Robotics: Science and Systems 2023
Daegu, Republic of Korea, July 10-July 14, 2023

Efficient Reinforcement Learning for Autonomous Driving with Parameterized Skills and Priors

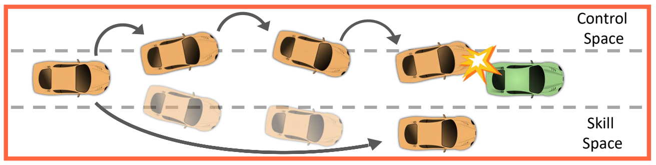
Letian Wang[1], Jie Liu[2], Hao Shao[3], Wenshuo Wang[4], Ruobing Chen[3], Yu Liu[2,3,†], Steven L. Waslander[1]
[1]University of Toronto, [2]Shanghai Artificial Intelligence Laboratory, [3]SenseTime Research, [4]McGill University

***Abstract*—When autonomous vehicles are deployed on public roads, they will encounter countless and diverse driving situations. Many manually designed driving policies are difficult to scale to the real world. Fortunately, reinforcement learning has shown great success in many tasks by automatic trial and error. However, when it comes to autonomous driving in interactive dense traffic, RL agents either fail to learn reasonable performance or necessitate a large amount of data. Our insight is that when humans learn to drive, they will 1) make decisions over the high-level skill space instead of the low-level control space and 2) leverage expert prior knowledge rather than learning from scratch. Inspired by this, we propose ASAP-RL, an efficient reinforcement learning algorithm for autonomous driving that simultaneously leverages motion skills and expert priors. We first parameterized motion skills, which are diverse enough to cover various complex driving scenarios and situations. A skill parameter inverse recovery method is proposed to convert expert demonstrations from control space to skill space. A simple but effective double initialization technique is proposed to leverage expert priors while bypassing the issue of expert suboptimality and early performance degradation. We validate our proposed method on interactive dense-traffic driving tasks given simple and sparse rewards. Experimental results show that our method can lead to higher learning efficiency and better driving performance relative to previous methods that exploit skills and priors differently. Code is open-sourced to facilitate further research.**

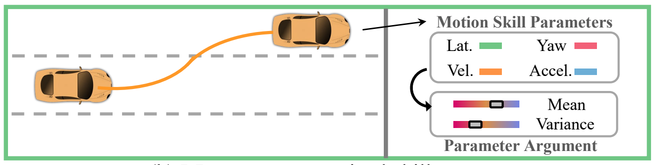
I. Introduction

Autonomous vehicles (AVs) on public roads will interact with other agents in various driving scenarios and situations characterized by traffic densities, road geometries, and traffic rules [47]. Many existing decision-making frameworks are based on elaborate hand-designed rules and decision hierarchies [6, 49]. Nonetheless, the joint consideration of multiple vehicles scales exponentially in dense traffic and is usually computational resource-hungry. Further, it can be challenging to design rules manually to cover all safety-critical cases, leading to severe generalizability issues. Fortunately, reinforcement learning (RL) requires little human labor in identifying policies across multiple different tasks by automatically interacting with the environment [40, 43, 21]. However, when it comes to an interactive multi-vehicle setting with a continuous action space, the learning efficiency of RL algorithms remains notoriously low because RL agents either fail to learn reasonable performance or necessitate a large amount of data and resources to make significant progress.

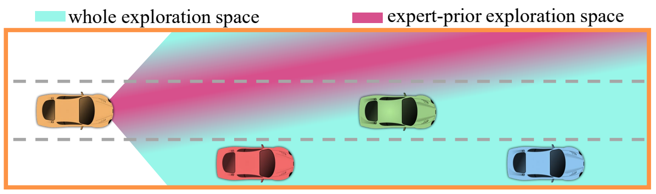
†Corresponding author. liuyuisanai@gmail.com
First author contact: lt.wang@mail.utoronto.ca



(a) RL in control space VS skill space



(b) RL over parameterized skill space



(c) RL exploiting expert priors

Fig. 1: (a) RL-based AVs learning over the control space will exhibit inconsistent action sequences. In comparison, RL over the skill space can generate a sequence of consistent low-level actions with more informative exploration and accelerated reward signaling. (b) The parameterized motion skill provides a key interface for RL agents to explore and learn. (c) The expert demonstration can provide prior knowledge of which regions of the action space are more promising in getting rewards than others, which can accelerate learning.

One great insight to improve learning efficiency is that, there can be different choices of the action space for the RL agent, and a proper choice or design of the action space can significantly simplify learning [23]. Most existing RL methods directly learn over vehicles' control space, such as the steering and the pedal commands of one time step [39, 3, 28]. However, a sequence of single-step control signals with inconsistent exploration rarely achieves typical driving maneuvers and lacks informative reward signals for the agent to learn. For example, as shown in the upper part of Fig. 1(a), the orange vehicle might move erratically when exploring the control space, thereby failing to achieve driving maneuvers such as overtaking a leading car. Such agents with frequent failures rarely receive informative reward signals for the agent to improve. In contrast, behavioral science has revealed that human behavior is naturally temporally hierarchical [12, 2, 7]: humans learn and make decisions over abstracted high-level options, which we call motion skills, while the low-level control commands are not an action space to learn but simply a sequence of muscle responses/executions to achieve high-level motion skills. As shown in the lower part of Fig. 1(a), a sequence of consistent low-level control commands is generated correspondingly after the driver decides to perform the motion skill of overtaking a lead vehicle. Such temporally-extended motion skills enable efficient learning through structured exploration and improved reward signaling.

However, proper design and learning of motion skills are non-trivial. There are two main ways to define and learn the motion skills in autonomous driving and robotics: (1) Manually designing delicate task-specific or object-centric motion skills [46, 5, 4, 48, 14, 31, 44], such as merging into a target lane before a car. However, such task-specific or object-centric motion skills are usually too complex to design manually for autonomous driving in multi-agent dense-traffic settings, nor can such delicately-designed motion skills cover the wide variety of driving and interaction situations that arise. (2) Extracting or learning motion skills from offline motion datasets, such as clustering or segmenting motion trajectories [15, 33, 30], or distilling offline motions into low-dimension latent space [29, 24, 25]. However, motion datasets are usually unbalanced in distribution and lack diversity, making it hard to learn all the needed skills. Inspired by motion planning in autonomous driving, we propose to exploit motion skills in the pure ego vehicle motion view, which is diverse and thus generalizable to complex driving tasks. Fig. 1(b) depicts that with such naturally-parameterized motion skills, AVs can directly learn over the parameters of motion skills with little design effort.

In addition to using motion skills, the other widely-recognized attempt to improve learning efficiency is leveraging prior knowledge from expert demonstrations. The critical insight is that the value of the whole exploration space is not uniform. Some regions of the action space are more promising in getting rewards than others. As shown in Fig. 1(c), it would be more rewarding for the vehicle to run in the left-front direction with sparse traffic than in the right-front direction with dense traffic. Many works integrate such expert prior knowledge by utilizing expert demonstrations to initialize the RL agent [20, 10, 19, 27, 32] and/or training an expert policy based on expert demonstrations to guide reinforcement learning as a regularizer or reward term [29, 34, 21]. However, simultaneously leveraging the expert's prior knowledge and the parameterized motion skills is non-trivial, since most expert demonstrations only include control information but miss skills or rewards annotation. To this end, we propose an inverse optimization method to recover the corresponding motion skill parameters given expert demonstrations using sequential quadratic programming (SQP) [16]. Thus, we can convert the expert demonstration from control space to skill space, allowing us to take advantage of both skills and priors simultaneously.

While actor pertaining [20, 10, 19, 27, 32] and expert regularization [29, 34, 21] have been widely used in previous methods to leverage expert priors, such methods could either suffer from performance drop at early training iterations due to actor/critic mismatch or performance suppression due to expert suboptimality. To this end, we propose a simple but effective double initialization method. We first pretrain the actor with the expert demonstration in skill space, then pretrain the critic by rolling out the pretrained actor to collect expert demonstrations with reward and skill information. We will show how this simple but effective double initialization method can bypass the issue of suboptimality and initial performance drop.

