SPATIALLY AND SEAMLESSLY HIERARCHICAL REINFORCEMENT LEARNING FOR STATE SPACE AND POLICY SPACE IN AUTONOMOUS DRIVING

Anonymous authors
Paper under double-blind review

ABSTRACT

Despite advances in hierarchical reinforcement learning, its applications to path planning in autonomous driving on highways are challenging. One reason is that conventional hierarchical reinforcement learning approaches are not amenable to autonomous driving due to its riskiness: the agent must move avoiding multiple obstacles such as other agents that are highly unpredictable, thus safe regions are small, scattered, and changeable over time. To overcome this challenge, we propose a spatially hierarchical reinforcement learning method for state space and policy space. The high-level policy selects not only behavioral sub-policy but also regions to pay mind to in state space and for outline in policy space. Subsequently, the low-level policy elaborates the short-term goal position of the agent within the outline of the region selected by the high-level command. The network structure and optimization suggested in our method are as concise as those of single-level methods. Experiments on the environment with various shapes of roads showed that our method finds the nearly optimal policies from early episodes, outperforming a baseline hierarchical reinforcement learning method, especially in narrow and complex roads. The resulting trajectories on the roads were similar to those of human strategies on the behavioral planning level.

1 INTRODUCTION

The essential idea of hierarchical reinforcement learning (HRL) is to find a proper hierarchy of abstractions for tasks on loosely coupled Markov decision process (MDP) (Parr & Russell, 1998). One predominant technique is temporal abstraction of policies, in which high-level command operates on Semi-MDP that has lower temporal resolution than MDP, and the low-level policy specifies primitive actions more frequently under the command (Sutton et al., 1999b). Another technique is state abstraction, for which similar concepts are suggested in Dayan & Hinton (1992), Singh et al. (1994), and Dietterich (2000). Regarding the two types of abstraction, Sutton et al. (1999b) proposed option framework, where high-level policy selects an option that consists of an initiation set, a sub-policy, and a termination condition. In respect of rewarding, two major directions exist: reward hiding (Dayan & Hinton, 1992) empowers the manager to independently reward sub-managers according to their compliance with the commands, while MAXQ value decomposition (Dietterich, 2000) assigns credit for task rewards by temporally decomposing a state-action value in the parent task into expected total rewards during the execution of sub-policy and after the execution of sub-policy.

Advances in hierarchical approaches with deep neural networks, rooted in the aforementioned techniques, engage with multiple challenges. Kulkarni et al. (2016) presented hierarchical deep Q networks where every policy is learned as a separate DQN and intrinsic rewards encourage sufficient exploration in the subtask. For tasks on MDPs that are densely coupled in temporal dimension, high-level goals are modified by a network module (Nachum et al., 2018). For discrete sub-policies, option-critic (Bacon et al., 2017) learns the options end-to-end by policy gradient, without having to manually define options. More diverse strategies became available through the use of a maximum entropy objective that enables a latent layer of high-level policy networks to directly control the sub-policy (Haarnoja et al., 2018). HiPPO (Li et al., 2020) adopted the control, formulated an unbiased latent dependent baseline, and derived a new hierarchical policy gradient that allows joint training of all levels with proximal policy optimization (PPO).

Recent approaches of HRL for autonomous driving (AD) (Paxton et al., 2017; Chen et al., 2019; Rezaee et al., 2019) kept up with some of these advances but did not fully reckon with following task-specific attributes. In AD, vehicle states can change unpredictably, rapidly, and frequently due to behaviors of ego agent and others. This does not satisfy the prerequisite of canonical HRL, a loosely coupled MDP. Additionally, the agent must drive through obstacles including other agents and avoid collision. Otherwise, it will be given a huge penalty and encounter the end of an episode without reaching the goal that accompanies delayed positive rewards. Therefore, the agent is demanded to keep an appropriate distance from obstacles concerning expected returns. In this respect, one way to efficiently learn navigation in AD is to sample the short-term goal only in collision-free regions which are relatively safe in near future and modify or re-sample them at subsequent time steps.

Even when given the knowledge about collision-free regions, the optimization is still tricky with gradient-based methods when the structure of regions is complex. In dense traffic, locally optimal intermediate goal position may exist for every inter-vehicle region (IVR) on a lane due to the contradiction of policy gradient directions between huge and hard-constrained objectives about collision avoidance and small and soft-constrained objectives about travel time reduction. On this ground, choosing one region among reachable regions in near future can be an effective task decomposition where the subtask is to find the local optimum in the given IVR.

Thus, we propose a reinforcement learning method with a two-level hierarchy that accommodates those attributes of AD on highways. Our agent can estimate the value of multi-lateral strategies including short-term goals that can be local optima in IVRs and select the best one given the circumstance at every time step. Our contribution to HRL and AD is three-fold:

We rethink the role of hierarchical policies, and propose a deep reinforcement learning method of spatial hierarchy. Our high-level policy selects a combination of behavioral sub-policy and its components, the IVRs to be used as a part of state for the two levels and as the outline of the sub-policy space. Our low-level policy generates an action in continuous space by elaborating within the outline, using a unique network structure and algorithm that we name neural elaboration within reinforced outline (NEWTRO).

We suggest a seamless HRL method for smooth state transition in short-term goal planning. The high-level policy selects one of either the current subspace or a candidate goal subspace from their subspace features, and the low-level policy can bridge the two subspaces. By sharing the value network for the two-level policies, our network structure and memory are designed to be as concise as those of a single-level actor critic, and learned without temporal abstraction.

Our methods showed drastic improvement of performance with fast optimization speed. In experiments, the agent using our method received higher rewards from early episodes than a baseline HRL, especially on narrow and complex roads. The traveling trajectory of the agent using our method followed general human tactics, while the agent using the existing HRL did not.

2 Background and related work

In this section, we introduce task definition in several branches of reinforcement learning (RL) and advantage actor-critic underlying our method to help understanding it. Then we compare our method in comparison with other HRL methods for autonomous driving on highways.

2.1 Task definition in branches of reinforcement learning

Canonical RL consists of the agent making action $a \in \mathcal{A}$ and the environment of the state $s \in \mathcal{S}$, interacting with each other. The environment follows MDP: it assumes the situation that given the current state s_t at time t, the next state s_{t+1} does not depend on the past states and actions. When the state is fully observable, the policy π can be established from the s to make a. In multi-agent RL, MDP is expanded to the actions of multi-agents a, where the next state is determined by s and a. In certain RLs with feature set or hierarchical RL, s can be abstracted to parts of s, or the abstracted state can be defined in different space from (S). For goal-based RL methods of Universal Value Function Approximators (Schaul et al., 2015) or the policies given goal from the high level in hierarchy, the state-goal value $V(s, g)$ or the state-action-goal value $Q(s, a, g)$ can be estimated

and learned. In single-agent RL or multi-agent RL with self-interested agents, a typical objective is the expected long-term discounted rewards $\rho(\pi) = \mathbb{E}\{\sum_{t=1}^{\infty} \gamma^{t-1} r_t | s_0, \pi\}$. In RL with multi-objectives, the reward can be multi-dimensional, expressible as $r_t = \{r_{1,t}, r_{2,t}, \cdots, r_{N-1,t}, r_{N,t}\}$ for N number of objectives. In this paper, we design a spatially hierarchical RL agent receiving a long-term goal and multi-objectives who operates on the environment with self-interested multi-agents.

