \subset \mathbb{R}^{2\times N_t}. \tag{17}$$

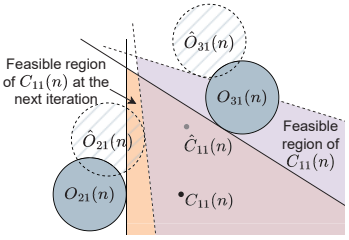


Fig. 2: Illustration of feasible sets

C. Trust Region Sequential Convex Programming

We have used linear approximation of (10) and (13) to establish the convex programming problem. It is well known that the linearization error becomes large when the true trajectory is far from the reference trajectory. Thus we introduce trust region constraints with varying radii to limit the difference between the planned and referenced state and control trajectories. With the trust region constraints and the convex constraints formulated in the last subsection, the original non-convex continuous optimization problem (7) can be rewritten as the following convex programming problem:

$$\begin{aligned}
\min_{z_i, u_i r_{iz}, t_{N_t}}\ & J_1 = t_{N_t} + w_z\|\boldsymbol{r}_z\|_2 \\
\text{s.t. } & \forall i \in I_a, n = 0, 1, \ldots, N_t \\
& x_i(n+1) = x_i(n) + [A_i(n)x_i(n) + B_i(n)u_i(n) + c_i(n)]\Delta t \\
& x_i(t_0) = x_{i0},\ x_i(t_f) = x_{if},\ x_i(n) \in [x^{\min}, x^{\max}] \\
& g(\hat{z}_i) + \nabla g(\hat{z}_i)^T(z_i - \hat{z}_i) = 0, \\
& u_i(t_f) = u_i(t_0),\ u_i(n) \in [u^{\min}, u^{\max}] \\
& C_{ik} = p_i + \left(\frac{(k-\frac{1}{2})L}{N_c} - L_r\right)\frac{C_{i1} - p_i}{\frac{L}{2N_c} - L_r},\ k \in I_c\backslash\{1\} \\
& z_i \in F_i(\hat{\boldsymbol{z}}) := \bigcap_{n=1}^{N_t}\bigcap_{k\in I_c} F_{ik}(\hat{\boldsymbol{z}}(n)) \\
& \|z_i(n) - \hat{z}_i(n)\|_2 \leq r_{iz}(n)
\end{aligned} \tag{18}$$

where $\boldsymbol{r}_z$ is the trust region radius which is optimization variable and w_z is a positive weighting parameter. It was shown that a sufficiently large w_z reduces the convexification error and alleviates the oscillation phenomenon in trust region-based methods [24]. Denoting $\boldsymbol{z} = [z_1^T, \ldots, z_{N_a}^T]^T$, $\boldsymbol{u} = [u_1^T, \ldots, u_{N_a}^T]^T$, the decision variables in this formulation are $\boldsymbol{d} = [\boldsymbol{z}^T, \boldsymbol{u}^T, \boldsymbol{r}_z^T, t_{N_t}]^T$.

In this article, a convex feasible set-based SCP algorithm is designed to solve the optimization problem (18). The overall algorithm consists of two layers. The outer layer generates the convex collision avoidance constraint (17) for (18). Every time the collision avoidance constraints are determined, the inner layer solves the problem (18) iteratively and the solution obtained in one iteration is regarded as the reference trajectory for the next iteration, that is, $\hat{\boldsymbol{z}}^{m+1} = \boldsymbol{z}^m$, where the superscript is the number of iterations. This iteration terminates when a certain stopping condition is achieved and the solution is used to update (17). The main steps are summarized in Algorithm 1. A line-search strategy between each iteration is established based on a metric function that evaluates the inconsistency caused by convexification. At time node n and iteration m, define the linearization error of equality constraints by

$$h_n(\boldsymbol{d}^m) = \begin{bmatrix} \sum_{i\in I_a} x_i^m(n+1) - x_i^m(n) - f(x_i^m(n), u_i^m(n))\Delta t \\ \sum_{i\in I_a} g(z_i^m(n)) \end{bmatrix} \tag{19}$$

and the convexification error of collision avoidance constraints by

$$g_n(\boldsymbol{d}^m) = \sum_{\substack{i\in I_a \\ j\in I_c}} \sum_{\substack{k>i, k\in I_a \\ l\in I_c}} 2r + \underline{r} - \|C_{ik}^m(n) - C_{jl}^m(n)\|_2. \tag{20}$$

The overall convexification error can be defined as

$$\ell(\boldsymbol{d}^m; \beta_1, \beta_2) = \beta_1 \textstyle\sum_{n=0}^{N_t} \|h_n(\boldsymbol{d}^m)\|_1 + \beta_2 \sum_{n=0}^{N_t} \|g_n^+(\boldsymbol{d}^m)\|_1 \tag{21}$$

where $g_n^+(\cdot) = \max\{g_n(\cdot), 0\}$ and $\beta_1, \beta_2 > 0$ are penalty factors. When a solution is obtained by solving (18), the constraint violation regarding the original problem (7) is quantified by the l_1 norm of the approximation error at all time instant. The line-search process is summarized in Algorithm 2.

Algorithm 1 Convex feasible set-based SCP algorithm

Input: Algorithm and task-related parameters $x_{i0}, x_{if}, N_t, N_c, \underline{r}$. The convergence threshold ϵ;
Step 1: Obtain initial trajectory $\boldsymbol{d}^0$ and set $m = 0$;
Step 2: Compute convex feasible sets $z_i \in F_i(\boldsymbol{z}^m), \forall i \in I_c$ and set $h = 0$, $\boldsymbol{d}^h = \boldsymbol{d}^m$;
Step 3: Perform the linearization (10), (13) with respect to $\boldsymbol{d}^h$;
Step 4: Address the convex optimization problem (18) and denote the obtained solution as $\boldsymbol{d}^{h+1}$;
Step 5: Set $h = h + 1$. If the stopping condition $\|\boldsymbol{d}^h - \boldsymbol{d}^{h-1}\|_2 \leq \epsilon$ is satisfied, set $m = m + 1$ and go to Step 6, else go back to Step 3;
Step 6: Set $\boldsymbol{d}^m = \boldsymbol{d}^h$. If the stopping condition $\|\boldsymbol{d}^m - \boldsymbol{d}^{m-1}\|_2 \leq \epsilon$ is satisfied, then output $\boldsymbol{d}^m$ as the optimal solution. Otherwise, perform Algorithm 2 and go back to Step 2.
Output: The optimal trajectory $\boldsymbol{d}^* = \boldsymbol{d}^m$.