In summary, the contributions of the proposed ASAP-RL (RL with p**A**rameterized **S**kills **a**nd **P**riors) are threefold:

- Propose an RL method to learn over the parameter of motion skills for more informative exploration and improved reward signaling. Such skills are defined in the ego vehicle motion view, which are diverse and thus generalizable to different complex driving tasks.
- Propose an inverse skill parameter recovery method to convert expert demonstration from control space to skill space, and a simple but effective double initialization method to better leverage expert prior without issues of performance drop or suppression. Thus we can take advantage of both skills and priors simultaneously.
- Validate our method for autonomous driving tasks in three challenging dense-traffic scenarios and demonstrate our method outperforms previous methods that consider skills and priors differently.

II. Related Works

In this section, we will discuss related works on reinforcement learning with skills and priors, with a focus on autonomous driving and robotics. To the best of our knowledge, we are the very first to simultaneously leverage skills and expert priors to improve reinforcement learning for autonomous driving. Moreover, our proposed skill representation is motion-centric, not task-centric, implying that it can be flexibly extended to navigation for other moving robots, such as for mobile robots, UAVs, and the end-effector of manipulation robots, by accordingly modifying the dynamic constraint of the skill. Our proposed skill recovery and double-initialization method is flexible to leverage prior knowledge for general robot embodiment. We believe the proposed method could help to close the gap between skill and priors, and spark further research in this direction.

A. Reinforcement Learning with Skills

Toward defining or getting the skill that can be exploited in reinforcement learning for autonomous driving and robotics, two main approaches exist: (1) manually designing task-specific or object-centric motion skills [46, 5, 4, 48, 14, 31], such as cutting in a target lane before a specific car, or grasping a specific object. However, when AVs run on actual roads, a dense-traffic setting, AVs' motion usually should be synthesized considering relationships between multiple

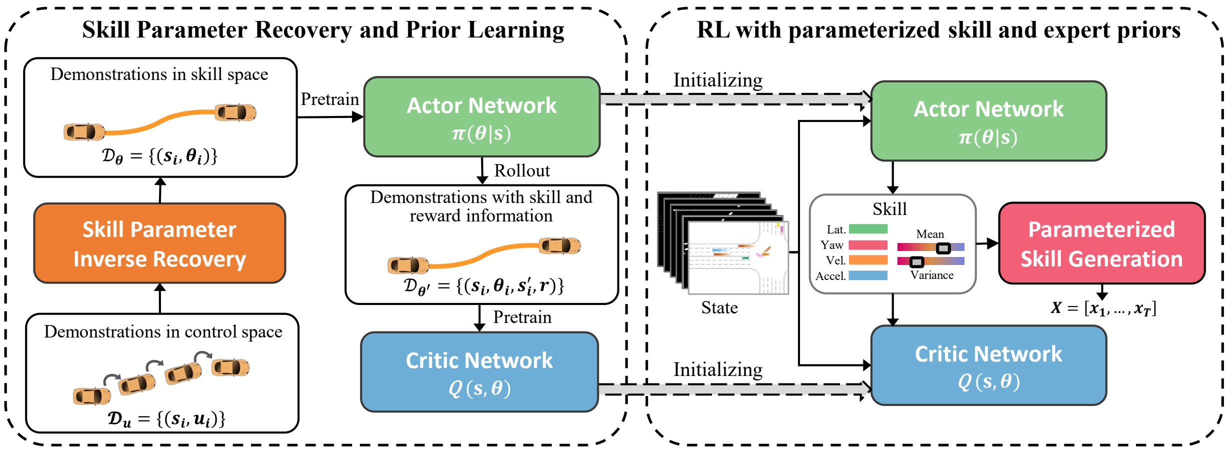

Skill Parameter Recovery and Prior Learning
Demonstrations in skill space
$\mathcal{D}_{\theta} = \{(s_i, \theta_i)\}$
Pretrain
Actor Network
$\pi(\theta|s)$
Rollout
Demonstrations with skill and reward information
$\mathcal{D}_{\theta'} = \{(s_i, \theta_i, s_i', r)\}$
Pretrain
Skill Parameter Inverse Recovery
Demonstrations in control space
$\mathcal{D}_u = \{(s_i, u_i)\}$
Critic Network
$Q(s, \theta)$
RL with parameterized skill and expert priors
Initializing
Actor Network
$\pi(\theta|s)$
Skill
Lat.
Yaw
Vel.
Accel.
Mean
Variance
State
Parameterized Skill Generation
$X = [x_1, \ldots, x_T]$
Initializing
Critic Network
$Q(s, \theta)$

Fig. 2: The pipeline of the proposed ASAP-RL method. An inverse skill parameter recovery method is proposed to convert expert demonstration from control space to skill space. A double initialization method is introduced to initialize both actor and critic to inject the expert's prior knowledge into RL. The RL agent can learn and explore in the skill space instead of the control space while leveraging the expert priors, which leads to high learning efficiency and improved final performance.

surrounding vehicles, which is usually too complicated to design manually. With limited flexibility and expressiveness, such manually-designed skills can hardly cover diverse driving and interaction situations. (2) another approach is to extract or learn skills from offline motion datasets, such as clustering or segmenting motion skills [15, 33, 30], or distill offline motions into low-dimension latent space [29, 24, 25], so that the issues and labor of manual design can be bypassed. However, such datasets can be expensive and labor-intensive to collect if they are not already available. It would also be difficult for such learned skills to transfer to new tasks or environments as they are usually task-specific and conditioned on the environment. Moreover, it is also not guaranteed that all necessary skills are covered in the datasets as they are usually unbalanced in distribution and sub-optimal. To tackle the limitations above, one recent work [50] combines the two approaches by first building a task-agnostic and ego-centric motion skill library in a pure ego vehicle motion perspective and then encoding the motion skills into a low-dimension latent skill space. However, the skill library construction still requires considerable effort, and the latent space encoding further makes the decision-making less interpretable. Thus, in this work, we propose to directly learn over the skill parameter space instead of the latent space, to spare the efforts in skill library construction, enable more interpretability in the decision-making process, and also provide an parameterized interface to further leverage priors.

B. Reinforcement Learning with Expert Priors

Although RL has been shown to be effective in several problems, learning efficiency issues [11] limit its applications. Inspired by human decision-making processes where prior knowledge is quite helpful when we learn new tasks, many works use expert prior knowledge to avoid learning from scratch with exhaustive interactions. There exist three approaches to leveraging prior knowledge: (1) utilize expert demonstration for a warm-up pretraining or policy (actor) initialization before RL [20, 10, 19, 27, 32]; (2) train an expert policy based on expert demonstrations, which is then used to guide RL process [29, 34, 21]; (3) maintain an expert data buffer, which is mixed with interaction buffer during RL for more fruitful experiences [20, 10, 27, 32, 42, 26, 22]. The first method is effective in policy-based RL methods but only initializing the actor was reported to be not helpful in the actor-critic framework [29]. This is because the actor and critic are interacting with each other during the learning process, and the actor's objective is to maximize the Q-value output by the critic without pretriang. Thus, the actor could quickly lose the prior knowledge learned from the expert after several updates. Empirically, we found performance drops can happen at early training iterations when adopting the first method (see Section IV-C2). The second and third methods use expert priors as guidance during RL training but might suppress the performance when the expert performance is suboptimal [34]. To overcome the limitations of the three methods, we propose a 'double initialization' technique with both the actor and critic initialized simultaneously and demonstrate this approach can achieve strong performance even when a suboptimal expert is employed.

