Learn Zero-Constraint-Violation Safe Policy in Model-Free Constrained Reinforcement Learning

Haitong Ma, Changliu Liu, *Member, IEEE*, Shengbo Eben Li, *Senior Member, IEEE*, Sifa Zheng, *Member, IEEE*, Wenchao Sun, and Jianyu Chen

***Abstract*— We focus on learning the *zero-constraint-violation* safe policy in *model-free* reinforcement learning (RL). Existing model-free RL studies mostly use the posterior penalty to penalize dangerous actions, which means they must experience the danger to learn from the danger. Therefore, they cannot learn a zero-violation safe policy *even after convergence*. To handle this problem, we leverage the safety-oriented energy functions to learn zero-constraint-violation safe policies and propose the safe set actor-critic (SSAC) algorithm. The energy function is designed to increase rapidly for potentially dangerous actions, locating the *safe set* on the action space. Therefore, we can identify the dangerous actions *prior to* taking them and achieve zero-constraint violation. Our major contributions are twofold. First, we use the data-driven methods to learn the energy function, which releases the requirement of known dynamics. Second, we formulate a constrained RL problem to solve the zero-violation policies. We prove that our Lagrangian-based constrained RL solutions converge to the constrained optimal zero-violation policies theoretically. The proposed algorithm is evaluated on the complex simulation environments and a hardware-in-loop (HIL) experiment with a real autonomous vehicle controller. Experimental results suggest that the converged policies in all environments achieve zero-constraint violation and comparable performance with model-based baseline.**

***Index Terms*— Constrained reinforcement learning (RL), safe RL, safety index, zero-violation policy.**

I. Introduction

REINFORCEMENT learning (RL) has drawn rapidly growing attention for its superhuman learning capabilities in many sequential decision-making problems like Go [1], Atari Games [2], and Starcraft [3]. However, such a powerful technique has not been quickly put into widespread real-world applications. For instance, most of the remarkable improvements in robotics control [4], [5], autonomous driving [6], [7], or healthcare applications [8], are only limited in the simulation platforms. One of the major issues of the real-world applications of RL algorithm is the safety guarantee. Model-free safe RL will benefit many engineering applications with unknown dynamics, like industry controller design [9], [10] and fuzzy system control [11], [12]. One of the core safety-related problems can be summarized as the notorious paradox: how to learn a safe policy without enough data or prior model about the dangerous region?

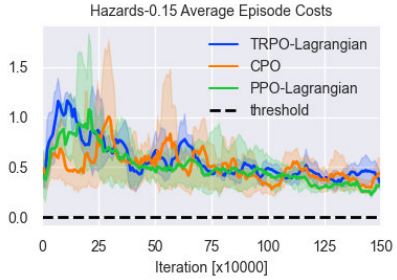


Fig. 1. Failure to guarantee safety of the posterior safety measurement by model-free constrained RL baselines with *zero-constraint threshold*. All baseline algorithms cannot converge to a zero-constraint-violation policies.

Most current safety-critical model-free RL studies do not explicitly model the safe region. The commonly used problem formulation in the RL community is the constrained Markov decision process (CMDP), where safety is modeled by limits on the total number of dangerous actions in one episode [13]. The number of dangerous actions is acquired by the cost function, usually the binary cost, which returns 1 if dangerous otherwise 0 [14], [15], [16], [17], [18]. This binary cost is a *posterior* measurement of safety, which means that the agent is penalized only after taking the dangerous action. It causes that the agent must *consistently* collect dangerous samples be aware of danger. To demonstrate the necessity of consistent dangerous action, we provide a training sample in Fig. 1.[1] In this task, the safety requirement is *zero violation*. We test the commonly used model-free constrained RL baselines that use these posterior safety measurements. The results in Fig. 1 show that they cannot ensure zero violation *even after convergence*. Similar problems also exist in other constraint formulation like risk-sensitive or chance constraint [19], [20], [21]. Therefore, most model-free constrained RL studies

Manuscript received 30 May 2022; revised 12 May 2023, 27 July 2023, and 15 October 2023; accepted 22 December 2023. Date of publication 17 January 2024; date of current version 6 February 2025. This work was supported in part by the National Key Research and Development Program of China under Grant 2020YFB1600202 and in part by the National Natural Science Foundation of China under Grant 52221005. *(Corresponding authors: Sifa Zheng; Shengbo Eben Li.)*

Haitong Ma, Shengbo Eben Li, Sifa Zheng, and Wenchao Sun are with the State Key Laboratory of Automotive Safety and Energy, School of Vehicle and Mobility, and the Center for Intelligent Connected Vehicles and Transportation, Tsinghua University, Beijing 100084, China (e-mail: maht19@tsinghua.org.cn; lisb04@tsinghua.edu.cn; zsf@tsinghua.edu.cn; swc21@tsinghua.edu.cn).

Changliu Liu is with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213 USA (e-mail: cliu6@andrew.cmu.edu).

Jianyu Chen is with the Institute of Interdisciplinary Information Science, Tsinghua University, Beijing 100084, China, and also with the Shanghai Qizhi Institute, Shanghai 200232, China (e-mail: jianyuchen@tsinghua.edu.cn).

Digital Object Identifier 10.1109/TNNLS.2023.3348422

[1]The agent is trained in a commonly used safe RL benchmark, Safety Gym. Refer to Section V-A for details.

allow *a small tolerance of constraint violation* in their problem settings [15], [16], [17], [20], [22]. However, the zero violation is required in many real-world safety-critical problems, where even small amount of danger will cause severe consequences.

Model-based safe learning adopt model information to predict future trajectory to enhance safety rather than taking the risk to explore the danger [9], [23], [24]. One commonly used technique is the energy-function-based safety certificates. Intuitively, the energy function is assigned to be low for those safe states in the *state safe set*. Typical energy functions includes the control barrier function (CBF) [25], [26], [27], [28] and the *safety index* in the safe set algorithm (SSA) [29], [30]. The energy-function-based methods first determine the *control safe set* as the set of actions to make the state safe set attractive and forward invariant. Actions in the control safe set should dissipate the system energy. The energy functions are usually integrated with RL by solving local convex optimizations. If given nominal controllers (e.g., a policy in RL problems) output actions outside the control safe set, the safe controllers find the optimal safe actions through projecting the action into the control safe set [25], [26]. After the projection, the persistent safety, or *forward invariance*, is guaranteed [31]. Therefore, the energy function is a *prior* measurement of danger and enhances safety inside the state safe set. Although providing provable safety guarantee, most energy-function-based safe controller relies on explicit dynamics model for the action projections, for example, learning-calibrated models [26], specific type of models [32], or linearity assumptions [33]. Applying these methods to general safety-critical tasks with no prior models is still challenging. Rule-based methods are another commonly seen controllers to handle complex safety-critical fuzzy control tasks, and the applications are commonly seen in fuzzy control systems [11], [12] and nonlinear model predictive controls [34]. For instance, the rules are designed to convert nonlinear systems with trade-off between conservativeness and performance [11]. Although the fuzzy system allows some unknown model information, the techniques are difficult to be migrated to model-free cases since there is no model information and there is no evidence to start with for the rule design.

In this article, we present the safe set actor-critic (SSAC) that can learn zero-constraint-violation policy in a model-free manner. In this way, the safe control policy could be learned with a limited number of constraint violations. The convergence to zero-violation safe policy is guaranteed theoretically. Without dynamics model to construct the safety monitor, we formulate a constrained RL problem with the constraints to dissipate the energy, called the *safe action constraints*. We directly learn the energy function in a model-free manner similar to the value function learning. Therefore, we can simultaneously learn the energy function transition, or the safe action set, and the optimal safe policy. The resulting constrained RL problem is solved by with the Lagrangian-based approach. The most challenging part is the state-dependent constraints, and we handle it with Lagrangian multiplier neural networks. We also provide the proof of convergence to the optimal safe policy. The main contribution of this article is listed as follows.

1) We propose the SSAC algorithm to learn a zero-constraint-violation policy in a model-free manner. We learn the energy function from data and enforce the policy to dissipate the energy. The energy function releases the requirements of learning from posterior dangerous action and thus enhances the safety.
2) We first formulate the constrained RL problems with control safe set constraints. The major difficulty is each state has a constraint, which cannot be solved by previous constrained RL algorithms. We handle it by using the Lagrange multiplier network [35]. Convergence to zero-violation safe policy is proved theoretically.
3) We evaluate the proposed algorithm on both the simulations on the state-of-the-art safe RL benchmarks and hardware-in-loop (HIL) autonomous driving tasks with a controller from a real autonomous car. Experiment results show that policies learned by the proposed algorithm achieve stable zero-constraint violation in all the experimental environments.

The article is organized as follows. Section II introduces the related work. Section III gives our problem formulation. Section IV is the practical algorithm and convergence analysis. Section V demonstrates the experimental results, and Section VI concludes the article.

II. Related Work

This section introduces the related work. Existing safe RL studies can be classified into two categories, constrained RL and RL with safe controllers. Section II-A introduces the studies of constrained RL, and Section II-B discusses about the RL with safe controllers.

A. Constrained RL

Existing model-free constrained RL studies mostly take the form of CMDP, which enforces the constraint satisfaction on the *expectation* of cost function on the trajectory distribution while maximizing the expected return or reward [13], [15], [16], [17], [18], [19], [20], [22], [23], [27], [36], [37], [38], [39], [40], [41]. As they consider the safety-critical RL problem as a constrained optimization, and various constrained optimization techniques are applied in the constrained RL methods. For example, feasible descent direction [15], [17], [36], [42], primal-dual optimization [5], [16], [20], [22], [37], [40], [41], penalty function [18], [38], and augmenting violation counts to construct new Markov decision progress (MDPs) [21], [43]. The expectation-based constrained objective function is not applicable for the state-dependent constraint commonly used in the safety-critical control problems, and our prior work proposes a neural multiplier to handle the state-dependent safe constraints in RL [35], [44], [45], [46]. However, our prior study still use the value function $v_C^\pi(s)$ for state s under policy π, or the discounted return of cost function as the constraint objective

$$v_C^\pi(s) = \mathbb{E}_{\tau\sim\pi}\left\{\sum_{t=0}^{\infty}\gamma^t c_t | s_0 = s\right\}.$$

The binary signal formulation still fails to ensure zero violation for the reason stated above. By introducing the control safe set condition as a state-dependent safety constraint, we avoid the sparsity or conservativeness issues, and the proposed algorithm is able to learn a policy with zero-constraint violation while accomplishing the task.

Model-based RL, or approximate dynamic programming (ADP), is another promising method to learn complex optimal or safe controllers. Model-based approach has been shown to be data-efficient [27], [44], [45], [46] and the Hamilton-driven methods could also handle input constraints [47], [48]. However, the requirement of an exact system dynamics is not realistic in many real-world tasks, for example, the robotics tasks with raw sensor inputs, and the uncertain pedestrian model in autonomous driving tasks. Our model-free approach can learn safe control policy from pure data, which is more powerful for those complex data-driven tasks.

B. RL With Safe Controllers

RL with safe controllers or shielding RL means that the policy output is modified or projected to a safe action before execution [26], [32], [33], [49], [50]. In some studies, the control safe set is determined by the energy-based methods, for example, the CBF [51], [52] and the SSA [29]. These energy-function-based methods can provide provable safety guarantee by enforcing the *forward invariance* of at least a subset of the state safe set [25], [30]. Nevertheless, it is difficult to find the safe projection, for example, projecting the output to a safe set on the action space [26], [53], [54], or adding a safe layer to the policy network [33]. However, to obtain the safe action and to prove forward invariance of the state safe set pose limitations on the model formulation, for example, it usually requires explicit knowledge [26], kinematic bicycle model [32], or linear property [33]. A sample-based safe set boundary estimation is proposed for black-box model [30], but it requires a digital-twin simulator of the environment. Although some recent studies proposed data-driven projection learning [54], the projection also cannot guarantee policy optimality. The novelties of our approach to handle energy function are: 1) we learn the transition model of energy function similar like the Q-function learning in a model-free manner, so we no longer rely on explicit or implicit dynamics model; and 2) we formulate constrained RL problems to learn constrained optimal *policies* rather than locally optimal safe *actions*.

III. Problem Formulation

In this section, we first describe the notion and then discuss the problem formulation of RL with the safe set safety constraints.