Remark 1. Note that problem (18) is an approximation of the original problem (7). It is possible that a solution to (18) does not satisfy the original system equations and collision avoidance constraints in (7) and vice versa. Several strategies have been proposed to deal with the inconsistencies caused by convexification. The authors in [25] suggested adding unbounded virtual control variables to the linearized system equations and penalizing it in the objective. It is shown in [26] that a shrinking trust region can also reduce convexification error. The trust region constraint in (18) serves the same purpose. Nevertheless, in these methods, the reference to each iteration is the optimal solution from the previous iteration, which does not solve the original problem and thus can be modified. The proposed line search process modifies it along a decreasing direction of the merit function. Such a design may further reduce the approximation error and accelerate convergence.

Algorithm 2 Line-Search Process

1: **Input:** $\boldsymbol{d}^m$, $\boldsymbol{d}^{m-1}$. Line-search parameters β_1, β_2 and $0 < c_1 < c_2 < 1$;
2: Calculate $\xi^m = \boldsymbol{d}^m - \boldsymbol{d}^{m-1}$;
3: Calculate the directional derivative of the error function ℓ by

$$\Delta(\ell(\boldsymbol{d}^m;\beta_1,\beta_2);\xi^m) = \lim_{\epsilon\to 0}\frac{\ell(\boldsymbol{d}^m+\epsilon\xi^m;\beta_1,\beta_2)-\ell(\boldsymbol{d}^m;\beta_1,\beta_2)}{\epsilon}$$

4: Search α^m that satisfies the following condition:

$$\begin{aligned}&\ell(\boldsymbol{d}^m;\beta_1,\beta_2)+c_1\alpha^m\Delta(\ell(\boldsymbol{d}^m;w_1,w_2);\xi^m)\\&\le \ell(\boldsymbol{d}^m+\alpha^m\xi^m;\beta_1,\beta_2)\\&\le \ell(\boldsymbol{d}^m;\beta_1,\beta_2)+c_2\alpha^m\Delta(\ell(\boldsymbol{d}^m;\beta_1,\beta_2);\xi^m);\end{aligned}$$

5: Set $\boldsymbol{d}^m = \boldsymbol{d}^{m-1}+\alpha^m\xi^m$;
6: **Output:** The updated solution $\boldsymbol{d}^m$.

The suggested SCP algorithm requires an initial trajectory to start the optimization process, as shown in Step 1 of Algorithm 1. The initial value $p_i^0 = (p_{ix}^0, p_{iy}^0)$ is obtained by the A* algorithm, and the other variables at all time instants are computed iteratively by

$$\begin{aligned}\theta_i^0(n) &= \arccos\big(\frac{p_{ix}^0(n+1)-p_{ix}^0(n)}{\|p_i^0(n+1)-p_i^0(n)\|_2}\big)\\ v_i^0(n) &= \frac{\|p_i^0(n+1)-p_i^0(n)\|_2}{\Delta t}\\ \delta_i^0(n) &= \arctan\big(\frac{\theta_i^0(n+1)-\theta_i^0(n)}{\Delta t}\frac{L_w}{v_i^0(n)}\big)\\ \eta_i^0(n) &= \frac{\|\delta_i^0(n+1)-\delta_i^0(n)\|_2}{\Delta t}\\ a_i^0(n) &= \frac{\|v_i^0(n+1)-v_i^0(n)\|_2}{\Delta t}.\end{aligned} \tag{22}$$

The trajectories computed in this approach may violate the bounds on them and, if exist, the bounds on their difference. Therefore, a modification process is introduced to ensure the initial trajectory lies in the feasible region. The modification of $\eta_i^0(n)$ is given by

$$\eta_i^0(n) = \begin{cases}\eta^{\max} & \text{if } \eta_i^0(n) > \eta^{\max},\\ \eta^{\min} & \text{if } \eta_i^0(n) < \eta^{\min}.\end{cases} \tag{23}$$

and $a_i^0(n)$ are modified in similar form to (23). The modification of $v_i^0(n)$ is given by

$$v_i^0(n) = \begin{cases}v_i^0(n)+a^{\max}\Delta t & \text{if } \frac{v_i^0(n+1)-v_i^0(n)}{\Delta t} > a^{\max},\\ v_i^0(n)+a^{\min}\Delta t & \text{if } \frac{v_i^0(n+1)-v_i^0(n)}{\Delta t} < a^{\min},\\ v^{\max} & \text{if } v_i^0(n) > v^{\max},\\ v^{\min} & \text{if } v_i^0(n) < v^{\min},\end{cases} \tag{24}$$

and $\theta_i^0(n)$, $\delta_i^0(n)$ are modified similar to (24). Based on this initial trajectory generation strategy, which is referred to as E-A* algorithm in the following context, the sequential convex programming starts at a collision-free reference trajectory. Since some of the collision avoidance constraints become inactive and no longer affect the optimization process, the efficiency of the proposed algorithm is likely to be improved.

IV. DRL-based Collision-Free Tracking Control

In this section, we establish a DRL-based collision-free tracking control scheme that is capable of steering the AGV as it moves along the optimized trajectory $[\boldsymbol{z}^*, \boldsymbol{u}^*]^T$, and avoiding any suddenly appearing obstacles, which commonly exist in real-world scenarios. Note that obstacles that are not modeled in the optimization problem may appear on the pre-optimized trajectory. Thus this problem cannot be solved by traditional trajectory tracking methods such as the combined feed-forward and feedback control method or pure pursuit control method. In this research, we design a consensus-based deep reinforcement learning framework to allow for multi-agent training to improve the training speed and achieve better training performance than single-agent training. We also propose a deep reinforcement learning algorithm based on an end-to-end mapless collision avoidance algorithm for training in the proposed distributed learning framework.