III. Approach

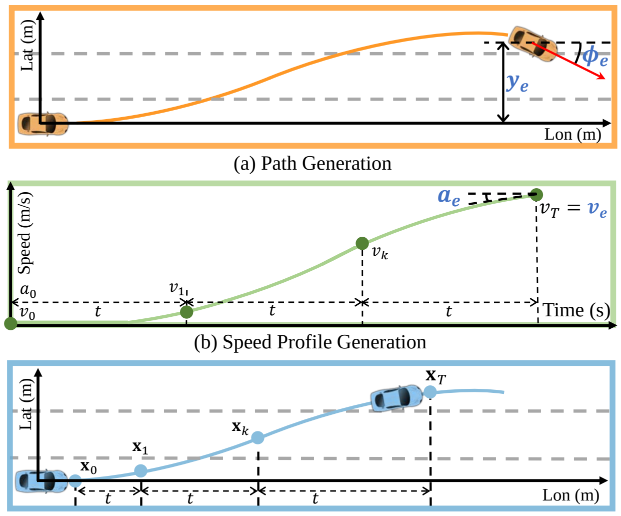
Let the Markov Decision Process be defined as $\{\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma\}$, a tuple of states, actions, transition probabilities, rewards, and discount factor. Our goal is to leverage parameterized motion skills and priors to accelerate reinforcement learning in continuous space for autonomous driving in dense traffic settings. Fig. 2 illustrates the architecture of our method, and Algorithm 1 outlines the pipeline. In this section, we first define the parameterized motion skill, which is defined in pure ego vehicle motion view and thus generalizable to different driving scenarios. Then the RL agent only needs to focus on the high-level decision-making task by learning over the skill parameter space, and the low-level motion control task is handled by generating the motion trajectory from the skill parameters. To further utilize priors in expert demonstrations $\mathcal{D}_u = \{(s_i, u_i)\}$, which only include state s_i and control action u_i but lack skill or reward information, we proposed an inverse optimization process to recover skill parameters $\boldsymbol{\theta}_i$, constructing the expert demonstrations in skill space $\mathcal{D}_{\boldsymbol{\theta}} = \{(s_i, \boldsymbol{\theta}_i)\}$. In a maximum entropy actor-critic framework, we propose a simple but effective 'double initialization' method to pretrain both the actor and critic, which can bypass the issue of suboptimality and early performance drop when leveraging experts prior. With these methods, we can leverage the parameterized skill and prior simultaneously.

A. *Motion Skill Generation*

Inspired by the sampling-based motion planning [18, 8, 45], we first introduce the parameterized motion skill. Defined in a pure ego-motion perspective, such motion skills are task-agnostic and ego-centric and can cover diverse motions needed in dense-traffic settings, which can be generalizable to different scenarios. Technically, the beginning boundary of the motion skill is determined by the vehicle's current state, and one motion skill further necessitates four parameters of the end boundary i.e., the lateral position y_e, heading angle ϕ_e, speed v_e, and acceleration a_e at the end of the motion skill. RL agents can learn and explore these parameters to generate diverse motion skills. Given the skill parameters, the generation of one motion skill consists of three steps, as shown in Fig. 3: (a) path generation, generating future path on the road, (b) speed profile generation, specifying the variation of speed in the skill time window, and (c) trajectory (motion skill) generation, projecting the integral of the speed profile onto the path to generate motion trajectory. All details are introduced below.

Path Generation. The path is generated by connecting the origin and an endpoint on the road by cubic splines, in the ego vehicle's coordinate system. The endpoint is characterized by three parameters: longitudinal position x_e, lateral position y_e, and heading angle ϕ_e. To ensure feasible speed-to-path projection, the path length should be longer than the integral of the speed profile. We thus fix the longitudinal position of the endpoint as the longest distance the ego vehicle can reach within the skill time window T. Therefore, the path generation consists of two free parameters the RL agent needs to learn: the lateral position y_e and heading angle ϕ_e of the endpoint, which can cover diverse lateral driving intentions and maneuvers, such as lane keeping, overtaking, and cutting-in.

Speed Profile Generation. The speed profile is represented by a cubic polynomial in the time horizon $[0, T]$, parameterized by the speed v and acceleration a at the beginning and end of the time horizon. When generating the speed profile, we specify the speed and acceleration at the beginning time by the vehicle's current state. Thus, the speed profile generation phase provides two free parameters that the RL agent needs to learn: the speed v_e and acceleration a_e at the end time, which can cover diverse temporal intentions, such as accelerating, decelerating, and emergent stop.



(c) Trajectory (Motion Skill) Generation

Fig. 3: An illustration of parameterized motion skill generation process. One motion skill is determined by four skill parameters (shown in blue color) that RL agents directly learn and explore. (a) The path is generated by connecting a start point and an endpoint (parameterized by the lateral position y_e and heading angle ϕ_e of the endpoint) by the cubic polynomial. (b) The speed profile is represented by a cubic polynomial within the time window T, which is parameterized by the speed v_0 and acceleration a_0 at the beginning time and v_e and a_e at the end time. (c) Motion skill generated by projecting the integral of the speed profile onto the path.

Parameterized Motion Skill Generation. Given the path and speed profile, the motion skill is generated by projecting the integral of the speed profile onto the path. Each motion skill is a sequence of vehicle states $\mathbf{X} = [\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_T}]$, with each state as a tuple $\mathbf{x_t} = \{x_t, y_t, \phi_t, v_t, a_t\}$.

Note that each skill generation is conditioned on the vehicle's current state, which is the final state of the last executed skill, which ensures smoothness between skill segments. The dynamic constraint (acceleration, curvature) is also enforced by restricting the planning parameters in reasonable ranges.

B. *Skill Parameter Recovery*

Another way to accelerate RL is by leveraging prior knowledge from expert demonstrations. However, most expert demonstrations $\mathcal{D}_u = \{(s_i, u_i)\}$ are in control space and do not contain the skill and reward information, making it not readily usable. We propose an inverse parameter recovery procedure to annotate expert data with skill parameters and construct the expert demonstration in skill space $\mathcal{D}_{\boldsymbol{\theta}} = \{(s_i, \boldsymbol{\theta}_i)\}$.

Consider the motion skill generation (see Section III-A) as a forward process, where one motion skill $\mathbf{X}$ is generated given skill parameters $\boldsymbol{\theta}$. Then the skill parameter recovery can be regarded as an inverse procedure, where $\boldsymbol{\theta}$ are determined given one demonstrated motion skill $\mathbf{X}_d$. Practically, the motion skills are retrieved by sequentially splitting the demonstration trajectory into segments, each with a length of T. Formally, this recovery process can be formulated as

$$\begin{aligned} \boldsymbol{\theta} &= \arg\min_{\boldsymbol{\theta}} ||\mathbf{X}_d - \mathbf{X}||_2 \\ \text{s.t.} \quad \mathbf{X} &= f_s(\boldsymbol{\theta}) \end{aligned} \tag{1}$$

where f_s denotes the motion skill generation process in Section III-A. Specifically, we use the sequential quadratic programming (SQP) [16] as the optimization method. For each motion skill, we run the optimization process multiple times with different skill parameter initializations to mitigate issues of local optima and achieve better recovery accuracy.

C. Expert Prior Learning - Actor and Critic Pretraining

With the expert demonstration in skill space $\mathcal{D}_{\boldsymbol{\theta}} = \{(s_i, \boldsymbol{\theta}_i)\}$, we can leverage the skill and expert prior simultaneously in RL. In this paper, we modify the maximum-entropy RL to do so. Specifically, we adopt the Soft Actor-Critic framework (SAC [9]), which consists of an actor $\pi(\boldsymbol{\theta}|s)$ for policy improvement and a critic $Q(s, \boldsymbol{\theta})$ for policy evaluation. This section will introduce how the two networks can be pretrained to capture expert priors.

Actor Pretraining To leverage the expert demonstration in skill space $\mathcal{D}_{\boldsymbol{\theta}}$ as priors, we first pretrain an actor $\pi(\boldsymbol{\theta}|s)$ to capture the skill priors conditional on the current state. The training of this model aims at maximizing the log probability of the recovered expert skill parameter in the distribution output by the actor:

$$\mathbb{E}_{(s,\boldsymbol{\theta})\sim\mathcal{D}_{\boldsymbol{\theta}}}\Big[\log\pi(\boldsymbol{\theta}|s) + \beta\mathcal{H}(\boldsymbol{\theta})\Big] \tag{2}$$

where for each data $(s, \boldsymbol{\theta})$ in the expert demonstration $\mathcal{D}_{\boldsymbol{\theta}}$, the pretrained actor $\pi(\boldsymbol{\theta}|s)$ takes the current state s as inputs and outputs the Gaussian distribution of the skill parameters $\boldsymbol{\theta}$. $\mathcal{H}(\boldsymbol{\theta})$ denotes the entropy regularization term and β denotes entropy weight. The pretrained actor can provide prior knowledge on which skills are more promising to explore conditioning in the current situation.