2.2 POLICY GRADIENT AND ADVANTAGE ACTOR-CRITIC

Policy gradient methods aim to find optimal parametrized policies by performing gradient descent to optimize an objective. The policy gradient theorem of Sutton et al. (1999a) derives the gradient with respect to the parameters of stochastic policy θ as

$$\frac{\partial \rho(\pi_\theta)}{\partial \theta} = \sum_s d^{\pi_\theta}(s) \sum_a \frac{\partial \pi_\theta(s,a)}{\partial \theta} Q^\pi(s,a), \tag{1}$$

where $d^{\pi_\theta}(s)$ is the stationary distribution following the parameterized policy π_θ, and $Q^\pi(s,a)$ is the value of a state-action pair given a policy $\sum_{t=1}^{\infty} \mathbb{E}\{\gamma^{t-1} r_t - \rho(\pi) | s, a, \pi\}$.

For a designated start state s_0, only the long-term rewards are cared for $Q^\pi(s,a)$. ***** add math!!! In Konda (2002), $Q^\pi(s,a)$ is designed as a feature vector ϕ_θ, which is critic. Advantage actor-critic methods employ the advantage function $A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s)$, where $V^\pi(s)$ is the state value used as the base line. The underlying reason is that the policy gradient suffers from its high variance because of the dependency on the state-action values during a trajectory. In our method we adopt n-step advantage estimate (Schulman et al., 2015) because it fits our hierarchical actor-critic design aiming at efficient policy decomposition for domain-specific subspaces.

2.3 HIERARCHICAL POLICIES FOR AUTONOMOUS DRIVING ON HIGHWAYS

Reinforcement learning approaches have been adopted for autonomous driving on highways to teach a vehicle agent several behaviors, such as cruise control (Chen et al., 2017; Zhao et al., 2017), lane keeping (Kendall et al., 2019), lane changing (Wang et al., 2018), and traffic merging (Wang & Chan, 2017). Recently, Paxton et al. (2017); Rezaee et al. (2019); Chen et al. (2019) suggested hierarchical methods where the high-level policies commonly function as a type of behavior planner, while the low-level policies take different roles: primitive control (Paxton et al., 2017; Chen et al., 2019) and motion planning (Rezaee et al., 2019).

The method of Paxton et al. (2017) generates motion plans by using tree search, where the high-level options and the low-level actions are selected in turn to extract the best option sequence. For the low-level policy, world states are abstracted to the set of continuous, logical, and agent-relative features. Rezaee et al. (2019) designed a cruising method on multi-lanes, where the high-level behavioral planner makes a decision on a discretized state-action space. They regarded path planning as a sequence of symbolically punctuated behaviors, and handed over authority of reactive planning to the low-level policy. They were concerned about specifying deadlines of a fixed expiration time of the high-level command in AD, so designed the low-level policy as a motion planner that prioritizes safety over the high-level decisions. Chen et al. (2019) designed networks with spatial and temporal attention mechanism for input images of the front view. While a CNN encoder is shared for actor and critic, the spatial and temporal attention is applied only for hierarchical policies. On top of the encoder, salient regions are chosen by spatial attention networks, and the sequence of the regions runs through LSTM, whose outputs are applied with temporal attention.

Our HRL method is limited to short-term goal planning because no temporal abstraction is used. Putting aside the concerns on the temporal aspect in Rezaee et al. (2019) and Chen et al. (2019), our HRL method focuses on policies with spatial decomposition because we consider that for driving on the highway with dense traffic, spatially decomposing a state can be much easier than temporally decomposing a sequence of multi-dimensional states that are entangled with hierarchical actions of multi-agents. For state abstraction, rather than using popular approaches of abstraction for sub-policy or spatial attention networks, we aimed at the high-level policy that can choose its own features resembling our eyes looking at several focus points and deciding to return to the salient one.

In previous research, the hierarchical policies have respective roles in different levels, such as abstracted behavior, motion planning, and control. In this work, however, the two-level policies work in the same policy space, which is the short-term goal generation.

3 ALGORITHM

In the temporal respect, if a short-term goal is reachable without the episode ending midway, the expected returns of choosing a short-term goal position are the sum of expected returns during a behavioral motion before reaching the goal and expected returns at the future state when ego vehicle reaches the goal position. Although other agents are highly unpredictable, learning through multiple explorations enables estimating the expected returns of the goal selection given a state s_t, a long-term goal region l_t, and a behavioral motion to reach its short-term goal g_t at time t. The details of a long-term goal region are in Section A.5 of Appendix, and we fix the region as l for each episode. In this setting, our HRL algorithm additionally employs the features of inter-vehicle regions (IVRs) that are subspaces of the action space acquired from our method using domain knowledge. The details of IVRs are in Section A.4 of Appendix.

The subspaces are used in three ways, 1) as additional state features of the IVR that ego-vehicle currently belongs to c_t, 2) as selectable state features to pay mind to, and 3) as the outline that confines the low-level policy space. The high-level policy π chooses one of the candidates $h \in \mathbb{H}$ whose behavioral mode $b \in \mathbb{B}$ specifies a set of features of available subspaces to pay mind to, $\mathbb{M}_b$, and the subspace for the outline, o_b. Thus, a candidate command is defined as $h = \{b, m_b, o_b\}$ where features of subspace to pay mind to is following $m_b \in \mathbb{M}_b$. Table 1 in Appendix lists a set of candidates determined by the condition of current state.

The high-level policy π at time step t is to select the candidate of maximum state-candidate value Q_h:

$$\begin{aligned} \pi &= \arg\max_{h_t \in \mathbb{H}_t} Q(l_t, s_t, c_t, h_t) \\ &\simeq \arg\max_{\tilde{h}_t \in \tilde{\mathbb{H}}_t} Q(l_t, s_t, c_t, \tilde{h}_t) \\ &= \arg\max_{\tilde{h}_t \in \tilde{\mathbb{H}}_t} Q(l_t, s_t, c_t, m_{b,t}, b_t), \end{aligned} \tag{2}$$

where $\tilde{h}$ is a partial candidate $\{b, m_b\}$, and the command is chosen by exhaustive search for all possible partial candidates $\tilde{\mathbb{H}}_t$ at time step t. The features of outline region o_b are not explicitly indicated in inputs for the value estimation, since o_b is either the current IVR or an IVR in mind, and each behavioral mode includes the decision whether to stay in the current IVR or to move on to IVR in mind given the features of both IVRs as inputs c_t and $m_{b,t}$. The information on encoding process is described in Section 4 in detail.