A. Notion

We consider the constrained RL problems wherein an agent interacts with its environment with safety constraints. This problem can be formulated an MDPs, defined by the tuple ($\mathcal{S}$, $\mathcal{A}$, $\mathcal{R}$, $\mathcal{C}$, and $\mathcal{F}$). The state space $\mathcal{S}$ and action space $\mathcal{A}$ are set to be continuous, and the reward function $r : \mathcal{S} \times \mathcal{A} \to \mathcal{R}$ maps a state–action pair to the rewards $r(s_t, a_t)$. Notably, the cost function $c : \mathcal{S} \times \mathcal{A} \to \mathcal{C}$ also maps a state–action pair to a cost function $c(s_t, a_t)$. It has a similar formulation with the reward function, which is used to formulate the constraint objective function later. For simplicity, state–action pair in the next time step of (s_{t+1}, a_{t+1}) is denoted as (s', a'). $\mathcal{F}$ represents the unknown environment dynamics for state transition. We assume a deterministic environment dynamics (the deterministic assumption is common and reasonable in safe control problems). We consider that the agent starts from an initial state distribution f_i. The initial state set $\mathcal{I} = \{s | f_i(s) > 0\} \subseteq \mathcal{S}$ is the set of all possible state in the state distribution, where $f_i(s)$ is the probability density function for the initial state distribution. The parameterized function is represented by function with right subscripts, for example, the policy $\pi_\theta(\cdot)$ with parameter θ.

B. Safe Set Algorithm

The SSA is a closely related design techniques for the safety-oriented energy function [29] $\phi : \mathbb{R}^n \to \mathbb{R}$. The energy function or the safety index is designed to satisfy (1) low energy states are safe and (2) there always exist actions to dissipate the energy. For a given energy function, the SSA indicates that the energy function cannot increase rapidly, or exceed zero. The discrete-time constraints are

$$\phi(s') < \max\{\phi(s) - \eta, 0\} \tag{1}$$

where the hyperparameter $\eta > 0$ is a slack variable to confine how fast the energy function decreases.

Definition 1 (Control Safe Set): The safe set on the action space is defined by

$$\mathcal{U}_S^D(s) := \{a \in \mathcal{A} \mid \phi(s') < \max\{\phi(s) - \eta, 0\}\}. \tag{2}$$

We explain how to construct a safety index with the case of collision avoidance safety constraint, which is one of the most common safety constraints in real-world tasks. The safety measurement to define which states are safe is called the original safety index ϕ_0. As for collision avoidance, the *original safety index* is $\phi_0 = d_{\min} - d$, where d denotes the distance between the agent and obstacle, and $d_{\min}$ is the minimal safe distance. $d > d_{\min}$ means that the current distance is greater than the minimal safe distance, which means current state is safe.

Definition 2 (State Safe Set): The safe set on the state space includes state satisfying the original safety index

$$\mathcal{S}_S := \{s \in \mathcal{S} \mid \phi_0(s) < 0\}. \tag{3}$$

However, we may not guarantee all states in the state safe set to be safe persistently, which is also defined as the *forward invariance* of the safe state set. One reason is that the input-to-constraint relation is high order ($(\partial\phi_0/\partial u) = 0$). In the collision avoidance case, if the input is some type of force-like traction, the position constraint with force input is second order. This high-order property would cause that there exist some states that would inevitably go unsafe, and we must exclude them and focus on a *subset* of the state safe set. The commonly used form was proposed as adding high-order derivatives to the safety index. The general parameterization

rules for safety index are $\phi = \phi_0 + k_1\phi_0' + \cdots + k_n\phi_0^{(n)}$, where $\phi_0^{(n)}$ is the n-order derivative of ϕ_0 [29]. In this article, we adopt a recent modified form of parameterization for the collision avoidance safety constraint in [30]

$$\phi(s) = \sigma + d_{\min}^n - d^n - k\dot{d} \tag{4}$$

where $\dot{d}$ is the derivative of distance with respect to time. $\sigma, n, k > 0$ is the parameters to tune. It is proved in [30] that if the control safe set $\mathcal{U}_S^D(s)$ is always nonempty in $\mathcal{I}$, we can conclude that agent start from the initial set $\mathcal{I}$ would converge into a *subset of state safe set* $\mathcal{I}_s = \{s|\phi(s) \le 0\} \cap \{s|\phi_0(s) \le 0\}$, *and* $\mathcal{I}_s$ *is controlled forward invariant.* Therefore, we conduct the feasibility assumption.

Assumption 1 (Feasibility): The control safe set is nonempty for $\forall s \in \mathcal{I}$ where $\phi(s)$ is the safety index function of state s.

In practical implementation, we follow the design rule of parameters σ, n, k that the control safe set is nonempty all over the state space in [30] to satisfy Assumption 1.

C. RL With Control Safe Set Constraint

We consider a constrained RL problem that maximizes the expected return while ensuring the control safe set constraint (1) on all its possible initial states

$$\begin{aligned} &\max_{\pi} \; J(\pi) = \mathbb{E}_{\tau\sim\pi}\left\{\sum_{t=0}^{\infty}\gamma^t r_t\right\} = -\mathbb{E}_{s\sim f_i(s)}v^{\pi}(s) \\ &\text{s.t. } \phi\left(s'\right) - \max\{\phi(s) - \eta, 0\} \le 0 \forall s \in \mathcal{I} \end{aligned} \tag{5}$$

where $v^{\pi}(s) = \mathbb{E}_{\tau\sim\pi}\{\sum_{t=0}^{\infty}\gamma^t r_t|s_0 = s\}$ is the value function of state s.

Proposition 1: If π^* is a feasible solution of RL problem (5) with constraint (9), then π^* will not cause any constraint violation under Assumption 1 if starting with a safe state $s \in \mathcal{I}_s$.

The proposition holds since π^* is a control law that guarantees the forward invariance of the subset of state safe set $\mathcal{I}_s$. The state will never leave $\mathcal{I}_s$ to be unsafe.

IV. Safe Set Actor-Critic

In this section, we explain the solution methods of (5) and practical algorithm. Section IV-A introduce the Lagrangian-based approach with neural multipliers. Section IV-B discusses about how we learn the transition model of safety index as a state–action value function. Section IV-C demonstrates the practical algorithm and gradient computation, and Section IV-D gives the convergence proof of the proposed SSAC algorithm.

A. Lagrangian-Based Solution

The major difficulty of solving problem (5) is the number of constraints are infinite for continuous state space. Our prior work [35] uses the Lagrangian-based approach with *neural multipliers* to approximate a mapping from states to multipliers to handle the problem (each state has a corresponding multiplier for the state-dependent control safe set constraints), denoted by the mapping $\lambda(s) : \mathbb{R}^n \to \mathbb{R}$. The Lagrange function of (5) is

$$\begin{aligned} L(\pi, \lambda) = &-\mathbb{E}_{s\sim f_i(s)}v^{\pi}(s) \\ &+ \int_{s\in\mathcal{I}} \lambda(s)\left(\phi\left(s'\right) - \max\{\phi(s) - \eta, 0\}\right). \end{aligned} \tag{6}$$

We take the negative of the reward part since standard Lagrangian function computes the minimum of the optimization objective. The mapping $\lambda(s)$ can be approximated as neural network $\lambda(s;\xi)$ with parameters ξ. The integral over the state space in (6) could not be calculated since the data is only available in the form of reply buffer distribution. Our prior work [35] solves this problem with an equivalent loss function

$$\bar{L}(\pi, \lambda) = \mathbb{E}_{s\sim f_i(s)}\left\{-v^{\pi}(s) + \lambda(s)\left(\phi\left(s'\right) - \max\{\phi(s) - \eta, 0\}\right)\right\}. \tag{7}$$

We have proved the equivalence between two functions: (6) and (7) in our prior work [35]. Intuitively, the loss function (7) can be regarded as a Lagrange function with reshaped constraints

$$f_i(s)\left(\phi\left(s'\right) - \max\{\phi(s) - \eta, 0\}\right) \le 0. \tag{8}$$

The equivalence comes from the expectation taking on the initial state distribution $f_i(s)$. If the state lies in the initial state set, $f_i(s) > 0$, and the constraint (8) is equal to the original constraint. Otherwise, $f_i(s) = 0$, it means that those states are outside the initial state set, so their safety constraints are not considered in our original problem (5).

B. Learn the Safety Index Transition

The control safe set constraint (1) consists $\phi(s')$ which we need to use *model* to predict. Related studies mostly use explicit model (both prior model or learning-calibrated model) to compute $\phi(s')$. We directly learn the transition of safety index, or the LHS of (1). This learning is similar to learning a state–action value function with a zero discounted factor, or focusing only on the energy function transition on the instant step

$$q_c^{\pi}(s, a) \doteq \phi\left(s'\right) - \max\{\phi(s) - \eta, 0\}. \tag{9}$$

For consistency with the form of value function learning, we also define the RHS of (9) as the cost function

$$c(s, a) \doteq \phi(s) - \max\{\phi(s) - \eta, 0\}. \tag{10}$$

The cost function c is a similar environment response with reward r, and they are not necessary to be related. It is easy to modify existing RL algorithm to learn safety index transition by setting the corresponding discounted factor as zero. Similar to learn the Q-function in RL, we would use a neural network $Q_{w_c}(s, a)$ to approximate $q_c^{\pi}(s, a)$, which is discussed in Section IV-C.

We claim that the cost function (10) can be obtained in the transitions (s, s') from samples in RL. Under the MDP formulation, these information should be contained in the observations of RL. Otherwise, the observation would not be enough to describe all the characteristics of the environment,

which does not satisfy the requirement of an MDPs. Take the case of collision avoidance of an autonomous vehicle, the distance relative speed to other traffic participants can be either directly measured by the Lidar or predicted by the perception module of the autonomous driving system. If some information is missing (e.g., we do not know the speed of the surrounding vehicles), using current observation to make decision is inappropriate and dangerous, which is beyond our discussion. For example, according to the parameterization (4), the required information for computing the safety index is $(d, \dot{d}, d', \dot{d}')$.

C. Algorithmic Implementation

We develop practical implementation of SSAC based on the state-or-the-art off-policy maximum entropy RL framework. Compared to unconstrained soft actor-critic [4], [55], we learn two extra neural networks, a state–action value function representing the change of safety index $Q_{w_c}(s,a)$, and a statewise Lagrange multiplier $\Lambda_\xi(s)$. We revise the notations here to make the following section clear. We have five neural networks to learn, which are the double Q network Q_{w_1}, Q_{w_2} with parameters w_1, w_2, cost Q network Q_{w_c} with parameters w_c, policy network π_θ with parameters θ, and Lagrange multiplier network Λ_ξ with parameters ξ. Their respective learning rates are $\beta_\#$, where $\# \in \{w_1, w_2, w_c, \theta, \xi\}$ denotes the parameters. Their respective loss functions are $J_Q(w_1)$, $J_Q(w_2)$, $J_C(w_c)$, $J_\pi(\theta)$, $J_\lambda(\xi)$. The detailed algorithm is demonstrated in Algorithm 1.