Critic Pretraining Since the actor and critic interact with each other during RL training, only pretraining the actor does not fully leverage the prior knowledge. However, pretraining the critic $Q(s, \boldsymbol{\theta})$ is not always available since there is no reward information in the expert demonstration in either control space $\mathcal{D}_u$ or skill space $\mathcal{D}_{\boldsymbol{\theta}}$. Fortunately, we already have the pretrained actor who has learned expert priors. Thus we propose to roll out the pretrained actor in the environment to collect an expert demonstration with both skill and reward information $\mathcal{D}'_{\boldsymbol{\theta}} = \{(s_i, \boldsymbol{\theta}_i, s'_i, r)\}$, which is then used to pretrain the critic $Q(s, \boldsymbol{\theta})$. The pretaining of the critic follows typical policy evaluation in SAC, as in Line 38 of Algorithm 1. As in Sec IV-C2, we will discuss in detail how the simple but effective double initialization method can outperform other methods to incorporate expert priors.

Algorithm 1 ASAP-RL

```
1: Input: Raw demonstrations $\mathcal{D}_u$, discount $\gamma$, target entropy $\delta$, learning rates $\lambda_\pi, \lambda_Q, \lambda_\alpha$, target update rate $m$, temperature hyperparameter $\alpha$, motion skill model $f_s$
2: Require: actor $\pi_\varphi(\boldsymbol{\theta}_t|s_t)$, critic $Q_\phi(s_t, \boldsymbol{\theta}_t)$, target network $Q_{\bar{\phi}}(s_t, \boldsymbol{\theta}_t)$, replay buffer $\mathcal{D}$, demonstration in skill space $\mathcal{D}_{\boldsymbol{\theta}}$, demonstration with skill and reward information $\mathcal{D}'_{\boldsymbol{\theta}}$.
3: Skill Parameter Recovery
4: for each trajectory in $\mathcal{D}_u$ do
5:     for every $\mathbf{X}_d$ split from the trajectory do
6:         $\boldsymbol{\theta} = \arg\min ||\mathbf{X}_d - f_s(\boldsymbol{\theta})||_p$ (Eq 1)
7:         $\mathcal{D}_{\boldsymbol{\theta}} \leftarrow \mathcal{D}_{\boldsymbol{\theta}} \cup \{(s, \boldsymbol{\theta})\}$
8:     end for
9: end for
10: Prior Learning
11: for each iteration do                                  % Actor Pretraining
12:     Sample $(s, \boldsymbol{\theta})$ from $\mathcal{D}_{\boldsymbol{\theta}}$
13:     Update $\pi_\varphi$ according to Eq 2
14: end for
15: for each iteration do              % Roll-out $\pi_\varphi$ to Collect $\mathcal{D}'_{\boldsymbol{\theta}}$
16:     for every $T$ environment step do
17:         rollout pretrained actor to collect $\{s_t, \boldsymbol{\theta}_t, \tilde{r}, s_{t'}\}$
18:         $\mathcal{D}'_{\boldsymbol{\theta}} \leftarrow \mathcal{D}'_{\boldsymbol{\theta}} \cup \{s_t, \boldsymbol{\theta}_t, r, s_{t'}\}$
19:     end for
20: end for
21: for each iteration do                                  % Critic Pretraining
22:     Sample $(s, \boldsymbol{\theta}, r, s')$ from $\mathcal{D}'_{\boldsymbol{\theta}}$
23:     Update $\pi_\varphi$ according to Line 38
24: end for
25: RL with Parameterized Skills and Priors
26: Initialize actor and critic with the pretrained weight
27: for each iteration do
28:     for every $T$ environment step do
29:         $\boldsymbol{\theta}_t \sim \pi_\varphi(\boldsymbol{\theta}_t|s_t)$                  % sample skill parameter
30:         $\mathbf{X}_t = \{\mathbf{x_i}\}_{i=1}^T \sim f_s(\mathbf{X}_t|\boldsymbol{\theta}_t)$          % generate skill
31:         $s_{t'} \sim p(s_{t+T}, r_{t:t+T}|s_t, \mathbf{X}_t)$          % execute skill
32:         $\tilde{r}_t(s_t, \boldsymbol{\theta}_t) = \sum_{i=0}^{T-1} r_{t+i}$        % reward calculation
33:         $\mathcal{D} \leftarrow \mathcal{D} \cup \{s_t, \boldsymbol{\theta}_t, \tilde{r}, s_{t'}\}$              % replay buffer
34:     end for
35:     for every gradient step do                    % typical SAC training
36:         $\bar{Q} = \tilde{r}(s_t, \boldsymbol{\theta}_t) + \gamma[Q_{\bar{\phi}}(s_{t'}, \pi_\varphi(\boldsymbol{\theta}_{t'}|s_{t'})) - \alpha\mathcal{H}(\pi_\varphi(\boldsymbol{\theta}_{t'}|s_{t'}))]$
37:         $\varphi \leftarrow \varphi + \lambda_\pi\nabla_\varphi[Q_\varphi(s_t, \pi_\varphi(\boldsymbol{\theta}_t|s_t)) + \alpha\mathcal{H}(\pi_\varphi(\boldsymbol{\theta}_t|s_t))]$
38:         $\phi \leftarrow \phi - \lambda_Q\nabla_\phi\big[\frac{1}{2}\big(Q_\phi(s_t, \boldsymbol{\theta}_t) - \bar{Q}\big)^2\big]$
39:         $\alpha \leftarrow \alpha - \lambda_\alpha\nabla_\alpha\big[\alpha \cdot \big((\mathcal{H}\big(p_{\boldsymbol{\theta}}(\boldsymbol{\theta}_t|s_t) - \delta)\big]$
40:         $\bar{\phi} \leftarrow m\phi + (1-m)\bar{\phi}$
41:     end for
42: end for
```

D. RL over parameterized skill with priors

To simultaneously leverage parameterized motion skill and expert priors for higher learning efficiency, we modify the maximum-entropy RL, whose objective function is to encourage reward maximization and exploration in skill space:

$$J = \mathbb{E}_\pi \left[\sum_{t=1}^{T} \gamma^t r_t + \alpha \mathcal{H}\big(\pi(\boldsymbol{\theta}|s)\big) \right] \tag{3}$$

where $\sum_{t=1}^{T} \gamma^t r_t$ denotes the accumulated discounted reward return from the environment after one motion skill of length T is executed, $\mathcal{H}\big(\pi(\boldsymbol{\theta}|s)\big)$ denotes the entropy term, and α denotes the temperature parameter. Instead of learning a policy over raw control actions $\pi(u|s)$ at a single time step, we learn a policy that outputs skill parameters, $\pi(\boldsymbol{\theta}|s)$, which is then used to generate a motion skill by the procedure defined in Section III-A. Each motion skill is tracked for T time steps before the next skill is generated, which follows a typical semi-MDP process [41, 1] with temporal abstraction and accelerated reward signaling [24, 25, 21]. The time horizon, T, is fixed and consistent with the motion skill length in Section III-A and Section III-B. While some works investigated variable-length policy conditional on the task and environment [30, 13, 37], we empirically found a fixed-length policy can achieve strong performance and leave the extension to dynamic-length policies to future work. On top of parameterized motion skills, we adopt the double initialization method to further leverage priors, where we initialize the actor and the critic with the pretrained weight in Section III-B.

IV. Experiments

In this section, we will investigate the following sub-problems to evaluate our proposed ASAP-RL method:

- **Performance**: Can our method learn driving strategies with higher learning efficiency and performance than other methods considering skills and priors differently? (Fig. 5)
- **Influence of the length of skill**: The length of motion skill T is an important hyper-parameter. How does it influence the performance of our ASAP-RL? (Fig. 6)
- **Influence of expert prior**: Does the expert prior knowledge accelerate RL? How does the proposed double initialization perform with respect to other methods to incorporate expert prior? (Fig. 7)

A. Experiment Setup

1) Environment: We evaluate ASAP-RL on the MetaDrive simulator [17] under different dense-traffic scenarios (highway, roundabout, and intersection) to verify its performance for autonomous driving. At each run, the driving environment is generated with a random lane number and a random order of roadblocks. Traffic vehicles are spawned at a random location with a random target speed, and these vehicles are controlled by the default rule-based planner in MetaDrive. The ego vehicle needs to navigate in the traffic and arrive at the destination within the required time, without collisions or running off the roadway. As shown in Fig. 4, we follow the default setting in MetaDrive to use a 5-channel birds-eye-view (BEV) image with a size of $200 \times 200 \times 5$ as the input for the RL agent, which includes spatiotemporal information of the ego agent and surrounding agents, and the information of the

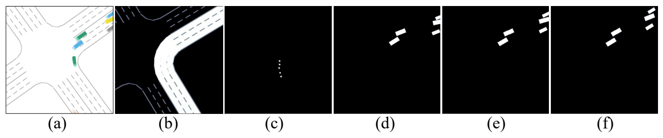


Fig. 4: The birds-eye view (BEV) images used as the observation and policy input. (a) the current scene; (b) road information (dashed line) and navigation lanes (in white color); (c) historical waypoints of the ego vehicle; (d-f) surrounding objects (white rectangles) at time t, $t-1$, and $t-2$.

road geometry and navigation. In practice, such observations are usually available in modern autonomous driving perception systems[38]. The focus of this paper is how to make decisions efficiently and safely in diverse dense-traffic scenarios with access to these observations. All experiments are run with three different seeds.