In accordance with the high-level policy, our low-level stochastic policy search ϕ at time step t generates a two-dimensional goal position $a_t \in \mathbb{N}^2$, where $\mathbb{N} = (0, 1)$, given a high-level command h_t, such that

$$\begin{aligned} a_t &\sim \phi(l_t, s_t, c_t, h_t) \\ &= \phi(l_t, s_t, c_t, b_t, m_{b,t}, o_{b,t}). \end{aligned} \tag{3}$$

The normalized goal position a_t is interpreted as a position in the local coordinate of the outline region $o_{b,t}$, and our manually designed function $\mathcal{T} : \mathbb{N}^2 \to \mathbb{R}^2$ transforms the local position a_t to the corresponding position in the global coordinate g_t given $o_{b,t}$:

$$g_t = \mathcal{T}(a_t, o_{b,t}). \tag{4}$$

The details of the transformation function $\mathcal{T}$ are in Subsection A.2 and A.4 of Appendix. For control of vehicle in reaching the goal g_t, the outline region $o_{b,t}$ also determines the target heading direction ψ.

Given high-level command at every time step, the actual goal position of the agent is determined by ϕ, thus the high-level state-command value is equal to the state value of low-level policy working in accordance with the command,

$$Q_h(s_t, h_t) = V_l(s_t, h_t). \tag{5}$$

By designing ϕ to be learned through actor-critic with the state value estimation, all state-candidate values for our high-level policy can be estimated. When any low-level state value is well estimated, and the low-level policy is locally optimal in a given inter-vehicle region, we can naturally assume that the high-level policy that selects the candidate of maximum state-candidate value as the command is also optimal.

To achieve this global optimum requires nothing but quality learning through actor-critic for action in continuous space, and we adopted proximal policy optimization (PPO) (Schulman et al., 2017) to prevent an abrupt decrease in the optimality of learning. The surrogate objective of PPO for our advantage actor-critic given high-level command is

$$L^{CLIP}_{NEWTRO}(\theta) = \mathbb{E}_{\tau} \sum_{t=t_0}^{T-1} min\{w_t(\theta)A_l(l_t, s_t, c_t, \tilde{h}_t, a_t), w_t^{clip}(\theta)A_l(l_t, s_t, c_t, \tilde{h}_t, a_t)\}, \quad (6)$$

where the clipped ratio is defined as $w_t^{clip} = clip(\frac{\pi_\theta(a_t|l_t,s_t,c_t,\tilde{h}_t)}{pi_{\theta_{old}}(a_t|l_t,s_t,c_t,\tilde{h}_t)}, 1-\epsilon, 1+\epsilon)$ and the trajectory is given for time t_0 to T.

The advantage is equal to the difference between the expected return and the state value given high-level command,

$$A_l(l_t, s_t, c_t, \tilde{h}_t, a_t) = G_t - V_l(l_t, s_t, c_t, \tilde{h}_t). \quad (7)$$

Canonical policy gradient methods (Sutton et al., 1999a; Konda, 2002) learn from recent trajectories for smooth update of policy parameters, and PPO employs a type of estimator for expected returns G introduced in past work (Williams, 1992; Mnih et al., 2016b). We expand the estimator as

$$\hat{G}_t = r_t + \gamma r_{t+1} + \cdots + \gamma^{T-t+1} r_{T-1} r_{t+1} + \gamma^{T-t} V_{max}(l_T, s_T, c_T, \tilde{\mathbb{H}}_T). \quad (8)$$

In the last term in the right side of Equation 8, $V_{max}(l_T, s_T, c_T, \tilde{\mathbb{H}}_T)$ is the maximum state-candidate value at time step T acquired from Equation 2 if T is a non-terminal time step, otherwise 0. Fortunately, this type of expected returns are not tricky to save and batch, since the returns do not require the set of $\tilde{\mathbb{H}}$ from t to $T-1$ that could include varying number of candidate regions to pay mind to for each behavior through time. In practice, G_t is easily implemented by adding $\tilde{\mathbb{H}}_T$ to the batched trajectory at the training time step.

Algorithm 1: SHRL with proximal policy optimization

Initialize the parameters of a type of Actor-Critic networks, $\Theta = \{\theta, \vartheta, ...\}$
for *iteration=1,2,...* **do**
  Choose $h = \{b, m_b, o_b\} \in \mathbb{H}$ given l, s, c, b, and m_b by max-Q policy of parameters ϑ
  Choose a given l, s, c, b, m_b, and o_b, by policy search of parameters θ
  Transform a to the goal position g by $\mathcal{T}(a, o_b)$, where o_b defines the local coordinate for a
  Calculate the target heading direction ψ by using o_b
  Apply g and ψ to the controller of the agent
  Save the transition $< l, s, c, b, m_b, a, r >$ to the memory
  if *memory size is equal to* $T - t_0$ **then**
    Batch transitions, and put the batch and $\tilde{H}_T$ as training data
    Optimize for policy surrogate objective and critic loss L w.r.t. Θ
    Clear memory
  end if
end for

In brief, the main computation flow of our method is summarized in Algorithm 1, where θ and ϑ are parameters of the low-level actor and critic shared in the hierarchy, respectively.

4 DEEP NEURAL NETWORK STRUCTURE AND PROCESSING

The general learning model for AD deals with multi-modal features. In our environment, the model receives three types of inputs: information on vehicles, obstacle positions, and inter-vehicle regions