In the training process, the loss and gradient computation of the soft Q-function $Q_{w_*}(s,a)$ are exactly the same as those in soft actor-critic [4], [55]. $Q_{w_c}(s,a)$ is to approximate q_c^π in (9). Recall the definition of cost signal in (10), the loss function and stochastic gradient to update Q_{w_c} are

$$\begin{aligned} &J_C(w_c) \\ &= \mathbb{E}_{(s_t,a_t)\sim\mathcal{B}}\left\{\frac{1}{2}\big(Q_{w_c}(s_t,a_t) - c(s_t,a_t)\big)^2\right\} \\ &= \mathbb{E}_{(s_t,a_t)\sim\mathcal{B}}\left\{\frac{1}{2}\big(Q_{w_c}(s_t,a_t) - (\phi(s_{t+1}) - \max\{\phi(s_t)-\eta, 0\})\big)^2\right\}. \end{aligned} \tag{11}$$

The stochastic gradient of parameter w_c is

$$\hat{\nabla} J_C(w_c) = \nabla_{w_c} Q_{w_c}(s_t,a_t)\big(Q_{w_c}(s_t,a_t) - c(s_t,a_t)\big) \tag{12}$$

where $\hat{\nabla}$ denotes the stochastic gradient. The objective function of updating the policy and multipliers is both the Lagrange function in (7) since we solve a maximin optimization. In practical implementations, we actually use $Q_{w_c}(s,a)$ to approximate the LHS of inequality constraints in (5). The objective function of policy update is

$$\begin{aligned} J_\pi(\theta) = \mathbb{E}_{s_t\sim\mathcal{B}}\big\{&\mathbb{E}_{a_t\sim\pi_\theta}\{\alpha\log(\pi_\theta(a_t \mid s_t)) \\ &- Q_w(s_t,a_t) + \Lambda_\xi(s_t) Q_{w_c}(s_t,a_t)\}\big\} \end{aligned} \tag{13}$$

where $\mathcal{B}$ is the state distribution in the replay buffer, the first two terms in the expectation correspond to $v^\pi(s)$ in (5), and the third term is multipliers and constraints. The policy gradient with the reparameterized policy $a_t = f_\theta(\epsilon_t; s_t)$ can be approximated by

$$\begin{aligned} &\hat{\nabla}_\theta J_\pi(\theta) \\ &= \nabla_\theta \alpha\log(\pi_\theta(a_t \mid s_t)) + \big(\nabla_{a_t}\alpha\log(\pi_\theta(a_t \mid s_t)) \\ &\quad - \nabla_{a_t}\big(Q_w(s_t,a_t) - \Lambda_\xi(s_t) Q_{w_c}(s_t,a_t)\big)\big)\nabla_\theta f_\theta(\epsilon_t; s_t). \end{aligned}$$

Neglecting the first two terms in the expectation that has no gradient with respect to the multipliers, the objective function of updating the multiplier network parameters ξ is

$$J_\lambda(\xi) = \mathbb{E}_{s_t\sim\mathcal{B}}\big\{\mathbb{E}_{a_t\sim\pi_\theta}\big\{\Lambda_\xi(s_t)\big(Q_{w_c}(s_t,a_t)\big)\big\}\big\}$$

where $\Lambda_\xi(s_t)$ is the multiplier network. The stochastic gradient of the multiplier network parameters ξ is

$$\hat{\nabla} J_\lambda(\xi) = Q_{w_c}(s_t,a_t)\nabla_\xi \Lambda_\xi(s_t). \tag{14}$$

We invoke a multiple delayed update mechanism which update policy and multiplier each m_π or m_λ step, respectively. This delayed mechanism helps to stabilize the training process explained on the level of tricks [56]. Furthermore, in the following theoretical convergence analysis, we assume that the learning rate of multiplier net is much smaller than the

Algorithm 1 safe set actor-critic

Input: $w_1, w_2, w_C, \theta, \xi$. ▷ Initial parameters
  $\bar{\diamond} \leftarrow \diamond$ for $\diamond \in \{w_1, w_2, w_C, \theta\}$ ▷ Initialize target network weights
  $\mathcal{B} \leftarrow \emptyset$ ▷ Initialize an empty replay buffer
  for each iteration **do**
    for each environment step **do**
      $a_t \sim \pi_\theta(a_t|s_t)$, $s_{t+1} \sim p(s_{t+1}|s_t, a_t)$ ▷ Sample transitions
      Compute $c(s_t, a_t)$ according to (10)
      $\mathcal{B} \leftarrow \mathcal{B} \cup \{(s_t, a_t, r(s_t, a_t), c(s_t, a_t), \phi(s_t), s_{t+1})\}$
    end for
    for each gradient step **do**
      $w_1 \leftarrow w_1 - \beta_Q \hat{\nabla}_{w_1} J_Q(w_1)$
      $w_2 \leftarrow w_2 - \beta_Q \hat{\nabla}_{w_2} J_Q(w_2)$ ▷ Update the Q-function weights
      $w_c \leftarrow w - \beta_Q \hat{\nabla}_{w_i} J_C(w_c)$ ▷ Update the safety index transition model
      if gradient steps $\mathtt{mod}\ m_\pi = 0$ **then**
        $\theta \leftarrow \theta - \beta_\pi \hat{\nabla}_\theta J_\pi(\theta)$ ▷ Update policy weights
        $\alpha \leftarrow \alpha - \beta_\alpha \hat{\nabla}_\alpha J_\alpha(\alpha)$ ▷ Adjust temperature
      end if
      if gradient steps $\mathtt{mod}\ m_\lambda = 0$ **then**
        $\xi \leftarrow \xi + \beta_\lambda \hat{\nabla}_\xi J_\lambda(\xi)$ ▷ Update multipliers weights
      end if
      if parameters are outside the compact domain **then**
        Project the parameter into the domain
      end if
      $\bar{\diamond} \leftarrow \tau\diamond + (1-\tau)\bar{\diamond}$ for $\diamond \in \{w_1, w_2, w_C, \theta\}$ ▷ Update target network weights
    end for
  end for
Output: $w_1, w_2, w_C, \theta, \xi$.

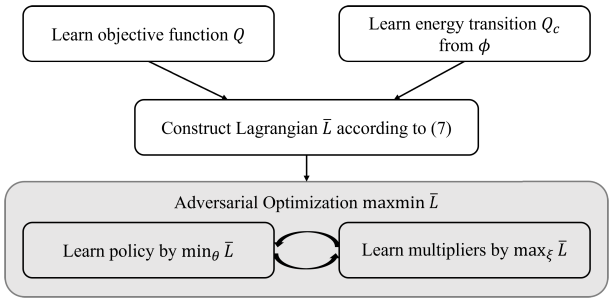


Fig. 2. Computational graph for SSAC.

policy learning rate. This delayed update mechanism can also be regarded as a practical implementation of Assumption 2.

D. Convergence Analysis

The convergence of maximum entropy RL relies on the soft policy iteration [55]. As for SSAC, the soft policy evaluation part is the same as the unconstrained soft actor-critic, which means that the Q-function $Q_w(s,a)$ and the safety index transition $Q_{w_c}(s,a)$ will converge to their real state–action value function $q^\pi(s,a), q_c^\pi(s,a)$. Therefore, we focus on the soft policy improvement. First, we give some necessary assumptions on the learning rates and objective functions.

Assumption 2: The learning rate schedules, $\{\beta_\theta(k), \beta_\xi(k)\}$, satisfy

$$\sum_k \beta_\theta(k) = \sum_k \beta_\xi(k) = \infty$$
$$\sum_k \beta_\theta(k)^2, \quad \sum_k \beta_\xi(k)^2 < \infty$$
$$\beta_\theta(k) = o\big(\beta_\xi(k)\big). \tag{15}$$

Assumption 3 (Differentiability): $\theta \in \Theta$, $\xi \in \Xi$, $\zeta \in \mathcal{Z}$ which are all compact sets. All the neural network approximations are Lipschitz continuous.

We use an ordinary differential equation (ODE) viewpoint to prove the convergence, which is a branch of standard convergence analysis of stochastic optimization algorithms [20], [39], [57], [58]. Take the case of θ update rule in Algorithm 1

$$\theta \leftarrow \theta - \beta_\theta \hat{\nabla}_\theta J_\pi(\theta). \tag{16}$$

It can be regarded as a stochastic approximation of a continuous dynamic system $\theta(t)$

$$\dot{\theta} = -\nabla_\theta J_\pi(\theta) \tag{17}$$

and we will prove that dynamic system $\theta(t)$ will converge to a local optimum θ^*, which is the solution of the ODE (17) and also a stationary point.

Theorem 1: Under all the aforementioned assumptions, the sequence of policy and multiplier parameters tuple (θ_k, ξ_k) converge almost surely to a locally optimal policy and multiplier parameters tuple (θ^*, ξ^*) as k goes to infinity.

The theoretical computational graph is shown in Fig. 2. Before going detailed into the proof, the high-level overview is as follows.

1) First, we show that each update cycle of the multiple timescale discrete stochastic approximation algorithm (θ_k, ξ_k) converges almost surely, but at different speeds, to the stationary point (θ^*, ξ^*) of the corresponding continuous time system.
2) By Lyapunov analysis, we show that the continuous time system is locally asymptotically stable at (θ^*, ξ^*).
3) We prove that (θ^*, ξ^*) is locally optimal solution, or a local saddle point for the constrained RL problems.

The convergence proof mainly follows the multitimescale convergence and Theorem 2 in Chapter 6 of Borkar's book [57], and also the standard procedure in proving convergence of stochastic programming algorithms [39], [58], [59]. The major difference and our contributions are as follows. 1) We prove the saddle point convergence while previous analyses are local minima. We improve the Lyapunov analysis to prove the saddle point is stable. 2) We consider the estimation error in the Lyapunov analysis. The intuition to guarantee safety behind this proof can be summarized as follows. For any system, as long as an energy function ϕ satisfying (2) is known, then we can use Lagrangian-based methods to solve a safe (but could be suboptimal) policy. The convergence to safe policy is proved by Lyapunov-based approach. We treat the policy gradient optimization as a dynamical system and prove that it is stable near the safe policy.

Proof of Theorem 1:

Step 1 (Convergence of θ Update):

The θ update can be also formulated as using the stochastic gradient descent

$$\theta_k = -\hat{\nabla}_\theta L(\theta, \xi)\big|_{\theta=\theta_k} + \delta\theta_{k+1}. \tag{18}$$

Consider the gradient with respect to θ at k step is as a random variable $G_{\theta_k}(s_t, a_t)$. First, we give the lemma about

Lemma 1: Lipschitz continuous of gradient respect to θ

$$\begin{aligned}\hat{\nabla}_\theta L(\theta,\xi) = G_{\theta_k}(s_t,a_t) = \nabla_\theta \alpha \log(\pi_\theta(a_t \mid s_t)) \\ + \big(\nabla_{a_t} \alpha \log(\pi_\theta(a_t \mid s_t)) \\ - \nabla_{a_t}\big(Q_w(s_t,a_t) - \lambda_\xi(s_t) J_{w_c}^{\pi_\theta}(s_t,a_t)\big)\big) \\ \nabla_\theta f_\theta(\epsilon_t; s_t).\end{aligned} \tag{19}$$

Proof: As the neural network policy and value are continuously differentiable, the squashed Gaussian function is constructed by two steps, i.e., compute the action by a normal distribution $u \sim N(\mu(\theta), \sigma^2(\theta))$, and squashing the action by tanh. As both the probability density of normal distribution and tanh are Lipschitz, the $G_{\theta_k}(s_t, a_t)$ or $\hat{\nabla}_\theta L(\theta, \xi)$ is also Lipschitz □

$$\theta_k = -\hat{\nabla}_\theta L(\theta, \xi)\big|_{\theta=\theta_k} + \delta\theta_{k+1} \tag{20}$$

and the Martingale difference term with respect to θ is computed by

$$\delta\theta_{k+1} = \hat{\nabla}_\theta L(\theta,\xi)|_{\theta=\theta_k} - \mathbb{E}_{s\sim d_\gamma^{\pi_\theta}}\big\{\mathbb{E}_{a\sim\pi_\theta} G_{\theta_k}(s_t,a_t)\big\}. \tag{21}$$

The state–action pair used to estimate $\nabla_\theta L$ is sampled from buffer $\mathcal{D}$ which use policy π_θ to collect, so $\mathbb{E}\{\delta\theta_{k+1} \mid \mathcal{F}_{\theta,k}\} = 0$, which means $\{\theta_{k+1}\}$ is a Martingale difference sequence.

Besides, the θ update in (18) is a stochastic approximation of ODE

$$\dot{\theta} = -\nabla_\theta L(\theta, \xi)|_{\theta=\theta_k}. \tag{22}$$

Now, we prove that the Martingale difference is square integrable

$$\begin{aligned}&\mathbb{E}\left\{\|\delta\theta_{k+1}\|^2 \mid \mathcal{F}_{\theta,k}\right\}\\ &\quad\le 2\|d_\gamma^{\pi_\theta}(s)\pi(a|s)\|_\infty^2\left(\|G_{\theta_k}(s,a)\|_\infty^2 + |G_{\theta_k}(s_t,a_t)|^2\right)\\ &\quad\le 6\|d_\gamma^{\pi_\theta}(s)\pi(a|s)\|_\infty^2\|G_{\theta_k}(s,a)\|_\infty^2.\end{aligned} \tag{23}$$

Combining all the assumptions and derivations from (18) and (23) and invoking Theorem 2 in Chapter 2 of Borkar's book [57] (stochastic approximation theory for continuous dynamic systems), the sequence $\{\theta_k\}$ converges almost surely to a fixed point θ^* for ODE (22).

