2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 23-27, 2022, Kyoto, Japan

Cola-HRL: Continuous-Lattice Hierarchical Reinforcement Learning for Autonomous Driving

Lingping Gao[1†], Ziqing Gu[2†], Cong Qiu[1], Lanxin Lei[1], Shengbo Eben Li[2], Sifa Zheng[2], Wei Jing[1*], Junbo Chen[1*].

***Abstract*— Reinforcement learning (RL) has shown promising performance in autonomous driving applications in recent years. The early end-to-end RL method is usually unexplainable and fails to generate stable actions, while the hierarchical RL (HRL) method can tackle the above issues by dividing complex problems into multiple sub-tasks. Prior HRL works either select discrete driving behaviors with continuous control commands, or generate expected goals for the low-level controller. However, they typically have strong scenario dependence or fail to generate goals with good quality. To address the above challenges, we propose a Continuous-Lattice Hierarchical RL (Cola-HRL) method for autonomous driving tasks to make high-quality decisions in various scenarios. We utilize the continuous-lattice module to generate reasonable goals, ensuring temporal and spatial reachability. Then, we train and evaluate our method under different traffic scenarios based on real-world High Definition maps. Experimental results show our method can handle multiple scenarios. In addition, our method also demonstrates better performance and driving behaviors compared to existing RL methods.**

I. Introduction

The development of autonomous driving technology is of great significance to human society. Although the rule-based modular system has achieved adequate performance in specific driving scenarios like DARPA [1], it is still hard to attain high-level intelligence and automation in complex scenarios. In contrast, learning-based methods, such as supervised learning or unsupervised learning, extract knowledge from data with less effort to improve performance continuously. Following the success of AlphaGO [2], reinforcement learning (RL) has shown great potential in the autonomous driving field through trial-and-error interaction with the environment [3], [4].

Several prior end-to-end RL (E2E-RL) works attempt to learn a complete control policy directly from perception scratches, such as raw pixels [3], [5], [6]. However, they lack clear interpretability and make it harder to utilize extra modules to ensure security or add new functions. For tasks with multiple sub-goals, traditional E2E-RL methods [7] are usually less stable in generating actions.

The hierarchical RL (HRL) methods usually decompose complex tasks into several sub-problems that are easy to solve, making them better interpretable and be reusable among different tasks. There are roughly two categories: task-oriented and goal-reach HRLs. The task-oriented HRL (Tao-HRL) approach is a temporal abstraction on tasks. The high-level module discretizes the behavior selection, and the low-level module outputs continuous control commands. They utilized discrete driving behaviors to expand a new state space, coordinating with different scenarios [8]–[16], such as lane-changing or lane-keeping on the highway, stopping or passing at the traffic light, turning right, or going straight at intersections. They often trained different layers separately with the manual decomposition of the value function, reward, and even termination conditions. However, the method and scenario are always strongly dependent. It is hard to design all possible discrete behavior transitions in a complex traffic environment. Furthermore, the frequent switching of behavior selection will affect the performance and efficiency of control commands.

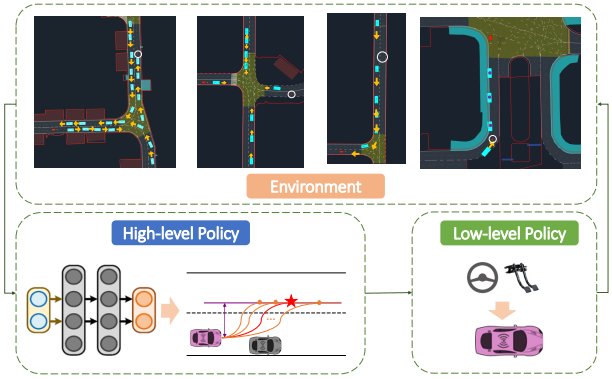


Fig. 1: Goal-reach Hierarchical framework in autonomous driving

Goal-reach HRL (Goal-HRL) method utilizes the high-level layer to generate a goal, while the low-level layer is responsible for making the agent towards these goals. In autonomous driving tasks, these goals are usually regarded as trajectory points, or waypoints [17], [18]. The challenge is that unreachable goals will directly affect the temporal and spatial rationality of control commands. Some methods failed to consider the kinematic reachability of goals, which put the most burden on the lower controller [19]. Other works may utilize smoothing or sampling techniques to fit one or multiple curves that reach or near the goal to keep reachability, regarded as a planning process or trajectory generation process [10]. However, these methods required

Lingping Gao and Ziqing Gu contributed equally to this work. All correspondence should be sent to Junbo Chen and Wei Jing. Email: gaolingping.glp@alibaba-inc.com, gzq_goodlee@163.com, junbo.chenjb@taobao.com, jw334405@alibaba-inc.com.

[1]L. Gao, C. Qiu, L. Lei, W. Jing, J. Chen are with Alibaba Group, Hangzhou, 310000, China.

[2]Z. Gu, S. Li, S. Zheng are with State Key Lab of Automotive Safety and Energy, School of Vehicle and Mobility, Tsinghua University, Beijing, 100084, China.

2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) | 978-1-6654-7927-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/IROS47612.2022.9982041

a lot of post-processing processes, even handcraft-based cost functions, to select one appropriate trajectory [20].

In this paper, we propose an adaptive **Co**ntinuous-**La**ttice **HRL** (Cola-HRL) method for high-quality decision-making and trajectory generation in various autonomous driving scenarios, as shown in Fig. 2. The proposed method generates suitable goals under the Frenet coordinate system, satisfying the temporal and spatial reachability. The main contributions of this paper are summarized as follows:

1) The Cola-HRL method utilizes the continuous-lattice module to assist the high-level RL policy in generating goals, which are temporal-and-spatial reachable.
2) Based on the Frenet coordinate system, we design continuous action space to further constrain high-level goals in the driveable area, which leads to better training performance, driving behaviors, and action stability.
3) The proposed method has good adaptability and is demonstrated to handle autonomous driving tasks in multiple scenarios. It also outperforms several other baseline RL/HRL methods in various tasks.