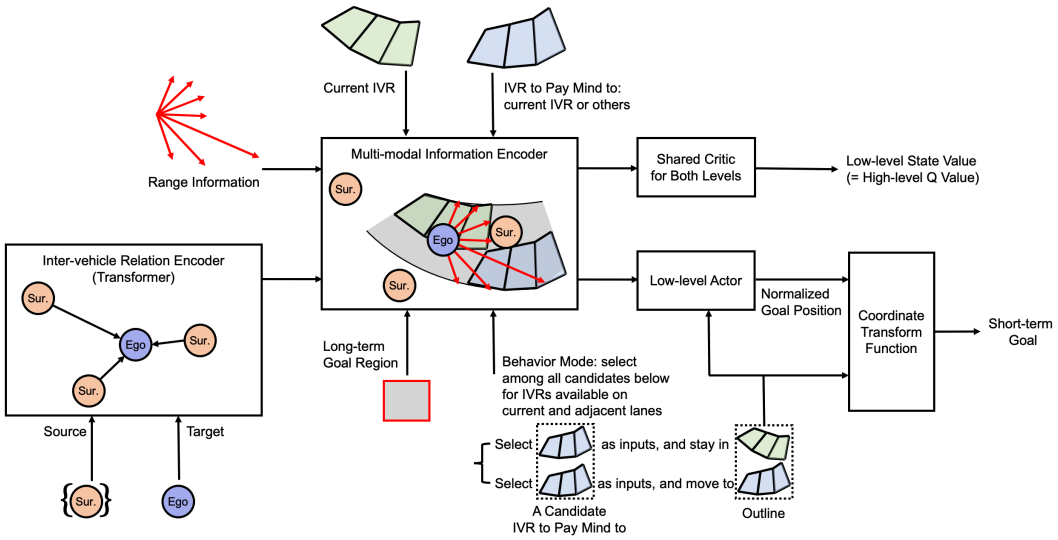

Figure 1: Model architecture for our method in autonomous driving

(IVRs). The detailed features are described in Subsection A.5 in Appendix. All types of information include two-dimensional global positions such that the relative information of all feature types is expected to be effectively encoded in our model. Our model consists of four distinctive modules: inter-vehicle relation encoder, multi-modal information encoder, policy networks, and value networks. The overall structure of the modules is described in Figure 1. Actor and critic are implemented with fully-connected layers, and other modules are explained in Subsections below. We practically employed double-critics for stable value learning, which is proposed for deep Q networks (van Hasselt et al., 2016).

4.1 Inter-vehicle relation encoder

Inter-vehicle relation encoder deals with feature information on ego vehicle and surrounding vehicles. To integrate the features of a varying number of surrounding vehicles in respect of ego vehicle, an architecture of graph attention networks, Transformer (Mnih et al., 2016a) is used for the encoder. Transformer was originally proposed for natural language processing, and Leurent & Mercat (2019) used it in AD to encode the state of vehicles to make high-level decisions for longitudinal control. Transformer consists of encoder layers and decoder layers, and other operations. We used one layer of decoder without encoding layers, where the $source$ inputs receive the features of surrounding vehicles and the $target$ inputs receive the features of ego vehicle. The outputs of inter-vehicle relation encoder are used as one type of the inputs for multi-modal information encoder.

4.2 Multi-modal information encoder

As seen in Figure 1, multi-modal information encoder integrates the long-term goal l, the vehicle encoding v, range information r, the IVR that the ego belongs to c, and the IVR in mind m, depending on the behavior mode b. The outputs are the encoding of goal, state, and the high-level command used for actor and critic. Each behavioral mode relates these features while interpreting them on the corresponding outline region whose features are received as c or m. For behavioral mode, we will present the performance two types of designs: 1) indexed selection for tabular encoding outputs, 2) network attention from the one-hot vector b on hidden layers, $a = f^{attend1}(b), h = f^{abstract_1}(m), and e = f^{abstract_2}(h \odot a)$, which is similarly used in HRL (Vezhnevets et al., 2017; Earle et al., 2018). The latter two designs allow that common features and skills are shared across behavioral modes. The three different designs were compared in our experiment to find a better structure.

4.3 NEURAL ELABORATION WITHIN REINFORCED OUTLINE (NEWTRO)

The actor plays the role of the low-level policy, generating a goal for the two-dimensional center position, which should be confined to the inside of the outline of the given IVR. For the confinement, we adopt a simple normalization, applying a sigmoid function with the input coefficient $sigmoid(cx)$ as the activation function for the outputs of the actor network. This compares favorably with constraint optimization methods in that it provides faster optimization speed, and that the corners of the ego vehicle mostly stay inside the outline since the goal does not shift closer to the border of the outline unless it is inevitable and values of huge magnitudes are output before the activation. One problem of simply using the output normalization is that both the positions of multi-modal input features that construct the encoding along with behavior and the featuring positions of the outline IVR are specified with respect to the global coordinate frame, while the outputs of the actor are normalized between 0 and 1. Thus, we devise an additional normalization technique for the multi-modal information encoding e to train the actor to be invariant of the varying size of the outline. Before being used as inputs for the actor network, e is simply copied and divided by the length and sampled widths of the outline. An IVR is given as the positions of the left and right sides sampled for N times along the progressing direction of vehicles. The length of the outline along the progressing direction, l_o, and the widths of the region, $w_{o,1}, w_{o,2}, \cdot, w_{o,N-1}, w_{o,N}$, which are the distances between the two sides of each sample, are defined by the method in Subsection A.4 and A.4 of Appendix. Thus, e of d dimensions is copied $1 + N$ times, divided by these length and widths, and concatenated so that e turns to $e/l_o, e/w_{o,1}, e/w_{o,2}, \cdot, e/w_{o,N-1}, e/w_{o,N}$. Using the copied encodings of $(1 + N)$ dimension normalized by the outline that is selected by the reinforced high-level policy, the local sub-policy network elaborates the outline by generating a goal position.

5 EXPERIMENTAL RESULTS

5.1 ENVIRONMENT

In our environment, agents appear at random lanes at the left end of the road at random time steps, and an agent ends the episode when it reaches the goal region, which is the right end of the road, or collides with either other vehicles or lane shoulders. When a collision occurs, vehicles stay stuck on the road for a fixed delay time and then disappear from the road. The reward given to the agent differs depending on the significance of objectives. Big reward or penalty is given at the end of the episode while small rewards or penalties is given during the episode. The details of rewards are described in the table in Section A.5 of Appendix. We prepared environments with various types of roads, which are in Figure 4 of Appendix. To average the optimization of multi-agents and simplify the comparison of performance for different methods in Figure 2, we adopted a joint training method of Mnih et al. (2016a), which is originally for single-agent in multi-environments.

5.2 EVALUATION

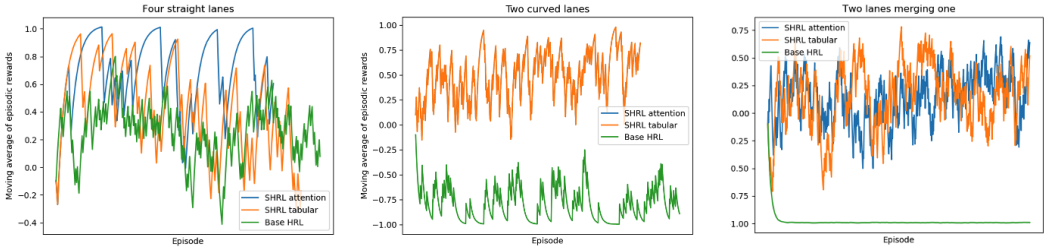

Figure 2: Training evolution of our methods and a baseline on three types of roads

We compare various structures of our algorithm to a baseline HRL method with domain knowledge about sub-goals. Figure 2 shows the performance of agents on the roads of four straight lanes (left), two curved lanes (center), and two lanes merging into one (right). During the joint training of all

vehicle agents using each method, we plotted moving average of the sum of returns for each episode with initiation to 0 and $alpha = 0.1$.