Then, we show that the fixed point θ^* is a stationary point using Lyapunov analysis.

Proposition 2: For dynamical system (22) with given ξ, we can define $\mathcal{L}_\xi(\theta)$ as

$$\mathcal{L}_\xi(\theta) = L(\theta,\xi) - L\left(\theta^*,\xi\right) \tag{24}$$

where $\theta^* \in \Theta$ is a local minimum. Then, $\mathcal{L}_\xi(\theta)$ is a Lyapunov function for dynamical system (22) stabilizing at θ^*, which means

$$d\mathcal{L}_\xi(\theta)/dt \le 0. \tag{25}$$

Proof: As we assume that $\theta \in \Theta$ which is a compact set, if θ lies on the boundary and gradient direction points out of Θ, the update should project θ into Θ. Then, we separate the proof into two cases.

Case 1: $\theta \notin \partial\Theta$ or $\theta \in \partial\Theta$, and the gradient points inside Θ. Then, we have

$$\frac{dL(\theta,\xi)}{dt} = -\|\nabla_\theta L(\theta,\xi)\|^2 \le 0. \tag{26}$$

The equality holds only when $\nabla_\theta L(\theta,\xi) = 0$.

Case 2: $\theta \in \partial\Theta$, and the gradient points outside Θ, and the time derivative of Lyapunov function is

$$\frac{dL(\theta,\xi)}{dt} = \nabla_\theta L(\theta,\xi)^T \lim_{\eta\to 0}\frac{\Gamma_\Theta\{\theta - \nabla_\theta\eta L(\theta,\xi)\} - \theta}{\eta} \tag{27}$$

where $\Gamma_\Theta\{\cdot\}$ is to project a vector to θ. Denote $\theta_\eta = \theta - \eta\nabla_\theta L(\theta,\xi)$ for a positive η. According to proposition 2.1.3 in [60]

$$\left(\theta_\eta - \Gamma_\Theta\{\theta_\eta\}\right)^T\left(\theta - \Gamma_\Theta\{\theta_\eta\}\right) \le 0 \quad \forall\theta \in \Theta \tag{28}$$

which further implies that

$$\begin{aligned}&\left(\theta - \theta_\eta\right)^T\left(\Gamma_\Theta\{\theta_\eta\} - \theta\right)\\ &= \left(\theta_\eta - \Gamma_\Theta\{\theta_\eta\} + \Gamma_\Theta\{\theta_\eta\} - \theta\right)^T\left(\theta - \Gamma_\Theta\{\theta_\eta\}\right)\\ &= -\|\theta - \Gamma_\Theta\{\theta_\eta\}\|^2 + \left(\theta_\eta - \Gamma_\Theta\{\theta_\eta\}\right)^T\left(\theta - \Gamma_\Theta\{\theta_\eta\}\right) \le 0.\end{aligned} \tag{29}$$

Then the time derivative is

$$\begin{aligned}\frac{dL(\theta,\xi)}{dt} &= \frac{\left(\theta - \theta_\eta\right)^T}{\eta}\lim_{\eta\to 0}\frac{\Gamma_\Theta\{\theta_\eta\} - \theta}{\eta}\\ &= \frac{1}{\eta}\lim_{\eta\to 0}\left(\theta - \theta_\eta\right)^T\left(\Gamma_\Theta\{\theta_\eta\} - \theta\right) \le 0\end{aligned} \tag{30}$$

for positive η. The equality only holds when the directional derivative term $\|\lim_{\eta\to 0}(\Gamma_\Theta\{\theta - \nabla_\theta\eta L(\theta,\xi)\} - \theta/\eta)\| = 0$. Therefore, θ^* is a stationary point of dynamic system $\theta(t)$. □

Combining the aforementioned conclusions, $\{\theta_k\}$ converges almost surely to a local minimum point θ^* for given ξ.

Step 2 (Convergence of λ Update):

The stochastic gradient with respect to ξ is

$$G_\lambda(s_t,a_t) = \hat{\nabla}L_\xi\left(\theta_\xi^*,\xi\right) = \left(J_{w_c}(s_t,a_t)\right)\nabla_\xi\lambda_\xi(s_t). \tag{31}$$

Proposition 3: $\nabla_\xi L(\theta_\xi^*,\xi)$ is Lipschitz on ξ.

Proof: $J_{w_c}(s_t,a_t)$ takes finite values by (4). Considering that Ξ is a compact set and the continuous differentiable assumption of neural network, the $\nabla_\xi L(\theta_\xi^*,\xi)$ is Lipschitz. □

Similarly, the ξ update can also be formulated as

$$\xi_{k+1} = \hat{\nabla}_\xi L(\theta,\xi)|_{\theta=\theta_k,\xi=\xi_k} + \delta\xi_{k+1} + \delta\theta_\epsilon \tag{32}$$

where $\theta_k = \theta_\xi^* + \epsilon_\theta$, ϵ_θ is an estimation error in θ update. Similarly, the error term

$$\delta\xi_{k+1} = \hat{\nabla}_\xi L\left(\theta^*(\xi),\xi\right)|_{\xi=\xi_k} - \mathbb{E}_{s\sim d_\gamma^{\pi_\theta}}\left\{\mathbb{E}_{a\sim\pi_\theta}G_{\xi_k}(s_t,a_t)\right\} \tag{33}$$

is square integrable since

$$\begin{aligned}&\mathbb{E}\left\{\|\delta\xi_{k+1}\|^2 \mid \mathcal{F}_{\xi,k}\right\}\\ &\le 4\|d_\gamma^{\pi_\theta}(s)\pi(a|s)\|_\infty^2\left(\max\|Q_{w_C}^{\pi_\theta^*}(s_t,a_t)\|^2 + d^2\right)\|\nabla_\xi\lambda_\xi(s_t)\|_\infty^2.\end{aligned} \tag{34}$$

The error term caused by θ estimation error is

$$\begin{aligned}\delta\theta_\epsilon &= \hat{\nabla}_\xi L(\theta,\xi)|_{\theta=\theta_k,\xi=\xi_k} - \hat{\nabla}_\xi L\left(\theta^*(\xi),\xi\right)|_{\xi=\xi_k}\\ &= \left(Q^{\pi_{\theta_k}}(s_t,a_t) - Q^{\pi_{\theta^*}}(s_t,a_t)\right)\nabla_\xi\lambda_\xi(s_t)\\ &= \left(\nabla_a Q(s_t,a_t)\nabla_\theta\pi(s_t)\epsilon_{\theta_k} + o\left(\|\epsilon_{\theta_k}\|\right)\right)\nabla_\xi\lambda_\xi(s_t).\end{aligned} \tag{35}$$

Therefore, $\|\delta\theta_\epsilon\| \to 0$ as $\|\epsilon_\theta\| \to 0$. Combining the Lipschitz continuity and error property, one can invoke Theorem 2 in Chapter 6 of Borkar's book [57] to show that the sequence $\{\xi_k\}$ converges to the solution of the following ODE:

$$\dot{\xi} = -\nabla_\xi L(\theta,\xi)|_{\theta=\theta^*(\xi)}. \tag{36}$$

Then, we show that the fixed point ξ^* is a stationary point using Lyapunov analysis. Note that we have to take ϵ_θ into considerations.

Proposition 4: For the dynamic system with error term

$$\dot{\xi} = -\nabla_\xi L(\theta,\xi)|_{\theta=\theta^*(\xi)+\epsilon_\theta}. \tag{37}$$

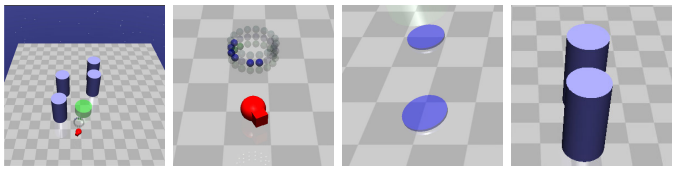
Define a Lyapunov function to be

$$\mathcal{L}(\xi) = L\left(\theta^*,\xi\right) - L\left(\theta^*,\xi^*\right) \tag{38}$$

where ξ^* is a local maximum point. Then, $(d\mathcal{L}(\xi)/dt) \le 0$.

Proof: The proof is similar to Proposition 2, only the error of θ should be considered. We prove that the error of θ does not affect the decreasing property here

$$\begin{aligned}\frac{d\mathcal{L}(\xi)}{dt} &= -\left(\nabla_\xi L(\theta,\xi)|_{\theta=\theta^*(\xi)+\epsilon_\theta}\right)^T\nabla_\xi L(\theta,\xi)|_{\theta=\theta^*(\xi)}\\ &= -\|\nabla_\xi L(\theta,\xi)|_{\theta=\theta^*(\xi)}\|^2 - \delta\theta_\epsilon^T\nabla_\xi L(\theta,\xi)|_{\theta=\theta^*(\xi)}\end{aligned}$$



(a) (b) (c) (d)

Fig. 3. Safe exploration environments in Safety Gym. The agent should reach the goal position, the green cylinder in (a), while avoiding other obstacles. Hazards are not solid objects, hence can be crossed over. Pillars are solid and cause physical contact with agents. (a) Safety Gym. (b) Point agent. (c) Hazard. (d) Pillars.

$$\leq -\|\nabla_\xi L(\theta,\xi)|_{\theta=\theta^*(\xi)}\|^2 + K_1\|\epsilon_\theta\|\|\nabla_\xi L(\theta,\xi)|_{\theta=\theta^*(\xi)}\| = \left(-K_2\|\epsilon_\xi\| + o(\|\epsilon_\xi\|) + K_1\|\epsilon_\theta\|\right)\|\nabla_\xi L(\theta,\xi)|_{\theta=\theta^*(\xi)}\|. \tag{39}$$

As θ converges faster than ξ, $d\mathcal{L}_\zeta(\xi)/dt \leq 0$, so there exists trajectory $\xi(t)$ converges to ξ^* if initial state ξ_0 starts from a ball $\mathcal{B}_{\xi^*}$ around ξ^* according to the asymptotically stable systems. □

V. Experimental Results

We demonstrate the effectiveness of the proposed algorithm by evaluating them on multiple safety-critical tasks in this section. Section V-A demonstrates four safe exploration tasks on the state-of-the-art safe RL benchmark, Safety Gym [16], and Section V-B depicts an HIL experiment of autonomous driving tasks with a real controller module of autonomous vehicle.

A. Safe Exploration Task

1) Environment Setup: Safety Gym (shown in Fig. 3) includes a series of benchmark environments specially designed for safe RL and safe exploration tasks using Gym API and MuJoCo simulator [16], [61], [62]. The task of safety gym is to control an agent (we choose the `Point` agent) to accomplish different types of tasks in a 2-D arena. There exist hazards and different kinds of obstacles in the 2-D arena with random positions, and the agent must avoid them when reaching the goal. Four environment setups with different types and sizes of the obstacles are considered, named as `{Type}-{Size}`. The safety constraint is not to touch any obstacles or reach any hazards while accomplishing the tasks. The parameters we choose for (4) satisfies the safety index synthesis rule proposed in [30], which satisfies the feasibility in Assumption 1.

We compare the proposed algorithm against different types of baseline algorithms: 1) commonly used constrained RL baseline algorithms: constrained policy optimization (CPO) [15], Lagrangian modification version of trust-region policy optimization (TRPO-Lagrangian), and proximal policy optimization (PPO-Lagrangian) [16]; 2) a state-of-the-art safety monitor projecting the action through an adaptive boundary estimation of control safe set, called the implicit SSA (PPO-ISSA) [30]; and 3) state-of-the-art model-free constrained RL algorithms, first-order CPO (FOCOPS) [22], Saute RL [43], and penalized proximal policy optimization (P3O) [63]. PPO-ISSA has a known online-query environment simulator. The online-query simulator means that PPO-ISSA can reset the system states freely. As we do not have the dynamics model for Safety Gym, other safe monitor baselines (such as Cheng et al. [26]) cannot be implemented. Other experimental details are listed in Appendix A.