II. Related Works

This section mainly summarizes previous works related to this paper, including learning-based methods and two application branches of hierarchical reinforcement learning methods.

A. Reinforcement Learning methods in autonomous driving

Learning-based methods have been increasingly applied to autonomous driving tasks. The first end-to-end decision-making system was established with a fully connected network in the late 1980s. With the success of AlphaGO [2], some works attempted to implement RL to eliminate the demand for labeled driving data [5], [21]–[23]. Wayve first successfully applied the DDPG algorithm to a real vehicle equipped with sensors [6]. Besides, other RL algorithms, such as A3C [24], SAC [7], inverse RL [23] were also utilized to learn control policy. They usually adopted an end-to-end framework and directly output control commands from perception scratches. However, it is hard to interpret the results and generalize them to other scenarios. Comparatively, the hierarchical RL method that separates the high-level decision-making from the low-level control has better interpretability than the end-to-end method.

B. Task-oriented Hierarchical Reinforcement Learning (Tao-HRL)

The Tao-HRL method originated from the MAXQ [25], which utilized the temporal abstraction [8], [13], [26]–[28] to guide the final policy, such as the option-critic framework. They usually decomposed the autonomous driving task into multiple sub-tasks, where each sub-task corresponds to an individual Markov Decision Process for continuous motion control. The sub-tasks were designed according to scenarios, such as "lane changing" or "keep following" on highways, "turn left" or "turn right" at intersections [14], [20], or even low-level controller's id [16]. However, scenario dependence makes it difficult for such methods to be applicable in diversified traffic environments.

C. Goal-reach Hierarchical Reinforcement Learning (Goal-HRL)

The Goal-HRL method focused on outputting goals for the lower control module to track, regarded as trajectories or waypoints [17], [19]. The critical sight lies in that goals of the network output may be unreachable, which could cause the problem that the lower controller is difficult to track. Works in the robotic domain would combine the accurate inverse dynamic model [29], [30] to assist in generating goals indirectly. Some methods utilized Monte Carlo tree search or dynamic window approach to random sample goals and construct candidate trajectories [18], [20]. Other works outputted planner's prediction horizon [15] to decide the expected position of the goal. Although these practices provide an excellent experience for solving high-level decision-making problems, they still cannot guarantee the temporal and spatial reachability of generated goals; some even need to select a best candidate trajectory with handcraft cost functions [15], [20].

III. Problem Statement

Our **Cola-HRL** method is mainly for decision-making and planning tasks of autonomous vehicle. The agent receives the information of surrounding vehicles and map from the environment, and outputs the longitudinal speed and the lateral offset from the reference line at the future time step t. Furthermore, the **Cola-HRL** method consists of two parts: a high-level layer utilizes RL policy to generate reasonable goals, and a low-level layer tracks them with the controller stably. We first formulate the problem generally and then explain the method in modules.

A. General goal-reach HRL Problem Formulation

We describe the sequential decision-making problem as a Markov Decision Process (MDP), which is defined by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$. $s \in \mathcal{S}$ is the continuous state space. $a \in \mathcal{A}$ represents the continuous action space. $\mathcal{P}$ stands for the state transition probability. $\mathcal{R}$ denotes the reward from the environment at each transition. $\gamma \in (0, 1)$ is the discount factor. The RL problem aims to learn a policy to maximize the expected accumulated rewards as follows:

$$\begin{aligned} &\max_{\pi} J(\pi) = \mathbb{E}_{s\sim\rho_\pi}[\Sigma_{i=0}^{\infty}\gamma^i r(s,a)], \\ &with \quad a = \{x^{goal}, y^{goal}\}, \end{aligned} \tag{1}$$

where $r(\cdot,\cdot)$ is the reward function (details are shown in Section IV-A). ρ_π is the state-action distribution under the policy π. a is the output action, including the lateral y^{goal} and longitudinal position x^{goal}. At time step t, the high-level layer guides the generation of goal , (x_t^{goal}, y_t^{goal}), while the low-level layer navigates between adjacent goals, from the current goal, (x_t^{goal}, y_t^{goal}), to the next goal, $(x_{t+1}^{goal}, y_{t+1}^{goal})$.