The total episode rewards are close to 1 when an agent successfully learns the task including goal reaching, and when an agent fails and have an accident, -1. In comparison with the baseline HRL, our methods showed decent performance from early episodes for all types of roads taking advantage of the outline. After then, the episodic returns tended to drop for a moment in cases, and steadily increase through learning while adjusting to multi-agent interactions. We also tested the structural design of separate modules for each mode, but it was not as effective as the two methods. The baseline method showed fluctuating performance for straight lanes, low performance for the curved lanes, and failed to learn in the merging lanes since safe short-term goals were hardly constructed as the chance of collision increases.

5.3 VISUALIZATION

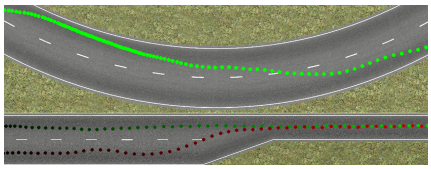

Figure 3: Trajectories of vehicles on two curved lanes

We qualitatively examined the traveling trajectory of vehicle agents on curved two lanes in our method against that of the baseline HRL method. General human tactics for the agent is to take the shortest possible path, which could be a straight line in the optimal environment. The agent should consider collision avoidance, travel time reduction, and compliance with speed limits. In our experiment, two vehicle agents start to drive from the left end of each lane almost simultaneously with slight randomness. When an inter-vehicle region on the shortest possible path is available, then the agent will move on to occupy the region. Figure 3 shows agents using our method. The agent on two curved lane tried to take the inner lane at first, and then the outer lane in the end. The red agent on merging lane wait until the green agents pass the lane in the middle, and follow the green agent. Agents using the baseline HRL, with frequent accidents, and seemingly not following the optimal path even when they complete the journey.

6 DISCUSSION AND FUTURE WORK

We presented hierarchical policies that can utilize space decomposition of state and policy by selecting a subspace at the high level and learning low-level policy with the given subspace. Although the hierarchical policies are from different base value definitions that are high-level state-action value and low-level state value given command, we integrated them as the same network structure learned by single-level training. Thus, our method can be utilized as a part of a nested spatio-temporal hierarchy of reinforcement learning where inner spatial hierarchy is implemented with our method and outer temporal hierarchy is designed with canonical HRL methods.

For autonomous driving, our experiments showed that learning the short-term planning using domain knowledge about collision-free space and its decomposition is extremely efficient. We expect that the use of domain knowledge in deep reinforcement learning with integrated value in the hierarchy, which is seemingly close to the basis of human cognitive thinking and proven to be efficient can be applied to complex levels of reinforcement learning: model-based reinforcement learning with predictive control or self-interested multi-agent reinforcement learning in game-theoretic situations.

The limitation of our method is that a separate process for spatial decomposition is required, and that the performance highly depends on the decomposition ability. Effective automatic spatial decomposition methods through unsupervised learning or reinforcement learning will be able to generalize our HRL algorithm to different tasks.

REFERENCES

Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In Satinder P. Singh and Shaul Markovitch (eds.), *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, pp. 1726–1734. AAAI Press, 2017. URL http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14858.

Xin Chen, Yong Zhai, Chao Lu, Jianwei Gong, and Gang Wang. A learning model for personalized adaptive cruise control. In *IEEE Intelligent Vehicles Symposium, IV 2017, Los Angeles, CA, USA, June 11-14, 2017*, pp. 379–384. IEEE, 2017. doi: 10.1109/IVS.2017.7995748. URL https://doi.org/10.1109/IVS.2017.7995748.

Yilun Chen, Chiyu Dong, Praveen Palanisamy, Priyantha Mudalige, Katharina Muelling, and John M. Dolan. Attention-based hierarchical deep reinforcement learning for lane change behaviors in autonomous driving. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2019, Macau, SAR, China, November 3-8, 2019*, pp. 3697–3703. IEEE, 2019. doi: 10.1109/IROS40897.2019.8968565. URL https://doi.org/10.1109/IROS40897.2019.8968565.

Peter Dayan and Geoffrey E. Hinton. Feudal reinforcement learning. In Stephen Jose Hanson, Jack D. Cowan, and C. Lee Giles (eds.), *Advances in Neural Information Processing Systems 5, [NIPS Conference, Denver, Colorado, USA, November 30 - December 3, 1992]*, pp. 271–278. Morgan Kaufmann, 1992. URL http://papers.nips.cc/paper/714-feudal-reinforcement-learning.

Thomas G Dietterich. Hierarchical reinforcement learning with the maxq value function decomposition. *Journal of artificial intelligence research*, 13:227–303, 2000.

Adam C Earle, Andrew M Saxe, and Benjamin Rosman. Incremental hierarchical reinforcement learning with multitask lmdps. 2018.

Tuomas Haarnoja, Kristian Hartikainen, Pieter Abbeel, and Sergey Levine. Latent space policies for hierarchical reinforcement learning. In Jennifer G. Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1846–1855. PMLR, 2018. URL http://proceedings.mlr.press/v80/haarnoja18a.html.

Alex Kendall, Jeffrey Hawke, David Janz, Przemyslaw Mazur, Daniele Reda, John-Mark Allen, Vinh-Dieu Lam, Alex Bewley, and Amar Shah. Learning to drive in a day. In *International Conference on Robotics and Automation, ICRA 2019, Montreal, QC, Canada, May 20-24, 2019*, pp. 8248–8254. IEEE, 2019. doi: 10.1109/ICRA.2019.8793742. URL https://doi.org/10.1109/ICRA.2019.8793742.

Vijaymohan Konda. *Actor-critic algorithms*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 2002. URL http://hdl.handle.net/1721.1/8120.

Tejas D Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper/2016/file/f442d33fa06832082290ad8544a8da27-Paper.pdf.

Edouard Leurent and Jean Mercat. Social attention for autonomous decision-making in dense traffic. *CoRR*, abs/1911.12250, 2019. URL http://arxiv.org/abs/1911.12250.

Alexander C. Li, Carlos Florensa, Ignasi Clavera, and Pieter Abbeel. Sub-policy adaptation for hierarchical reinforcement learning. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=ByeWogStDS.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pp. 1928–1937. PMLR, 2016a.

Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In Maria-Florina Balcan and Kilian Q. Weinberger (eds.), *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pp. 1928–1937. JMLR.org, 2016b. URL http://proceedings.mlr.press/v48/mniha16.html.