A. State Space and Action Space

The state space at time step t is represented by vector s_t and it includes range sensor data vector r_t, velocity data vector v_{t-1} at the previous time step, and relative target position vector p_t. The range sensor data is normalized to the range of $[0, 1]$. The action space at time step t is represented by a_t that is made of angular speed η_t and linear velocity v_t.

B. Reward Design

Reward functions are designed to encourage vehicles to achieve the goal and avoiding potential collision risks. The reward function is defined as

$$r_{\text{total}} = r_{\text{d}} + r_{\text{c}} + r_{\text{vc}} \tag{25}$$

where r_{total} represents the total reward, r_{d} describes distance reward for reaching the goal, r_{c} is the clearance reward for safety, r_{vc} represents the velocity control reward. r_{d} can be calculated as

$$r_{\text{d}} = \begin{cases}r_{\text{a}} & \text{if } d_{\text{rg}} < d^{\text{rgmin}}\\ k\Delta d_p & \text{if } d_{\text{rg}} \ge d^{\text{rgmin}}\end{cases} \tag{26}$$

where vehicle receives arrival reward r_{a} when the distance between vehicle and goal d_{rg} is less than the threshold d^{rgmin}. Otherwise, r_{d} is proportional to Δd_p, which is the distance the vehicle moves towards the goal at the last time step, where k is a user-configurable parameter. The clearance reward r_{c} can be calculated using

$$r_{\text{c}} = \begin{cases}-r_{\text{cp}} & \text{if } d_{\text{ro}} < \underline{r}\\ 0 & \text{if } d_{\text{ro}} \ge \underline{r}\end{cases} \tag{27}$$

When the distance between vehicle and obstacle d_{ro} is less than the threshold $\underline{r}$, the vehicle will be given negative reward/punishment $-r_{\text{cp}}$. The velocity control reward r_{vc}, angular velocity reward r_η and linear velocity r_{v} reward are given by

$$r_{\text{vc}} = r_\eta + r_{\text{v}}$$
$$r_\eta = \begin{cases}-r_{\text{ap}} & \text{if } |\eta| > \eta^{\max}\\ 0 & \text{if } |\eta| \le \eta^{\max}\end{cases}, r_{\text{v}} = \begin{cases}-r_{\text{lp}} & \text{if } v < v^{\min}\\ 0 & \text{if } v \ge v^{\min}\end{cases} \tag{28}$$

where the subscript i is omitted here as the vehicles have the same reward design. The angular velocity threshold and the linear velocity threshold are denoted by $\eta^{\max}$ and $v^{\min}$, respectively. As we want the vehicle to move to the goal waypoint at high linear speed and low angular speed, if the angular velocity η is larger than $\eta^{\max}$ or the linear velocity v is less than $v^{\min}$, then vehicle will receive punishment value $-r_{\text{ap}}$ or $-r_{\text{lp}}$, respectively.

C. Network Structure

The architectures of actor and critic networks are shown in Fig. 3. The input layer of actor network integrates laser data (24-dimensional vector), relative target position (2-dimensional vector), and current speed data (2-dimensional vector). The following layers include three fully connected layers. Each has 512 nodes and a rectified linear unit function (ReLU). The output includes linear velocity and angular velocity, generated by a sigmoid function and a hyperbolic tangent function, respectively. The critic network merges the input and output layers of the actor network as its own input. The critic network includes three fully connected layers and the ReLU function. The output of the critic network is the Q-value.

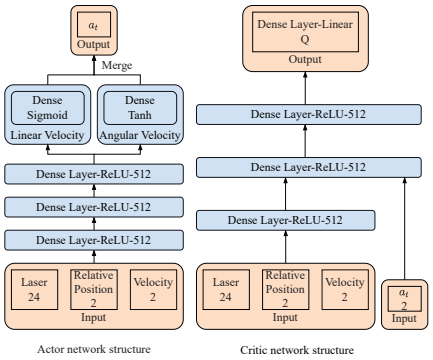


Fig. 3: Network structure

D. Integration of Actor-critic Off Policy Reinforcement Learning and Consensus Algorithm

The proposed training algorithm is now introduced in which the actor and critic training parameters are shared among the agents and updated simultaneously. For agent i, denoting the training parameter in actor and critic networks as $\xi_i \in \mathbb{R}^{n_\xi}$ and $\zeta_i \in \mathbb{R}^{n_\zeta}$, respectively. Let $\chi \in \{\xi, \zeta\}$ denote either training parameter. For an undirected graph $\mathcal{G}$, if $\hat{\chi}_i$ represents the updated training parameter of χ_i after a single consensus step and χ_i stands for the row vector of the training parameter for agent i in the graph, the distributed consensus algorithm updates the actor and critic parameters of an agent i using the following scheme:

$$\hat{\chi}_i = \chi_i + \sum_{k=1}^{N_a} w_{ik}(\chi_k - \chi_i) = \chi_i - \sum_{k=1}^{N_a} l_{ik}\chi_k = \mathcal{C}_i(\chi_k, l_{ik}) \quad (29)$$

where w_{ik} is the element of the graph adjacency matrix which is engendered by the undirected graph $\mathcal{G}$, l_{ik} is the element of the Laplacian matrix $\mathcal{L}$ and $\mathcal{C}_i$ represents the consensus protocol of the i^{th} agent. The training parameter update of all the N_a agents with a single consensus step can be described as

$$\hat{\chi} = ((I_{N_a} - \mathcal{L}) \otimes I_{n_\chi})\chi = \mathcal{C}(\chi, \mathcal{L}) \quad (30)$$

where $\mathcal{C}$ stands for the distributed consensus protocol for all agents, I_{N_a} is a $N_a \times N_a$ identity matrix, and I_{n_χ} denotes a $n_\chi \times n_\chi$ identity matrix and $\mathcal{L}$ represents the Laplacian matrix of undirected graph. For undirected communication topology, by applying (30) in the parameter training process, this distributed consensus algorithm is verified to make all agents' parameters converge to an average value if the following assumptions are satisfied [27]: 1) All agents are trained for the same task in the same scenario; 2) Each agent is able to complete the training independently; 3) The state space and action space are bounded. The design of policy function, the value function, and the process of updating the actor and the critic training parameter are extensions of [28] and are omitted here to save space.