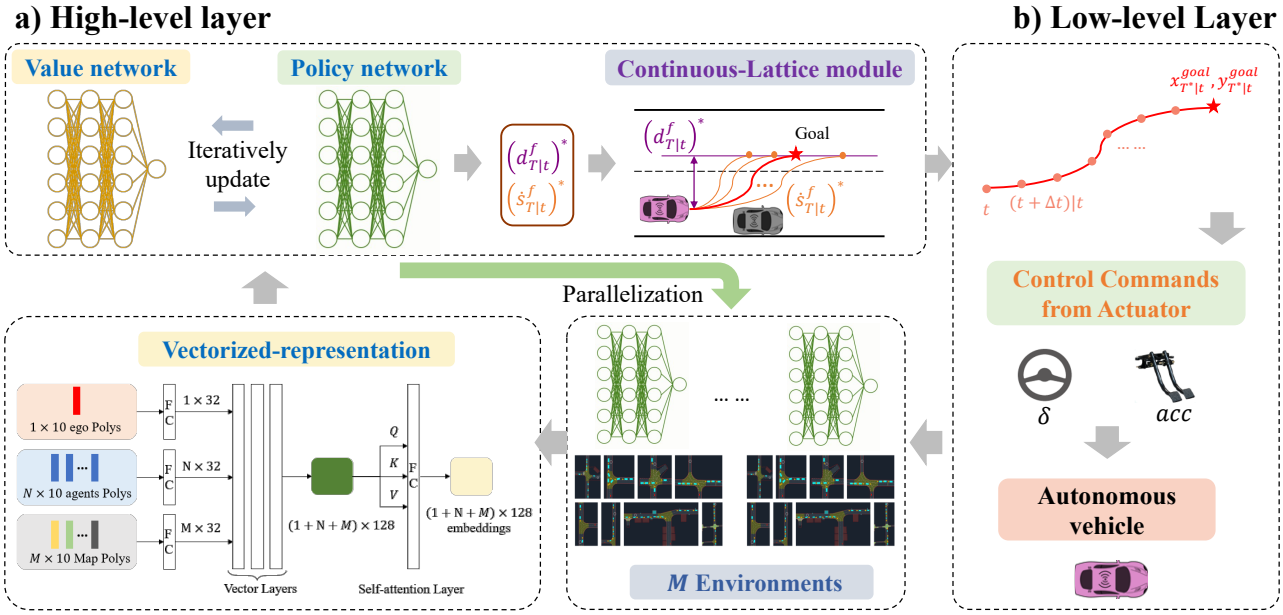


Fig. 2: Continuous-Lattice Hierarchical Reinforcement Learning Framework

B. Continuous-Lattice HRL Problem Reformulation

In order to solve the problem defined in 1, the policy network should directly output the goal at different time steps. However, there lacks constraints between target points at two consecutive time steps, which may make the generated goal difficult to reach and put more burdens on the lower controller.

Therefore, we utilize the continuous-lattice module to avoid the kinematic unreachability problem among goals in general Goal-HRL methods. Besides, we build our problem under the Frenet coordinate system to limit exploration in the drivable area and improve the sample efficiency. Therefore, we reconstruct the problem in (1). Rather than outputting goal, $a := \{x_t^{goal}, y_t^{goal}\}$, directly at each time step t, we utilize $\bar{a} := \{d_{T|t}^f, \dot{s}_{T|t}^f\}$ instead, where $d_{T|t}^f$ and $\dot{s}_{T|t}^f$ are lateral offset and longitudinal velocity of the goal at the future time step T under the Frenet coordinate system, respectively. The key point lies in that the Frenet coordinate system can ensure that the generated goal locates in the driveable area, which could avoid unnecessary and dangerous exploration out of the road. Furthermore, the position of the goal satisfies the following relationship:

$$\begin{aligned} &s^f(t) = \Sigma_{k=0}^{n} \alpha_k t^i,\ d^f(s^f) = \Sigma_{j=0}^{m} \beta_j (s^f)^i, \\ &(s_t^f, \dot{s}_t^f, \ddot{s}_t^f, d_t^f, d_t'^f, d_t''^f) = F_{coor}(x_t, y_t, \theta_t, \kappa_t, v_t, acc_t), \\ &(s_T^f, \dot{s}_T^f, \ddot{s}_T^f, d_T^f, d_T'^f, d_T''^f) = (s_{T|t}^f, \dot{s}_{T|t}^f, \ddot{s}_{T|t}^f, d_{T|t}^f, d_{T|t}'^f, d_{T|t}''^f) \end{aligned} \tag{2}$$

where m and n are degree of polynomials. α_k and β_j are weight parameters. $(x_{(\cdot)}, y_{(\cdot)}, \theta_{(\cdot)}, \kappa_{(\cdot)}, v_{(\cdot)}, acc_{(\cdot)})$ indicates longitudinal and lateral coordinates, heading angle, curvature, velocity, and acceleration under the world coordinate system, respectively. $(s_{(\cdot)}^f, d_{(\cdot)}^f, \dot{s}_{(\cdot)}^f, d_{(\cdot)}'^f, \ddot{s}_{(\cdot)}^f, d_{(\cdot)}''^f)$ represent the longitudinal and lateral coordinates, velocity and acceleration under the Frenet coordinate system. The point represents the derivative of the time, and the prime represents the derivative of the variable. We abbreviate the transformation between the two coordinate systems as F_{coor} [31]. Then, the goal at the future time step T is generated as following:

$$\begin{aligned} &(x_T^{goal}, y_T^{goal}, \theta_T^{goal}, \kappa_T^{goal}, v_T^{goal}, acc_T^{goal}) \\ &= F_{coor}^{-1}(s_T^f, \dot{s}_T^f, \ddot{s}_T^f, d_T^f, d_T'^f, d_T''^f) \end{aligned} \tag{3}$$

IV. Methodology

This section mainly describes specific design of state, action and reward, network structure, training framework and algorithm. Detailed parameters are shown in Table. (I).

A. Definition of State, Action and Reward

1) State: Features of the state space is composed of vectors, $\mathbf{v}$, categorized into three parts. Ego vehicle features, $\mathbf{v}^{ego}$, include $(x, y, \theta, \kappa, v, acc)$ defined in Section III-B, vehicle length and width, (l, w), steering angle change rate, $\Delta\delta$, vehicle ID, id, lateral and longitudinal distance related to goals, $(\Delta x, \Delta y)$. Surrounding vehicle i features, $\mathbf{v}^{sur,i}$, contain position, (x_i, y_i), velocity v_i, vehicle length and width, (l_i, w_i), heading angle θ_i, vehicle ID, id_i, relative distance d_i to the ego vehicle. High Definition (HD) maps features, $\mathbf{v}^{map}$, includes type, length, id, coordinates of start-points and endpoints of lane boundaries, road boundaries, and centerlines within the perception range.