Ofir Nachum, Shixiang Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pp. 3307–3317, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/e6384711491713d29bc63fc5eeb5ba4f-Abstract.html.

Ronald Parr and Stuart Russell. Reinforcement learning with hierarchies of machines. *Advances in neural information processing systems*, pp. 1043–1049, 1998.

Chris Paxton, Vasumathi Raman, Gregory D. Hager, and Marin Kobilarov. Combining neural networks and tree search for task and motion planning in challenging environments. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2017, Vancouver, BC, Canada, September 24-28, 2017*, pp. 6059–6066. IEEE, 2017. doi: 10.1109/IROS.2017.8206505. URL https://doi.org/10.1109/IROS.2017.8206505.

Kasra Rezaee, Peyman Yadmellat, Masoud S. Nosrati, Elmira Amirloo Abolfathi, Mohammed Elmahgiubi, and Jun Luo. Multi-lane cruising using hierarchical planning and reinforcement learning. In *2019 IEEE Intelligent Transportation Systems Conference, ITSC 2019, Auckland, New Zealand, October 27-30, 2019*, pp. 1800–1806. IEEE, 2019. doi: 10.1109/ITSC.2019.8916928. URL https://doi.org/10.1109/ITSC.2019.8916928.

Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In Francis R. Bach and David M. Blei (eds.), *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pp. 1312–1320. JMLR.org, 2015. URL http://proceedings.mlr.press/v37/schaul15.html.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL http://arxiv.org/abs/1707.06347.

Satinder P. Singh, Tommi S. Jaakkola, and Michael I. Jordan. Reinforcement learning with soft state aggregation. In Gerald Tesauro, David S. Touretzky, and Todd K. Leen (eds.), *Advances in Neural Information Processing Systems 7, [NIPS Conference, Denver, Colorado, USA, 1994]*, pp. 361–368. MIT Press, 1994. URL http://papers.nips.cc/paper/981-reinforcement-learning-with-soft-state-aggregation.

Richard S. Sutton, David A. McAllester, Satinder P. Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In Sara A. Solla, Todd K. Leen, and Klaus-Robert Müller (eds.), *Advances in Neural Information Processing Systems 12, [NIPS Conference, Denver, Colorado, USA, November 29 - December 4, 1999]*, pp. 1057–1063. The MIT Press, 1999a.

Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999b.

Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In Dale Schuurmans and Michael P. Wellman (eds.), *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, pp. 2094–2100. AAAI Press, 2016. URL http://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/12389.

Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. Feudal networks for hierarchical reinforcement learning. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pp. 3540–3549. PMLR, 2017. URL http://proceedings.mlr.press/v70/vezhnevets17a.html.

Pin Wang and Ching-Yao Chan. Formulation of deep reinforcement learning architecture toward autonomous driving for on-ramp merge. In *20th IEEE International Conference on Intelligent Transportation Systems, ITSC 2017, Yokohama, Japan, October 16-19, 2017*, pp. 1–6. IEEE, 2017. doi: 10.1109/ITSC.2017.8317735. URL https://doi.org/10.1109/ITSC.2017.8317735.

Pin Wang, Ching-Yao Chan, and Arnaud de La Fortelle. A reinforcement learning based approach for automated lane change maneuvers. In *2018 IEEE Intelligent Vehicles Symposium, IV 2018, Changshu, Suzhou, China, June 26-30, 2018*, pp. 1379–1384. IEEE, 2018. doi: 10.1109/IVS.2018.8500556. URL https://doi.org/10.1109/IVS.2018.8500556.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 8:229–256, 1992. doi: 10.1007/BF00992696. URL https://doi.org/10.1007/BF00992696.

Dongbin Zhao, Zhongpu Xia, and Qichao Zhang. Model-free optimal control based intelligent cruise control with hardware-in-the-loop demonstration [research frontier]. *IEEE Comput. Intell. Mag.*, 12(2):56–69, 2017. doi: 10.1109/MCI.2017.2670463. URL https://doi.org/10.1109/MCI.2017.2670463.

A Appendix

A.1 Roads and lane approximation

Our highway driving environments have one of three types of roads: four straight lanes, two curved lanes, and two lanes merging into one, which are in Figure 4. To handle the information on these various shapes of lanes, the agent uses the sampled points from each lane. For each lane, positions of the left and right sides of the lane are sampled along the lane such that a concatenation of convex quadrilaterals can be created. For each quadrilateral, information on adjacency to another quadrilateral in bordering lanes is given as well.The quadrilaterals approximating a lane are used to define inter-vehicle regions (IVRs). When two positions at the rear end (q_r, v_r) and the front end (q_f, v_f) are given for a lane l, the distance between the two $D(l, (q_r, v_r), (q_f, v_f))$ is defined by the sum of distances, $(1 - v_r)d(Q(l, q_r)) + \sum_{i=q_r+1}^{q_f-1} d(Q(l, i)) + v_f d(Q(l, q_f))$, where $Q(l, q)$ is the quadrilateral search function and $d(Q)$ is the Euclidean distance between the center positions of the rear end $(0.5, 0)$ and the front end $(0.5, 1)$ of the given quadrilateral.

A.2 Normalized quadrilateral coordinate

We are given four positions of a quadrilateral in a lane with respect to global coordinate or egocentric coordinate, (x_{fl}, y_{fl}), (x_{fr}, y_{fr}), (x_{rl}, y_{rl}), (x_{rr}, y_{rr}), which are positions of front-left, front-right, rear-left, and rear-right. The goal of our transformation is two-fold: First, a position inside the quadrilateral (x, y) should be transformed to the normalized position (u, v) of the given quadrilateral so that (x_{fl}, y_{fl}), (x_{fr}, y_{fr}), (x_{rl}, y_{rl}), and (x_{rr}, y_{rr}) correspond to $(1, 0)$, $(1, 1)$, $(0, 0)$, and $(0, 1)$. Second, (x, y) should be the interpolation of positions on both left and right sides that divide the length of each side in the same proportion v along the progression. Thus, we designed the transformation so that (x, y) is represented as the interpolation of (x_{vl}, y_{vl}) and (x_{vr}, y_{vr}) with