2) Simulation Results: The training curves in Safety Gym environments can be found in Fig. 4. Results show that the converged policy of SSAC and PPO-ISSA both achieves solid zero-constraint violation. Notably, SSAC is model-free, which means we only learn from data and do not know the world model or has the capability to reset agent state. Although PPO-ISSA achieves zero violation even during the training process, it actually requires a simulator which can reset at any time. This is a much more strict assumption than SSAC. Therefore, our algorithm can learn a solid safe policy with the same or better performance compared to PPO-ISSA with much more strict assumptions. Moreover, all constrained RL baseline in type (1) and (2) show little but existing constraint violations *with a zero-constraint threshold*. SSAC has comparable reward performance with PPO-ISSA and learns much faster than PPO-ISSA, which shows that learning an integrated policy (optimum on the policy space) and using an external safe monitor (optimum on local action space) have nearly the same reward performance. PPO-ISSA has slower learning process, which might be explained by that the initial violations give SSAC samples of better reward performance. The total reward of SSAC is not as good as baselines, which is reasonable since some baseline achieves higher reward because they directly cross the hazards instead of avoiding them [15], [16].

To better understand the zero-violation property, we count the accumulative cost rate in the training process in Fig. 4. Results reveal that the cost rate of SSAC *converges to zero*, while the curves of baseline algorithms remain at a positive number. The facts verify that baseline algorithms need consistent cost signals to identify danger, while SSAC does not need explorations in the dangerous region once converged.

B. Autonomous Driving Control Task

1) Environment Setup: The autonomous driving environment models a real intersection located at (31°08′13″N, 120°35′06″E) in the SUMO network editor shown in Fig. 5. The random traffic flow is generated by the SUMO traffic simulator. The vector-level observation (not raw data from Lidars) includes three parts: the state of the ego vehicle, a series of tracking points from a pre-defined static reference trajectory, and also the state of surrounding vehicles (including positions, velocity, and heading angle). Detailed observations are shown in Table I.

The control input is the desired steering angle δ_{des} and acceleration a_{des} of ego vehicle, where details are shown in Table II. The static reference trajectories are generated by Bezier curves with several reference points according to the road topology. We selected *focused surrounding vehicles* in all the surrounding vehicles to handle the variable surrounding vehicles and safety constraints. The surrounding vehicle is filtered and listed in a specific order (first by route, then by lateral position) to avoid the permutation error in state representation. We name them as the *filtered surrounding vehicles* in the sections below. The filtering rule is a fixed number of surrounding vehicles in each route, such as four

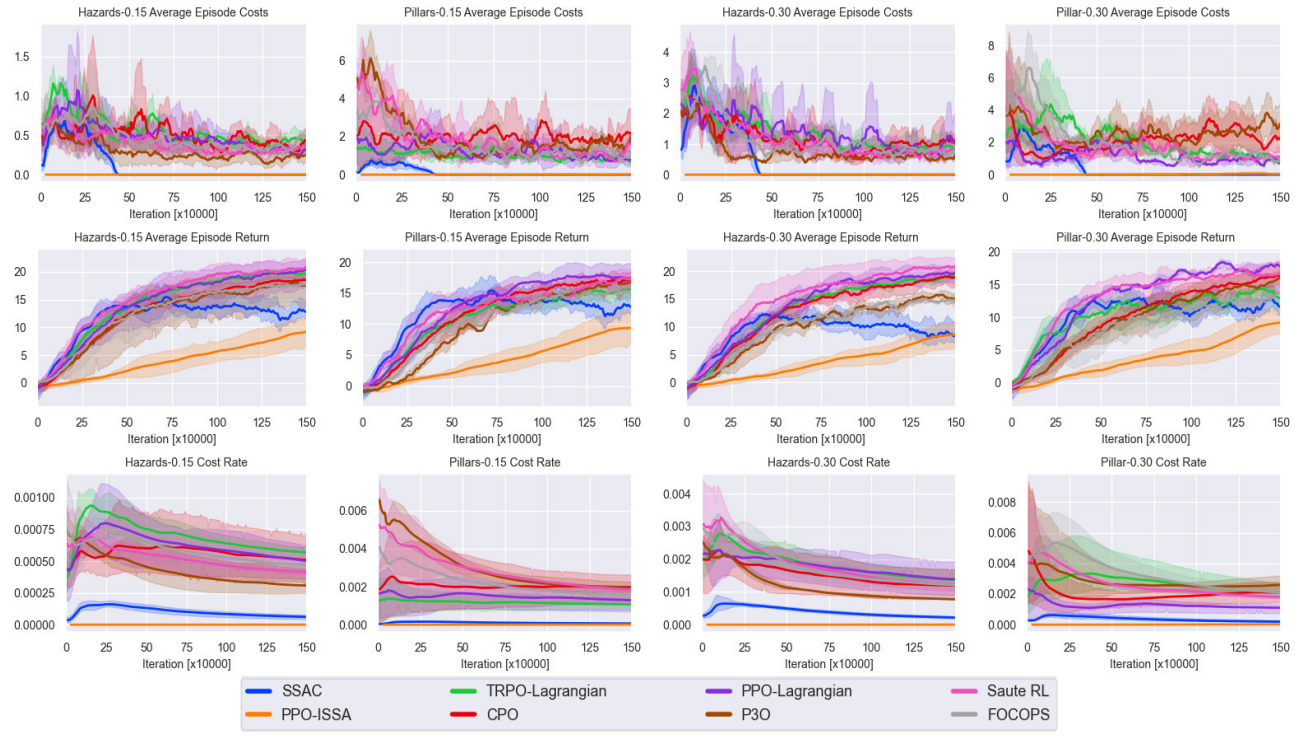
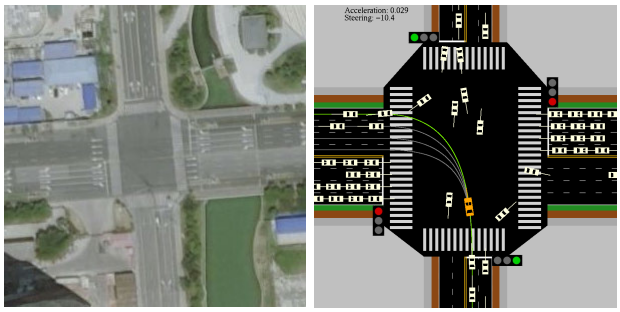


Fig. 4. Learning curves for safe exploration in Safety Gym environments. The x-axis represents the training iterations; and the y-axis represents the respective value in the title of the last 20 episodes. The solid lines represent the mean value over five random seeds. The shaded regions represent the 95% confidence interval. SSAC converges to a *zero-constraint-violation* policy with a reasonable performance sacrifice, while baseline algorithms all cannot achieve zero violation.



(a) (b)

Fig. 5. Environment scene of autonomous driving control of turning left at intersection. The ego vehicle (yellow one) needs to track one of the reference paths or the highlighted one, while avoiding collisions with the surrounding vehicles (those white ones). (a) Real scene. (b) Virtual scene.

vehicles turning left from the current road. If there are not enough vehicles, we add virtual vehicles in the filtered surrounding vehicles in distant places (outside the intersection) to make up the state dimension without affecting the ego vehicle [64]. Notably, the position, heading angle, and speed of both ego vehicle and filtered surrounding vehicles are in the observations, and we can use them to compute the necessary distance term and its first-order time derivative in (4). We use a kinetics bicycle vehicle model with a nonlinear tire model to simulate the ego vehicle response [65], and the filtered surrounding vehicles are modeled and controlled by SUMO. The reward is designed with a linear combination of squared tracking errors of position and heading angle with respect to the highlighted reference trajectory in Fig. 5

$$r(s,a) = -\left(0.04\delta_p^2 + 0.01\delta_v^2 + 0.1\delta_\phi^2 + 0.1\delta^2 + 0.005a_{\text{des}}^2\right). \tag{40}$$

TABLE I

STATE IN THE INTERSECTION ENVIRONMENT

Observations	Notions	Meanings
Ego vehicle	p_x	Longitudinal position [m]
	p_y	Lateral position [m]
	η	Heading angle [°]
	v_x	Longitudinal velocity [m/s]
	v_y	Lateral velocity [m/s]
	r	yaw rate [rad/s]
Surrounding vehicle (i^{th})	p_x^i	Longitudinal position [m]
	p_y^i	Lateral position [m]
	v^i	velocity [m/s]
	η^i	Heading angle [°]
Tracking error	δ_p	position error [m]
	δ_η	Heading angle error [°]
	δ_v	velocity error [m/s]

The safety constraints are considered with a double circle design as shown in Fig. 6. For each pair of ego and filtered surrounding vehicles, we have four collision avoidance constraints. We set the output dimension of the multiplier network to match the constraint dimension (4× number of filtered surrounding vehicles). Other environment setup details can be found in our previous studies using the same simulation environment [27], [64].

2) HIL Simulation System: We design an HIL system to verify the effectiveness of the proposed algorithm. The controller is a GEAC91S series AI computing unit from TZTEK, as shown in Fig. 7, based on an NVIDIA Jetson AGX Xavier. First, the policy is trained on a personal computer or cloud computing services, and then we convert the neural network

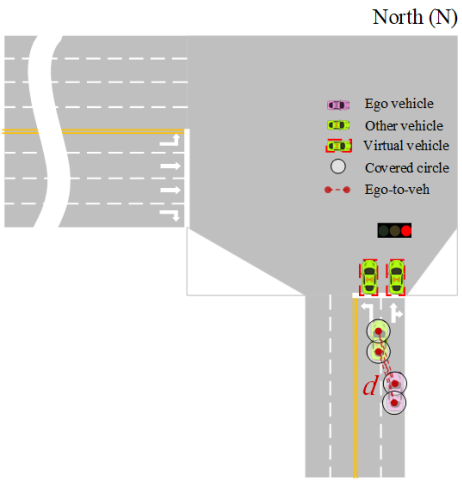


Fig. 6. Collision avoidance constraints by double circle.

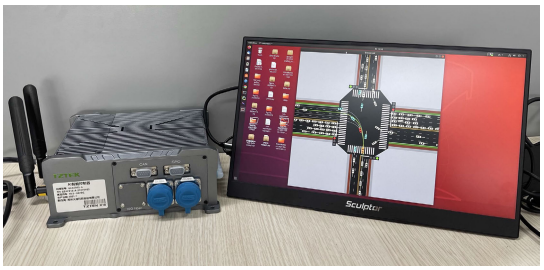


Fig. 7. HIL system with real autonomous vehicle controller.

to a hardware-compatible model and implement it on the real autonomous driving controller. The neural network model of the policy is converted from the pb model (trained and saved in TensorFlow) to the TensorRT engine, which is optimized and accelerated by TensorRT, an inference framework from Nvidia. We do not train the model on the controller. Instead, we use a communication message queue, ZeroMQ, to connect the controller module and a personal computer with simulation environments shown in Fig. 7. The environment observations are transmitted to the controller and fed into the network. The network then outputs the control command using the inference API inside the controller. Finally, the control action is sent back to the simulation environment using the communication message queue.

3) Training Results: The training process is purely on the simulation environments before implementing on the hardware, and the learning curves are shown in Fig. 8. The implicit SSA requires a simulator that could be reset to a previous state [30], and SUMO simulator does not support the vehicles reset. Therefore, we do not use the safe controller baseline, PPO-ISSA, for this experiment. The episodic cost performance is similar to the previous safe exploration tasks. SSAC consistently enforces the episodic costs to stay at zero. Fig. 8 shows the training performance in the hardware in loop tasks, including average episodic constraint violations (AverageEpCost), episodic sum of reward (AverageEpRet), and constraint violation rate (CostRate). The episodic constraint violation results indicate that the episodic

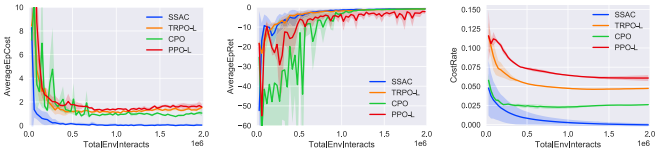


Fig. 8. Training curves of the offline training stage. All baseline algorithms still have a small but non-negligible constraint violations. SSAC achieves zero-constraint violations.