2) Action: The continuous action space includes high-level policy defined in Section III-B, $a^{high} := \{d_{T|t}^f, \dot{s}_{T|t}^f\}$, and low-level control commands, steering angle and acceleration, $a^{low} := \{\delta_t, acc_t\}$. To guarantee the temporal and spatial reachability of the generated goal, we sample the time domain T at equal time interval Δt to attain reasonable $s_{T|t}^f$, and then sample $s_{T|t}^f$ along the outputted $d_{T|t}^f$ at equal distance Δs^f, as shown in Fig. 3. Besides, we assume

$(\ddot{s}_T^f, d_T'^f, d_T''^f)$ in (2) equals to zero. Then, we can calculate weight parameters in (2), and generate n trajectory points between the current position and the sampled goal at the future time step T:

$$\{x_{i|t}^{goal}, y_{i|t}^{goal}, \theta_{i|t}^{goal}, \kappa_{i|t}^{goal}, v_{i|t}^{goal}, acc_{i|t}^{goal}\}, i = \{0, \Delta t, ..., n\Delta t\} \tag{4}$$

where $n\Delta t = T^*$. T^* is the fast time to satisfy the $\dot{s}_{T|t}^f$. Besides, each trajectory point should satisfy kinematic constraints as following:

$$\kappa_{i|t}^{goal} \notin \kappa_{absorb}, v_{i|t}^{goal} \notin v_{absorb}, acc_{i|t}^{goal} \notin acc_{absorb} \tag{5}$$

where $(\cdot)_{absorb}$ represents the state space exceeding the threshold. Through sampling the time domain T and s^f, and the threshold check, we can ensure the temporal and spatial reachability of the goal, which is beneficial for smooth and stable tracking. Besides, we extend the longitudinal length of trajectory to ensure equal time horizon by (2).

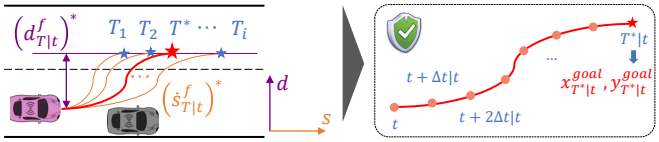


Fig. 3: Continuous-lattice module. The star represents potential goals, where red is the selected one.

3) Reward: We design the reward function to encourage safe, comfortable and stable driving, penalize control bias and collision, which includes the dense reward at each training step and the sparse reward at the termination state:

$$\begin{aligned} r = & k_1(d_t^{ego} - d_{t-1}^{ego}) + k_2(d_{T|t}^f - d_{T|t+1}^f) \\ & + k_3(\dot{s}_{T|t}^f - \dot{s}_{T|t+1}^f) + \mathcal{R}_{step} \end{aligned} \tag{6}$$

where at each time step, the reward includes: 1) relative distance with target driving area between adjacent time step, $d_t^{ego} - d_{t-1}^{ego}$; 2) unsmoothness penalty if acceleration is too large, $\dot{s}_{T|t}^f - \dot{s}_{T|t+1}^f$; 3) unstable penalty if lateral offset change is too large, $d_{T|t}^f - d_{T|t+1}^f$; 4) time step penalty, $\mathcal{R}_{step}$.

At the terminal condition, the reward contains: 1) collision with road or other vehicles penalty $\mathcal{R}_{coll}$; 2) out of time penalty $\mathcal{R}_{out}$; 3) success reward $\mathcal{R}_{suc}$.

B. Neural Network Architecture

We encode all features with the vectorized representation into a fixed-length embedding through a fully connected graph neural network, which draws on the idea of VectorNet [32], as shown in Fig. 2. Each polyline $\mathcal{P}$ is composed of vector nodes, $\{\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_P\}$, which are defined in Section. IV-A.1. Each vector layer is defined as

$$\mathbf{v}_i^{(l+1)} = \varphi_{rel}\left(g_{enc}(\mathbf{v}_i^{(l)}), \varphi_{agg}\left(\left\{g_{enc}(\mathbf{v}_j^{(l)})\right\}\right)\right), \tag{7}$$

where $\mathbf{v}_i^{(l)}$ is the input of the l-th layer. $\mathbf{v}_j$ is other vectors connected with $\mathbf{v}_i$ in $\mathcal{P}$. φ_{rel} is feature concatenation operation. g_{enc} is a multi-layer perception (MLP) and φ_{agg} is the max-pooling operation. For the output embedding $\{\mathbf{v}_1^o, \mathbf{v}_2^o, ..., \mathbf{v}_P^o\}$, self-attention [33] is used to further extract more information that integrates the shape information of entire polylines. Besides, both value network and policy network are MLPs, respectively.

C. Training framework

Based on the Proximal Policy Optimization [34], we extend it to a distributed framework with multiple policy networks as parallel workers for data sharing. Then we construct the surrogate loss on these data, and optimize it with minibatch stochastic gradient descent. The pseudocode is described in Algorithm 1.