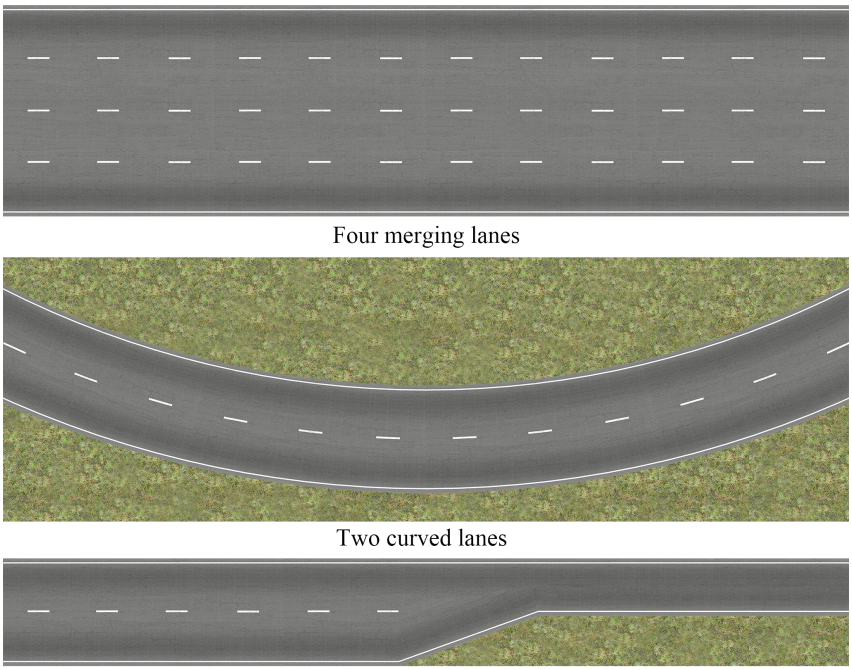

Figure 4: Three types of roads for our highway driving environment

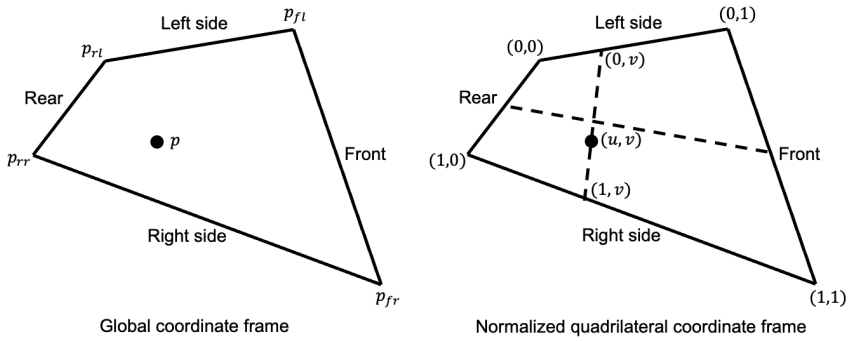

Figure 5: A position in a quadrilateral represented in normalized quadrilateral coordinate frame

ratio u, which are the interpolation of (x_{fl}, y_{fl}) and (x_{rl}, y_{rl}) with ratio v and the interpolation of (x_{fr}, y_{fr}) and (x_{rr}, y_{rr}) with ratio v, respectively. Figure 5 shows the positions on an arbitrary quadrilateral represented in both global coordinate frame and normalized quadrilateral coordinate frame. The transformation from normalized position (u, v) to global position (x, y) is easily done by the two steps of interpolation, while transformation from (x, y) to (u, v) is performed by solving the following equations. For x axis,

$$\begin{aligned} x_{vl} &= x_{rl} + v(x_{fl} - x_{rl}) \\ &= x_{rl} + vd_l, \end{aligned} \tag{9}$$

$$\begin{aligned} x_{vr} &= x_{rr} + v(x_{fr} - x_{rr}) \\ &= x_{rr} + vd_r, \end{aligned} \tag{10}$$

$$
\begin{aligned}
u &= \frac{(x - x_{vl})}{(x_{vr} - x_{vl})} \\
&= \frac{x - (x_{rl} + vd_l)}{(x_{rr} + vd_r) - (x_{rl} + vd_l)} \\
&= \frac{(x - x_{rl}) - vd_l}{v(d_r - d_l) + (x_{rr} - x_{rl})} \\
&= \frac{e - vd_l}{f + vg},
\end{aligned}
\tag{11}
$$

where $d_l = x_{fl} - x_{rl}$, $e = x - x_{rl}$, $f = x_{rr} - x_{rl}$, and $g = (x_{fr} - x_{rr}) - (x_{fl} - x_{rl})$.

In the same way for y axis,

$$
u = \frac{i - vh_l}{j + vk}, \tag{12}
$$

where $h_l = y_{fl} - y_{rl}$, $i = y - y_{rl}$, $j = y_{rr} - y_{rl}$, and $k = (y_{fr} - y_{rr}) - (y_{fl} - y_{rl})$.

Now a quadratic equation about v is formed by combining the equations about u on both axes.

$$
\frac{e - vd_l}{f + vg} = \frac{i - vh_l}{j + vk} \tag{13}
$$

The Equation 13 can be developed to

$$
(d_l k - h_l g)v^2 + (gi + d_l j - h_l f - ek)v + (ej - fi) = 0, \tag{14}
$$

which is a quadratic equation for v, and this can be substituted to

$$
av^2 + bv + c = 0, \tag{15}
$$

where $a = d_l k - h_l g$, $b = (gi + d_l j - h_l f - ek)$, and $c = ej - fi$.

Since (x, y) is inside the quadrilateral, u and v are between 0 and 1. Thus, the solution to Equation 15 satisfying the condition is $v = \frac{-b+\sqrt{b^2-4ac}}{2ac}$ if $a > 0$, $v = \frac{-b-\sqrt{b^2-4ac}}{2ac}$ if $a < 0$, otherwise, $v = -c/b$. Then, $u = \frac{e-vd_l}{f+vg}$ if $f + vg \neq 0$, otherwise, $\frac{i-vh_l}{j+vk}$.

A.3 HASHING QUADRILATERALS

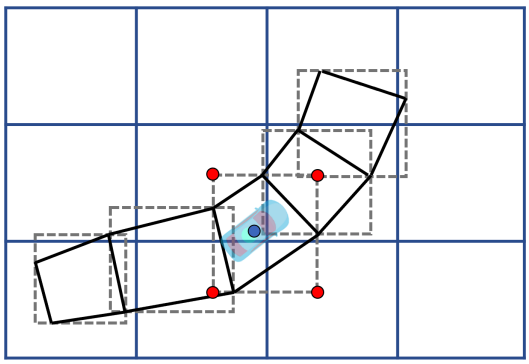

Figure 6: An exampling for understanding the hashing algorithm of the quadrilaterals of lanes

Lanes are static features with respect to a global reference point. Thus, if all lanes can be approximated before driving, or parts of lanes ahead of ego vehicle can be approximated before they come in the view range, the approximated parts can be used repeatedly during driving. In addition, if the quadrilaterals that approximate the lanes can be hashed, the time to search for the quadrilateral inside which the center or a corner position of a vehicle exists can be reduced from $\mathcal{O}(n)$ to $\mathcal{O}(1)$ where n is the number of quadrilaterals to be searched for. Thus, we devise a hashing algorithm that