Remark 2. The consensus algorithm mentioned above is theoretically applicable to balanced graph topology. Since any undirected graph is balanced, the communication topology is chosen to be undirected and the main focus is on the performance of the designed DRL-based collision-free tracking control. Balanced digraphs may further save the network bandwidth. Future works should investigate the training performance of the designed consensus-based DRL framework for arbitrary balanced communication topology.

Remark 3. The proposed consensus algorithm with deep reinforcement learning allows multiple agents to share actor-critic policy model parameters, instead of training data, in a distributed manner. In centralized training, the central agent exchanges information with all other agents, which limits the number of agents for limited network bandwidth. Compared with centralized training, the designed method not only protects the data privacy of each agent but also saves the network bandwidth for communication.

V. Simulation and HIL Studies

A. Lane Change Scenarios and Parameter Specification

In this section, Matlab and CARLA [29] simulation results and HiL test results in different lane change scenarios are provided to demonstrate the performance of the proposed hierarchical control framework. The simulation and HiL test setup are presented in Fig. 4.

The physical constraints of vehicles, together with task and algorithm-related parameters are assigned in Table II. Three scenarios are designed and are summarized in Table III, where Vehicle i is abbreviated as Vi. Scenario 1 and Scenario 2 are used in simulation and Scenario 3 is used in the HiL test. Boundary conditions of vehicles' state variables are $[\theta_i(t_0), \delta_i(t_0), v_i(t_0), \eta_i(t_0), a_i(t_0)] = [0, 0, 10, 0, 0]$, $[\theta_i(t_f), \delta_i(t_f), v_i(t_f), \eta_i(t_f), a_i(t_f)] = [0, 0, 10, 0, 0]$ $\forall i \in I_a$.

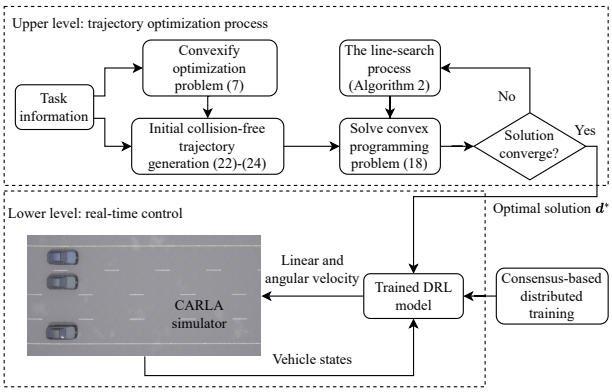


Fig. 4: Simulation setup

TABLE II: Parameters and physical constraints

Parameter/Variable	Value/Range	Parameter	Value
p_{ix}, m	[0, 80]	L_r, L_w, L, m	0.7, 2.5, 4
p_{iy}, m	[0, 14]	$\underline{r}$, m	0.2
θ_i, rad	$[-\pi/2, \pi/2]$	Y_1, Y_2, Y_3, Y_4	1.75, 5.25, 8.75, 12.25
δ_i, rad	[-0.576, 0.576]	N_c	2
η_i, rad/s	[-1.5, 1.5]	N_t	40
v_i, m/s	[-15,15]	ϵ	10^{-3}
a_i, m/s^2	[-2.5, 2.5]	w_z	20
$[y_l, y_u]$, m	[0,14]	$\beta_1, \beta_2, c_1, c_2$	10, 10, 0.01, 0.9

TABLE III: Scenario settings

Scenario	Initial lanes	Desired lanes	Additional setting
1	V1 in lane 4 V2 in lane 3 V3 in lane 1	V1 to lane 3 V2 to lane 1 V3 to lane 2	Road width changes
2	V1 in lane 1 V2, V3 in lane 2 V4, V5 in lane 3 V6 in lane 4	V1 to lane 4 V2, V3 to lane 3 V4, V5 to lane 2 V6 to lane 1	\
3	V1 in lane 4 V2 in lane 3 V3 in lane 1	V1 to lane 3 V2 to lane 1 V3 to lane 2	Unmodeled obstacle appears

B. Multi-Vehicle Lane Change Simulation Results

In this subsection, the performance of the proposed convex feasible set-based SCP algorithm for the multi-vehicle lane change problem is evaluated and simulation results are presented. The vehicles' positions

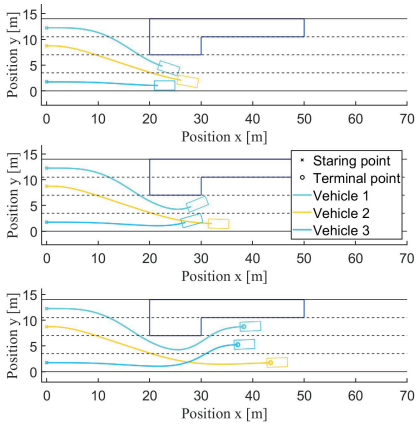


Fig. 5: Multi-vehicle lane change trajectory in Scenario 1

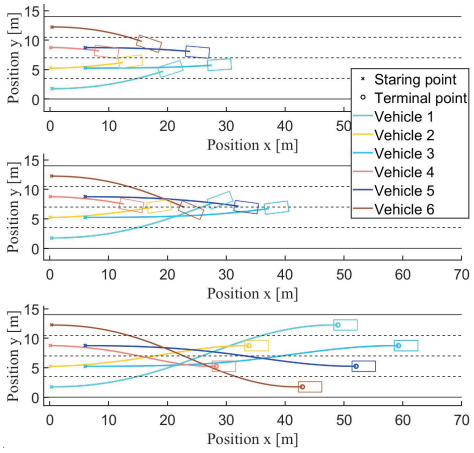


Fig. 6: Multi-vehicle lane change trajectory in Scenario 2

and lane change trajectories at different time points in the two scenarios are depicted in Fig. 5 and Fig. 6. From the presented lane change profiles, it is clear that the proposed algorithm is able to generate trajectories that satisfy the variable constraints and fulfill the lane change mission for the two scenarios. In addition to completing the lane change task, collision-free is also a necessary condition for the multi-vehicle lane change maneuver and should be checked. During the lane change process, the minimum distance among the vehicles in the two scenarios are 0.616 m and 0.244 m, respectively, which are relatively small. Since the algorithm aims to minimize the traveling time, it will choose an aggressive trajectory as long as this trajectory satisfies the constraints and shortens the traveling time. Since we set $\underline{r} = 0.2$ and the minimum distances in both scenarios are smaller than this value, the simulation result is reasonable. If additional safety margins are desired, $\underline{r}$ can be increased. It is worth noting that due to the existence of nonlinear system equations and multiple nonconvex collision avoidance constraints in the original problem, there is currently no approach that can guarantee finding the global optimal solution. The proposed strategy approximates the original nonconvex optimization problem by a convex one and only locally optimal solutions can be obtained. There may be some other local optima with objective values different from those obtained here.