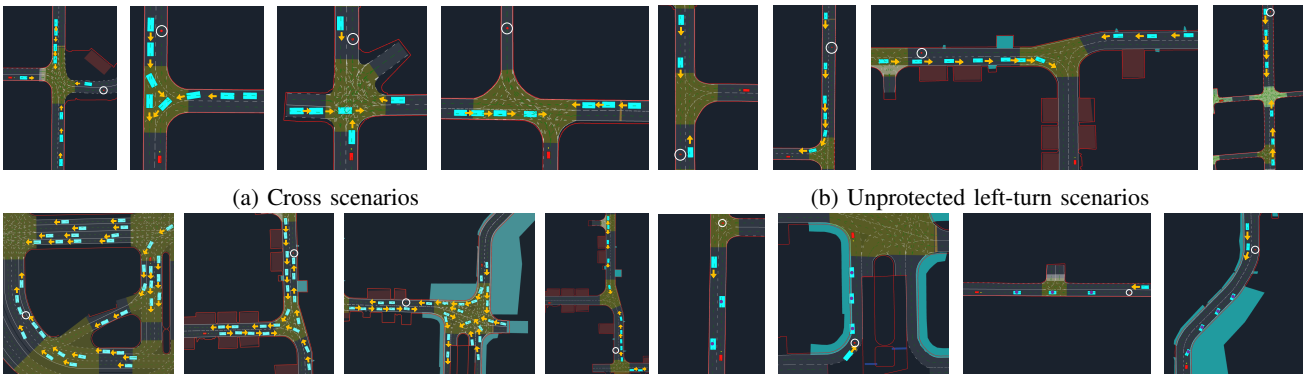
Algorithm 1: Continuous-Lattice HRL Algorithm

```
Initialize value network V_ν, policy network π_ω,
  learning rate β_ν and β_ω, time interval δt,
  iteration = 0.
for 0 ≤ iteration ≤ N_max^iter do
    Initialize step = 0.
    for 0 ≤ step ≤ N_max^step do
        Initialize M environments in parallel and
          env_step = 0, synchronization policy network
          as {π_ω^i}_i^M ← π_ω.
        if env_step ≤ N_max^env and not done then
            // High-level layer
            In environment i, run policy π_ω^i and
              output (d_{T|t}, ṡ_{T|t}),
            Sample T ∈ (T_s, T_e) and s_{T|t}^f ∈ (S_s, S_e)
              with Δt and Δs^f, calculate (4), until (5)
            // Low-level layer
            Calculate the control commands (δ_t, acc_t)
              with the low-level controller.
            Interact with environment and calculate
              the reward to construct (s_t, a_t, r_t, s_{t+1}).
            env_step += 1
            step += 1
        end
        Estimate advantages {Â_i}_{i=0}^{env_step}
        Store trajectory information.
    end
    Sample N transitions from replay buffer D.
    // Policy evaluation
    Update the value network:
    ν ← ν − β_ν ∇_ν J_PPO(ν) [34]
    // Policy improvement
    Update the policy network:
    ω ← ω − β_ω ∇_ω J_PPO(ω) [34]
    iteration += 1
end
```

V. Experiments

A. Simulator

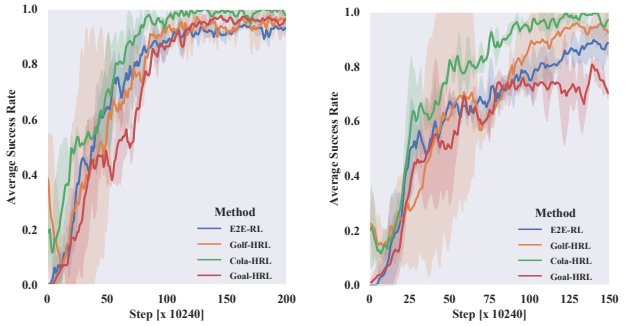
We constructed the simulation environment based on our customized autonomous driving simulator and the OpenAI GYM toolkit [35]. Simulation scenarios for training are built



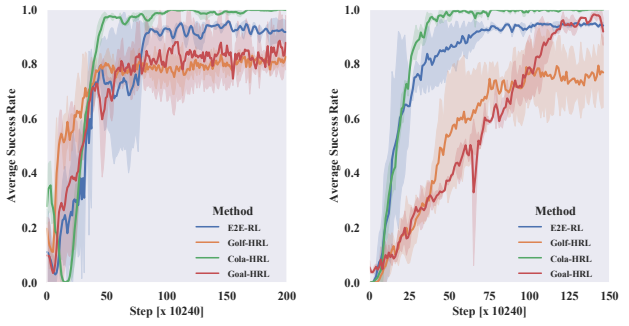
(a) Cross scenarios (b) Unprotected left-turn scenarios

(c) Merge scenarios (d) Meeting scenarios

Fig. 4: Scenario setting. Our map is generated by importing real-world data into the simulator, where the green dotted line, white dotted line, and the solid red line is the centerline, lane boundary, and road boundary, respectively. The yellow-green area represents the junction. The blue area is the bicycle lane. The red box represents the ego vehicle, and the blue box represents the surrounding dynamic or static obstacles randomly generated by the environment. Surrounding vehicles are set to drive along the centerline. The orange arrow represents the driving direction of the surrounding dynamic vehicles, and the purple circle marks the stationary vehicles. The yellow dot in front of the red vehicle represents the current heading direction, and the white circle is the target area of the ego vehicle. After the ego vehicle reaches the target area, the episode will end.



(a) Cross scenario (b) Unprotected left-turn scenario



(c) Merge scenario (d) Meeting scenario

Fig. 5: Performance Comparison

by importing HDMap from real-world traffic scenarios. Each simulation step corresponds to $0.1s$.

B. Experiment Settings

Four scenarios coordinated with real-world maps are considered to verify the proposed method, Cola-HRL, including cross, merge, meeting, and unprotected left-turn scenarios. Each traffic scenario contains multiple cases, as shown in Fig. (4). We adopt a parallel training framework based on RLlib [36] with multiple processors training these cases simultaneously. In the training process, the high-level policy generates 50 goals in the next $5s$ with a time interval of $0.1s$. The current training episode ends if the ego vehicle reaches the preset target area or has a collision. Table. (I) shows additional hyperparameters.