Table 1: Candidate behaviors with their available IVRs to pay mind to and the outline IVR

Behavioral mode	Set of available IVRs to pay mind to	Outline IVR
stay in current IVR	{current IVR}, or {front IVR} or {rear IVR} if a vehicle invades the lane of ego	current IVR
maneuver to the other in lane (optionally selectable)	{front IVR} if ego is in rear IVR {rear IVR} if ego is in front IVR	IVR in mind
pay mind to the left	{IVRs on left lane in mind} if the lane exists, otherwise $\emptyset$	current IVR
maneuver to the left	{IVRs on left lane ahead in mind} if the lane exists, otherwise $\emptyset$	IVR in mind
pay mind to the right	{IVRs on right lane in mind} if the lane exists, otherwise $\emptyset$	current IVR
maneuver to the right	{IVRs on right lane ahead in mind} if the lane exists, otherwise $\emptyset$	IVR in mind

can quickly search for quadrilaterals to which a given position can belong. Figure 6 is an example for understanding the hashing algorithm. Our hash function can hash a position in two-dimension to a bin, which can be represented as a blue rectangle of the smallest possible unit size. To store the information of quadrilaterals in bins, we hash the vertices of the rectangular bounding box that covers a quadrilateral and put the quadrilateral in the bin of each vertex so that the quadrilateral is stored in one to four bins. The grey boxes in the dashed line and the red dots are examples of the bounding boxes for quadrilaterals and the vertices of a bounding box, respectively. Practically, the duplicate storing of a quadrilateral in one bin is avoided for fast search. The width and height of any bin are kept longer than those of the bounding box that covers any quadrilateral, ensuring that vertices of any bounding box are hashed to the same bin or bins that are adjacent to each other. This also guarantees that for a position p, the hashing function $b = H(p)$ can give the bin that contains the quadrilateral to which p belongs because p is inside the bounding box that aligns with the bins. The blue dot in Figure 6 is the center position of the blue vehicle, inside the bounding box with the red dots as its vertices.

A.4 INTER-VEHICLE REGIONS

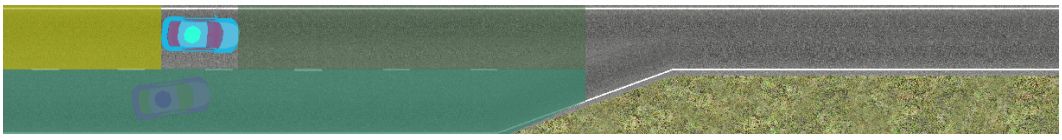

Figure 7: An example of inter-vehicle regions for the purple-colored vehicle

An inter-vehicle region (IVR) in a lane is defined as the region between the rear end NQC, $n_r = (q_r, v_r)$, and the front end NQC, $n_f = (q_f, v_f)$, bordered by surrounding vehicles or any end of the view range. Figure 7 shows an example of IVRs for the purple-colored vehicle. Given normalized positions of both ends, the featuring global positions, $(p_{0,l}, p_{0,r}), (p_{1,l}, p_{1,r}), \cdot, (p_{N-1,l}, p_{N-1,r}), (p_{N,l}, p_{N,r})$, are acquired by sampling N times with uniform distance interval on the left and right sides along the progressing direction, where $(p_{0,l}, p_{0,r})$ and $(p_{N,l}, p_{N,r})$ are defined by n_r and n_f, respectively. By using the distance function A.2, the distance between the rear and front $D_{total} = D(l, n_r, n_f)$ is acquired, and from the rear, we can get the next sample at the distance of $D_s = D_{total}/(N-1)$ in turn. Table 1 specifies candidate high-level commands of behaviors, the corresponding set of available IVRs to pay mind to, and the outline IVR.

Table 2: Reward types and their definition given in our environment

Reward Type	Definition
Goal Reward	1 if the ego center position gets into the long-term goal
Collision Penalty	-1 if the vehicle image overlays on the lane shoulders or other vehicles
Progression Reward	$c_{pro}v_{heading}$ where $0 < c_{pro} < 1$
Max Speed Penalty	$c_{max}min(v_{heading} - l_{max}, 0)$ for speed limit l_{max} where $-1 < c_{max} < -c_{pro}$
Min Speed Penalty	$c_{min}max(l_{min} - v_{heading}, 0)$ for speed limit l_{min} where $-1 < c_{min} < 0$

A.5 Other features, goal, and rewards

In our autonomous driving environment, on top of inter-vehicle regions, two additional types of state features exist: state of vehicles and range-sensing.

The feature of a vehicle j is

$$s_j = [p_{fl,j}, p_{fr,j}, p_{rl,j}, p_{rr,j}, v_j, \psi_j]^T, \tag{16}$$

where $p_{fl,j} = (x_{fl,j}, y_{fl,j})$, $p_{fr,j} = (x_{fr,j}, y_{fr,j})$, $p_{rl,j} = (x_{rl,j}, y_{rl,j})$, and $p_{rr,j} = (x_{rr,j}, y_{rr,j})$. are the positions of the front-left, front-right, rear-left, and rear-right corners, respectively, $v_j = v_{x,j}, v_{y,j}$ is the velocity, and psi_j is the orientation in radians. State of vehicles consists of features of ego vehicle and arbitrary number of surrounding vehicles on the road in the view range.

Range-sensing features are acquired by simulating range sensing of ego vehicle and processing the information about global or ego-centered positions. An odd number of rays are shot at uniform angular intervals, θ, forming bilateral symmetry where the center ray is directed to the front of the vehicle. Figure 8 shows an example of rays shot from each vehicle. Total features of range-sensing include the positions of N number of rays: the start, r_0 and the ends, $r_1, r_2, \cdot, r_{N-1}, r_N$. Rays end where they hit obstacles such as other vehicles or lane shoulders, or at the maximum distance, d_{max}. We used rays with $N = 25$, $N\theta = 120$, and $d_{max} = 1000$. Inter-vehicle regions can be

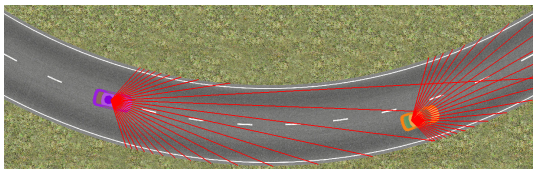

Figure 8: Range sensing of vehicles

applicable as the long-term goal. For experiments, we simply define a long-term goal region as the rectangular box (l, r, t, b) to deal with two types of goal regions, the region reached after the right end of a random lane and all lanes. Including the goal reaching reward, four types of rewards and their definitions are as in Table 2.