2) Reward Definition: s We adopt the sparse reward setting to enable minimum effort in reward engineering:

$$r_t = R_{progress} + R_{destination} + R_{crash} + R_{overtaking}. \tag{4}$$

- $R_{progress}$: The agent gets a sparse reward of 1 for every 10 m distance completed.
- $R_{destination}$: The agent gets a reward of 1 if it reaches the destination.
- R_{crash}: If the agent collides with other vehicles or the road curbs, it gets a negative reward of -5.
- $R_{overtaking}$: If the agent passes one vehicle, it gets a reward of 0.1.

In the ASAP-RL pipeline, a motion skill step contains T simulation steps, where each step lasts for 0.1 seconds. When the simulator performs one motion skill, it performs T simulation steps and gets rewards T times. Therefore, the skill reward will be the sum of T-step rewards, as shown in Line 32 of Algorithm 1. We empirically found our method performed well even without $R_{destination}$ in the highway and roundabout environments.

3) Expert Demonstration Collection: We have two options to collect expert demonstrations, one hand-designed rule-based expert planner, and one trained RL expert agent. The rule-based planner achieved higher performance than the RL expert agent after time-consuming manual tuning. However, the rule-based planner option required more designs in data collection (e.g. perturbance to the expert actions) to collect more diverse and react-to-danger actions, without which the agent trained from the demonstration will fail due to error accumulation [35]. In comparison, we found the trained RL expert agent can collect high-quality and more diverse demonstrations, and agents trained from such demonstrations achieved stronger performance. Thus we opt for the RL expert agent for data collection. In the future, the expert demonstrations can also be retrieved by human driving data, if available.

4) Baselines: We compare the performance of our ASAP-RL with the following baselines:

- **PPO [36]**: Train an agent from scratch by Proximal Policy Optimization, a typical single-step on-policy algorithm.

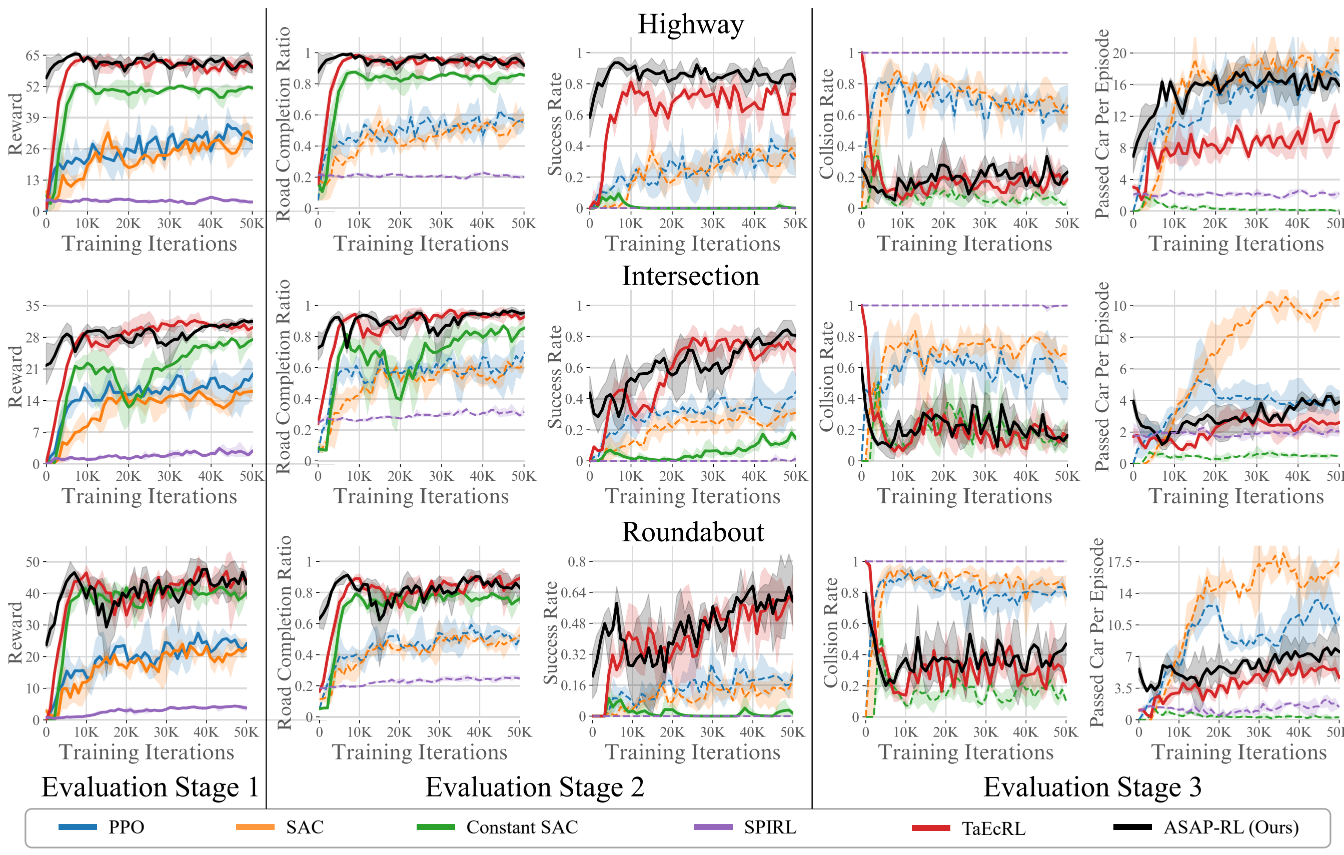


Fig. 5: Comparison of our method with baselines on the highway, intersection, and roundabout scenarios. PPO and SAC are classical RL algorithms over control space. Constant SAC repeats the same action for the skill horizon T. SPiRL and TaEcRL learn in low-dimension latent skill space and SPiRL also leverages expert priors. As detailed in Sec. IV-A6, the performance evaluation follows three stages to gradually distinguish the differences between methods with increasingly more specific metrics: 1) reward; 2) success rate, and road completion ratio; 3) collision rate, and passed car per episode. We only need to inspect metrics in later stages when the methods perform similarly in previous stages. The methods that are outperformed by other methods in previous evaluation stages are marked as dashed lines in later stages. Our ASAP-RL outperforms all other methods, and the margin between ASAP-RL and other methods increases as we move from stage 1 to stage 3.

- **SAC [9]**: Train an agent from scratch with Soft Actor-Critic, a typical single-step off-policy algorithm.
- **Constant SAC**: Train an agent from scratch with SAC, whose action is repeated T times (the same as the skill horizon of ASAP-RL). This method verifies the performance of simply temporally extending actions from one-step to multi-step without skill design.
- **SPiRL [29]**: Train an agent with the Skill-Prior RL algorithm, which leverages both skills and priors in the expert demonstration. SPiRL first embeds skills from the demonstration into a low-dimension latent skill space, where the RL agent can learn and explore efficiently. SPiRL then trains a skill prior network using the expert demonstrations, which is utilized as a KL divergence term to guide the RL agent and prevent deviation from the expert policy. Since the open-source code of SPiRL does not include the MetaDrive environment, we reproduced SPiRL in our codebase.
- **TaEcRL [50]**: Train an agent with task-agnostic and ego-centric motion skills. The difference between TaEcRL and SPiRL lies in a) TaEcRL learns latent skill space through the task-agnostic and ego-centric motion skill library proposed by TaEcRL, while SPiRL directly learns latent space from collected expert data with limited skill diversity; b) SPiRL leverages expert priors but TaEcRL does not.