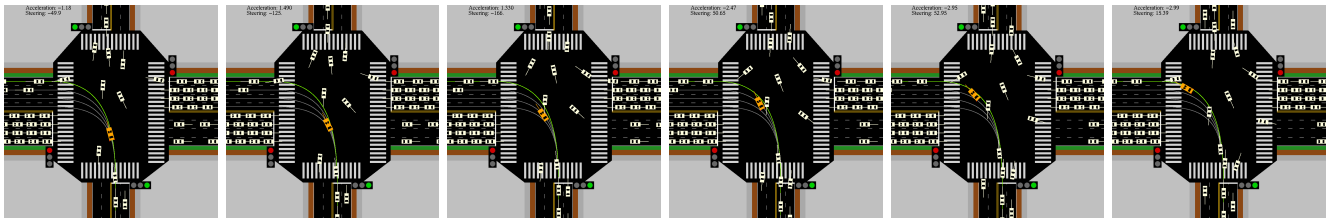
TABLE II
ACTION IN THE INTERSECTION ENVIRONMENT

Outputs	Notions	Limits
Desired Steering Angle	δ_{des}	$[-229°, 229°]$ †
Desired Acceleration	a_{des}	$[-3.0\text{m/s}^2, 1.5\text{m/s}^2]$

† Negative steering angle means turning left.

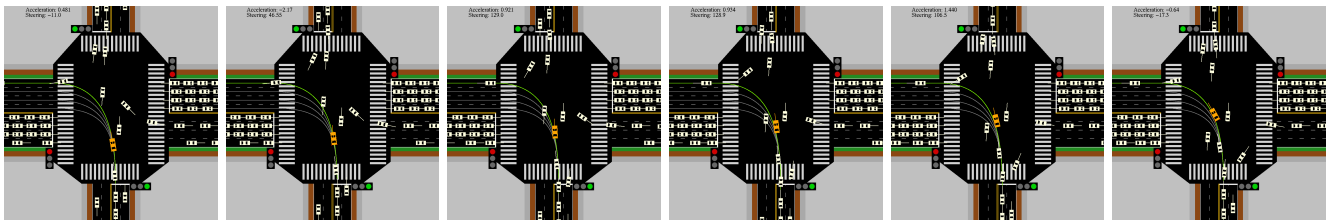
costs converge to zero for SSAC, which corresponds to our claimed zero-violation safe policy. For the baselines, their average episode costs all oscillate near zero, which is similar to Fig. 1. The reason is that baseline-constrained RLs can only learn safe policy from violations. Every time the agent achieves zero violation, the agent is not aware of danger and the reward will push the agent to danger again. This is the reason of oscillation near zero and proves the contribution and necessity of the proposed algorithm. The sum of reward results shows that SSAC's tracking performance is comparable to baselines while guarantee zero violation. We count the cost rate to further demonstrate the zero-constraint violation property. The cost rate of the proposed algorithm *converges to zero*, while the curves of other algorithms remain at a positive number.

4) Implementation Results: After training, we implemented the algorithm on the controller with the hardware compatible inference module. We choose the left turn, the task with the most potential conflicts with surrounding vehicles, to compare the differences in the collision avoidance maneuvers between the proposed and baseline algorithms. We select a typical case when the ego vehicle has to avoid a fast vehicle from the opposite straight line. We capture the screen every 500 ms, and select the conflict periods as shown in Figs. 9–11. The first two episodes in Figs. 9 and 10 demonstrate two different avoidance maneuvers of SSAC policy. We add the reference trajectory (the highlighted one) to compare how much the ego vehicle deviates from it under different algorithms. SSAC in Fig. 9 first decelerates and then accelerate to pass from below (or left from the perspective of the ego vehicle). The ego vehicle keeps a small but distinguishable distance from the opposite vehicle. SSAC in Fig. 10 brakes harder and deviates to the right of the reference trajectory to wait for the opposite vehicle to pass and then continue to pass the intersection. As for baseline algorithm like CPO in Fig. 11, it does not choose the wait maneuver since it hurries to pass the intersection. CPO policy does not take enough steering or decelerate when passing for below, so it collides with the opposite vehicle. CPO turns smaller than SSAC and does not decelerate, resulting in collisions with the a white vehicle from the opposite side.



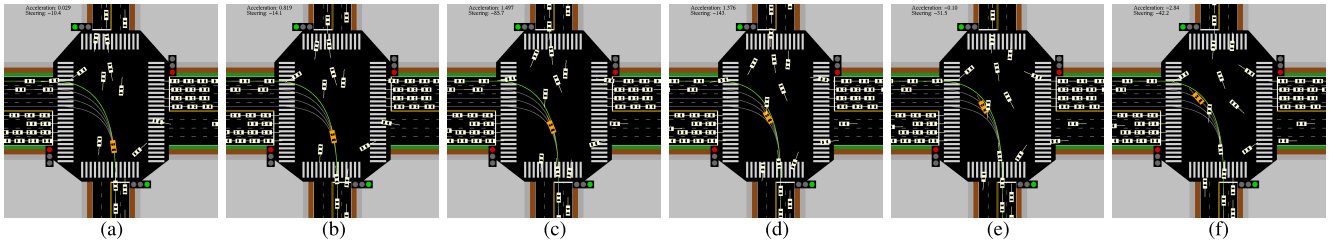
(a) (b) (c) (d) (e) (f)

Fig. 9. Typical collision avoidance maneuvers using SSAC: *left bypassing*. The ego vehicle first brakes a bit, then turn left and accelerate to bypass the opposite vehicle. The ego vehicle then brakes hard to avoid the next vehicle turning right in front. (a) 7.0 s, $\delta_{\text{des}} = -49.9°$, $a_{\text{des}} = -1.18\text{m/s}^2$. (b) 7.5 s, $\delta_{\text{des}} = -125°$, $a_{\text{des}} = 1.49\text{m/s}^2$. (c) 8.0 s, $\delta_{\text{des}} = -166°$, $a_{\text{des}} = 1.33\text{m/s}^2$. (d) 8.5 s, $\delta_{\text{des}} = 50.7°$, $a_{\text{des}} = -2.47\text{m/s}^2$. (e) 9.0 s, $\delta_{\text{des}} = 53.0°$, $a_{\text{des}} = -2.95\text{m/s}^2$. (f) 9.5 s, $\delta_{\text{des}} = 15.4°$, $a_{\text{des}} = -2.99\text{m/s}^2$.



(a) (b) (c) (d) (e) (f)

Fig. 10. Typical collision avoidance maneuvers using SSAC: *right bypassing*. The ego vehicle first brakes a bit, then deviate to the right of the reference trajectory and wait the opposite vehicle to pass. Then, ego vehicle accelerates to pass the intersection. (a) 6.5 s, $\delta_{\text{des}} = -11.0°$, $a_{\text{des}} = 0.48\text{m/s}^2$. (b) 7.0 s, $\delta_{\text{des}} = 46.6°$, $a_{\text{des}} = -2.17\text{m/s}^2$. (c) 7.5 s, $\delta_{\text{des}} = 129°$, $a_{\text{des}} = 0.92\text{m/s}^2$. (d) 8.0 s, $\delta_{\text{des}} = 128.9°$, $a_{\text{des}} = 0.94\text{m/s}^2$. (e) 8.5 s, $\delta_{\text{des}} = 106.5°$, $a_{\text{des}} = 1.44\text{m/s}^2$. (f) 9.0 s, $\delta_{\text{des}} = -17.3°$, $a_{\text{des}} = -0.64\text{m/s}^2$.



(a) (b) (c) (d) (e) (f)

Fig. 11. Typical collision avoidance maneuvers using baseline algorithm (CPO): *collision*. The ego vehicle does not brake in advance, and turn left later than SSAC, which causes that the ego vehicle collides with the vehicle from the opposite direction. (a) 6.5 s, $\delta_{\text{des}} = -10.4°$, $a_{\text{des}} = 0.03\text{m/s}^2$. (b) 7.0 s, $\delta_{\text{des}} = -14.1°$, $a_{\text{des}} = 0.82\text{m/s}^2$. (c) 7.5 s, $\delta_{\text{des}} = -85.7°$, $a_{\text{des}} = 1.50\text{m/s}^2$. (d) 8.0 s, $\delta_{\text{des}} = -143°$, $a_{\text{des}} = 1.38\text{m/s}^2$. (e) 8.5 s, $\delta_{\text{des}} = -31.5°$, $a_{\text{des}} = -0.10\text{m/s}^2$. (f) 9.0 s, $\delta_{\text{des}} = -42.2°$, $a_{\text{des}} = -2.84\text{m/s}^2$.

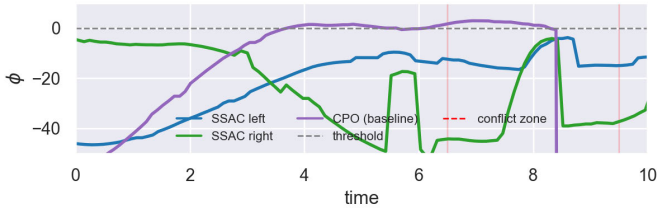


Fig. 12. Safety index $\phi(s)$ value with three corresponding cases. The curve indicates the highest value function among all the filtered surrounding vehicles. The sudden decreasing of safety index is because the vehicles on the right side of ego vehicle are not considered in the state representation.

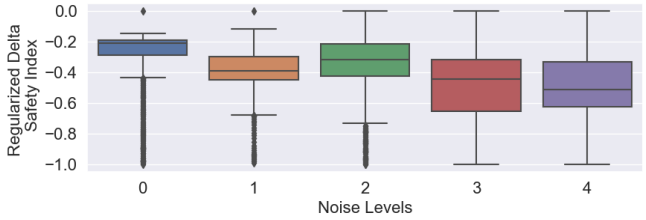


Fig. 13. Safety index transition value $\phi(s') - \max\{\phi(s) - \eta, 0\}$ distribution with different observation noise levels.

For the purpose of verifying the constraint-satisfying property of control safe set conditions (9), we first select four random episodes to see the safety index value shown in Fig. 12. As the maximum distance of each random task is different, we regularize the safety index value to demonstrate the trends in the whole episode. It is obvious that $\phi(s)$ is effectively confined when it reaches closely to 0. It verifies the satisfaction of the safe set safety constraint (9). Considering real-world sensor noise, we further add white noise term on the observation passed to the controllers, and Table III depicts the noise level. Fig. 13(b) depicts the distribution of delta safe index, i.e., the LHS of (9). As the noise level increases, the safety index changes more violently, but the max value maintains as zero indicating that the constraints are not violated. It shows the great robustness of the proposed algorithms.

Moreover, regarding the real-world applications, we compare the calculation time and power consumption between the proposed methods and two commonly used autonomous driving controller. One baseline is the nonlinear model predictive controller (NMPC) [34]. Another baseline is the hierarchical planning and control (shorted as hierarchical), where planning algorithm is based on the random-exploring search (RRT) [66], and the controllers are decomposed to longitudinal controller

TABLE III
NOISE LEVEL AND THE CORRESPONDING STANDARD DEVIATION

Noise level	0	1	2	3	4
δ_p [m]	0	0.017	0.033	0.051	0.068
δ_ϕ [°]	0	0.017	0.033	0.051	0.068
p_x^j, p_y^j [m]	0	0.05	0.10	0.15	0.20
v^j [m/s]	0	0.05	0.10	0.15	0.20
η^j [°]	0	1.4	2.8	4.2	5.6

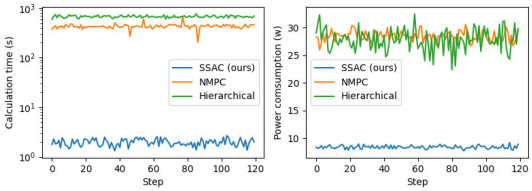


Fig. 14. Calculation time and power consumption comparison. We did not add previous learning-based controllers since their inference time and power consumption are similar.

TABLE IV
PERFORMANCE COMPARISON BETWEEN THREE AUTONOMOUS DRIVING CONTROLLERS

	SSAC	NMPC	Hireachical
Collisions	0	0	0
Normalized average tracking error	1.00	1.120	1.002
Average intersection pass time	12.04	12.32	14.65

using linear quadratic regulator (LQR) and lateral controller using PID. The inference/calculation time comparisons are shown in Fig. 14. The results have shown that the proposed SSAC outperforms baselines a lot in terms of calculation time and power consumption. Notably, the noted power of our controller is 32 W. The baselines are achieving the rated power, which means if the performance might be worse in the real-world where extra tasks need to be done in parallel. Regarding the performance comparison between three controllers, the results are shown in Table IV. All three controllers achieve zero violation in 100 times of intersection pass. The tracking performance is calculated by calculating the distance between trajectory points and their projections on the fixed references, and then normalized by the tracking error of SSAC. Results have shown that the tracking performance of SSAC is the best but the gap is not large. We also calculate the average time to pass the intersection. The proposed algorithm also achieves the fastest pass time. Combining with the calculation time saving, SSAC outperforms two nonlinear controllers and other learning-based controllers in terms of autonomous driving control.