The performance of the proposed CFS-based SCP algorithm in Scenario 2 is further tested in CARLA simulator. The road width and vehicle specifications in this test are identical to the aforementioned simulations. The lane change result in CARLA simulator is shown in Fig. 7, where the plotted coordinate is shifted from the coordinate in CARLA to be consistent with previous simulations. Although the trajectories generated by the proposed algorithm in CARLA are slightly different from the simulation in Matlab, they completed the lane change task in CARLA without collision, which further demonstrated the effectiveness of the proposed method.

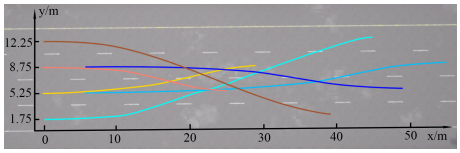


Fig. 7: Multi-vehicle lane change profile in CARLA

C. Comparative Studies

Comparative studies are conducted in this subsection to verify the effectiveness of the initial trajectory generation strategy introduced in (22)-(24) and the designed trust region constraints and the line-search strategy in the CFS-based SCP algorithm. All the methods mentioned in this subsection are evaluated 200 times in both Scenario 1 and 2 with different initial positions given by

$$p'_{ix}(t_0) = p_{ix}(t_0) + d_{ix},\ p'_{iy}(t_0) = p_{iy}(t_0) + d_{iy},\ i \in I_a,$$

where $d_{ix}, d_{iy}, i \in I_a$ are perturbations uniformly distributed on $[-0.7, 0.7]$. Four indicators are selected, namely the instances that can be solved successfully out of 200 trials, the objective t_f, the approximation error ℓ given by (21), and the computation time t_c. The last three indicators are reported by median value / worst case value on success in Table IV.

First, the effectiveness of the designed line search strategy is studied and SCP without line search is evaluated for comparison. Without line search, Algorithm 2 is no longer used in Step 6 of Algorithm 1. When Algorithm 1 is not converging, $\boldsymbol{d}^m$ is used as the reference for the next iteration. The results of these methods are presented in Table IV. It can be observed that although SCP with or without line search achieved a similar objective, the computation time of SCP with line search is smaller in general. The approximation error of SCP with line search tends to be smaller since the search direction reduces the merit function ℓ. Note that the computation demand of SCP with line search comprises solving convex subproblems and executing the line search. It is possible that compared to the time reduced in convex programming iterations, the line search itself requires a longer computation time. This may be why the median computation time of SCP with line search in Scenario 1 is longer. We infer that the benefit of line search is more significant in more complex problems, as the computation time is considerably shorter in Scenario 2. Note that SCP-based methods have approximate errors since each convex sub-problem is not the same as the original nonconvex problem. There is a trade-off between dynamic feasibility and computation time. One may allow for more iterations to get results closer to the solution of the original problem [26]. In the considered task, the SCP solution is a reference for the DRL control module, and spontaneous obstacles appear during the lane change task. Therefore, we believe that the reference trajectories generated by the proposed CFS-based SCP algorithm are sufficient for the task and the results in Table IV are acceptable.

The proposed initialization method is compared with different initialization methods and the results are also shown in Table IV. The first sets $u_i^0(t) = 0$ and propagates the system equation (1) from the initial state $x_i(0)$ [30] (referred to as Propagate in Table IV). The second is the classic A* algorithm [6]. Initialized with these two methods, SCP cannot solve all the 200 trials. Even if these methods can obtain a solution, the optimality and algorithm

TABLE IV: Monte Carlo Comparison of Trajectory Optimization Methods

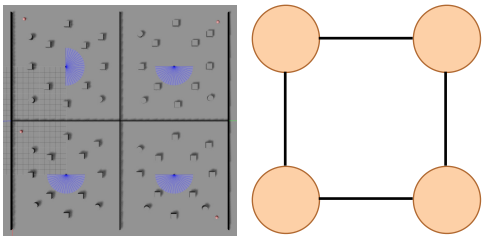
Different methods	Scenario 1			
	Solved	t_f [s]	ℓ	t_c [s]
E-A*+SCP	200	4.46 / 4.59	0.010 / 0.027	5.88 / 6.00
E-A*+SCP w/o LS	200	4.54 / 4.95	0.016 / 0.034	5.83 / 7.32
Propagate+SCP	194	4.71 / 5.18	0.032 / 0.049	6.34 / 7.59
A*+SCP	197	4.67 / 4.98	0.019 / 0.033	5.93 / 6.36
Method 1 [31]	171	4.66 / 4.96	N/A	39.50 / 78.26
Method 2 [22]	183	4.60 / 5.13	N/A	56.28 / 102.87
Different methods	Scenario 2			
	Solved	t_f [s]	ℓ	t_c [s]
E-A*+SCP	200	5.74 / 5.81	0.043 / 0.081	11.06 / 11.77
E-A*+SCP w/o LS	200	5.72 / 5.94	0.052 / 0.085	12.83 / 14.65
Propagate+SCP	186	6.14 / 6.78	0.142 / 0.190	15.92 / 16.62
A*+SCP	190	5.83 / 5.98	0.072 / 0.084	13.11 / 15.24
Method 1 [31]	122	6.02 / 7.74	N/A	154.84 / 339.16
Method 2 [22]	161	5.78 / 7.98	N/A	100.71 / 145.04

runtime are generally worse than the proposed method. It can be seen from Table IV that in Scenario 1 when SCP initialized with classic A* algorithm converges, the performance is similar to that of SCP initialized with E-A*. This is probably because Scenario 1 is relatively simple and using the trajectory generated by the classic A* algorithm is sufficient to find the optimal solution. In Scenario 2, SCP with the proposed initialization method achieved better performance in all three indices, showing the advantage of starting from an initial collision-free trajectory.