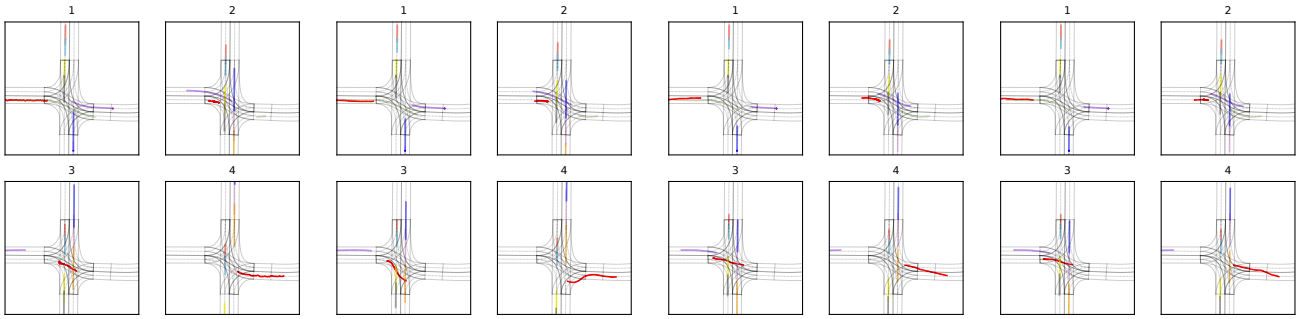
TABLE I: Hyperparameters

Item	Value
Training Batch Size	10240
Mini-batch Size	256
Gradient update iteration	10
cliping ϵ	0.1
Learning Rate Schedule	Anneal linearly to 1e-6
Discount Factor γ	0.99
Target KL	0.05
GAE-λ	0.95
β_ν and β_ω	(3e-4,3e-4)
Environment number	30
$(N_{max}^{iter}, N_{max}^{step}, N_{max}^{env})$	(200,10240,600)
$(\Delta t, \Delta s^f)$	(0.1s, 0.5m)
(m, n) in (2)	(6,6)
$(\mathcal{R}_{step}, \mathcal{R}_{coll}, \mathcal{R}_{out}, \mathcal{R}_{suc})$	(-1,-15,0,5)
P in Section. IV-B	5

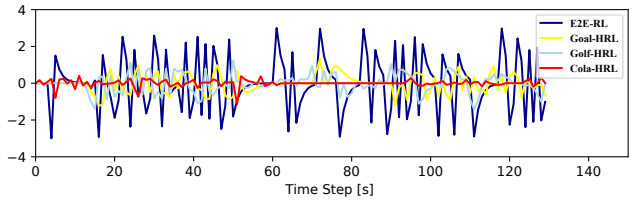
C. Baseline Methods

We argue that the task-oriented HRL method has a dependence on scenarios, so we choose goal-reach HRL methods and E2E-RL methods for comparison as follows:

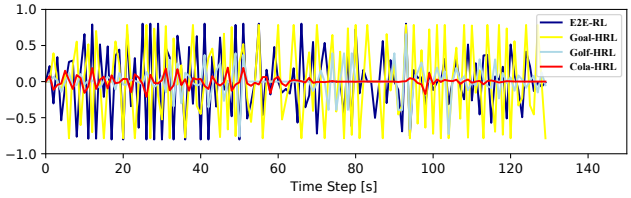
1) **Goal-HRL**: an original goal-reach method, which directly outputs a goal at each time step t, (x_t^{goal}, y_t^{goal}).



(a) **Cola-HRL** (b) **Golf-HRL** (c) **Goal-HRL** (d) **E2E-RL**



(e) Acceleration change rate (m/s^3)



(f) Steer change rate $(\deg/s)$

Fig. 6: Trajectory visualization and rate curve. (a) - (d) represent visualized trajectories of different methods in the cross scenario. The red represents the ego vehicle, while other colors represent the surrounding vehicles. 1 - 4 represents the time sequence. (e) and (f) represent acceleration change rate and steer change rate of corresponding methods.

(a) Meeting case (b) Merge case (c) Unprotected left-turn case

Fig. 7: Visualized trajectories of other three cases during one episode. The red represents the ego vehicle. Other colors indicate surrounding vehicles, where rectangles show their initial position.

We assume that the goal satisfies the uniform acceleration assumption. MPC controller is utilized to track it.

2) **Golf-HRL**: different with the Goal-HRL, the policy network outputs **goal**under the **F**renet coordinate system (golf) at each time step t, (s_t^{goal}, d_t^{goal}), which aims at limiting the goal in the driveable area.
3) **E2E-RL**: an E2E-RL method which directly outputs control commands at each time step t, (δ_t, acc_t), from perception scratches.

TABLE II: Comparison of average success rate

Method	Cross	Merge	Meeting	Left-turn
Cola	0.99 ± 0.01	1.00 ± 0.01	1.00 ± 0.01	0.98 ± 0.02
Golf	0.97 ± 0.10	0.79 ± 0.15	0.76 ± 0.25	0.92 ± 0.08
Goal	0.98 ± 0.02	0.83 ± 0.33	0.96 ± 0.02	0.70 ± 0.21
E2E	0.91 ± 0.05	0.91 ± 0.08	0.94 ± 0.02	0.89 ± 0.15

D. Performance Comparison

Cases belonging to the same scenario are trained simultaneously, where Fig. (5) shows the average success rate in the training process. If the ego vehicle does not collide and reaches the target area stably, this episode is defined as "success." Combined with the average success rate of each method shown in Table (II), the proposed Cola-HRL has the best performance in terms of convergence speed, training stability, and average success rate. The comparison between Goal-HRL and Golf-HRL shows that sampling in the Frenet coordinate system can limit action exploration space and achieve faster convergence, while the average success rate will decrease. In contrast, Cola-HRL utilizes the continuous-lattice module to compensate the exploration space in the driveable area, which could achieve the best average success rate and training performance.

Furthermore, Table (III) quantitatively compares the mean and variance of **s**teering angle **c**hange **r**ate (**scr**) $[rad/s]$, and **a**cceleration **c**hange **r**ate (**acr**) $[m/s^3]$ during the evaluation