VI. CONCLUSION

In this article, we proposed the SSAC, which can learn a zero-constraint violation policy using model-free constrained RL. The major contribution of this article is achieving zero-violation safe control policy using RL. We leverage the safety-oriented energy function as the model-free *prior* safety measurement, which is designed to be lower for safe state. Then the RL algorithm is designed to prevent the energy from increasing too fast, then the agent do not need to experience the danger for learning the safe policy. Compared to other safety-critical RL based on CMDP or risk-sensitive measure, the zero-violation policy better aligns with general real-world safety requirements. Furthermore, we proved the convergence of the proposed algorithm theoretically.

Limitations include violations during training and energy function parameters learning. The first affects the safety during training, and the second affects the actual size of the safe set. We will consider change the initial conditions and use neural energy function to further handle these limits.

APPENDIX

Step 3 (Local Saddle Point (θ^*, ξ^*)):
As we provided in the previous, $L(\theta^*, \xi^*) \leq L(\theta, \xi^*)$, so we need to prove here $L(\theta^*, \xi^*) \geq L(\theta^*, \xi)$. To complete the proof we need that

$$J_{w_c}^{\pi_{\theta^*}}(s_t, a_t) \leq d \text{ and } \lambda^*(s_t)\left(J_{w_c}^{\pi_{\theta^*}}(s_t, a_t) - d\right) = 0 \tag{41}$$

for all s in the discounted distribution. Recall that λ^* is a local maximum point, we have

$$\nabla_\xi L\left(\theta^*, \xi^*\right) = 0. \tag{42}$$

Assume there exists s_t so that $J_{w_c}^{\pi_{\theta^*}}(s_t, \pi^*(s_t)) > 0$. Then for λ^* we have

$$\nabla_\xi L\left(\theta^*, \xi^*\right) = d_\gamma^{\pi_\theta^*}(s_t) J_{w_c}^{\pi_{\theta^*}}\left(s_t, \pi^*(s_t)\right) \nabla_\xi \lambda_\xi(s_t) \neq 0. \tag{43}$$

The second part only requires that $\lambda^*(s_t) = 0$ when $J_{w_c}^{\pi_{\theta^*}}(s_t, \pi^*(s_t)) < 0$. Similarly, we assume that there exists s_t and ξ^* where $\lambda_{\xi^*}(s_t) > 0$ and $Q^{\pi_{\theta^*}}(s_t, \pi^*(s_t)) < d$. there must exists a ξ_0 subject to

$$\xi_0 = \xi^* + \eta_0 d_\gamma^{\pi_{\theta^*}}(s_t)\left(J_{w_c}^{\pi_{\theta^*}}\left(s_t, \pi^*(s_t)\right) - d\right) \nabla_\xi \lambda_\xi(s_t) \tag{44}$$

for any $\eta \in (0, \eta_0]$ where $\eta_0 \geq 0$. It contradicts the statement of the local maximum ξ^*. Then we get

$$\begin{aligned} L\left(\theta^*, \xi^*\right) &= J_r\left(\theta^*\right) + \mathbb{E}_{s\sim d_\gamma}\left\{\lambda_{\xi^*}(s) \mathbb{E}_{a\sim\pi}\left\{J_{w_c}^{\pi_{\theta^*}}(s, a)\right\}\right\} \\ &= J_r\left(\theta^*\right) \\ &\geq J_r\left(\theta^*\right) + \mathbb{E}_{s\sim d_\gamma}\left\{\lambda_{\xi^*}(s) \mathbb{E}_{a\sim\pi}\left\{J_{w_c}^{\pi_{\theta^*}}(s, a)\right\}\right\} \\ &= L\left(\theta^*, \xi\right). \end{aligned} \tag{45}$$

So (θ^*, ξ^*) is a locally saddle point.

A. Experimental Details

1) Implementation Details: Implementation of FAC is based on the Parallel Asynchronous Buffer-Actor-Learner (PABAL) architecture proposed by [67]. All experiments are implemented on 2.5 GHz Intel Xeon Gold 6248 processors with 12 parallel actors, including four workers to sample, four buffers to store data, and four learners to compute gradients.[2] The simulation environments run on a personal computer with 3.0 GHz AMD Ryzen 4900HS processors with 16 GB memory.

[2]The original implementation of PABAL can be found at https://github.com/idthanm/mpg. The baseline implementation is modified from https://github.com/openai/safety-starter-agents and https://github.com/ikostrikov/pytorch-a2c-ppo-acktr-gail

TABLE V
DETAILED HYPERPARAMETERS

Algorithm	Value
SSAC	
Optimizer	Adam ($\beta_1 = 0.9, \beta_2 = 0.999$)
Approximation function	Multi-layer Perceptron
Number of hidden layers	2
Number of hidden units	256
Nonlinearity of hidden layer	ELU
Nonlinearity of output layer	linear
Actor learning rate	Linear annealing $3e{-}5 \rightarrow 1e{-}6$
Critic learning rate	Linear annealing $8e{-}5 \rightarrow 1e{-}6$
Learning rate of multiplier net	Linear annealing $5e{-}5 \rightarrow 5e{-}6$
Learning rate of α	Linear annealing $5e{-}5 \rightarrow 1e{-}6$
Reward discount factor (γ)	0.99
Policy update interval (m_π)	3
Multiplier ascent interval (m_λ)	12
Target smoothing coefficient (τ)	0.005
Max episode length (N)	
Safe exploration tasks	1000
Autonomous driving task	200
Expected entropy ($\overline{\mathcal{H}}$)	$\overline{\mathcal{H}} = -$Action Dimentions
Replay buffer size	5×10^5
Replay batch size	256
Safety index hyperparameters	
(η, n, k, σ)	$(0,\ 2,\ 1,\ 0.04)$
CPO, TRPO-Lagrangian	
Max KL divergence	0.1
Damping coefficient	0.1
Backtrack coefficient	0.8
Backtrack iterations	10
Iteration for training values	80
Init λ	$0.268(softplus(0))$
GAE parameters	0.95
Batch size	2048
Max conjugate gradient iters	10
PPO-Lagrangian	
Clip ratio	0.2
KL margin	1.2
Mini Bactch Size	64

2) Hyperparameters: The hyperparameters of FAC and baseline algorithms are listed in Table V. We explain some intuitions of how to select the hyperparameters here. For those parameters already existing in [55], like the network structure, discounted factor, we used the same, or similar parameters which has been proved to be stable. Other important parameters include the learning rate of five different learning objectives. We follow Assumption 2 to design the initial learning rate. The learning rate annealing is helpful for stable training.

REFERENCES

[1] D. Silver et al., "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, Jan. 2016.

[2] V. Mnih et al., "Playing Atari with deep reinforcement learning," 2013, *arXiv:1312.5602*.

[3] O. Vinyals et al., "Grandmaster level in StarCraft II using multi-agent reinforcement learning," *Nature*, vol. 575, no. 7782, pp. 350–354, Nov. 2019.

[4] T. Haarnoja et al., "Soft actor–critic algorithms and applications," 2018, *arXiv:1812.05905*.

[5] L. Zhang, R. Zhang, T. Wu, R. Weng, M. Han, and Y. Zhao, "Safe reinforcement learning with stability guarantee for motion planning of autonomous vehicles," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 12, pp. 5435–5444, Dec. 2021.

[6] J. Chen, S. E. Li, and M. Tomizuka, "Interpretable end-to-end urban autonomous driving with latent deep reinforcement learning," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 6, pp. 5068–5078, Jun. 2022.

[7] Y. Zhang, B. Gao, L. Guo, H. Guo, and H. Chen, "Adaptive decision-making for automated vehicles under roundabout scenarios using optimization embedded reinforcement learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 12, pp. 5526–5538, Dec. 2021.

[8] F. Richter, R. K. Orosco, and M. C. Yip, "Open-sourced reinforcement learning environments for surgical robotics," 2019, *arXiv:1903.02090*.

[9] F. Berkenkamp, M. Turchetta, A. P. Schoellig, and A. Krause, "Safe model-based reinforcement learning with stability guarantees," 2017, *arXiv:1705.08551*.

[10] A. Traue, G. Book, W. Kirchgässner, and O. Wallscheid, "Toward a reinforcement learning environment toolbox for intelligent electric motor control," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 3, pp. 919–928, Mar. 2022.

[11] X. Xie, C. Wei, Z. Gu, and K. Shi, "Relaxed resilient fuzzy stabilization of discrete-time Takagi–Sugeno systems via a higher order time-variant balanced matrix method," *IEEE Trans. Fuzzy Syst.*, vol. 30, no. 11, pp. 5044–5050, Nov. 2022.

[12] Z. Gu, D. Yue, J. H. Park, and X. Xie, "Memory-event-triggered fault detection of networked IT2 T–S fuzzy systems," *IEEE Trans. Cybern.*, vol. 53, no. 2, pp. 743–752, Feb. 2023.

[13] E. Altman, *Constrained Markov Decision Processes*, vol. 7. Boca Raton, FL, USA: CRC Press, 1999.

[14] P. Geibel and F. Wysotzki, "Risk-sensitive reinforcement learning applied to control under constraints," *J. Artif. Intell. Res.*, vol. 24, pp. 81–108, Jul. 2005.

[15] J. Achiam, D. Held, A. Tamar, and P. Abbeel, "Constrained policy optimization," in *Proc. Int. Conf. Mach. Learn.*, Sydney, NSW, Australia, 2017, pp. 22–31.

[16] S. Fujimoto, E. Conti, M. Ghavamzadeh, and J. Pineau, "Benchmarking batch deep reinforcement learning algorithms," 2019, *arXiv:1910.01708*.

[17] T.-Y. Yang, J. Rosca, K. Narasimhan, and P. J. Ramadge, "Projection-based constrained policy optimization," 2020, *arXiv:2010.03152*.

[18] C. Tessler, D. J. Mankowitz, and S. Mannor, "Reward constrained policy optimization," 2018, *arXiv:1805.11074*.

[19] Q. Yang, T. D. Simão, S. H. Tindemans, and M. T. Spaan, "WCSAC: Worst-case soft actor critic for safety-constrained reinforcement learning," in *Proc. 31st AAAI Conf. Artif. Intell.*, 2021, pp. 10639–10646.

[20] Y. Chow, M. Ghavamzadeh, L. Janson, and M. Pavone, "Risk-constrained reinforcement learning with percentile risk criteria," *J. Mach. Learn. Res.*, vol. 18, no. 1, pp. 6070–6120, 2017.

[21] A. Sootla, A. Cowen-Rivers, J. Wang, and H. B. Ammar, "Enhancing safe exploration using safety state augmentation," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 35, Dec. 2022, pp. 34464–34477.

[22] Y. Zhang, Q. Vuong, and K. Ross, "First order constrained optimization in policy space," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, Eds., 2020, pp. 15338–15349.

[23] J. Duan, Z. Liu, S. Eben Li, Q. Sun, Z. Jia, and B. Cheng, "Adaptive dynamic programming for nonaffine nonlinear optimal control problem with state constraints," 2019, *arXiv:1911.11397*.

[24] J. Chen, W. Zhan, and M. Tomizuka, "Constrained iterative LQR for on-road autonomous driving motion planning," in *Proc. IEEE 20th Int. Conf. Intell. Transp. Syst. (ITSC)*, Oct. 2017, pp. 1–7.

[25] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *Proc. 18th Eur. Control Conf. (ECC)*, Bochum, Germany, Jun. 2019, pp. 3420–3431.

[26] R. Cheng, G. Orosz, R. M. Murray, and J. W. Burdick, "End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks," in *Proc. AAAI Conf. Artif. Intell.*, Jul. 2019, vol. 33, no. 1, pp. 3387–3395.

[27] H. Ma et al., "Model-based constrained reinforcement learning using generalized control barrier function," 2021, *arXiv:2103.01556*.

[28] Z. Qin, K. Zhang, Y. Chen, J. Chen, and C. Fan, "Learning safe multi-agent control with decentralized neural barrier certificates," 2021, *arXiv:2101.05436*.