Furthermore, two motion planning methods are compared with the proposed hierarchical method. Method 1 [31] suggested a sequential quadratic programming-based optimization algorithm that solves the nonconvex problem (7) directly. Method 2 [22] also generates an initial feasible trajectory first, and then solves the optimization problem by an adaptive gradient particle swarm optimization (PSO) algorithm. The lane change results obtained by these algorithms are also presented in Table IV. Both methods solve the nonconvex optimization problem directly and therefore have no approximation error ℓ. The success rates of these methods are much lower than SCP-based methods while requiring longer computation time. These phenomena are more noticeable in Scenario 2, indicating that these nonconvex methods may not be suitable for complex multi-vehicle trajectory optimization problems. It is also shown that SCP-based methods have better optimality when solving the considered multi-vehicle lane change trajectory optimization problem. Specifically, while the median objective values of these methods are in general similar to that of SCP-based methods, the worst-case objective values are much higher than that of the proposed method in Scenario 1, and in Scenario 2, they are much higher than those of all the SCP-based methods.

D. Comparison of Proposed Training Method and Standard DDPG Algorithm

In this section, the proposed consensus-based distributed deep reinforcement learning method and standard DDPG algorithm are compared in terms of reward value. We simulated the learning environment using a Gazebo simulator and the Robot Operating System (ROS). The rewards are assigned as $r_a = 100$, $r_{\text{cp}} = 80$, and $r_{\text{ap}} = r_{\text{lp}} = 1$ based on empirical experience. As shown in Fig. 8a, four vehicles were simulated and each vehicle learned collision avoidance policy in the same environment. The red spheres represent targets. The grey cubes and cylinders represent obstacles. The blue lines are lidar rays and the center of the lidar ray is a vehicle. The vehicle navigated itself to the target. Once it collides with the obstacle or arrives at the target, it is reset to the original position which is the center of its own cell. If the vehicle arrives at the target, the target resets its own position (that is in one of the four corners of its own cell). We made four vehicles learn collision avoidance policy simultaneously and used the proposed consensus algorithm to allow them to share the learned parameters. We compared the proposed consensus algorithm with a single vehicle learning using DDPG. As shown in Fig. 9, the proposed method achieves faster reward growth and higher reward value than standard DDPG. It is noted that the proposed consensus-based algorithm only used 32629 steps to achieve reward value 6. However, the single training method required more than 100000 steps. Besides, the proposed method achieves a higher reward value than the single training method.



(a) Gazebo simulator environment. (b) Interaction topology of multiple vehicles.

Fig. 8: Simulation environment and interaction topology

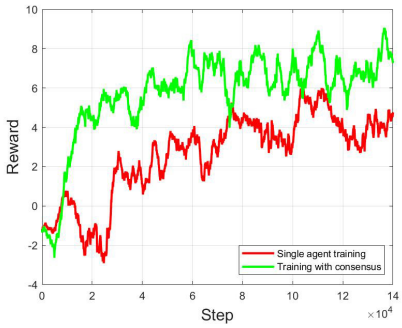


Fig. 9: Comparison of single-agent training and consensus based multi-agent training

E. Hardware-in-the-loop Tests

HiL tests were conducted to demonstrate the validity of the proposed hierarchical collision-free control framework. The executable scripts are created by an IPC-610MB-30LDE/I5-2400/DDR3 industrial PC via LabVIEW real-time mode. A NI PXI-8820 embedded controller is used to test the performance of the proposed framework. The test platform is shown in Fig. 10. The processing time of the trained DRL model in this platform is 0.027s to 0.034s and the time-step in CARLA is set to 50ms. To validate the robustness of the proposed training method, the policy trained in the simulated environment was transferred to HiL tests without fine-tuning.

TABLE V: Quantitative Results of the HiL Test

Methods	Proposed	Reward set 1	Reward set 2	DDPG	DWA [16]
t_f	4.347s	4.628s	4.591s	4.676s	5.056s
I_1	8.583	8.934	8.956	8.800	9.902
I_2	0.449	0.481	0.341	0.562	0.392

In HiL tests, the goal of the vehicles was still to change the line in the shortest time, and the trajectories generated by the algorithm presented in Section III served as a reference. Unlike the simulation tests, an unexpected obstacle appears in front of a vehicle during

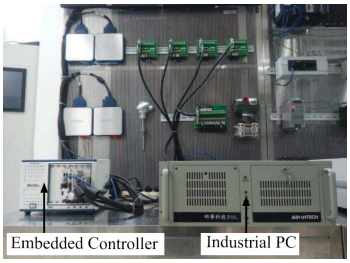


Fig. 10: Front view of real-time simulator

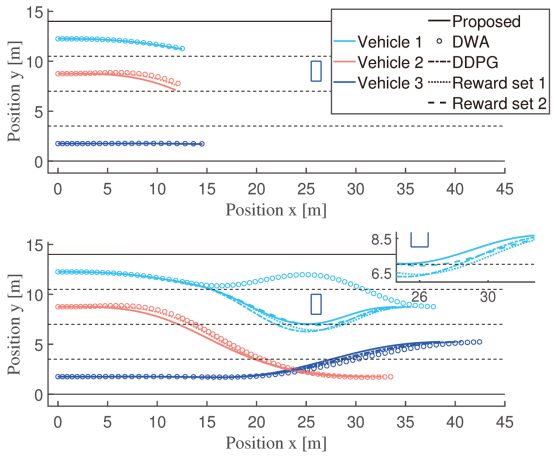


Fig. 11: Trajectories in the HiL test

the lane change and the proposed DRL-based scheme is used to avoid collision. Note that there is no communication among vehicles. For each vehicle, other vehicles are considered as obstacles, and their positions can only be obtained by laser scanner. Two other methods are used for comparison, namely the DWA [16] and the standard DDPG algorithm. Moreover, the proposed DRL methods with different reward values are also compared to analyze the effect of reward values on collision avoidance performance. The reward combination $r_a = 100$, $r_{\text{cp}} = 200$, and $r_{\text{ap}} = r_{\text{lp}} = 1$, which imposes higher collision punishment, is denoted as Reward set 1. The combination $r_a = 100$, $r_{\text{cp}} = 80$, and $r_{\text{ap}} = r_{\text{lp}} = 2$, which imposes higher punishment on low linear velocity and high angular velocity, is denoted as Reward set 2.