5) Evaluation metric: To compare the performance of ASAP-RL with other methods, we adopt the following metrics, which reflect the performance of autonomous vehicles from different aspects:

- Episode Reward: the sum of all the rewards in an episode.
- Success Rate: the percentage of episodes where the agent reaches the destination on time without collisions.
- Road Completion Ratio: the ratio of road length completed by the agent to the total road length per episode.
- Collision Rate: the percentage of episodes in which a collision occurs.
- Passed Cars Per Episode: the number of cars overtaken by the agent in each episode.

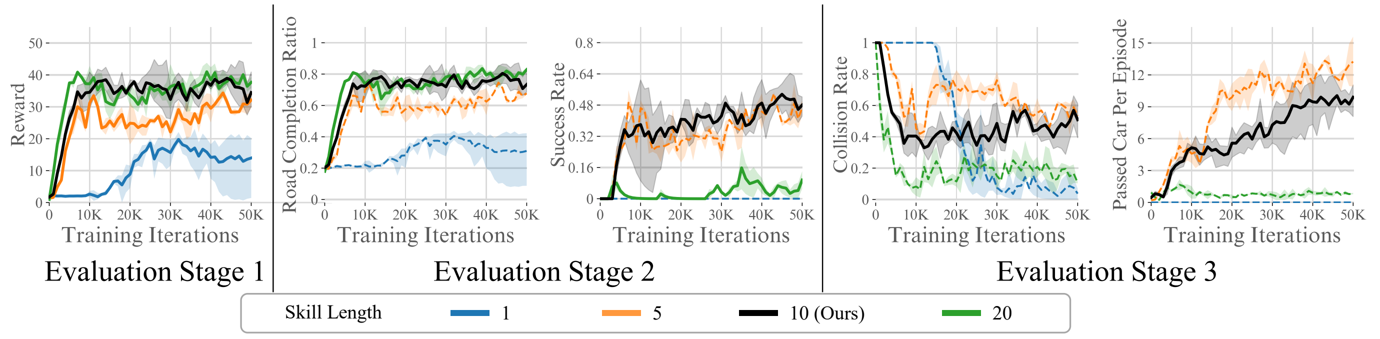


Fig. 6: Ablation analysis of skill length in the roundabout scenario. When T increases from 1 to 10, performance improvement is observed, benefiting from temporal abstraction. But when T reaches 20, it is too long for the agent to react to accidents during the skill execution due to delayed replanning. We observe that a skill length of 10 reached a good trade-off.

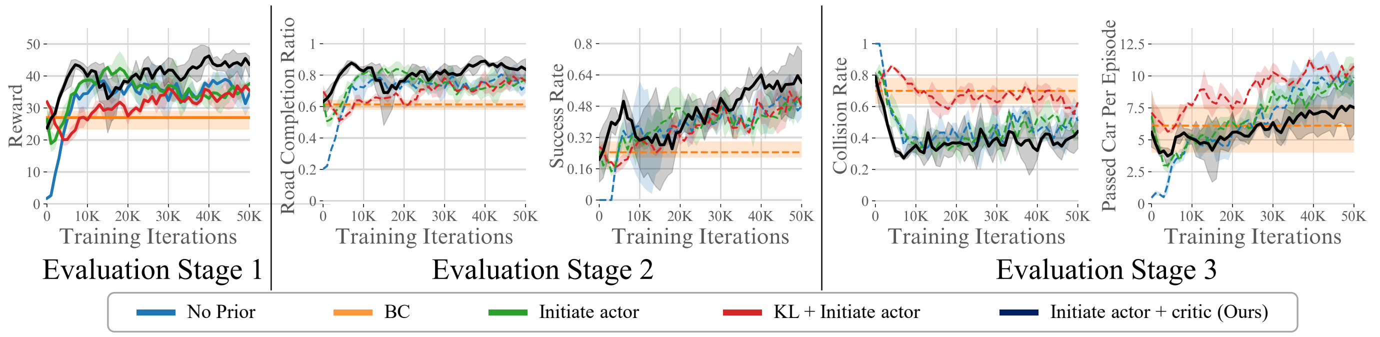


Fig. 7: Ablation analysis of different ways to incorporate expert priors in the roundabout scenario. Our method has a good starting performance and reached the highest final performance without a performance drop in early training iterations ('Initiate actor') or performance suppression due to the expert suboptimality ('KL + Initiate actor').

6) A three-stage strategy for performance evaluation: During the evaluation, the metrics above should be combined and sequentially inspected, since they evaluate the driving performance from different aspects and granularity. For example, a high passed-car-per-episode represents a good agent only when the agent also remains a decent success-rate and road-completion-ratio, and a reasonably low collision-rate. Thus we propose a three-stage strategy to evaluate the methods' performance, where we only need to inspect metrics in later stages when the methods perform similarly in metrics of previous stages:

- **Stage one** Since the reward is the direct optimization objective in RL, it can straight-forwardly reflect the performance of an algorithm. Therefore, in the first stage, we treat reward as the most important evaluation metric.
- **Stage two** Sometimes it can be difficult to distinguish methods on the reward metric alone. In this case, we move to the second stage focusing on other metrics with more driving contexts, namely the success rate, and the road completion ratio.
- **Stage three** If the performance on previous metrics is still close, we move to the third stage and focus on the collision rate and passed car per episode. The two metrics are somewhat trade-offs between each other. The passed car per episode has the smallest reward weight and can better distinguish the methods' performance.

B. Performance Comparison

We follow the proposed three-stage strategy to evaluate the performance of baselines and our method. As illustrated in Fig. 5, the reward metric in the first evaluation stage indicates that the performance of PPO, SAC, and SPiRL is significantly lower than that of the other three methods. Zhou et al. [50] reported that PPO and SAC had poor performance in MetaDrive tasks even under dense reward settings, so it is not surprising that the performance is poor under the more difficult sparse reward conditions used in this work. Though SPiRL was shown to be effective in manipulation tasks [4, 29], it performs poorly in our driving setting, likely due to the fact that the driving task requires very diverse skills while SPiRL can only learn skills from limited expert demonstrations: since the expert demonstrations of driving tasks are usually unbalanced to mostly consist of data where the vehicle is driving forward, it is hard to ensure that all essential skills are learned or covered in SPiRL agent. We empirically found that the agents learned with SPiRL mostly drive straight forward and can barely exhibit maneuvers, leading to a high collision rate.

The constant-SAC, which simply repeats the same action in the skill window T, obtains a slightly lower reward in the three scenarios compared to TaEcRL and ASAP-RL. However, when we move to the second/third evaluation stage and focus on the detailed driving-context metrics, the Constant-SAC performed much worse than TaEcRL and ASAP-RL: the constant-SAC agent drives very conservatively and slowly to avoid collisions. Though a low collision rate and a decent road completion ratio are achieved, the agent usually cannot arrive at the destination within the given time, which leads to a close-to-zero success rate and passed cars per episode. Thus it is not a good driving strategy and shows the importance of proper skill design.

Finally, we compare the results between TaEcRL and ASAP-RL. ASAP-RL achieves better performance in the first 10k iterations in terms of all metrics due to the usage of the expert prior. In addition, ASAP-RL has more passed cars per episode than TaEcRL by a large margin, with better or similar success rates, road completion ratio, and collision rates. This is hypothetically because 1) our ASAP-RL can generate very diverse skills by combining different skill parameters, while TaEcRL relies on offline datasets and can only use limited skills. For example, we found agents learned by TaEcRL usually drive in an erratic way, leading to a low ability to overtake. 2) ASAP-RL can leverage expert priors to accelerate convergence and bias the agent toward better optima and higher performance, due to our proposed skill parameter recovery and double-initialization method. In contrast, TaEcRL is limited by its own algorithm design and cannot utilize expert data. These results suggest ASAP-RL could be a useful tool to enable autonomous driving in interactive, diverse dense-traffic scenarios.

C. Ablation study

1) Influence of the length of skill: We ablate the skill horizon parameter, T, on the roundabout scenario to examine its influence. The results are shown in Fig. 6. When $T = 1$, ASAP-RL degenerates into a single-step method with poor performance. When $T = 5$ or $T = 10$, we observed a performance improvement benefiting from making long-term decisions. However, if T is too large ($T = 20$), during the skill execution, an overly-long skill length can make the agent less reactive to accidents and emergencies due to delayed replanning. Moreover, the richness of the skill set will be insufficient, and thus some driving strategies cannot be covered. Overall, a skill length of $T = 10$ reached a good trade-off.