TABLE III: Comparison of quantitative metrics

Comparative Scenarios	Cola-HRL		Golf-HRL		Goal-HRL		E2E-RL	
	scr	acr	scr	acr	scr	acr	scr	acr
Cross	0.05 ± 0.01	0.26 ± 0.03	0.12 ± 0.01	0.74 ± 0.09	0.36 ± 0.05	0.78 ± 0.07	0.39 ± 0.08	2.58 ± 0.11
Merge	0.07 ± 0.01	0.47 ± 0.05	0.21 ± 0.02	1.00 ± 0.08	0.41 ± 0.01	0.99 ± 0.10	0.47 ± 0.01	2.78 ± 0.04
Meeting	0.06 ± 0.01	0.27 ± 0.02	0.29 ± 0.04	1.03 ± 0.09	0.39 ± 0.08	0.98 ± 0.12	0.44 ± 0.10	2.98 ± 0.17
Left-turn	0.06 ± 0.01	0.02 ± 0.01	0.30 ± 0.02	0.58 ± 0.08	0.37 ± 0.06	0.61 ± 0.09	0.36 ± 0.07	2.41 ± 0.13

period. The **scr** and **acr** can qualitatively describe the stability and comfort during the driving process, respectively. The value is smaller, and the performance is better. Compared with the E2E-RL method, Goal-HRL significantly reduces acr by assuming uniform acceleration on generated goals. Golf-HRL has a significant decrease in scr compared with Goal-HRL, because of which the Frenet coordinate system limits the exploration and generation space of goals. It shows that the restriction from road topology is effective. In contrast, the proposed Cola-HRL is more stable in scr and acr. It can be concluded that goals generated by Cola-HRL can make control commands from the lower controller more stable and comfortable.

E. Advantage analysis

Fig.(6) shows visualized trajectories of one cross case, acr curves, and scr curves. Visualized trajectories in other cases are shown in Fig. 7. Besides, Table. (IV) shows the comfortable index I_c, where I_c is defined as the root mean square of longitudinal and lateral acceleration of all points on a trajectory.

Compared with E2E-RL, HRL methods generally have smaller acr due to the reasonable velocity assignment on goals. Significantly, the Cola-HRL has smaller scr, acr, and comfortable index, I_c [37], showing better stability and comfort. Furthermore, Cola-HRL tends to drive along the lane's centerline, which implies that it will be more compliant with the traffic rules. Golf-HRL, the goal of which is generated under the Frenet coordinate system, prefers to detour in the lane rather than directly traverse compared with Goal-HRL, so the corresponding scr is also smaller. The trajectory of E2E-RL has apparent jitter with higher scr and acr.

TABLE IV: Comfortable Index Comparison

Method	Cola-HRL	Golf-HRL	Goal-HRL	E2E-RL
I_c	0.28	2.06	2.33	4.02

F. Calculation time analysis

Our proposed Cola-HRL has good interpretability and can meet the real-time demand of autonomous vehicles in practical application, where the Table (V) shows the time consumption of each module. In each forward inference, the high-level layer generates 50 points in the next $5s$ with a time interval of $0.1s$. Then, the actuator in the low-level layer can recurrently solve control commands at the frequency of 100Hz, which could perfectly satisfy the practical requirement. In contrast, the control frequency of E2E-RL will be limited by the inference time of networks, which may lead to danger and reduce robustness.

TABLE V: Calculation Time

Different module	Vectorized extract	Policy inference	Goal generation	Control command
Time (ms)	7.1	13	16.5	9.3

G. Supplementary experiment

We also compare Cola-HRL with the Goal-HRL method using an optimization-based method [38] to deal with the generation of goals. We utilize the same action space with the Golf-HRL method to train it in meeting and cross scenarios, considering Ackerman kinematic and collision constraints. The former can complete the task while the latter cannot. Table VI shows the result in the meeting scenario. Limited by the Ackerman kinematic and sampling constraints under the Frenet coordinate system, scr is small, while acr delivers poor performance. The low success rate shows that the optimization-based method has a limited impact on the reachability of goals. However, we think the optimization process may lead to the mismatch between the RL policy and the actual action, resulting in incorrect environmental feedback to RL training, thus affecting the convergence process.

TABLE VI: Results of Optimization-based HRL

Scenario	Average Success Rate	scr	acr
Meeting	0.57 ± 0.12	0.06 ± 0.01	1.23 ± 0.10

VI. Conclusion

This paper propose a Continuous-Lattice hierarchical RL method for autonomous driving tasks to make high-quality decisions in various scenarios. We utilize a continuous-lattice module to ensure the temporal and spatial reachability of generated goals. We reconstruct traffic scenarios based on real-world HD maps to train and test our proposed method. Our method is effective in various realistic simulation scenarios. Compared with E2E-RL and other Goal-HRL methods, the proposed Cola-HRL shows better performance and driving behaviors in comfort and stability.

VII. Acknowledgement

This work was supported in part by the National Key Research and Development Program of China (Grant NO. 2020AAA0108104) and Alibaba Innovative Research (AIR) Program.

References

[1] M. Buehler, K. Iagnemma, and S. Singh, *The DARPA urban challenge: autonomous vehicles in city traffic*. springer, 2009, vol. 56.

[2] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton *et al.*, "Mastering the game of go without human knowledge," *nature*, vol. 550, no. 7676, pp. 354–359, 2017.

[3] Y. Guan, Y. Ren, Q. Sun, S. E. Li, H. Ma, J. Duan, Y. Dai, and B. Cheng, "Integrated decision and control: Towards interpretable and computationally efficient driving intelligence," *arXiv preprint arXiv:2103.10290*, 2021.

[4] Z. Zhang, A. Liniger, D. Dai, F. Yu, and L. Van Gool, "End-to-end urban driving by imitating a reinforcement learning coach," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 15 222–15 232.

[5] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[6] A. Kendall, J. Hawke, D. Janz, P. Mazur, D. Reda, J.-M. Allen, V.-D. Lam, A. Bewley, and A. Shah, "Learning to drive in a day," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 8248–8254.

[7] J. Chen, B. Yuan, and M. Tomizuka, "Model-free deep reinforcement learning for urban autonomous driving," in *2019 IEEE intelligent transportation systems conference (ITSC)*. IEEE, 2019, pp. 2765–2771.

[8] J. Duan, S. E. Li, Y. Guan, Q. Sun, and B. Cheng, "Hierarchical reinforcement learning for self-driving decision-making without reliance on labelled driving data," *IET Intelligent Transport Systems*, vol. 14, no. 5, pp. 297–305, 2020.