At the beginning of the HiL test, three vehicles changed lanes with no obstacles in the road. As shown in the upper figure in Fig. 11, during the lane change, an obstacle (blue box) appears. The affected vehicle started an avoidance process immediately, based on different collision avoidance policies. All methods steered the vehicle to bypass the obstacle and the vehicle continued its lane change task, as shown in the lower figure in Fig. 11. Quantitative results are provided in Table V. Since smoother velocity and heading angle trajectories not only contribute to the stability of vehicle operation but also reduce abrupt changes in control variables, $\mathrm{I}_1 = \frac{1}{N_t}\sum_{i\in I_a}\sum_{j=1}^{N_t} a_i(j)^2$ and $\mathrm{I}_2 = \frac{1}{N_t}\sum_{i\in I_a}\sum_{j=1}^{N_t} \eta_i(j)^2$ are selected as performance indicators of different local motion planning techniques. The DWA-based method chose a significantly different path which requires more time to complete the task. The result of the DDPG-based method is slightly worse than that of the proposed method, showing the advantages of the proposed consensus-based training method in achieving higher reward value. Note that the proposed training method also converges faster. The proposed method trained with Reward set 1 gets higher punishment on collision and the resulting path is more conservative, as the minimum distance between the vehicle and the obstacle is larger. As for the model trained with Reward set 2, since it seeks a lower angular velocity and a higher linear velocity, the resulting path has a smaller orientation change when the vehicle starts the avoidance process. However, the safety margin $\underline{r}$ must be kept and the vehicle takes a longer time to bypass the obstacle.

VI. Conclusion

In this paper, the lane change motion planning and control problem for multi-vehicles in an uncertain environment has been addressed using a hierarchical control scheme. A trust region sequential convex programming algorithm, equipped with a convex feasible set-based collision avoidance strategy, served as the motion planner to generate the optimized collision-free maneuver trajectory for a group of vehicles. A line search process was designed to reduce the error caused by the convexification process. To remove the collision risk between vehicles and unexpected obstacles during lane changing, we explore the possibility of applying a DRL-based collision-free control method. A distributed consensus method is designed to accelerate multi-agent training. Numerical simulation and HiL tests were performed to confirm the enhanced performance of the proposed design. From the results, it can be concluded that the applied successive convex set approximation strategy and trust region mechanism can noticeably contribute to improving the convergence of the optimization process. In addition, the DRL-based controller was able to drive vehicles to fulfill different lane change missions without colliding with unexpected obstacles. As a result, we believe the presented hierarchical framework is of interest to the research community which is focusing on the development of optimization-based multi-vehicle lane change motion planners and deep learning-based collision-free motion controllers.

References

[1] Y. Ji, Y. Tanaka, Y. Tamura, M. Kimura, A. Umemura, Y. Kaneshima, H. Murakami, A. Yamashita, and H. Asama, "Adaptive motion planning based on vehicle characteristics and regulations for off-road UGVs," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 1, pp. 599–611, Jan. 2019.

[2] H. Guo, C. Shen, H. Zhang, H. Chen, and R. Jia, "Simultaneous trajectory planning and tracking using an MPC method for cyber-physical systems: a case study of obstacle avoidance for an intelligent vehicle," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 9, pp. 4273–4283, Sep. 2018.

[3] J. Nilsson, J. Silvlin, M. Brannstrom, E. Coelingh, and J. Fredriksson, "If, when, and how to perform lane change maneuvers on highways," *IEEE Intelligent Transportation Systems Magazine*, vol. 8, no. 4, pp. 68–78, Oct. 2016.

[4] R. Shome, K. Solovey, A. Dobson, D. Halperin, and K. E. Bekris, "dRRT*: Scalable and informed asymptotically-optimal multi-robot motion planning," *Autonomous Robots*, vol. 44, p. 443–467, 2020.

[5] S. D. Bopardikar, B. Englot, and A. Speranzon, "Multiobjective path planning: Localization constraints and collision probability," *IEEE Transactions on Robotics*, vol. 31, no. 3, pp. 562–577, 2015.

[6] C. Sun, Q. Li, and L. Li, "A gridmap-path reshaping algorithm for path planning," *IEEE Access*, vol. 7, pp. 183 150–183 161, 2019.

[7] A. Wächter and L. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical Programming*, vol. 106, no. 1, p. 25–57, Mar. 2006.

[8] B. Li, H. Liu, D. Xiao, G. Yu, and Y. Zhang, "Centralized and optimal motion planning for large-scale AGV systems: A generic approach," *Advances in Engineering Software*, vol. 106, pp. 33–46, 2017.

[9] B. Gutjahr, L. Gröll, and M. Werling, "Lateral vehicle trajectory optimization using constrained linear time-varying MPC," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 6, pp. 1586–1595, 2017.

[10] N. Goulet and B. Ayalew, "Distributed maneuver planning with connected and automated vehicles for boosting traffic efficiency," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 8, pp. 10 887–10 901, 2022.

[11] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[12] D. Malyuta, T. P. Reynolds, M. Szmuk, T. Lew, R. Bonalli, M. Pavone, and B. Açıkmeşe, "Convex optimization for trajectory generation: A tutorial on generating dynamically feasible trajectories reliably and efficiently," *IEEE Control Systems Magazine*, vol. 42, no. 5, pp. 40–113, 2022.

[13] Z. Wang, G. Li, H. Jiang, Q. Chen, and H. Zhang, "Collision-free navigation of autonomous vehicles using convex quadratic programming-based model predictive control," *IEEE/ASME Transactions on Mechatronics*, vol. 23, no. 3, pp. 1103–1113, 2018.