2) Influence of expert prior: We conduct ablation studies on the roundabout scenario to investigate the effect of the expert prior and compare the proposed double initialization method with other methods to incorporate the expert prior. As shown in Fig. 7, we can distinguish their differences simply using the reward metric in the first evaluation stage.

No Prior. No Prior has to learn from scratch with low performance at the beginning with close-to-zero rewards, high collision rates, and data collection costs. Though its performance then goes up quickly, it fails to improve further. For example, in the reward metric, No Prior is constantly lower than our ASAP-RL at any training iteration, and the peak reward of our ASAP-RL is 20% higher than that of No Prior. These results demonstrate the effect of priors to bias agents toward better optima and higher performance.

Behavior Cloning (BC). The BC method trains a policy using expert demonstration without further reinforcement learning. Technically, the training is the same as the actor pretraining in our method as in Eq 2. Compared to 'No Prior', the BC method achieves better performance at the beginning but does not improve further, so it has a lower final performance.

Initiate Actor. The 'Initiate Actor' method uses the pre-trained policy weight to initialize the actor in SAC, and then continue with reinforcement learning. This method has a good initial performance and continues to improve. However, there is a performance drop in the first 5K iterations, likely due to the mismatch between the actor and the critic: though the actor is pretrained to have expert prior knowledge, the critic has no knowledge of the prior. Because the actor needs to interact with the critic during RL with the objective to maximize the Q-value output by the critic, as in Line 37 of Algorithm 1, the actor could quickly lose the prior knowledge learned from the expert after several updates.

KL + Initiate Actor. In addition to actor initialization, the 'KL + Initiate Actor' method also measures the KL divergence between the pre-trained actor and the RL actor during training. The KL term was added into the objective function to replace the original entropy term in SAC as in [29]. This is an ablation setting that incorporates expert prior during the whole RL training to guide the RL and prevent deviating from the expert policy. However, similar to the results reported in [34], the performance of the method is suppressed due to expert suboptimality: though this method has a good starting point, the reward is growing much more slowly than other methods in later training.

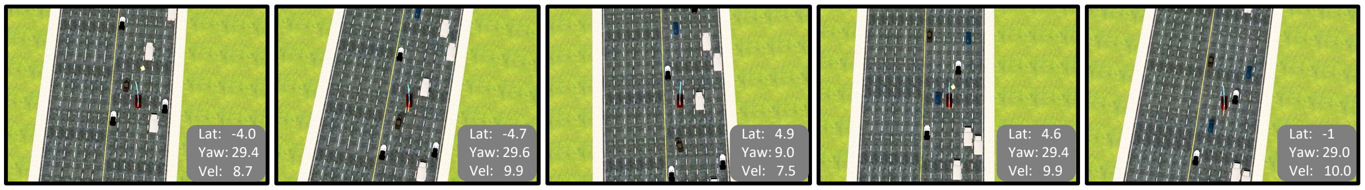
Initiate Actor and Critic. This is our proposed method to incorporate expert priors. Our 'double initialization' method has a good starting reward and the highest final performance, without the issue of performance drop caused to a mismatch between actor and critic, or performance suppression due to the expert suboptimality.

D. Visualizations

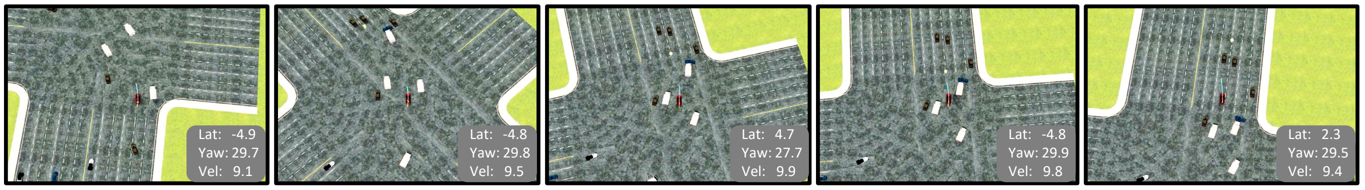
Visualizations of our ASAP-RL agent running on the three scenarios can be found in Fig. 8. The agent drives in dense traffic efficiently and safely with diverse maneuvers such as lane changing, cutting in, braking, etc.

V. Conclusion

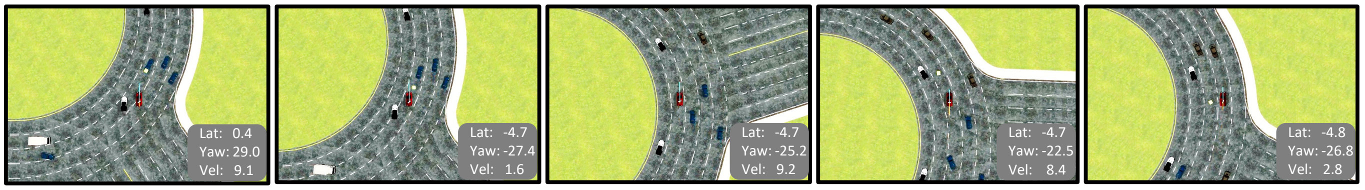
We present an efficient reinforcement learning (ASAP-RL) that simultaneously leverages parameterized motion skills and expert priors for autonomous vehicles to navigate in complex dense traffic. We first introduce parameterized motion skills and enable RL agents to learn over the skill parameter space instead of the control space. To further leverage expert priors on top of skills, we propose an inverse skill parameter recovery technique to convert expert demonstrations from control space to skill space. A simple but effective double initialization technique is also introduced to better leverage expert priors.



(a) Highway



(b) Intersection



(c) Roundabout

Fig. 8: Visualizations of the trained agents in three dense-traffic scenarios. The dark red vehicle denotes the ego agent and the blue line denotes the motion skill trajectory output by the agent. Generated parameters corresponding to the skill are shown in the bottom right. (a) In the highway scenario, the agent performs consecutive lane changes to overtake vehicles ahead. (b) In the intersection scenario, the agent turns left and overtakes vehicles ahead with a high velocity. (c) In the roundabout scenario, the agent slows down to cautiously overtake the vehicle on the left (the second image), and drives slowly when vehicles ahead block the way (the last image).

Validations on three challenging dense-traffic driving scenarios demonstrate that our ASAP-RL significantly outperforms previous methods in terms of learning efficiency and performance.

References

[1] Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

[2] Matthew M Botvinick, Yael Niv, and Andew G Barto. Hierarchically organized behavior and its neural foundations: a reinforcement learning perspective. *Cognition*, 113(3):262–280, 2009.

[3] Jianyu Chen, Shengbo Eben Li, and Masayoshi Tomizuka. Interpretable end-to-end urban autonomous driving with latent deep reinforcement learning. *IEEE Transactions on Intelligent Transportation Systems*, 23 (6):5068–5078, 2021.

[4] Murtaza Dalal, Deepak Pathak, and Russ R Salakhutdinov. Accelerating robotic reinforcement learning via parameterized action primitives. *Advances in Neural Information Processing Systems*, 34, 2021.

[5] Nachiket Deo, Akshay Rangesh, and Mohan M Trivedi. How would surround vehicles move? a unified framework for maneuver classification and motion prediction. *IEEE Transactions on Intelligent Vehicles*, 3(2):129–140, 2018.

[6] Wenchao Ding, Lu Zhang, Jing Chen, and Shaojie Shen. Epsilon: An efficient planning system for automated vehicles in highly interactive environments. *IEEE Transactions on Robotics*, 38(2):1118–1138, 2021.

[7] Yannis Flet-Berliac. The promise of hierarchical reinforcement learning. *The Gradient*, 2019.

[8] Tianyu Gu. *Improved trajectory planning for on-road self-driving vehicles via combined graph search, optimization & topology analysis*. PhD thesis, Carnegie Mellon University, 2017.

[9] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.

[10] Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Ian Osband, et al. Deep q-learning from demonstrations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[11] Chi Jin, Qinghua Liu, and Sobhan Miryoosefi. Bellman eluder dimension: New rich classes of rl problems, and sample-efficient algorithms. *Advances in neural information processing systems*, 34:13406–13418, 2021.

[12] Jian Jing, Elizabeth C Cropper, Itay Hurwitz, and Klaudiusz R Weiss. The construction of movement with behavior-specific and behavior-independent modules. *Journal of Neuroscience*, 24(28):6315–6325, 2004.