[9] Y. Chen, C. Dong, P. Palanisamy, P. Mudalige, K. Muelling, and J. M. Dolan, "Attention-based hierarchical deep reinforcement learning for lane change behaviors in autonomous driving," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2019, pp. 0–0.

[10] K. Rezaee, P. Yadmellat, M. S. Nosrati, E. A. Abolfathi, M. Elmahgiubi, and J. Luo, "Multi-lane cruising using hierarchical planning and reinforcement learning," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE, 2019, pp. 1800–1806.

[11] C. You, J. Lu, D. Filev, and P. Tsiotras, "Highway traffic modeling and decision making for autonomous vehicle using reinforcement learning," in *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018, pp. 1227–1232.

[12] J. Chen, Z. Wang, and M. Tomizuka, "Deep hierarchical reinforcement learning for autonomous driving with distinct behaviors," in *2018 IEEE intelligent vehicles symposium (IV)*. IEEE, 2018, pp. 1239–1244.

[13] Z. Qiao, Z. Tyree, P. Mudalige, J. Schneider, and J. M. Dolan, "Hierarchical reinforcement learning method for autonomous vehicle behavior planning," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 6084–6089.

[14] B. Gangopadhyay, H. Soora, and P. Dasgupta, "Hierarchical program-triggered reinforcement learning agents for automated driving," *arXiv preprint arXiv:2103.13861*, 2021.

[15] Z. Wang, Y. Zhuang, Q. Gu, D. Chen, H. Zhang, and W. Liu, "Reinforcement learning based negotiation-aware motion planning of autonomous vehicles," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 4532–4537.

[16] J. Li, L. Sun, J. Chen, M. Tomizuka, and W. Zhan, "A safe hierarchical planning framework for complex driving scenarios based on reinforcement learning," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 2660–2666.

[17] Z. Qiao, J. Schneider, and J. M. Dolan, "Behavior planning at urban intersections through hierarchical reinforcement learning," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 2667–2673.

[18] C.-J. Hoel, K. Driggs-Campbell, K. Wolff, L. Laine, and M. J. Kochenderfer, "Combining planning and deep reinforcement learning in tactical decision making for autonomous driving," *IEEE transactions on intelligent vehicles*, vol. 5, no. 2, pp. 294–305, 2019.

[19] K. B. Naveed, Z. Qiao, and J. M. Dolan, "Trajectory planning for autonomous vehicles using hierarchical reinforcement learning," in *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*. IEEE, 2021, pp. 601–606.

[20] J. Wang, Y. Wang, D. Zhang, Y. Yang, and R. Xiong, "Learning hierarchical behavior and motion planning for autonomous driving," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 2235–2242.

[21] L. García Cuenca, E. Puertas, J. Fernandez Andres, and N. Aliane, "Autonomous driving in roundabout maneuvers using reinforcement learning with q-learning," *Electronics*, vol. 8, no. 12, p. 1536, 2019.

[22] A. E. Sallab, M. Abdou, E. Perot, and S. Yogamani, "Deep reinforcement learning framework for autonomous driving," *Electronic Imaging*, vol. 2017, no. 19, pp. 70–76, 2017.

[23] C. You, J. Lu, D. Filev, and P. Tsiotras, "Advanced planning for autonomous vehicles using reinforcement learning and deep inverse reinforcement learning," *Robotics and Autonomous Systems*, vol. 114, pp. 1–18, 2019.

[24] E. Perot, M. Jaritz, M. Toromanoff, and R. De Charette, "End-to-end driving in a realistic racing game with deep reinforcement learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2017, pp. 3–4.

[25] T. G. Dietterich, "Hierarchical reinforcement learning with the maxq value function decomposition," *Journal of artificial intelligence research*, vol. 13, pp. 227–303, 2000.

[26] R. S. Sutton, D. Precup, and S. Singh, "Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning," *Artificial intelligence*, vol. 112, no. 1-2, pp. 181–211, 1999.

[27] P.-L. Bacon, J. Harb, and D. Precup, "The option-critic architecture," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, no. 1, 2017.

[28] N. Gopalan, M. L. Littman, J. MacGlashan, S. Squire, S. Tellex, J. Winder, L. L. Wong *et al.*, "Planning with abstract markov decision processes," in *Twenty-Seventh International Conference on Automated Planning and Scheduling*, 2017.

[29] W. Tan, X. Fang, W. Zhang, R. Song, T. Chen, Y. Zheng, and Y. Li, "A hierarchical framework for quadruped locomotion based on reinforcement learning," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 8462–8468.

[30] X. Zhang, X. Guo, Y. Fang, and W. Zhu, "Reinforcement learning-based hierarchical control for path following of a salamander-like robot," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 6077–6083.

[31] M. Werling, J. Ziegler, S. Kammel, and S. Thrun, "Optimal trajectory generation for dynamic street scenarios in a frenet frame," in *2010 IEEE International Conference on Robotics and Automation*. IEEE, 2010, pp. 987–993.

[32] J. Gao, C. Sun, H. Zhao, Y. Shen, D. Anguelov, C. Li, and C. Schmid, "Vectornet: Encoding hd maps and agent dynamics from vectorized representation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 525–11 533.

[33] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[34] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[35] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[36] E. Liang, R. Liaw, R. Nishihara, P. Moritz, R. Fox, K. Goldberg, J. Gonzalez, M. Jordan, and I. Stoica, "Rllib: Abstractions for distributed reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2018, pp. 3053–3062.

[37] I. S. Organization, *ISO 14040: Environmental Management-Life Cycle Assessment-Principles and Framework*, 1997.

[38] C. Rösmann, W. Feiten, T. Wösch, F. Hoffmann, and T. Bertram, "Efficient trajectory optimization using a sparse model," in *2013 European Conference on Mobile Robots*. IEEE, 2013, pp. 138–143.