[14] Y. Li and J. Ibanez-Guzman, "Lidar for autonomous driving: The principles, challenges, and trends for automotive Lidar and perception systems," *IEEE Signal Processing Magazine*, vol. 37, no. 4, pp. 50–61, 2020.

[15] J. Yang, C. Wang, B. Jiang, H. Song, and Q. Meng, "Visual perception enabled industry intelligence: State of the art, challenges and prospects," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 3, pp. 2204–2219, 2021.

[16] E. J. Molinos, Ángel Llamazares, and M. Ocaña, "Dynamic window based approaches for avoiding obstacles in moving," *Robotics and Autonomous Systems*, vol. 118, pp. 112–130, 2019.

[17] S. Xie, J. Hu, P. Bhowmick, Z. Ding, and F. Arvin, "Distributed motion planning for safe autonomous vehicle overtaking via artificial potential field," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 11, pp. 21 531–21 547, 2022.

[18] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. A. Sallab, S. Yogamani, and P. Pérez, "Deep reinforcement learning for autonomous driving: A survey," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–18, 2021.

[19] Y. Zhang and M. M. Zavlanos, "Distributed off-policy actor-critic reinforcement learning with policy consensus," in *2019 IEEE 58th Conference on Decision and Control (CDC)*, pp. 4674–4679, 2019.

[20] Y. Song, A. Romero, M. Müller, V. Koltun, and D. Scaramuzza, "Reaching the limit in autonomous racing: Optimal control versus reinforcement learning," *Science Robotics*, vol. 8, no. 82, p. eadg1462, 2023.

[21] C. Liu, C. Lin, and M. Tomizuka, "The convex feasible set algorithm for real time optimization in motion planning," *SIAM Journal on Control and Optimization*, vol. 56, no. 4, pp. 2712–2733, Jun. 2018.

[22] R. Chai, A. Tsourdos, A. Savvaris, S. Chai, and Y. Xia, "Two-stage trajectory optimization for autonomous ground vehicles parking maneuver," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 7, pp. 3899–3909, 2019.

[23] J. Ziegler and C. Stiller, "Fast collision checking for intelligent vehicle motion planning," in *2010 IEEE Intelligent Vehicles Symposium*, pp. 518–522, Jun. 2010.

[24] L. Xie, R.-Z. He, H.-B. Zhang, and G.-J. Tang, "Oscillation phenomenon in trust-region-based sequential convex programming for the nonlinear trajectory planning problem," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 58, no. 4, pp. 3337–3352, 2022.

[25] Y. Mao, M. Szmuk, and B. Açıkmeşe, "Successive convexification of non-convex optimal control problems and its convergence properties," in *2016 IEEE 55th Conference on Decision and Control (CDC)*, pp. 3636–3641, 2016.

[26] R. Bonalli, T. Lew, and M. Pavone, "Analysis of theoretical and numerical properties of sequential convex programming for continuous-time optimal control," *IEEE Transactions on Automatic Control*, vol. 68, no. 8, pp. 4570–4585, 2023.

[27] R. Olfati-Saber and R. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Transactions on Automatic Control*, vol. 49, no. 9, pp. 1520–1533, Sep. 2004.

[28] W. Liu, H. Niu, I. Jang, G. Herrmann, and J. Carrasco, "Distributed neural networks training for robotic manipulation with consensus algorithm," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 35, no. 2, pp. 2732–2746, 2024.

[29] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, pp. 1–16, 2017.

[30] Z. Wang, "Optimal trajectories and normal load analysis of hypersonic glide vehicles via convex optimization," *Aerospace Science and Technology*, vol. 87, pp. 357–368, 2019.

[31] D. Pardo, L. Möller, M. Neunert, A. W. Winkler, and J. Buchli, "Evaluating direct transcription and nonlinear optimization methods for robot motion planning," *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 946–953, 2016.

Tianhao Liu (Student Member, IEEE) received the B.S. degree in automatic control from Nanjing University of Science and Technology, Nanjing, China in 2020 and received the master's degree in control science and engineering at Beijing Institute of Technology, Beijing, China, in 2023, where he has since been working on his Ph.D. degree. His research interests are in the area of optimal control and its applications in unmanned vehicles.

Runqi Chai (Member, IEEE) received the B.S. degree in information and computing science from the North China University of Technology, Beijing, China, in 2015 and the Ph.D. degree in Aerospace Engineering from Cranfield University, Cranfield, U.K, in August 2018. He is currently a research fellow at Cranfield University. His research interests include unmanned vehicle trajectory optimization, guidance and control.

Senchun Chai (Senior Member, IEEE) received the B.S. and master's degrees in control science and engineering from Beijing Institute of Technology, Beijing, China in 2001 and 2004. He received the Ph.D. degree in Networked Control System from University of South Wales, Pontypridd, U.K., in 2007. He is currently a professor of School of Automation with Beijing Institute of Technology. His current research focuses on flight control system, networked control systems, and embedded systems.

Farshad Arvin (Senior Member, IEEE) received the BSc degree in Computer Engineering, the MSc degree in Computer Systems engineering, and the PhD degree in Computer Science, in 2004, 2010, and 2015, respectively. Farshad is an Associate Professor in Robotics in the Department of Computer Science at Durham University in UK. His research interests include swarm robotics and autonomous multi-agent systems.

Jinning Zhang received the MSc and PhD degree in Aerospace Propulsion with Cranfield University, U.K in 2019 and 2022. She is currently a Lecturer (Assistant Professor) in Aerospace and Computational Engineering with University of Leicester, UK. Her research interests include energy management strategies and integrated control system design for aerospace propulsion, AI for transport and energy systems, and transportation-energy nexus.

Barry Lennox (Senior Member, IEEE) is a Professor of Applied Control and Nuclear Engineering Decommissioning and holds a Royal Academy of Engineering Chair in Emerging Technologies. He is Director of the Robotics and Artificial Intelligence for Nuclear (RAIN) Robotics Hub and Research Director of the Dalton Cumbrian Facility. He is a Fellow of the Royal Academy of Engineering, Senior Member of the IEEE, Fellow of the IET and InstMC, and a Chartered Engineer.