[29] C. Liu and M. Tomizuka, "Control in a safe set: Addressing safety in human–robot interactions," in *Proc. Dyn. Syst. Control Conf.*, Oct. 2014, pp. 1–10.

[30] W. Zhao, T. He, and C. Liu, "Model-free safe control for zero-violation reinforcement learning," in *Proc. 5th Annu. Conf. Robot Learn.* (Proceedings of Machine Learning Research), A. Faust, D. Hsu, and G. Neumann, Eds. 2022, pp. 784–793. [Online]. Available: https://proceedings.mlr.press/v164/zhao22a/zhao22a.pdf

[31] T. Wei and C. Liu, "Safe control algorithms using energy functions: A unified framework, benchmark, and new directions," in *Proc. IEEE 58th Conf. Decis. Control (CDC)*, Dec. 2019, pp. 238–243.

[32] J. F. Fisac, A. K. Akametalu, M. N. Zeilinger, S. Kaynama, J. Gillula, and C. J. Tomlin, "A general safety framework for learning-based control in uncertain robotic systems," *IEEE Trans. Autom. Control*, vol. 64, no. 7, pp. 2737–2752, Jul. 2019.

[33] G. Dalal, K. Dvijotham, M. Vecerik, T. Hester, C. Paduraru, and Y. Tassa, "Safe exploration in continuous action spaces," 2018, *arXiv:1801.08757*.

[34] L. Grüne, J. Pannek, L. Grüne, and J. Pannek, *Nonlinear Model Predictive Control*. Cham, Switzerland: Springer, 2017.

[35] H. Ma, Y. Guan, S. Eben Li, X. Zhang, S. Zheng, and J. Chen, "Feasible actor–critic: Constrained reinforcement learning for ensuring statewise safety," 2021, *arXiv:2105.10682*.

[36] E. Uchibe and K. Doya, "Constrained reinforcement learning from intrinsic and extrinsic rewards," in *Proc. IEEE 6th Int. Conf. Develop. Learn.*, Lugano, Switzerland, Jul. 2007, pp. 163–168.

[37] A. Stooke, J. Achiam, and P. Abbeel, "Responsive safety in reinforcement learning by PID Lagrangian methods," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 9133–9143.

[38] B. Peng, Y. Mu, J. Duan, Y. Guan, S. Eben Li, and J. Chen, "Separated proportional-integral Lagrangian for chance constrained reinforcement learning," 2021, *arXiv:2102.08539*.

[39] S. Bhatnagar, R. S. Sutton, M. Ghavamzadeh, and M. Lee, "Natural actor–critic algorithms," *Automatica*, vol. 45, no. 11, pp. 2471–2482, Nov. 2009.

[40] D. Ding, K. Zhang, T. Basar, and M. R. Jovanovic, "Natural policy gradient primal-dual method for constrained Markov decision processes," in *Proc. NeurIPS*, 2020, pp. 8378–8390.

[41] D. Ding, X. Wei, Z. Yang, Z. Wang, and M. Jovanovic, "Provably efficient safe exploration via primal-dual policy optimization," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2021, pp. 3304–3312.

[42] Y. Yang, B. Kiumarsi, H. Modares, and C. Xu, "Model-free λ-policy iteration for discrete-time linear quadratic regulation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 2, pp. 635–649, Feb. 2023.

[43] A. Sootla et al., "Saute RL: Almost surely safe reinforcement learning using state augmentation," in *Proc. 39th Int. Conf. Mach. Learn.*, Jun. 2022.

[44] J. Duan, Z. Liu, S. E. Li, Q. Sun, Z. Jia, and B. Cheng, "Adaptive dynamic programming for nonaffine nonlinear optimal control problem with state constraints," *Neurocomputing*, vol. 484, pp. 128–141, May 2022.

[45] Y. Yang, D.-W. Ding, H. Xiong, Y. Yin, and D. C. Wunsch, "Online barrier-actor–critic learning for H_∞ control with full-state constraints and input saturation," *J. Franklin Inst.*, vol. 357, no. 6, pp. 3316–3344, Apr. 2020.

[46] Y. Yang, K. G. Vamvoudakis, H. Modares, Y. Yin, and D. C. Wunsch, "Safe intermittent reinforcement learning with static and dynamic event generators," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 12, pp. 5441–5455, Dec. 2020.

[47] Y. Yang, H. Modares, K. G. Vamvoudakis, W. He, C.-Z. Xu, and D. C. Wunsch, "Hamiltonian-driven adaptive dynamic programming with approximation errors," *IEEE Trans. Cybern.*, vol. 52, no. 12, pp. 13762–13773, Dec. 2022.

[48] Y. Yang, Y. Pan, C.-Z. Xu, and D. C. Wunsch, "Hamiltonian-driven adaptive dynamic programming with efficient experience replay," *IEEE Trans. Neural Netw. Learn. Syst.*, pp. 1–13, 2022.

[49] T.-H. Pham, G. De Magistris, and R. Tachibana, "OptLayer–practical constrained optimization for deep reinforcement learning in the real world," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 6236–6243.

[50] J. Ferlez, M. Elnaggar, Y. Shoukry, and C. Fleming, "ShieldNN: A provably safe NN filter for unsafe NN controllers," 2020, *arXiv:2006.09564*.

[51] A. Agrawal and K. Sreenath, "Discrete control barrier functions for safety-critical control of discrete systems with application to bipedal robot navigation," in *Proc. Robot., Sci. Syst.*, Cambridge, MA, USA, Jul. 2017, doi: 10.15607/RSS.2017.XIII.073.

[52] Y. Zhang et al., "Barrier Lyapunov function-based safe reinforcement learning for autonomous vehicles with optimized backstepping," *IEEE Trans. Neural Netw. Learn. Syst.*, 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9827972

[53] T. Mannucci, E.-J. van Kampen, C. de Visser, and Q. Chu, "Safe exploration algorithms for reinforcement learning controllers," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 4, pp. 1069–1081, Apr. 2018.

[54] Z. Zhou, O. S. Oguz, M. Leibold, and M. Buss, "Learning a low-dimensional representation of a safe region for safe reinforcement learning on dynamical systems," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 5, pp. 2513–2527, May 2023.

[55] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor–critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proc. Int. Conf. Mach. Learn.*, Stockholm, Sweden, 2018, pp. 1861–1870.

[56] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor–critic methods," in *Proc. Int. Conf. Mach. Learn.*, Stockholm, Sweden, 2018, pp. 1587–1596.

[57] V. S. Borkar, *Stochastic Approximation: A Dynamical Systems Viewpoint*, vol. 48. Cham, Switzerland: Springer, 2009.

[58] S. Bhatnagar and K. Lakshmanan, "An online actor–critic algorithm with function approximation for constrained Markov decision processes," *J. Optim. Theory Appl.*, vol. 153, no. 3, pp. 688–708, 2012.

[59] L. Prashanth and M. Ghavamzadeh, "Actor–critic algorithms for risk-sensitive MDPs," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 26, C. J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds. Curran Associates, 2013, pp. 1–9. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2013/file/eb163727917cbba1eea208541a643e74-Paper.pdf

[60] D. P. Bertsekas, "Nonlinear programming," *J. Oper. Res. Soc.*, vol. 48, no. 3, p. 334, 1997.

[61] G. Brockman et al., "OpenAI gym," 2016, *arXiv:1606.01540*.

[62] E. Todorov, T. Erez, and Y. Tassa, "MuJoCo: A physics engine for model-based control," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2012, pp. 5026–5033.

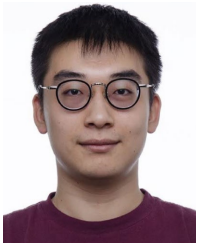
[63] L. Zhang et al., "Penalized proximal policy optimization for safe reinforcement learning," 2022, *arXiv:2205.11814*.

[64] Y. Guan et al., "Integrated decision and control: Towards interpretable and computationally efficient driving intelligence," 2021, *arXiv:2103.10290*.

[65] Q. Ge, S. Eben Li, Q. Sun, and S. Zheng, "Numerically stable dynamic bicycle model for discrete-time control," 2020, *arXiv:2011.09612*.

[66] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *Int. J. Robot. Res.*, vol. 20, no. 5, pp. 378–400, May 2001.

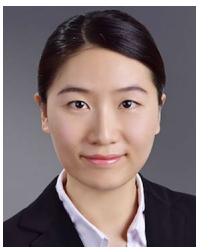
[67] Y. Guan, J. Duan, S. Eben Li, J. Li, J. Chen, and B. Cheng, "Mixed policy gradient," 2021, *arXiv:2102.11513*.



Haitong Ma received the B.S. and M.S. degrees from the School of Vehicle and Mobility, Tsinghua University, Beijing, China, in 2019 and 2022, respectively. He is currently pursuing the Ph.D. degree with the Harvard School of Engineering and Applied Sciences, Cambridge, MA, USA.

His current research interest lies on the intersection between control theory and machine learning, with special interests on sim-to-real transfer, and safe control via reinforcement learning applied on real robots.

Mr. Ma has received the L4DC Best Paper Finalists, the ITSC Best Student Paper Award, and the ICUS Best Paper Award.



Changliu Liu (Member, IEEE) received the B.S. degree in mechanical engineering and economics from Tsinghua University, Beijing, China, in 2012, and the M.S. degree in mechanical engineering, the M.A. degree in mathematics, and the Ph.D. degree in mechanical engineering from the University of California at Berkeley, Berkeley, CA, USA, in 2014, 2016, and 2017, respectively.

She is an Assistant Professor with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA. Her research focuses on designing and verifying intelligent systems with applications to manufacturing and transportation.

Dr. Liu was a recipient of the NSF Career Award, the Amazon Research Award, and the Ford URP Award.

Shengbo Eben Li (Senior Member, IEEE) received the M.S. and Ph.D. degrees from Tsinghua University, Beijing, China, in 2006 and 2009, respectively.

Before joining Tsinghua University, he has worked at Stanford University, Stanford, CA, USA; the University of Michigan, Ann Arbor, MI, USA; and the University of California at Berkeley, Berkeley, CA, USA. He is the author of over 190 peer-reviewed journal/conference papers and co-inventor of over 40 patents. His active research interests include intelligent vehicles and driver assistance, deep reinforcement learning, and optimal control and estimation.

Dr. Li has received over 20 prestigious awards, including the Youth Science and Technology Award of Ministry of Education (annually ten receivers in China), the Natural Science Award of Chinese Association of Automation (First level), the National Award for Progress in Science and Technology of China, and the best (student) paper awards of IET ITS, IEEE INTELLIGENT TRANSPORTATION SYSTEMS (ITS), IEEE ICUS, and CVCI. He also serves as the Board of Governor of IEEE ITS Society, a Senior AE of IEEE OPEN JOURNAL OF INTELLIGENT TRANSPORTATION SYSTEMS (OJ ITS), and an AEs of *IEEE Intelligent Transportation Systems Magazine (ITSM)*, IEEE TITS, IEEE TRANSACTIONS ON INTELLIGENT VEHICLES (TIV), and IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS (TNNLS).

Sifa Zheng (Member, IEEE) received the B.E. and Ph.D. degrees from Tsinghua University, Beijing, China, in 1993 and 1997, respectively.

He is currently a Professor with the School of Vehicle and Mobility and the State Key Laboratory of Automotive Safety and Energy, Tsinghua University. He is also the Deputy Director of the Suzhou Automotive Research Institute, Tsinghua University. His current research interests include autonomous driving and vehicle dynamics and control.

Wenchao Sun received the B.E. degree from Tsinghua University, Beijing, China, in 2021, where he is currently pursuing the Ph.D. degree in mechanical engineering with the School of Vehicle and Mobility.

His research interests include object detection and online map construction in autonomous driving.

Jianyu Chen received the bachelor's degree from Tsinghua University, Beijing, China, in 2015, and the Ph.D. degree from the University of California at Berkeley, Berkeley, CA, USA, in 2020.

Since 2020, he has been an Assistant Professor with the Institute for Interdisciplinary Information Sciences (IIIS), Tsinghua University. Prior to that, he was working with Prof. Masayoshi Tomizuka at the University of California at Berkeley. He is working in the cross fields of robotics, reinforcement learning, and control and autonomous driving. His research goal is to build advanced robotic systems with high performance and high intelligence.