[13] Thomas Kipf, Yujia Li, Hanjun Dai, Vinicius Zambaldi, Edward Grefenstette, Pushmeet Kohli, and Peter Battaglia. Compositional imitation learning: Explaining and executing one task at a time. *arXiv preprint arXiv:1812.01483*, 2018.

[14] Jens Kober and Jan Peters. Learning motor primitives for robotics. In *2009 IEEE International Conference on Robotics and Automation*, pages 2112–2118. IEEE, 2009.

[15] George Konidaris, Scott Kuindersma, Roderic Grupen, and Andrew Barto. Robot learning from demonstration by constructing skill trees. *The International Journal of Robotics Research*, 31(3):360–375, 2012.

[16] Dieter Kraft. A software package for sequential quadratic programming. *Forschungsbericht- Deutsche Forschungs- und Versuchsanstalt fur Luft- und Raumfahrt*, 1988.

[17] Quanyi Li, Zhenghao Peng, Lan Feng, Qihang Zhang, Zhenghai Xue, and Bolei Zhou. Metadrive: Composing diverse driving scenarios for generalizable reinforcement learning. *IEEE transactions on pattern analysis and machine intelligence*, 2022.

[18] Zhaoting Li, Wei Zhan, Liting Sun, Ching-Yao Chan, and Masayoshi Tomizuka. Adaptive sampling-based motion planning with a non- conservatively defensive strategy for autonomous driving. In *The 21st IFAC World Congress*, 2020.

[19] Xiaodan Liang, Tairui Wang, Luona Yang, and Eric Xing. Cirl: Controllable imitative reinforcement learning for vision-based self-driving. In *Proceedings of the European conference on computer vision (ECCV)*, pages 584–599, 2018.

[20] Haochen Liu, Zhiyu Huang, and Chen Lv. Improved deep reinforcement learning with expert demonstrations for urban autonomous driving. *2022 IEEE Intelligent Vehicles Symposium (IV)*, pages 921–928, 2022.

[21] Siqi Liu, Guy Lever, Zhe Wang, Josh Merel, SM Ali Eslami, Daniel Hennes, Wojciech M Czarnecki, Yuval Tassa, Shayegan Omidshafiei, Abbas Abdolmaleki, et al. From motor control to team play in simulated humanoid football. *Science Robotics*, 7(69):eabo0235, 2022.

[22] Yiren Lu, Justin Fu, George Tucker, Xinlei Pan, Eli Bronstein, Becca Roelofs, Benjamin Sapp, Brandyn White, Aleksandra Faust, Shimon Whiteson, et al. Imitation is not enough: Robustifying imitation with reinforcement learning for challenging driving scenarios. *arXiv preprint arXiv:2212.11419*, 2022.

[23] Roberto Martín-Martín, Michelle A Lee, Rachel Gardner, Silvio Savarese, Jeannette Bohg, and Animesh Garg. Variable impedance control in end-effector space: An action space for reinforcement learning in contact-rich tasks. In *2019 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 1010–1017. IEEE, 2019.

[24] Josh Merel, Leonard Hasenclever, Alexandre Galashov, Arun Ahuja, Vu Pham, Greg Wayne, Yee Whye Teh, and Nicolas Heess. Neural probabilistic motor primitives for humanoid control. *arXiv preprint arXiv:1811.11711*, 2018.

[25] Josh Merel, Saran Tunyasuvunakool, Arun Ahuja, Yuval Tassa, Leonard Hasenclever, Vu Pham, Tom Erez, Greg Wayne, and Nicolas Heess. Catch & carry: reusable neural controllers for vision-guided whole-body tasks. *ACM Transactions on Graphics (TOG)*, 39(4):39–1, 2020.

[26] Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Overcoming exploration in reinforcement learning with demonstrations. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 6292–6299. IEEE, 2018.

[27] Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. Awac: Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020.

[28] Zhenghao Peng, Quanyi Li, Ka Ming Hui, Chunxiao Liu, and Bolei Zhou. Learning to simulate self-driven particles system with coordinated policy optimization. *Advances in Neural Information Processing Systems*, 34: 10784–10797, 2021.

[29] Karl Pertsch, Youngwoon Lee, and Joseph J Lim. Accelerating reinforcement learning with learned skill priors. *arXiv preprint arXiv:2010.11944*, 2020.

[30] Karl Pertsch, Oleh Rybkin, Jingyun Yang, Shenghao Zhou, Konstantinos Derpanis, Kostas Daniilidis, Joseph Lim, and Andrew Jaegle. Keyframing the future: Keyframe discovery for visual prediction and planning. In *Learning for Dynamics and Control*, pages 969–979. PMLR, 2020.

[31] Jan Peters and Stefan Schaal. Reinforcement learning of motor skills with policy gradients. *Neural networks*, 21 (4):682–697, 2008.

[32] Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*, 2017.

[33] Dushyant Rao, Fereshteh Sadeghi, Leonard Hasenclever, Markus Wulfmeier, Martina Zambelli, Giulia Vezzani, Dhruva Tirumala, Yusuf Aytar, Josh Merel, Nicolas Heess, et al. Learning transferable motor skills with hierarchical latent mixture policies. *arXiv preprint arXiv:2112.05062*, 2021.

[34] Desik Rengarajan, Gargi Vaidya, Akshay Sarvesh, Dileep Kalathil, and Srinivas Shakkottai. Reinforcement learning with sparse rewards using guidance from offline demonstration. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=YJ1WzgMVsMt.

[35] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.

[36] 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.

[37] Tanmay Shankar, Shubham Tulsiani, Lerrel Pinto, and Abhinav Gupta. Discovering motor programs by recomposing demonstrations. In *International Conference on Learning Representations*, 2019.

[38] Hao Shao, Letian Wang, Ruobing Chen, Hongsheng Li, and Yu Liu. Safety-enhanced autonomous driving using interpretable sensor fusion transformer. In *Conference on Robot Learning*, pages 726–737. PMLR, 2023.

[39] Sahand Sharifzadeh, Ioannis Chiotellis, Rudolph Triebel, and Daniel Cremers. Learning to drive using inverse reinforcement learning and deep q-networks. *arXiv preprint arXiv:1612.03653*, 2016.

[40] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529 (7587):484–489, 2016.

[41] 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, 1999.

[42] Mel Vecerik, Todd Hester, Jonathan Scholz, Fumin Wang, Olivier Pietquin, Bilal Piot, Nicolas Heess, Thomas Rothörl, Thomas Lampe, and Martin Riedmiller. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *arXiv preprint arXiv:1707.08817*, 2017.

[43] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782): 350–354, 2019.

[44] Letian Wang, Yeping Hu, Liting Sun, Wei Zhan, Masayoshi Tomizuka, and Changliu Liu. Hierarchical adaptable and transferable networks (hatn) for driving behavior prediction. *arXiv preprint arXiv:2111.00788*, 2021.

[45] Letian Wang, Liting Sun, Masayoshi Tomizuka, and Wei Zhan. Socially-compatible behavior design of autonomous vehicles with verification on real human data. *IEEE Robotics and Automation Letters*, 6(2):3421–3428, 2021.

[46] Letian Wang, Yeping Hu, Liting Sun, Wei Zhan, Masayoshi Tomizuka, and Changliu Liu. Transferable and adaptable driving behavior prediction. *arXiv preprint arXiv:2202.05140*, 2022.

[47] Wenshuo Wang, Letian Wang, Chengyuan Zhang, Changliu Liu, Lijun Sun, et al. Social interactions for autonomous driving: A review and perspectives. *Foundations and Trends® in Robotics*, 10(3-4):198–376, 2022.

[48] Zi Wang, Caelan Reed Garrett, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. Learning compositional models of robot skills for task and motion planning. *The International Journal of Robotics Research*, 40(6-7):866–894, 2021.

[49] Lu Zhang, Wenchao Ding, Jing Chen, and Shaojie Shen. Efficient uncertainty-aware decision-making for automated driving using guided branching. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3291–3297. IEEE, 2020.

[50] Tong Zhou, Letian Wang, Ruobing Chen, Wenshuo Wang, and Yu Liu. Accelerating reinforcement learning for autonomous driving using task-agnostic and ego-centric motion skills. *arXiv preprint arXiv:2209.12072*, 2022.